UNIVERSIDAD VERACRUZANA

FACULTAD DE FÍSICA E INTELIGENCIA ARTIFICIAL

SisNeρ: Un Robot Parcialmente Mecatrónico

TESIS

Que para obtener el grado de:

Maestro en Inteligencia Artificial

Presenta:

Roberto Cruz Estrada

Dirigida por:

Dr. José Negrete Martínez

Xalapa de Enríquez, Ver. Marzo 2002

Maestría en Inteligencia Artificial

Universidad Veracruzana

Departamento de Física e Inteligencia Artificial

Tesis

SisNeρ: Un Robot Parcialmente Mecatrónico

Presenta:

Roberto Cruz Estrada

Dirigida por:

Dr. José Negrete Martínez

Asesor:

M. en C. V. Angélica García Vega

Asesor:

Dr. Ronald Vogel

Revisor:

Dr. Fernando M. Montes González

AGRADECIMIENTOS

Al Dr. José Negrete Martínez, con la deferencia que me merece, por la confianza que tuvo en mi al invitarme a participar en este proyecto.

A la Mtra. V. Angélica García Vega, por toda su ayuda y por alentarme en la realización de este trabajo.

Al Dr. Ronald Vogel, por su infinita paciencia y valiosa ayuda en el diseño de los circuitos y consecución de componentes, así como por los inolvidables momentos compartidos con él y Feather.

AL Dr. Fernando M. Montes González, por sus puntuales observaciones y consejos para enriquecer la calidad de este trabajo.

A mis compañeros de generación Ana Silvia, Marisol, Sonia, Alberto, Alejandro, Gildardo, José Luis, Juan Manuel, Odin y Pedro, por su amistad y las muchas gratas experiencias compartidas.

A los demás compañeros y a todo el personal de la M.I.A., por su apoyo y amistad, así como por el inmejorable ambiente de convivencia y trabajo.

A mis padres Gloria y Pedro, a mis hermanas Coco y Vero, a mis hermanos Eloy, Pedro y Jorge y a toda la familia, disculpen los buenos momentos que dejé de estar con ustedes, espero que este logro sea motivo de orgullo también para ustedes.

Finalmente a Ana por instarme a seguir adelante en este proyecto y a Fer, por ser ustedes para mi, motivo especial de superación.

ÍNDICE

ÍNDICE DE TABLAS Y FIGURAS

TABLAS

FIGURAS

INTRODUCCIÓN

¿ Qué es un robot antropomimético totalmente mecatrónico ?

Cuando se habla de un robot, las imágenes que nos vienen a la mente son las de un artefacto electromecánico con rasgos antropomórficos y antropodinámicos. Sin embargo ninguna atención se presta a la antropomorfología y antropodinámica de su sistema de control. Estas características, como no se ven, las ponemos de alguna manera en manos de variados programas de computadora; la antropoanalogía es solo superficial, no se persigue hasta ahí. Esta tesis es parte de un proyecto, en el que precisamente se quiere llevar la antropoanalogía mecatrónica así de lejos, para crear un robot antropomimético completamente mecatrónico.

En el campo de la inteligencia artificial (IA), la creación de robots es una tarea muy relevante. Uno de sus objetivos (quizá el más ambicioso) es el de crear, a futuro, robots que tenga un desempeño similar al del ser humano, aunque para algunos pueda parecer algo poco viable [Penrose 95]. Posiblemente lograr emular al ser humano tanto en sus funciones básicas como en su comportamiento, implicaría conocer algo esencial: cómo reproducir físicamente la dinámica de su sistema nervioso central (SNC), el cual ha sido un enigma para sus estudiosos desde tiempos inmemoriables. Actualmente neurólogos, etólogos, psicólogos, computólogos y otros científicos, estudian la dinámica del SNC, en otros vertebrados e invertebrados, para desentrañar la esencia de su funcionamiento; sin embargo, el camino esta cerrado a menos que se marche por la vía de la síntesis física [Steels 95], esto es, emplear los conocimientos adquiridos en la construcción de sistemas artificiales.

Varias investigaciones, han propuesto que el SNC de los vertebrados superiores, puede considerarse una arquitectura de módulos organizados en capas [Leise 90]. Cada módulo actúa como un circuito complejo neuronal que frecuentemente está activo “paralelamente”, junto con otros módulos, lo que los hace competir entre sí por una salida motora. Recientemente Prescott [Prescott 99] ha planteado la posibilidad de que esta competencia se “resuelva” por un mecanismo similar al de subsumción entre módulos, por ello establece que hay analogía directa entre la organización modular del SNC y la arquitectura de subsumción (ASu) de Brooks [Brooks 86].

Brooks ha planteado la necesidad de llevar la teoría de IA a artefactos “realmente inteligentes” que se desempeñen en un mundo real, sensando variables del ambiente con actuación sensorimotora: “la simulación es muy útil pero no suficiente” [Brooks 91]. Esto es algo de lo que se pretende en esta tesis, al aplicar una arquitectura de control, que tiene cierta analogía con la organización modular del SNC, en un robot real.

I1. Antecedentes

Esta tesis forma parte del proyecto de J. Negrete, en el cual se pretende construir un robot completamente mecatrónico, que sea controlado por un mecanismo físico que emule, en forma simple, el funcionamiento modular del sistema nervioso central (SNCM). Como primera parte de dicho proyecto, Negrete [Negrete-Mtz 00] propuso una arquitectura modular, en la que sugiere

que existan módulos que cooperen entre ellos, en la consecución de un objetivo común. Esta arquitectura de módulos cooperativos (AMC), fue aplicada en la simulación computacional de un SNC que como "sociedad cooperativa", actúa en un robot simulado por una animación gráfica bidimensional, programada en lenguaje C. A esta primera parte del proyecto le llamó SNCM.1.

La segunda parte del proyecto, el SNCM.2, es precisamente el trabajo de esta tesis, en la que tratamos la implementación de la AMC en un robot real, el cual tendrá como propósito específico, localizar un emisor infrarrojo (faro IR), apuntar hacia él y alcanzarlo. Requeriremos que los módulos de la AMC estén al servicio de un objetivo común, que cada uno de ellos pueda disminuir la distancia del robot al objetivo y que los módulos puedan auto-descalificarse cuando, en un momento conductual determinado, no puedan contribuir más a la consecución del objetivo.

El robot construido como parte del trabajo de la tesis, lo llamamos SisNeρ (acrónimo de sistema nervioso robótico)[1], consiste en un brazo único montado sobre un carro con ruedas. El robot es controlado desde una computadora personal y la arquitectura de control, AMC, fue implementada en lenguaje C.

I2. Objetivos de la tesis.

Construir un robot físico real (electromecánico) y los accesorios necesarios para que cumpla con un propósito especifico: localizar, apuntar y alcanzar, con el brazo del robot, un faro infrarrojo. El robot servirá de plataforma de trabajo para la arquitectura de control a emplear y de otras que en un futuro se quieran aplicar.

Implementar la arquitectura AMC en un robot electromecánico y experimentar con las características modulares del SNC, que subyace en la estructura, también modular de dicha arquitectura.

Enfrentar la problemática que resulta de llevar una simulación robótica gráfica (bidimensional) a un robot real, incluyendo las limitaciones, en primer lugar físicas del robot y en segundo lugar, de la implementación de la AMC, en los diferentes módulos articulares y de movimiento del robot.

I3. Contenido de la tesis.

Presentaremos el trabajo de la siguiente forma: esta introducción que describe los motivos que dieron origen a la tesis, seguida del desarrollo de la misma en seis capítulos, de los cuales hacemos un breve resumen de cada uno de ellos a continuación y al final los anexos con referencias e información general, que incluyen una descripción del robot Herbert de Brooks y Conell así como el código completo del programa de control del robot.

Capítulo 1. El SNC y su estructura modular. Abordaremos algunos antecedentes biológico – robóticos de la estructura modular del SNC, consideraremos la ASu de Brooks, que ha sido

[1] "Cierto" parecido físico del robot con un cisne animal, nos motivó también a denominarlo con el nombre de SisNeρ.

inspiración de un sin número de trabajos robóticos en el mundo y algunos mecanismos de selección de acción para arquitecturas de capas de control.

Capítulo 2. La arquitectura del SNCM.2. Presentaremos la arquitectura AMC, la estructura modular del robot SisNeρ, sus módulos componentes, el principio de detección del faro IR que es muy importante en el desempeño general del robot y el algoritmo del "bandido" utilizado en los módulos de la AMC.

Capítulo 3. El SisNeρ. Detallaremos la implementación mecatrónica del robot SisNeρ, incluyendo la base móvil, el brazo mecánico, los servos, los elementos y montaje del sensor IR, del faro IR y los circuitos electrónicos utilizados. En este capítulo también presentaremos la implementación computacional de la arquitectura de control del robot en lenguaje C, incluyendo las funciones que operan los diferentes módulos y registros así como las funciones que manejan los elementos de hardware. Por convención, el `texto con tipografía Courier` que aparezca en la tesis corresponde a instrucciones en lenguaje C ó pseudo-código.

Capítulo 4. Experimentos con el SisNeρ. Mostraremos las pruebas de caracterización del sensor IR y diferentes conductas de desempeño del robot incluyendo su actuación cuando se usa el algoritmo del "bandido invertido".

Capítulo 5. Análisis de nuestros resultados y de trabajos afines. Haremos un análisis de los resultados obtenidos y una comparación cualitativa, con los resultados reportados de un trabajo similar de robot físico.

Capítulo 6. Conclusiones y trabajo futuro. Puntualizaremos las principales conclusiones obtenidas de la tesis y algunos trabajos que vislumbramos para desarrollar a futuro.

CAPÍTULO 1. EL SNC Y SU ESTRUCTURA MODULAR

Con el devenir del tiempo, el ser humano y otros vertebrados han evolucionado para adaptarse a los diferentes ambientes en los que se desenvuelven; sin embargo, lo que permanece invariable, en buena parte, es el "plan de construcción" de la estructura de los organismos y los puntos fijados para la operación de sus partes. Es decir: permanecen el espíritu de la forma y el espíritu de la función [Damasio 00].

Un componente fundamental de los seres vivos es su SNC, el cual también ha evolucionado – el órgano más evolucionado del universo, según Brailowsky [Brailowsky 95] – permitiendo la supervivencia de especies, aún cuando en ocasiones el SNC sea muy sencillo, como en el caso de gusanos ó el más complejo, el caso de los seres humanos.

A continuación trataremos brevemente la organización modular del SNC, y sus similitudes con la ASu de Brooks [Brooks 85], la cual en su momento, fue propuesta como una lejana aproximación a la organización biológica del SNC.

1.1 Arquitectura de capas de control: perspectiva biológica.

Desde el punto de vista funcional, el SNC de los vertebrados está formado básicamente por la médula espinal y el cerebro. La médula espinal, además de un centro nervioso, es una especie de "conducto" a través del cual los nervios y algunas substancias conducen información sensorimotora. Una buena parte de la información sensorial que circula a través de la médula espinal se dirige al cerebro (entrando por los nervios aferentes), donde entre otras muchas funciones, se generan "patrones motores complejos", que conducidos de regreso por la médula (a través de los nervios eferentes), van a dar a los músculos.

En cuanto a la organización del SNC, todo indica que éste tiene una estructura modular. Cada unidad anatómica (física) básica o módulo, se piensa que puede actuar tanto en paralelo como en serie [Leise 90].

Hay evidencias de que existen también submódulos que forman bloques neuronales que se conectan como unidades cohesivas "desarrolladoras" que proveen algún tipo específico de excitación o inhibición [Levine 85].

En el SNC se tienen identificados los siguientes tipos de módulos [Mountcastle 79]:

1. De procesamiento de entrada – salida con conexión a un número limitado de regiones en el sistema nervioso.

2. Módulos en los que se pueden mapear diferentes nervios sensoriales (de modalidad).

3. Módulos que mantienen conexiones específicas, a manera de ordenamiento entre los módulos del cerebro.

4. Módulos que pueden ser identificados como mapas únicos de parámetros.

5. Módulos que permiten procesamiento selectivo de señales, conduciendo parámetros específicos a destinos de salida particulares.

Los circuitos fisiológicos dentro de los varios módulos mencionados, realizan una amplia variedad de tareas, aunque en términos cualitativos son los módulos de procesamiento paralelo el fundamento de las conductas complejas en los animales.

La construcción modular del tejido del cerebro es, no sólo una característica del tejido nervioso de los vertebrados, también lo es de muchos sistemas nerviosos de invertebrados. Datos fisiológicos avalan la idea de que los módulos neurales no son solamente entidades anatómicas, sino que también son circuitos adaptativos locales; huelga decir entonces que tales módulos tienen un rol funcional muy importante en el desempeño de los vertebrados.

1.1.1. Propuesta jacksoniana.

En el año de 1884 el neurólogo John Hughlings Jackson propuso un "esquema de capas" del sistema nervioso (SN), en donde el cerebro es considerado "una implementación de niveles múltiples de competición jerárquica sensorimotora" [Prescott 99]. El punto de vista de Jackson se inspiró en la revolución darwiniana [Darwin 1859] del siglo diecinueve, la que ya se basaba, no en las divisiones usuales morfológicas del SN, sino en aspectos funcionales.

Jackson dividió el SN en centros bajo, medio y alto y propuso que esta secuencia representaba una progresión desde el "mas organizado" (mas fijo) al "menos organizado" (mas modificable), desde el "mas automático" al "menos automático" y del mas "perfectamente reflejo" al menos "perfectamente reflejo". Esta secuencia puede verse como una progresión de adaptabilidad modular.

La evolución del SN era considerada por Jackson como un proceso incremental, en el cual centros bajos son retenidos intactos pero son suprimidos por nuevos centros altos. Dentro de los diferentes centros, él consideraba que había distintas "capas funcionales". Argumentó una disociación entre las capas altas y las bajas, de tal forma que el rompimiento –una "disolución" en la terminología de Jackson– causaba una reversión a la siguiente capa superior de control.

Hay una gran similitud en la esencia de lo que escribió Jackson y algunos trabajos contemporáneos en robótica [Brooks 85], [Conell 90]. En general sus escritos muestran las bases de las funciones de percepción y acción (un punto de vista radical en su época), que ahora vemos, tienen clara analogía con propuestas recientes de acción situada [Brooks 90].

En sus propias palabras, Jackson sostenía:

> *El hombre físicamente tiene que ver con un mecanismo sensorimotor. Yo particularmente deseo insistir en que los centros altos –bases físicas de mente y consciencia– tienen esta clase de constitución, que ellos representan diferentes e innumerables impresiones y movimientos de todas las partes del cuerpo (...) esto puede ser interpretado como que los centros altos son "para la mente". Yo afirmo que ellos son "para el cuerpo" también. Si la doctrina de evolución es cierta, todos los centros nerviosos deben ser de constitución sensorimotora.*

El punto de vista de Jackson acerca de la organización funcional del SN, continúa influenciando e inspirando la investigación nueurocientífica y robótica de nuestros días, habiendo muchas evidencias empíricas –anatómicas, fisiológicas y conductuales– que respaldan la noción de capas de control en el cerebro de los vertebrados.

1.1.2. Algunos experimentos que soportan la propuesta jacksoniana.

Una prueba experimental de la existencia de una organización de capas del SN es la disociación que se puede lograr entre ellas. Es posible observar que cuando hay daño en capas superiores del SN, la competencia de los niveles bajos permanece intacta. Un ejemplo común son los experimentos con gatos o ratas, que cuando se les remueve la corteza cerebral (se eliminan la mayoría de los centros sensoriales, motores y cognitivos) conservan intacta la habilidad de generar conductas motivadas: el animal aún busca comida, come manteniendo su peso corporal, siente frío, pelea o escapa cuando es atacado, etcétera. En el caso en que la mayor parte de los hemisferios cerebrales son removidos, todas las conductas anteriores ya no se producen pero la capacidad de acciones menos complejas, como el estar de pie, caminar y comer se mantienen. Cuando en este caso se remueven además el cerebelo y el bulbo espinal, el animal ya no puede coordinar movimientos, no se puede parar ni iniciar la marcha; sin embargo los movimientos que componen las acciones antes mencionadas, aún son posibles.

Los experimentos aquí relatados llevan a concluir que los niveles anatómicos inferiores del SN de los vertebrados, organizan conductas simples mientras que los niveles superiores realizan formas más complejas de "cooperación" conductual.

1.2 Arquitectura de capas de control: perspectiva robótica.

1.2.1. Arquitectura de subsumción de Brooks.

Resulta interesante hacer una revisión de la ASu de Brooks [Brooks 85], ya que es una de las propuestas más importantes en robótica y que, como veremos, tiene analogía con la organización de capas de control del SNC, aunque en su momento Brooks no la generó por inspiración basada en el modelo biológico del SNC [Prescott 99].

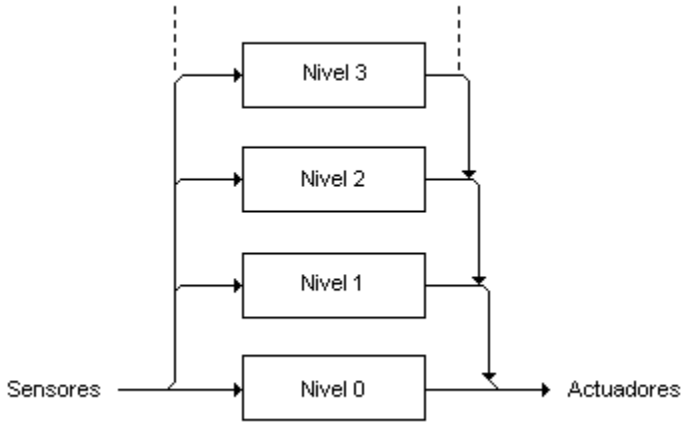


Figura 1.1. Arquitectura de capas de control de Brooks.

La ASu de Brooks se aplica por primera vez en la implementación de un robot, fundamentalmente basado en capas de control (con entradas sensoriales y módulos que compiten

por la realización de acciones motoras en los actuadores) y que se integran de manera incremental. En dicho robot, lo que corresponde al sistema de control completo, constituye el Nivel 0 de competencia o nivel cero del sistema de control (ver figura 1.1). Luego se construye y adiciona otra capa de control, que se llamará el primer nivel (Nivel 1) del sistema de control. Este nivel puede examinar datos del nivel cero del sistema y le está permitido suministrar datos a las interfases internas del Nivel 0, suprimiendo el flujo normal de datos. Esta capa, con la ayuda de la cero, logra el Nivel 1 de competencia[2]; la capa cero continúa operando a pesar de la capa superior, la cual en ocasiones interfiere con el flujo de datos.

De la misma manera se construyen y agregan niveles superiores, repitiéndose el proceso de operación antes mencionado para niveles más altos. Los niveles inferiores quedan entonces supeditados o subsumidos a la operación de los niveles altos.

Es importante que notemos aquí, que la forma en que Brooks adiciona capas de control en su ASu, tiene cierto paralelo con la evolución biológica de capas de control en el SNC de los vertebrados.

En el capítulo 2 trataremos la AMC, que guarda cierta similitud modular con la Asu de Brooks, por ello es importante tenerla en mente esta última como un referente fundamental.

1.2.2. Selección de acción en arquitecturas de capas de control.

Cuando se tienen varios módulos compitiendo por accionar un número limitado de actuadores, surge un problema de resolución de conflictos, ya que en un determinado momento, solo a un módulo le es permitido operar los actuadores. En la literatura esta situación es comúnmente conocida como el problema de **selección de acción** [Maes 90].

Brevemente mencionaremos algunos de los mecanismos más comunes de selección de acción:

Selección de acción emergente: este mecanismo pertenece al campo de conductas adaptativas de funcionalidad emergente, en donde un sistema de control es descompuesto en conductas múltiples orientadas por metas. En dicho sistema, se configura una red de conexiones excitatorias e inhibitorias entre conductas para permitir la generación de una secuencia apropiada de acciones. Inhibición mutua entre conductas incompatibles asegura que ellas no actuarán simultáneamente [Maes 90].

Selección de acción especializada: esta es una alternativa a la selección emergente, que consiste en definir componentes específicos y rutas de acción en el sistema de control, dando ventajas de modularidad a favor de la circuitería especializada inmanente, ya que con componentes separados, cada uno puede ser modificado o mejorado independientemente. Un mecanismo como este se encuentra en [Rosenblatt 97], en donde el robot construido incorpora un mecanismo de arbitraje central en una arquitectura distribuida.

Mecanismo de selección distribuida, inhibición recurrente recíproca (IRR): tenemos aquí una forma de conectividad neural, la cual es frecuentemente asociada con selección emergente. Las redes con IRR son integradas por dos o más módulos que están conectados de manera que cada uno tiene una liga inhibitoria a los otros. Aparentan tener una retroalimentación positiva,

[2] Usaremos la palabra competencia como “habilidad” y no como “competición”.

donde incrementando el nivel de activación de un módulo, causa incremento de inhibición en los otros módulos, por lo tanto, se reduce el efecto inhibitorio en el primero. De esta manera el mecanismo de IRR puede soportar una implementación del tipo “el ganador toma todo” [Holland 90].

Mecanismo centralizado de selección de acción: sistemas con este mecanismo emplean un dispositivo especializado de conmutación central (una especie de árbitro) que se encarga de la coordinación de los diferentes módulos. Su funcionamiento es similar al anterior, solo que este economiza en conexiones, ya que cada módulo que se agrega, adiciona solo dos conexiones al conmutador central (2n conexiones en total), mientras que en IRR, cada módulo que se agregue implica que se adicionarán dobles conexiones hacia cada uno de los módulos (n(n-1)) conexiones en total [Holland 90].

Ante las dificultades de selección de acción y las posibles alternativas, el mismo Brooks ha hecho notar que el problema de diseñar un esquema de subsumción apropiado, se hace más difícil cuanto más capas de control se agregen. Él mismo ha investigado otras arquitecturas que incluyen mecanismos diseñados para mejorar la selección de acción [Brooks 91]. Es claro que esto permite abrir una línea de investigación que considere los descubrimientos de los sistemas naturales (biológicos, conductuales, etológicos) de control, como una fenomenología que puede ser útil en el diseño de “robots físicos”.

CAPÍTULO 2. LA ARQUITECTURA DEL SNCM.2

En el capítulo anterior pudimos darnos cuenta que el SNC biológico tiene una composición de capas o módulos funcionales en competencia. Considerando una estructura similar, presentaremos ahora la arquitectura de módulos cooperativos: un esquema alternativo a la ASu de Brooks, la cual implementamos en el robot que denominamos SisNeρ.

2.1. Arquitectura de módulos cooperativos.

Con los antecedentes de la estructura modular del SNC y la semejanza de esta estructura con la ASu de Brooks, corresponde ahora presentar la arquitectura subyacente del SNCM.2, la AMC, la cual trata de emular en forma sencilla, las características modulares del SNC biológico. La figura 2.1 muestra la estructura básica de la arquitectura.

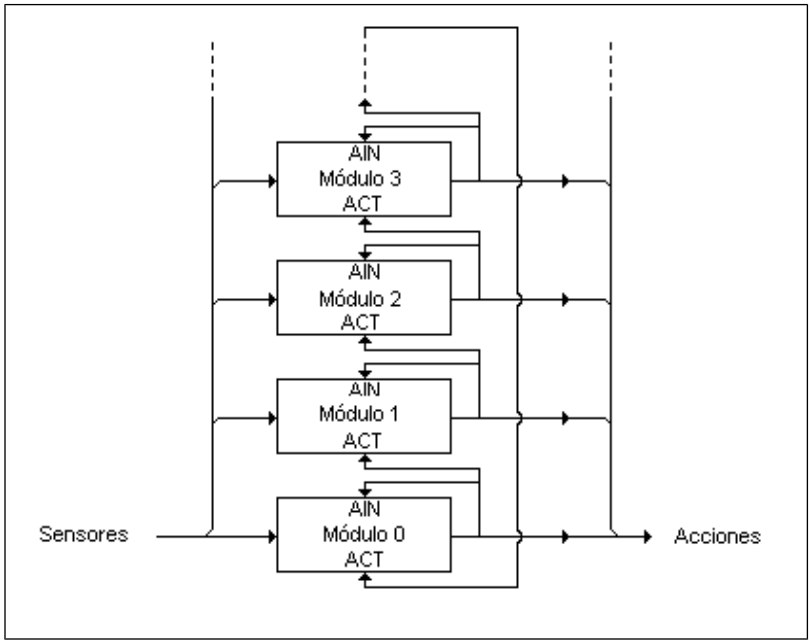


Figura 2.1 Arquitectura de módulos cooperativos.

La figura 2.1 muestra la organización modular de la AMC, donde observamos que cada uno de los módulos tiene la capacidad de ingresar la información de los sensores y también cada uno de ellos puede realizar determinadas acciones. Cada módulo tiene una entrada de activación (ACT) que le permite actuar exclusivamente a él hasta que genere una señal de autoinhibición (AIN), con la que detiene su actuación y cede el turno al siguiente módulo y así sucesivamente. La construcción de una AMC se realiza de forma incremental, el diseñador decidirá la jerarquía de prioridad que otorgará a cada módulo, siendo el módulo 0 el más prioritario, por realizar el proceso más importante del sistema, el que reditúa mayores beneficios para el alcance de un objetivo común.

El módulo 0 (o cualquiera otro de los módulos), tiene acceso a la información generada por los sensores solo cuando le corresponde el turno de actuar. Al recibir la señal ACT, el módulo inicia su proceso, el que luego de ejecutarse, genera una señal de AIN para terminar con su actuación, "pidiendo ayuda" (cediendo el turno de actuación) al siguiente módulo, modulo 1, que recibe la señal ACT. Este modo de actuación lo denominaremos "el jugador que toma ayuda" y de "paso de estafeta", el cual se repetirá en los siguientes módulos, para luego reiniciar con el módulo 0 y así sucesivamente hasta lograr el objetivo general del sistema. Puede darse el caso de que algún módulo alcance su "nivel de ineficiencia", en este caso generará también la señal de AIN y "cederá la estafeta" al módulo siguiente, para que este a su vez, ejecute su proceso continuando con la secuencia. Emerge entonces una conducta cooperativa, en lo posible, para lograr un objetivo común.

Características de la AMC:

Como se puede observar en la figura 2.1, la AMC tiene una organización secuencial y cíclica, ya que solo actúa un módulo a la vez y en el momento que actúa el último módulo, cede el turno nuevamente al módulo 0.

Cada módulo tiene registros de parámetros propios y no hay comunicación o paso de información entre módulos, teniendo cada uno de ellos un desempeño independiente.

Se pueden agregar tantos módulos como se quiera, la única limitante es que el proceso del nuevo módulo coopere con la actuación de los demás y que se le dé la jerarquía adecuada en el sistema.

Es posible tener disociación entre capas, ya que si se eliminan o fallan capas superiores, las capas inferiores continuarán efectuando sus procesos.

El sistema no presenta conflictos de selección de acción, ya que la organización de operación de los módulos es, en este momento, secuencial.

2.2. Implementación de SisNeρ con la AMC.

La aplicación de un modelo teórico en un ente real, conlleva un sin número de detalles, no obstante los describiremos en los siguientes párrafos así como en el capítulo 3.

Pasemos ahora a la implementación de la AMC, esta la realizamos en el SisNeρ, un robot real que consiste en un brazo mecánico con movimientos análogos a los de un brazo humano (movimientos articulares de cintura, hombro, codo y muñeca) y una base móvil con ruedas. A este robot se le agregó en su "instrumento" (pinza), un sensor infrarrojo montado en un mecanismo de "escáner", para integrar lo que llamaremos módulo de rastreo. La figura 2.2 muestra la organización de los diferentes módulos de SisNeρ.

El proceso de control de cada uno de los módulos que integran el SisNeρ lo simulamos en software, teniendo cada uno de ellos acceso a la información sensorial (ver figura 2.2) y salida a un actuador o servomotor que efectúa alguna función motriz en el robot.

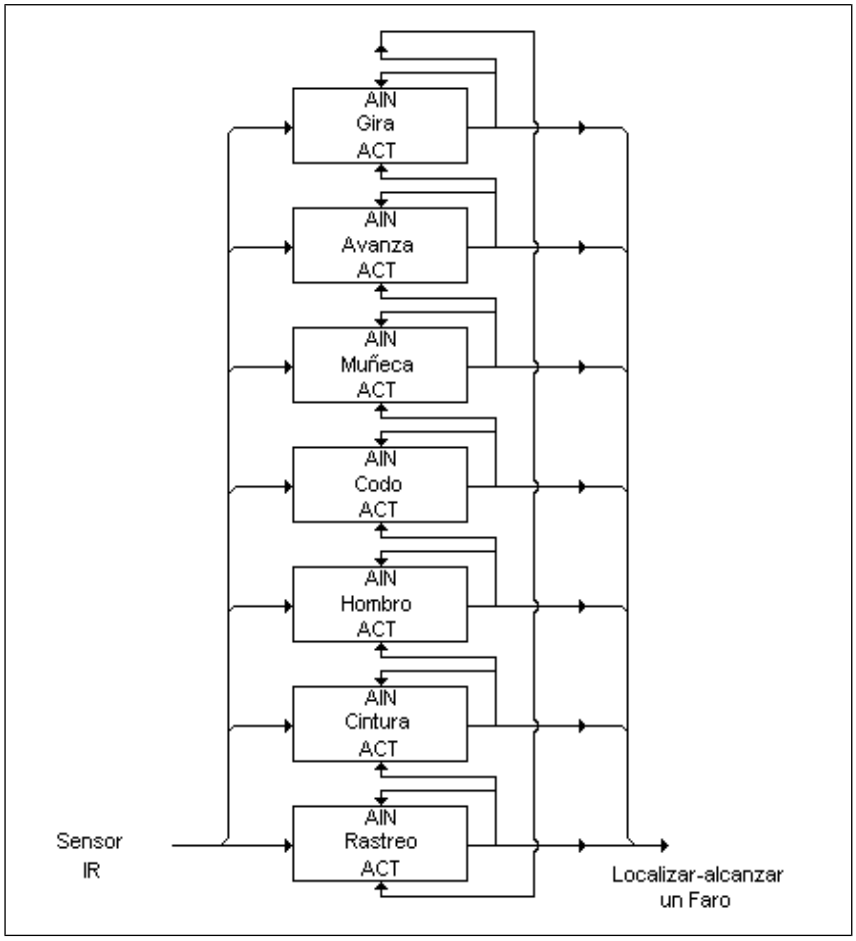


Figura 2.2. Arquitectura de módulos cooperativos de SisNeρ.

2.2.1. Módulos componentes de SisNeρ.

A continuación describimos los módulos que conforman al SisNeρ:

Módulo de rastreo. Este módulo ejecuta el proceso más importante para la realización de la conducta que pretendemos emerja en el robot, ya que si el objetivo es que localice y alcance un faro IR, la primera acción del robot tendrá que ser la localización del faro. El módulo de rastreo emplea un mecanismo de escáner (ver sección 3.1.1.2), sobre el cual está colocado un sensor IR.

El proceso del módulo de rastreo efectúa en el escáner un movimiento angular incremental del sensor IR hasta localizar el faro, en ese momento el proceso se detiene. En la sección 2.2.2 explicamos con más detalle cómo es que actúa el sensor IR para detectar al faro.

Módulos de cintura, hombro, codo y muñeca. Estos módulos realizan movimientos angulares en las articulaciones correspondientes del brazo del robot. Cada uno, en su turno, toma la información del sensor, efectúa su proceso y como consecuencia, realiza movimiento en la articulación respectiva, buscando una posición óptima para afinar la acción de apuntar hacia el faro con el brazo. El proceso que empleamos en estos módulos utiliza un algoritmo de

aprendizaje por reforzamiento conocido como **algoritmo del "bandido"** [Kaelbling 93] y [Sutton 98], el cual consiste en la aplicación de una "estrategia" de juego por parte de un apostador. En la sección 2.2.3 describimos el funcionamiento de este algoritmo.

Módulos de avanza y gira. Ambos módulos se encargan de las funciones de movimiento de las ruedas. El módulo de avanza usa en su proceso una versión modificada del algoritmo del "bandido", para decidir si avanzando, el robot puede acercarse al faro (utilizar el algoritmo del "bandido" tal cual, implicaría que el robot avanzara y retrocediera, esto último no tiene caso para nuestros fines, por eso modificamos el algoritmo original, para que el robot solo avance). El módulo de gira emplea el dato de desplazamiento angular del sensor IR y lo imita con las ruedas, claro esta, con un escalamiento adecuado, para lograr un efecto de "alineación" entre el sensor IR y la base del robot, es un módulo "ejecutor".

2.2.2. Principio de localización del faro IR.

Una premisa elemental en el funcionamiento de los diferentes módulos del robot SisNeρ, es que siempre actuarán para mover las diferentes partes del robot, de tal manera que la pinza del mismo se posicione dentro del ángulo de emisión del faro IR, para ubicar precisamente el centro del "cono" de luz IR (considerando un espacio tridimensional) donde la emisión IR es más intensa. La figura 2.3 muestra esquemáticamente cómo cuando actúa alguno de los módulos en el plano XY (rastreo ó cintura), se obtienen datos de intensidad luminosa para diferentes ángulos del sensor IR respecto al faro.

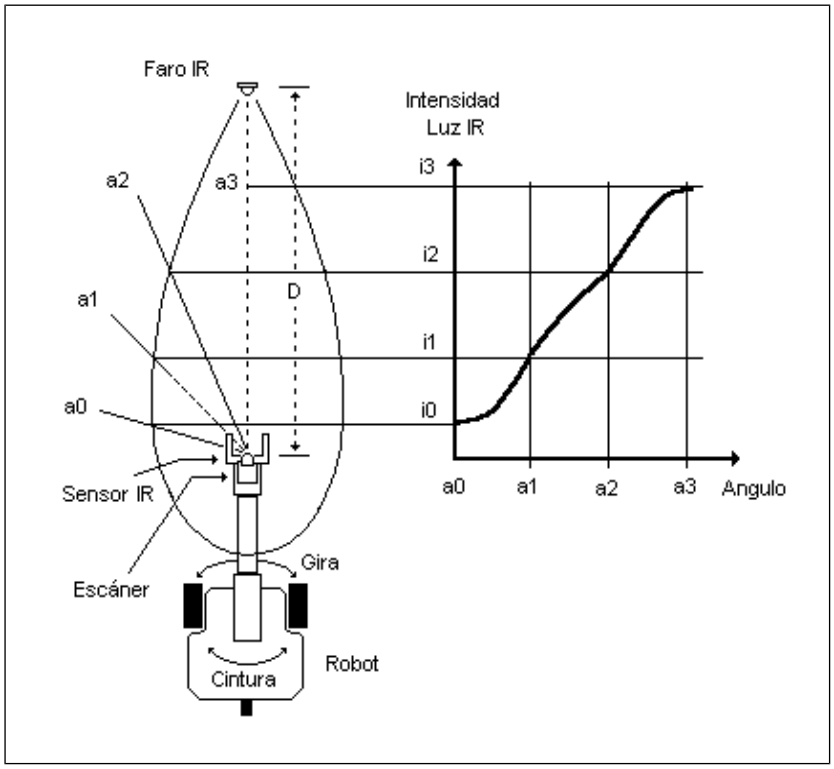


Figura 2.3. Localización del cono de luz en el plano XY.

Consideremos el caso del módulo de rastreo, donde inicialmente el escáner posiciona al sensor IR en el ángulo **a0**, se obtiene una lectura de intensidad luminosa **i0** (ver figura 2.3). Al mover el sensor a la posición **a1**, el valor de **i1** aumentará debido a que el sensor se está moviendo hacia el centro del cono de luz. Cuando en nuestro ejemplo, el sensor llega hasta la posición **a3**, se obtiene la máxima intensidad luminosa **i3**, en este momento se ha encontrado el centro del cono de luz infrarroja y se detendrá el proceso del módulo.

Cuando el robot se encuentra en las condiciones ideales de la figura 2.3, es decir, perfectamente alineado con el faro y lo hacemos avanzar hacia él con pequeños desplazamientos, disminuyendo la distancia D, obtendremos que la intensidad luminosa aumenta, esto es, que la intensidad luminosa es inversamente proporcional a la magnitud de la distancia entre el robot y el faro.

En el caso de la actuación de los módulos del brazo (hombro, codo y muñeca), que generan movimientos angulares en el eje Z del espacio de actuación del robot, la respuesta en el sensor es similar a como se muestra en la figura 2.3, con lo que realmente se completa que el robot actúe en el cono de luz infrarroja del faro.

2.2.3. El algoritmo del "bandido".

En la literatura de aprendizaje por reforzamiento, el algoritmo del "bandido" es un ejemplo clásico, de un apostador que aplica una estrategia de "mantente con el ganador pero cambia con el perdedor", esto es, mientras el apostador gane, permanecerá con su plan de juego, si pierde, tendrá que cambiar su plan [Kaelbling 93]. La figura 2.4 muestra esquemáticamente la organización de un módulo que usa el algoritmo mencionado, así como los elementos que lo componen.

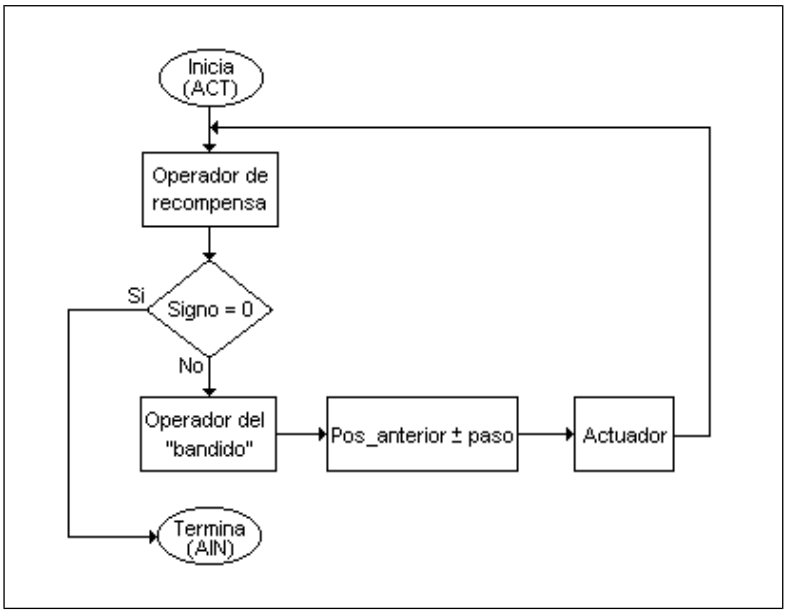


Figura 2.4. Esquema de un módulo que aplica el algoritmo del "bandido".

Como podemos observar en la figura 2.4, tenemos varios elementos que conforman a los módulos que usan el algoritmo del "bandido", estos son: operador de recompensa, operador del

"bandido", un registro de posición, un actuador y una condición de terminación de la actuación del módulo; a continuación los describimos.

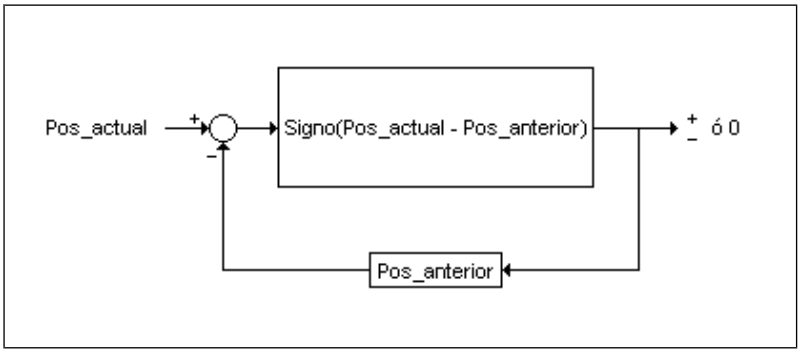


Figura 2.5. Operador de recompensa.

El operador de recompensa recibe como entrada la información del sensor IR, que indica la posición del sensor respecto al faro (Pos_actual, ver figura 2.5). Este operador genera un signo +, - ó cero, dependiendo de la comparación que realiza; si el dato de entrada (Pos_actual) es mayor que el dato anterior (Pos_anterior), la salida será un signo positivo; si la entrada es menor que la anterior, la salida será un signo negativo y si ambos datos son iguales, la salida es cero.

Cuando el operador de recompensa ha realizado su función, transfiere el resultado obtenido a un comparador (ver figura 2.4), el cual verifica el resultado: si es cero el proceso termina, en caso contrario, continúa.

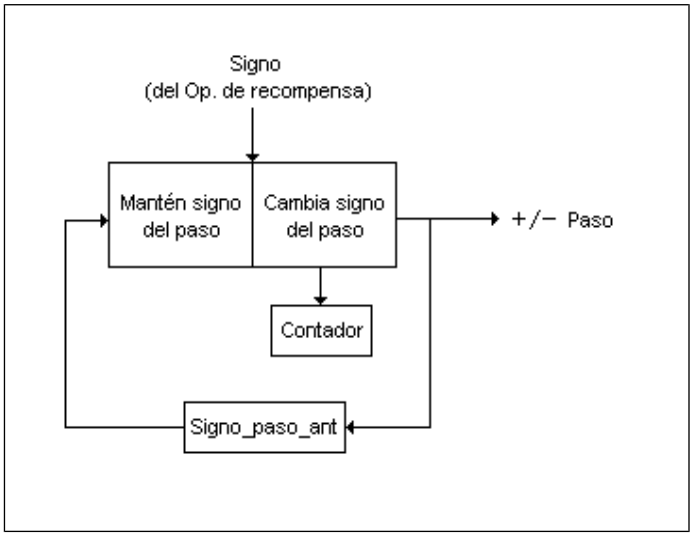


Figura 2.6. Operador del "bandido".

El operador del "bandido", como ya se mencionó, emplea un algoritmo de aprendizaje por reforzamiento. En la figura 2.6 mostramos esquemáticamente este operador, el cual podemos considerar como una especie de arreglo combinacional. Dicho operador funciona de la siguiente manera: recibe un signo del operador de recompensa, si es positivo, arrojará como salida un paso del mismo signo que Signo_paso_ant, es decir, se mantendrá la dirección de movimiento del

actuador (mantén signo del paso); en el otro caso, si el signo de recompensa es negativo, a la salida tendremos un paso con signo contrario al de Signo_paso_ant, esto es, cambiará el sentido de giro del actuador (cambia signo del paso). Para mayor claridad, la tabla 2.1 muestra combinacionalmente cómo actúa el operador del "bandido".

En la figura 2.6 podemos ver que existe un contador en el operador del "bandido", que realmente no interviene en el resultado del operador, pero que registra los cambios de signo del paso. El número acumulado por el contador, servirá también como mecanismo de autoinhibición del módulo, cuando llegue al límite de intentos preestablecidos (nivel de ineficacia), de esta manera evitaremos que el actuador pueda llegar a permanecer oscilando indefinidamente.

Signo Op. de recompensa	Signo_paso_ant	Signo del Paso de salida	Acción
+	-	-	Se mantiene el signo del paso
+	+	+	Se mantiene el signo del paso
-	-	+	Cambia el signo del paso
-	+	-	Cambia el signo del paso

Tabla 2.1. Resultados de las combinaciones en el operador del "bandido".

Por último, volviendo a la figura 2.4, tenemos que el resultado de la actuación del operador del "bandido" es transferido a un registro, para actualizar el dato de la posición anterior y así tener la posición actual, la cual pasa al actuador (lo que moverá alguna articulación del robot) y se repetirá el ciclo del módulo.

CAPÍTULO 3. EL SisNeρ

3.1. Implementación mecatrónica de SisNeρ

Una vez que se decide llevar la AMC a un robot real, había dos caminos: construir el robot o comprarlo. Construirlo implicaba tiempo y conocimientos que no teníamos, así que se decidió comprarlo, siempre y cuando la inversión no resultara onerosa.

Para encontrar un robot comercial, se tuvieron dos premisas principales, funcionalidad y bajo costo. Considerando la primera, el robot debería ser similar en características al de la simulación de J. Negrete [Negrete-Mtz 00], es decir, un brazo mecánico sobre un carro con ruedas, que le permitan desplazarse y segundo, que como primer intento, su costo no debería ser elevado, para que resultara accesible a los presupuestos de la MIA.

El robot que se consiguió fue el **kit para armar de Lynxmotion, Inc, Modelo 5AA-KT**, que cumple con las espectativas de funcionalidad a un bajo costo, incluye accesorios como baterías, cargador, servos, etc. La figura 3.1 muestra un esquema a bloques de los diferentes elementos físicos de nuestro sistema alrededor del robot, el cual describimos a continuación.

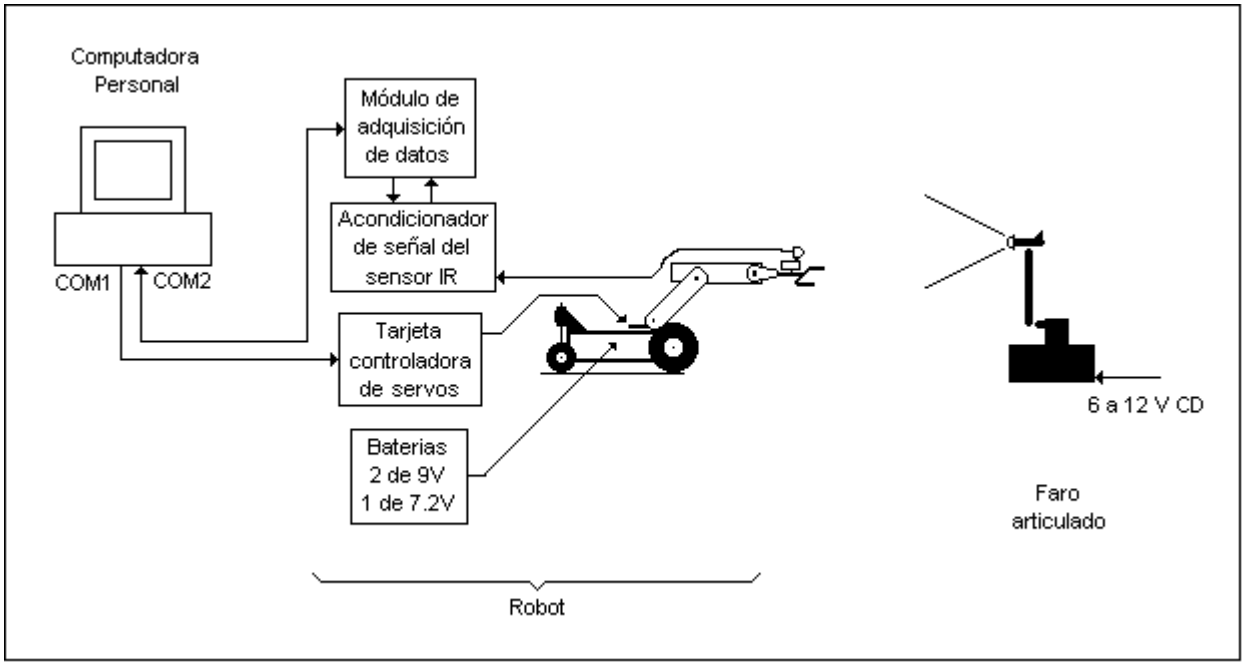


Figura 3.1. El robot SisNeρ y los elementos físicos del sistema.

3.1.1. Robot

3.1.1.1. Base móvil.

La base del robot, es una estructura que semeja una especie de emparedado formado por soportes y dos placas de acrílico. En medio de la base hay un platillo movido por un servo (en donde se coloca el brazo mecánico). A los lados de la base van colocados dos servos más para las

ruedas motrices y en el interior de la misma están colocadas las baterías (9V y 7.2V) . En la parte posterior de la base va colocada una rueda loca, que no tiene funciones motrices, pero que sirve como tercer punto de soporte de la base del robot, junto con las dos ruedas motrices, ver figuras 3.2 y 3.9.

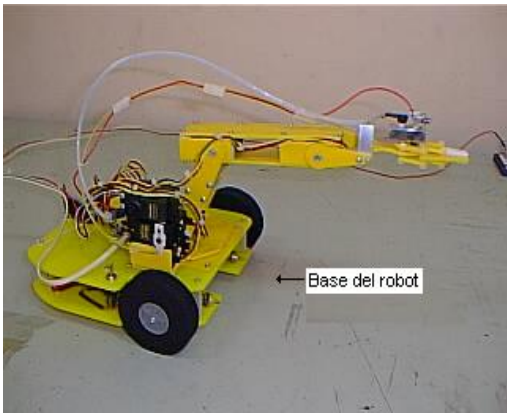


Figura 3.2. El robot SisNeρ.

3.1.1.1.1. Servos.

Los servos ó servomotores son los elementos motrices del robot, utiliza tres en la plataforma y seis más en el brazo. Los servos son pequeños sistemas electromecánicos que están integrados por un motor de c.d., un juego de engranes reductores de velocidad, un sensor de posición y un control de posición, como se aprecia en las figuras 3.3 y 3.4.

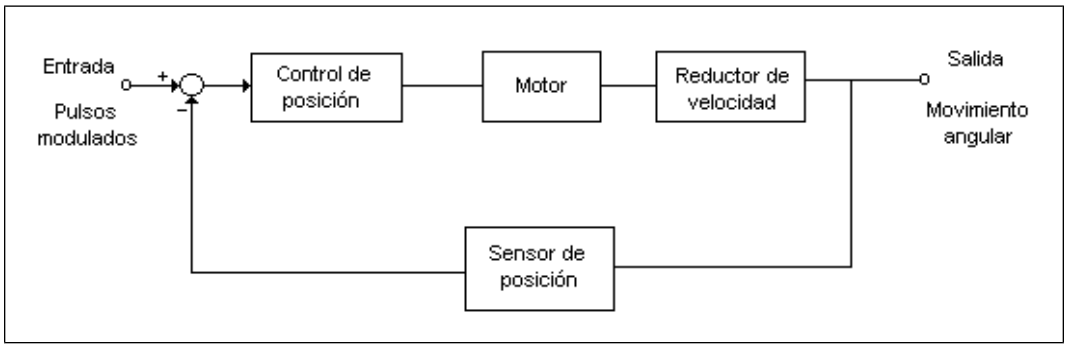


Figura 3.3. Diagrama a bloques de un servo.

En la figura 3.3 observamos que el servo es un pequeño sistema de control con retroalimentación de posición (el sensor de posición es un resistor variable, un potenciómetro), en el que se tiene de entrada una señal eléctrica de pulsos modulados y como salida un movimiento angular máximo de aproximadamente media vuelta.[3]

[3] En el presente trabajo se usan indistintamente servos HITEC Mod. HS-300 y FUTABA Mod. FP-S148.

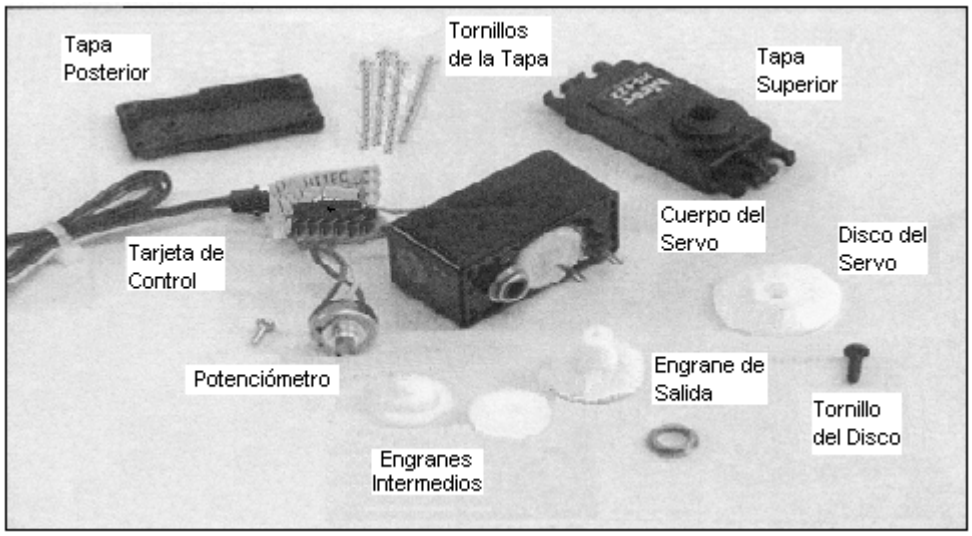


Figura 3.4. Partes de un servo convencional.

La señal que el servo recibe a la entrada, es una secuencia de pulsos, en los que se modula el nivel alto [Edwars 99], como se aprecia en la figura 3.5, con lo que se obtienen movimientos angulares en el sentido de las manecillas del reloj ó al contrario, partiendo normalmente de la posición de centrado. Para propósito de control de los servos vía la PC, cada servo está identificado por un número entre 0 y 7 y los desplazamientos angulares se especifican numéricamente con valores entre 0 y 255, teniéndose la posición central en 127.

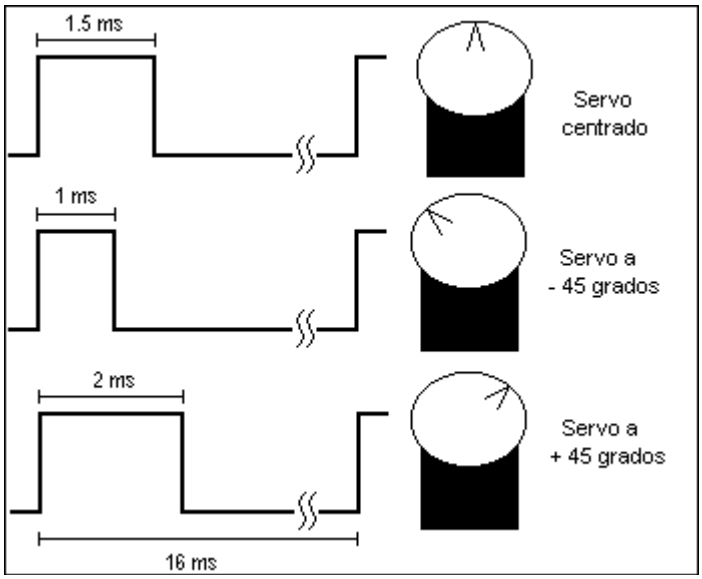


Figura 3.5. Señales que reciben los servos.

La tabla 3.1. enumera los servos utilizados para el robot, su ubicación y los límites de desplazamiento angular.

Servo	Función en el robot	Límite inferior	Límite superior
0	Movimiento de cintura	60	190
1	Movimiento de hombro	80	190
2	Movimiento de codo	80	190
3	Movimiento de muñeca	60	190
4	Movimiento de pinza	90 (cerrada)	200 (abierta)
5	Movimiento de rueda izquierda	> 127 adelante	< 127 reversa
6	Movimiento de rueda derecha	> 127 adelante	< 127 reversa
7	Movimiento de escáner (componente agregado)	20 izquierda	230 derecha

Tabla 3.1. Especificaciones del uso de los servos en el Robot.

3.1.1.1.2. Servos de las ruedas.

Para los servos de las ruedas hicimos algunas modificaciones, debido a que, como se mencionó en el punto anterior, los servos por naturaleza están limitados para moverse menos de una vuelta y en las ruedas requeríamos de un giro continuo del engrane ó eje de salida. La primera modificación la realizamos en los servos de ambas ruedas, consistió (como lo muestra la figura 3.6) en remover una muesca del engrane de salida, que sirve normalmente como límite mecánico. La segunda modificación la hicimos también en ambos servos, consistió en remover de su lugar el potenciómetro de posición, realizando un corte en el cuerpo del servo (como aparece en la figura 3.7) para sacar el potenciómetro, pasando los cables por el agujero que se hizo en el corte y dejándolo fuera del servo. El potenciómetro lo utilizaremos para ajustar que el servo no gire al polarizarlo.

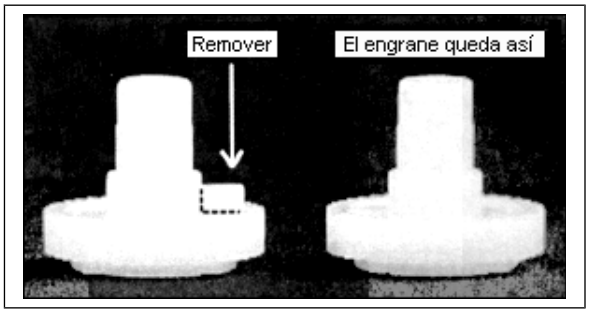


Figura 3.6. Modificación al engrane de salida.

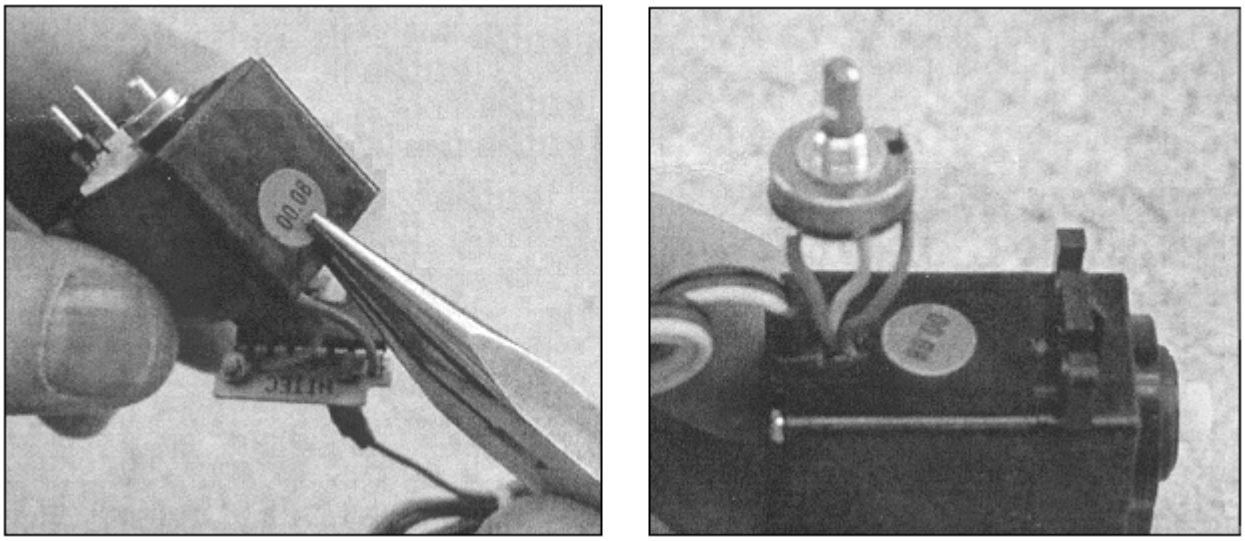


Figura 3.7. Sacando el potenciómetro del servo.

La última modificación, solo en el servo de la rueda izquierda, consistió (ver detalle en la figura 3.8) en desoldar los cables que alimentan al motor del servo e invertir la posición de los mismos.[4]

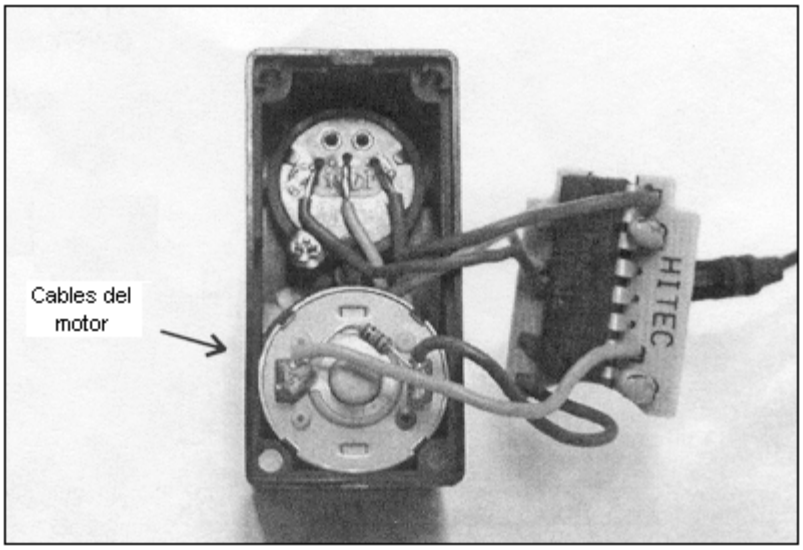


Figura 3.8. Modificación de invertir la polaridad del motor del servo.

En cuanto a la colocación de los servos de las ruedas van insertados en los costados de la base del robot y el eje de salida de cada servo, está acoplado directamente a cada una de las ruedas motrices. La figura 3.9 muestra la colocación de los servos y las ruedas.

[4] · La información completa de las modificaciones se puede ver en [Lynxmotion 00].

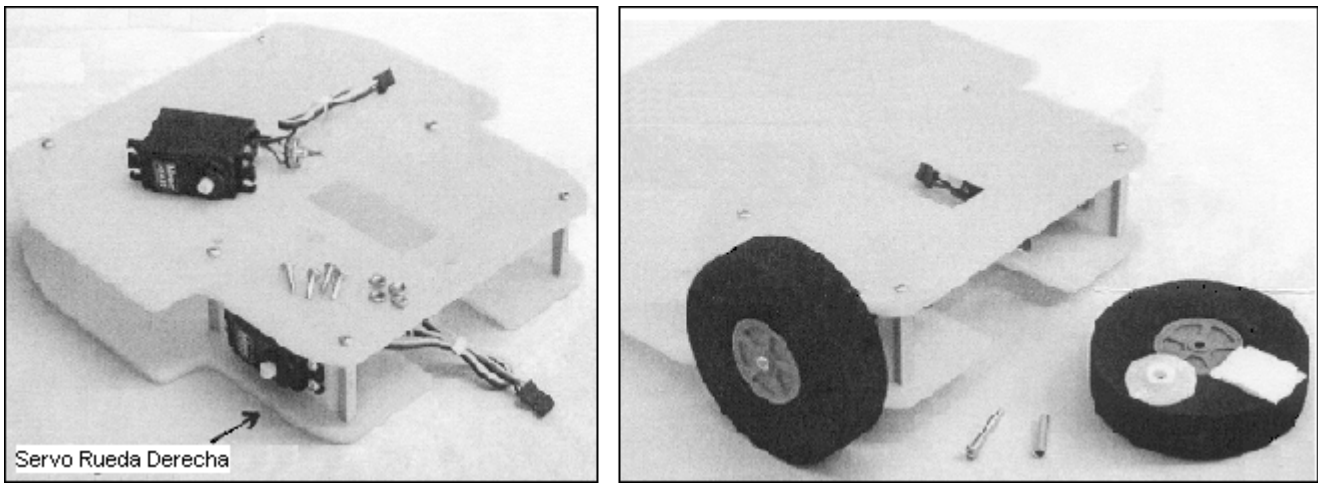


Figura 3.9. Detalles de la colocación de los servos y las ruedas en la base.

Por último, colocamos en su parte posterior de la base, una "rueda loca" que permite que el robot sea soportado por tres ruedas, dos de las cuales generan movimiento y que harán posible que el robot pueda avanzar, retroceder y girar en un sentido u otro, sobre su centro de gravedad.

3.1.1.2. Brazo.

El brazo robótico Lynxmotion cuenta con los movimientos que a continuación de mencionan y que semejan un brazo humano (figura 3.10). Cada movimiento es realizado por un servo[5] acoplado directamente en la articulación, excepto en la pinza, que es movida indirectamente mediante un cable de acero que pasa a través de una funda semirígida. Movido también por un cable de acero, incorporamos un escáner que lleva montado un sensor IR, este es un dispositivo que no venía originalmente en el kit del robot y que construimos y adaptamos específicamente para este trabajo.

Partes del brazo:

Cintura: permite giro del brazo completo a la izquierda o derecha, teniendo como eje fijo la plataforma del robot.

Hombro: es una articulación que permite un movimiento angular respecto a la cintura y que sirve como soporte del codo y hasta la pinza.

Codo: articulación con movimiento angular similar al hombro, al cual tiene como referencia. Es soporte de la muñeca y hasta la pinza.

[5] Se habla en general de que cada servo mueve una articulación, sin embargo el hombro es una excepción, ya que es movido por dos servos en paralelo, para que la articulación tenga la potencia suficiente que le permita soportar el peso del brazo.

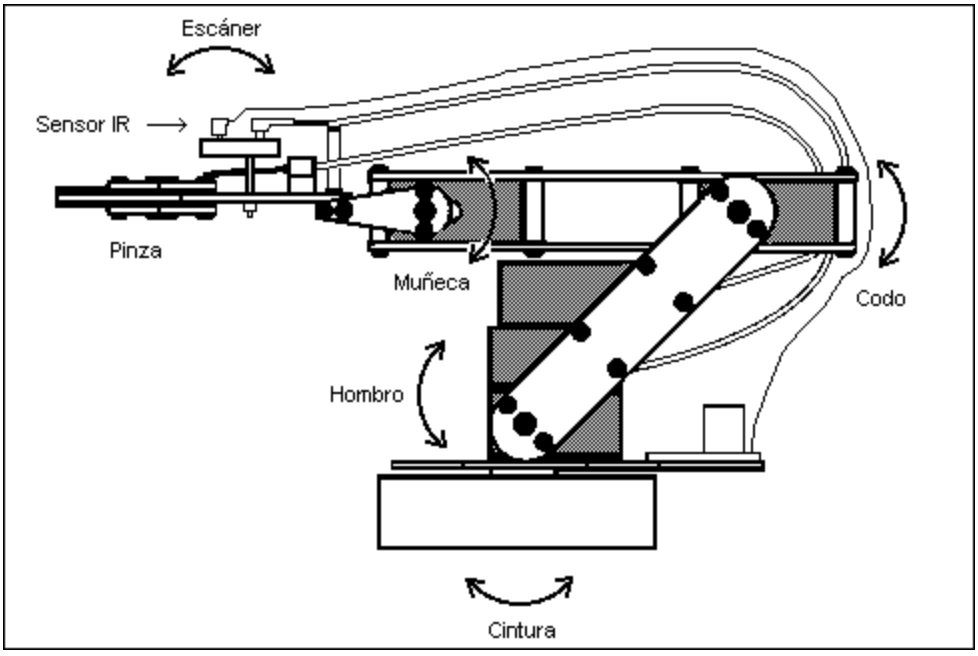


Figura 3.10. Brazo robótico con escáner.

Muñeca: articulación similar a las anteriores pero con un brazo de palanca mucho menor, como la del brazo humano, solo que a diferencia de esta, solo tiene un movimiento, como se marca en la figura 3.10. La muñeca tiene como referencia al codo y en el otro extremo tiene unida la pinza.

Pinza: es una pinza mecánica articulada con un recubrimiento de goma en los bordes, tiene movimientos de abrir y cerrar para poder sujetar pequeños objetos. Cabe mencionar que para los fines de esta tesis, la pinza no fue usada.

Escáner: dispositivo integrado por dos engranes dentro de un encapsulado metálico, sobre el cual está montado el sensor IR. Para que el escáner actúe, uno de los engranes del ensamble es movido por un servo a través de un cable flexible de acero; dicho engrane transmite movimiento al segundo engrane, sobre el que está colocado el sensor IR. Mediante el escáner, el sensor IR tiene un movimiento angular de aproximadamente 180° hacia el frente de la pinza y está montado sobre esta, sin interferir ni con su apertura o cierre, ni con el espacio de sujeción de objetos. La función del escáner es permitir explorar el ambiente hacia el frente del robot para localizar el faro infrarrojo. El movimiento del escáner aunado al de la muñeca, se equiparan a los movimientos de la herramienta terminal de un brazo robótico convencional [Asfall 92].

3.1.1.3. Controlador de servomotores.

Para poder comandar la totalidad de los servos del robot desde la PC, vía el puerto serial, es necesario contar con una interfaz, ésta es la tarjeta identificada como "Mini SSC II" (controlador serial de servos) de Scott Edwars Electronics, Inc [Edwars 99] y que va colocada en la base del robot, junto al brazo. Es una tarjeta fabricada ex profeso para manejar servos como los que usa el robot, mediante un puerto "serial RS232" (con cable y conectores tipo telefónico); por este conducto se informa a la tarjeta qué servo se requiere mover (0 a 7) y a qué posición (como lo especifican los límites de la tabla 3.1). La tarjeta maneja hasta 8 servos (ó más si se colocan tarjetas similares en "cascada"). Para operar, la tarjeta requiere de dos baterías, una de 9 V que alimenta los circuitos digitales y otra de 4.8 a 6 V para energizar los servos, ver figura 3.11.

La tarjeta se configura mediante los "puentes" de configuración y se usó como sigue (más detalles en [Edwars 99]):

- Velocidad de comunicación: 9600 bauds
- Identificación de los servos: 0 a 7
- Desplazamiento angular de cada servo 90°

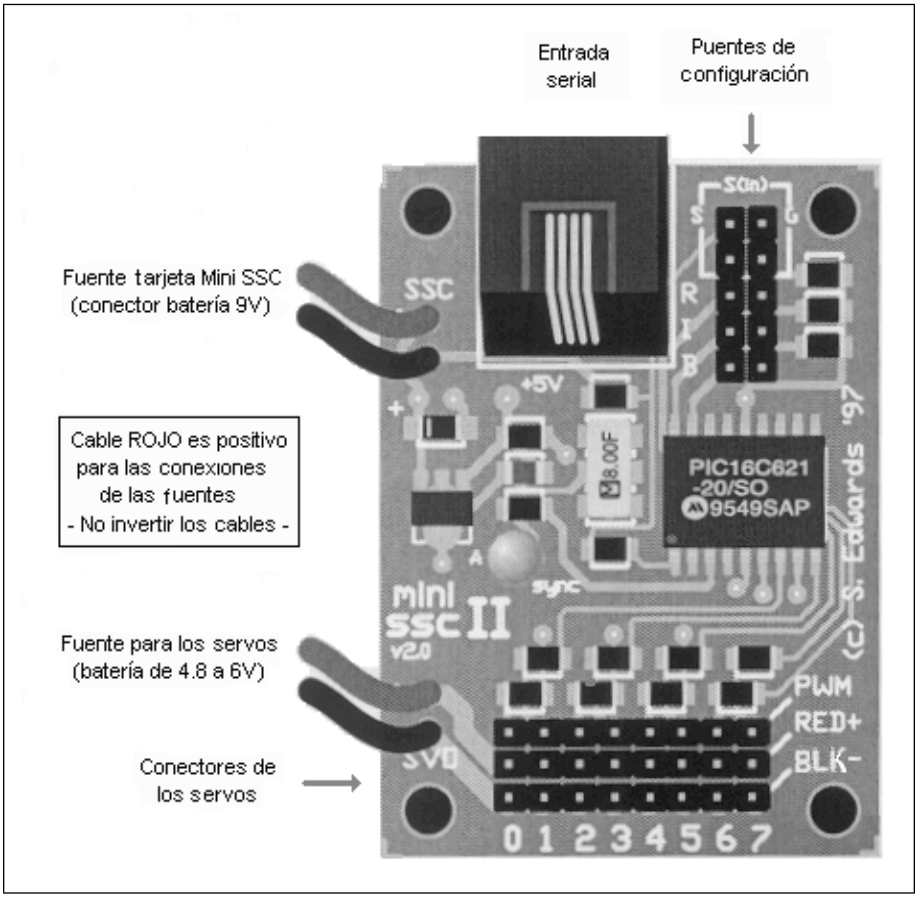


Figura 3.11. Esquema de la tarjeta Mini SSC II.

Cuando ponemos en funcionamiento al robot, colocando en posición de encendido su interruptor (a un costado del brazo), se energizan la tarjeta Mini SSC II y los servos; la tarjeta envía a cada uno de los servos una señal como la mostrada en la figura 3.5, que corresponde a una posición de 127 en todos los casos, lo que permite que el robot adopte siempre su posición de inicio y ahí espera para recibir comandos de posición hacia los diferentes servos. Es claro que la tarjeta Mini SSC II mantiene energizados a los servos en todo momento y hasta la última posición que hayan recibido, esto permite al robot auto-sostener su estructura, cosa que no sucede cuando se apaga el interruptor de encendido del robot, en ese momento pasa como si cortáramos las cuerdas a un títere y se colapsa.

3.1.2. Sensor infrarrojo.

En esta sección describiremos el sensor IR del robot y los circuitos acondicionadores de señal, diseñados en base a circuitos analógicos con amplificadores operacionales. El sensor y los circuitos asociados deben detectar una luz infrarroja que es generada en forma intermitente por un emisor infrarrojo. La información de intensidad luminosa que el sensor IR detecte será transferida a la PC mediante un módulo de adquisición de datos (MAD) a través de otro puerto "serial RS232".

3.1.2.1. Sensor.

El sensor utilizado para este proyecto, lo buscamos entre los más sensibles que había disponibles comercialmente. Encontramos un fototransistor tipo NPN marca SILONEX, Inc, modelo SLT-50HL4 (ver apéndice A2). Este fototransistor está contenido en un encapsulado metálico, con un lente óptico en la ventana de entrada, lo que lo hace altamente sensible y confiable en ambientes hostiles. Lo polarizamos en configuración de "emisor común", como se aprecia en la figura 3.12, en donde mostramos el circuito acondicionador de señal.

3.1.2.2. Acondicionador de señal.

El circuito acondicionador de señal, se encarga de polarizar al sensor, recibir su señal, amplificarla, filtrarla y dejarla en condiciones adecuadas para que pueda ser procesada por un MAD y después enviada a la PC vía un puerto serial. El diagrama esquemático del circuito es el mostrado en la figura 3.12.

En el diagrama del circuito acondicionador de señal podemos observar básicamente cinco etapas: la del sensor y su buffer, la del amplificador, la del circuito de ajuste de sensibilidad, una etapa de filtrado y la de detección de pico con su buffer de salida.

Sensor y su buffer: como ya se mencionó, el sensor lo polarizamos en configuración de emisor común; la señal que de él obtenemos es enviada a un amplificador operacional (AmpOp) en su configuración más sencilla [Franco 88], la de seguidor ó buffer. Aquí el AmpOp actúa como reforzador de la señal del sensor y como desacoplador de impedancias.

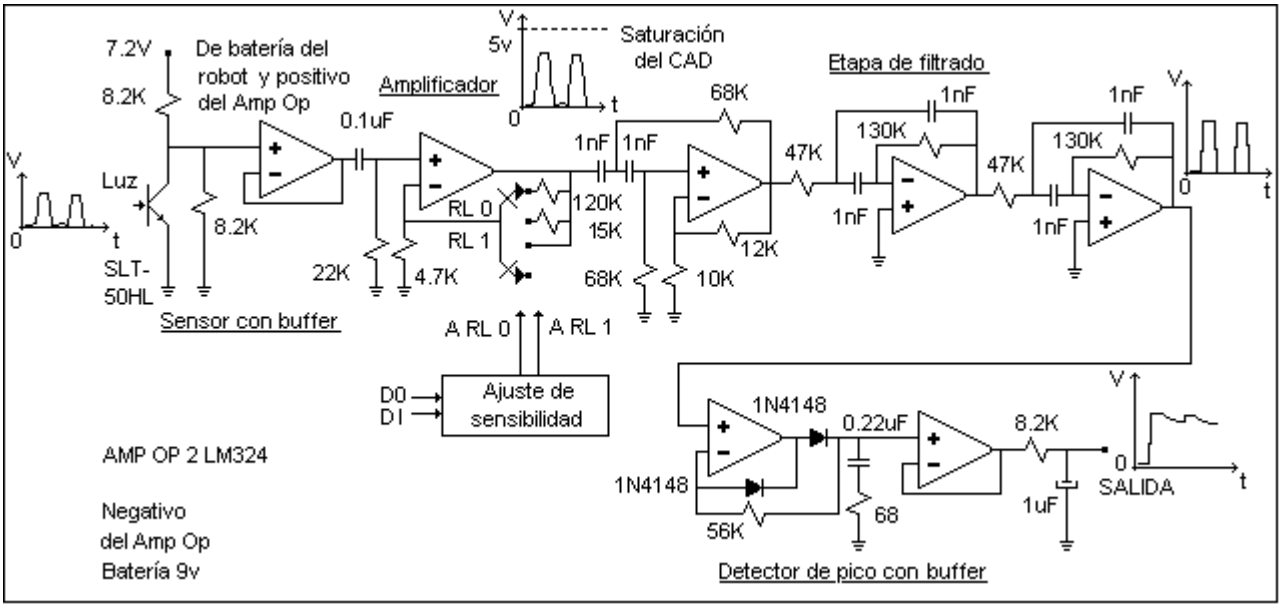


Figura 3.12. Diagrama del circuito acondicionador de señal del sensor IR.

Amplificador: esta etapa eleva el nivel de la señal (voltaje) para permitir su procesamiento, ya que en la etapa anterior, es posible que apenas se generen unos cuantos milivolts. Primeramente la señal pasa por un sencillo filtro pasivo pasa-altos (formado por el capacitor de 0.1uF y el resistor de 22k) para luego entrar al AmpOp. La configuración que ahora tenemos es la de un amplificador, con la particularidad de que tenemos un circuito de ajuste de sensibilidad, que permite seleccionar una de tres posibles combinaciones de ganancia (las que dan las resistencias de retroalimentación); esto para evitar que el MAD se sature conforme el robot se acerca al faro (ver especificaciones del MAD en la sección 3.1.2.3). La ganancia del amplificador está dada por la siguiente ecuación:

$$\mathbf{Vo = (1 + R2 / R1) Vi}$$

Si substituimos valores tendremos tres casos:

- Caso 1. R1 = 4.7K y R2 = 120K Vo = (1 + 120K / 4.7K) Vi = **25.5 Vi**

- Caso 2. R1 = 4.7K y R2 = 15K Vo = (1 + 15K / 4.7K) Vi = **3.2 Vi**

- Caso 3. R1 = 4.7K y R2 = 0 Vo = (1 + 0 / 4.7K) Vi = **Vi**

Esto nos permite tener tres diferentes ganancias en el amplificador, que serán seleccionadas por el programa de control del robot mediante el circuito de ajuste de sensibilidad, conforme el sensor IR se vaya acercando al faro; de esta manera tendremos una operación continua del robot y sin saturación del MAD, hasta tener un acercamiento entre el sensor IR y el faro, a una distancia mínima de aproximadamente 1 cm.

Circuito de "ajuste de sensibilidad": se puede decir que el circuito de "ajuste de sensibilidad" es parte del amplificador, sin embargo preferimos separarlo para mayor claridad de su explicación, ver figura 3.13. La función de este circuito es la de seleccionar qué resistor de retroalimentación actuará en el amplificador, para definir la ganancia, ya sea no habilitando los relevadores ó habilitando uno u otro (los tres posibles casos de ganancia ya mencionados) . El circuito tiene dos entradas (D0 y D1) que reciben señales digitales del programa a través del MAD, las cuales definen la actuación del circuito y dos salidas, que manejan la actuación de los relevadores, los cuales mientras no estén energizados, permanecen en su posición de normalmente cerrado (NC mostrado en el esquemático de la figura 3.12). En la figura 3.13 podemos ver el circuito de ajuste de sensibilidad, con los transistores BC547B junto con los resistores asociados de 68 y 150 ohms, los cuales forman cada uno, un circuito amplificador de corriente, que permite energizar la bobina de los relevadores con una corriente de 80 mA, suficiente para hacer actuar el interruptor del relevador y evitar sobrecargar las salidas digitales del MAD, que tienen una capacidad de corriente de salida muy baja, de apenas unos cuantos miliamperes.

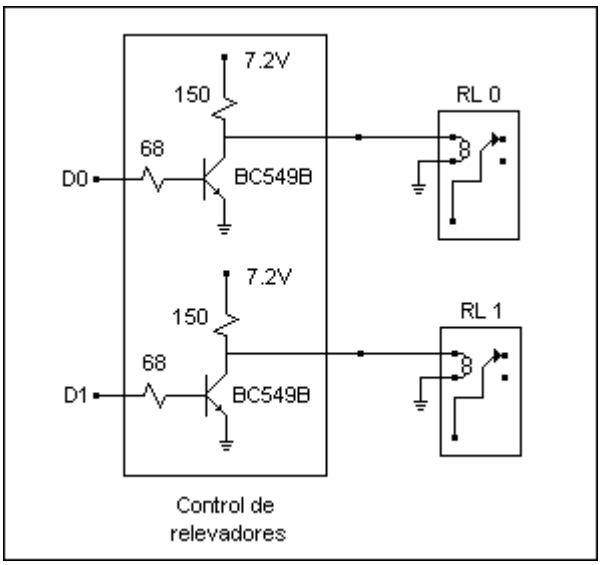


Figura 3.13. Esquemático del circuito de "ajuste de sensibilidad".

Etapa de filtrado: cuando se realiza el diseño de un circuito, normalmente se considera que los componentes y señales tienen un comportamiento ideal, sin embargo, la realidad no es así. Es por esto que generalmente se requiere de una etapa de filtrado, para tratar de minimizar el ruido inherente del circuito y para seleccionar la frecuencia de emisión requerida. Nuestra etapa de filtrado está integrada por una serie de tres AmpOp que forman dos filtros activos: el primer OpAmp y los componentes asociados tienen una configuración de filtro activo pasa altos y los otros dos OpAmps con sus componentes forman un filtro activo pasa banda.

Detector de pico: la función de esta etapa es mantener registro del nivel de amplitud o voltaje de la señal que es recibida; de una señal alterna, tendremos después de esta etapa una señal continua cuya magnitud corresponde al nivel de amplitud de la señal que recibe. El capacitor de 0.22uF y el resistor de 68 ohms permiten que las variaciones de voltaje de salida sean lentas para que el MAD, al recibir esta señal, pueda realizar su función de conversión analógica digital correctamente. El buffer de salida refuerza la señal y hace un desacoplamiento de impedancias con la entrada del MAD.

3.1.2.3. Módulo de adquisición de datos (MAD).

Para poder ingresar señales eléctricas a la computadora ó para poder extraerlas, es necesario disponer de un módulo ó tarjeta de adquisición de datos. Comercialmente existe una gran variedad de estos dispositivos, desde algunos muy sencillos con unas cuantas entradas y salidas, de poca resolución, que tienen un bajo costo, hasta algunos muy sofisticados, con un sin número de entradas y salidas, de alta resolución y con un costo alto. Para nuestro trabajo no son muchos los requerimientos, solamente necesitamos ingresar una señal analógica y es necesario contar con dos salidas digitales, así que conseguimos a un bajo costo, un pequeño modulo de adquisición de datos, de fácil manejo vía el puerto serial y que satisface nuestras pretensiones. Este módulo es de la marca B&B Electronics, Inc modelo 232SDA12 (ver figura 3.14) con las siguientes características (para mayor detalle referirse a [B&B 95]):

- Entradas analógicas : 11, de 0 a 5v.
- Resolución del convertidor analógico digital (CAD): 12 bits.
- Entradas digitales: 3 , compatibles TTL.
- Salidas digitales: 3, compatibles TTL.
- Alimentación: vía puerto serial o con adaptador externo de 12V.

En las especificaciones podemos observar que la señal analógica que necesitemos ingresar al MAD, debe de estar acotada en un intervalo de 0 a 5v, para evitar saturación del CAD interno. En la sección 3.2.5.2 se detalla su programación.

Figura 3.14. Módulo de adquisición de datos.

3.1.3. Emisor infrarrojo.

El emisor IR ó faro, es un circuito formado por tres partes: una etapa reguladora basada en el circuito integrado (CI) LM 2931, un oscilador construido con el CI temporizador LM555 y un emisor infrarrojo, que es un diodo emisor de luz (LED) con un ángulo de emisión amplio. La figura 3.15 muestra el diagrama esquemático del emisor IR. En la figura 3.16 podemos ver el faro completo como un dispositivo articulado direccionable en forma manual.

3.1.3.1. Regulador.

La etapa del regulación está formada por el CI LM2931, que es un regulador que da como salida 5V y tres capacitores que actúan como filtros de voltaje, ver figura 3.15. En general, al circuito del emisor IR lo podemos alimentar con cualquier voltaje mayor a 6V y hasta 30V (para que el regulador tenga margen de actuación, ya que voltajes menores causarán que el regulador no opere correctamente y por ende todo el circuito) con lo que el regulador mantendrá sin cambios el voltaje de salida, a un valor constante de 5V.

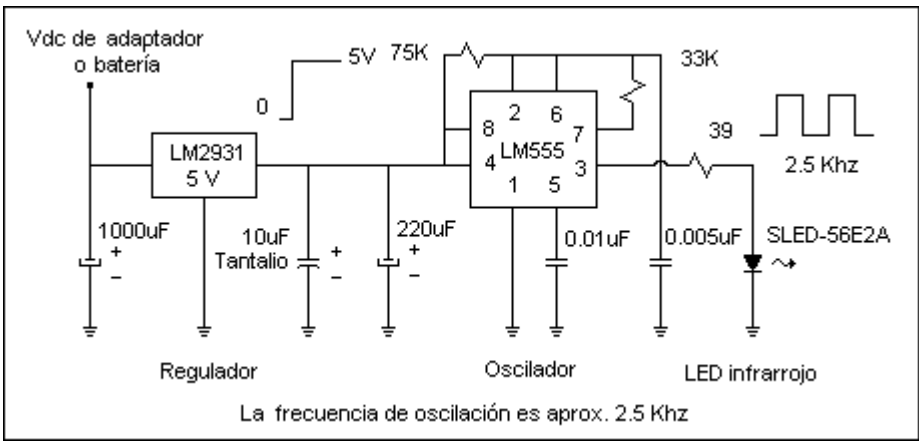


Figura 3.15. Diagrama esquemático del emisor infrarrojo.

Figura 3.16. Faro articulado.

3.1.3.2. Oscilador.

Como se aprecia en la figura 3.15, el oscilador esta integrado por el CI LM555, dos capacitores y dos resistores. El 555 (como popularmente se le conoce) es un circuito muy versátil (ver la gran cantidad de aplicaciones en [Jung 91]), en este caso lo utilizamos en una configuración de "multivibrador astable", que nos da una secuencia de pulsos cuadrados de aproximadamente 2.5 khz de frecuencia, determinada esta por la combinación RC de los resistores (75K y 33K) y el capacitor de 0.005uF.

3.1.3.3 LED infrarrojo.

La etapa de salida del circuito (ver figura 3.15), la forman un LED infrarrojo y un resistor de 39 ohms que funciona como limitador de corriente. El LED es de arsenuro de aluminio, que emite radiación a 880nm, de la marca SILONEX, Inc, modelo SLED-56E2A, el cual tiene una emisión infrarroja intensa, con un ángulo de emisión amplio y un consumo bajo de corriente (ver hoja de especificaciones, anexo A3).

La salida del circuito del faro es entonces una señal intermitente de luz infrarroja, que es más fácil de procesar por el sensor IR que una señal continua, lo que permite minimizar el "ruido" infrarrojo ambiental.

Existe una limitación en el uso del faro, esta es que puede ser colocado en cualquier posición, pero siempre deberá apuntar, de preferencia radialmente al centro de masa del robot; de otra manera será difícil que el robot lo localice si es orientado de otra forma.

3.2. Implementación computacional de SisNeρ

El programa con el que implementamos la AMC para controlar al robot SisNeρ, fue realizado en lenguaje **C** por las facilidades que ofrece de manejo de puertos de la PC, apuntadores, etc. y para preservar compatibilidad con el simulador robótico de J. Negrete [Negrete-Mtz 00], ya que ambos se sustentaron en las mismas bases conceptuales.

La estructura del programa contempla entre sus componentes esenciales al "programa principal", que consiste en los llamados de activación de los procesos de los diferentes módulos, las funciones que usan los procesos de los módulos con el algoritmo del "bandido", el módulo que usa el algoritmo del "bandido" modificado, módulos que emplean un algoritmo propio y funciones misceláneas. A continuación describimos el programa del robot y los diferentes elementos. En el anexo A6 adjuntamos el programa completo para cualquier referencia.

3.2.1. Programa principal

La sección de programa principal de la implementación computacional de SisNeρ, consiste en llamado a las funciones que activan la operación de los diferentes procesos de los módulos. El programa principal en pseudo-código quedó como mostramos a continuación.

```
            // ************** Programa principal **********

while (no se presione una tecla) // Ejecuta los procesos de los módulos
  {
    rastreo(escaner, pos_escaner)  // Activacion de módulo de rastreo
                                   // Activación de los demás módulos
    frac = 0                      // Inicializa contador de fracasos
    do { algoritmo(cintura, registros_cintura) } // Módulo de cintura
      while( (payoff no sea 0 ) y (frac menor ó igual a lim_frac) )
    frac = 0
    do { algoritmo(hombro, registros_hombro) }   // Módulo de hombro
      while( (payoff no sea 0 ) y (frac menor ó igual a lim_frac) )
    frac = 0
    do { algoritmo(codo, registros_codo) }       // Módulo de codo
      while( (payoff no sea 0 ) y (frac menor ó igual a lim_frac) )
    frac = 0
    do { algoritmo(muñeca, registros_muñeca) }   // Módulo de muñeca
      while( (payoff no sea 0 ) y (frac menor ó igual a lim_frac) )
    do { algoritmo_avanza(registros_avanza) }    // Módulo avanza
      while( payoff no sea 0 )
    gira(pos_escaner)                            // Módulo gira
  }
apagar_relevadores
```

Una vez inicializados de los parámetros de posición de los diferentes módulos (ver anexo A6), el programa principal entra en un ciclo `while`, comienza entonces el llamado a los procesos de los módulos del robot. Primeramente se llama al proceso del módulo de rastreo con el escáner, el cual actuará hasta encontrar el faro. Una vez que lo encuentra, se autoinhibe y activa al módulo de cintura, el cual actuará tratando de optimizar la posición de la pinza respecto al faro, acercándola lo más posible a éste, mientras no se cumplan las condiciones de autoinhibición, esto es, que el operador de recompensa (`payoff`) sea diferente de cero y que el nivel de ineficiencia (contador `frac`), sea menor o igual al límite del contador (`lim_frac`); si alguna de las dos condiciones deja de cumplirse, termina la actuación del módulo y activará al "relevo", el módulo de hombro.

Los módulos de hombro, codo y muñeca, actuarán individualmente en forma similar al de cintura, hasta que finalmente actúan los de avanza y gira, para luego repetir el ciclo completo, iniciando con el rastreo y así sucesivamente, haciendo que el robot presente conductas de localizar el faro, moverse hacia él y de extender el brazo lo más posible hasta prácticamente tocarlo con la pinza. En lo que resta del capítulo 3, describiremos cómo operan las diferentes funciones que los módulos utilizan.

3.2.2. Módulos que emplean el algoritmo del "bandido".

Los módulos que aplican el algoritmo del "bandido", son básicamente los encargados de los movimientos angulares del brazo robótico, estos son, los módulos de cintura, hombro, codo y muñeca.

Cada módulo está integrado como sigue: tiene un identificador, el cual es un número que corresponde al número de servo definido para la tarjeta Mini SSC II (controladora de servos), registros de posición, de intensidad luminosa (lectura del odómetro) y de la última acción

realizada. La tabla 3.2 muestra los diferentes módulos que usan el algoritmo del “bandido”, sus identificadores y registros.

Nombre	Servo	Registro de Posición	Registro de Lectura Odómetro	Última Acción Realizada
Cintura	0	pos_cint	kt_cint	act_cint
Hombro	1	pos_homb	Kt_homb	act_homb
Codo	2	Pos_codo	Kt_codo	act_codo
Muneca	3	Pos_mune	Kt_mune	act_mune

Tabla 3.2. Los módulos, sus identificadores y los registros que manejan.

La función del programa que utilizamos para operar el proceso de cada uno de los cuatro módulos es `algoritmo`, dicha función es la implementación del algoritmo del “bandido” descrito en el capítulo 2 y la describimos a continuación.

```
función algoritmo( servo, registros_servo) // Algoritmo del “bandido”

   kt = lectura odometro
   payoff = signo(kt - kt_1)

                 // Calculo de la nueva Act con la tabla 2.1
   if( payoff > 0 ) { if( act_1 > 0 ) act =  1 }
   if( payoff > 0 ) { if( act_1 < 0 ) act = -1 }
   if( payoff < 0 ) { if( act_1 > 0 ) act = -1, frac++ }
   if( payoff < 0 ) { if( act_1 < 0 ) act =  1, frac++ }
   pos_servo = pos_servo + (act * paso); // Nueva posición
   lims_art(servo)               // Normalización de la posición
   mueve(servo, pos_servo) // Movimiento del servo o articulación
  }
```

Al invocar la función algoritmo, el proceso del módulo que la llama es identificado y transfiere los valores de sus registros para que estos sean considerados como valores anteriores (`kt_1, act_1), kt` es actualizada (lectura del sensor IR) con el llamado a la función `odometro` para obtener la entrada al operador de recompensa (payoff), como el signo de las diferencias entre `kt` y `kt_1`. A continuación es evaluado el signo del operador de recompensa con el algoritmo del “bandido” (tabla 2.1), para decidir la nueva acción con el operador de integración (tres posibilidades: mover el servo un paso continuando con el movimiento anterior, “mantén signo del paso”, moverlo en sentido contrario, “cambia signo del paso” ó no moverlo, recompensa = 0) considerando normalización de la nueva posición (tomando en cuenta la posición de los límites mecánicos para no llevar a una posición inválida) y de ser posible, entonces se realiza el movimiento.

En la función `algoritmo` podemos notar que cuando hay un cambio de dirección en el movimiento del servo, es incrementado el valor del nivel de ineficiencia, el contador `frac;` éste contador lo utilizaremos como mecanismo de terminación del proceso del módulo, con el operador de interrupción como ya se mencionó, para limitar el número de cambios de signo y evitar que el servo del módulo pueda quedar oscilando.

3.2.3. Módulo con algoritmo del “bandido” modificado

En un principio, cuando implementamos la AMC en el robot, pretendimos aplicar el algoritmo del “bandido” en todos los módulos, sin embargo, por la naturaleza de los movimientos que realizan algunos de ellos, no fue posible. El módulo de avance fue uno de estos, ya que, aunque puede realizar movimientos de avanzar y retroceder, era ilógico que pretendiéramos que realizara esto último, si el propósito del robot es avanzar para alcanzar un objetivo. Con esto en mente, hicimos modificaciones al algoritmo del “bandido” para el módulo avanza, las cuales describimos a continuación.

El módulo avanza mueve simultáneamente y hacia adelante las ruedas del robot, es el módulo más “provechoso” cuando se pretende que el robot alcance un faro. En la tabla 3.3 se especifican identificadores y registros del módulo. En esencia, éste módulo hace que el robot evalúe si la lectura del odómetro en su posición anterior es menor que la lectura del mismo en su posición actual (recompensa (+)), si es así, avanzará un paso (avanza a una velocidad determinada por un lapso de tiempo especificado) y volverá a evaluar; se mantendrá avanzando mientras gane, es decir, que el signo de recompensa sea (+).

Nombre	Servo	Registro de Posición	Registro de Lectura Odómetro	Última Acción Realizada
Avanza	5 y 6	No Aplica	kt_av	act_av

Tabla 3.3. Parámetros del módulo avanza.

Cuando el signo de recompensa sea (-), terminará la actuación de este módulo, cediendo el control a otro, hasta una nueva oportunidad. La función `algoritmo_avanza` que mostramos a continuación, realiza lo que acabamos de describir.

```
función algoritmo_avanza( registros_avanza )
                                  // Algoritmo del “bandido” modificado
  kt = lectura odometro
  payoff = signo(kt - kt_1)
                                  // Calculo de la nueva Act
  if( payoff > 0 ) { if( act_1 > 0 ) act =  1 }
  if( payoff <= 0 ) { payoff = 0, termina }
  ruedas(act, paso_avance)       // Avanza el robot
  termina
```

En la función `algoritmo_avanza` sabemos de antemano que los servos que se moverán son los de las ruedas, así que no se especifican como parámetros (están definidos en la función `ruedas`). Los valores de los registros que se transfieren a la función son los de última lectura del odómetro (`kt_1`) y la última acción (`act_1`). Es de hacer notar que el módulo avanza no tiene variable de posición, esto es debido a que los servos 5 y 6 pueden girar indefinidamente, entonces a través de estos servos no se puede tener referencia de posición; en realidad no nos interesa la situacionalidad del robot en el plano X Y, así que el módulo funciona sin este parámetro.

3.2.4. Módulos con algoritmo propio.

Se tienen dos módulos que no aplican de ninguna manera el algoritmo del “bandido”, debido a que se requiere que realicen acciones específicas que no tienen que ver con la lógica de dicho algoritmo. Estos módulos son: giro de la base del robot y rastreo del sensor; a continuación los describimos.

3.2.4.1. Módulo de giro de la base del robot: gira.

El módulo gira con las ruedas, es un módulo que causa desplazamiento angular del robot completo en el plano XY. El módulo de cintura realiza un movimiento similar, aunque en este caso mueve solo al brazo. El módulo de rastreo con el escáner también realiza un movimiento angular en el plano XY, aunque no causa ningún desplazamiento, por que solo mueve el sensor IR. Para coordinar los tres módulos mencionados, decidimos que solo el módulo de cintura actuara con el algoritmo del “bandido” y que el módulo gira se alineara con la posición del sensor IR, para que el robot siempre muestre una actitud de orientarse hacia su objetivo (el faro), dado que es dicho sensor IR quien se encarga de localizarlo. El módulo gira es diferente de los demás, ya que no tiene un proceso, solo recibe información, la escala, la manda al actuador y termina, por eso también le llamamos modulo “ejecutor”. Los parámetros de este módulo son los que se muestran en la tabla 3.4.

Nombre	Servo	Registro de Posición	Registro de Lectura Odómetro	Última Accion Realizada
Gira	5 y 6	pos_corr ó pos_scan.	No Aplica	No Aplica

Tabla 3.4. Parámetros del módulo gira.

La función que utiliza el módulo gira es la siguiente:

```
función gira(pos_scan)  // Función para giro del robot
                        // que lo alinea con el ángulo del sensor IR
  eo = pos_scan – posición inicial servo
  ec = eo * paso_giro     // Valor de posición a corregir
  if( eo > 0 ) ruedas( giro_der,  ec) // Giro del robot a la der
  if( eo < 0 ) ruedas( giro_izq, -ec) // Giro del robot a la izq
```

La función `gira` recibe una posición (`pos_scan`) que se ha de transferir a las ruedas para que el robot gire y quede alineado con el sensor IR. Primeramente obtenemos `eo`, que es el desplazamiento efectivo del sensor IR una vez que se sustrae el valor de su posición inicial, luego hacemos un escalamiento de `eo` multiplicando por `paso_giro`. Obtenemos `ec` para hacer corresponder el ángulo de desplazamiento del sensor IR con el de las ruedas. Por último la función decide, de acuerdo al valor de `eo`, en qué sentido se ha de mover el robot, considerando que si el dato `pos_scan` es mayor a 0, el robot se moverá a la derecha y si es menor a 0, el robot se moverá a la izquierda; para realizar esto, `gira` invoca a la función `ruedas`, a la que le indica la dirección (`giro_der` ó `giro_izq`) y duración del giro (`ec`).

3.2.4.2. Módulo de rastreo del sensor.

El módulo de rastreo tiene como propósito localizar un emisor de luz infrarroja (faro). Para lograr esto, empleará el mecanismo de escáner con el sensor IR, el cual efectuará un "rastreo" hacia el frente del robot, buscando el faro infrarrojo. El proceso de rastreo es realizado moviendo el sensor IR un ángulo pequeño y luego repitiendo la operación incrementando el ángulo en cada intento, hasta hallar el cono de luz infrarroja que emite el faro. Una vez que el faro es localizado, termina la actuación del módulo, se autoinhibe y permite la actuación del siguiente módulo. Los parámetros del módulo de rastreo aparecen en la tabla 3.5, a continuación.

Nombre	Servo	Registro de Posición	Registro de Lectura Odómetro	Última Accion Realizada
Rastreo	7	Pos_art ó pos_scan	Ktmax	No Aplica

Tabla 3.5. Parámetros del módulo de rastreo.

La función que ocupa el módulo de rastreo es la siguiente:

```
función rastreo(servo, pos_art)  // Función de rastreo

  for(i = 1, i < 10, i++)
    {
      pos_art = ref - i * step   // Mover un paso al sensor y comienza
      ktmax = lectura odometro   // a buscar una lectura máxima
      mueve(servo, pos_art)
      do
        {
          buscar una lectura del sensor con máxima intensidad (ktmax),
          por encima del ruido, incrementando en pequeños pasos la
          posición del sensor, una vez que sea encontrada, termina
        }
      while( pos_art < (ref + 2*i*step) ) // Mientras no se llegue al
    }                                     // límite del escáner
  for(aux1 = pos_art, aux1 > pos_max, aux1 = aux1 - paso )
    {                                   // Mueve ahora a posición donde
      mueve(servo, aux1)                // se encontró máximo por
    }                                   // encima del ruido
  pos_art = pos_max - (2 * paso)
  mueve(servo, *pos_art)
  termina
```

Cuando la función de `rastreo` es invocada, se le transfieren los parámetros de `servo` y de `pos_art`, siendo esta última la posición que toma como referencia para comenzar el rastreo. Efectuará hasta 10 intentos de rastreo (primer `for`), incrementando en cada ciclo un escalón dado por `step` de 10 unidades de desplazamiento angular del servo. En seguida toma una lectura del `odometro`, la define como referencia de comparación `ktmax`, luego ejecuta el movimiento del escáner al extremo izquierdo dado por `step`, invocando la función `mueve` e iniciará la búsqueda del faro, moviendo el servo con incrementos de posición de un `paso a la derecha`, hasta encontrar lecturas del odómetro que indiquen que se está arribando a una cresta de intensidad luminosa, por lo tanto si la siguiente lectura cae (descontando el ruido), se puede

considerar que se encontró una lectura máxima `ktmax,` en ese momento se detiene la actuación del módulo y guarda la posición en el registro de parámetros. En caso de que no encontrara el faro en el primer intento, el ciclo se repite ampliando el ángulo de desplazamiento cada vez más, hasta un máximo de diez intentos (lo que representa un desplazamiento angular máximo de aproximadamente 180°). Si aún así el módulo no encontrara un faro (por que a propósito no se colocó, se colocó al robot lejos de él ó se fijó mal), en algún momento la función obtendrá una lectura de intensidad que sobresalga de las demás, en la dirección en la que la haya encontrado, hacia allá se dirigirá el robot, aunque no necesariamente en esa dirección se encuentre un faro.

3.2.5. Funciones misceláneas.

3.2.5.1. Apertura de puerto

La función de apertura de puerto serial (`abrir_pto`), consiste básicamente en indicarle a la PC la configuración de cómo usaremos el puerto serial (`PORT`), es decir, para este caso bebemos deshabilitar las interrupciones en el puerto (por que se usará por ejemplo, para hacer pooling con el módulo de adquisición de datos), definir la velocidad de transmisión de datos (baud rate) en 9600 bauds, el tamaño de la palabra de datos de 8 bits, sin paridad, con un bit de parada y un control FIFO del registro. Esta es una configuración típica del puerto serial y la función que la realiza, se muestra a continuación [Peacock 97]. Para detalles del uso de la función, ver en el programa del anexo A6.

3.2.5.2. Manejo del módulo de adquisición de datos.

De las diferentes opciones de entrada/salida del módulo de adquisición de datos (MAD), utilizamos dos salidas digitales (salidas D0 y D1) y una entrada (entrada 0) del convertidor analógico digital (CAD). El manejo de las salidas digitales se realiza con la función `sal_dig` que en seguida mostramos y el manejo del CAD se realiza con la función `odometro` que describimos en el punto 3.2.5.2.1.

```
función sal_dig(n)     // Función de manejo de las salidas digitales

  saca a(puerto 1, !,0,S,O) // Protocolo para manejo de salida digital
  saca a(puerto 1, n)
```

Al utilizar la función `sal_dig,` ésta recibe como parámetro el número `n,` que corresponde al estado binario de las tres salidas del MAD, de acuerdo a la tabla 3.6 (los detalles de cómo manejar el MAD se pueden encontrar en [B&B 95]). Entonces la función indica al puerto 1 (puerto serial conectado al MAD) que va a iniciar comunicación con el MAD, mandando caracteres de protocolo de manejo de las salidas digitales, indicandole en qué estado deberá poner dichas salidas, de acuerdo al número `n` y la tabla 3.6 (para el robot, utilizamos los estados binarios 0, que pone en cero todas las salidas, 1 que pone un uno lógico en la salida D0 y 2 que pone la salida D1 en uno lógico.

Número Binario	Salidas Digitales		
	D2	D1	D0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Tabla 3.6. Manejo de las salidas del MAD

3.2.5.2.1. Odómetro.

La función `odometro` es una función muy importantes del programa, ya que nos permite cuantificar en forma digital la intensidad luminosa que percibe el sensor IR una vez que ha pasado por el circuito acondicionador de señal. La entrada analógica 0 del MAD recibe la señal analógica del sensor IR, la procesa en su CAD y pone el dato digital de intensidad luminosa a disposición de la PC en un puerto serial. La función `odometro` es la que mostramos a continuación.

```
función odometro          // Función odómetro para medir intensidad de luz
                          // en el sensor infrarrojo de del escáner.
  saca a(puerto 1, !,O,R,A,0) // Protocolo para lectura del CAD
  do
    {
      c = lee(puerto 1) // Verifica si en el puerto hay un dato.
    }
    while( no haya dato)
    ch1 = lee(puerto 1)         // Lectura de MSB del CAD
    ch2 = lee(puerto 1)         // Lectura de LSB del CAD
    voltaje = conversión a decimal de ch1 y ch2  // Conversión hex a dec
    v_satura(voltaje)            // Verifica saturación del DAC
    devuelve(voltaje)            // La función devuelve el valor
                                 // de voltaje.
```

La función `odometro` tiene como característica, que una vez ejecutada, devuelve un valor, lo que permite emplearla en instrucciones de asignación. Al comenzar la ejecución de esta función, establece comunicación con el MAD, indicándole que hará lectura del CAD, se ejecuta entonces un ciclo de “pooling” para esperar a que el CAD indique que está listo el dato. Cuando esto sucede, el dato hexadecimal es ingresado en dos bytes, primero el más significativo (MSB) y luego el menos significativo (LSB). En seguida el dato leído es convertido de hexadecimal a decimal y se verifica que no haya saturación del convertidor (valor de `voltaje` cercano a 4095); en caso de que la hubiera, la función de ajuste de sensibilidad del sensor IR (que describiremos en el punto 3.2.5.4) accionará alguno de los relevadores del amplificador del sensor IR, para corregir la saturación del CAD (ver sección 3.1.2.2). Por último, la función devuelve el valor leído como resultado de su ejecución.

3.2.5.3. Manejo de tarjeta Mini SSC II y servos

La tarjeta Mini SSC II es la interfaz de la PC con los servos, vía un puerto serial, para realizar movimientos del robot. En el programa hay dos funciones que realizan los movimientos de todos los servos, una es `mueve` para los servos de articulaciones y la otra es `ruedas`, para el movimiento de las ruedas. A continuación mostramos la función `mueve`.

```
función mueve(articulación, posición ) // Movimiento de articulación

  lims_art(articulación)                // Obtiene límites para articulación
  if(posición esta dentro de los límites) // para normalizar posición
    {
      enviar(articulación, posición)      // Envía posición a articulación
    }
     else imprime("Numero invalido...")
```

El propósito de la función `mueve` es accionar los servos del brazo del robot, incluyendo el escáner. Para efectuar esto, al invocar la función se transfieren los parámetros: numero de servo (`articulación`) y la posición a la que debe de ir (`posición`). En seguida la función realiza la normalización de la posición, es decir, verifica que dicha posición, para el servo especificado, esté dentro de los limites permitidos de desplazamiento (datos de la tabla 3.1 en la función `lims_art`). Una vez normalizada la posición, la función envía los datos (`enviar`), indicando qué servo mover (`articulación`)[6] y finalmente la posición a la que moverá el servo(`posición`). En caso de que la posición especificada no corresponda a los límites del servo dado, aparecerá el siguiente mensaje: `Numero invalido...` y no habrá movimiento alguno del servo.

Respecto a la función `ruedas` es útil únicamente para manejar los servos 5 y 6. A continuación presentamos la función `ruedas`.

```
función ruedas(direccion, tiempo )  // Función de movimiento
                                    // de las ruedas dando
  if(direccion < 0){ servo_5 = 120, servo_6 = 132 // Giro a Izq }
  if(direccion > 0){ servo_5 = 132, servo_6 = 120 // Giro a Der }
  if(direccion ==0){ servo_5 = 133, servo_6 = 133 // Avance }
  enviar(5, servo_5)   // Envía a servo 5 velocidad servo_5
  enviar(6, servo_6)   // Envía a servo 6 velocidad servo_6
  retardo(tiempo)      // Duración del movimiento
  enviar(5, 127)       // Parar el servo 5
  enviar(6, 127)       // Parar el servo 6
```

La función `ruedas` recibe dos variables, de dirección en donde se indica con un número qué movimiento se debe realizar (-1 giro a la izquierda, 1 giro a la derecha y 0 avanzar) y de `tiempo` donde se especifica la duración que tendrá el movimiento. Dadas las variables de la función, lo que sigue es definir en ella los valores de velocidad para los servos 5 y 6. La tabla 3.7 muestra los valores de velocidad que se usaron[7].

[6] Con la función mueve se puede accionar cualquier servo (del 0 al 7), excepto los de las ruedas (servos 5 y 6).

[7] Los valores de velocidad que aplicamos no son los únicos, pero si los que consideramos más apropiados para no causar problemas de inercia apreciable en el robot.

Movimiento del Robot	Servo 5	Servo 6	Movimiento de las Ruedas
Girar a la Izquierda	120	132	Rueda izquierda reversa, rueda derecha avance
Girar a la Derecha	132	120	Rueda izquierda avance, rueda derecha reversa
Avanzar	133	133	Ambas ruedas en dirección de avance

Tabla 3.7. Valores de velocidad para las ruedas.

A continuación, la función `ruedas` envía los datos de velocidad a los servos 5 y 6, que permanecen actuando el lapso de `tiempo` especificado por `retardo(tiempo);` transcurrido el lapso de tiempo, con el envío de un valor de velocidad 127, ambos servos dejarán de actuar (recordar que un valor de 127 en los servos de las ruedas implica inmovilizarlos; un valor menor a 127 causará que gire en un sentido con más velocidad cuanto menor sea el valor, hasta cero como mínimo y un dato mayor a 127 causará que el servo gire en sentido contrario, con más velocidad cuanto mayor sea el número, hasta 255 como máximo).

3.2.5.4. Ajuste de sensibilidad del sensor IR.

En condiciones iniciales, cuando el robot comienza su actuación para localizar el faro y se acerca a él, observamos un problema de saturación en el CAD del MAD. Como ya se comentó, es necesario operar el circuito de ajuste de sensibilidad del amplificador del sensor IR para evitar dicha situación. La función `v_satura` es la encargada de habilitar las salidas digitales del MAD para que actúen los relevadores del circuito de ajuste de sensibilidad.

A continuación presentamos la función `v_satura`:

```
función v_satura(medida) // Función de verificación de saturación
                         // si el valor dado es mayor que 4050
  if( (val == 0) y (medida > 4050) ) { val = 1, sal_dig('1') termina }
  if( (val == 1) y (medida > 4060) ) { val = 2, sal_dig('2') }
  termina
```

Al invocar la función `v_satura`, recibe el valor `medida,` el cual viene directamente del CAD de la función odómetro. En la función tenemos una variable que sirve de “bandera”, `val` la cual nos indica el estado de los relevadores. Por defecto al iniciar la operación del robot, `val` es cero, lo que indica que ninguno de los relevadores está activo. Al ejecutar la función e ingresar un dato de `medida,` éste es comparado para verificar si es mayor de `4050;` si es así y `val` es cero, se cumple la primera instrucción `if` y la bandera toma el valor 1 al tiempo que habilita la salida digital 0 (`sal_dig('1')`), con esto actúa el relevador 0 y termina la ejecución de la función. Cuando la bandera `val` está en 1 y es invocada la función `v_satura`, el dato de `medida` es `comparado:` el primer `if` ya no actúa por el estado de la bandera, así que el segundo `if` solo se cumplirá si `medida` es mayor que `4060,` en ese caso quedará habilitada solo la salida digital 1 y con ello el respectivo relevador. La bandera tendrá ahora el valor de 2. Con lo anterior logramos en el robot un efecto de ajuste de sensibilidad del sensor IR, para poder acercarlo al faro hasta una distancia mínima de aproximadamente 1 cm.

3.2.5.5. Utilería.

Las funciones de utilería son las siguientes: `lims_art,` que efectúa la normalización de una posición a enviar a un servo, para verificar si es válida; `signo` que nos da un valor de +1, -1 ó 0, dependiendo del valor del dato que se le proporcione y `enviar` que indica al robot el servo individual a mover y la posición a la que irá. Para detalles, estas funciones aparecen en el programa del anexo A6.

CAPÍTULO 4. EXPERIMENTOS CON EL SisNeρ

A continuación presentamos algunos de los experimentos que mejor ilustran la respuesta de los componentes esenciales del robot SisNeρ y el desempeño general del mismo. Los experimentos muestran cómo el robot logra el objetivo principal de localización, acercamiento y alcance del faro con la pinza; comprueban la robustez del funcionamiento del robot en condiciones adversas para lograr alcanzar su objetivo y de corroboran algunas de las bondades de la AMC, por su similitud modular con el SNC, como son: su capacidad de disociación entre módulos y la plasticidad para lograr conductas como la de rehuir al objetivo.

4.1. Respuesta del sensor IR.

El fototransistor SILONEX Modelo SLT-50HL4 (ver anexo A2), es una parte muy importante del circuito del sensor IR. Para determinar si éste elemento podía ser útil para medir distancia entre él y el faro en términos de intensidad luminosa, hicimos la siguiente prueba: construimos un arreglo direccional que nos permitió colocar al fototransistor frente al faro infrarrojo, con la posibilidad de permitir variar el ángulo de incidencia de luz en el fototransistor y también la distancia entre éste y el faro. Consideramos cinco casos: uno con el fototransistor perfectamente alineado con el faro, dos casos con el fototransistor desviado a la izquierda en 5 y 10 grados y dos casos más, en los cuales el fototransistor estaba desviado a la derecha en 5 y 10 grados respectivamente. La figura 4.1 nos muestra los resultados obtenidos.

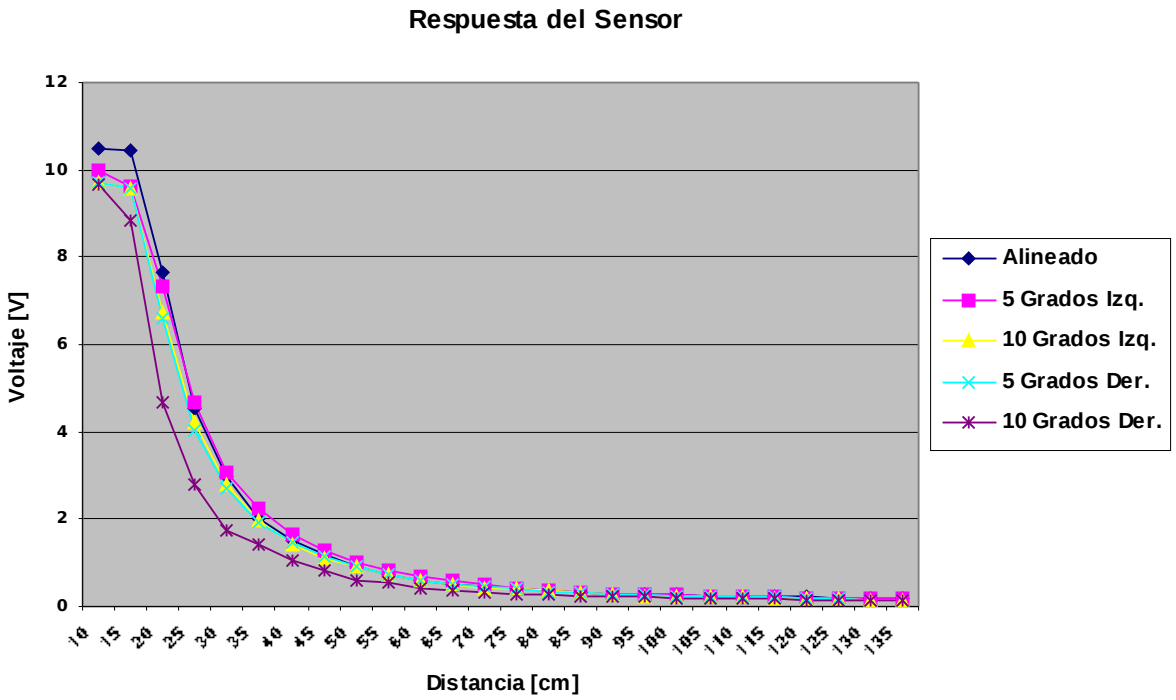


Figura 4.1. Respuesta del fototransistor del sensor IR respecto al ángulo de incidencia y distancia al faro.

Es posible apreciar claramente en la gráfica de la figura 4.1, que no hay una dependencia notable de la respuesta del fototransistor respecto del ángulo de incidencia del faro (en el intervalo de prueba direccional de 20 grados en total), ya que solo en el caso de la desviación a la derecha de 10 grado, se notan ligeras diferencias.

En cuanto a la variación de la distancia, se observa que la intensidad luminosa en términos de voltaje, es inversamente proporcional al aumento de la distancia, es decir, a mayor distancia menor voltaje; es también una relación no lineal. De lo mostrado por la gráfica de la figura 4.1, podemos considerar un intervalo útil de proporcionalidad de entre 20 y 100cm, esto sin considerar saturación en el MAD (5 volts), ya que teniendo en cuenta esto, el intervalo se reduce a entre 30 y 100cm. Esta prueba se realizó con una ganancia de 25.5 en el amplificador del circuito del sensor IR, su resultado nos llevó a considerar la necesidad del circuito de ajuste de sensibilidad, para reducir ganancia evitando saturación del MAD y para poder lograr una distancia mínima de acercamiento de 1cm entre el fototransistor y el faro, o lo que es lo mismo, entre la pinza del robot y el faro.

4.2. Localización, acercamiento y alcance del faro con la pinza.

Es necesario mencionar que para llegar a realizar estos experimentos, la AMC de SisNeρ fue implementada en forma gradual o incremental, hicimos pruebas con los módulos individuales y poco a poco los fuimos acoplando para lograr un robot que muestra un comportamiento bien definido. A continuación en la figura 4.2, presentamos los elementos físicos del robot y los accesorios usados para las pruebas.

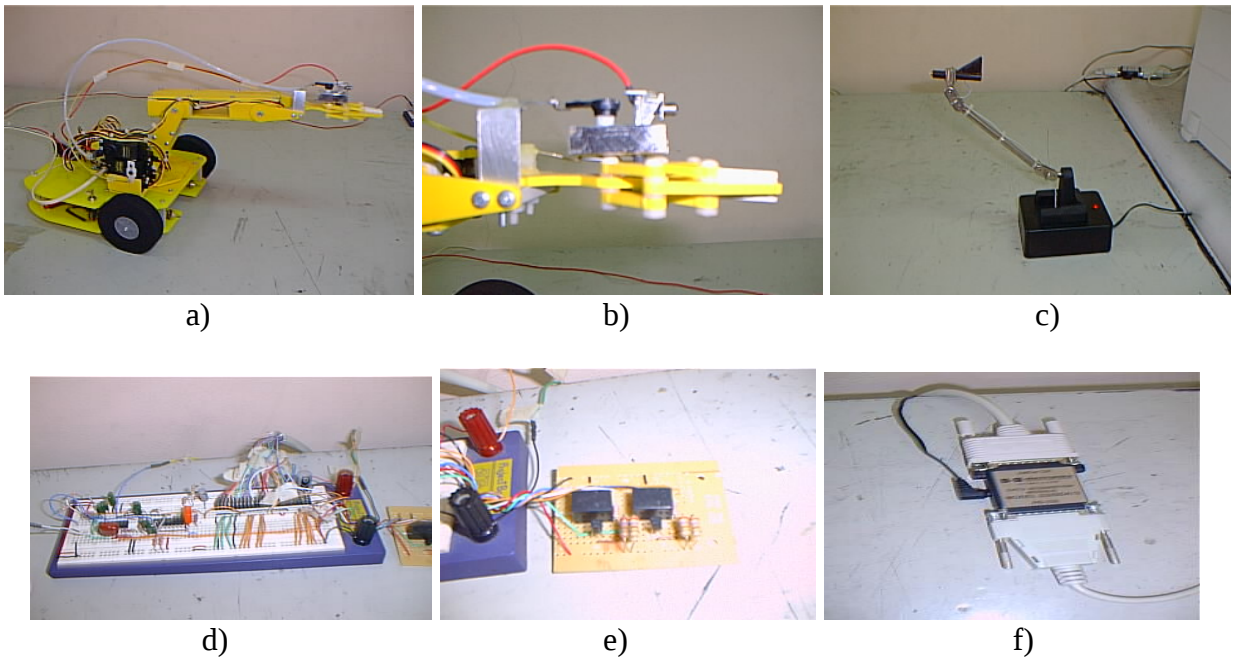
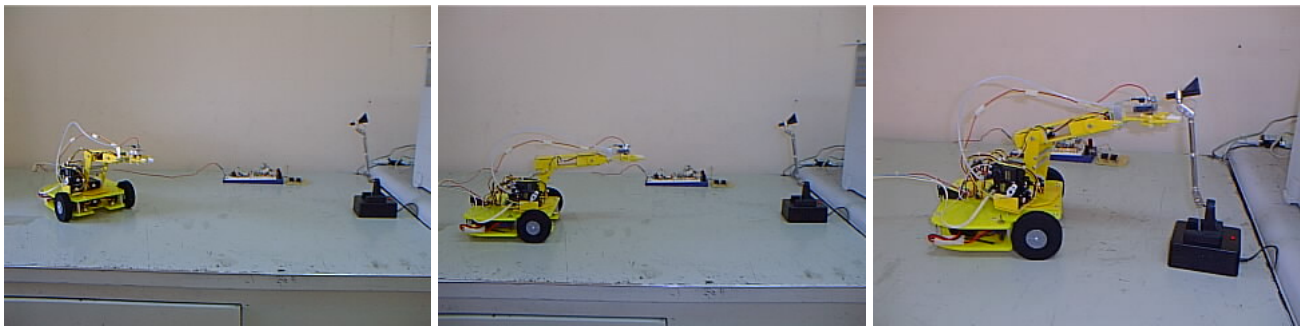


Figura 4.2. a) robot, b) detalle del escáner del robot, c) faro articulado, d) circuito del sensor IR, e) circuito de ajuste de sensibilidad, e) MAD.

De la serie de experimentos realizados con el robot SisNeρ, describiremos primeramente el que corresponde a la tarea que es el objetivo principal del robot, de localización, acercamiento y alcance del faro con la pinza. Para esto ilustraremos la actuación con una serie de imágenes, como se muestra a continuación, en dos casos: cuando el faro está completamente extendido en lo más alto y cuando el faro lo posicionamos hacia abajo.



a) b) c)

Figura 4.3. El robot alcanzando el faro: a) posición inicial, b) moviéndose, c) faro alcanzado.

En la serie de imágenes de la figura 4.3, tenemos inicialmente al robot con el brazo en su posición por defecto, orientado parcialmente hacia la cámara, figura 4.3 a), con una separación de aproximadamente 70cm entre el robot y el faro. Una vez que comienza la actuación del robot, se activa el proceso del módulo de rastreo para localizar el faro, una vez localizado actúan los módulos del brazo para tratar de afinar el efecto de apuntar al faro, y en seguida avanza el robot para tratar de alcanzarlo. El ciclo se repite mostrando el robot una conducta coherente de dirigirse al faro tratando de ubicarlo con el brazo también, como se muestra en la figura 4.3 b). Finalmente, después de la actuación cooperativa de los diferentes módulos, el robot alcanza una posición muy cercana al faro, con el brazo extendido y el sensor IR prácticamente tocandolo, figura 4.3 c).

El siguiente experimento se realizó colocando el faro en una posición, la cual es prácticamente la más baja que se puede lograr con él, sin interferir con su función. La figura 4.4 nos muestra la secuencia de actuación

a) b) c)

Figura 4.4. Tras el faro hacia abajo, a) posición inicial, b) acercándose c) faro alcanzado.

En una secuencia de imágenes similares a las del caso anterior, la figura 4.4 a) muestra al robot en su posición inicial, un poco hacia la izquierda del faro, próximo a iniciar el rastreo. La figura 4.4 b) muestra al robot en acción, ya habiendo localizado al faro y avanzando hacia él. Por último en la figura 4.4 c) podemos observar al robot con el brazo inclinado alcanzando su objetivo.

Intento	Tiempo primer rastreo [seg]	Tiempo total [min:seg]	Ciclos	Observaciones
1	36	01:52	9	Menor tiempo total y ciclos
2	38	02:12	13	
3	36	02:35	15	
4	35	02:46	19	
5	37	02:13	12	
6	36	02:33	18	
7	32	02:34	16	Mínimo tiempo de rastreo
8	35	02:45	18	
9	36	02:47	18	
10	34	02:24	15	
Promedios	**35.5**	**02:28**	**15.3**	

Tabla 4.1. Datos cuantitativos de una serie de diez pruebas.

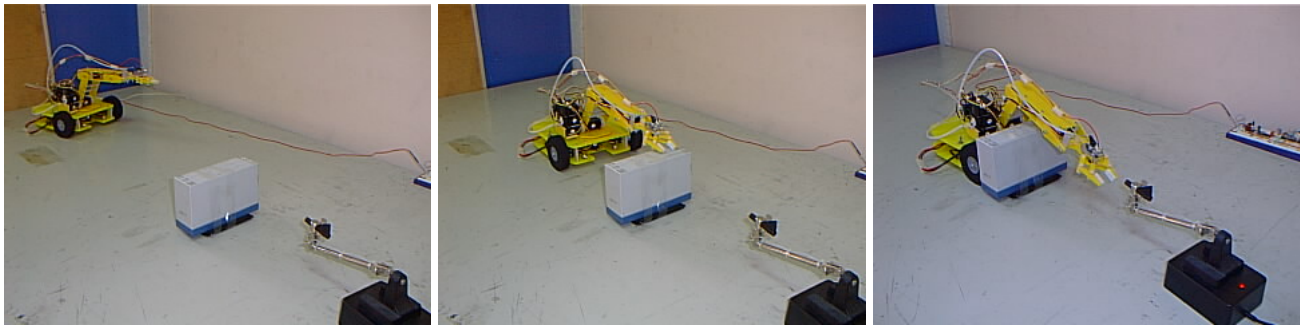
Con la finalidad de tener datos cuantitativos del desempeño del robot, hicimos una serie de diez pruebas consecutivas de localización y alcance del faro en las condiciones descritas en el primer experimento. Los resultados son los mostrados en la tabla 4.1. Observamos que el tiempo promedio que le toma al robot realizar el primer rastreo es de 35.5 segundos, el número de ciclos de la ejecución de la AMC es de 15.3 en promedio y el tiempo promedio que le toma al robot desde que inicia hasta que alcanza el faro, es de 2 minutos 28 segundos. Podemos apreciar cierta homogeneidad en el desempeño del robot, excepto en el primer intento, en el que rápidamente logró su objetivo. Otro punto que vale la pena destacar es que el desempeño del robot es confiable, debido a que prácticamente no se presentaron errores, a excepción de cuando se hacían pruebas con interferencia de luz solar, que eventualmente causaba problemas de localización del faro.

4.3. Robustez en la operación del robot.

Para poner a prueba la robustez del robot ante situaciones adversas, realizamos dos pruebas que consisten, una en poner un obstáculo al avance con las ruedas del robot y la otra en que una vez que el robot ha localizado el faro en una dirección, súbitamente cambiamos la posición del mismo, es decir, hay una modificación en el ambiente, con un cambio dinámico del objetivo.

4.3.1. Actitud robótica ante obstáculos en las ruedas.

Este experimento consiste en colocar un obstáculo al paso de las ruedas del robot, que no solo obstruye el acercamiento del robot al faro, sino que el obstáculo también bloquea parcialmente el haz de luz infrarroja hacia el robot. La secuencia de imágenes de la figura 4.5 muestra el comportamiento del robot ante dicho escenario.



a) b) c)

Figura 4.5. Actitud ante un obstáculo, a) inicio, b) librando obstáculo, c) alcanzando faro.

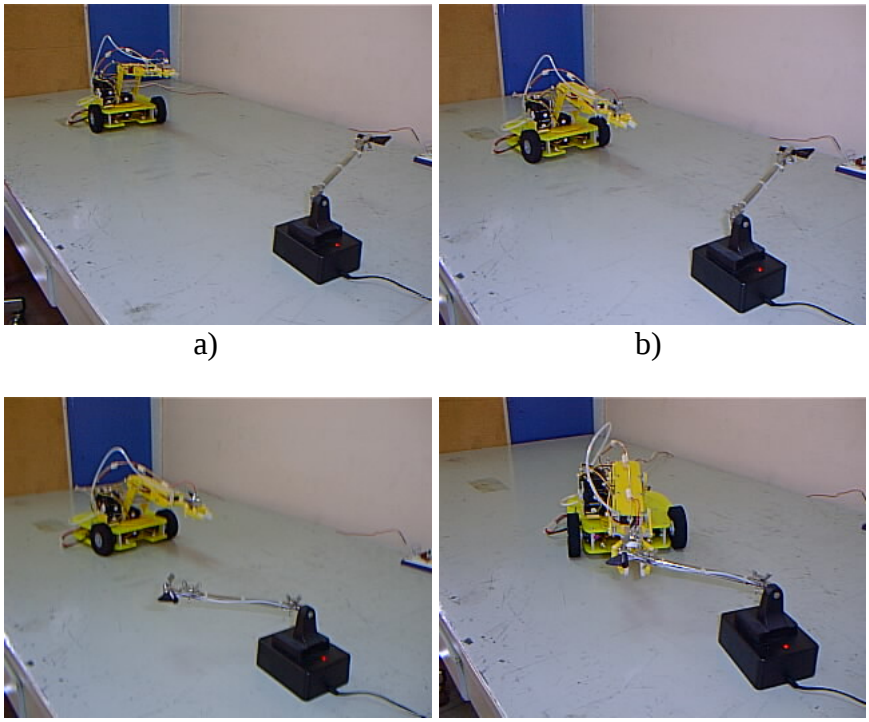
Como podemos apreciar en las imágenes de la figura 4.5, ésta es una prueba un tanto difícil, ya que la separación entre el robot y el faro es de aproximadamente 80cm, el robot apunta de inicio ligeramente hacia la derecha de su objetivo, el faro está en una posición muy baja y el obstáculo, además de bloquear el camino del robot hacia el faro, bloquea también parcialmente el haz de luz IR del faro. Estas condiciones son observables en la figura 4.5 a).

En la figura 4.5. b), es notable que el robot ha localizado el faro, avanza hacia él y apenas es suficiente el movimiento del brazo para librar la altura del obstáculo. En la figura 4.5 c), apreciamos que el avance con las ruedas ha sido bloqueado por el obstáculo y que la actuación de los módulos del brazo tratan aún de acercar la pinza al faro. Nótese que la morfología del brazo permite que aparte de librar exitosamente el obstáculo, el brazo se adapta a las condiciones y le es posible, todavía, alcanzar una posición muy por debajo del obstáculo, acercándose lo más posible al faro, es una actitud robótica emergente muy interesante.

4.3.2. Actitud robótica ante cambios dinámicos en el objetivo.

El escenario de este experimento se plantea de tal forma, que existen unas condiciones iniciales del ambiente entre el robot y el faro, mostradas en la figura 4.6. a), en donde el robot está frente al faro y éste está inclinado hacia su lado izquierdo.

La figura 4.6. b) muestra que el robot ha localizado el faro y se dirige hacia él; en la figura 4.6 c) observamos que hubo un cambio dinámico en la posición del faro, esto es, repentinamente movimos en forma manual el mismo y ahora se encuentra en una posición completamente opuesta a la que tenía inicialmente. A continuación el módulo de rastreo localiza la nueva posición del faro, el robot se enfila hacia él y finalmente la figura 4.6 d), muestra cómo el robot logra dar alcance exitosamente al faro.



a) b)

c) d)

Figura 4.6. Robot ante cambios dinámicos en el objetivo.

4.4. Actuación del robot cuando el algoritmo del “bandido” es invertido.

En este experimento hacemos leves modificaciones a la lógica de los procesos de algunos módulos, principalmente con aquellos que tienen que ver con el algoritmo del “bandido” y el movimiento de las ruedas.

La modificación que hicimos en los módulos que usan el algoritmo del “bandido” consiste básicamente en cambiar la lógica de emisión del signo del operador de recompensa (ver sección 2.2.3), ahora cuando se hace la diferencia entre el dato de *pos_actual* y el de *pos_anterior*, si el resultado es mayor que cero, se emite un signo (-), si el resultado es menor que cero, se emite un signo (+) y si el resultado es cero, no hay cambio, se emitirá un cero. Tendremos ahora en el operador de recompensa, una lógica de “mantente con el perdedor pero cambia con el ganador”.

En el módulo avanza, las ruedas harán el movimiento contrario cuando se ejecute el proceso del módulo, esto es, el robot retrocederá en vez de avanzar y se mantendrá retrocediendo mientras perciba que se está alejando del faro.

En el módulo gira, en lugar de que la base del robot tienda a alinearse con el sensor IR, hará lo contrario, es decir, se desalineará en un ángulo proporcional al que tenga el sensor IR respecto a su referencia.

La actitud presentada ahora por el robot, se muestra en la secuencia de imágenes de la figura 4.7.

a) b)

Figura 4.7. Actuación del robot con su lógica invertida.

La figura 4.7 a), muestra las condiciones iniciales del robot y del faro frente a frente, perfectamente alineados. La 4.7 b) muestra la nueva actitud del robot, que tiene una conducta de rehuir al faro, alejándose tanto con el brazo como con las ruedas. Es importante notar que el módulo de rastreo nunca pierde de vista el faro, lo que le permite al robot “saber” de qué huye.

CAPÍTULO 5. ANÁLISIS DE NUESTROS RESULTADOS Y DE TRABAJOS AFINES

El análisis de los resultados obtenidos de nuestro trabajo, nos indica hasta qué punto el desempeño del robot cumple con el objetivo principal planteado inicialmente, de efectuar la tarea de localización, acercamiento y alcance de un faro infrarrojo con la pinza aplicando la AMC; el comportamiento del robot ante ciertas variantes del ambiente y con modificaciones en el programa al aplicar el algoritmo del "bandido" invertido. Haremos también un análisis cualitativo para comparar nuestros resultados, con los del robot Herbert de Brooks y Conell [Conell 90], para establecer, guardando las proporciones del caso, similitudes y diferencias entre ambos sistemas.

5.1. Resultados experimentales.

El fototransistor y el LED infrarrojo que usamos para el robot SisNeρ, tienen cierto patrón de detección y emisión respectivamente (ver los anexos A2 y A3), además de un límite de detección en la distancia de separación entre ambos, también es importante su direccionalidad, es decir, la posición de encuentro de uno respecto al otro.

En cuanto a las pruebas realizadas al fototransistor del sensor IR, observamos en la respuesta un comportamiento bastante aceptable (ver figura 4.1), cuando la prueba se realizó en forma individual y considerando un entorno bidimensional, variando el ángulo de incidencia respecto al faro y la distancia al mismo. Cuando el fototransistor se monta en el mecanismo de escáner y luego sobre la pinza del brazo del robot, el entorno del fototransistor y del faro es más complicado, en tres dimensiones, ya que aparte de tener un ángulo de incidencia y una distancia variables entre los elementos (plano XY), también se tiene la altura en el eje Z; por lo que es importante considerar la direccionalidad al colocar el robot respecto al faro, para que al actuar el proceso del módulo de rastreo, puedan coincidir el patrón de detección del sensor IR con el patrón de emisión del faro, de lo contrario, será difícil localizar a este último.

Vale la pena mencionar que en un principio dudábamos que un sensor IR funcionara para nuestros propósitos, en la implementación de un método de sensado "monocular" para medir distancia en función de intensidad luminosa, sin embargo la adaptación del escáner en el sensor IR resultó de gran ayuda, ya que indirectamente permite ampliar el patrón de detección del sensor, hasta poco mas de 180 grados en el plano XY hacia el frente del robot y aproximadamente 90 grados en el eje Z, con el movimiento del brazo.

En cuanto a los experimentos de localización, acercamiento y alcance del faro con la pinza del robot, se prueban dos casos: primero con el faro extendido y luego colocado hacia abajo. Se observa que aún cuando inicialmente el robot no apunte hacia el faro, el proceso del módulo de rastreo es capaz de lograr hacer coincidir parcialmente el patrón de detección del sensor IR con el patrón de emisión del faro, lo que permite al sensor IR localizar el faro y que actúen luego los diferentes módulos de SisNeρ. En ambos casos, la actitud del robot en general muestra cómo las conductas individuales y cooperativas de los procesos de los módulos, llevan finalmente a la consecución del objetivo: alcanzar el faro con la pinza.

En los resultados mostrados en la tabla 4.1, de la realización consecutiva de diez repeticiones del caso del primer experimento de localización y alcance del faro, observamos que existe bastante uniformidad en la actuación del robot ya que en prácticamente todos los casos, los

tiempos y ciclos de ejecución fueron muy similares, excepto en el primer intento, en el que suponemos que fortuitamente, el robot alcanzó rápidamente su objetivo. Considerando que el robot no cuenta con ningún planificador, ni representación del ambiente ni prácticamente registro en memoria de su actuación (solo mantiene registro de la última acción de cada módulo), deducimos que el robot es suficientemente confiable en la realización de la tarea para la que fue diseñado.

En lo que respecta a la robustez del robot, era de esperarse que no presentara problemas ante cambios dinámicos del objetivo, cuando se mueve manualmente al faro de un lado a otro completamente opuesto, una vez que ya inicialmente el robot lo ha localizado y se dirige hacia él, luego corrige y se reorienta para, ahora si, alcanzar el objetivo; en el experimento en el que se coloca un obstáculo en las condiciones descritas en la sección 4.3.1, francamente se antojaba una prueba difícil para el robot, no obstante, su comportamiento emergente nos hizo notar un caso de disociación entre módulos (disociación de capas altas, por analogía modular con el SNC), cuando los módulos superiores de la AMC fueron obstruidos, es decir, quedaron bloqueados los movimientos de las ruedas y el resto de las conductas de los módulos (las que corresponden al brazo del robot) siguieron actuando para acercar, lo más posible, la pinza del robot al faro.

Por último, en el experimento de modificar la lógica del programa para obtener una actitud "invertida" del robot, haciendo cambios mínimos en tres funciones del programa (en algoritmo del “bandido”, avanza y gira), como se describe en la sección 4.4, observamos, como esperábamos, una actitud en la que el robot tiende a rehuir al faro, llegando en algunos casos hasta prácticamente “darle la espalda”. Esto siempre y cuando el proceso del módulo de rastreo no “pierda de vista” al faro (por cierto, el módulo de rastreo quedó intacto para este experimento), el cual sirve como referencia al robot para “saber” de qué huye. En algunos de los intentos que se hicieron para este experimento, en los que por la actuación de las conductas de los módulos, el robot “perdía de vista” al faro, se observa parcialmente la actitud de rehuir y luego que el módulo de rastreo ya no lo ubica, se realizan pocos movimientos al no haber punto de referencia del cual alejarse. Los resultados de este experimento permiten notar que, como en el SNC, nuestro robot también presenta características de plasticidad, entendida esta como la capacidad del SN de adaptarse a cambios en su estructura (por ejemplo la pérdida de un sentido).

5.2. Comparación cualitativa entre los robots SisNeρ y Herbert.

Herbert [7] es un robot construido en el Instituto Tecnológico de Massachussets (MIT, por sus siglas en inglés), como trabajo de tesis doctoral de Jonathan H Conell [Conell 90] (para más detalles de este robot, ver la descripción del anexo A5). El robot fue diseñado para realizar una tarea específica: deambular, identificar y levantar latas vacías, en un ambiente no estructurado. Usa la arquitectura de subsumción, manejando un mínimo de representación del ambiente, emplea diferentes sensores, alrededor de 40 conductas y 24 procesadores que operan en paralelo. Las características y resultados de los robots SisNeρ y Herbert, nos permiten hacer un ilustrativo ejercicio de comparación cualitativa entre ambos, enfatizando semejanzas y diferencias, obviamente, guardando las debidas proporciones entre uno y otro.

Entre las semejanzas de los robots, podemos mencionar que cada uno fue construido considerando un objetivo esencial específico, el SisNeρ para localización, acercamiento y alcance de un faro IR y Herbert para localización y levantamiento de latas vacías en un ambiente no estructurado. Si quisiéramos que estos robots operaran como recoge pelotas en una cancha de tenis ó como repartidor de cartas en una oficina, seguramente no sería fácil ni siquiera adaptarlos, por lo que para estos últimos propósitos, habría que diseñar robots apropiados que resultaran eficientes para esas tareas.

Ambos robots emplean arquitecturas modulares, el SisNeρ utiliza la AMC, mientras que Herbert usa la ASu.

En cuanto a representación del ambiente y registro de su actuación, ambos manejan lo mínimo, el SisNeρ no usa representación alguna y solo mantiene en memoria registro de la última acción de cada módulo de la AMC, por su parte Herbert utiliza su información sensorial como representación al momento del ambiente y no guarda en memoria información de trayectorias seguidas. Podemos decir que ambos robots se guían por un objetivo preconcebido y su desempeño se realiza de acuerdo a las condiciones del ambiente en que se desenvuelven.

Entre las diferencias notables, Herbert es un robot completamente autónomo en cuanto a fuente de energía mediante baterías en su interior, en sus funciones sensorimotoras y en el manejo de sus conductas con procesadores alambrados en el cuerpo del robot. El nuestro, aunque está energizado parcialmente con baterías y que podemos colocar el circuito del sensor IR sobre el robot, no sería autónomo, ya que por la simulación de los procesos, requerimos de una computadora externa, entonces de momento, no nos podemos librar del "cordón umbilical" que representan los cables de comunicación serial. Superar esto, para hacer autónomo al SisNeρ, sería un buen trabajo a futuro.

Otra diferencia entre los robots es la forma en que actúan las conductas, en el nuestro es de manera secuencial cooperativa mientras que en Herbert actúan en forma paralela.

[7] Se le dio este nombre en honor a Herbert Simon por el trabajo de la hormiga metafórica [Simon 69] en donde expone que la complejidad de la acción de una criatura no se debe a una profunda introspección cognitiva sino a la complejidad del ambiente en el que se desenvuelve. Esto y la navegación del caracol sirvieron de inspiración en la creación de Herbert.

Una diferencia más es la relativa complejidad en los sensores, Herbert usa alrededor de 40 sensores, entre cámara, láser, infrarrojos e interruptores. El SisNeρ solo tiene un arreglo de dos elementos, un emisor y un sensor, sin embargo hemos demostrado que es suficiente para lograr el objetivo establecido. No debemos soslayar que uno de los propósito principal de nuestro robot es probar que funciona una arquitectura modular de control.

Recapitulando, nos damos cuenta que hay muchas diferencias entre Herbert y SisNeρ, en cuanto a construcción y complejidad, no obstante en su estructura esencial son parecidos, ya que ambos emplean arquitecturas modulares de control. A manera de corolario, podemos decir que a pesar de la limitación en recursos, nuestra experiencia con el SisNeρ y los resultados obtenidos, nos dan la confianza de que podemos hacer cosas notables con los conocimientos e ideas (creatividad), que si están a nuestro alcance.

CAPÍTULO 6. CONCLUSIONES Y TRABAJO FUTURO

El sistema computacional-mecatrónico descrito en esta tesis, representa apenas un segundo paso en el camino para lograr un verdadero robot mecatrónico autónomo. Las experiencias logradas con nuestro sistema, nos hacen vislumbrar un panorama en el que hay un número importante de posibilidades, que llevan a pensar que a futuro, es factible escalar la arquitectura empleada a sistemas mas completos, como los que se mencionan a continuación; también puntualizaremos a manera de conclusiones, algunos de los aspectos más relevantes de este trabajo.

6.1. Etapa computacional y mecatrónica.

La etapa computo-mecatrónica abordada en la tesis, nos permitió crear un sistema (para el proyecto global SNCM de J. Negrete) que para nuestra sorpresa, terminó por generar un brazo articulado funcional, montado en un carro con ruedas, que acerca exitosamente su instrumento (sensor IR) a un emisor IR, como se mostró en los experimentos del capítulo 4.

6.2. Restricciones de la implementación.

La implementación y las características funcionales del robot, nos llevaron a construir tres tipos de módulos, en lugar de uno solo como se usó en la simulación, SNCM.1 [Negrete-Mtz 00]: la implementación “se vio forzada” a aumentar la heterogeneidad modular de la AMC.

6.3. Plasticidad de SisNeρ.

Nuestro robot SisNeρ, exhibe la misma plasticidad que el robot simulado SNCM.1, como pudimos observar en el experimento de la sección 4.4, en donde con pequeños cambios en la lógica del algoritmo, logramos cambiar el desempeño de las conductas y por ende la actitud global del robot, que en este caso tiende a rehuir del objetivo.

6.4. Compatibilidad de la arquitectura de SisNeρ.

La AMC del SisNeρ no es incompatible con la ASu de Brooks, ni con alguna otra selectora de conductas, ya que esta podría subsumir a otras no cooperativas.

6.5. Plataforma SNCM.2.

El robot SisNeρ, construido y empleado en la tesis, queda como una plataforma de trabajo experimental, para modificaciones a futuro de la arquitectura AMC ó algunas otras arquitecturas que se requieran evaluar.

6.6. Versión SNCM.3

Como trabajo a futuro (cercano), se planea implementar el SNCM.3, todavía en una versión computacional-mecatrónica, pero con una actuación distribuida. Esta versión de robot aparte de tener un objetivo específico, podría contar con más conductas, para evadir obstáculos, un sistema de visión, etc.

6.7. Versión SNCM.4.

Una vez implementado un robot con las características del punto anterior, se espera estar preparados para la etapa puramente mecatrónica, la versión SNCM.4, en la que cada módulo o conducta será un circuito puramente electrónico y en la que el conjunto de conductas actuaran en un auténtico sistema paralelo distribuido.

Esperamos que, por otra parte, el SNCM.4 nos pueda acercar a un mejor entendimiento de lo que vagamente llamamos fisiología del cerebro, la que en un futuro haremos equivalente a un cerebro mecatrónico.

APÉNDICES

A1. Referencias

[Arbib 64] Arbib, M.A. *Brains, Machines and Mathematics*, Mc Graw-Hill, New York, New York,1964.

[Arkin 93] Arkin, R.C., *Reactive Robotics Systems*, College of Computing, Georgia Institute of Tchnology, Atlanta Georgia, 1993.

[Asfall 92] Asfall, C.R., *Robots and Manufafturing Automation*, John Wiley & Sons, Inc., 1992.

[B&b 95] B&B Electronics, *RS-232 Data Adquisition Module 232SDA12 Instruction Manual*, Document No. 232SDA124597, 1995.

[Brailowsky 95] Brailowsky, Simón, *Las Sustancias de los Sueños*, Ed. Fondo de Cultura Económica , México, 1995.

[Brooks 85] Brooks, R.A., *A Robust Layered Control Systems for a Mobile Robot*, A.I. Memo 864, MIT Artificial Intelligence Laboratory, Sep. 1985.

[Brooks 90] Brooks, R.A., *Elephants Don't Play Chess*, Robotics and Autonomous Systems 6, 1990.

[Brooks 91] Brooks, R.A., *Intelligence without Reason*, in The Artificial Route to Artificial Intelligence, building embodied Situated Agents, Lawrence Erlbaum, 1991.

[Brooks 94] Brooks, R. A., *Building Brains for Bodies*, Autonomous Robots, Vol 1, No. 1, 1994.

[Brooks 97] Brooks, R.A., *From Earwings to Human, Robotics and Autonomous Systems*, Vol. 20, 1997.

[Churchland 92] Churchland, P.S., and Sejnowsky, T.J., *The Computational Brain*, MIT Press, Cambridge Massachusetts, 1992.

[Conell 90] Conell, Jonathan H., *Minimalistic Mobile Robotics, A Colony-style Architecture for an Artificial Creature*, Academic Press Inc, 1990.

[Damasio 00] Damasio, Antonio R, *Sentir lo que Sucede, Cuerpo y Emoción en la Fabrica de la Consciencia*, Editorial Andrés Bello, 2000.

[Darwin 1859] Darwin, Charles, *El Origen de las Especies*, Editores Unidos Mexicanos, S.A., 2001.

[Edwards 99] Edwards, S., *Mini SSC II Serial Servo Controller, User's Manual*, Fronage Road, Arizona, 1999.

[Everett 95] Everett, H.R., *Sensors for Mobile Robots, Theory and Applications*, AK Peters, Ltd, Wellesley, Massachusetts, 1995.

[Ewert 80] Ewert, J.O., *Neuroethology*, Springer – Verlang, 1980.

[Franco 88] Franco, Sergio, *Design with Operational Amplifiers and Analog Integrated Circuits*, Mc Graw-Hill, 1988.

[Holland 90] Holland, O. and Smith, S., *An Investigation of Two Mediation Strategies Suitable for Behavioral Control in Animals and Animats*, In J. A. Meyer & S. Wilson Eds. , From Animal to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior (pp. 255 – 262), Cambridge, MA, MIT Press, 1990.

[Kaelbling 93] Kaelbling, Leslie Pack, *Learning in Embedded Systems*, The MIT Press, Cambridge, MA, 1993.

[Jones 93] Jones, Josep and Flinn, Anita, *Mobile Robots, Inspiration to Implementation*, AK Peters, Wellesley, Massachusetts, 1993.

[Jung 91] Jung, Walter G., *IC Timer Cook Book*, SAMS, 1991.

[Leisle 90] Leise, Esther M., *Modular Construction of Nervous Systems: a Basic Principle of Design for Invertebrates and Vertebrates*, Brain Research Review 15, 1990.

[Levine 85] Levine, R.B. and Truman, J.W., *Dentritic reorganization of Abdominal Motoneurons during Metamorphosis of the Moth Manduca Sexta*, Journal of Nueuroscience 5, pp 2424-2431, 1985.

[Lynxmotion 00] Lynxmotion, Inc. *Mobile Base Kit for SAA-KT Manual*, Pekin, Il., 2000.

[Maes 90] Maes, Pattie, *How do the Right Thing*, in Connection Science J., special issue on Hibrid Systems, February 1990.

[Minsky 86] Minsky, Marvin, *The Society of Mind*, Simon and Schuster, 1986.

[Mountcastle 79] Mountcastle, V.B., *An Organizing Principle for Cerebral Function: The Unit Module and the Distributed System*, In Schmitt F.O. and Worden F.G.(Eds.), The Neurosciences Fourth Study Program, MIT, Cambridge, MA, pp. 21-42, 1979.

[Murphy 00] Murphy, Robin R., *Introduction to AI Robotics*, Bradford Book, MIT Press, Cambridge, Massachusetts, 2000.

[Negrete-Mtz. 98] Negrete-Martínez, José and Pensado, J.M.A., *"Attentively Situated Multi-Agency Robots"*, MIA, U.V., Iberamia, 1998.

[Negrete-Mtz. 00] Negrete-Martínez, José, *A Multi-Process Central Nervous System for a Robot*, ISRA 2000, Monterrey Nuevo Leon, México, 2000.

[Nilsson 98] Nilsson, Nils J., *Artificial Intelligence: a New Syntesis*, Standford University, Morgan Kaufmann Publishers, Inc., 1998.

[Peacock 97] Peacock, C., *http://www.senet.com.au , 1997.*

[Penrose 95] Penrose, Roger, *La Nueva Mente del Emperador*, Oxford University Press, Trad. Ed. Grijalbo Mondadori, 1995

[Prescott 99] Prescott, Tony J., Redgrave, Peter and Gurney, Kevin, *Layered Control Architectures in Robots and Vertebrates*, Adaptive Behavior, 7, 99-127, 1999.

[Rich & Knight 94] Rich, Elaine and Knight, Kevin, *Inteligencia Artificial*, McGraw-Hill, 1994.

[Rosenblatt 97] Rosenblatt, J., *DAMN: a Distributed Architecture for Mobile Navigation*, Journal of Experimental and Theoretical Artificial Intelligence, Vol. 9 No. 2/3, pp. 339-360, April-September, 1997.

[Russel 96] Russel, S. y Norving, P., *Inteligencia Artificial: Un Enfoque Moderno*, Prentice Hall 1996.

[Schildt 90] Schildt, Herbert, *Manual de Referencia de C*, Osborne Mc Graw-Hill, 1990.

[Simon 69] Simon, Herbert A., *the Sciences of the Artificial*, MIT Press, Cambridge MA, 1969.

[Steels 95] Steels, L. and Brooks, R.A., *The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents*,Lawrence Earlbaum Associates Hillsdale, New Jersey, 1995.

[Sutton 98] Sutton, Richard S., and Barto, Andrew G., *Reinforcement Learning*, The MIT Press, Cambridge Massachusetts, 1998.

A2. Especificaciones del SILONEX SLT-50HL

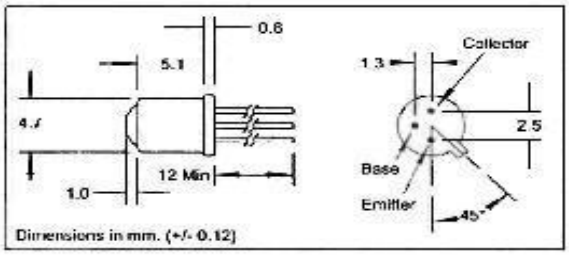
SILONEX® inc

SLT-50HL Series

NPN Phototransistor

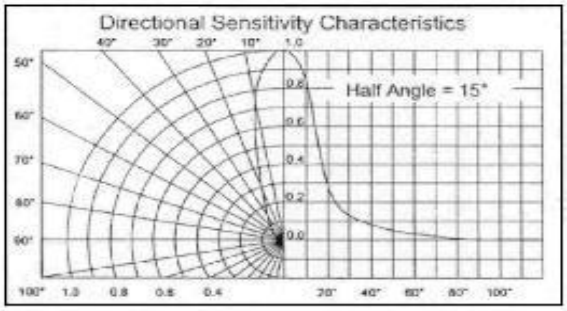
Features

- Narrow receiving angle
- Spectrally Matched to IRED
- NPN Planar Epitaxial Process
- Multiple Sensitivity Ranges
- Extended Temperature Range
- Hermetic TO-18 case with high dome lens



Description

The SLT-50HL series consists of an NPN silicon planar epitaxial phototransistor mounted in a hermetically sealed TO-18 dome lens package. The TO-18 hermetic package provides high reliability in hostile environments. The various sensitivity ranges available provide the desired output to meet multiple application demands.



Absolute Maximum Ratings

Storage Temperature Range	-65°C to +150°C
Operating Temperature Range	-55°C to +125°C
Soldering Temperature (1)	260°C
Power Dissipation @ 25°C (2)	250mW

Electrical Characteristics (T_A=25°C unless otherwise noted)

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
$I_{C(ON)}$	On-State Collector Current:					
	SLT-50HL1	0.6			mA	V_{CE}=5V, Ee=5mW/cm² (3)
	SLT-50HL2	1.2			mA	V_{CE}=5V, Ee=5mW/cm² (3)
	SLT-50HL3	2.4			mA	V_{CE}=5V, Ee=5mW/cm² (3)
	SLT-50HL4	4.0			mA	V_{CE}=5V, Ee=5mW/cm² (3)
	SLT-50HL5	6.0			mA	V_{CE}=5V, Ee=5mW/cm² (3)
	SLT-50HL6	12.0			mA	V_{CE}=5V, Ee=5mW/cm² (3)
I_{CEO}	Collector Dark Current			50	nA	V_{CE}=10V, Ee=0
BV_{CEO}	Collector-Emitter Breakdown Voltage	40			V	I_C=100µA, Ee=0
BV_{CBO}	Collector-Base Breakdown Voltage	60			V	I_C=100µA, Ee=0
BV_{ECO}	Emitter-Collector Breakdown Voltage	7.5			V	I_C=-100µA, Ee=0
$V_{CE(SAT)}$	Collector-Emitter Saturation Voltage			0.4	V	I_C=1.0mA, I_B=0.5mA, Ee=0
t_r, t_f	Rise Time, Fall Time		3		µS	R_L=100 Ω, I_C=800µA, V_{CC}=5V (4)
λ_P	Maximum Sensitivity Wavelength		930		nm	
λ_R	Sensitivity Spectral Range	400		1100	nm	
$\theta_{1/2}$	Acceptance Half Angle		15		deg	(off center-line)

Specifications subject to change without notice.

100224 REV 0

Notes: (1) >2 mm from case for <5 sec.
(2) derate @ 2.5mW/°C above 25°C.
(3) Ee = source @ 2854 °K
(4) Ee = source @ λ = 880 nm

5200 St. Patrick St., Montreal
Que., H4E 4N9, Canada
Tel: 514-768-8000
Fax: 514-768-8889

The Old Railway, Princes Street
Ulverston, Cumbria, LA12 7NQ, UK
Tel: 01 229 581 551
Fax: 01 229 581 554

QF-84

A3. Especificaciones del SILONEX SLED-56E2

SILONEX® inc

SLED-56E2

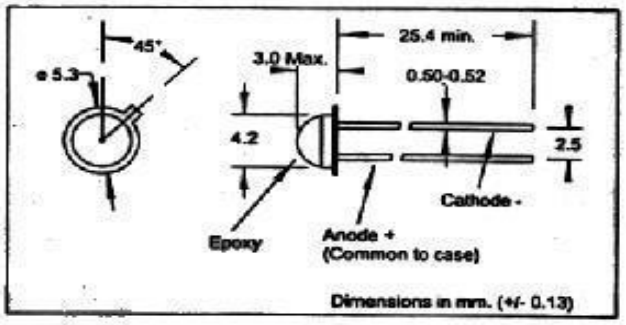
GaAlAs Infrared Emitting Diode

Features

- High output at low current
- Very wide emission angle
- Multiple power ranges
- TO-46 base package

Description

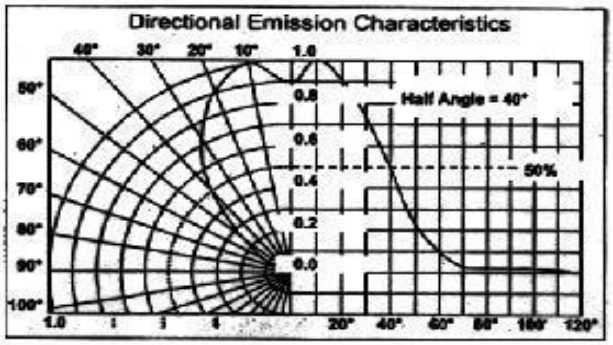
This Silonex device is a high output Gallium Aluminum Arsenide infrared emitting diode which produces a peak radiation at 880 nm when forward biased. It is packaged in a low profile clear epoxy dome for wide angle radiation emission.



Absolute Maximum Ratings

Storage Temperature	-40 to +85°C
Operating Temperature	-40 to +85°C
Soldering Temperature (1)	260°C
Average Forward Current	100 mA
Power Dissipation (2)	150 mW

Notes: (1) >2mm from case for <5 sec.
(2) derate 2.5 mW/°C above 25°C
(3) This is the average radiant intensity on a 0.250" diameter surface at a distance of 0.5" from the lens side of the tab to the sensing surface, forming a 30° cone.



Electrical Characteristics (T_A=25°C unless otherwise noted)

Symbol	Parameter	MIN	TYP	MAX	UNITS	TEST CONDITIONS
P_O	Output Power					
	SLED-56E2A		4		mW	I_F = 50 mA
	SLED-56E2B		6		mW	I_F = 50 mA
	SLED-56E2C		8		mW	I_F = 50 mA
	SLED-56E2D		10		mW	I_F = 50 mA
$Ee_{(APT)}$	Aperture Radiant Intensity					
	SLED-56E2A	0.75			mW/cm²	I_F = 50 mA, @ 30° (3)
	SLED-56E2B	1.0			mW/cm²	I_F = 50 mA, @ 30° (3)
	SLED-56E2C	1.5			mW/cm²	I_F = 50 mA, @ 30° (3)
	SLED-56E2D	2.0			mW/cm²	I_F = 50 mA, @ 30° (3)
λ_P	Peak Wavelength		880		nm	
λ_{BW}	Bandwidth		50		nm	
t_R, t_F	Rise Time, Fall Time		600		ns	I_F = 20 mA
V_F	Forward Voltage			1.6	V	I_F = 60 mA
V_{BR}	Reverse Breakdown Voltage	5	30		V	I_R = 10 µA
I_R	Reverse Current			10	µA	V = - 3.0 V
$\theta_{1/2}$	Half Power Point		40		deg	(off center-line)

Specifications subject to change without notice

103206 REV 1

5200 St. Patrick St., Montreal
Que., H4E 4N9, Canada
Tel: 514-768-8000
Fax: 514-768-8889

The Old Railway, Princes Street
Ulverston, Cumbria, LA12 7NQ, UK
Tel: 01 229 581 551
Fax: 01 229 581 554

QF-84

A4. Dimensiones Físicas del Robot

Plataforma o Base Móvil:

- Forma y Tamaño: Semicuadrada de 20 por 20 cm.
- Ruedas Motrices (2): 7.5 cm de diámetro por 1.5 cm de ancho.
- Rueda Loca: 3 cm de diámetro por 0.5 cm de ancho.
- Altura del Piso: 1.4 cm.
- Separación entre las Placas de Acrílico: 3.3 cm.

Brazo Robótico:

- Tamaño del Brazo Completamente Extendido: 33 cm.
- Tamaño del Brazo (entre Hombro y Codo): 9 cm de largo por 5 cm de ancho.
- Tamaño del Antebrazo (entre Codo y Muñeca): 9.5 cm de largo por 4 cm de ancho.
- Tamaño de la Muñeca: 4.5 cm por 5.5 cm de ancho.

Pinza:

- Tamaño de la Pinza Abierta: 8.5 cm de largo por 6.3 cm de ancho.
- Tamaño de la Pinza Cerrada: 9.5 cm de largo por 3.5 cm de ancho.
- Apertura Máxima de la Pinza: 2.3 cm.

A5. El Herbert de Brooks.

Herbert es un robot que fue construido en el Laboratorio de Inteligencia Artificial del MIT, bajo la dirección de Rodney A. Brooks, como trabajo de tesis doctoral de Jonathan H. Conell [Conell 90]. Es un robot de forma cilíndrica de 45cm de diámetro por 120cm de altura, como lo muestra la figura A5.1. Cuenta con una base de tres ruedas, un brazo robótico de dos grados de libertad mas la pinza, tiene como sensores de proximidad dos anillos de sensores infrarrojos, uno en la parte inferior con 16 sensores y otro en la parte superior con 14, cuenta además con un escáner láser y una camara CCD, ambos para identificación de latas y objetos.

El robot fue diseñado para realizar una tarea con un objetivo específico: deambular, identificar y levantar latas vacías de refresco en un ambiente no estructurado. Una de las premisas que se consideraron en el diseño, construcción y operación del robot, fue que debería manejar un mínimo de representación del ambiente así como un número mínimo de procesos ó conductas para lograr su objetivo, de ahí que a este tipo de robot le denominen "minimalista". En este caso el robot, emplea 40 conductas distribuidas en 24 procesadores que operan en paralelo.

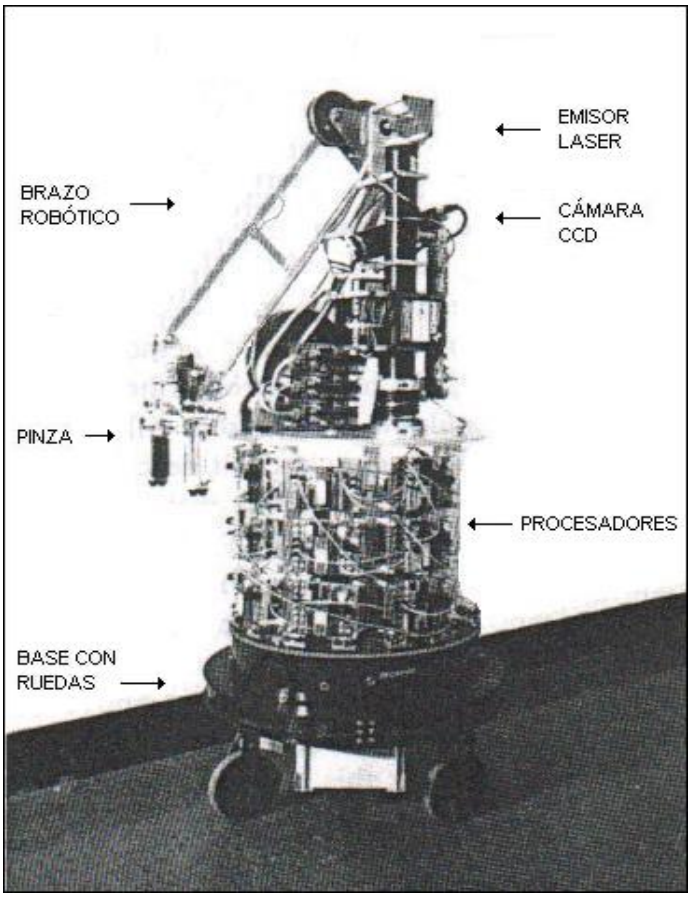


Figura A5.1. El robot Herbert.

Herbert usa un tipo de representación mínima, inspirada en la percepción de algunos animales, como las gaviotas, para la identificación de objetos y el caracol, para la navegación, lo que demuestra que es posible construir robots razonablemente competentes sin invertir mucho esfuerzo en construir representaciones detalladas. Usa ciertos patrones como agentes, cada uno de los cuales realiza una función específica para lograr conductas en paralelo, con un estilo reactivo de control.

La construcción del robot se hizo de forma incremental, agregando sucesivamente nuevos grupos de conductas encima de las ya existentes, empleando el principio de subsumción de agentes ó conductas independientes. Se implementó usando una arquitectura distribuida, empleando un mecanismo de arbitraje simple de prioridad de conductas.

Funcionamiento del robot:

Brazo manipulador: El brazo manipulador del robot fue diseñado para esta aplicación, buscando que fuera funcional y ligero. Cuenta con dos grados de libertad y funciona con el principio del paralelogramo, como los soportes de las lámparas de escritorio. Un extremo del brazo va fijo en la parte superior del cuerpo del robot y en el otro extremo tiene una pinza. Dicha pinza tiene una serie de sensores infrarrojos, de presión e interruptores. Es controlada en forma cooperativa por un conjunto de seis conductas independientes, que le hacen posible operar en un espacio rectangular vertical de trabajo (aproximadamente 30cm por 90 de altura), que es ampliado por el giro de la base del robot. Cuando el brazo del robot tiene a su alcance una lata, la información de los sensores y las conductas le permiten ser autosuficiente para ubicar la lata y tomarla evadiendo bordes y algunas imperfecciones del ambiente alrededor de ella.

Sistema de visión: Este sistema permite al robot tener un panorama hacia el frente de si mismo, con un ángulo de visión de 30 grados y una profundidad de 180cm. El sistema de visión habilita al robot para evitar obstáculos e identificar latas a lo lejos, para luego acercarse y dejar al brazo manipulador que realice la tarea de tomar la lata; así el proceso de localización y aprehensión de la lata es compartido entre el sistema de visión y el controlador del brazo, facilitándose el problema de coordinación ojo-mano. El sistema de visión está basado en las imágenes captadas con una cámara CCD cuando realiza un barrido con el emisor láser (construido por los autores para esta aplicación) sobre el volumen al frente del robot. Una vez obtenida la imagen es procesada para obtener los patrones de objetos que se encuentren en ella y luego con la referencia del patrón de la lata (una forma rectangular colocada de manera vertical), identifica si hay una lata u objetos que pudieran ser una lata. Con este proceso actúan varias conductas como acercamiento, alineación, avance, buscar, mirar, apuntar, etc. para llevar al robot hacia su objetivo, ampliando así el ángulo de visión a 60 grados.

Navegación: Para desplazar al robot en un ambiente no estructurado, es suficiente la información de los sensores infrarrojos y de una brújula, para aplicar una estrategia de “ir a tientas”. Esto permite al robot poder navegar siguiendo las paredes a determinada distancia, evitando obstáculos que pueda encontrar a su paso y obviamente, en paralelo, buscando latas para recogerlas. En la función de navegación actúan conductas como: detenido para colectar imágenes, caminar para comandar movimiento, permanecer indicando que no hay movimiento de

la base, vuelta para girar levemente, bloquear para determinar si el robot puede pasar en forma segura entre obstáculos, virar para girar el robot un cuarto de vuelta, etc.

Algunos aspectos importantes que hay que hacer notar en el robot Herbert son:

El sistema es controlado por un conjunto de conductas independientes, empleando una cantidad mínima de estados. Se aplica un esquema de toma de decisiones distribuida.

El robot usa varias representaciones espaciales, pero ninguna es completa ni detallada; no tiene una preconcepción del ambiente ni guarda la información de trayectorias seguidas y aún así es eficiente en su desempeño. Se usan procedimientos locales en vez de modelos detallados y persistentes, el robot debe de ser su propio “maestro” en cuestiones de aprendizaje, utilizando la información sensorial temporal generada.

Muchas de las conductas pueden ser usadas como una forma de representación y en lugar de acciones, como primitivas para planeación. Cada conducta puede ser considerada como un proceso, cuyo orden de invocación es determinado por el ambiente en el que se desenvuelve el robot.

Como podemos darnos cuenta, Herbert no es un robot muy sofisticado que hubiera usado el “estado del arte” de la robótica en su momento, mas bien es una prueba de que se puede construir un sistema moderadamente complejo, el cual funciona de una manera descentralizada y que no necesita de modelos explícitos del mundo para cumplir con su objetivo.

A6. Programa

```
      /* Nombre     : SisNeρ.C                                          */
      /* Objetivo   : Programa de localización de faro IR con           */
      /*              Arquitectura de Módulos Cooperativos para         */
      /*              controlar el robot SisNeρ Ver. SNCM.2             */
      /*                                                                */
      /* Escrito por    : JNM, RCE      Fecha :  8 de agosto 2001       */
      /* Modificado por : RCE           Fecha : 21 de agosto 2001       */
      /*                                                                */
      /*        Copyright 2001 MIA, UV                                  */

#include <dos.h>
#include <stdio.h>
#include <conio.h>

                      // ************** Declaración de Constantes ***************
#define PORT1 0x3F8     // Pto. Serial 1
#define PORT2 0x2F8     // Pto. Serial 2
#define paso 2          // Paso de avance de rastreo
#define paso_giro 20    // Paso de giro del robot sobre su eje
#define paso_avan 500   // Paso de avance
#define cintura 0       // Servo 0 = Cintura
#define hombro 1        // Servo 1 = Hombro
#define codo 2          // Servo 2 = Codo
#define muneca 3        // Servo 3 = Muneca
#define pinza 4         // Servo 4 = Pinza,     Servo 5 = Rueda Izq,
#define escaner 7       // Servo 6 = Rueda Der, Servo 7 = Scan
#define set_art 127     // Posición inicial de articulaciones
#define ruido 0         // Nivel de ruido para articulaciones
#define ruido_s 8       // Nivel de ruido para scanning
#define lim_frac 3      // Limite de fracasos (para evitar oscilación)
#define derecha 1       // Palabra para giro a la derecha
#define izquierda -1    // Palabra para giro a la izquierda

                     // ************** Declaración de Variables *****************
int kt, kt_1, ktmax, kt_gi, kt_av;  // Kt's para manejar intensidad de luz
int kt_cint, kt_homb, kt_codo, kt_mune; // Kt's para cada módulo
int act_1, act_cint, act_homb, act_codo, act_mune, act_gi, act_av;
int lim_inf, lim_sup;    // Limites de desplazammiento de articulaciones
int payoff, delta, frac; // Para operaciones de diferencia y conteo de fracasos
int pos_scan, pos_mune, pos_max; // Variables de posición para cada módulo
int pos_cint, pos_codo, pos_homb, pos_pinza;
int val;                 // Auxiliar para prevenir saturación

                      // ************** Declaración de Funciones **************
int odometro(void);             // Su operación se explica abajo en la
int signo(int numero);          // descripción de cada una de ellas.
void enviar(int servo, int pos);
void sal_dig(char n);
void v_satura(int medida);
void abrir_pto(int PORT);
void mueve(int artic, int pos );
void ruedas(int direccion, int tiempo);
void lims_art(int artic);
void gira(int pos_corr);
```

```
void algoritmo_avanza(int *kt, int *act);
void algoritmo(int servo, int *pos_servo, int *kt, int *act);
void rastreo(int servo, int *pos_art);

void main(void)   // ************** Programa Principal ***************
  {
    pos_scan = set_art; // Posicion de inicio del Escáner
    pos_mune = set_art; kt_mune = 0; act_mune = 1; // Inicializacion de
    pos_cint = set_art; kt_cint = 0; act_cint = 1; // de variables de
    pos_homb = set_art; kt_homb = 0; act_homb = 1; // los modulos
    pos_codo = set_art; kt_codo = 0; act_homb = 1;
    pos_pinza = 200;
    kt_gi = 0; act_gi = 1; kt_av = 0; act_av = 1; val = 0;

    abrir_pto(PORT1);             // Apertura de los puertos seriales
    abrir_pto(PORT2);
    sal_dig('0');                 // Asegura que los relevadores estén apagados
    mueve(pinza, pos_pinza);
    while (!kbhit())              // Ejecuta programa mientras no se
      {                           // presione teclado de la PC
        rastreo(escaner, &pos_scan);  // Activación de modulo de rastreo
        frac = 0;                 // Continua activación de los módulos
        do { algoritmo(cintura, &pos_cint, &kt_cint, &act_cint); }
          while( (payoff != 0 ) & (frac <= lim_frac) );
        frac = 0;
        do { algoritmo(hombro, &pos_homb, &kt_homb, &act_homb); }
          while( (payoff != 0 ) & (frac <= lim_frac) );
        frac = 0;
        do { algoritmo(codo, &pos_codo, &kt_codo, &act_codo); }
          while( (payoff != 0 ) & (frac <= lim_frac) );
        frac = 0;
        do { algoritmo(muneca, &pos_mune, &kt_mune, &act_mune); }
          while( (payoff != 0 ) & (frac <= lim_frac) );
        do{ algoritmo_avanza(&kt_av, &act_av); } while( payoff != 0 );
        gira(pos_scan);
      }
    sal_dig('0');                 // Asegura apagar los relevadores al salir
  }                 // ************** Termina Programa Principal **


  void abrir_pto(int PORT) // Función de apertura (definición) del puerto
    {
      outportb(PORT + 1 , 0);    // Deshabilita interrupciones en el puerto
      outportb(PORT + 3 , 0x80);
      outportb(PORT + 0 , 0x0C); // Definición del baud rate
      outportb(PORT + 1 , 0x00); // a 9600 bauds
      outportb(PORT + 3 , 0x03); // Palabra 8 bits, sin paridad, un bit paro
      outportb(PORT + 2 , 0xC7); // Control FIFO de registro
      outportb(PORT + 4 , 0x0B); // Habilita transmisión y recepción de datos
    }



  odometro(void)           // Funcion odómetro para medir intensidad de luz
    {                      // en el sensor infrarrojo del escáner.
      int c;
      int ch1, ch2;           // El voltaje obtenido proporcional a
```

```
    int voltaje;            // intensidad. Continua el protocolo de
    outportb(PORT1,'!');    // inicio de comunicación con el DAC
    outportb(PORT1,'0');    // Byte de dirección
    outportb(PORT1,'R');    // 1er. byte del comando
    outportb(PORT1,'A');    // 2o.  byte del comando
    outportb(PORT1,'0');    // Byte del canal
    do
      {
        c = inportb(PORT1 + 5); // Verifica si en el puerto hay un dato.
        delay(10);
      }
    while( !c & 1);
    ch1 = inportb(PORT1);     // Lectura de MSB del DAC
    ch2 = inportb(PORT1);      // Lectura de LSB del DAC
    voltaje = (256 * ch1) + ch2;  // Conversión hex a dec
    v_satura(voltaje);       // Verifica saturación del DAC
    return(voltaje);         // La función devuelve el valor
  }                          // de voltaje.


void enviar(int servo, int pos) // Protocolo de envío de posición a un servo
  {
    outport(PORT2, 255);           // Inicio de envío de datos.
    outport(PORT2, servo);         // El servo se moverá a la
    outport(PORT2, pos);           // posición indicada.
  }


void lims_art(int artic)   // Función define limite de articulaciones
  {
    switch(artic)             // dependiendo de articulación invocada
      {
        case 0: { lim_inf = 59; lim_sup = 191; break; }
        case 1: { lim_inf = 79; lim_sup = 191; break; }
        case 2: { lim_inf = 79; lim_sup = 191; break; }
        case 3: { lim_inf = 59; lim_sup = 191; break; }
        case 4: { lim_inf = 89; lim_sup = 201; break; }
        case 7: { lim_inf = 19; lim_sup = 221; break; }
        default: printf("\n Numero invalido...\n");
      }
  }


void mueve(int artic, int pos )  // Funcion de movimiento articulación
  {
    lims_art(artic);               // Obtiene limites para artic
    if((pos > lim_inf) & (pos < lim_sup)) // Continua si se
      {                                   // esta dentro de
        printf(" Servo: %d ", artic);     // los limites
        enviar(artic, pos);               // Envía posición a articulación
        delay (200);

      }
    else printf("\n Número invalido...\n");
  }
```

```
void ruedas( int direccion, int tiempo )  // Función de movimiento
  {                                       // de las ruedas dando
    int servo_5, servo_6;                 // dirección y tiempo de mov.
    if(direccion < 0) { servo_5 = 120; servo_6 = 132; printf(" Giro a Izq"); }
    if(direccion > 0) { servo_5 = 132; servo_6 = 120; printf(" Giro a Der"); }
    if(direccion == 0) { servo_5 = 133; servo_6 = 133; printf( " Avanzando"); }
    printf("\n");
    enviar(5, servo_5);   // Envía a servo 5 velocidad servo_5
    enviar(6, servo_6);   // Envía a servo 6 velocidad servo_6
    delay(tiempo);        // Duración del movimiento
    enviar(5, 127);       // Parar el servo 5
    enviar(6, 127);       // Parar el servo 6

    delay(1000);
  }


signo(int numero)        // Función para obtener signo de la recompensa
  {
    if(numero > ruido) return 1;   // considerando ruido, devuelve
    if(numero < -ruido) return -1; // +1 si numero es positivo y
    else return 0;                 // -1 si es negativo. Cero en
  }                                // otro caso


void sal_dig(char n)     // Función de manejo de las salidas digitales
  {
    outportb(PORT1,'!');  // Inicio de comunicación con el DAC
    outportb(PORT1,'0');  // Byte de dirección
    outportb(PORT1,'S');  // 1er. byte del comando
    outportb(PORT1,'O');  // 2o.  byte del comando
    outportb(PORT1,n);    // Pone salidas digitales en 0 o 1 lógico
    printf("\n Salida Digital: %c \n", n);
    delay(1000);

  }


void v_satura(int medida) // Función de verificación de saturación
  {                          // si el valor dado es mayor que 4050
    if( (val == 0) & (medida > 4050) ) { val = 1; sal_dig('1'); goto fin; }
    if( (val == 1) & (medida > 4060) ) { val = 2; sal_dig('2'); }
    fin:                     // habilitando salidas digitales 0 o 1
  }                          // para cambiar ganancia del amplificador


void algoritmo(int servo, int *pos_servo, int *kt, int *act)
  {                       // Algoritmo del "bandido"
    kt_1 = *kt;             // Intensidad y acción anteriores
    act_1 = *act;
    *kt = odometro();
    printf(" Servo: %d  kt_1: %d  kt: %d ", servo, kt_1, *kt);
    payoff = signo(*kt - kt_1);
    delta = *kt - kt_1;
    printf(" delta: %d  act_1: %d  Payoff: %d \n ", delta, act_1, payoff);
                            // Calculo de la nueva act con la tabla
    if( payoff > 0 ) { if( act_1 > 0 ) *act =  1; }
```

```
    if( payoff > 0 ) { if( act_1 < 0 ) *act = -1; }
    if( payoff < 0 ) { if( act_1 > 0 ) *act = -1; frac++; }
    if( payoff < 0 ) { if( act_1 < 0 ) *act =  1; frac++; }
    printf(" Act: %d \n ", *act);            // Nueva acción
    *pos_servo = *pos_servo + (*act * paso); // Nueva posición
                                             // Normalización de la posición
    lims_art(servo);
    if( *pos_servo > lim_sup ) *pos_servo = lim_sup;
    if( *pos_servo < lim_inf ) *pos_servo = lim_inf;
    mueve(servo, *pos_servo);   // Movimiento del servo o articulación
  }


void gira(int pos_corr) // Funcion para giro del robot
  {                     // que lo alinea con el ángulo del sensor IR
    int eo, ec, giro_der = 1, giro_izq = -1;
    eo = pos_corr - set_art;
    ec = eo * paso_giro;
    if( eo > 0 ) ruedas( giro_der, ec); // Giro del robot a la der
    if( eo < 0 ) ruedas( giro_izq, -ec);
  }


void algoritmo_avanza(int *kt, int *act)
  {                     // Algoritmo del "bandido" modificado
    kt_1 = *kt;            // Intensidad y acción anteriores
    act_1 = *act;
    *kt = odometro();
    printf(" kt_1: %d  kt: %d ", kt_1, *kt);
    payoff = signo(*kt - kt_1);
    delta = *kt - kt_1;
    printf(" delta: %d  act_1: %d  Payoff: %d  ", delta, act_1, payoff);
                          // Calculo de la nueva act
    if( payoff > 0 ) { if( act_1 > 0 ) *act =  1; }
    if( payoff <= 0 ) { payoff = 0; goto para; }
    printf(" Act: %d \n ", *act); // Nueva acción
    ruedas(*act - 1, paso_avan);  // Avance
    para:
  }


void rastreo(int servo, int *pos_art)  // Función de Rastreo
  {                                      // Comienza a rastrear el faro
    int i, step, aux, aux1, ref;         // con un ángulo pequeño y lo
    step = 10; ref = *pos_art;           // aumenta gradualmente hasta
    for(i = 1; i < 10; i++)              // encontrar el faro. Una vez
      {                                  // que lo encuentra se detiene
        *pos_art = ref - i * step;       // y habilita al siguiente
        ktmax = odometro();              // módulo.
        mueve(servo, *pos_art);
        printf(" Movimiento a Izq, Pos: %d \n", *pos_art);
        do
          {
            *pos_art = *pos_art + paso;  // Mueve incrementando posición
            mueve(servo, *pos_art);
            kt = odometro();
            printf("Pos: %d kt: %d ktmax: %d \n", *pos_art, kt, ktmax);
```

```
          if( kt > ktmax )             // Busca máximo
            {
              if( (kt - ktmax) > ruido_s )
                {
                  ktmax = kt;
                  pos_max = *pos_art;
                  *pos_art = *pos_art + paso;
                  mueve(servo, *pos_art);
                  kt = odometro();
                  printf("Pos: %d kt: %d ktmax: %d \n", *pos_art, kt, ktmax);
                  if( (ktmax - kt) > ruido_s ) printf("\n Maximo Global: %d ",
                       ktmax); goto termina;
                 }
              }
           }
        while( *pos_art < (ref + 2*i*step) ); // Llegada a limite superior
     }                                        // sin hallar máximo global
  for(aux1 = *pos_art; aux1 > pos_max; aux1 = aux1 - paso )
    {                                  // Mueve ahora a posición donde
      mueve(servo, aux1);              // se encontró máximo por
    }                                  // encima del ruido
  *pos_art = pos_max - (2 * paso);
  mueve(servo, *pos_art);
  termina:
}
```