UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Generación de la locomoción de un robot hexápodo LEGO Mindstorms por medio de osciladores neuronales inspirados en redes neuronales biológicas**.**

T E S I S

Que para obtener el título de:

Ingeniero Eléctrico Electrónico.

P RE S E N T A N

Yuri Sebastián Martínez

Nicolás Vicente Flores

Director de tesis:

Dr. José Ismael Espinosa Espinosa

RECONOCIMIENTOS.

Todo me fue dado aquí:
Mi abanico, mi plumaje de quetzal, los perfumes,
Mi curvo cayado, mi florón de papel,
En la casa de los musgos acuáticos,
En la casa de la luz.
Cuantos son tus flores,
Cuantos son tus cantos.
Con ellas deleito en tiempo de lluvia.
No soy mas que un cantor:
Flor es mi corazón:
Ofrezco mi canto.

Fragmento. Poemas de Yoyontzin.[*]

En la elaboración del presente trabajo, se conjugan sinnúmero de esfuerzos de toda índole. No es solamente una labor personal, pues ésta, es quizá la punta del *iceberg* de una serie de lazos y esperanzas en que se ha desarrollado. Y aquí, es el momento de dilucidar, palpar, reconocer esas grandes contribuciones que sin ellas hubiese sido imposible el principio del placer.

Que mejor espacio sino el que brindó la Universidad Nacional Autónoma de México, alma *mater*, que además de formarme en mi espíritu profesional, dio cobijo al alma en esa búsqueda incansable e interminable del saber. A la Facultad de Ingeniería, soto de encuentros y desencuentros, al cual representaremos *ad infinitum*, y con gran orgullo. Entre sus espacios, rincones, silencios, palabras, aún resuellan las voces de un ayer cargado de vicisitudes y nostalgias, de ecuaciones, de discusiones infinitas entre mis invaluables profesores. Para un gran maestro y amigo, que en algún lugar del espacio humano ha de encontrarse, el Lic. Fausto Hernández, que con su sensibilidad humanística, supo transmitir esa metamorfosis del hombre, esa arqueología del saber. A ellos, que con su espíritu humano y académico, forjaron estos días que se simbolizan en esta tesis.

A la Facultad de ciencias y en especial al Laboratorio de Cibernética, que a través de ella y con los recursos humanos disponibles, estuvo abierta en la mejor forma a colaborar en la realización de las tareas solicitadas.

Para el Dr. José Ismael Espinosa Espinosa, Coordinador del Laboratorio de Cibernética, mi más profundo agradecimiento, por haber colaborado con su tiempo y atención, con sus palabras e impresiones, que hicieron flaquear nuestras dudas. Por su valiosa disposición en dirigir este trabajo y más aún, su sensible conocimiento puesta a

*. Garibay, K., A. M. ***Poesia Náhuatl***. Vol. II. UNAM. México, 2000.

nuestra disposición. En el término de estos resultados, quedan no solo la huella académica lograda, sino también los avatares de un afecto y amistad enormes.

Para mi amigo y compañero de tesis, Yuri Sebastián Martínez, por su dedicación, entrega, y su disposición a enfrentar las dificultades que se presentaron a lo largo de la elaboración del trabajo. En el matiz de discusiones e inquietudes, quedan plasmados los días de dudas y aciertos, los anecdóticos; los cuales amalgamaron y maduraron los resultados obtenidos, y aquí conlleva ese espíritu de compañerismo y amistad.

A mi madre Pascuala Flores Flores, dedico este pequeño trabajo y comparto mi felicidad. Pues ella ha sido un pilar en mi formación profesional. En su espíritu de profunda humanidad cinceló en mí la esperanza, la búsqueda, ese impulso vital que da uno de sus primeros pasos. A mi Padre Mario Vicente del Ángel, orgullosamente dedico esta tesis. Agradezco profundamente su amor y esperanza, su temple, sus alegrías y palabras que son invaluables en los días aciagos. A ambos, por creer en mí y haber estado en el momento oportuno; gracias infinitamente y el logro también es de ellos.

A mis hermanos y hermanas, Edith, Isabel, Pedro, Brígida, Carmen y claro, José,. Aquí también se plasma su letra, acá su cariño y sus miradas, sus palabras; por haber estado conmigo siempre, dispuestos a apoyarme, a escucharme en la magnitud necesaria. Les comparto esta emoción de dar un pequeño paso en mi formación profesional, puesto que el esfuerzo es mutuo.

No quedan fuera de este logro mis amigos, amigas, que han llenado los días con su compañía, sus recomendaciones, sus discusiones y todo un cúmulo de instantes que se guardan en los pasillos de esta hermosa Universidad. A Claudia que vio el discurrir de los días con sus claro-oscuros de la letra aquí allanada, y estuvo siempre a mi lado ofreciéndome su apoyo; a Pablo, heraldo báquico, mítico en la embriaguez: del saber. A todos ellos, muchas gracias.

México, D. F. Ciudad Universitaria, agosto de 2003.

Nicolás Vicente Flores.

AGRADECIMIENTOS

A MI MADRE

Profra. Ernestina Martínez Vargas.

Por haberme inculcado con cariño su visión de la vida, que me ha permitido alcanzar metas. Gracias Mamá, por tu paciencia, apoyo y sabiduría en los momentos difíciles que hemos pasado juntos.

A MI PADRE

Prof. Florentino Sebastián Avendaño.

Por su apoyo y consejos

Al DIRECTOR DE TESIS

DR. José Ismael Espinosa Espinosa:

Por haber dirigido y apoyado este trabajo, además de habernos impulsado a incursionar en áreas nuevas e interesantes. Y compartir sus experiencias que nos enriquecen como personas.

A MI FAMILIA.

A los que están y los que se fueron. Que con su cariño, consejos y ejemplo me impulsaron a lograr mis objetivos y saber lo importante que es la familia.

A YENNI:

Por brindarme lo mejor de si y crecer juntos.

A MIS AMIGOS:

Por ser parte de mi historia y ayudarme a construirla.

A NICOLÁS:

Por trabajar conmigo en este proyecto que nos trajo satisfacciones, a pesar de las dificultades presentadas.

A LOS COMPAÑEROS DEL LABORATORIO DE CIBERNÉTICA:

Por su amistad y conocimientos.

A MI UNIVERSIDAD Y MI FACULTAD

A todos aquellos profesoras y profesores que integran esta hermosa Universidad y sobre todo a los de la Facultad de Ingeniería que con su labor y dedicación dieron lo mejor de sí, permitiéndome entender la importancia del saber y el comprender sucesos de la vida.

Ciudad Universitaria a 22 de agosto del 2003

Yuri Sebastián Martínez

ÍNDICE

CAPITULO IV

OSCILADORES GENERADOS CON NEURORED

CAPITULO V

PROGRAMACIÓN DE LOS OSCILADORES

CAPITULO VI

RESULTADOS

CAPITULO VII

APÉNDICE A.

SIMULADOR DE REDES NEURONALES

APÉNDICE B.

AMBIENTES DE PROGRAMACIÓN DEL RCX

V. Sensores

OBJETIVOS

-Diseñar y construir un robot insecto, que tenga seis patas —hexápodo—, para la locomoción, a partir de un robot LEGO MINDSTORM.

-Aplicar una red neuronal biológica funcionando como un oscilador, para el control de la locomoción del robot hexápodo. El robot únicamente moverá las articulaciones de las patas y será capaz de coordinarlas. Los movimientos serán los más básicos, tales como avanzar, retroceder y girar en cierta dirección.

- Lograr la imitación de un sistema de locomoción biológico, su implementación y medios de programación del robot, como premisas torales del presente trabajo.

INTRODUCCIÓN

El propósito de esta tesis está enfocado a buscar alternativas en la locomoción de robots con patas. Es un acercamiento a los modelos basados en sistemas de movimiento biológicos, alimentado en el estudio fisiológico de los vertebrados e invertebrados, especialmente entre los artrópodos, cuyos resultados han demostrado la capacidad de adaptabilidad de este tipo de organismos en ambientes inestables. Los artrópodos, tales como arañas, cucarachas, cangrejos, grillos, etc., y muchos insectos y animales de mayor tamaño, presentan patas; inclusive el 90% de las especies existentes sobre la tierra, son de este tipo. Esto genera una gran capacidad de movimiento en un mundo naturalmente hostil, tanto en su composición superficial como en las especies depredadoras.

El número de patas, dimensiones y coordinación, dependen de la especie y las necesidades de sobrevivencia en los cuales la selección natural los ha colocado. En ese sentido, la misma naturaleza traduce la importancia de los artrópodos y su alto grado de adaptabilidad.

Imitar los sistemas naturales en los modelos de comportamiento artificial, como el movimiento de los robots y en especial los robot con patas, es una tarea que involucra diversas ciencias, tales como las ingenierías, las matemáticas y por supuesto las ciencias biológicas. En los inicios sobre locomoción de robots, los prototipos se basaron en el sistema de ruedas; sin embargo, estos se han visto limitados en aplicaciones más delicadas y en relieves abruptos. Basta mencionar las dificultades que se presentaron en la reciente exploración del planeta Marte mediante el *Pathfinder*, un robot autónomo con sistema de locomoción basado en ruedas.

Es aquí, donde resulta relevante el análisis de sistemas de locomoción en los cuales involucren modelos capaces de adaptarse a los sistemas reales, inesperados y variados. Los robots con patas han dado la esperanza de resolver este tipo de dificultades, debido a que imitan los patrones de locomoción de sistemas naturales.

Este análisis se ha hecho desde distintas perspectivas, uno de ellos es la inteligencia artificial y sobre todo en un campo que estuvo en las fronteras del olvido por la ingeniería, las neurociencias o los sistemas de conexión paralela comúnmente conocidas como redes neuronales. El desarrollo del conocimiento fisiológico y neurológico de los sistemas nerviosos en los animales ha llevado a imitar respuestas de las neuronas biológicas en modelos artificiales. De esta manera, el desglose se ha extendido en diversos campos de investigación y aplicación que muchos de ellos resultan cotidianos, hasta ser desapercibidos.

El modelo básico de red neuronal es el oscilador, inspirado en una conexión neuronal oscilatoria de un sistema biológico, resultado de un conjunto de neuronas activándose en un patrón dado de manera oscilatoria. Mover un músculo en un sistema biológico, es lograr que un conjunto de neuronas se activen o desactiven mutuamente y oscilen de acuerdo a un determinado esquema de tren de pulsos.

La utilización de redes neuronales en este proyecto no implica el aprendizaje o un acercamiento de un sistema inteligente, sino la ejecución de una tarea básica como es la locomoción mediante neuro-osciladores. El aprendizaje es un elemento relevante, sin embargo queda fuera del alcance de esta investigación dada su complejidad y los medios desde los cuales se está comenzando. Queda abierta el aprendizaje como una segunda etapa de investigación.

El comportamiento biológico basado en estudios fisiológicos en insectos se imita a través de un simulador neuronal que define parámetros biológicos en su modelado. El robot construido se encuentra en el Laboratorio de Cibernética de la Facultad de Ciencias de la Universidad Nacional Autónoma de México. Las pruebas llevadas a cabo fueron variadas en todo el desarrollo del proyecto, sin embargo, se cuenta con un video que documenta y muestra los resultados obtenidos con el robot.

CAPITULO I.

METODOLOGÍA

1.1 METODOLOGÍA

Hablar de la metodología llevada a cabo en esta tesis, es dilucidar la estructura de cada uno de los capítulos en sus rasgos generales y el orden que guardan entre ellos en el desarrollo de esta investigación. Palpar las relaciones intrínsecas tejidos entre los diversos puntos y matices de esta investigación es el primer paso de un recorrido fugaz, efímero y con carácter introductorio. Desde luego, el primer enfoque ha sido la investigación documental, que define las bases y el entorno sobre el cual ha de estructurarse esta tesis. Busca separar dentro del infinito orden de posibilidades el cuerpo aprensible del cual ha de conformarse. Es necesario, en este sentido, buscar el respaldo biológico del tema; ubicar la importancia que guarda una investigación sobre locomoción y sus alternativas, así como su aplicación sobre la robótica.

Desde el capitulo uno, se da un desglose de los sistemas de locomoción y las investigaciones realizadas en un margen histórico, con el fin de documentar los esfuerzos y las alternativas logradas y observadas en modelos de locomoción. Buscando conformar un entendimiento real y un respaldo generado en las últimas investigaciones sobre sistemas de locomoción en insectos y diversos animales, se hace mención de las características fisiológicas y comportamientos neuronales de movimientos y actividades musculares. La actividad neuromotora en los sistemas nerviosos de distintos animales e insectos y las características neuronales más simples. Luego es necesario mantener con brevedad las actividades y comportamientos básicos de neuronas biológicas para comprender el modelado matemático y artificial de ellas.

En este punto se desdoblan los modelos matemáticos de sistemas neuronales que permiten la simulación y representación de procesos biológicos, similares a los que la naturaleza ha desarrollado. El énfasis en las características oscilatorias de ciertos grupos neuronales y sus actividades en la realización de diversas tareas de sobrevivencia en los seres vivos, principalmente en los insectos, es una de las columnas vertebrales de esta investigación y el fuerte respaldo considerado en esta tesis.

Desde los modelos inspirados en los sistemas biológicos, persisten las aseveraciones y acercamientos que tiene nuestro proyecto con los variados campos de investigación que se realizan en forma paralela a dicha tesis. Así, recorremos ejemplos de robots con patas que asemejen e imiten sistemas biológicos de pasos, basados en patrones de oscilación. Uno de los

trabajos más destacados y con una semejanza notable a la presente investigación es el desarrollado por Randal Beer y otros investigadores en el Massachussets Institute of Tecnology (MIT). Este consiste de un robot hexápodo de 50 cm de largo por 30 cm de ancho con un peso aproximado de 1 Kg ; cada una de las seis patas del robot tiene dos grados de libertad que son controlados independientemente por motores de dc con una potencia de 2 Watts. Los movimientos y el software de control, están basados sobre una red neuronal (figura PAG 295 R. *BEER. Robustness of a Distributed Neural Network*…), con pesos de conexión entre las neuronas definidas por el valor de entrada de las neuronas.

Así, esta investigación alimentada desde distintas disciplinas busca lograr un sistema de locomoción inspirado en la naturaleza, pero aplicado a un robot LEGO de limitadas capacidades de almacenamiento y control, dados los cánones de fabricación y comercialización. Se ha de contar con algunos sensores de toque y luz, que se extienden en el capitulo dedicado al diseño y características del Kit LEGO. Con todo esto se construye un robot hexápodo con las piezas armables del *kit* LEGO, haciendo hincapié en que no se han introducido componentes adicionales. Tal robot controlará seis patas con seis motores, una por cada pata. Este robot se adecua al mejor dinamismo que haya podido lograrse en esta investigación, desechando distintos modelos por sus desventajas mecánicas y de agilidad, así como el control sobre sus motores y en consecuencia su bajo rendimiento de potencia. Los distintos modelos probados se incluyen como evolución en las distintas etapas de armado.

Hacer un sistema de locomoción para un robot hexápodo invita a cuestionar una tarea de alternativa de movimiento y ventajas sobre otros sistemas. El de esta investigación, como ya se ha mencionado, desborda en la frontera ya construida por la naturaleza; sin embargo, es necesario imitarla y el método o modelo sobre el cual ha de manifestarse. La simulación de tales tareas de locomoción, ha sido el cariz conformador de otra vertiente, y de algún modo, la relación formal con la simulación neuronal. NEURORED, es un simulador que realiza esta tarea, considerado como fuente de resultados y alternativas de oscilación.

Este simulador de gran utilidad, difiere con el robot debido al ambiente de programación en el que se establece. Las capacidades en memoria y almacenamiento del cerebro del robot son otra limitante. NEURORED simulará esta parte de los osciladores pero será una herramienta primordial para buscarse las respuestas de diferentes arreglos de osciladores. Finalmente a través de un lenguaje de programación compatible con el cerebro del robot —RCX—, se tendrán las instrucciones para que los motores actúen de acuerdo al sistema de oscilación óptima, en cada una de las tareas solicitadas.

CAPITULO II

REDES NEURONALES Y ROBOTS MÓVILES

2.1. OSCILADORES BIOLÓGICOS

2.1.1. NEURONA BIOLÓGICA

Una neurona biológica es una célula del sistema nerviosos central que procesa información procedente de otras neuronas. Esta compuesta de un cuerpo o soma y de dos tipos de ramificaciones: el axón y las dendritas, figura 1. En el soma se localizan: el núcleo, que contiene la información hereditaria, y el plasma, que es la maquinaria molecular que produce los elementos necesarios para la vida de la célula.

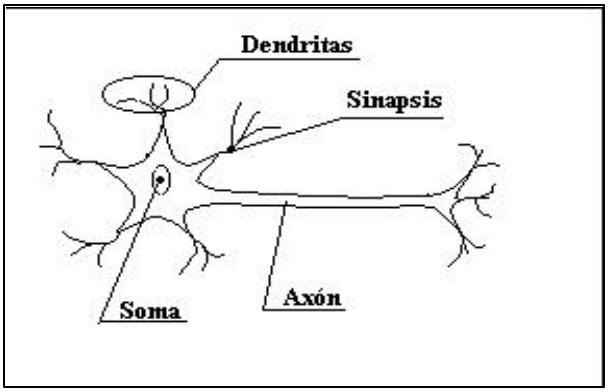


Figura 1. Neurona biológica. Tomado de Santos, S. [30]

En particular las dendritas aparecen como extensiones sumamente ramificadas en el cuerpo de las neuronas y es a través de ellas donde se recibe información de otras. La neurona emite impulsos de actividad eléctrica a lo largo de la estructura llamada axón, que puede ser muy larga y se escinde en millares de bifurcaciones. Todas las neuronas conducen la información de manera similar. En la extremidad de cada rama, se presenta una estructura llamada sinapsis; estructura elemental de pseudocontacto, “sin contacto físico”, entre el axón y la dendrita de dos neuronas que convierte la actividad procedente del axón en efectos eléctricos que inhiben o provocan actividad en las neuronas a las que está conectado. Cuando un impulso llega a la terminal de la sinapsis se libera una sustancia química llamada neurotransmisor. Al alcanzar suficiente intensidad las señales excitadoras que una neurona recibe frente a las señales inhibidoras, la neurona envía a lo largo de su axón un breve pulso de actividad eléctrica: los potenciales de acción que alcanzan una amplitud máxima de 100 milivolts y duran un

milisegundo. Son resultado del desplazamiento a través de la membrana celular de iones sodio, dotados de carga positiva, que pasan desde el fluido extracelular hasta el citoplasma intracelular.

Los potenciales de acción no pueden saltar de una célula a otra, esta comunicación es realizada por los neurotransmisores que se alojan en diminutas vesículas, y posteriormente vertidas en hendiduras de unos 20 nanómetros de anchura que separa la membrana *presináptica* de la *posináptica*. En la máxima intensidad del potencial de acción, penetran iones de sodio en la terminal nerviosa y su movimiento constituye la señal determinante de la excitación sincronizada (la liberación coordinada de moléculas neurotransmisoras).

2.1.2. OSCILACIÓN BIOLÓGICA

Muchos Comportamientos están basados en actividades rítmicas y se han encontrado ejemplos tanto en invertebrados y vertebrados, incluso el hombre. Ejemplos bien estudiados mediante los métodos actuales de análisis de actividad neuronal, han arrojado evidencias de circuitos simples con capacidades de oscilación. Las oscilaciones son variadas y estos dependen de la composición de la red neuronal. Tales patrones rítmicos —señales eléctricas—, son encontradas en la corteza extracelular e intracelular dentro del área cortical y subcortical, en cerebros del hombre y animales, figura 2.

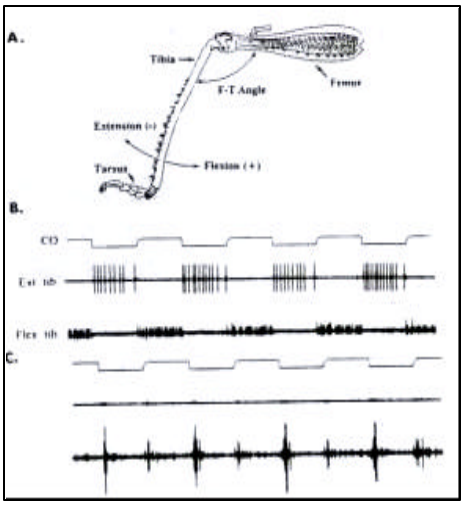


Figura 2. Un oscilador neuronal activado para accionar de las patas de un insecto, mostrándose el disparo del extensor y el músculo tensor. Tomado de Beer, R. [2]

Las observaciones de actividad oscilatoria, se remontan en trabajos realizados desde 1906 por Carlson y Sherrington, por Gray and Lissman en 1950 y en el trabajo de Wilson en 1961, entre los mas destacados. Un análisis sobresaliente, realizado en esta época, fueron la observación de los disparos neuronales en el axón gigante del calamar, con el cual fue posible realizar muchas mediciones, puesto que se podía mantener vivo por mucho tiempo.

Una respuesta muy sencilla que realiza un grupo de neuronas es el llamado oscilador. La respuesta de este grupo de neuronas es un patrón de pulsos realizándose de manera repetitiva. Estas pulsaciones se presentan en distintas tareas; actividades en los cuales se hace necesario realizar una serie de estímulos o acciones estereotipadas. Desde el punto de vista fisiológico, diversas actividades cotidianas que involucran la sobrevivencia, son resueltos mediante patrones rítmicos: nadar,

caminar, respirar, ritmos cardiacos, inclusive enfermedades del sistema nervioso central—como la epilepsia en los humanos—, son consecuencia de disparos de neuronas en forma oscilatoria. Estos patrones son generados por conexiones de neuronas, localizadas en el sistema nervioso central y pueden generar un ritmo de salida aun sin mantener señales de retroalimentación de los sensores. Las áreas de localización de estas conexiones neuronales se conoce como generador central de patrones (CPG), por sus siglas en inglés.

2.1.3. LOCOMOCIÓN EN INSECTOS

El estudio de los sistemas de locomoción en los insectos ha tomado gran relevancia en las últimas décadas. Con la llegada de los nuevos métodos de fotografía se han logrado observar de manera más detallada el caminar de los insectos, esto es, los patrones de locomoción. Uno de los insectos mas estudiados es la cucaracha, aunque se ha extendido a un sinnúmero de artrópodos, inclusive en los sistemas de locomoción de gusanos y serpientes, figura 3. Simulaciones sobre el movimientos de sus patas, su manera de caminar y hasta sus acciones de escape ante sus depredadores han sido acercamientos más detallados sobre el comportamiento y locomoción de las cucarachas. Siendo un insecto cuya única defensa ante sus depredadores es la huida, ésta arroja características de suma importancia en el estudio de la locomoción y los sistemas rítmicos de accionar sus patas.

En general, se puede hablar de un elevado porcentaje de vertebrados e invertebrados que pueblan la tierra y que presentan patas, como sistemas de locomoción. En este sentido, la misma naturaleza apunta la importancia de animales e insectos con patas. Las patas permiten la movilidad óptima en terrenos con superficies ásperas, en contraposición a los sistemas basados en ruedas. Así, el estudio de los sistemas de locomoción que la misma naturaleza ha perfeccionado, ha sido una tarea relevante que conjuga distintas disciplinas.

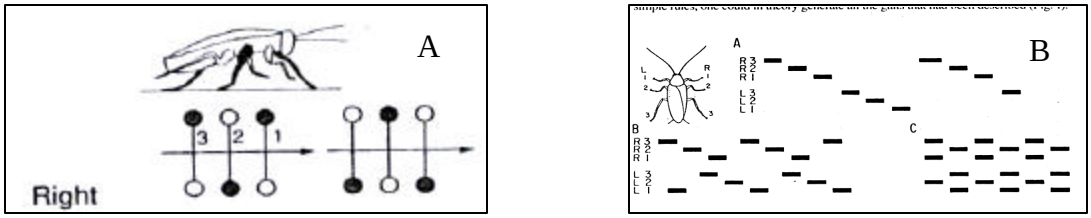


Figura 3. a) Formación del tripie en el caminado. b) Distintos patrones de pasos. Tomados de Beer, R.[2]

"Muchos de los insectos exhiben un simple estereotipo de paso (o patrón de pasos) durante el caminar, independientemente de cuán rápido o lento sea su movimiento"[1], escribe Fred Delcomyn. Los patrones de pasos son generados en el generador central de patrones.

En los últimos años, se han considerado grandemente el estudio de estos sistemas biológicos, debido a la importancia y capacidad que tienen para transportarse y la característica oscilatoria, no solamente en la locomoción, sino en distintos procesos biológicos.

2.2. OSCILADORES ARTIFICIALES.

2.2.1. MODELOS DE OSCILACIÓN.

Uno de los modelos más simples de redes neuronales, es el llamado oscilador. Este modelo genera un comportamiento periódico y puede construirse mediante la conexión de dos neuronas activas, con efectos mutuos de manera inhibitoria; así, se forma un sistema de autoinhibición. En la salida del sistema, se obtiene una serie de pulsos oscilatorios. Este modelo fue inicialmente propuesto por T.G. Brown, desde 1914, y describe la activación alterna de tensores y extensores en la pierna de un gato al caminar. Tal modelo es representado por la ecuación matemática:

$$Tr\ \frac{dxi}{dt} + x_i = -\sum_{j=1}^{n} a_{ij} y_i + u_i - bf_i + S_i$$

$$T_a \frac{df_i}{dt} + f_i = y_i$$

Tomado de Matzuoko [3]

Donde xi es el potencial de la membrana de la neurona i, u_i es la excitación externa, y f_i describe el grado de autoinhibición. El efecto inhibitorio es manipulado por la constante b. Las constantes de tiempo T_r y T_a, cambian la frecuencia del oscilador. La amplitud es afectada por la excitación externa u_i y por un sensor de señales Si.

Los modelos de oscilación de redes neuronales están basados sobre respuestas de oscilación biológica. Pueden estar compuestas mediante dos neuronas, tres o más. Los inicios del modelado matemático de neuronas se remonta desde varias décadas. En 1943 se publica el reporte de Warren S. McCulloch y Walter Pitts , *"A logical Calculus of Ideas Immanent in Nervous Activity"*, en el cual mediante una riguroso análisis matemático, muestra como las neuronas interconectadas entre sí, transmitiendo impulsos eléctricos, pueden desarrollar operaciones lógicas.

En el modelo de McCulloch-Pitts (figura 4) cada neurona consta de un conjunto de entradas, Xi, y una sola salida Yj. Cada entrada i está afectada por un coeficiente que se denomina peso y se representa por la letra Wij (gráficamente se representa por una media luna).

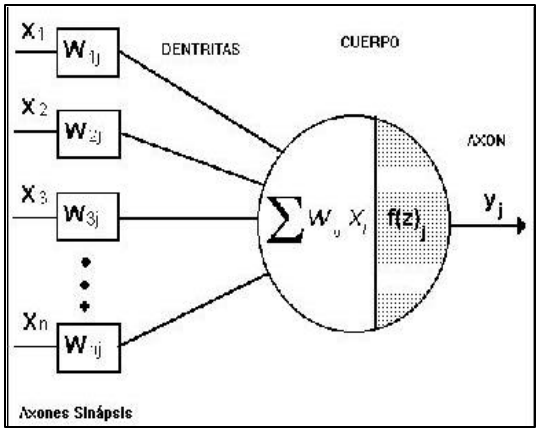


Figura 4. Modelo de McCulloch-Pitts.
Tomado de Santiago., S. [30]

El subíndice i refleja que el peso afecta a la entrada i, y el subíndice j que se trata de la neurona j.

Seis años después del modelo de McCulloch-Pitts, en 1949, en su libro *The organization of Behavior,* Donald Hebb presenta una regla de aprendizaje. Sostuvo que las conexiones entre las neuronas cambian conforme el cerebro aprende nuevas actividades. Posteriormente en 1958, Frank Rosemblatt da a conocer la primera red neuronal que produciría resultados prácticos*: el perceptrón*. El primer modelo del perceptrón consistía en un conjunto de 512 unidades de asociación (neuronas del tipo de McCulloch-Pitts, que recibiría información del exterior mediante fotoceldas, denominada retina. Los resultados se desplegaban sobre otro conjunto llamado unidad de respuesta o activación.

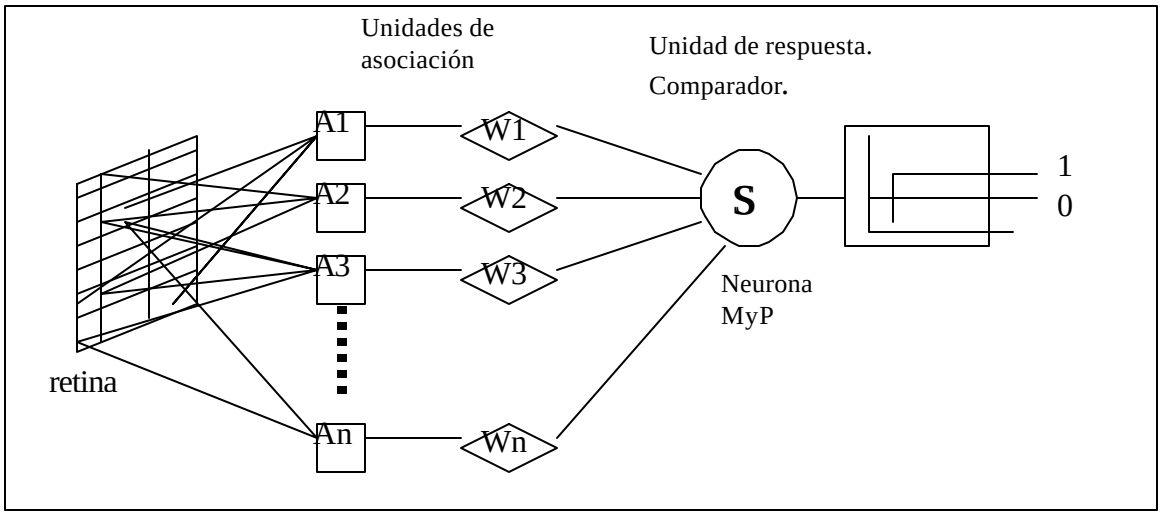


Figura 5. Perceptrón de Rosenblatt

Los modelos de oscilación de redes neuronales, están basados sobre respuestas de oscilación biológica. Pueden estar compuestas mediante dos, tres o más neuronas. El primer modelo neuronal fue propuesto por Rosenblatt —basado en el modelo de McCulloch-Pitts, figura 5—, en la década de los 40's, el cual consiste en un sumador de múltiples entradas y una sola salida. La neurona se mantiene en un estado de disparo o de inhibición, dependiendo de las entradas y de los pesos asignados a cada señal de entrada. El modelo básico de neurona, permite realizar una conexión de una a mas neuronas; las redes obtenidas, resultan ser cada vez más complejas, en la medida que se adicionan más neuronas, sin embargo para obtener una oscilación, que imite los impulsos eléctricos observados en los sistemas biológicos, basta mantener una conexión entre dos neuronas, de acuerdo a la figura 6.

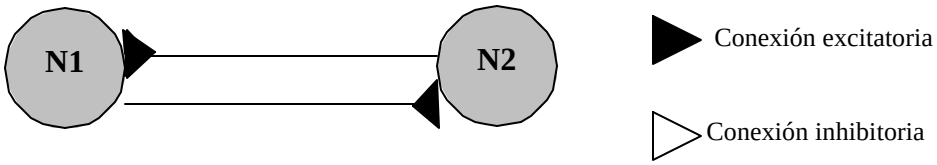


Figura 6. Oscilador con dos neuronas.

Sin embargo, es posible obtener oscilaciones de distintas características en redes neuronales de mas de dos neuronas. Uno de estas redes es el consistente de tres neuronas, en donde existe una conexión que puede ser de malla cerrada con inhibición en un solo sentido (figura 7a), o bien, inhibiciones mutuas en malla cerrada (figura 7b). Las respuestas de cada red difieren en sus características oscilatorias. En una red de tres neuronas con inhibición cíclica, se pueden obtener oscilaciones sincronizadas en cada neurona. Es decir, se puede lograr que las tres neuronas se disparen y oscilen todas al mismo tiempo. Esto no se posibilita con una inhibición en un solo sentido, ya que existe el efecto causal de una a otra neurona.

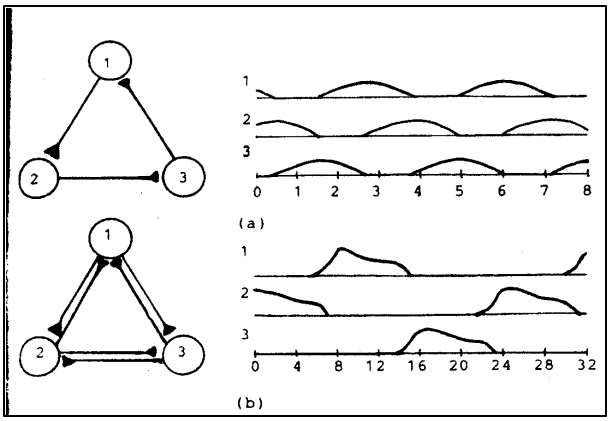


Figura 7. Redes oscilatorias de tres neuronas.
Tomado de Matzuoko, K.[3]

Osciladores de cuatro y de cinco neuronas en redes como muestra la figura 8 son de mayor complejidad y resultan ser mas inestables a pequeñas variaciones en sus parámetros.

2.3. ROBOTS ARTRÓPODOS

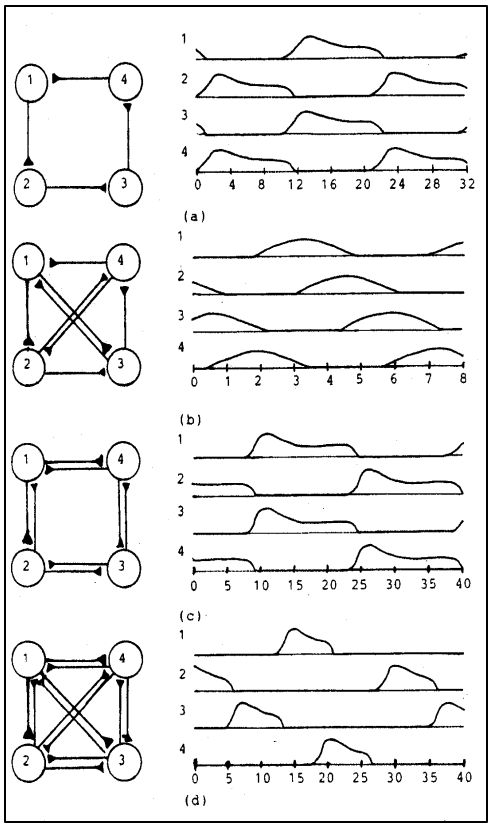


Figura 8. distintos arreglos para un oscilador de cuatro neuronas.
Tomado de Matzuoka, K.[3]

El surgimiento de robots con patas, a dado nuevas alternativas en los sistemas de locomoción de los robots autónomos, basados sobre ruedas. Se ha observado que la mayoría de los seres vivos que pueblan la tierra presentan patas, esto ha llevado a investigar el comportamiento mecánico y el sistema nervioso central, especialmente en los insectos. Se ha manejado que los sistemas de locomoción que utilizan patas presentan una mayor movilidad en terrenos rugosos, y en los insectos presentan mayor agilidad, velocidad, eficiencia y gracia. Para la aplicación de tales sistemas de locomoción en robots móviles se hace necesario la comprensión de los sistemas nerviosos; el caminar, correr, los sistemas mecánicas de equilibrio y los algoritmos de control así como los principios de locomoción de las patas.

El desarrollo de los robots con patas se ha dado desde hace un siglo y podrían mencionarse varios ejemplos, desde el documento publicado en 1893 acerca del mecanismo de un caballo y los muy variados sistemas de control de robots, unos manipulados por operadores y el trabajo mas reciente de Beer, Chile y Quinn[2], publicado en 1992 con un robot hexápodo controlando el sistema de locomoción mediante una red neuronal, basado sobre osciladores neuronales. Los trabajos se han enfocado a sistemas de locomoción de cuatro, seis, y hasta dos patas. Robots manipulados por sistemas de control de tipo oscilatorio, como el prototipo presentado por *Honda* de un humanoide, y el estudio del salto del canguro estudiado como alternativa de locomoción.

---

1 Delcomyng, F., *The walking of cockroaches,* Biological Neural Networks in Invertebrate Neuroethology and Robotics, USA, 1993

2 D. Beer, D. Quinn, J. Chile. *Robustness of a Distributed Neural Network Controller for locomotion in a hexapod Robot*. En IEEE Transactions on Robotics and Automation, vol 8, no. 3, junio 1992.

3 Matsuoka, K., *Sustained Oscillations Generated by Mutually Inhibiting Neurons with Adaptation,* Biological Cybernetics, vol. 52, pp. 367-376, 1985.

CAPITULO III

DISEÑO DEL ROBOT HEXÁPODO

3.1. CARACTERÍSTICAS DEL ROBOT LEGO

La decisión de implementar un robot hexápodo fue tomada debido a la existencia de estudios realizados a insectos con esta característica. Estos estudios se enfocan en el análisis de su dinámica, la cual depende de los estímulos que el medio ambiente, en el que se encuentra el hexápodo, le determine. Un ejemplo es la forma en que se desplazan al sentirse amenazados o simplemente cómo es su movimiento al buscar alimento. La simulación por computadora o la implementación física de modelos que se asemejan a éstos insectos ha sido posible gracias a los análisis mencionados anteriormente.

El diseño que se realizó en esta tesis utiliza piezas del *kit* LEGO MINDSTORMS, las cuales son de material plástico especialmente diseñadas tanto en forma como en tamaño para poder implementar robots. Algunas de las piezas más utilizadas son las que tienen formas cúbicas, planas y cilíndricas además de las barras y ejes. El *kit* también contiene dos motores de corriente directa (DC), dos sensores de tacto y un sensor de luz que permiten al robot interactuar con el medio ambiente y así poder tomar decisiones que influyan en su desplazamiento. El RCX es un bloque plástico de forma cúbica donde se encuentra el microcontrolador que controla tres motores por medio de tres salidas y sensores por medio de tres puertos de entrada. En él se almacenan los programas que simulan las salidas de los osciladores obtenidos con NEURORED. Contiene además una pantalla de cristal líquido en la cual se puede ver el numero de programa que se esta ejecutando, el valor de los sensores y el tiempo de ejecución del programa. Las opciones descritas se seleccionan por medio de cuatro botones colocados en su parte superior.

Uno de los objetivos de esta tesis es comprobar que es posible diseñar un robot hexápodo que sea capaz de desplazarse y evitar obstáculos que se le presenten, utilizando el *kit* de LEGO MINDSTORMS. Para obtener un diseño del robot que fuera robusto y funcional se realizaron seis diferentes prototipos, que se caracterizaban por el número de motores, su colocación, la combinación de diferentes engranes, la forma de las patas y la estructura misma. En un principio solamente se utilizó un solo *kit*, con el que se armó el primer prototipo, pero debido a sus diversas dificultades mecánicas se optó por la utilización de otros dos *kits*.

El trabajo realizado en el diseño mecánico fue paralelo a la simulación llevada a cabo desde NEURORED, y en distintas etapas, se hicieron pruebas en cada uno de los prototipos, esto con el fin de observar deficiencias, tanto mecánicas como en los programas de los osciladores.

DISEÑO DEL ROBOT HEXÁPODO

3.1.1. PRIMER MODELO

Para la implementación de este primer prototipo se utilizaron dos motores que generaban el movimiento de seis patas, tres en cada motor. El movimiento era transmitido a través de un eje conectado a uno de los motores; en su extremo contenía uno de los dos engranes de 8 dientes que a su vez se conectaba al engrane de 40 dientes. Las patas se sujetaban a este ultimo engrane, y cada una de éstas estaban formadas únicamente por una barra, figuras 9b y 9d.

El movimiento de las patas solamente tenia un grado de libertad. Lo que implicaba un movimiento torpe; aunado a esto, las tres patas de un sólo lado se unían a través de barras que impedían un control independiente de ellas, figura 9c. Solamente un RCX era utilizado para programar los movimientos de este robot, el cual se encontraba en la parte superior, figura 9a.

Debido a los inconvenientes presentados en este primer modelo, se propuso un nuevo diseño.

Figura 9a. Primer prototipo de seis patas

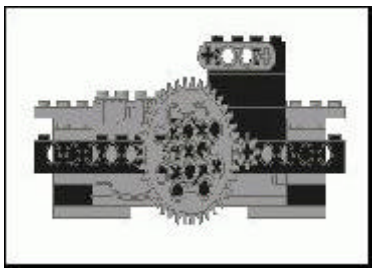


Figura 9b. Conexión del motor a los engranes.

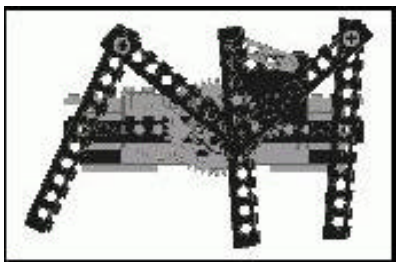


Figura 9c. Conexión de las patas al engrane y al cuerpo del Robot.

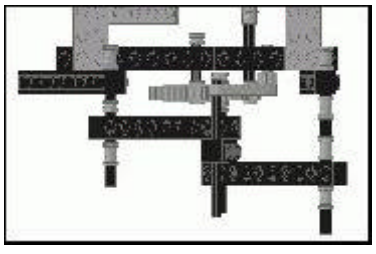


Figura 9d. Posición de los engranes.

3.1.2. SEGUNDO MODELO

En este segundo diseño, se utilizaron tres motores, los cuales generaban el movimiento en un par de patas cada uno. El robot resultaba ser demasiado inflexible para los sistemas de pasos que se intentaron simular. Dado que un motor tenía la finalidad de mover dos patas, éstas, se movían alternadamente, pero no podía mantener una misma posición en un mismo tiempo. Esto dificultaba y hacía inservible el manejo de pasos, en los cuales se necesita mantener un traslape de posición de ambas patas.

3.1.3. TERCER MODELO

Basándose en las dificultades presentadas en el segundo prototipo, se intentó una mejora en el diseño. Así, se introdujeron tres motores más, con la finalidad de mantener un control total e independiente de cada pata del robot. Sin embargo, esto traía ventajas y desventajas. Ventajas en el sentido de mayor flexibilidad en el movimiento de las patas y desventajas en el tamaño y peso. Con este modelo, era ya posible mantener las posiciones de las patas en cualquier situación y posición, sin embargo, el robot necesitaba agregársele un bloque RCX más. El consumo de energía se incrementaba, dado que había que alimentar seis motores, y las dimensiones del insecto se extendían.

Figura 10. Patas delanteras, motores y engrane

Las características que destacan a este modelo se presentan en los engranes de 40 dientes, de los cuales se utilizaban dos; uno de ellos recibía directamente el movimiento del motor a través del engrane de 16 dientes y el otro se unía paralelamente al primero por medio de un eje vertical que transmitía el movimiento, figura 10. Las patas estaban conformadas por cinco barras que definían un ángulo de 90°. En su parte media estaban conectadas al engrane inferior de 40 dientes, a través de dos barras curvas cuya función era la de provocar una extensión que generara un grado de libertad (el horizontal), y con una barra pequeña adherida en la parte superior se intentaba generar el segundo grado de libertad (el vertical), figura 11.

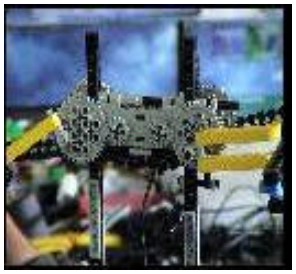


Figura 11. Conexión de los engranes al motor

Éste nuevo modelo de robot hexápodo presentó una debilidad mecánica en las articulaciones de las patas. A partir de aquí, se buscó diseñar un sistema mecánico robusto, que mantuviese el esqueleto del robot y los RCX.

3.1.4. CUARTO MODELO

El diseño de las patas empezó a tomar un papel importante a partir del modelo anterior, dado que una buena articulación daría un mayor dinamismo y control de éstas. Así, se fueron reforzando y haciéndose mejoras, hasta obtener el siguiente prototipo, figura 12.

En comparación con las patas de los modelos anteriores, éstas últimas se formaban con dos barras paralelas a 45° aproximadamente, las cuales se unían en su parte central por una barra que estaba fija al armazón del robot y a través de otra adherida a uno de los engranes de 40 dientes. El objetivo de estas barras era permitir que la pata se flexionara o extendiera, y al mismo tiempo, con dos barras curvas acopladas al engrane inferior de 40 dientes (conectado al motor) generará el movimiento que desplaza a la pata de izquierda a derecha.

Figura 12. Forma de la pata y conexión a los engranes.

El problema presentado por estas patas fue su inestabilidad para soportar el peso del robot además los grados de libertad no se presentaban de una manera amplia por lo que el caminar del robot era torpe y no definido.

Para poder corregir estos defectos se diseñó otro tipo de patas, como las mostradas en la figura 14. A diferencia de las patas de la figura 12, la barra curva (colocada al final de la pata) cambió su sentido, colocándola hacia adentro. Este fue el único cambio que sufrió, debido a que no hubo modificación en las barras que unen la pata a la estructura así como tampoco sufrieron modificación los engranes. Considerando a la fricción como otro de los problemas presentados en el diseño anterior, se colocaron en el extremo inferior de cada pata una liga, la cual evitaría un deslizamiento sobre el piso, figura13. Desgraciadamente los problemas presentados en estas modificaciones, impidieron que el robot tuviera una movilidad adecuada. Los movimientos seguían siendo torpes lo que implicaba un corto desplazamiento. Por su parte el engrane de 40 dientes localizado en la parte superior, forzaba su giro de tal forma que el movimiento no se transmitía de forma correcta a la pata.

Figura 13. Modificación de las patas

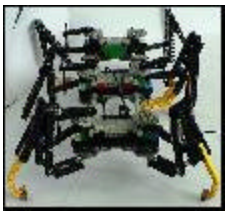


Figura 14. Hexápodo con patas mejoradas

3.1.5. QUINTO MODELO

Con la finalidad de tener una movilidad sin ningún contratiempo en los dos grados de libertad que queremos, se diseño una nueva relación de engranes, figura 15, al combinarlos de esta forma, se intentaba obtener un movimiento limpio de izquierda a derecha. Esto no fue posible debido a que los engranes se forzaban y pandeaban en un instante determinado del movimiento. Lo anterior sucedía por la barra que unía en su centro a los engranes.

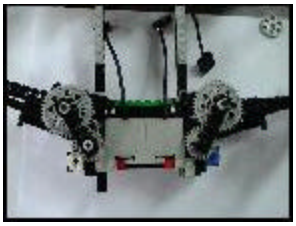


Figura 15. Engranes

Las patas tenían uno de sus puntos de giro acoplado a la estructura del robot, lo que provocaba un forzamiento en la barra en ciertos movimientos, y por lo tanto llegaba a desprenderse. Para corregir esto, se reforzó la unión con el agregado de barras que permitió un mejor amarre. El movimiento que transmitía el motor hacia las patas pasaba a través de una barra eje que bajaba de la parte central del engrane de 40 dientes a la barra unida a la estructura. La barra eje llagaba a doblarse o pandearse por ciertos esfuerzos provocados durante el movimiento que se tornaba torpe, figura16.

Figura 16. Quinto modelo

3.1.6. SEXTO MODELO

Figura 17. Diseño previo.

El hecho de poder obtener los dos grados de libertad con un solo movimiento y con la experiencia acumulada en el diseño de los modelos anteriores, se determinó rehacer el robot hexápodo. Para obtener este último modelo inicialmente se realizaron pruebas por separado. Éstas consistían en comprobar que con un solo movimiento se podía obtener los dos grados de libertad. Otro de las razones de trabajar separadamente el diseño fue para obtener un diseño de pata acorde al nuevo movimiento de funcionar, figura 17.

Después de comprobar la efectividad del diseño, se tomó la decisión de implementar un primer prototipo que tenía como característica principal su integración por módulos, los cuales constaban de la pata y de tres diferentes juegos de engranes que consistían de 8, 24 y 40 dientes. La ubicación de los motores se hizo en la parte superior del robot, en donde se encontraban uno

frente al otro. El movimiento era transmitido a través de juego de engranes colocados entre las barras de la estructura. El diseño implementado de las patas consistía de dos barras rectas en las cuales sus extremos cumplían las funciones de conectarla al motor a través de una barra-eje que provenía de los engranes por medio de una rueda con orificios y en el otro extremo estaba colocada la parte inferir de la pata,. Formada por dos barras curvas que estaban unidas a través de dos barras rectas.

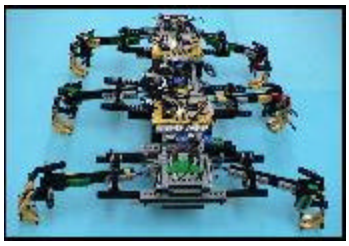


Figura 18. Diseño final

Los problemas presentados en este modelo se dieron en los engranes, ya que las patas no mantenían una posición firme determinada al final del movimiento debido a la fuerza ejercida por el piso sobre la pata. Aunado a esto, el diseño de las patas no era lo suficientemente robusto para soportar el peso de la estructura, que en ciertos movimiento se descuadraba, implicando que algunas piezas se desprendieran.

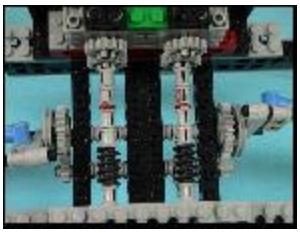


Figura 19. Ultimo diseño de engranes.

Corrigiendo estos inconvenientes se llegó al modelo definitivo, figura 19. En este modelo se corrigieron todos los problemas del diseño anterior. En lo referente a las patas, se reforzó la parte inferior con ligas que sirven como amortiguador y al mismo tiempo son soporte para el peso del robot, además de un agregado de barras curvas, barras eje y barras rectas que le dieron un mejor amare al resto de la estructura. El armado modular continuó igual, pero ahora los motores se colocaron en la parte superior uno junto al otro, colocándoles engranes de 16 dientes los cuales transmiten el movimiento a otro engranes de 16 dientes el cual esta conectado a la estructura del robot a través de una barra-eje colocada en posición paralela a las barras de la estructura.

Figura 20. Sensores de tacto.

En este modelo la utilización del tornillo sinfín permitió que la patas pudieran tener una posición final sin retroceder al termino del movimiento del motor. El tornillo está colocado en la barra eje, y transmite el movimiento a un engrane de 24 dientes sujeto por otra barra-eje que en uno de sus extremos se conecta un engrane de 8 dientes, este transmite el movimiento al último engrane que es de 24 dientes. En su centro una barra eje lo sujeta a la estructura del robot y a una rueda con orificios a la cual se sujeta la pata, figura 20. Solamente en la parte central los motores están ubicados frente a frente pero el engranaje es igual.

Los grados de libertad requeridos se obtuvieron gracias a que la patas están sujetas dentro de un marco. El marco principal contiene el eje del giro de izquierda a derecha de la pata, esto se debe a que ésta se conecta en su parte media. Esto permite colocar a la pata en distintas posiciones que generen un movimiento de arriba-abajo más amplio. Este marco está unido en su parte media lateral a dos barras rectas, que provienen del armazón del robot. La unión permite girar de arriba -abajo al primer marco, que por ende, mueve a la pata en este mismo sentido. De esta forma se provocan los dos movimiento buscados (o grados de libertad), con sólo el giro de un motor.

La utilización de dos bloques RCX que controlan tres patas cada uno, permite tener un mejor control del movimiento, pero el aspecto más importante es de la implementación de sensores de tacto en cada pata, los cuales indican la posición en la que se encuentra cada una de las patas en forma independiente, permitiendo elaborar una mejor programación, figura 21. La robustez del armazón incremento también el desplazamiento limpio del robot, cosa que no sucedía con los modelos anteriores.

CAPITULO IV

OSCILADORES GENERADOS CON NEURORED

4.1. RASGOS GENERALES DE NEURORED

Gracias al desarrollo de ciencias especializadas en el estudio del sistema nervioso como la Neurología y Neurofisiología, se ha logrado investigar más detenidamente las conexiones entre neuronas. El interés que con el paso del tiempo fue tomado éste tema, tanto para los sistemas computacionales como para la robótica, dieron pauta a la creación de programas que siguieran la lógica de los procesos neuronales o mejor aún, que simularan las conexiones neuronales biológicas. Uno de estos último programas es NEURORED.

NEURORED se inició con uno de los simuladores de MacGregor, llamado SYSTM11, diseñando a su alrededor programas que realizan tareas. En este simulador el núcleo dinámico está representado primordialmente por la variable conocida como el potencial de la membrana de la neurona y es alrededor de éste que se incluyen una serie significativa de parámetros neurofisiológicos como constantes de tiempo, umbral, conductancias, etc.

Es importante hacer notar las similitudes y las diferencias entre el programa NEURORED y las Redes Neuronales Artificiales. En ambos casos se trabaja con modelos de neuronas que pueden ser matemáticos o electrónicos. Pero en el caso de NEURORED su objetivo es que el modelo de red neuronal así como sus parámetros sean los más cercanos posible a un sistema biológico conocido total o parcialmente por medio de la experimentación neurofisiológica.[§]

El siguiente diagrama de bloques muestra los diferentes módulos que contiene el programa NEURORED los cuales permiten un mejor entendimiento y utilización tanto en el diseño como el análisis de la red neuronal biológica implementada.

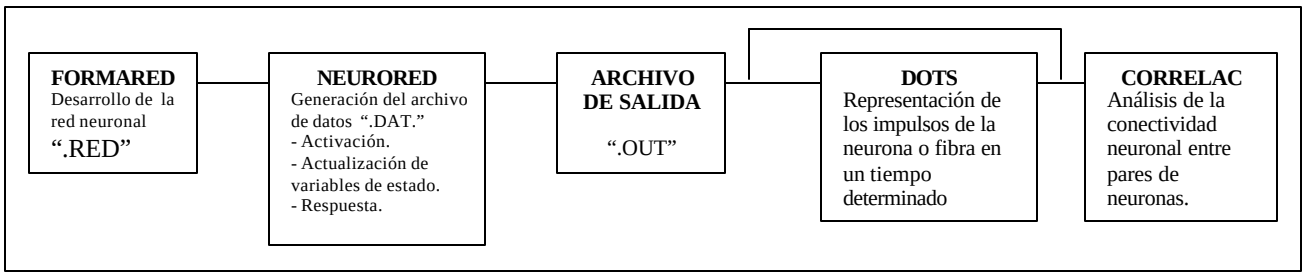


Figura 21. Módulos de NEURORED

[§] Para mas detalles sobre NEURORED consultar apéndice.

La decisión de utilizar a NEURORED como generador de Osciladores Biológicos fue debido a que es una herramienta fácil de utilizar y cumple con las características necesarias para la implementación de las redes neuronales biológicas. Además de lo anterior, la herramienta no necesita de un permiso especial para su utilización, como suceden con algunos otros programas que existen en el mercado.

4.1.1 PARÁMETROS

El archivo .DAT generado por el módulo FORMARED, contiene la información que permite ver la simulación de la red neuronal en forma de puntos (a los que llamaremos ***disparos***) con el módulo DOTS. Aunque son varios los parámetros que se encuentran en éste archivo, solamente los parámetros listados a continuación son los que permiten modificar la red neuronal de tal manera que no cambie la configuración inicial de la red, que en este caso es un Oscilador. Un elemento importante en la conexión de las redes neuronales es la fibra, esta determina la activación de las neuronas, y realización una especie de energización. El parámetro semilla es un valor aleatorio que permite al simulador arrojar resultados que tengan características similares a las que se obtendría desde el punto de vista biológico.

A. Intensidad sináptica: permite modificar el número de veces que una neurona se dispara o inhibe debido a la sinapsis con otra u otras.
B. Intensidad de la fibra hacia la neurona: permite modificar la excitación o inhibición que provoca una fibra hacia una o varias neuronas.
C. Tipo de sinapsis: permite elegir si la sinapsis será excitatoria o inhibitoria.
D. Umbral: permite elevar el potencial de la membrana.
E. Tiempo de simulación: permite modificar el tiempo en el cual se simulará la red neuronal biológica.

Dentro del archivo .DAT es importante identificar el renglón que en el cual están contenidos los parámetros antes descritos y la forma de identificarlos es por medio de los parámetros que a continuación se enuncian:

F. Número total de componentes: indica el número de fibras o axones más el numero de neuronas que existen en la red neuronal.
G. Número de fibras: indica el numero de fibras la red neuronal.

H. Número de neuronas: indica el número de neuronas que contiene la red neuronal.

Para poder identificar mejor los parámetros descritos, se muestra a continuación un ejemplo del archivo extensión .DAT

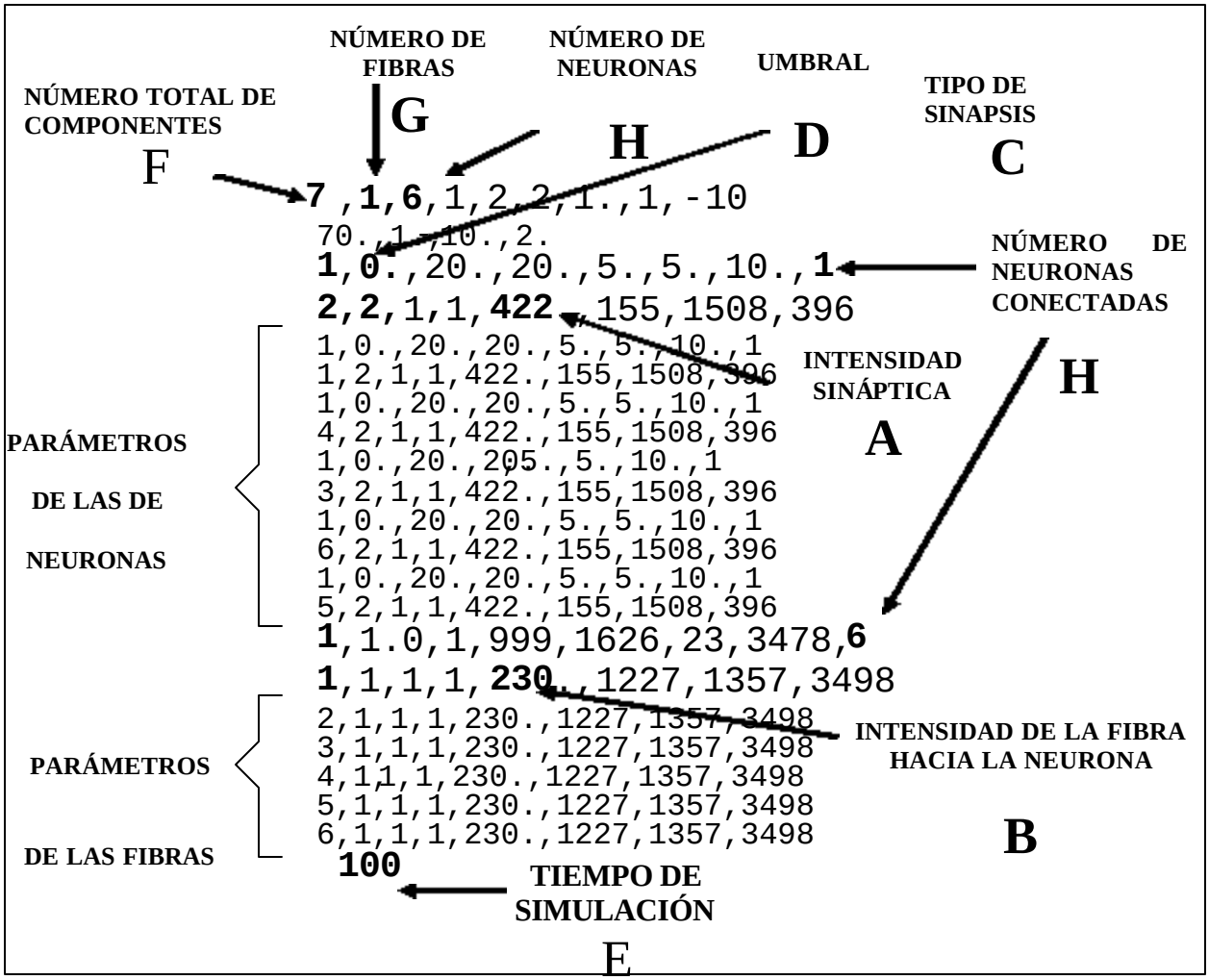


Figura 22. Identificación de algunos parámetros de un archivo .Dat

4.1.2. PATRONES DE PASOS

Los patrones de pasos son el parámetro que nos permite saber la forma en que las neuronas deben oscilar. Están basados sobre el análisis de los patrones de locomoción de los insectos, expuesto anteriormente, los cuales son representados con fragmentos de líneas (fig. 3 y fig.23). Con los patrones podemos implementar una red neuronal que tenga las mismas características de disparo que el patrón de pasos, lo que permitirá que el arreglo neuronal de tipo oscilatorio imite la acción de tensores y extensores de las pata de muchos insectos y animales. En general, se ha observado que se forman tripies en cada secuencia (véase figura 29), lo que significa que tres

patas sostienen el cuerpo y las otras tres se mueven hacia delante generando así el desplazamiento del robot hexápodo.

4.1.3. IMPLEMENTACIÓN DEL PATRÓN DE PASOS PARA CAMINAR

El modulo DOTS nos permite visualizar el comportamiento o actividad de cada una de las neuronas de una red específica, por medio de una serie puntos los cuales son llamados disparos o salvas, que integrarán las ráfagas. Éstos se despliegan en pantalla en alguno de los cuatro puertos de salida que aparecen en primer plano en este modulo; además de mostrar en la parte inferior el total de disparos generados por cada una de las neuronas y el número de puerto en el que se desplegaron.

Los disparos que se muestran en pantalla indican que tan alejado o tan cercano se está de los disparos con los cuales se representa un patrón de pasos.

Uno de los patrones básicos es el de caminar, el cual tiene una característica peculiar, que conjuga equilibrio y dinamismo del centro de masa del insecto, o el cuerpo que transporta, de modo que mantiene un movimiento óptimo en los pasos de un hexápodo. Para poder obtenerlo en un oscilador generado con NEURORED se realizaron diversas pruebas con redes neuronales que constaban de 2, 3, 5 y seis neuronas. Cada oscilador encontrado tuvo características diferentes entre sí, que permitieron elegir con el oscilador que más se apegara al patrón de pasos de caminar.

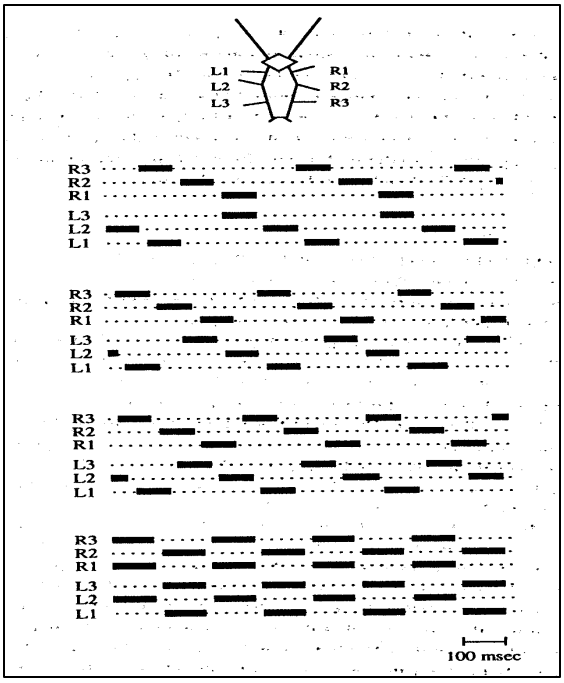


Figura 23. Patrón de pasos de caminar. Tomado de Beer, R. [6]

Una característica especial de los osciladores es la respuesta repetitiva y sencillas de sus disparos de salida, los cuales se obtienen de una combinación de los pesos inhibitorios entre las neuronas y la fibra que los alimenta. El intervalo de valores para los cuales es posible obtener patrones oscilatorios es amplio, en consecuencia dificulta la obtención de un resultado

determinado; así, lograr que un grupo de neuronas oscile en un determinado tiempo y orden, puede ser impreciso, coadyuvado a las ecuaciones no lineales que determinan el modelo de simulación.

Las simulaciones llevadas a cabo son de las redes neuronales más representativas y sencillas. Están acompañadas de las gráficas de configuración de puntos ejecutado por el programa DOTS que permitirán hacer más comprensible la explicación.

4.1.4. INHIBICIONES ENTRE NEURONAS

Una de las características que tienen las redes neuronales que se configuran como osciladores, es que la conexión existente entre las neuronas debe ser inhibitoria. Las conexiones tienen, además, una división: son cíclicas o mutuas. Se dice que una conexión es cíclica cuando la unión entre neuronas va en un solo sentido (ida o vuelta), al contrario de las conexiones mutuas donde las uniones van en ambos sentidos, tal como muestra la figura 25.

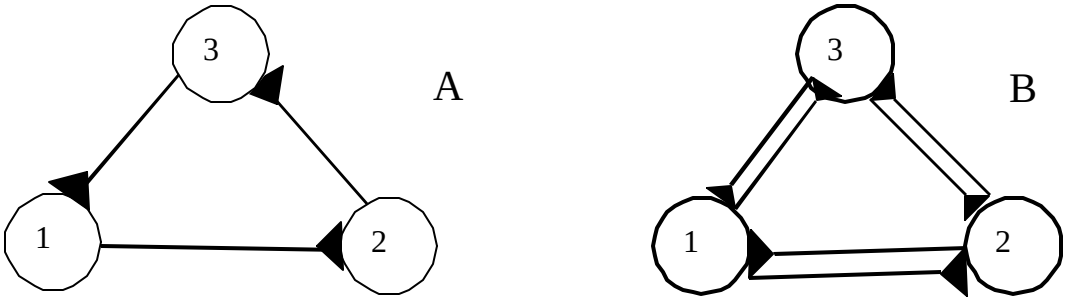


Figura 24. A) Inhibición Cíclica B) Inhibición Mutua.

Para una red de dos neuronas, la oscilación se mantendrá solamente con un arreglo mutuo.

4.2. OSCILADORES DE DOS NEURONAS

4.2.1. SIMULACIÓN DE OSCILADORES CON LOS DIFERENTES MÓDULOS DE NEURORED.

a) RED NEURONAL NY1

Una de las primeras simulaciones de oscilación obtenidas, es la llamada configuración básica de un oscilador y consta de dos neuronas. Debido a que su implementación y comportamiento son sencillos, permiten un mejor entendimiento de las características del oscilador generado en NEURORED.

La siguiente tabla contiene los valores utilizados en la primera red:

TABLA 1		
NOMBRE DEL ARCHIVO	**NY1**	
❖ NÚMERO DE LA NEURONA	**1**	**2**
❖ INTENSIDAD SINÁPTICA	**-86**	**-56**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**10**	**10**
❖ TIPO DE SINAPSIS	**INHIBICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**1**	**1**
❖ TIEMPO DE SIMULACIÓN [ms]	**300**	**300**

Los disparos para la red NY1 se muestran en la gráfica 1. Observando el gráfico en su tiempo inicial, se presenta un transitorio o adaptamiento en cada neurona, el cual se presenta en todas las redes. NY1 se comporta como un oscilador dado que existen patrones repetitivos, aún cuando no hay una sincronía en las salidas de cada Neurona. El oscilador presenta un traslape de dos disparos en el inicio y final de cada ráfaga de salida entre las dos neuronas.

Uno de los parámetros que permite lograr una oscilación sincronizada es la variación del umbral, que es un nivel de referencia de voltaje, el cual al ser rebasado genera un impulso que representa la excitación de la neurona. En NEURORED el valor del umbral es de 0; con ello la red mantiene un nivel de umbral constante, en contraparte, un umbral distinto de cero, presenta una movilidad o pendiente en cada uno de los disparos. los mismos resultados de la neurona "ganadora".

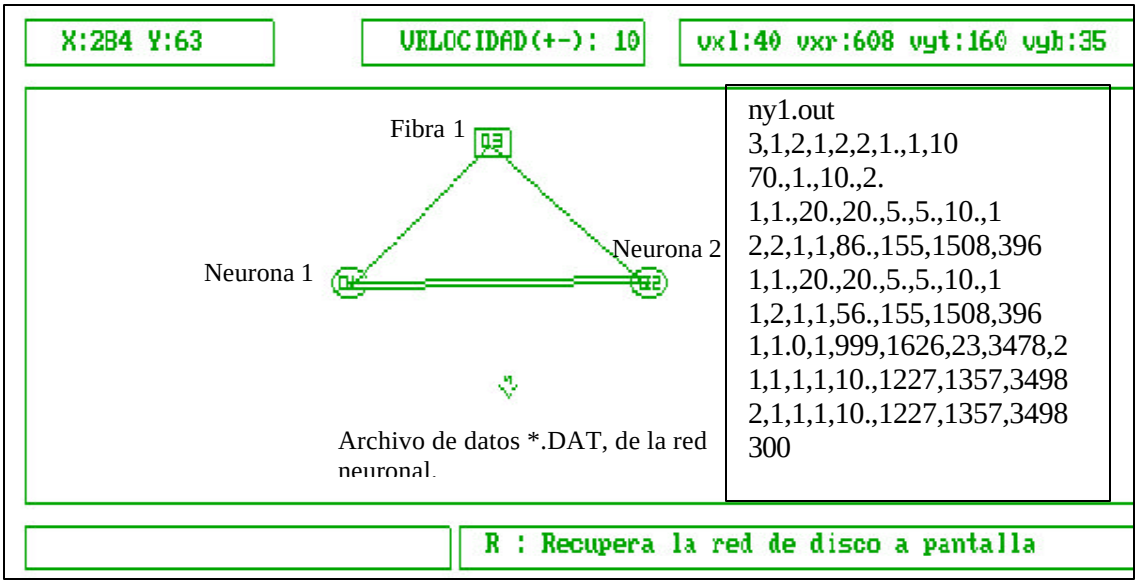
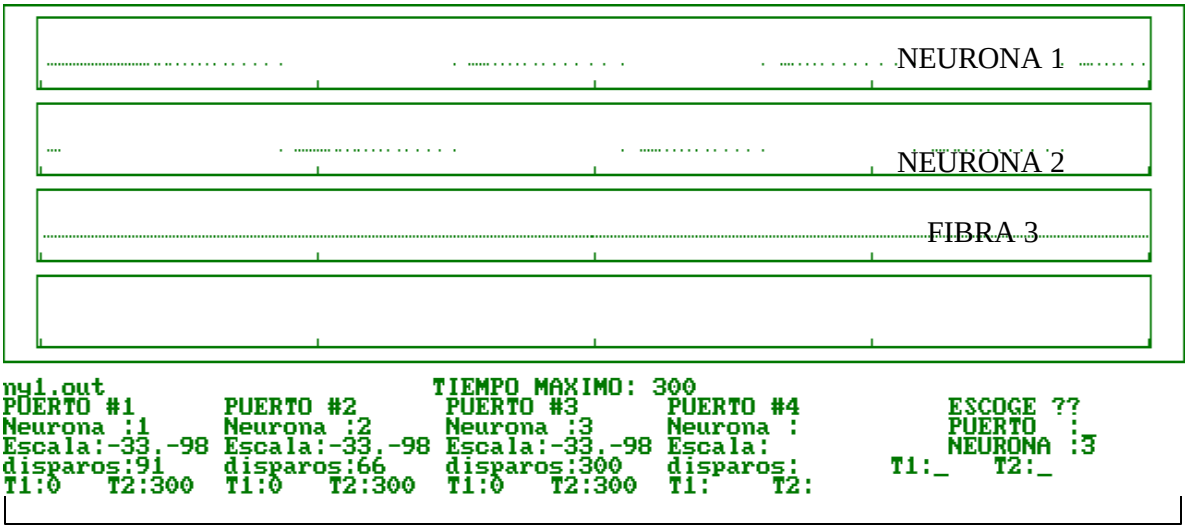


Figura 25. Red neuronal NY1 y archivo *.dat.



Gráfica 1. Resultados del oscilador NY1 analizado con DOTS.

Las oscilaciones tienden a una desaceleración, es decir, en cada ráfaga disminuye el número de disparos en ambas neuronas. Así, las ráfagas no contienen el mismo numero de disparos. Se debe hacer notar que al principio de las ráfagas existe un disparo aislado; en este sentido, se hace necesario lograr un patrón continuo.

b) RED NEURONAL NY3

En esta configuración se varia el valor de los pesos entre neuronas, así como el tiempo de simulación. La intensidad de la fibra y el umbral de disparo se mantienen con el mismo valor.

TABLA 2		
NOMBRE DEL ARCHIVO	**NY3**	
❖ NÚMERO DE LA NEURONA	**1**	**2**
❖ INTENSIDAD SINÁPTICA	**-346**	**-356**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**10**	**10**
❖ TIPO DE SINAPSIS	**INHIB ICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**1**	**1**
❖ TIEMPO DE SIMULACIÓN [ms]	**500**	**500**

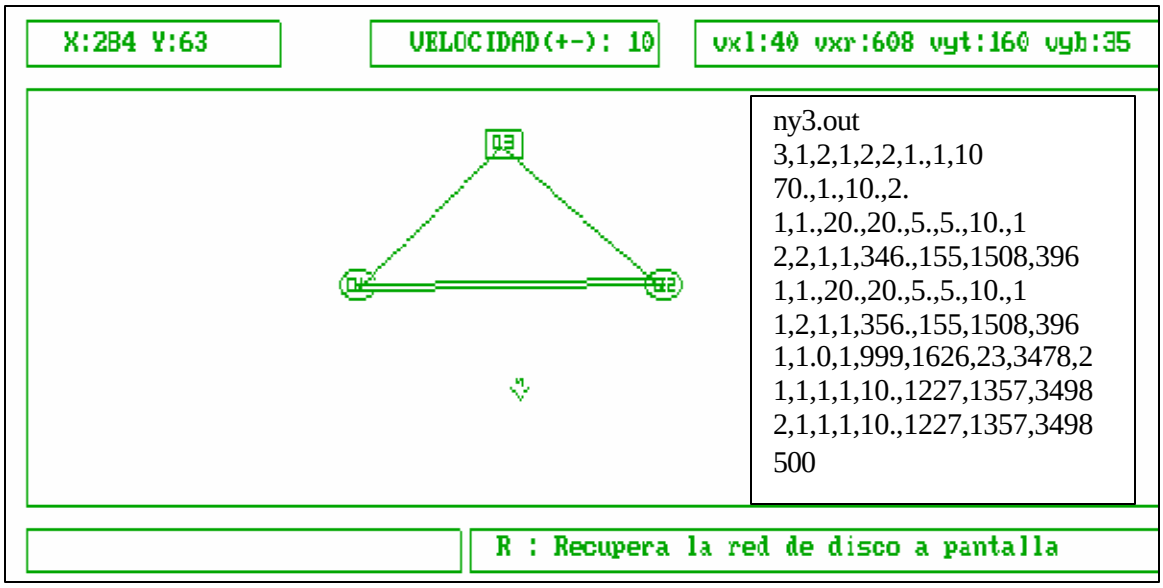
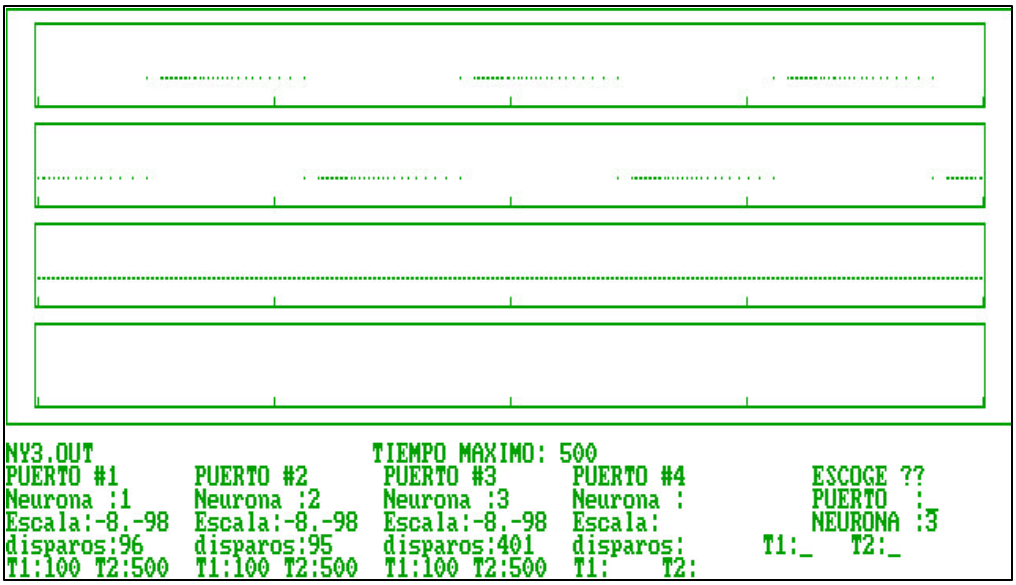


Figura 26. Red neuronal NY3

NY3 es un oscilador que presenta una diferencia respecto a la red anterior, en donde las ráfagas para cada neurona son idénticas. Mientras tanto, en esta red se mantienen los espaciamientos constantes entre las ráfaga de cada neurona, lo que implica que no existen espacios. Con respecto al número de disparos por ráfagas éstos aumentaron a 25, siendo este número constante en todas ellas y el total es prácticamente el mismo en las dos neuronas, característica importante que determina el tiempo en que una de las patas del robot hexápodo permanece en contacto con una superficie.



Gráfica 2. Resultados del oscilador NY3

c) RED NEURONAL NY5

NY5 es otro oscilador en el cual se logra mantener el mismo número de salvas en la salida de cada neurona. Para lograrlo, se realizaron cambios en el oscilador anterior, asignando un valor umbral de disparo de 1.1. Para la obtención de este valor, se realizaron aproximaciones. Como se puede apreciar en la tabla 3, la configuración continúa siendo la misma, así como la intensidad sináptica. Además se cambiaron los pesos de las fibras para mantener una probabilidad de disparo más estable.

TABLA 3

NOMBRE DEL ARCHIVO	**NY5**	
❖ NÚMERO DE LA NEURONA	**1**	**2**
❖ INTENSIDAD SINÁPTICA	**-346**	**-356**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**19**	**19**
❖ TIPO DE SINAPSIS	**INHIBICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**1**	**1**
❖ TIEMPO DE SIMULACIÓN [ms]	**500**	**500**

Gracias al incremento de la intensidad de la fibra, el disparo que permanecía aislado al inicio de cada ráfaga ya no aparece, mientras que el número de disparos para cada neurona es similar. La continuidad de ráfagas esta alterada, debido a la aparición de espacios en blanco existentes entre ellas, principalmente al final de cada patrón de disparo.

Los resultados obtenidos permitieron mejorar algunas características, pero otras se perdieron.

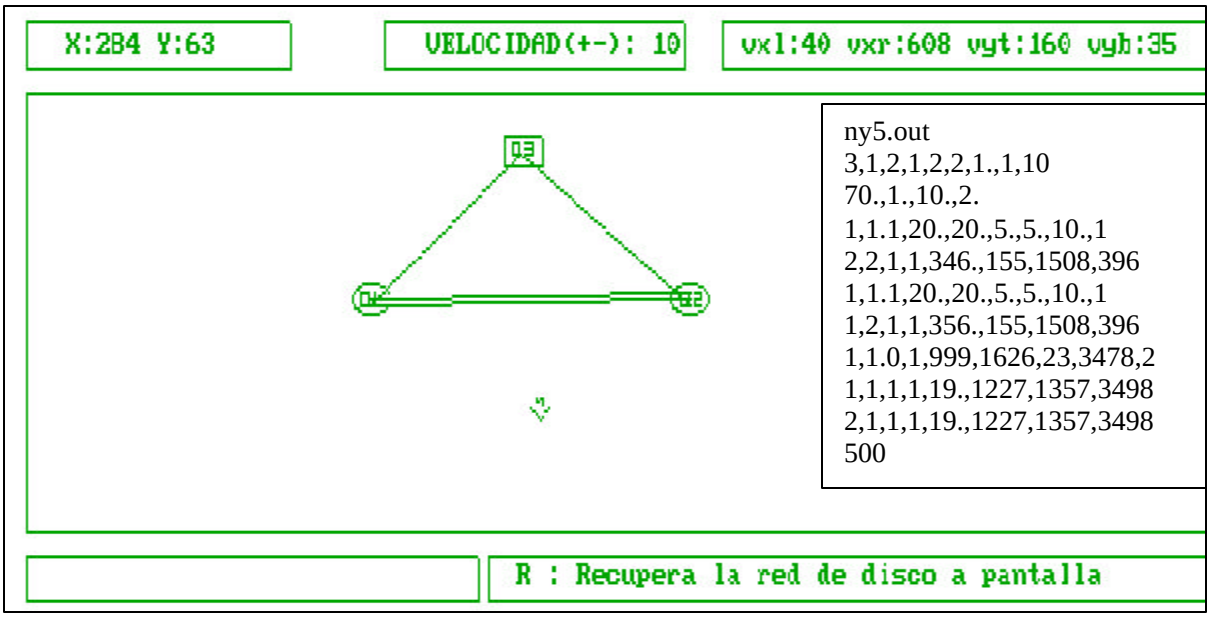
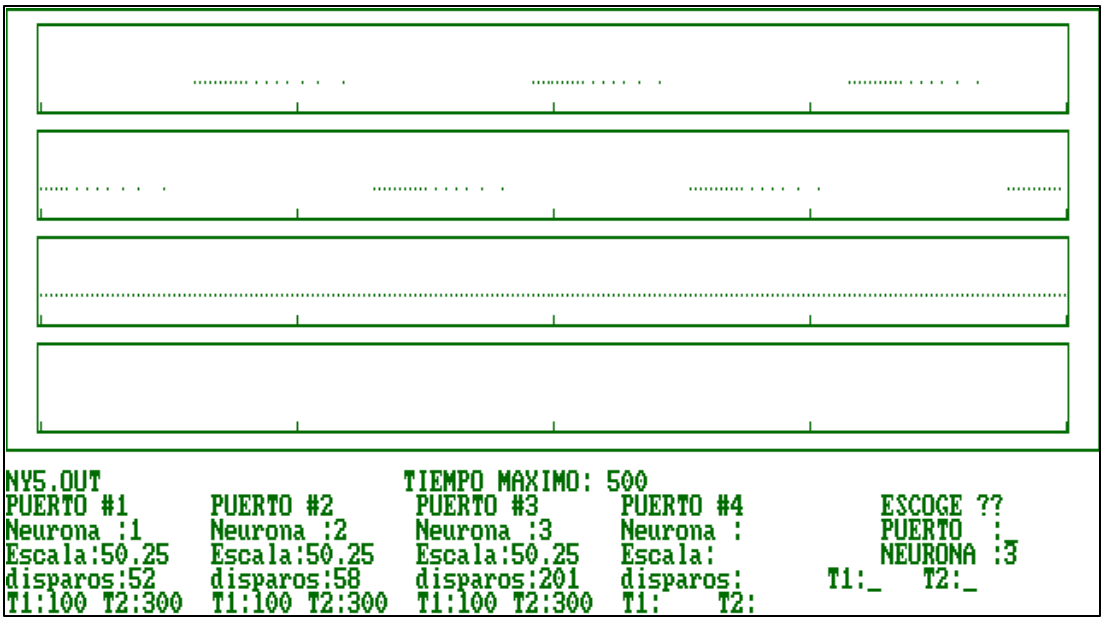


Figura 27. Red neuronal NY5



Gráfica 3. Resultados Del Oscilador Ny5

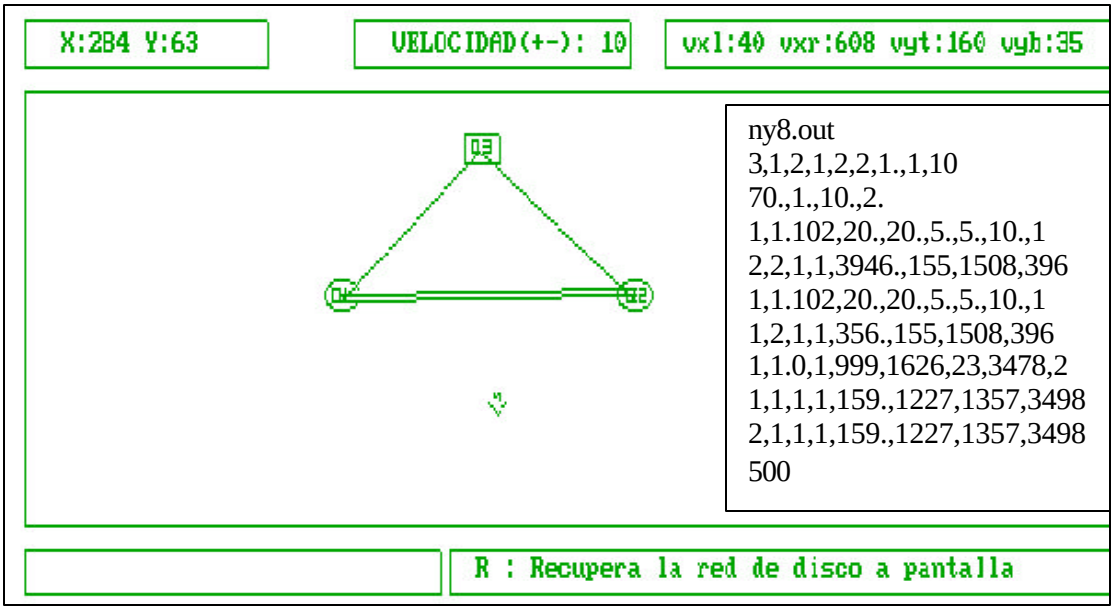


Figura 28. Red neuronal NY8

d) RED NEURONAL NY8

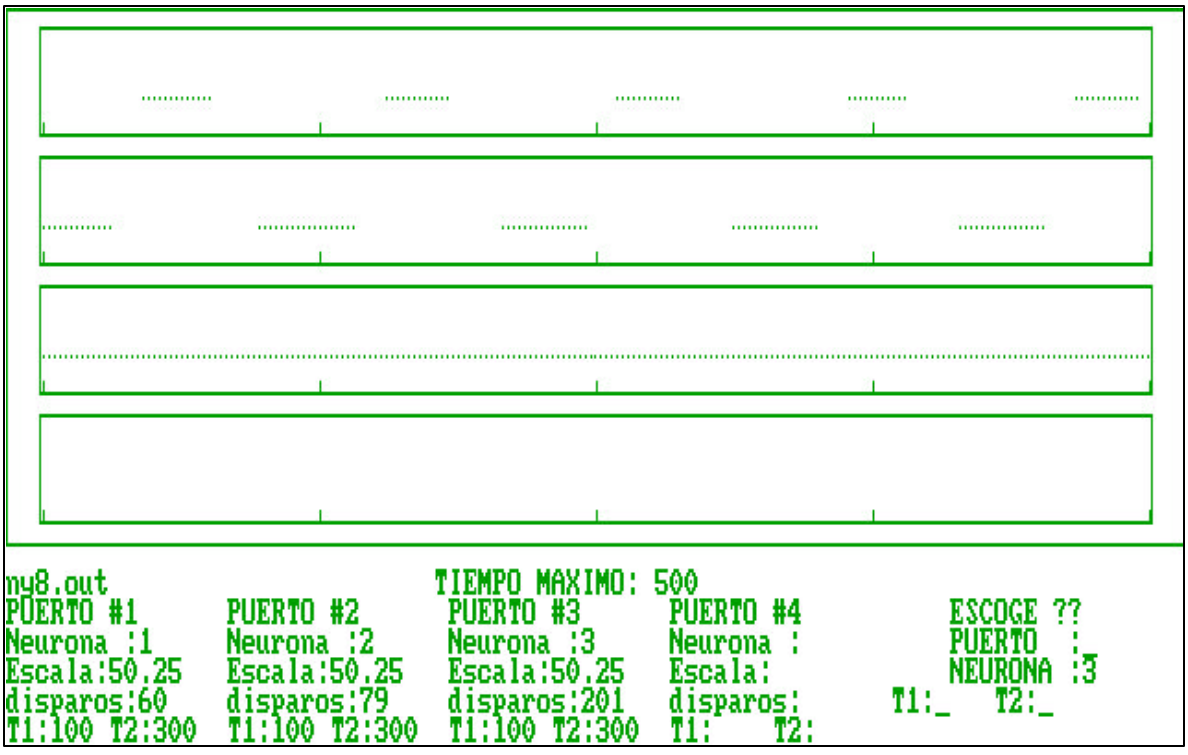
El peso que proporcionó la neurona uno hacia la dos en esta red, tuvo un incremento notable, pasó de cientos a miles. El peso de la fibra se incremento proporcionalmente en las dos neuronas, pero no fue así en el umbral de disparo. Los datos aparecen en la tabla 4.

TABLA 4

NOMBRE DEL ARCHIVO	NY8	
❖ NÚMERO DE LA NEURONA	**1**	**2**
❖ INTENSIDAD SINÁPTICA	**-3946**	**-356**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**159**	**159**
❖ TIPO DE SINAPSIS	**INHIBICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**1.102**	**1.102**
❖ TIEMPO DE SIMULACIÓN [ms]	**500**	**500**

Los resultados obtenidos de los valores de tabla permitieron obtener ráfagas de disparo sin una desaceleración, lo que permitió la aparición de ráfagas uniformes, pero la discontinuidad entre ráfagas continua apareciendo. El número de disparos existentes en cada ráfaga de ambas neuronas no fue la misma.

La aportación relevante de esta configuración son las ráfagas que tienen una forma uniforme, debido al incremento del peso de la neurona uno hacia la dos.



Gráfica 4. Resultados del oscilador Ny8

4.3.OSCILADORES DE TRES NEURONAS

La implementación de una tercera neurona se debe no solo a los resultados obtenidos en las redes de dos neuronas, sino también al diseño del robot. Dentro de la búsqueda de patrones oscilatorios adecuados al prototipo del robot con seis motores, se considera asignar una red de tres neuronas que representase el trípode observado por un insecto en su locomoción (figura 29) —patrón de paso básico—, en el cual las tres neuronas se disparen sincrónicamente. Se considera que cada neurona controla un motor en los extremos del robot.

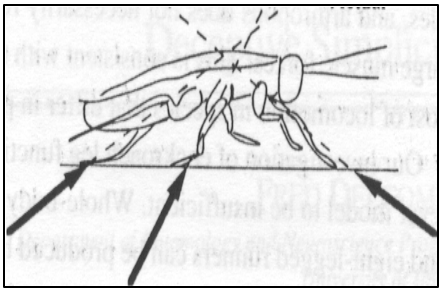


Figura 29. Tripie formado al caminar. Tomado de Beer, R.[6]

En este sentido se propuso la configuración de una red cíclica de tres neuronas.

4.3.1. INHIBICIÓN CÍCLICA

e) RED NEURONAL OSILA

OSILA es el primer oscilador obtenido con esta configuración.

La siguiente tabla muestra los valores que se utilizaron para generar esta primera red de tres neuronas.

TABLA DE RESULTADOS 5			
NOMBRE DEL ARCHIVO	**OSILA**		
❖ NOMBRE DE LA NEURONA	**1**	**2**	**3**
❖ INTENSIDAD SINÁPTICA	**-10**	**-10**	**-10**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**5**	**5**	**5**
❖ TIPO DE SINAPSIS	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**0**	**0**	**0**
❖ TIEMPO DE SIMULACIÓN [ms]	**100**	**100**	**100**

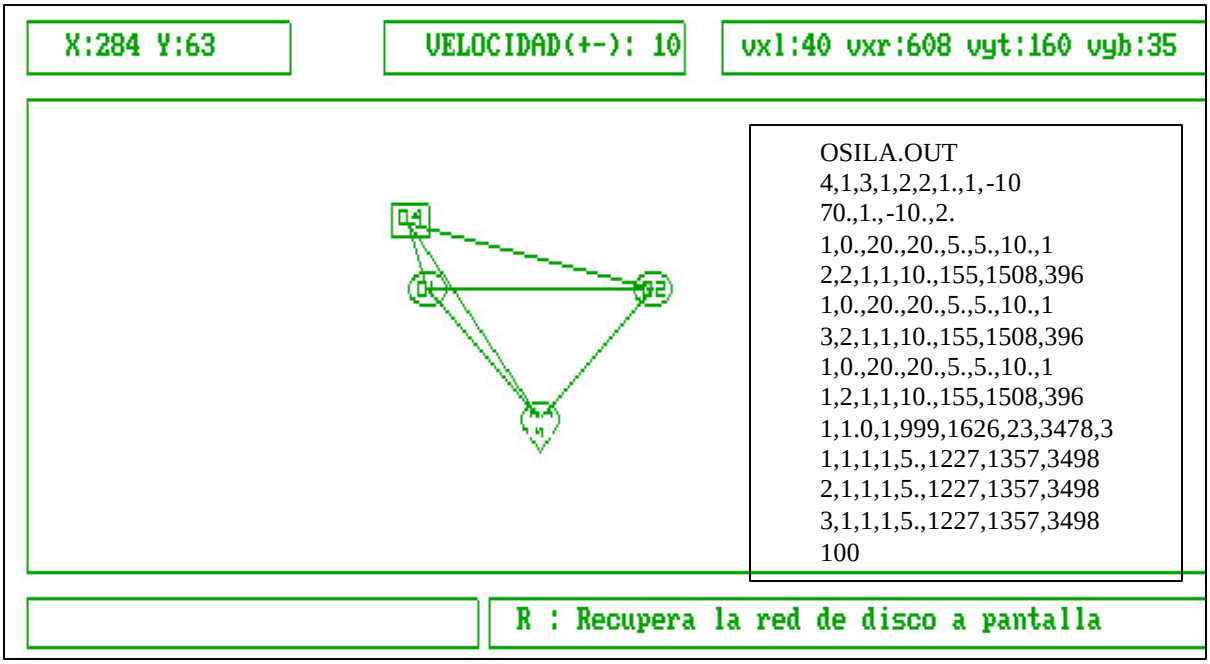
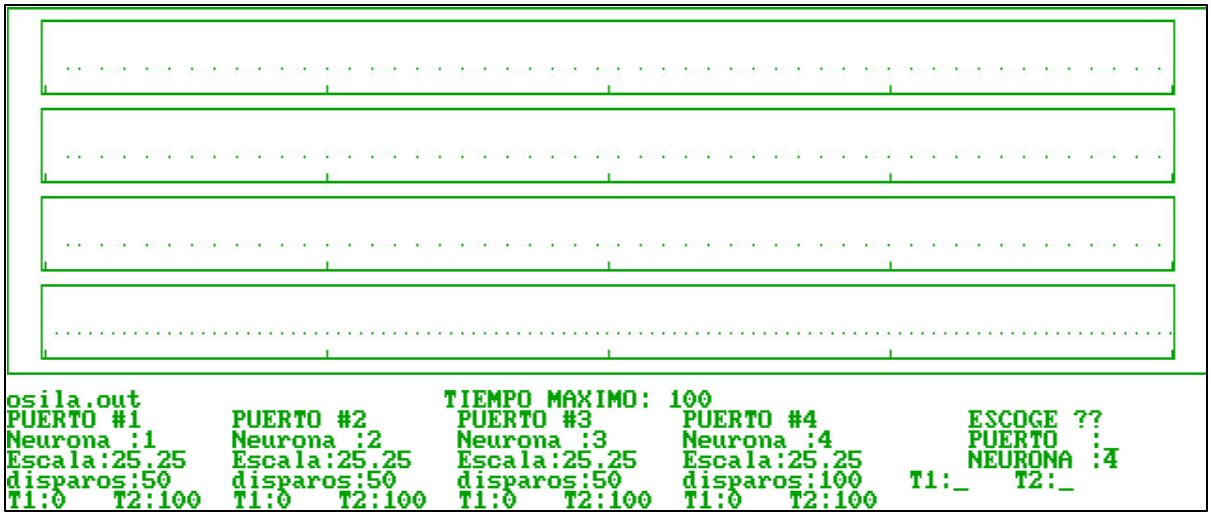


Figura 30. Red neuronal osila



Gráfica 5. Resultados del oscilador Osila

La red no presenta una oscilación en un rango de disparos determinado, sin embargo los disparos son unitarios y no existe una ráfaga constante con una determinación pronunciada. Las neuronas se disparan al mismo tiempo, tal como se pretendía inicialmente, pero los disparos son unitarios. Las semillas de las neuronas son idénticas, así como los pesos de las conexiones son iguales entre cada neurona. El umbral de disparo se mantiene en el valor constante en estas simulaciones (umbral = 0).

f) RED NEURONAL OSILA1

El oscilador OSILA1, es una red con la cual se obtienen ráfagas constantes para cada neurona. Los valores que se asignan a los pesos de la red neuronal, difieren de la red inicial dado que existe una variación tanto en los pesos entre cada neurona y las fibras que las alimentan. El valor umbral en cada neurona son iguales en el valor asignado por NEURORED así como las semillas.

TABLA DE RESULTADOS 6			
NOMBRE DEL ARCHIVO	**OSILA 1**		
❖ NOMBRE DE LA NEURONA	**1**	**2**	**3**
❖ INTENSIDAD SINÁPTICA	**-500**	**-300**	**-270**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**150**	**70**	**15**
❖ TIPO DE SINAPSIS	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**0**	**0**	**0**
❖ TIEMPO DE SIMULACIÓN [ms]	**100**	**100**	**100**

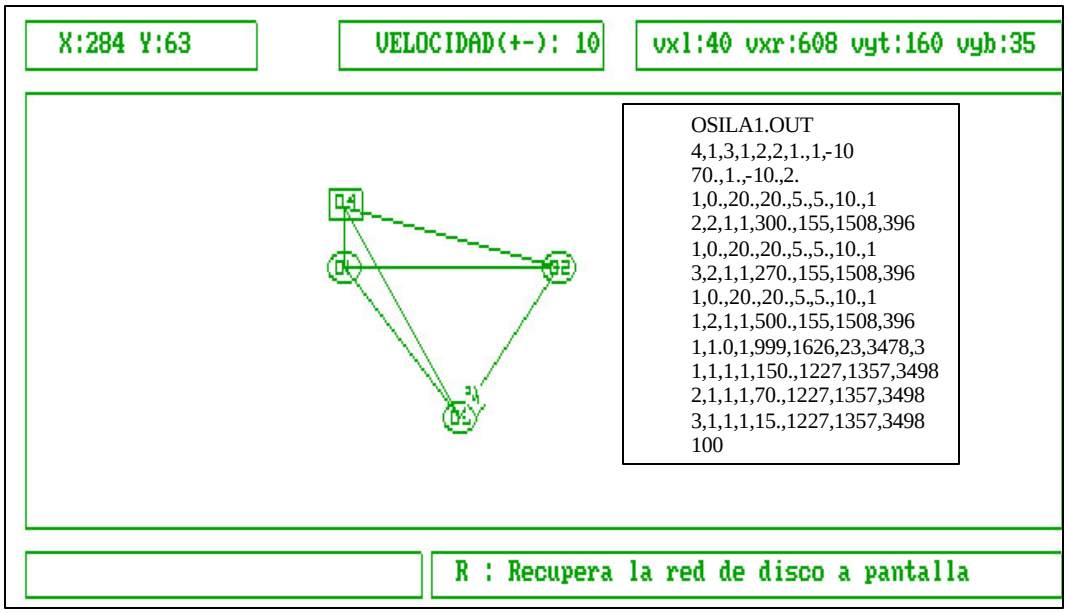
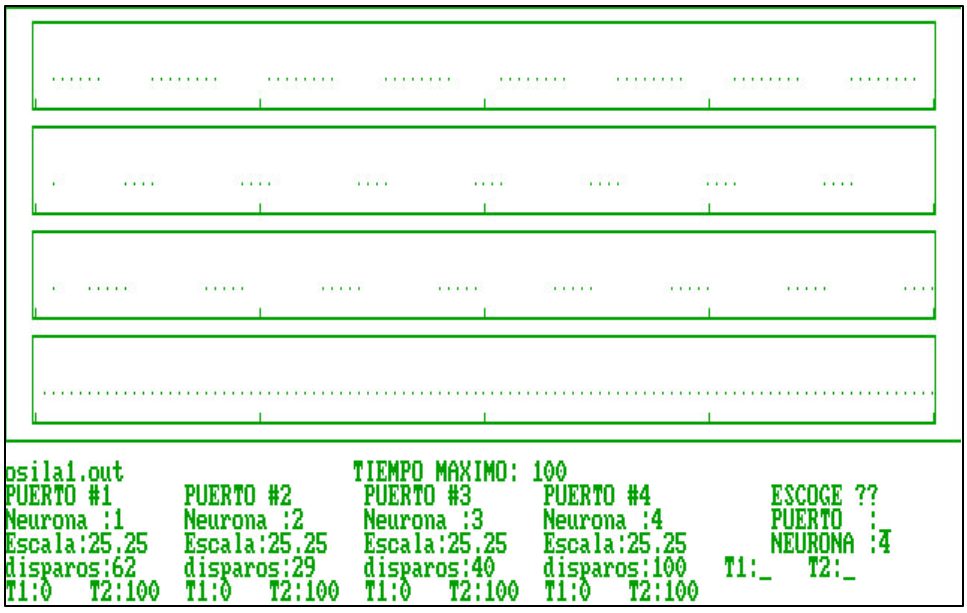


Figura 31. Red osila 1



Gráfica 6. Resultados del oscilador osila 1.

En la neurona uno se presentan ráfagas de ocho disparos cada una; la neurona dos de cuatro y la neurona tres de cinco disparos. Es necesario observar una causalidad en cada neurona, esto es, las ráfagas no se superponen, aunque existe un traslape de activación entre neuronas de por lo menos un disparo.

g) RED NEURONAL OSILA2

Este oscilador de tres neuronas que presenta la misma configuración que el oscilador anterior, tiene una respuesta oscilatoria para cada neurona. Además mantiene una ráfaga que sincroniza con las otras neuronas. Cada uno de los elementos neuronales de la red, oscila en el mismo tiempo y con el mismo número de disparos y en el intervalo de tiempo correspondiente. Esto se logra, asignando valores iguales en la intensidad sináptica entre cada neurona; análogamente se asignan magnitudes en la intensidad de la fibra.

TABLA 7. PARÁMETROS DE LA RED

NOMBRE DEL ARCHIVO	**OSILA 2**		
❖ NOMBRE DE LA NEURONA	**1**	**2**	**3**
❖ INTENSIDAD SINÁPTICA	**-100**	**-100**	**-100**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**50**	**50**	**50**
❖ TIPO DE SINAPSIS	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**0**	**0**	**0**
❖ TIEMPO DE SIMULACIÓN [ms]	**100**	**100**	**100**

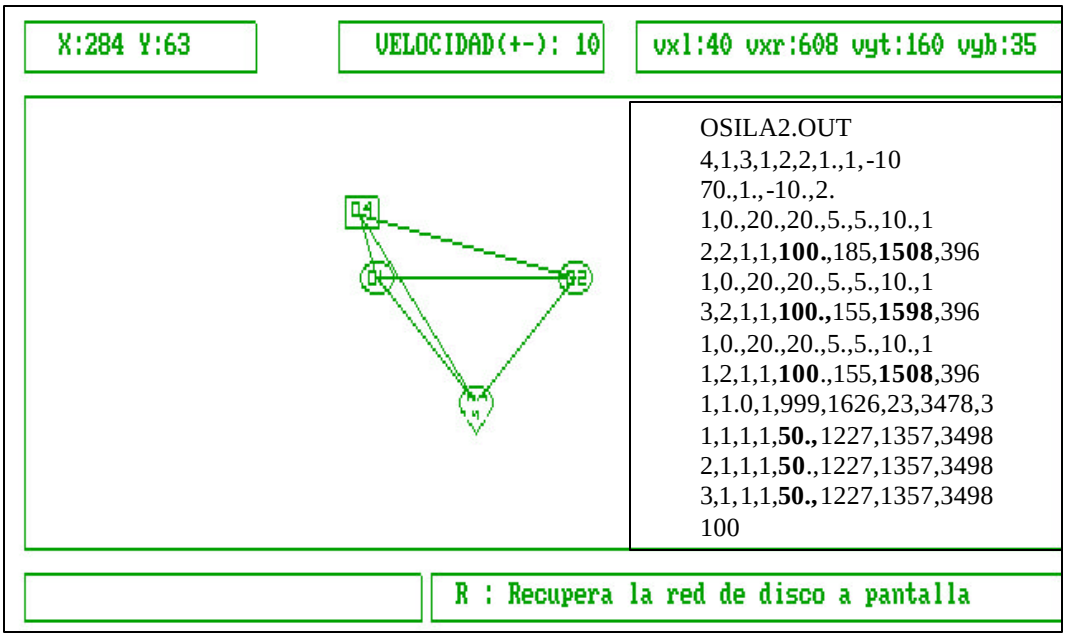


Figura 32. Red neuronal osila 2

Es de notar el manejo de magnitudes en las semillas (verificar valores de salida OSILA2.OUT), de cada neurona, estos resultan la característica primordial en el logro de este tipo de oscilación sincrónica.

Mantener un grupo de neuronas las cuales se disparan simultáneamente, es uno de los osciladores buscados que permite mantener una patrón de pasos en un hexápodo. Aún cuando se hace necesario la búsqueda de oscilaciones sincronizadas, con ráfagas variantes tanto en tiempo, como en el número de salvas.

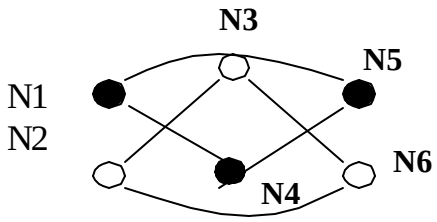
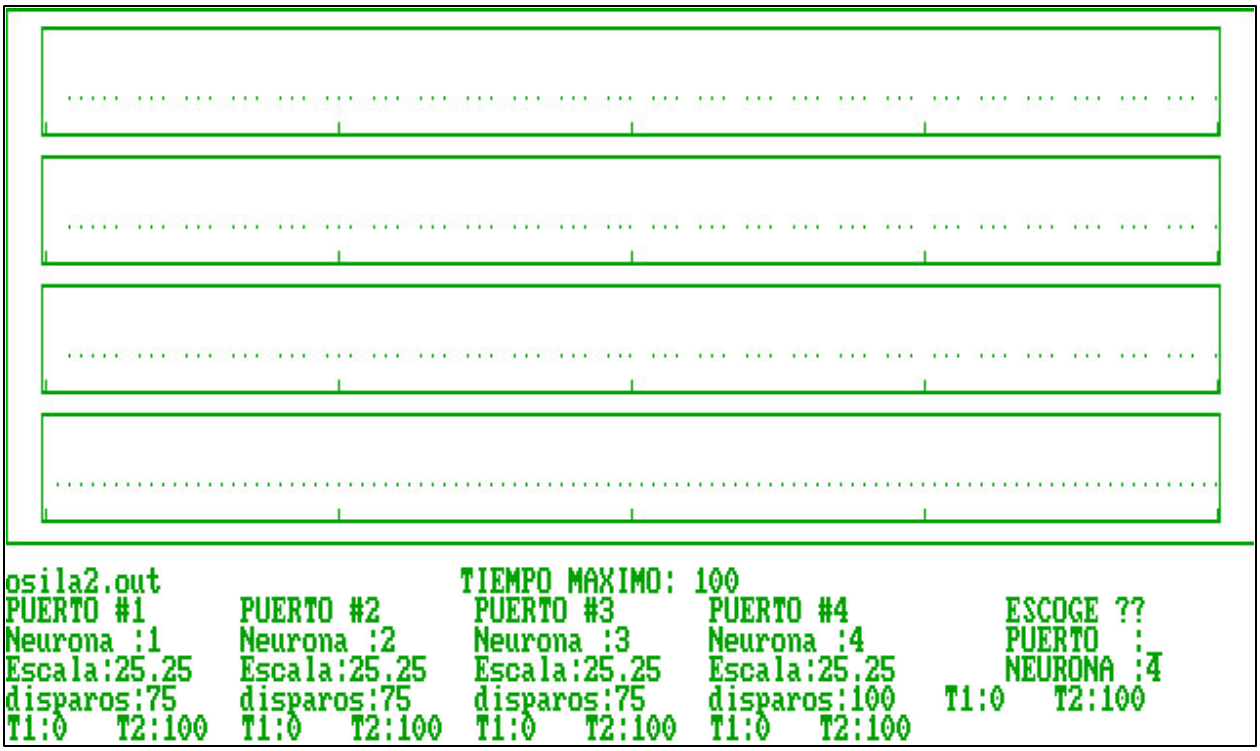


Figura 33. Representación del tripie con neuronas

En la figura 33 se representa una red oscilatoria de tres neuronas, cuyo objetivo es ilustrar la alternancia requerida para formar el tripie, siendo necesario utilizar como ilustración otras tres neuronas. Los valores asignados por NEURORED a las semillas son: 155,1508,396 y son iguales en todas las neuronas de una red. Los valores asignados a las semillas de en esta red son: 185,1508 y 396 para la primera neurona; la segunda y tercer neurona conservan los valores originales.



Gráfica 7. Resultados del oscilador osila2.

h) RED NEURONAL OSILA 3

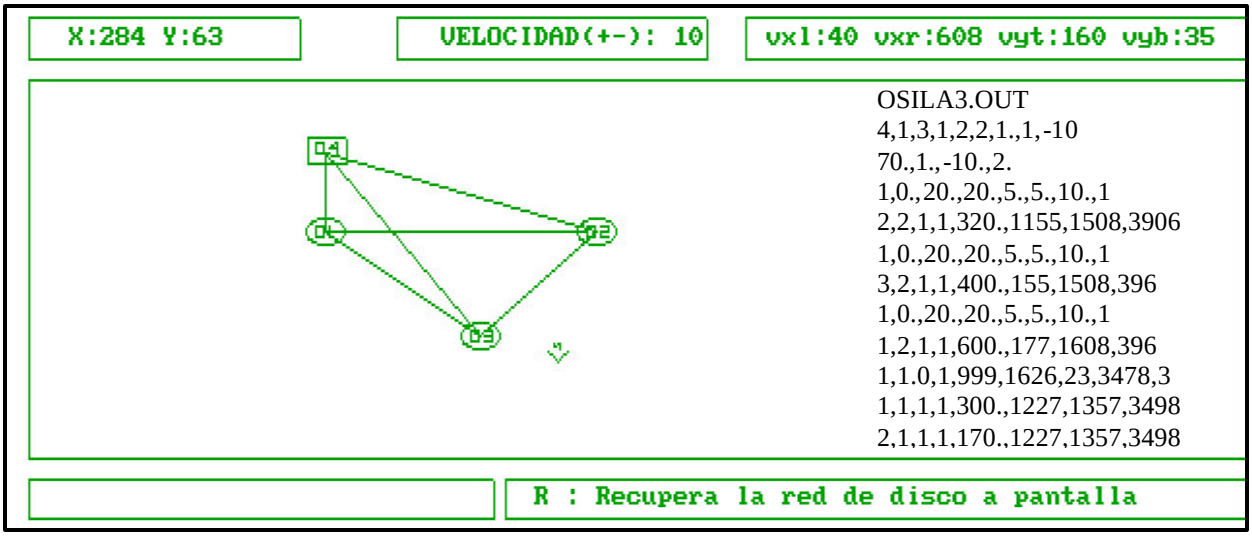
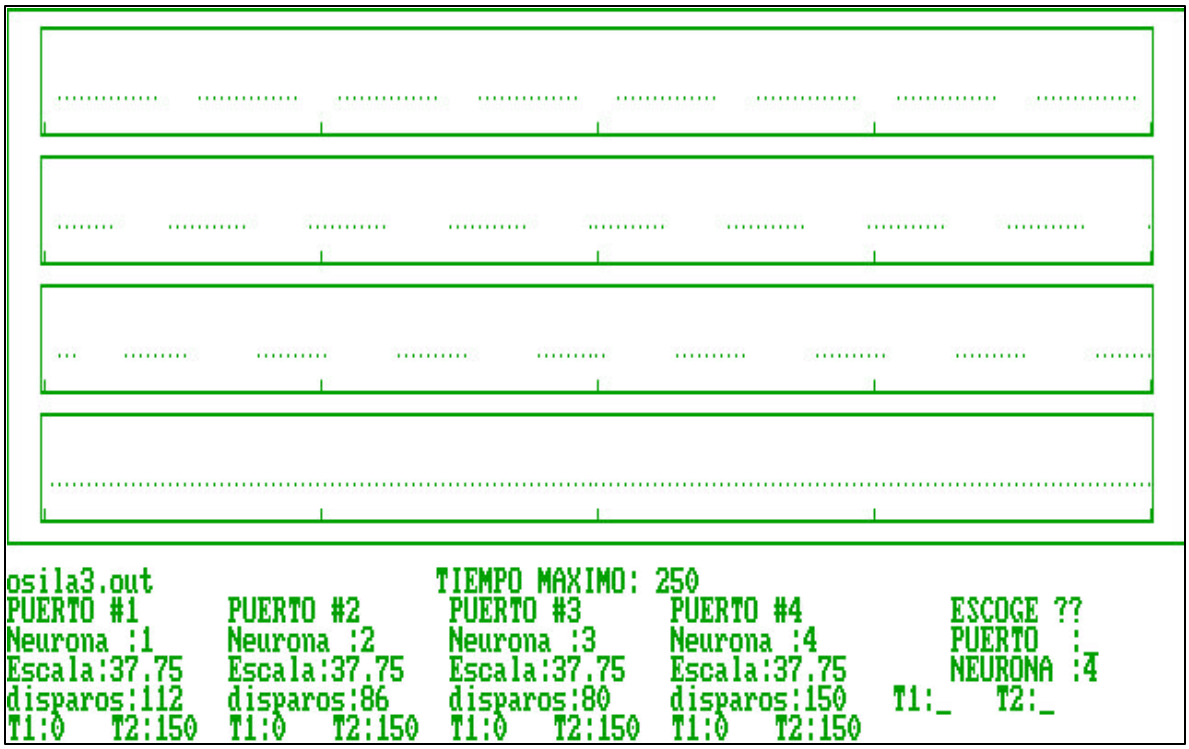


Figura 34. Red neuronal osila3.

Con las características del oscilador anterior, se intenta encontrar osciladores que generen una oscilación simultánea de las tres neuronas y más aún con tiempos de disparo en cada ráfaga más prolongadas. Es en este sentido, se genera la red OSILA3 en la cual se cambian los parámetros

de la intensidad sináptica entre cada neurona, los valores de las fibras, variándose las asignaciones de NEURORED para las semillas. La respuesta de la red es una oscilación en cada neurona, existiendo un traslape pronunciado entre ellas; inclusive se presenta —en un instante pequeño—, entre las tres neuronas, de al menos un disparo.

Asignar una representación a esta red que concuerde con un patrón de pasos tal como el correr en un insecto, es el alcance que se percibe con esta configuración y la nube de valores en sus parámetros. Sin embargo, el número de disparos en cada ráfaga para distintas neuronas no es de una constancia fija. Esto desemboca en un control de movimiento no sincronizado, con tiempos de disparo y acción de cada pata de una manera no coordinada.



Gráfica 8. Resultados del oscilador OSILA3

i) RED NEURONAL OSILA 4

Esta red neuronal es una mejora de la red anterior. Se han modificado la intensidad sináptica, a una de las neuronas se le han cambiado las semillas y la fibra se decrementa manteniéndose la misma para todas las neuronas. La red se comporta como un oscilador y es uno de los mejores resultados obtenidos en esta configuración con ráfagas de disparo constantes y

coordinados para todas las neuronas. Es un oscilador que imita cercanamente a un sistema de locomoción biológico.

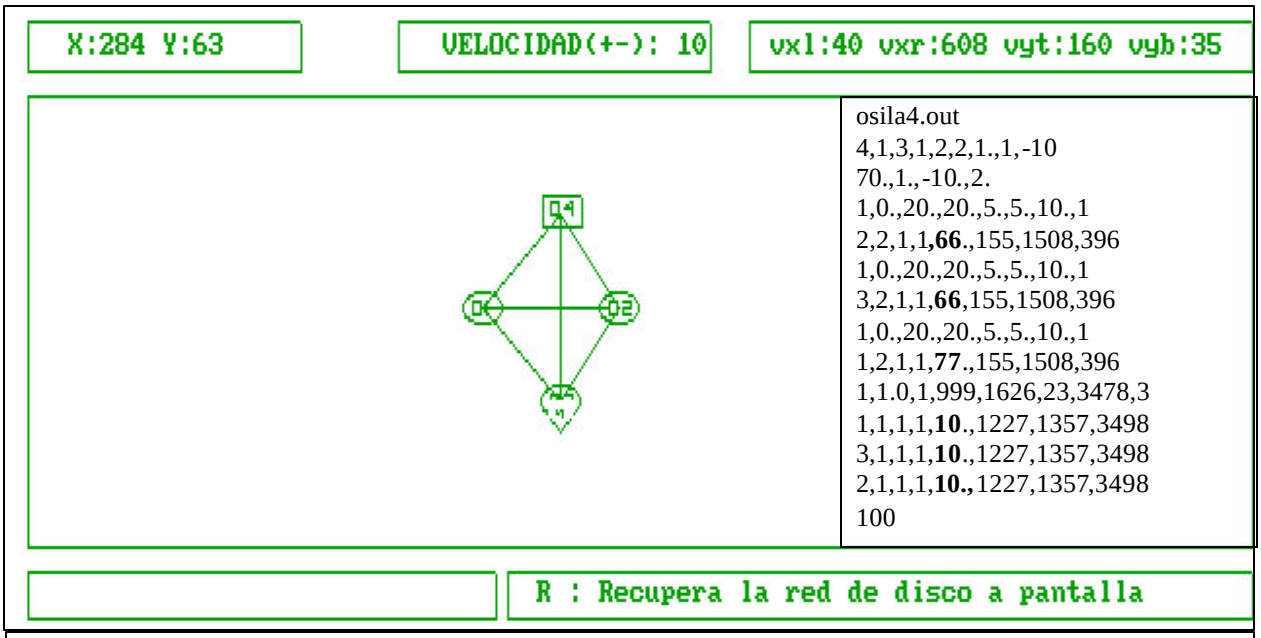


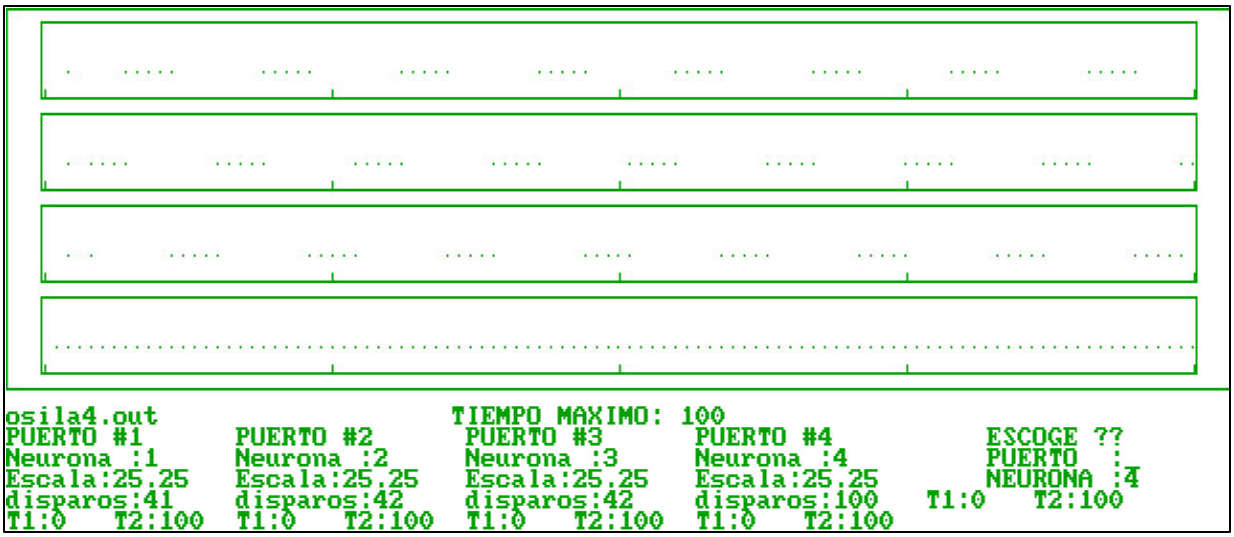
Figura 35. Red neuronal osila 4 y archivo .dat.

TABLA DE RESULTADOS 8			
NOMBRE DEL ARCHIVO	**OSILA 4**		
❖ NOMBRE DE LA NEURONA	**1**	**2**	**3**
❖ INTENSIDAD SINÁPTICA	**-600**	**-320**	**-400**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**10**	**10**	**10**
❖ TIPO DE SINAPSIS	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**0**	**0**	**0**
❖ TIEMPO DE SIMULACIÓN [ms]	**100**	**100**	**100**

4.3.2. INHIBICIÓN MUTUA

j) RED NEURONAL OSCM1

La simulación de redes neuronales de tres neuronas con inhibición mutua, está enfocado a encontrar un patrón de disparo en el cual se logren ráfagas simultáneas en todas las neuronas. Esta característica se basa en el hecho de colocar una neurona activa en cada movimiento de articulación, (como se mostró en la figura 33), en donde la red oscila simultáneamente y este ha



GRAFICA 9. RESULTADOS DEL OSCILADOR OSILA4

sido uno de los resultados obtenidos mediante la red neuronal OSILA2. Sin embargo este resultado no se enfoca en el número de disparos ya que la red es inestable a pequeñas variaciones en los parámetros.

TABLA DE RESULTADOS 9						
NOMBRE DEL ARCHIVO	**OSCM1**					
❖ NOMBRE DE LA NEURONA	**1 hacia 2**	**1 hacia 3**	**2 hacia 1**	**2 hacia 3**	**3 hacia 1**	**3 hacia 2**
❖ INTENSIDAD SINÁPTICA	**-37**	**-39**	**-39**	**-37**	**-37**	**-39**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**40**	**40**	**40**	**40**	**40**	**40**
❖ TIPO DE SINAPSIS	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**0**	**0**	**0**	**0**	**0**	**0**
❖ TIEMPO DE SIMULACIÓN [ms]	**100**	**100**	**100**	**100**	**100**	**100**

Los osciladores de tres neuronas, resultaron ser arreglos neuronales de mejor desempeño e hicieron posible la obtención de ráfagas de disparos sincronizados de diversos anchos de disparos; así la salida OSCW2.OUT muestra ráfagas de mayor duración en términos del número de disparos. Un hecho importante es el peso de inhibición mutua entre cada neurona, estas mantienen una cercanía de valores entre dos neuronas juntas.

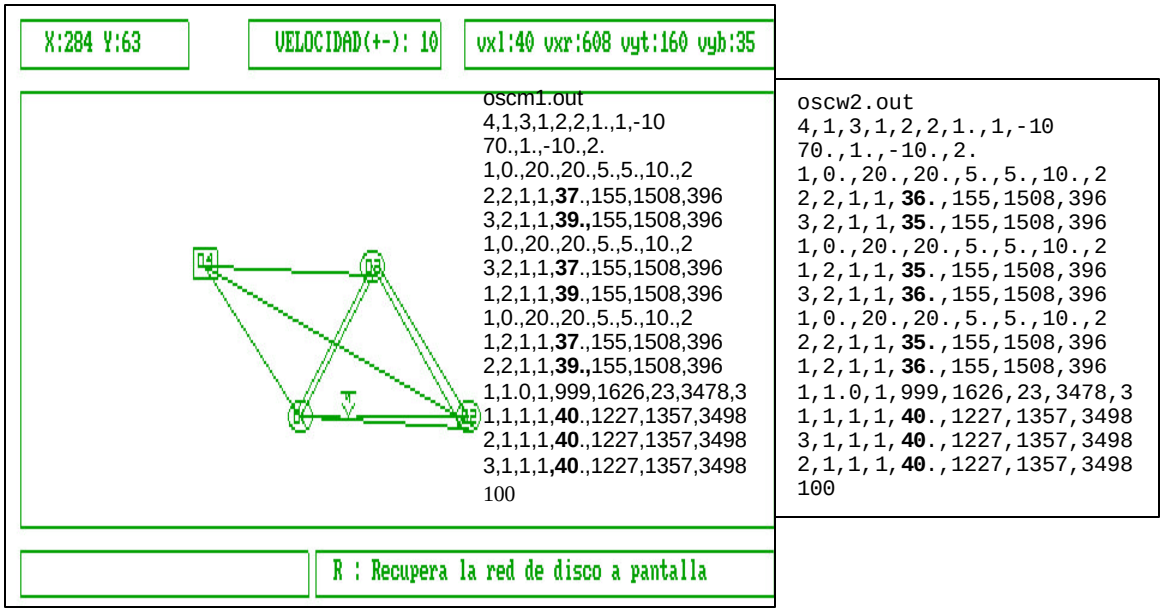
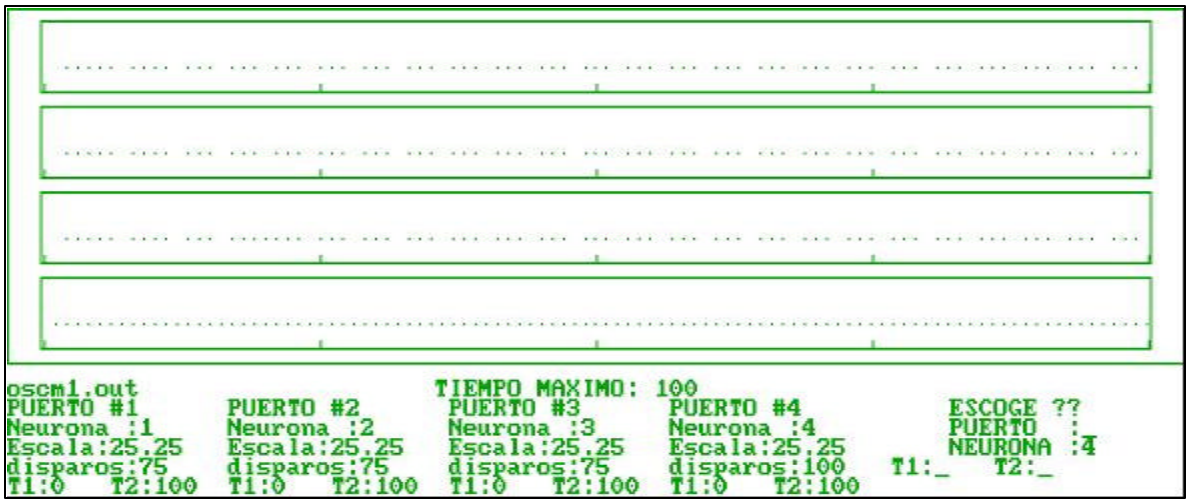
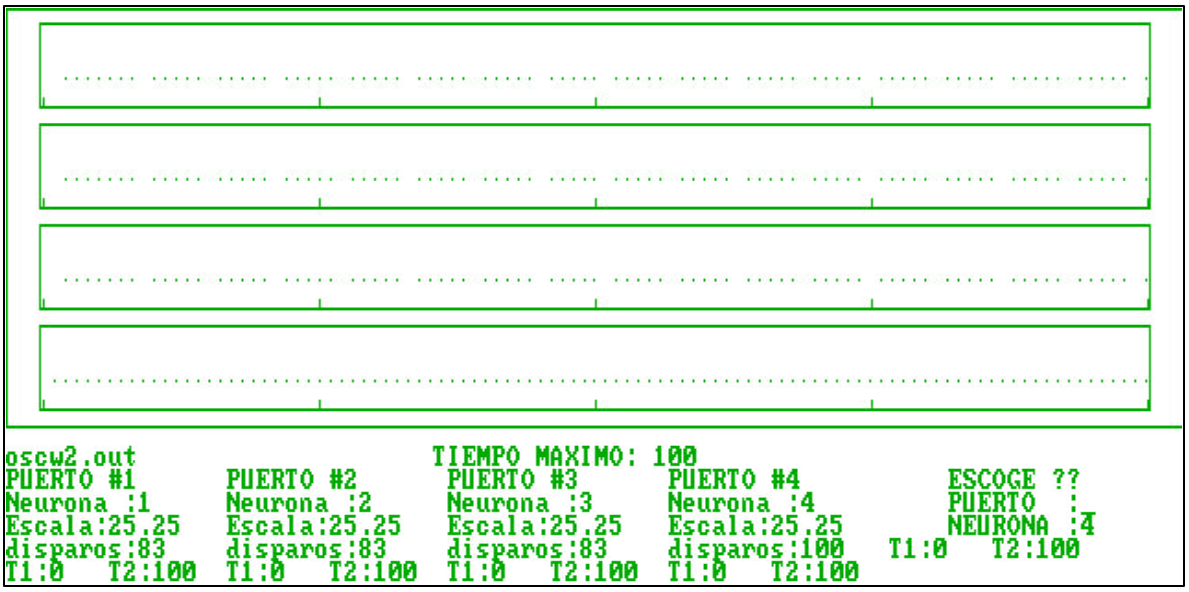


Figura 36. Red neuronal osm1



Gráfica 10. Resultados del oscilador oscm1



Gráfica 11. Resultados del oscilador oscw2

4.4. OSCILADOR DE CINCO NEURONAS

4.4.1. INHIBICIÓN CÍCLICA

k) RED NEURONAL OSC6

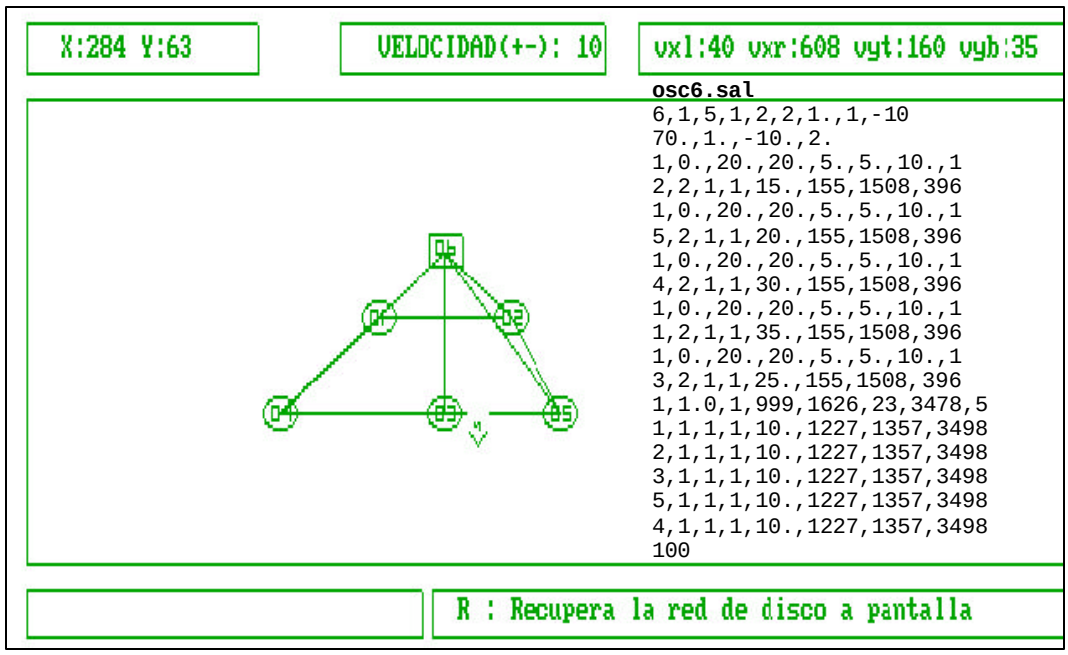
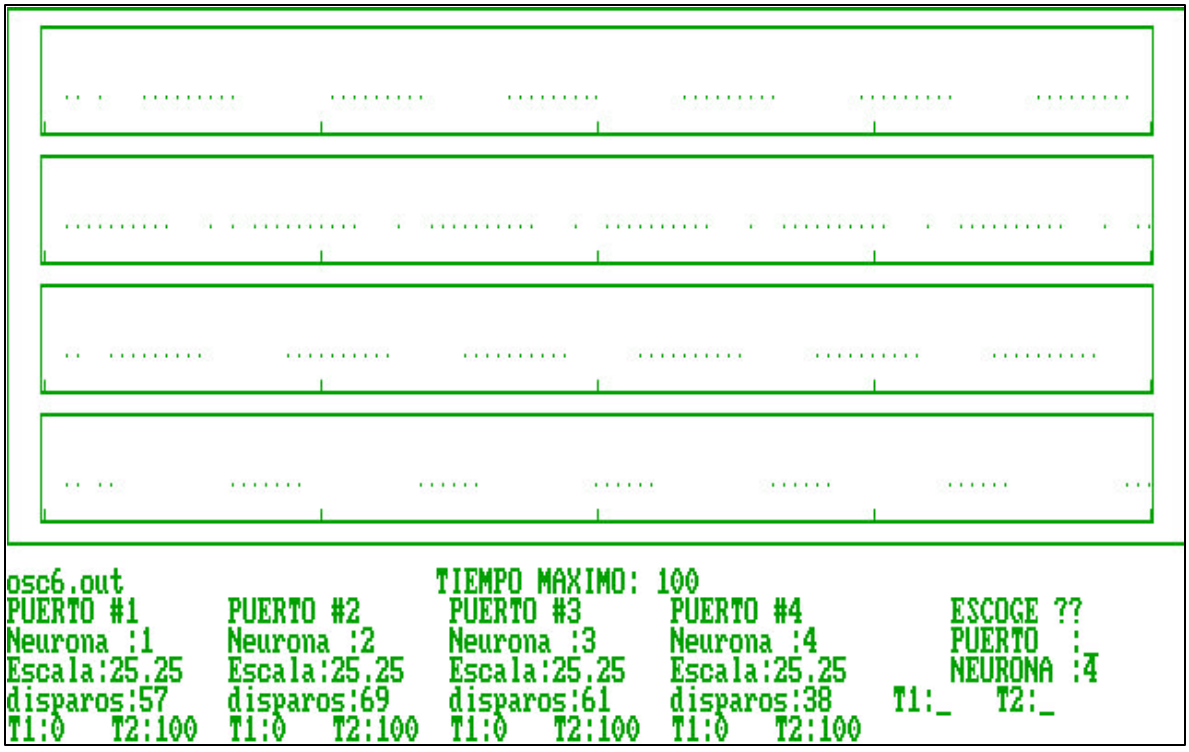


Figura 37. Red Neuronal OSC6

Con este oscilador se busca encontrar distintos comportamientos oscilatorios, ya sea en tiempos de disparo entre cada neurona, o bien tamaños de las ráfagas.

Esta red neuronal presenta una conexión entre neuronas de manera circular, con pesos en las conexiones de un mismo valor y la fibra que los alimenta es del mismo peso para todas las neuronas. El numero de neuronas está basado en tipos de oscilación alternativos a las demás redes y no como formas de activación de cada motor asimilado en las articulaciones del hexápodo.

El umbral de disparo es el determinado por NEURORED y con ello se logra una oscilación en cada neurona; sin embargo, los tamaños en el número de espigas en cada ráfaga, varía para cada neurona presentándose una especie de oscilación no sincronizada.



Gráfica 12. Tren de pulsos de la red OSC6

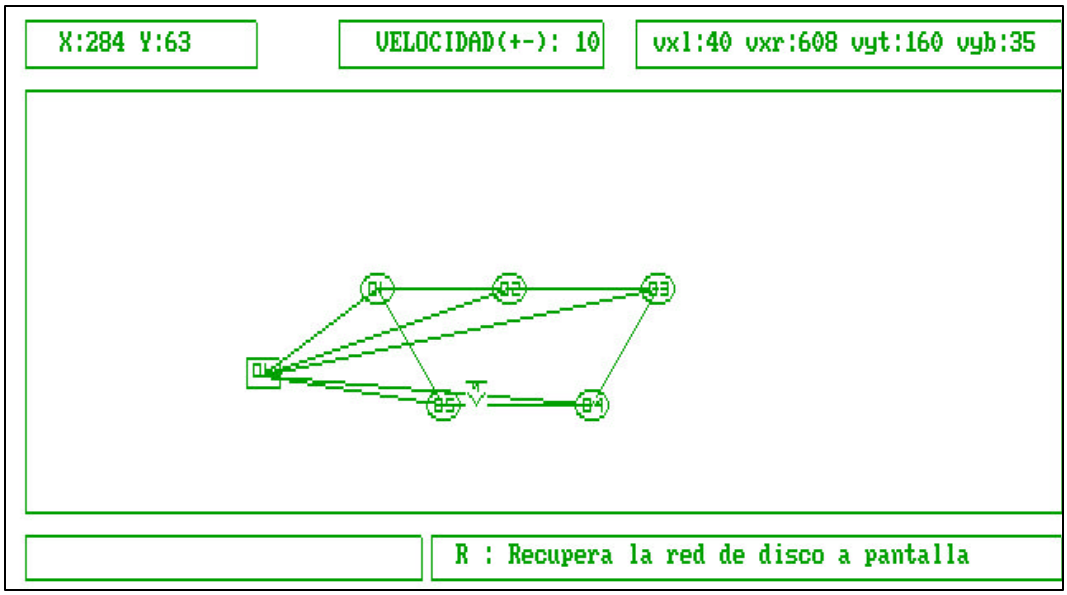
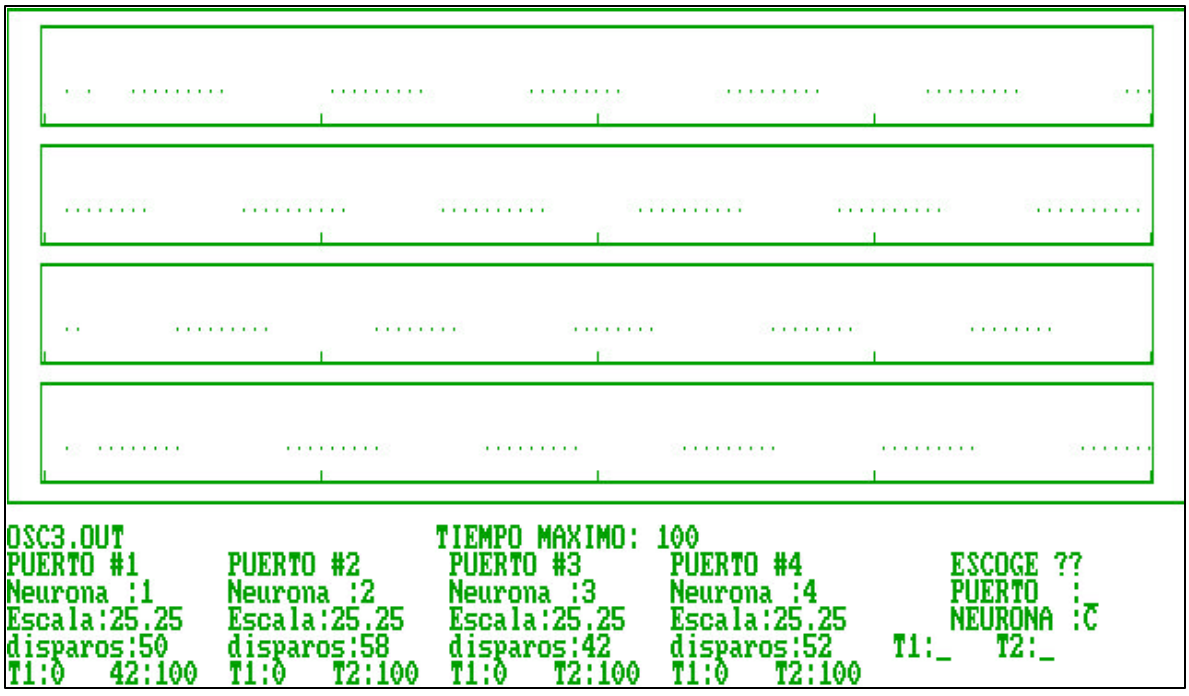


Figura 38. Red Neuronal OSC3

OSC3 es un oscilador similar al anterior, pero con distintos pesos entre cada neurona y los resultados obtenidos son ráfagas de aproximadamente el mismo número de disparos, existiendo un traslape en cada ráfaga del 50 %, tal como muestra la salida del archivo OSC3.OUT.



Gráfica 13. Tren de pulsos oscilador OSC3

4.5.OSCILADOR DE SEIS NEURONAS

El modelo del robot en esta etapa contiene seis motores (tres en cada lado del robot), uno en cada pata, lo que da como resultado seis neuronas en la red neuronal. A partir de esta distribución de los motores, se implementaran redes que contengan una neurona por motor. El objetivo no ha cambiado, lo que se debe encontrar es el patrón de pasos que forme un tripie. Esto se logrará haciendo que tres neuronas oscilen al mismo tiempo mientras la restantes permanezcan apagadas.

4.5.1. INHIBICIÓN CÍCLICA

l) RED NEURONAL PO2

Los valores que con los cuales se constituyo la red, fueron incrementándose de dos en dos.

TABLA DE RESULTADOS 4						
NOMBRE DEL ARCHIVO	**PO2**					
❖ NOMBRE DE LA NEURONA	**1**	**2**	**3**	**4**	**5**	**6**
❖ INTENSIDAD SINÁPTICA	**110**	**92**	**94**	**98**	**102**	**106**
❖ INTENSIDAD DE LA FIBRA HACIA LA NEURONA	**15**	**15**	**15**	**15**	**15**	**15**
❖ TIPO DE SINAPSIS	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**	**INHIBICIÓN**
❖ UMBRAL	**0**	**0**	**0**	**0**	**0**	**0**
❖ TIEMPO DE SIMULACIÓN [ms]	**100**	**100**	**100**	**100**	**100**	**100**

Con esta red neuronal busca obtener un patrón de oscilación que sea alternativo a las obtenidas mediante tres y dos neuronas. Tal modelo busca controlar o disparar una ráfaga de disparos que acentúe el accionar de cada articulación del hexápodo. En este sentido se presenta una conexión de carácter unidireccional. Todas las neuronas se conectan formando un ciclo y en una sola dirección, manteniéndose distintos pesos de conexión entre neuronas y con un peso de excitación fija, de la fibra a las neuronas.

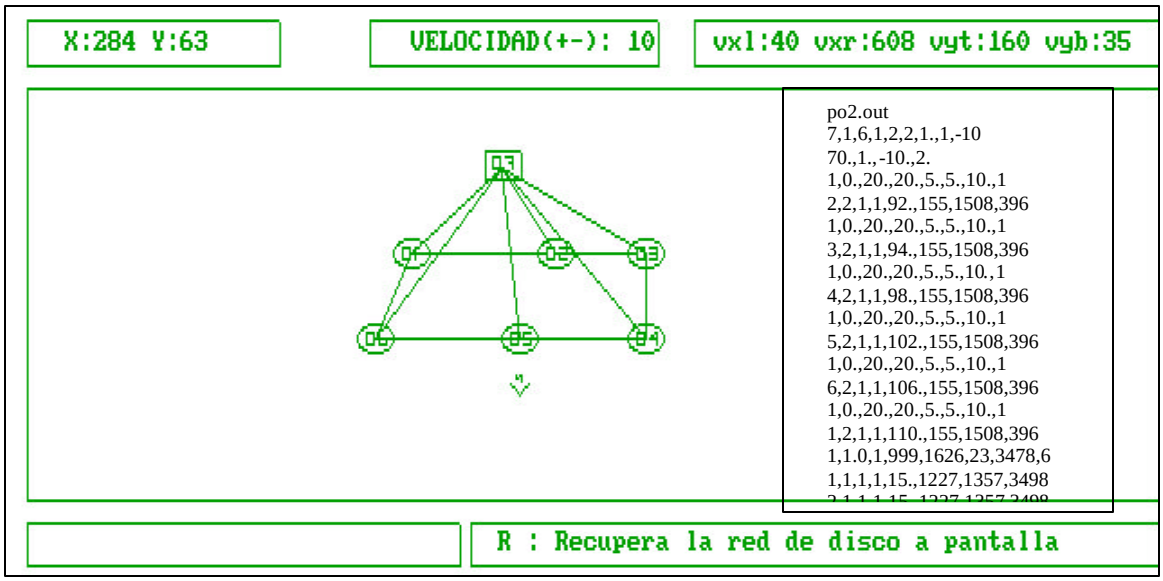


Figura 39. Red neuronal PO2

Gráfico 14. Tren de pulsos de la red neuronal PO2

4.5.2. INHIBICIÓN MUTUA

m) RED NEURONAL NPo3

Es un oscilador de conexión bidireccional o con inhibición mutua y que además mantiene una malla cerrada. Las respuestas observadas en cada una de las neuronas no son del tipo oscilatorio, puesto solo ciertas neuronas tienden a dispararse y las otras se inhiben. Esta es una característica ya observada en redes anteriores en los que alguna de las neuronas inhibe totalmente a otras para el caso de dos neuronas o más.

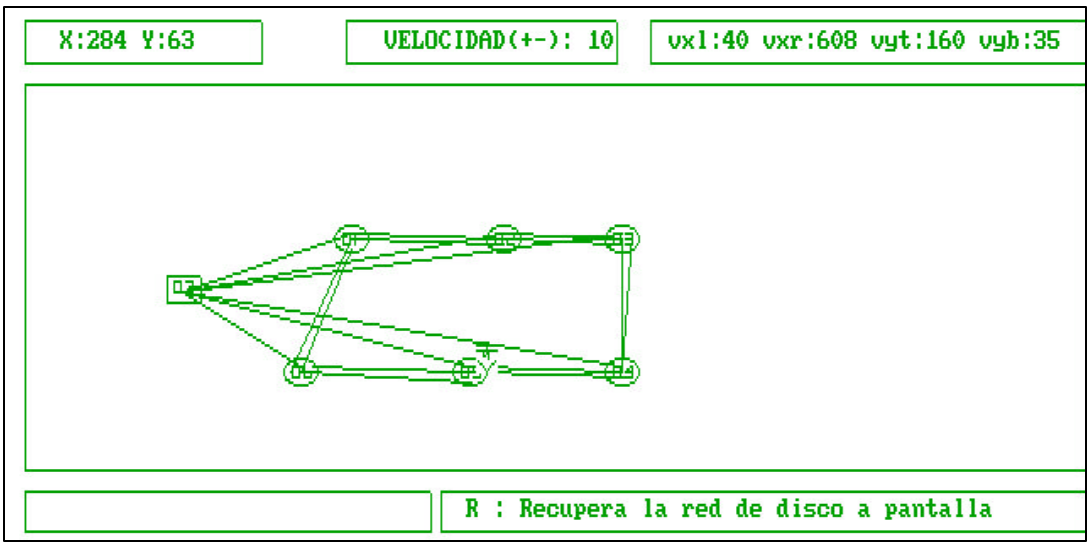
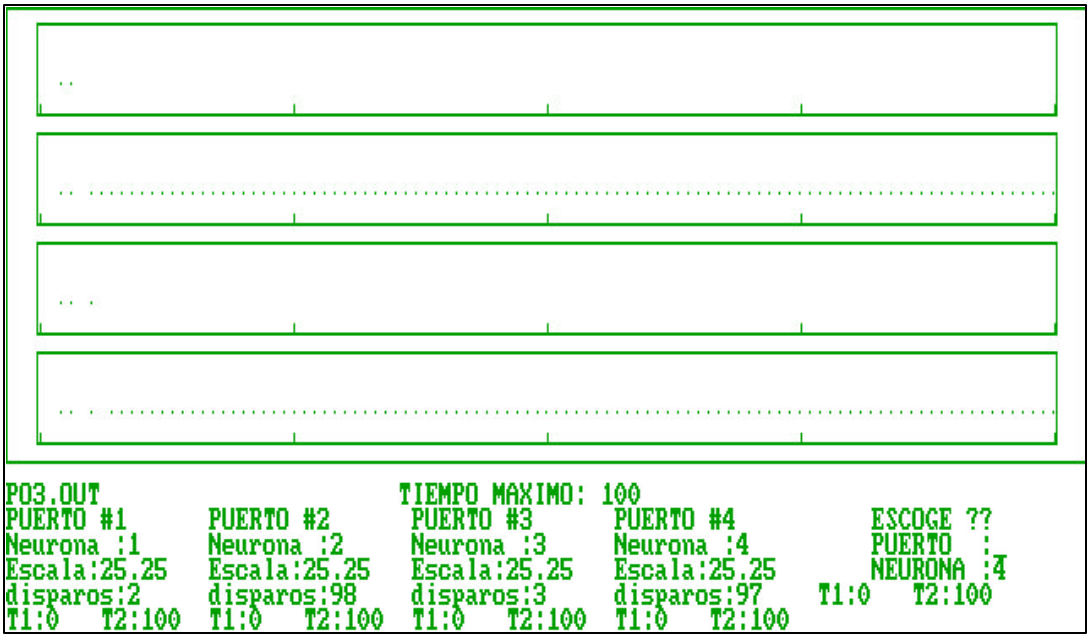


Figura 40. Red neuronal PO3



Gráfica 15. tren de pulsos de la red neuronal PO3

n) RED NEURONAL Po4

Siguiendo con la topología de seis neuronas, se ha agregado un lazo interno a la red, con lo cual se conforman dos mallas cerradas. Sin embargo, la malla interna únicamente inhibe en una sola dirección, a diferencia de las demás conexiones entre neuronas. En este sentido, las respuestas de las neuronas no son similares y no caracterizan un sistema oscilatorio. Solo se observa una pulsación casi imperceptible en la neurona 4, que oscila en pares y tercias de disparos de manera repetidas. Los hay que permanecen inactivos como las neuronas 1,2 y 3.

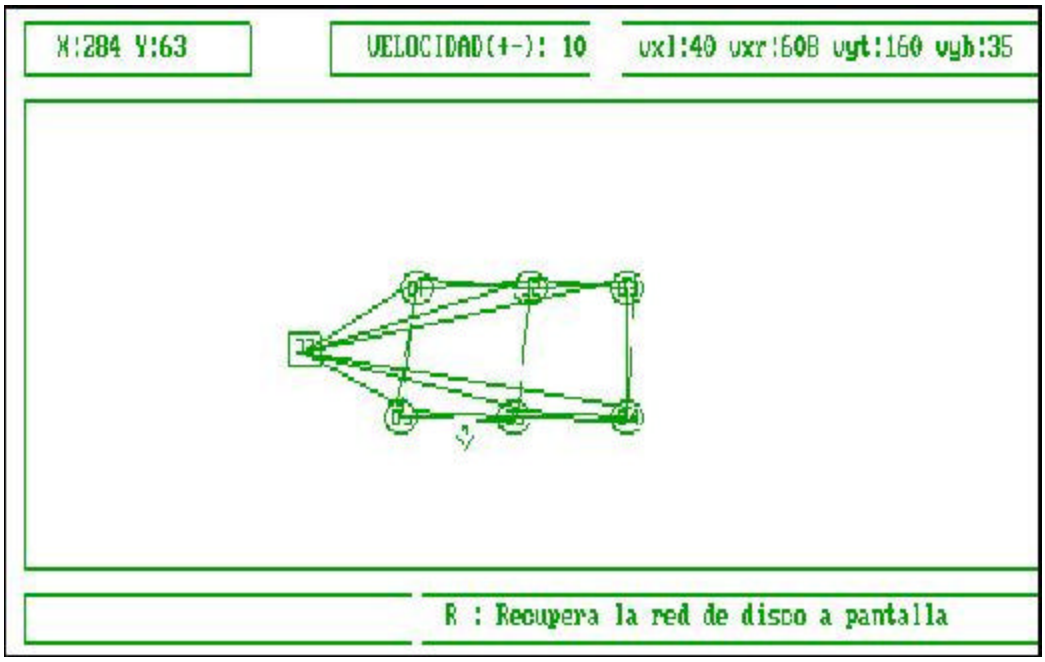
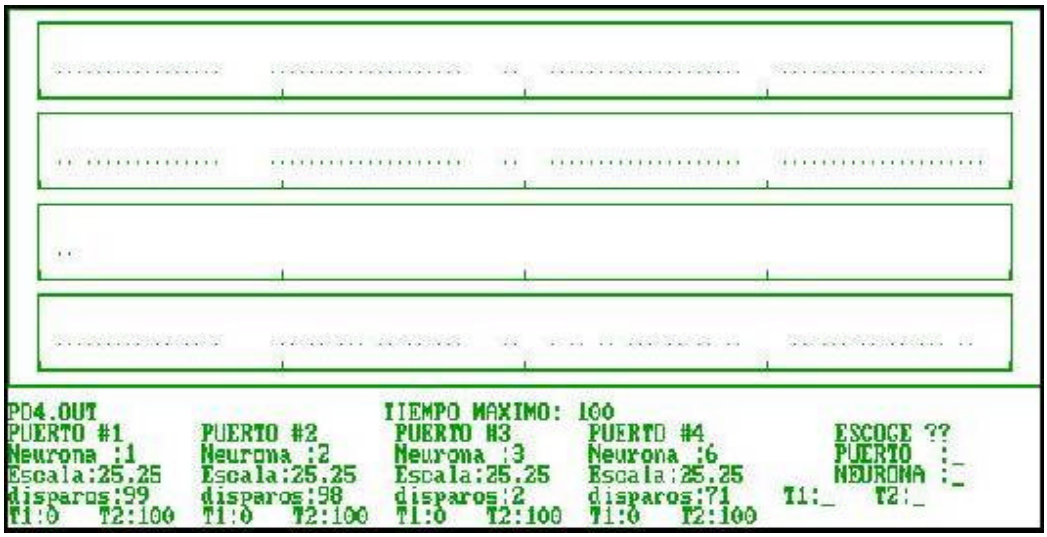


Figura 41. Redes neuronales PO4 y OSCN1



Gráfica 16. tren de pulsos de la red neuronal PO4

o) RED NEURONAL Po6

La siguiente red neuronal está constituida por una maya central que inhibe mutuamente a las neuronas centrales. La red que consta de seis neuronas mantiene una semejanza a la red utilizada por Randal Beer en el control de un hexápodo. Se busca disparar en forma constante y oscilatoria cada una de las neuronas con el fin de simular motores asignados en cada pata.

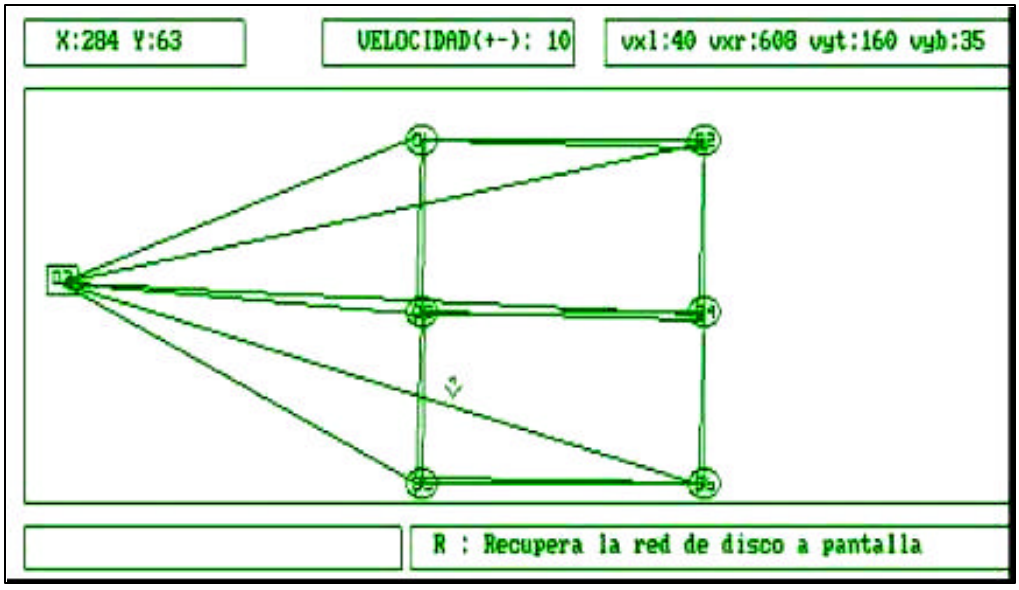
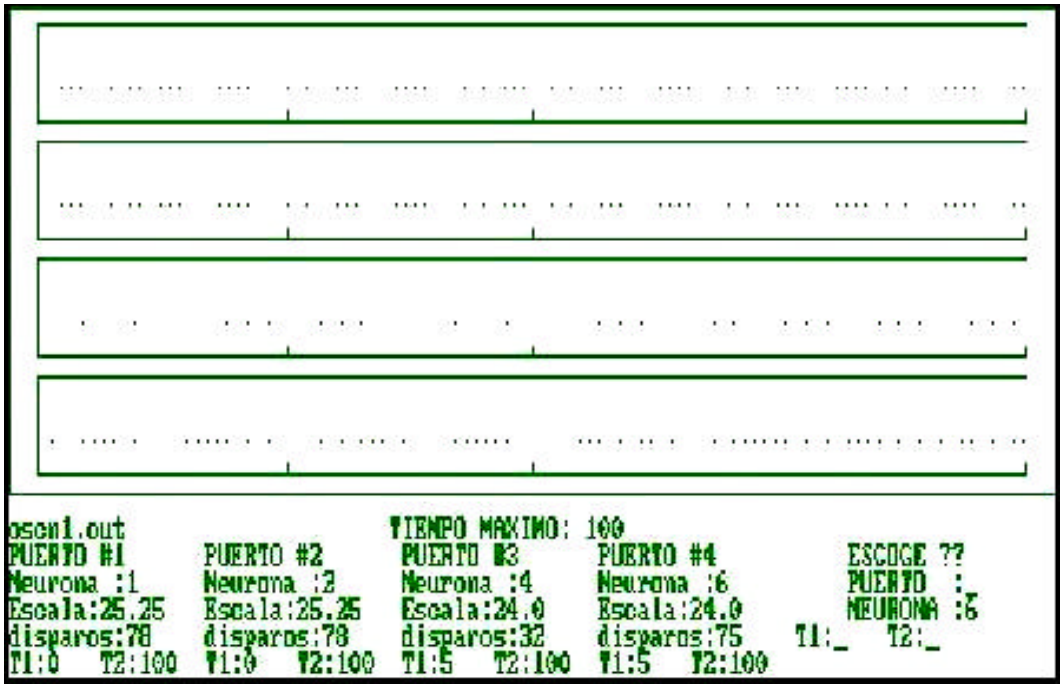


Figura 42. Red neuronal PO6



Gráfica 17. Tren de pulsos de la red neuronal OSN1.

Hasta aquí se han mostrado las simulaciones más representativas y con comportamientos diferenciables. No es necesario representar todas y cada una de las simulaciones realizadas, puesto que no es el objetivo de este proyecto. Únicamente se hace el esbozo de los que logran la característica oscilatoria y que de algún modo definen el trayecto recorrido en la búsqueda del oscilador adecuado.

Se han encontrado osciladores que representan patrones de locomoción. Los osciladores de dos neuronas ny*.*, redes muy simples de dos neuronas, han presentado oscilaciones en todas ellas con ráfagas de disparo distintos. Difieren en anchos de disparo y pequeños comportamientos como una desaceleración en cada ráfaga. Se consideran estos osciladores representativos de patrones rítmicos y para un sistema de seis patas, caso del robot construido, se le puede implementar tres osciladores de forma independiente.

En las redes de tres neuronas, se hace difícil mantener un oscilación simultánea entre las neuronas si las conexiones sinápticas son de forma cíclica en una sola maya (ver osciladores: osila1, osila3. osila4). Sin embargo, se comportan como excelentes sincronizadores cuando la conexión sináptica entre neuronas es mutua. Esto se puede observar en los osciladores oscw2 y oscm1. Estos osciladores se han considerado para la programación de un patrón de locomoción del tipo trípode. Con un solo oscilador se puede implementar este comportamiento, si se considera el primer ciclo de paso como red encendida y el siguiente, la red apagada.

En las redes de mayor número de elementos, de cinco y seis neuronas, la observación y control de los parámetros se hace más compleja. Los archivos *.dat de los osciladores OSC6, PO2, son cada vez mas largos y las relaciones entre cada neurona se hace más difícil. Los resultados obtenidos observables en los graficos *.dots, describen la dificultad de obtener pulsos rítmicos más controlados. Las ráfagas llegan a traslaparse de forma aleatoria. En general, es de observarse que en las redes más sencillas, la oscilación es más estable y controlado.

CAPITULO V

PROGRAMACIÓN DE LOS OSCILADORES

En este apartado, han de definirse las condiciones que ha de guardar el robot en la estructura de programación, considerando las condiciones iniciales, y no se referirá a explicaciones sobre el lenguaje de programación NQC, del cual se habla en el apéndice B.

5.1 ACONDICIONAMIENTO DE LAS SEÑALES DE ENTRADA

El robot cuenta con sensores de tacto colocados en cada pata, los cuales determinarán la lógica y posición de las patas, así cómo tres sensores de luz que se encuentran en el frente del robot. Podemos considerar los sensores como fibras de la red neuronal, ya que de su valor depende la ejecución del programa que coloque a las patas en una posición adecuada.

Debido a que el modulo RCX solamente tiene tres salidas, se acondicionó en la programación el modo llamado *RAW*, con el cual se puede obtener el valor de dos sensores conectados al mismo puerto de salida. En este caso se tienen sensores de luz y de tacto que son diferenciados por el valor de su lectura. Desgraciadamente en el modo *RAW* el sensor de luz depende de la condición de tacto, ya que si el sensor de tacto esta presionado enviará un "valor en bruto" menor a 10 de lo contrario se obtendrá el valor del sensor de luz que siempre estará arriba de 800. El sistema mecánico del robot se adecuó para que el microprocesador que controla los motores reaccione de acuerdo al estado que guardan los sensores. De este modo, en todos los programas se inicia con un posicionamiento adecuado de las patas, lo que significa colocar a los sensores de tacto en el estado que se requiera y de acuerdo al movimiento que ha de realizarse.

Son tres los sensores de luz colocados en el frente del robot. La razón de tener éste número de sensores es debida a la inestabilidad en los valores de intensidad de luz que proporciona la utilización de un solo sensor. Así, si el sensor detecta un obstáculo el valor de entrada al microprocesador será el promedio de los tres sensores, teniendo de esta manera un valor de intensidad de luz con menor incertidumbre. El rango de los valores será de 800 a 900 dando pauta a la ejecución del programa que genera el movimiento en reversa, hasta que el sensor no detecte ningún obstáculo enfrente, regresando al valor sensado en el intervalo de 0 a 10. Esto le indicará al microprocesador la ejecución de la parte del programa que contiene el oscilador que genera el movimiento de caminado hacia el frente, hasta que se detecte un nuevo obstáculo. Para

comprender claramente las acciones que el robot toma dependiendo del valor de los sensores de luz se muestra la tabla 11.

TABLA 11

INTERVALO DE VALORES DE LOS SENSORES DE LUZ Y TACTO	ACCIÓN A REALIZAR
0 – 10 (SENSOR DE TACTO)	Se ejecuta el programa de oscilación. La pata realiza movimiento oscilatorio.
830 – 890 (VALOR PROMEDIO, SENSORES DE LUZ)	Detección de obstáculo.

El siguiente fragmento de código es una subrutina o tarea, que ejecuta el posicionamiento de los sensores en las condiciones indicadas. Estas subrutinas permiten que el robot sea capaz de posicionarse y ajustarse al movimiento buscado, independientemente de las condiciones iniciales que guarde. Esta ventaja de autoposicionamiento y acondicionamiento de las patas, es permitida por la integración de los sensores de tacto. Se consideraron estos casos, dado que los motores no son de pasos y se hace complejo un control de posición, o el diseño de un sistema de control más robusto.

```
void  posicionate_en_0 ()
    {
         OnFwd(OUT_A+OUT_B+OUT_C); //enciende motores
         Wait(25); //hasta que los sensores sean >10.

while((SENSOR_1<10)||(SENSOR_2<10)||(SENSOR_3<10))
       {

         if(SENSOR_1>10){Off(OUT_A);Wait(50);}
         if(SENSOR_2>10){Off(OUT_B);Wait(50);}
         if(SENSOR_3>10){Off(OUT_C);Wait(50);}
         SendMessage(2);
         Wait(100);
        }

      }//fin subrutina posicionate
```

EJEMPLO 1. POSICIONAMIENTO DE LOS SENSORES EN CONDICIONES INICIALES

Estas rutinas de acondicionamiento se encuentran tanto en el RCX maestro como el esclavo, considerando que ambos controlan tres motores y están expuestos a las mismas condiciones. Sin embargo, el esclavo ha de estar sometido al designio del RCX maestro.

Los sensores de tacto funcionan al momento de que dos placas internas independientes entre si son presionadas, cerrando de esta forma el circuito, permitiendo el paso de la corriente entre ellas. La señal viaja a través de cables que están conectados a los puertos de entrada del bloque RCX. En la implementación del robot hexápodo se calibraron nuevamente los sensores, haciéndolos más sensibles al contacto, por medio de un pedazo de papel de aluminio colocado en la barra que provoca el contacto con las placas internas. La corriente que alimenta a los sensores proviene de las mismas salidas del microprocesador alimentándolos a través de los cables de conexión.

De la misma forma, el sensor de luz recibe una corriente por el cable de conexión al RXC. El sensor emite un haz de luz infrarroja que rebota en los objetos y es recibida por el sensor, transformándola en un valor numérico que es enviado hacia el microprocesador a través de los cables.

5.2 TRANSFERENCIA DE DATOS

Los programas generados en la computadora son transmitidos al microprocesador contenido en el RCX, a través de un dispositivo que emite y recibe luz infrarroja conectado al puerto serie de la computadora.

El RCX recibe la señal en su puerto de infrarrojos y de ahí el programa es almacenado en

```
void revisa_sensores()
{
   while(SENSOR_1==0||SENSOR_1==1)
   {
    if(SENSOR_1==0&&SENSOR_3==1&&SENSOR_2==0)
//010 ==001
     {
       ClearMessage();//borra mensage en buffer.
       SendMessage(1);//envía mensage
       Wait(100);
       until(Message()==101); //espera mensage
del esclavo
       OnFwd(OUT_C);  // motor C en On
       Off(OUT_A+OUT_B); //motor A y B en Off.
       until(SENSOR_3==0);
       Off(OUT_C);
     }//cierra if
```

A

```
while(SENSOR_1==0||SENSOR_1==1)
{
     ClearMessage();
     until(Message() !=0);

  if(Message()==1)
//si mensage  es 1.
    {
      posicionando_101();

until(SENSOR_1==1&&SENSOR_3==0&&
SENSOR_2==1);
      SendMessage(101); //envia
respuesta al maestro.
      Wait(10);
      OnFwd(OUT_A+OUT_B);
      Off(OUT_C);
```

B

EJEMPLO 2. ENVÍO Y RECEPCIÓN DE MENSAJES. A) MAESTRO. B) ESCLAVO

la memoria del microprocesador, donde se ejecutará dependiendo de las condiciones de los sensores. Cabe mencionar que además de recibir señales el puerto infrarrojo también emite señales, lo que permite enviar información a la computadora o a otro bloque RCX, cosa que fue útil en nuestra investigación. Los mensajes que se enviaban entre bloques RCX lograban una sincronía en las decisiones tomadas por el RCX que cumplía la función de maestro con el RCX esclavo, generando de esta forma un movimiento estable que permitía al robot desplazarse sin contratiempos en la formación del tripie.

El recuadro A es el programa que esta contenido dentro del RCX maestro y el recuadro B contiene el programa contenido en el RCX esclavo. Estos pedazos de programas tienen como objetivo el posicionar las patas en una condición inicial para de ahí comenzar el movimiento. El programa maestro depende de la señal de los sensores de tacto para poder ejecutar el programa, mientras que el programa esclavo depende, además de sus sensores, de la señal que se le envía desde el RCX maestro. Una vez que se ha ejecutado el programa esclavo, éste manda nuevamente una señal al RCX maestro para informarle en que parte del programa esclavo se encuentra y así el RCX maestro pueda tomar una decisión del paso a seguir.

5.3 PROGRAMANDO UN PATRÓN DE PASOS

Como todo ser vivo, el comportamiento de los insectos se basa en función de las necesidades requeridas por el medio ambiente o las que él mismo genere. Sobre la base de este principio se han obtenido diferentes patrones de pasos como el de caminar lento, paso corto o paso rápido (figura 23). El patrón de pasos que se determinó programar fue el de caminado rápido. Se tomo esta decisión debido a que los patrones están definidos de tal manera que su obtención en NEURORED es posible, así como en la programación.

Al terminar de compilar los datos de la red neuronal, NEURORED arroja una serie de unos y ceros que representan los disparos entre neuronas de una misma red. A este archivo se le conoce como archivo de salida y son estos datos los que se tomaron para realizar el programa. Se consideró el número unos o ráfagas como el tiempo en que la pata del robot esta realizando el movimiento y los espacios en blanco o ceros como el tiempo en que la pata esta tocando una superficie.

De los osciladores mostrados en el capítulo IV se analizaron aquellos que presentaban o se asemejaban al patrón de pasos mencionado anteriormente como: oscila4, oscm1.

Antes de programar el oscilador elegido, se realizaron programas que permitían comprobar parte por parte las herramientas que se emplearían durante el programa. El primer programa hecho fue el que controlaba a los motores y su velocidad, utilizando un prototipo en el cual además contenía los sensores de tacto. Los problemas presentados permitieron entender que los sensores debían estar calibrados con el fin de hacerlos más sensible.

Una vez librado este problema se realiza un programa en el cual se coloca al robot en una posición inicial, lo que requiere comprobar la comunicación entre robots. Se debe recordar que en un solo RCX están dos motores sincronizados y el tercero que forma el tripie se encuentra en el segundo RCX, por lo que es indispensable poder tener una comunicación entre RCX. Por medio de pausas entre el envió de los mensajes se resolvió el inconveniente, generando movimientos en los motores previamente programados, formando de esta manera el tripie buscado. En el ejemplo 2 se muestra parte de este programa.

El siguiente paso es la implementación de los sensores de luz y la generación de los pulsos obtenidos en el archivo de salida(*.OUT) de NEURORED. Al agregar los sensores de luz el programa se complica, no se tiene una lectura en cada instante de tiempo, pero gracias a la velocidad en la cual se desplaza el robot es posible utilizar su valor que determina si el robot continua o se detiene por haber detectado un obstáculo. Habiendo acondicionado las señales del infrarrojo se determina utilizar el reloj del microprocesador para generar los unos así mismo para generar los ceros.

El código siguiente ejecuta la oscilación de una sola neurona, la salida es visualizado en el encendido y apagado del motor. Los tiempos de oscilación o de disparo, así como el número de disparos en cada ráfaga, se controlan utilizando un *timer*, para cada neurona. En cada disparo, el motor se enciende por un tiempo de 20 mseg. La activación de cada moto-neurona se realiza en forma de tareas independientes, estos son controlados por las variables x, “y” y z, respectivamente para cada neurona. Cada una de las variables puede tomar tres valores entre 0 y 3. Si la variable de control vale cero, la moto-neurona genera ceros y en consecuencia el motor se apaga. Para los valores de 2 y 3, la neurona se dispara en ráfagas, sin embargo el motor avanza con variable igual a cero y retrocede cuando la variable toma el valor de 3. La elección de estos valores para la variables de control se ha designado de manera arbitraria.

Con estas variables de control se pueden generar diferentes patrones de locomoción, estos programas buscan generar los pulsos de oscilación de las simulaciones obtenidas desde NEURORED.

```
task    MOTO_NEURONA_A() //tarea que ejecuta el estado de la neurona.
   {
         n1=-3; //intervalo de disparo entre cada salva.
         ClearTimer(0); //inicializando timer.
        while(Timer(0)<=1000&&SENSOR_1<100)  //condición de oscilación.
         {
           if(n1+3==Timer(0))  //tiempo de disparo.
             {
               n1=n1+3;
            if(x>=1&&x<3)  // PARA X=1 la neurona se dispara con motor adelante
                 { OnFwd(OUT_A);
                   Wait(20);
                   Off(OUT_A);
                  }
             if(x<1)     // PARA X=0 la neurona no se dispara y el motor apagado.
                 {
                   Off(OUT_A);
                   Wait(20);

                                }
             if(x>=3)  //PARA  X=3 la neurona se dispara con motor en retroceso.
                 {
                   OnRev(OUT_A);
                   Wait(20);
                   Off(OUT_A);
                 }

                  }//cierre if
          }    // cierre del while
      }   //cierre de la tarea
```

El código que a continuación se muestra, ejecuta el trípode en el robot, este se carga en el bloque RCX maestro. Este programa ejecuta la oscilación anteriormente expuesta, intercambiando datos entre los dos bloques o microprocesadores sobre el estado que guardan los sensores que controlan la posición y movimiento de las patas, además de los sensores de luz como detectores de obstáculos.

```
int n1, n2, n3, x, y,z; //definición de variables.
int sensor_luz;
task main()  //programa principal.
  {
   //modos de operación de los sensores.
   SetSensorType(SENSOR_1,SENSOR_TYPE_LIGHT);
   SetSensorMode(SENSOR_1,SENSOR_MODE_RAW);
   SetSensorType(SENSOR_2,SENSOR_TYPE_LIGHT);
   SetSensorMode(SENSOR_2,SENSOR_MODE_RAW);
   SetSensorType(SENSOR_3,SENSOR_TYPE_LIGHT);
   SetSensorMode(SENSOR_3,SENSOR_MODE_RAW);

   ClearMessage();
   SendMessage(11); //envía mensaje para posicionamiento.
   Wait(10);
   posicionate_en_0(); //programa de posicionamiento inicial.

   until(Message()==21); //espera respuesta de fin de posicionamiento.
   do
    {
     if((SENSOR_1<10)||(SENSOR_2<10)||(SENSOR_3<10)) //Primera
condición; sensor de luz inactivo
```

```
      {
     if(sensor_luz>875)
       {
      PlaySound(3); Wait(40);
      if(SENSOR_1<10&&SENSOR_2<10&&SENSOR_3<10)
         {
          x=1; //ejecución del oscilador con movimiento hacia
             //adelante 1er medio ciclo de movimiento.
           y=1;
           z=0;
           until(Message()==27); //esperando mensage esclavo.
           Wait(10);
           SendMessage(13);
           Wait(20);
           start MOTO_NEURONA_A; // motor A= On.
           start MOTO_NEURONA_B; // motor B= On.
           start MOTO_NEURONA_C; // motor c= Off.

         }

        if(SENSOR_1>10&&SENSOR_2>10&&SENSOR_3<10)
         {
          x=0;  //ejecución de oscilador con movimiento hacia
                delante, segundo medio ciclo de paso.
           y=0;
           z=1;
           until(Message()==23); //esperando mensage del Rcx esclavo.
           Wait(5);
           SendMessage(14);
           Wait(20);
           start MOTO_NEURONA_A; // motor A= Off.
           start MOTO_NEURONA_B; // motor B= Off.
           start MOTO_NEURONA_C; // motor C= On.
         }
        }

       if(sensor_luz<=875)
        {
          PlaySound(2); Wait(40);  //ENCIENDE MOTOR
          if(SENSOR_1<10&&SENSOR_2<10&&SENSOR_3<10)
          {  x=3;  y=3;  z=0;
           // Se ejecuta el oscilador con movimiento en reversa y en
el primer ciclo de paso.
           SendMessage(15);
           Wait(20);
           start MOTO_NEURONA_A; // motor A= On.
           start MOTO_NEURONA_B; // motor B= On.
           start MOTO_NEURONA_C; // motor C= Off.
          }

       if(SENSOR_1>10&&SENSOR_2>10&&SENSOR_3<10)
          {  x=0;  y=0;  z=3;
                       // Se ejecuta el oscilador con movimiento en
reversa y en el segundo ciclo de paso.

           SendMessage(16);
           Wait(20);
           start MOTO_NEURONA_A; // motor A= Off.
           start MOTO_NEURONA_B; // motor B= Off.
           start MOTO_NEURONA_C; // motor C= On.
          }
        }
     }


  if((SENSOR_1>10)&&(SENSOR_2>10)&&(SENSOR_3>10))//Segunda condición;
sensor de luz activo
   {

    lee_sensores(); //obtiene el promedio de intensidad lumínica.
Define obstáculo.
    if(sensor_luz>875)
     {
```

```
      SendMessage(17); //subrutina de acondiconamiento de posición
par ejecución de oscilación.
      Wait(20);
      OnFwd(OUT_A+OUT_B+OUT_C); Wait(25);

       while((SENSOR_1>10)||(SENSOR_2>10)||(SENSOR_3>10))
       {
         if(SENSOR_1<10){Off(OUT_A);Wait(50);}
         if(SENSOR_2<10){Off(OUT_B);Wait(50);}
         if(SENSOR_3<10){Off(OUT_C);Wait(50);}
        }

     }

    if(sensor_luz<=875)
      {
         OnRev(OUT_A+OUT_B+OUT_C); Wait(25);
         SendMessage(18);
         Wait(10);
         while((SENSOR_1>10)||(SENSOR_2>10)||(SENSOR_3>10))
          {
            if(SENSOR_1<=10){Off(OUT_A);Wait(50);}
            if(SENSOR_2<=10){Off(OUT_B);Wait(50);}
            if(SENSOR_3<=10){Off(OUT_C);Wait(50);}

          }
      }
    }
  }//fin del do
   while(true);

}


 void    lee_sensores() //lee el sensor de luz y guarda.
         {


sensor_luz=(SensorValue(0)+SensorValue(1)+SensorValue(2))/3;

          }


 void  posicionate_en_0() //posicinamiento inicial.
       {
         OnFwd(OUT_A+OUT_B+OUT_C); Wait(25);
         while((SENSOR_1<10)||(SENSOR_2<10)||(SENSOR_3<10))
        {

         if(SENSOR_1>10){Off(OUT_A);Wait(50);}
         if(SENSOR_2>10){Off(OUT_B);Wait(50);}
         if(SENSOR_3>10){Off(OUT_C);Wait(50);}

        }

      }

task    MOTO_NEURONA_A() //ejecución de oscilación en motor A.
        {

           n1=-3;

           ClearTimer(0);

           while(Timer(0)<=1000&&SENSOR_1<10)
           {

                if(n1+3==Timer(0))
                 {n1=n1+3;
                 if(x>=1&&x<3)  // PARA X=1
                   {                        OnFwd(OUT_A);
                     Wait(20);
                     Off(OUT_A);}
```

```
            if(x<1)     // PARA X=0
            {
              Off(OUT_A);
              Wait(20);

            }
            if(x>=3)  //PARA  X=3
            {
              OnRev(OUT_A);
              Wait(20);
              Off(OUT_A);

            }


          }
      }    // cierre del while
    }   //cierre de la subrutina


task     MOTO_NEURONA_B() ////ejecución de oscilación en motor B.

    {
       n2=-3;

       ClearTimer(1);

       while((Timer(1)<=1000)&&(SENSOR_2<10))
       {

            if(n2+3==Timer(1))
             {n2=n2+3;
             if(y>=1&&y<3)
               { OnFwd(OUT_B);
                 Wait(20);
                 Off(OUT_B);}
               if(y<1)
               {
                 Off(OUT_B);
                 Wait(20);

               }
               if(y>=3)
               {
                 OnRev(OUT_B);
                 Wait(20);
                 Off(OUT_B);

               }


          }
      }    // cierre del while
    }   //cierre de la subrutina
task     MOTO_NEURONA_C() //ejecución de oscilación en motor C.

    {
       n3=-3;

       ClearTimer(2);

       while(Timer(2)<=1000&&SENSOR_3<10)
       {

            if(n3+3==Timer(2))
             {n3=n3+3;
             if(z>=1&&z<3)  //z=1;
               {
                 OnFwd(OUT_C);
                 Wait(20);
                 Off(OUT_C);}
```

```
            if(z<1)   // z=0;
            {
              Off(OUT_C);
              Wait(20);

            }
            if(z>=3)  //z=3
            {
              OnRev(OUT_C);
              Wait(20);
              Off(OUT_C);

            }

          }
        }   // cierre del while
    }   //cierre de la subrutina
```

El programa se divide en cuatro tareas: la tarea principal o *main*, y tres en las cuales se desarrollan las motoneuronas. En la tarea principal se encuentra las condicionales que definirán la acción a tomar dependiendo del estado de los sensores. Como se aprecia en el programa, el control del robot depende en gran medida del estado en el cual se encuentran los sensores, lo cual se ve reflejado en la aparición de la estructura condicional del *"if"* así como del envío y recepción de mensajes que determinan el principio y fin de un ciclo del motor. Las otras tres tareas son llamadas desde la tarea principal para ejecutar los pulsos de los osciladores. En el programa maestro la tarea principal se encuentra dentro de un ciclo *"do"*, mientras que el programa esclavo se encuentra en un ciclo "*while"*, lo que permitirá ejecutar todos los "*if"* que contienen los mensajes.

Además de lo anterior existen dos subrutinas; la primera llamada lee sensores, la cual obtiene el valor promedio de los sensores de luz que de forma individual arrojan valores oscilantes, siendo éstos de poca utilidad por su inestabilidad, por lo que se tomó la determinación de obtener un solo valor de las tres lecturas. La segunda subrutina posiciona a las patas en una condición inicial.

A diferencia del programa maestro el programa esclavo se concibió con la idea de ser la parte complementaria en el movimiento de las patas. Se debe recordar que dos grupos de tres patas de cada una esta manejado por un bloque RXC cada grupo. La forma de complementar al programa del RCX maestro es por medio de mensajes; el RCX esclavo no puede realizar ninguna tarea hasta que reciba del maestro el mensaje que determine la ejecución. A su vez el esclavo manda una señal al término de la ejecución. La estructura de este programa esclavo es similar a la del maestro, con cuatro tareas pero con una sola subrutina, la de posicionamiento.

El siguiente pedazo de código muestra la forma en la que se comunican los RXC esclavo y maestro.

```
if(Message()==13)
 {

   x=0; y=0; z=1;
   start MOTO_NEURONA_A;
   start MOTO_NEURONA_B;
   start MOTO_NEURONA_C;
   until( SENSOR_3>100);
   SendMessage(23);
   Wait(10);
 }
```

El programa esclavo está estructurado de manera muy similar al programa maestro, siendo necesario para su ejecución, la recepción de una señal desde el RCX maestro. En todas las subrutinas se ejecutan solamente ante la condición de una señal recibida del maestro, sin embargo, éste también envía una señal al término de cada subrutina con el fin de lograr una sincronía en los movimientos y tareas realizadas.

PROGRAMA ESCLAVO.

```
int n1, n2, n3, x, y,z;
int sensor_luz; //PROGRAMA ESCLAVO
task main()
  {
      SetSensorType(SENSOR_1,SENSOR_TYPE_TOUCH);
      SetSensorMode(SENSOR_1,SENSOR_MODE_RAW);
      SetSensorType(SENSOR_2,SENSOR_TYPE_TOUCH);
      SetSensorMode(SENSOR_2,SENSOR_MODE_RAW);
      SetSensorType(SENSOR_3,SENSOR_TYPE_TOUCH);
      SetSensorMode(SENSOR_3,SENSOR_MODE_RAW);
while(SENSOR_1<100||SENSOR_1>100)
    {
      ClearMessage();
      until(Message ()!=0); //ESPERANDO MENSAGE DEL MAESTRO.
       if(Message()==11)
         {  posicionate_en_0();
            SendMessage(21); // Aviso de termino de subrutina.
            Wait(5);}

       if(Message()==13)
        {

          x=0; y=0; z=1; // Ejecución de tripode
                         // 1er ciclo de paso.
          start MOTO_NEURONA_A; //motor A=Off.
          start MOTO_NEURONA_B; //motor B=Off.
          start MOTO_NEURONA_C; //motor C=On.
          until( SENSOR_3>100);
          SendMessage(23);   // Aviso de fin de oscilación.
          Wait(10);
        }

        if(Message()==14)
        {
          x=1; y=1; z=0; // Ejecución de tripode
                         // 2o. ciclo de paso.
```

```
      start MOTO_NEURONA_A; //motor A=On.
      start MOTO_NEURONA_B; //motor B=On.
      start MOTO_NEURONA_C; //motor C=Off.
      until((SENSOR_1>100)&&(SENSOR_2>100));
      SendMessage(24); // Aviso de fin de oscilación.

      Wait(10);
     }

    if(Message()==15)
    {
      x=0; y=0; z=3; // Ejecución de tripode
                     // 1er ciclo de paso con mov. en retroceso.

      start MOTO_NEURONA_A; //motor A=Off.
      start MOTO_NEURONA_B; //motor B=Off.
      start MOTO_NEURONA_C; //motor C=On.


    }

    if(Message()==16)
    {
      x=3; y=3; z=0; // Ejecución de tripode
                     // 2o. ciclo de paso con mov. en retroceso.

      start MOTO_NEURONA_A; //motor A=On.
      start MOTO_NEURONA_B; //motor B=On.
      start MOTO_NEURONA_C; //motor C=Off.

    }

    if(Message()==17)
     {
        OnFwd(OUT_A+OUT_B+OUT_C); Wait(25);
         while((SENSOR_1>100)||(SENSOR_2>100)||(SENSOR_3>100))
         {
          if(SENSOR_1<100){Off(OUT_A);Wait(50);}
          if(SENSOR_2<100){Off(OUT_B);Wait(50);}
          if(SENSOR_3<100){Off(OUT_C);Wait(50);}
         }
          Off(OUT_A+OUT_B+OUT_C); Wait(10);
          SendMessage(27);

      }

    if(Message()==18)
     {
       OnRev(OUT_A+OUT_B+OUT_C); Wait(25);
        while((SENSOR_1>100)||(SENSOR_2>100)||(SENSOR_3>100))
       {
          if(SENSOR_1<=100){Off(OUT_A);Wait(50);}
          if(SENSOR_2<=100){Off(OUT_B);Wait(50);}
          if(SENSOR_3<=100){Off(OUT_C);Wait(50);}
       }  Off(OUT_A+OUT_B+OUT_C); Wait(10);


      }
   }//terminacion del do


 }//terminacion  del main

void  posicionate_en_0 ()
  {
```

```
      while((SENSOR_1<100)||(SENSOR_2<100)||(SENSOR_3<100))
     {
        if(SENSOR_1<100){ OnFwd(OUT_A); Wait(10);}
        if(SENSOR_2<100){ OnFwd(OUT_B); Wait(10);}
        if(SENSOR_3<100){ OnFwd(OUT_C); Wait(10);}

        if(SENSOR_1>100){ Off(OUT_A); Wait(10);}
        if(SENSOR_2>100){ Off(OUT_B); Wait(10);}
        if(SENSOR_3>100){ Off(OUT_C); Wait(10);}
        PlaySound(3); Wait(10);

       }
        Off(OUT_A+OUT_B+OUT_C);
        Wait(10);


     }//fin subrutina posicionate


task    MOTO_NEURONA_A()  //Ejecución del oscilador en motor A
    {
          n1=-3;

           ClearTimer(0);

           while(Timer(0)<=1000&&SENSOR_1<100)
           {

              if(n1+3==Timer(0))
               {
                 n1=n1+3;
              if(x>=1&&x<3)  // PARA X=1
                   { OnFwd(OUT_A);
                     Wait(20);
                     Off(OUT_A);
                    }
               if(x<1)     // PARA X=0
                   {
                     Off(OUT_A);
                     Wait(20);

                   }
               if(x>=3)  //PARA  X=3
                   {
                     OnRev(OUT_A);
                     Wait(20);
                     Off(OUT_A);
                   }

              }//cierre if
            }    // cierre del while
        }   //cierre de la tarea


task     MOTO_NEURONA_B() //Ejecución del oscilador en motor B

        {
           n2=-3;

           ClearTimer(1);

           while((Timer(1)<=1000)&&(SENSOR_2<100))
           {
```

```
                if(n2+3==Timer(1))
                 {
                    n2=n2+3;
                    if(y>=1&&y<3)
                      { OnFwd(OUT_B);
                        Wait(20);
                        Off(OUT_B);
                       }

                    if(y<1)
                      {
                        Off(OUT_B);
                        Wait(20);
                      }

                    if(y>=3)
                      {
                        OnRev(OUT_B);
                        Wait(20);
                        Off(OUT_B);
                      }

                   } //cierre if
               }    // cierre del while
          }   //cierre de la tarea

task      MOTO_NEURONA_C()  //Ejecución del oscilador en motor C

          {
             n3=-3;

             ClearTimer(2);

             while(Timer(2)<=1000&&SENSOR_3<100)
             {

                  if(n3+3==Timer(2))
                   {
                     n3=n3+3;
                  if(z>=1&&z<3)  //z=1;
                     { OnFwd(OUT_C);
                       Wait(20);
                       Off(OUT_C);}
                  if(z<1)   // z=0;
                     {
                       Off(OUT_C);
                       Wait(20);

                     }
                  if(z>=3)  //z=3
                     {
                       OnRev(OUT_C);
                       Wait(20);
                       Off(OUT_C);

                     }
               }
        }//cierre while
    }//cierre tarea
```

CAPITULO VI.

6.1 ANÁLISIS DE RESULTADOS.

6.1.1 DEL DISEÑO MECÁNICO.

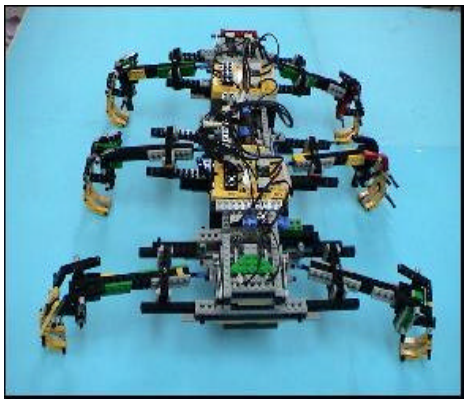


Fig. 44. Diseño final del hexápodo

Se construyó un robot de seis patas tal como se propuso. La estructura mecánica del robot es suficientemente fuerte como para realizar los movimientos que marca el objetivo. Es capaz de soportar su peso y la inercia de sus movimientos, sin presentar problemas de desajuste o dificultad en la interacción de engranajes y barras. El robot final (fig. 44), es el resultado de varios prototipos llevados a cabo, los cuales se analizan detalladamente en el capitulo III. En cada mejora, se buscó combinar las capacidades técnicas de los motores, tales como la potencia mecánica de salida y el peso que necesariamente habría de soportar. Este ultimo modelo tiene un peso de 3[kg], una altura de 15[cm], 30[cm]de ancho y una longitud de 50[cm].

La potencia real que aporta un solo motor no es suficiente para mantener el movimiento de todo el robot, sin embargo para solucionar este problema se hicieron juegos de engranes que dieran la amplificación de la potencia mecánica. Estas ventajas logradas fueron a costa de reducir la velocidad de giro en cada pata. Uno de los prototipos que le anteceden (prototipo 5), se realizó en esta tesitura, transmitiendo directamente la potencia del motor al movimiento de la articulación. Este modelo resultó un sistema débil y si bien logró realizar el movimiento buscado, fue insuficiente y de baja capacidad mecánica.

El primer modelo se estructuró con solamente dos motores que controlaban dos patas cada uno, lo que impedía un control de éstas ya que no eran independientes. Este primer prototipo se implemento de esta forma debido a que solamente se contaba con un solo *kit* LEGO MINDSTORMS que limitaba el diseño. El primer prototipo sirvió como base para la prueba de los primeros osciladores de dos neuronas. La obtención de resultados fue insuficiente, lo que dio pauta a un sistema mecánico más complejo y mejor controlado.

El número de motores se fue incrementando en la medida de una necesidad de movimiento mas detallado. Se hizo necesario controlar con precisión el movimiento de

cada una de las patas, esto con el fin de generar cualquier tipo de locomoción. Sin embargo, las características mecánicas del material de diseño, sensores, controladores de posición, fueron insuficientes para lograr un control exacto. De esta manera, se utilizaron sensores de tacto para determinar la posición de las patas. Tales sensores son sistemas muy simples y en este sentido, arrojaron resultados muy rudimentarios.

El control de posición actuado desde los sensores solucionó de algún modo el problema, sin embargo se logró una mejora al utilizar herramientas de programación, y así, se obtuvo una adecuación en la lectura de los sensores. Además se colocaron dos sensores en cada puerto, una de tacto y otro sensor de luz. Dichos sensores tenían funciones de un modo independientes, pero debido a las limitantes en los puertos de entrada de los RCX's, exigieron la búsqueda de nuevas alternativas. Los sensores de luz funcionan como un sistema de detección de objetos delante del robot, basados en valores leídos de la intensidad de reflexión de luz infrarroja; en su contraparte los sensores de tacto permiten generar un sistema de posición angular muy simple del brazo de giro de la pata y además de lograr sincronizar las patas restantes.

Si bien el sistema presenta inconvenientes, estas permiten un movimiento aceptable dentro

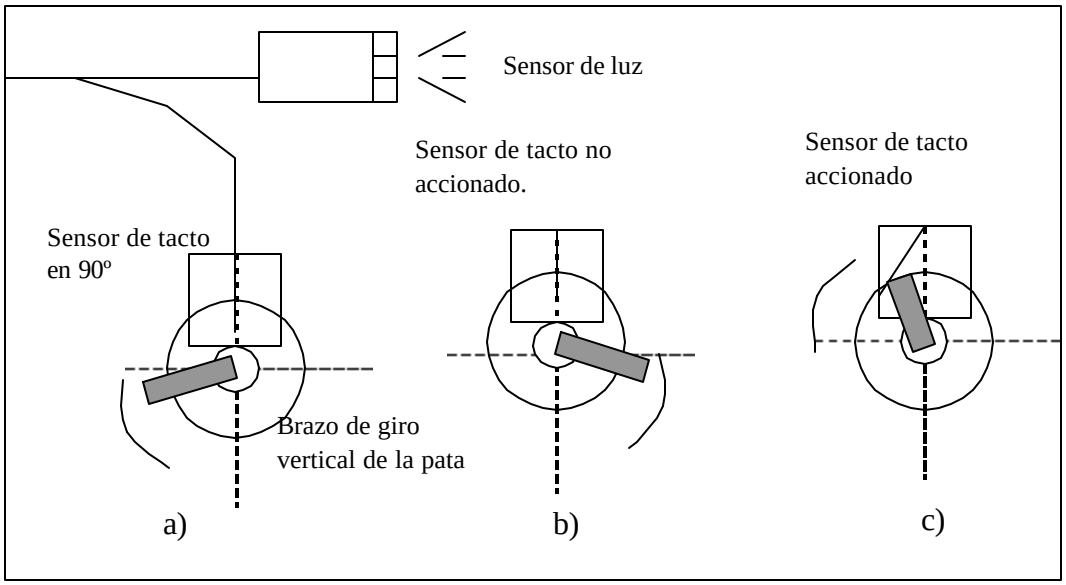


Fig. 45. Esquema de colocación de sensores en patas y con el brazo de giro en distintas posiciones; a) El sensor de luz no se activa y por tanto se lee un valor menor a 10, b) el sensor de tacto no está activado en esta posición y mantiene un valor de 10, c) el sensor de tacto es activado y se lee el valor del sensor de luz mayor a 100, con lo cual indica la posición de la pata en 90º.

de los términos buscados. Es de observar (fig. 45), que el sensor de tacto solo es accionado en la posición angular igual a 90º. Únicamente se está seguro de la posición del brazo de giro de la pata en el ángulo de 90º y esto sucede cada que se realiza un giro completo. No es posible realizar un

movimiento en cualquier otro ángulo, sin que se provoque una imprecisión y en consecuencia una desincronización del movimiento de las patas.

Otro de los inconvenientes dados en este sistema, se da en la lectura del sensor de luz. El valor en bruto del sensor de luz sucede cuando el brazo de giro activa el sensor de tacto y es un intervalo dado entre 20 a 30º, con lo cual no se pueden obtener las condiciones de objetos cercanos leídos por el sensor de luz, presentados en las posiciones angulares fuera del entorno de la posición del sensor de tacto. Esto no dificulta en mucho la ejecución de los movimientos, dado que estos son de manera lenta y considerando que entre un ciclo y otro, el robot no se desplaza considerablemente como para provocar un choque de consecuencias graves.

6.1.2 DEL PATRÓN DE PASOS

Una de las etapas medulares de esta investigación, fue la búsqueda de patrones de pasos mediante la simulación de redes de dos y más neuronas. Los primeros pasos se enfocaron a encontrar la oscilación entre dos neuronas, en los cuales se llegaron a resultados positivos. Se hicieron varias simulaciones combinando pesos de inhibición entre las neuronas, así como los valores de alimentación de las fibras. Se logró la oscilación de ambas neuronas y de los cuales se desprenden las siguientes observaciones:

En las simulaciones (osciladores ny*), las neuronas mantienen una oscilación óptima si los valores de intensidad sináptica entre ellas difieren al menos de 10 unidades. Se presentaron dos casos: cuando el valor de inhibición entre las neuronas se aleja considerablemente, el sistema deja de oscilar y se presenta la característica de “la neurona ganadora. Se observa una sola neurona disparándose sin ningún tipo de oscilación; a decir de lo analizado, no sucede abruptamente, sino empieza a degenerarse la oscilación, desde un valor central, como cabría observarse cerca de un sistema gravitacional, hasta un valor periférico y lejano. Un centro, o un punto de equilibrio que tiende a disminuir su comportamiento oscilatorio a medida que la inhibición neuronal se dispara en sentidos opuestos. La neurona inhibida es la que contiene el menor peso de conexión en su salida.

Y en la parte contraria, si los valores de inhibición entre cada neurona se acercan tanto hasta igualarse, la oscilación tiende a perderse y en este caso las neuronas se disparan mutuamente, hasta comportarse como dos neuronas sin conexión o independientes, alimentadas por sus fibras correspondientes.

Otro caso presentado en las simulaciones, refiere a las fibras de conexión, estos valores de disparo, si se varían considerando los puntos anteriores causan un incremento en el número de disparos en cada ráfaga. De este modo se pueden manipular los anchos de disparo de las ráfagas manipulando este parámetro.

Los cambios operados en la mecánica y número de patas en el robot, fueron el discurso de una nueva alternativa oscilatoria. Llegamos a una topología de tres neuronas, con el objetivo de colocar tres neuronas oscilando simultáneamente. Consideramos que si las tres neuronas se disparaban cabría la interpretación como el disparo de las tres patas en una locomoción característica del trípode. De estas simulaciones se habituaron dos tipos de conexión, ambas en una malla cerrada, pero en el primer caso con lazos de conexión simple entre cada neurona. Posteriormente se implementaron conexiones en malla cerrada y además de doble conexión o mutuamente inhibitorios entre cada neurona. Las topologías de red mostradas en la figura 46, dieron resultados satisfactorios desde el objetivo propuesto.

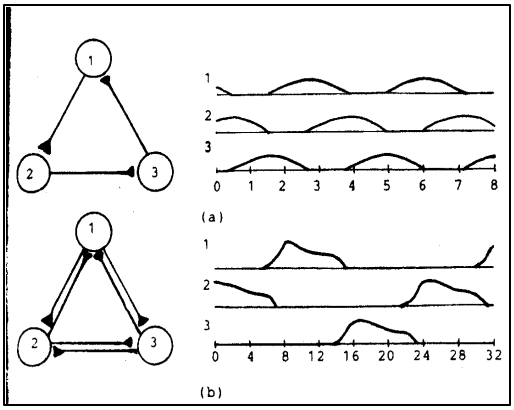


Fig. 46. Conexión de tres neuronas.
a) Conexión inhibitoria cíclica
b) Conexión inhibitoria mutua
Tomado de Matzuoko, k. [26]

Para la red de inhibición simple, se logró obtener la simultaneidad de disparo entre las tres neuronas. Cabe notar el oscilador OSILA2.OUT, donde se presenta esta característica, aquí fue necesario variar las semillas de cada neurona. Este es un parámetro que no mantiene una manipulación exacta en el número de disparos en cada ráfaga y es un valor aleatorio que se inicia en el valor definido. Entre las demás simulaciones se lograron que las tres neuronas oscilaran, pero con anchos de disparo diferentes y con un efecto causal entre cada una de ellas (ver gráficos de salida neuronal Osila.out, cap. IV).

Los osciladores mejor logrados con ráfagas de disparo muy cercanos entre sí, son también respuestas de esta topología de red, aun cuando no logra la simultaneidad en los disparos de cada neurona. Así mismo, en forma análoga a los osciladores de dos neuronas, también se observa un cierto efecto de la neurona ganadora. Para mantener una oscilación, aún con la característica causal y con ciertos traslapes, ha de mantenerse un similitud en los pesos de inhibición de al menos dos neuronas. La tercera neurona, si se aleja el valor de su peso, respecto a las otras dos neuronas, el oscilador llega a degenerar en dos vertientes. La ya mencionada neurona ganadora, que puede ser provocada por la ganancia del par de neuronas, esto si la tercera neurona

se disminuye su peso de inhibición; o bien, se presenta la ganancia de un sola neurona si esta se incremente el valor de su peso de conexión, por encima de las otras dos.

Los valores que se obtuvieron, como el caso del oscilador OSILA4 (gráfico 9), es representativo de un patrón de pasos en los cuales es posible llevar una cualidad en el movimiento de cada pata; esto es, que solamente se mantiene el movimiento de una sola pata cuando todas las demás se mantienen estáticas. De esta forma el resultado obtenido a través de esta red, aun cuando no enfoca las características de movimiento buscado, arroja un movimiento en el cual hacía también necesaria su búsqueda, como un comportamiento de locomoción alterno. Este movimiento o patrón de locomoción, está representado en la figura 24 y es el primer patrón de pasos esquematizado.

Debido a la falta de resultados esperados, se establecieron conexiones mutuas entre las neuronas. Así, se simularon redes de 3 neuronas inhibiéndose mutuamente y con ello las primeras oscilaciones simultáneas (véase Fig. 36 y gráficos 10 y 11). Esta topología presenta mayor flexibilidad en los tipos de oscilación buscados; además de caracterizarse por una menor inestabilidad en el cambio de sus parámetros.

Las redes de inhibición mutua oscilan con ciertas características en los valores de conexión. Los pesos en una malla unidireccional, ya sea externa o interna, deben de mantenerse de manera similar. Ambas mallas pueden separarse en términos de sus valores, con lo cual repercutirá en los anchos de cada disparo, así como en el espaciamiento entre cada ráfaga.

Con este tipo de redes neuronales, redes de inhibición mutua, se lograron oscilaciones con mayor control en los tamaños y número de salvas. Es representativo de un patrón de paso característico al trípode señalado en figuras anteriores, esto si se considera que las neuronas disparadas, representan dos patas extremas del robot y la tercera neurona activada, un pata intermedia. Este enfoque permite interpretar el trípode buscado.

El incremento de redes con mayor número de neuronas fue considerado como alternativo a redes más estables y de mayor control. Sin embargo, las simulaciones evidenciaron la complejidad de tales redes, observando la no-linealidad en los modelos neuronales. De esta forma al conectar un mayor número de neuronas, el control de los parámetros de la red se hace aún más complejo. La red se vuelve inestable a pequeñas variaciones en cualquiera de los parámetros involucrados, esto hace difícil mantener un comportamiento predecible de la red.

Otra faceta de las redes de seis neuronas, fue el encontrar una cercanía en los resultados obtenidos por Randal Beer, en la aplicación análoga a un sistema de locomoción de un robot hexápodo.

6.1.3 LÓGICAS DE PROGRAMACIÓN DE LOS OSCILADORES.

En cuanto a los programas de ejecución de los tipos de locomoción del robot, estos están basados sobre NQC, lenguaje cercano a C, pero con limitaciones muy notorias. Los programas estructurados permiten realizar los movimientos propuestos. Este apartado es de gran relevancia; representa el punto de intersección entre las simulaciones llevadas a cabo, desde un sistema externo como NEURORED y la ejecución lógica de los programas representativos de los patrones oscilatorios.

Se toman los archivos de salida *.out de cada uno de los osciladores que imitan los patrones buscados. El archivo de salida es un listado de números binarios, donde se representan los disparos neuronales con un valor de 1 y la ausencia de estos con ceros. Los vectores de salida en cada neurona se consideran como motoneuronas, esto es, una ráfaga es un tiempo de encendido de un motor que controla el movimiento de una pata.

Llevar a cabo la ejecución real de una simulación en NEURORED, es uno de los aspectos relevantes, puesto que no se tiene una interface entre el simulador de red y el cerebro del robot, el RCX. Luego entonces, se hace necesario una lógica de representación de los disparos neuronales. Los códigos que ejecutan los comportamientos neuronales, se basaron en los cronómetros que contiene el microprocesador del RCX. Se hizo posible generar pulsos de unos y ceros, semejantes a las salidas de los archivos de NEURORED. Los valores generados pueden articularse con tiempos entre disparos y anchos de ráfagas, según el oscilador deseado. Esta estructura se manifiesta para cada pata.

El código de programación genera las salidas y pueden ser dos estados para el motor: neurona activada (motor encendido) y neurona silente (motor apagado).

Además de la generación de los pulsos neuronales visualizado en el estado del motor, también se requiere la comunicación entre los dos RCX's, considerando que generar un trípode conjuga motores controlados por ambos RCX's. Tales comunicaciones permiten generar la sincronía y los pasos de movimiento solicitados por la red. Los resultados que se tienen al desarrollar los programas y observar su ejecución, fueron satisfactorios aunque no suficientes. La

raíz de esta deficiencia está enmarcada por los sistemas de detección de posición en articulaciones ya mencionado en los análisis de resultados mecánicos.

El robot imita los patrones de pasos aún cuando no presenta una exacta ejecución en los tiempos de la simulación. Se presentan retrasos en el movimiento del brazo de giro de la pata. Éstos retrasos en el brazo son de un modo soslayables, pues el sistema de control de posición se basa en medidas angulares y aunado a esto el control de velocidad es inexistente. Los retrasos son aleatorios, causados por la inercia de movimiento y la distribución de peso del robot. Se observa que la sincronía precisa en las patas en movimiento, es un atributo del sistema mecánico del robot, convenido con los aspectos de sensado y posicionamiento.

Los tres sensores infrarrojos colocados en uno de los RCX'x (maestro), lograron decidir las tareas adecuadas y detectar obstáculos. Solo se colocaron sensores en el RCx` maestro, pues se considera que el robot no ejecuta movimientos de retroceso considerables y queda abierta la opción de un control con mayor exactitud.

VII. CONCLUSIONES.

7.1 CONCLUSIONES

El proyecto desarrollado mantiene una cercanía, tanto en características y objetivos a los diseños de los robots hexápodos realizados por Randal Beer. Si bien Beer busca generar e imitar patrones de pasos en un robot hexápodo, el presente proyecto se direcciona en la utilización de herramientas disponibles y que no han sido diseñadas especialmente para dicha finalidad. En este sentido, la discusión se enfoca a generar espacios de solución y alcances de una continuidad futura, comparado desde el más cercano referenciador: el hexápodo de Beer. Otras investigaciones están enfocadas en robots mecánicos y de seis patas, en los que los sistemas de control y adquisiciones de datos del comportamiento de movimiento son de mayor complejidad.

Con base en esta idea se analizan las mejoras que se pueden hacer tanto a la estructura del robot como a los programas que generan el movimiento. En lo referente a la estructura del robot, los diseños y estructuras se adecuaron a las formas y diseños que permitió el *kit*-LEGO MINDSTORM. No se utilizaron piezas especiales o diseños alternos, esto con la finalidad de hacer notar el alcance y la capacidad de flexibilidad que permite un *kit* comercial. La necesidad de estructuras más rígidas y estables, conlleva la utilización de materiales ligeros y rígidos como sería el caso del aluminio. Los problemas presentados en el control de posición, velocidad del motor, pueden solucionarse adicionándose sensores más precisos y confiables, además de integrar un motor de pasos que permitiese el control de posición en cada ángulo. La composición de un motor de pasos permitirá el posicionamiento exacto de las patas, sin embargo esto también solicita un ambiente de almacenamiento en memoria más amplia y plataformas de programación más flexibles

Es deseable la integración completa de los sistemas de simulación y ejecución dentro del robot. En un sistema más complejo y de mayor capacidad, se esperaría que los sensores estuviesen directamente relacionadas con los valores de conexión entre la red; esto es, tener una red que se ejecutase en tiempo real, como en el hexápodo de Randal Beer, en las cuales los programas permiten la interacción entre la red y los cambios del medio, estas dadas a través de los sensores.

Los robots diseñados especialmente para proyectos como el mencionado arriba tienen otra ventaja, que su capacidad de memoria les permite almacenar programas mas complejos, limitante presentada en el presente proyecto. En un primer momento se tenia proyectado incluir el programa generador de redes neuronales (NEURORED) en el microprocesador de los RCX's, para de que esta manera se generaran las redes neuronales de manera independiente, haciendo al robot autónomo en las decisiones a tomar. Estas decisiones estarían incluidas en los programas realizados en el lenguaje NQC, pero sus limitaciones en cuanto a la librerías que contienen no permiten desarrollar programas sofisticados que actúen ante cualquier eventualidad que llegara a tener el robot al momento de desplazarse.

Por lo anterior se propone realizar programas especialmente diseñados para el microcontrolador de los RCX's que incluya todas la librerías necesarias (que tienen el lenguaje C de programación) y a la par diseñar en lenguaje C o en el nuevo NQC el generador de redes neuronales (NEURORED) con la finalidad de tener programas con interfaces iguales, permitiendo ampliar las áreas de investigación con el robot hexápodo y las redes Neuronales Biológicas. Otra alternativa utilizar un procesador con mayor capacidad de almacenamiento y procesamiento que vaya de la mano con lo expuesto en este párrafo.

Lograr que un sistema artificial se comporte desde una característica cercana a la naturaleza, es una tarea compleja que incluye la vinculación de diversas áreas de conocimiento. Desde este apartado las conclusiones sobre este trabajo caracterizan y comprueban la complejidad de tales tareas. Generar un patrón de pasos, es establecer un equilibrio entre los sistemas de control en el sistema en general. Define características óptimas en el diseño mecánico, aunado a ventajas tanto en software de control, como los sistemas de sensado.

Se enmarca el desarrollo complejo y multimodal de los sistemas naturales con tareas simples que requieren de una combinación de diversas capacidades al llevarlos a un modelo artificial. Si bien un sistema de locomoción basado en patas, resulta ser mas adaptado a cualquier terreno, y demuestra el camino que ha tomado la naturaleza en este hecho, estos resultan ser muy complejos en los sistemas de control y sincronización. Solicitan de

distintas necesidades que interactúan entre sí y se evidencia la rigidez de los modelos neuronales a utilizar.

Ciertamente que los comportamientos neuronales son de carácter no lineal, con lo cual complica la predicción y control exacto de sus respuestas. Desde esta perspectiva, comparado con robots basados sobre ruedas, se visualiza el porqué de la proliferación de este tipo de sistemas. Sin embargo, éstas han sido desplazados por las necesidades mismas en el desempeño delas tareas ejecutadas que la misma naturales solicita.

LEGO MINDSTORM, diseñado con fines comerciales, mantiene la capacidad de elaborar sistemas robóticos capaces de generar movimientos más allá para los fines de su diseño. Se ha mostrado como el robot grillo basado en piezas LEGO, pero en el que se adicionaron sistemas electrónicos externos, también percibe esta aseveración demostrada en el presente trabajo.

También se concluye que el diseño de robots hexápodos y su control de movimiento es similar a mantener un sistema de manipulación cibernética. De ahí se desprende que el control basado en redes neuronales puede generarse a partir de configuraciones sencillas, pero que solicitan sistemas de adquisición de datos de mayor complejidad. En la medida que las redes se hacen más complejas, las necesidades de procesamiento y ejecución son de mayor relevancia. Tales sistemas de control pueden tener mediante adecuaciones capacidades de aprendizaje y características autónomas en la toma de decisiones.

A P É N D I C E A.

I. SIMULADOR NEURORED

I.I. INTRODUCCIÓN

El simulador NEURORED fue adaptado en el laboratorio de Cibernética de la Facultad de Ciencias de la UNAM, en el año de 1992 por Mauro A. Alcántara Galindo bajo dirección del Dr. Ismael Espinosa Espinosa, teniendo como objetivo la creación de un programa que apoye la tarea de investigación en Neurofisiología mediante la elaboración de una red neuronal, simulación dinámica de la misma, análisis de la conectividad y el despliegue de los resultados. Otra de las finalidades del desarrollo del simulador es la orientación que tienen los paquetes existentes en el mercado que tratan sobre Neurociencia Computacional, es decir, se enfatizan en características macroscópicas o microscópicas para un problema o red neuronal biológica específica, siendo las posibilidades de desarrollo infinitas.

Estos programas que simulan por medio de computadoras — hardware o software —, la fisiología del cerebro basados en modelos matemáticos comprobados y validados por la experimentación, representan la interacción de neuronas conectadas entre sí y analizan la dinámica funcional por medio de arquitecturas diversas y parámetros muy similares a los fisiológicos.

NUERORED se inició con uno de los simuladores de MacGregor, llamado SYSTM11, diseñando a su alrededor programas que realizan tareas. En este simulador el núcleo dinámico está representado primordialmente por la variable conocida como el potencial de la membrana de la neurona y es alrededor de éste que se incluyen una serie significativa de parámetros neurofisiológicos como constantes de tiempo, umbral, conductancias, etc.

Es importante hacer notar las similitudes y las diferencias entre el programa NEURORED y las Redes Neuronales Artificiales. En ambos casos se trabaja con modelos de neuronas que pueden ser matemáticos o electrónicos. Pero en el caso de NEURORED su objetivo es que el modelo de red neuronal así como sus parámetros sean los más cercanos posible a un sistema biológico conocido total o parcialmente por medio de la experimentación neurofisiológica. En este sentido, NEURORED va de la mano con la Neurofisiología. En cambio las Redes Neuronales Artificiales son más una herramienta ingenieril que esta inspirada solamente en algunos conocimientos neurofisiológicos, por lo que no les interesa en particular la neurofisiología, sino las aplicaciones

ingenieriles donde se requiere que el sistema sea capaz de tomar decisiones, es decir, en optimización, reconocimiento de patrones, robótica flexible, y muchas más.

El siguiente diagrama de bloques (fig. i), muestra los diferentes módulos que contiene el programa NEURORED los cuales permiten un mejor entendimiento y utilización tanto en el diseño como el análisis de la red neuronal biológica implementada.

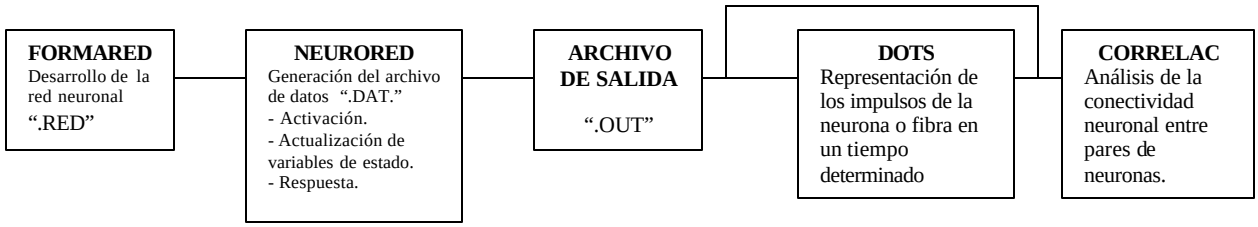


Fig. i) Módulos de NEURORED

I.II. FORMARED

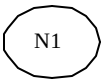
Es un programa que se desarrolló para poder implementar redes neuronales biológicas por medio de figuras — como cuadrados, círculos y líneas — además de generar los archivos de entrada al Simulador de McGregor. Al elaborar redes neuronales biológicas en éste programa, se genera el archivo con extensión ".RED"; una vez terminada la red y salir del programa, se genera el archivo con extensión ".DAT" el cual entra al simulador. Éstos archivos pueden ser modificado en cualquier editor de texto.

Éste módulo se creo bajo el concepto de apuntadores o variables dinámicas, que permite tener un número ilimitado de neuronas o fibras. Aunado a esto el poder respaldar la red generada en cualquier momento da la seguridad de mantener guardada la información.

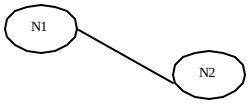
Desafortunadamente el programa tiene la limitante de tener poco espacio en pantalla y además de la capacidad de la memoria de la Computadora. Por ejemplo, una red neuronal de 55 neuronas perdería claridad y sería tediosa su compilación. Por otra parte sólo se permite modificar la intensidad sináptica entre pares de neuronas, los demás parámetros son dados por defecto.

Para poder utilizar el paquete se deben conocer la nomenclatura y las teclas con las cuales podemos generar la red neuronal biológica. A continuación se muestran las nomenclaturas del programa así como las teclas con las cuales se obtienen.

Representación de una fibra con el nombre F1.



Representación de una neurona llamada N1.



Conexión entre la neurona N1 y la neurona N2.

-i Representación del peso o intensidad de una conexión inhibitoria (negativo).

+e Representación del peso o intensidad de una conexión excitatoria (positivo).

Teclas:

H	Permite observar a pie de pantalla las diferentes opciones del paquete.
↑	Mueve el cursos "X" posiciones arriba dentro de la pantalla.
↓	Mueve el cursos "X" posiciones hacia abajo dentro de la pantalla.
← →	Mueve el cursor "X" posiciones a la izquierda o derecha dentro de la pantalla.
+	Incrementa en forma unitaria la velocidad de pantalla dentro del monitor.
—	Decrementa la velocidad del cursor.
F	Despliega una fibra en la pantalla y pide el nombre de la fibra.
N	Despliega una neurona en la pantalla y pide el nombre de la Neurona.
C	Inicia conectividad entre neuronas.
I	Termina la conectividad entre neuronas y pide el valor de la intensidad o peso de la conexión, que puede ser positivo o negativo.
E	Inicia la conectividad de una fibra hacia una neurona.
A	Termina la conectividad de una fibra hacia una neurona y pide el valor de la intensidad o peso de la conexión que puede ser positivo o negativo, aunque usualmente será positivo. Finalmente despliega la conexión.
S	Habilita la salida del sistema y pregunta si se desea generar el archivo de entrada al simulador NEURORED.
G	Guarda en disco la red que se está diseñando en pantalla.

R Recupera de disco la red y la despliega en la pantalla.

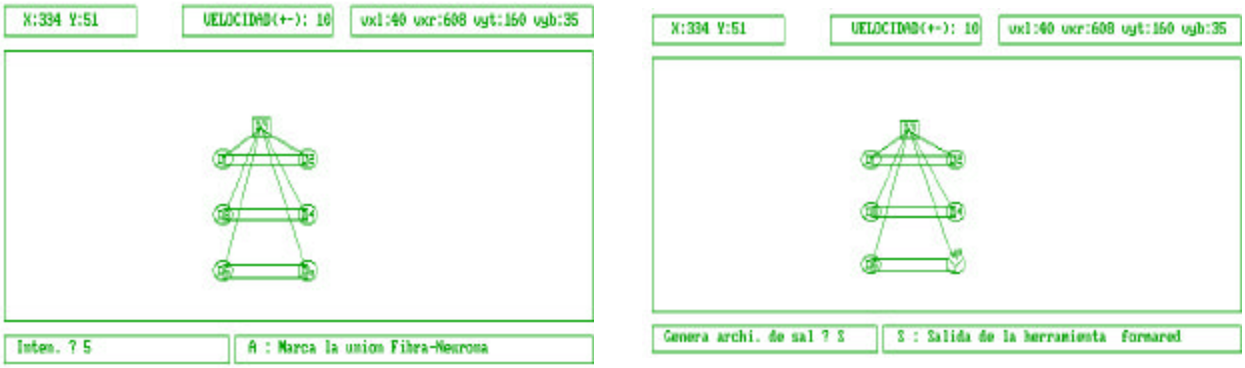


Fig. ii) Formación de una red neuronal con el modulo FORMARED

Las graficas (Fig. ii); ejemplifican la forma en que se elabora una red neuronal utilizando las teclas arriba descritas.

I.III. NEURORED

El simulador es una versión simplificada del simulador SYSTM11 de McGregor. NEURORED es un programa que simula el funcionamiento de un numero arbitrario de neuronas activadas por un número arbitrario de fibras o axones de entrada. El programa es general, lo que implica que se puede simular cualquier red neuronal biológica con sólo cambiar los parámetros y los datos. La introducción de datos para redes muy grandes presenta dificultades por ser una cantidad grande, pero para redes pequeñas, los datos se pueden ir proporcionando al momento de correr el programa o conforme los va necesitando.

El programa resuelve ecuaciones diferenciales discretizadas a partir de las funciones de entrada que dependen del potencial de la membrana y así obtienen el estado y las funciones de salida. Éstas ecuaciones son versiones similares a las de Hodgkin – Huxley, aunque más sencillas por estar linealizadas.

El paquete tiene dos subrutinas principales: una que está encargada de hacer llegar el mensaje de activación de las fibras a las neuronas y la segunda, que se encarga de actualizar las variables de estado de cada neurona en cada instante de integración. Existe una tercera subrutina que genera los números aleatorios.

I.IV. PUNTOS (DOTS)

Este módulo permite observar a la manera de los neurofisiólogos, el archivo generado por NEURORED; dicho archivo está compuesto por ceros (0) y unos (1), que representan los impulsos de la neurona o fibra en determinado instante de tiempo.

El despliegue de los puntos generados por la red neuronal se realiza en cuatro puertos de salida que se muestra en la pantalla al acceder al módulo y en la parte inferior de la pantalla se muestran los datos correspondientes a cada puerto como el nombre del puerto, la escala a la que se quiere graficar, el número de disparos generados por la red y el intervalo de tiempo graficado. Además de lo anterior se deben de teclear los datos requeridos para poder desplegar el archivo de salida (".OUT"), en forma de disparos de la red neuronal. Éstos datos son:

a) El puerto donde se va a desplegar la simulación.
b) La neurona que se va a simular.
c) El intervalo de tiempo en el cual se quiere analizar la neurona seleccionada.

Este módulo es de suma importancia para el análisis de las redes neuronales biológicas que se generaron (Osciladores), porque al desplegar los trenes de pulsos, permite ver la forma en que las neuronas están disparándose o inhibiéndose, dando una idea clara de lo que ocurre en las conexiones entre las neuronas, permitiendo saber qué parámetros serán modificados de ser necesario para poder generar los patrones de pasos que se requieren (ver fig. iii).

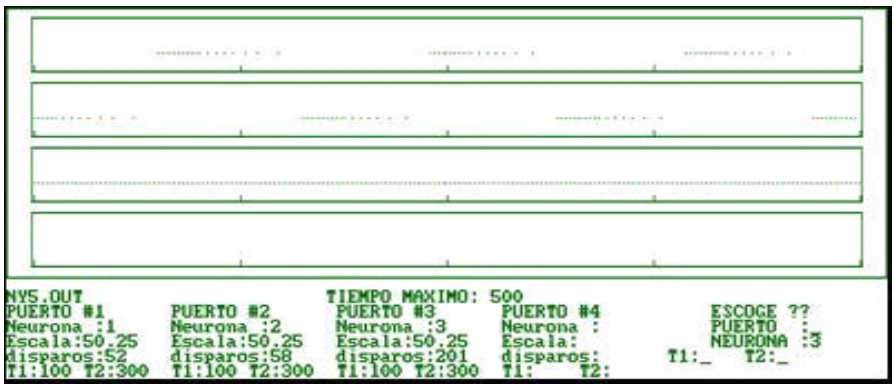


Figura iii) Ráfagas de disparo de la red NY5 en el modulo DOTS

I.V. CORRELACIÓN CRUZADA (CORRELAC)

Este módulo permite analizar la conectividad entre pares de neuronas, de fibras o de combinaciones de ambas. El módulo está basado en la obtención del histograma de correlación

cruzada. El histograma se puede calcular para diferentes intervalos de tiempo y luego se despliega en pantalla o se imprime. Las características del histograma indican el tipo de conexión entre neuronas que puede ser: excitación, inhibición, independencia, o entrada compartida. En la figura iv, se muestra el histograma de correlación cruzada de uno de los osciladores obtenidos en el capítulo IV.

El histograma muestra claramente la conducta oscilatoria de la red con las ráfagas de disparo alternándose con periodos de silencio.

El módulo sólo acepta archivos generados por NEURORED que son los archivos de impulsos generados anteriormente (.OUT). El primer dato que solicita el programa es el nombre del archivo de salida, el segundo dato pedido es el par de neuronas que se quiere correlacionar y por último el intervalo de tiempo en el cual se quiere calcular la correlación.

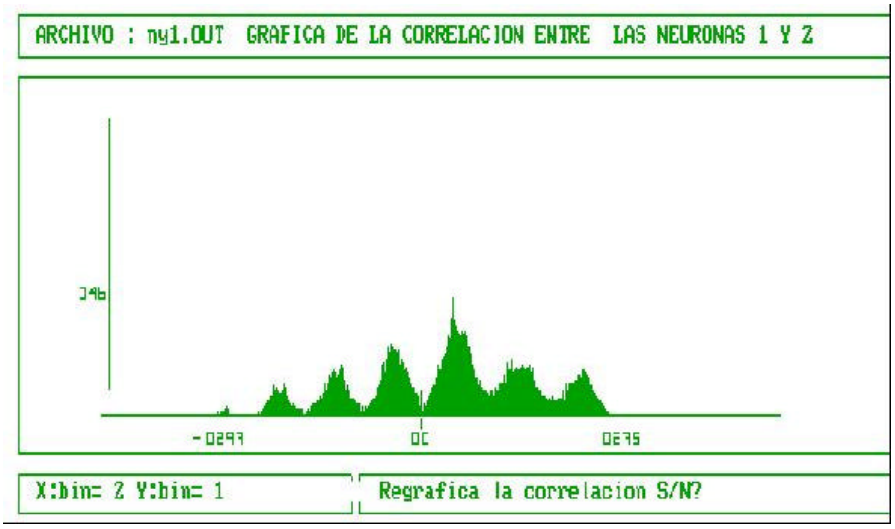


Figura iv) Histograma de la red NY1 graficada en el modulo CORRELAC

Las graficas ejemplifican la forma en que se despliegan los resultados de haber comparado la actividad neuronal entre dos neuronas.

I.VI. REQUERIMIENTOS MÍNIMOS

El simulador de redes neuronales NEURORED es un pequeño paquete diseñado para MICROCOMPUTADORAS IBM y compatibles, por lo que se requiere del siguiente equipo para su funcionamiento: 512K de memoria RAM, tarjeta de gráficos instalada y contar con un disco duro para así tener el paquete instalado en un solo menú con sus cinco módulos.

APÉNDICE B

II. PROGRAMACIÓN EN NQC

II.I. INTRODUCCIÓN

Con el fin de poder aprovechar las características que *LEGO MINDSTORM* ofrece para el diseño de Robots y su programación, se creó el lenguaje *NQC* (No Quit C) que permite una amplia variedad de instrucciones para el desarrollo de actividades que el robot pueda realizar. Este programa fue creado por el ingeniero de *Motorola* Dave Baum especialmente para *MINDSTORM* con base en el lenguaje de programación llamado C y utilizando el sistema operativo original de *LEGO.* Además de esto, el programa corre bajo otras plataformas además de *WINDOWS* como lo es *UNÍX O LINUX*.

Debido a que el programa esta basado en solo algunas librerías de C, se usan las principales estructuras tanto de control como de programación que permiten diseñar programas más elaborados a comparación del programa gráfico que originalmente maneja. El nuevo programa creó instrucciones que tuvieran una programación lógica y entendible. Todo esto se ha conseguido a costa de algunas limitantes que se mencionan a continuación:

- El número de variables se limita a 32.
- Solo se pueden usar variables globales que hacen que el espacio de nombres sea único, lo que no permite el desarrollo y mantenimiento de programas complejos.
- Las subrutinas no pueden devolver valores, lo cual las limita a ser subconjuntos de código, sin poder aportar funcionalidades nuevas, ocultando su implementación.
- Las subrutinas no admiten parámetros, por lo que nos es posible emplear realmente los principios de programación estructurada.
- No existen las estructuras de datos, ni las estáticas ni las dinámicas, que limitan el almacenamiento de cualquier tipo de estado, además que el espacio de memoria es de solo 6k.

III. EL LENGUAJE RCX

III.I. FIRMWARE

El RCX tiene su propio sistema operativo, el cual esta dividido en dos partes. La primera parte esta almacenada en la memoria ROM y la segunda en la memoria RAM y ésta debe ser

transmitida al RCX cuando es utilizado por primera vez. A esta segunda parte del sistema se le llama *firmware* y muestra todas la funciones básicas como guardar los registros de tiempo, controlar las salidas de los motores y monitorear los sensores.

III.II. ESTRUCTURA DEL PROGRAMA

Los programas en NQC están estructurados por tareas y una de ellas es la tarea llamada **main** o principal, que es la que será ejecutada por el robot. Una tarea está compuesta de varios comandos también llamadas declaraciones, que tienen corchetes alrededor para que esté claro que todos los comandos contenidos dentro pertenecen a esa tarea. Para definir el fin de una tarea se agrega un punto y coma al final de la línea que la describe. De esta manera se define donde una declaración acaba y donde la próxima declaración empieza.

III.III. VARIABLES

Las variables son lugares de memoria en las que se guarda un valor; este valor puede usarse en lugares diferentes y también podemos cambiarlo. Para definir una variable en el lenguaje NQC, escribimos las letras ***int*** seguidas por un nombre que nosotros escogemos, (normalmente los programadores usan minúsculas para las variables y mayúsculas para las constantes, pero esto no es necesario). El nombre debe empezar con una letra pero puede contener dígitos y guiones, ningún otro símbolo se permite. (Lo mismo pasa con las constantes, nombres de la tarea, etc). La palabra **int** está dada para enteros, es decir pueden guardarse sólo números enteros en ella. Además de agregar valores a una variable nosotros podemos realizar operaciones con las variables usando los símbolos: "* =", "- =" y "/ =". Una característica importante cuando manejamos solamente variables enteras es que para el resultado de la división se obtendrán sólo números.

IV. PROGRAMANDO CON NQC

IV.I. DECLARACIÓN IF

En ocasiones es necesario que una parte especial de un programa sólo se ejecute en ciertas situaciones. En este caso se usa la declaración if (sí) que es parecida a la declaración de while (mientras). La lógica de ésta declaración es la siguiente: sí la condición entre los paréntesis es

verdad, la parte entre los corchetes se ejecuta. Si la condición no se cumple, la parte entre los corchetes que está después de la palabra else (entonces haz...) se ejecuta. Es posible comparar valores de maneras diferentes. A continuación se anotan los más importantes:

==	igual a....
<	más pequeño que...
<=	más pequeño que o iguala a...
>	más grande que...
>=	más grande que o iguala a...
!=	no igual a...

Además, se pueden comparar las condiciones usando && que significa "y", o "||" que significa "o.". Ejemplos de condiciones:

true	siempre cierto
false	siempre falso ó nunca cierto
ttt != 3	verdadero cuando ttt no es igual a 3
(ttt >= 5) && (ttt <= 10)	verdadero cuando ttt se encuentra entre 5 y 10
(aaa == 10) || (bbb == 10)	verdadero si aaa o bbb (o ambos) es igual a 10

Debemos hacer notar que la declaración if (si) tiene dos partes. La parte inmediatamente después de la condición, que se ejecuta cuando la condición es verdad, y la parte después de la palabra else (entonces haz) que se ejecuta cuando la condición es falsa. La palabra else y la parte después de esta son opcionales. En español las declaraciones if / else se traduce como: Si (If)...la condición es cierta ...entonces haz (Then) ...si la condición no es cierta entonces haz (else)... lo que sigue...

IV.II. DECLARACIÓN DO Y DECLARACIÓN WHILE

Las declaraciones entre los corchetes después del **do,** se ejecuta con tal que la condición sea verdad. La condición **do** tiene la misma forma que la declaración **if** descrita anteriormente. Es importante distinguir que la declaración **do** (haz) casi se comporta igual que la declaración while

(mientras). Pero en la declaración de while (mientras) la condición se prueba antes de ejecutar las instrucciones siguientes, mientras la declaración do (haz) la condición se prueba al final. Es decir, que para la declaración while(mientras), las instrucciones podrían no ejecutarse nunca, pero para la declaración do(haz) se ejecutan por lo menos una vez.

IV.III. TAREAS, SUBRUTINAS Y FUNCIONES INLINE

Los programas de NQC pueden tener tareas múltiples. También es posible colocar pedazos de código en subprogramas llamados subrutinas, las cuales pueden usarse en lugares diferentes en un programa. Usando tareas y subprogramas, la lógica del programa será más fácil de entender y más compacto.

Un programa de NQC contiene a lo sumo diez tareas, por lo que cada tarea debe tener un nombre, diferente al de la tarea principal (main) para diferenciarse y determinar la ejecución de dicha tarea. Las otras tareas sólo se ejecutarán cuando alguna otra que esté en marcha les indique que sean ejecutadas usando una orden de salida. En este momento en ambas tareas (la principal y la secundaria) se están ejecutando simultáneamente (para que la primera tarea continúe corriendo). Una tarea también puede detener a otra usando la orden de alto “stop”. Después esta tarea puede reiniciarse de nuevo, pero empezará desde el principio, no del lugar donde fue detenido.

Los subprogramas causan ciertos problemas. La parte buena es que ellos sólo se guardan una vez en el RCX lo que ahorra memoria debido a que el RCX no tiene tanta memoria libre. Pero cuando los subprogramas son cortos es mejor utilizar funciones inline (en línea) que no se guardan separadamente pero se copian en cada lugar en que se usan. Esto cuesta más memoria, pero problemas como el de usar expresiones complicadas, ya no se presentarán. Además no hay ningún límite en el número de funciones inline.

Definir y llamar funciones inline se hace exactamente de la misma manera que usamos para los subprogramas. Sólo que en este caso se utiliza la palabra clave void (nulo) lugar de sub. La palabra void (nulo) se usa porque esta misma palabra aparece en otros lenguajes de programación como C. Las funciones Inline tienen otra ventaja más sobre los subprogramas, pueden tener argumentos para pasar un valor a ciertas variables a una función inline.

V. SENSORES

V.I. EL SOFTWARE DEN LOS SENSORES

El robot hexápodo podrá interactuar con el medio que le rodea gracias a los sensores de luz y de tacto que están colocados en su periferia. Esto permitirá que la programación pueda estar dirigida para que el robot siga instrucciones que dependan de los sensores.

Existen cuatro tipos diferentes de sensores y la forma de especificar el tipo de sensor se designa con el comando *SetSensorType*():

1) SENSOR_TYPE_TOUCH: para el sensor de tacto.
2) SENSOR_TYPE_LIGHT: para el sensor de luz.
3) SENSOR_TYPE_TEMPERATURE: para el sensor de temperatura.
4) SENSOR_TYPE_ROTATION: para el sensor de rotación.

La siguiente tabla muestra los niveles de voltaje requeridos en modo RAW y sus equivalentes en grados Celsius, luxes y código binario de los sensores al momento de estar activos:

Volt	Modo Raw	Sensor Ohms	Sensor de luz	Sensor de temperatura[°C]	Sensor de tacto
0.0	0	0	-	-	1
1.1	225	2816	-	70.0	1
1.6	322	4587	100	57.9	1
2.2	450	7840	82	41.9	1
2.8	565	12309	65	27.5	0
3.8	785	32845	34	0.0	0
4.6	945	119620	11	-20.0	0
5.0	1023	Inf	0	-	0

V.II. DE LUZ Y DE TACTO (fig. V).

El sensor de luz además de emitir una señal luminosa mide la cantidad de esta energía en una dirección particular. De esta manera es posible apuntar el sensor de luz en una dirección particular y hacer una distinción entre la intensidad del objeto en esa dirección. Esto es útil cuando intentamos hacer un robot que siga una línea en el suelo.

El RCX tiene sólo tres entradas para conectar tres sensores a él. Cuando se requieran hacer robots más complicados, y se tengan algunos sensores extras, es posible conectar dos o más a una entrada. Un ejemplo fácil es conectar dos sensores de tacto a una entrada. Si uno de ellos o ambos está presionado, el valor es 1, si no esta presionado el valor es 0.

Es posible conectar un sensor de tacto y un sensor de luz a una entrada. Debemos adoptar el modo en bruto (raw) y cuando el sensor de tacto se presiona se consigue un valor en bruto debajo de 100. Si no se empuja se consigue el valor del sensor de luz, el cual nunca está debajo de 100.

Existe la posibilidad de construir un sensor de proximidad por medio del sensor de infrarrojos que esta en el RCX y del sensor de luz que es sensible a ésta luz. La forma de lograrlo es mediante el envío de mensajes infra-rojos incluidos en una tarea, mientras que otra tarea, mida las fluctuaciones en la intensidad de luz que se refleja de los objetos. Entre más alta esté la fluctuación, más cerca estaremos a un objeto.

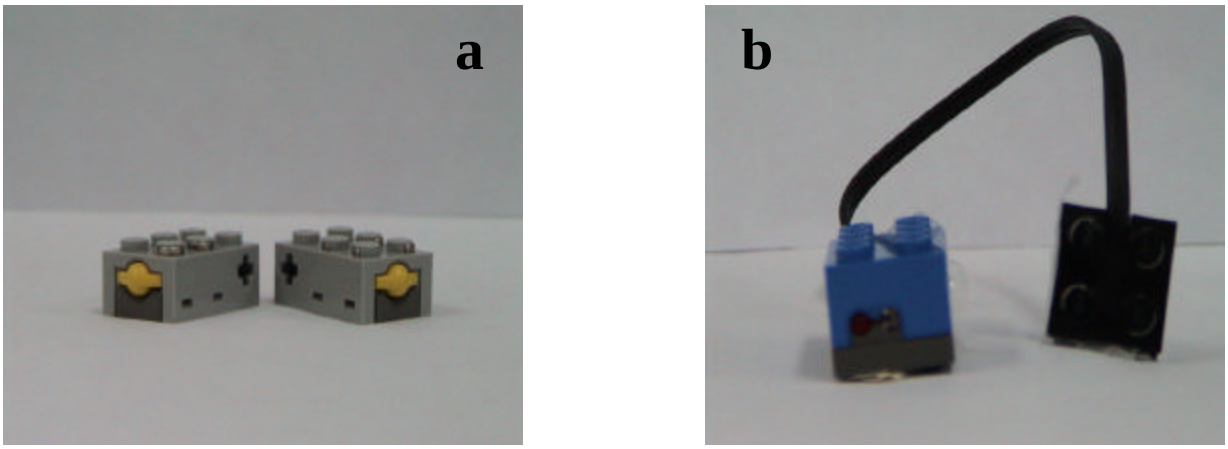


Fig. v) a)Sensor Tacto b) Sensor de Luz

VI. MOTORES

Al RCX se le pueden colocar tres motores en sus salidas nombradas como puerto A, B y C. Éstos motores son de corriente directa, por lo tanto su giro no es exacto como los motores de pasos, impidiendo situarlos en un lugar y tiempo determinado. El control de los motores es por medio de comandos especialmente diseñados que se encuentran en NQC; con ellos se indica la velocidad y sentido con cual se moverá el motor además de determinar el puerto al cual esta conectado, lo que

permite elegir el encendido y apagado. La siguiente descripción indica el uso de los comandos además de su sintaxis:

OnFwd(OUT_A):

Esta declaración dice al robot que debe encender la salida A (output A).

Wait();

Esta declaración indica que esperemos un determinado número de centésimas de segundo.

OnRev(OUT_A+OUT_C);

Esta instrucción permite invertir el giro de los motores que están conectados a las salidas A y C.

Off(OUT_A+OUT_C);

Al incluir esta instrucción en un programa se está indicando que los motores se detendrán y se apagaran.

SetPower:

Para cambiar la velocidad se usa el comando SetPower(), que podrá variar la intensidad del motor en un intervalo comprendido entre 0 y 9.

Off() y Float():

El motor se detiene inmediatamente usando el freno. A diferencia del programa que originalmente viene en el kit de LEGO, en NQC es posible detener los motores de una manera más suave sin usar el freno, empleando el comando Float().

OnFwd()

Ésta instrucción realiza dos operaciones: enciende los motores y hace que estos den vuelta hacia adelante.

OnRev():

Al igual que la instrucción anterior tiene dos aplicaciones: enciende los motores y hace que estos giren hacia atrás.

SetDirection(): que indica la dirección del motor mediante las expresiones OUT_FWD, OUT_REV o OUT_TOGGLE.

SetOutput(): que enciende o apaga los motores mediante las expresiones OUT_ON, OUT_OFF o OUT_FLOAT.

VI.I. VELOCIDAD DEL MOTOR

Debido a que los motores no son de pasos, el cambiar su velocidad no tiene mucho efecto; la razón es que se está cambiando el torque y no la velocidad. El efecto de cambio de velocidad se notará solamente cuando el motor tenga una carga pesada. Para tener mejores efectos en el incremento dela velocidad de los motores se debe encender y apagar los mismos en sucesión rápida, esto se realiza por medio de un programa que realiza esta operación.

VI.II. COMUNICACIÓN ENTRE ROBOTS.

Para poder tener un control mas adecuado de los motores y sensores, el robot hexápodo está controlado por dos RCX, lo que implica una comunicación entre ellos. Las acciones que deban tomar depende en gran medida de lo que los sensores reciban del medio donde se encuentra, por lo que una coordinada comunicación entre RCX le permitirá desenvolverse mejor al robot hexápodo.

La comunicación entre robots en general trabaja de la siguiente forma: un robot puede usar el comando `SendMessage()` para enviar un valor entre 0-255 hacia el puerto infrarrojo. El robot que cumple el papel de receptor guarda el mensaje y verifica qué valor tiene el mensaje enviado usando `Message()`. Basado en este valor el robot emisor podrá hacer que el robot receptor inicie ciertas tareas asignadas a este número.

El nombre asignado al robot emisor es el de maestro. El robot receptor recibe el nombre de esclavo. A veces el robot esclavo envía información de vuelta al robot maestro, por ejemplo el valor de un sensor. Por lo que es necesario hacer dos programas, uno para el amo y uno para el esclavo.

Las siguientes instrucciones son utilizadas para enviar los mensajes de un robot a otro:

ClearMessage():con esta instrucción se limpia el espacio de memoria donde va a estar alojado el mensaje.

Message(): es la variable donde se guarda el número o nombre del mensaje enviado.

SendMessage(): envía el pedazo de código que lleva las instrucciones a seguir por el robot esclavo.

Existen dos problemas fundamentales en la comunicación entre robots:

1. Si dos robots (o un robot y la computadora) envían información al mismo tiempo ésta podría perderse.

2. El segundo problema es cuando la computadora envía un programa al mismo tiempo a robots que estén encendidos, esto generará una confusión de direccionamiento del mensaje.

Para poder solucionar este segundo problema se debe apagar uno de los robots con los que se están trabajando para que el robo

t que queda encendido pueda almacenar el codigo o programa que se requiere bajar. Esto es importante a la hora de manejar un robot esclavo y uno maestro ya que para poder comunicarse y realizar tareas, deben tener programas distintos.

La solución al segundo problema es escribir un código que indique que ya se ha recibido el mensaje. Lo que impedirá que se envíen mensajes a destiempo o que no se les de la pausa de terminar con la tarea que se ha enviado en el mensaje.

VII. DATALOGGING

El RCX puede guardar valores de variables, lecturas del sensor y cronómetros en un pedazo de memoria llamado ***DATALOG***. Los valores en el datalog dentro del RCX no pueden usarse, pero pueden ser leídos por la computadora. Esto es útil, por ejemplo, para revisar la secuencia del programa. RCX Command Center tiene una ventana especial en la que es posible ver los contenidos actuales del datalog.

Para poder utilizar correctamente el datalog es necesario ejecutar los siguientes pasos:

1. El programa en NQC debe definir el tamaño del datalog que se va a utilizar con el comando `CreateDatalog()`. Esto también limpia los contenidos actuales del *DATALOG*.
2. Los valores pueden ser escritos en el DATALOG usando al comando AddToDatalog().
3. El tercer paso es subir el DATALOG a la computadora. Para esto se elige en RCX Command Center el commando DATALOG en el menú de Herramientas. Después se aprieta el botón etiquetado como Upload Datalog, y todo los valores aparecen. Es posible modificarlos o solamente guardarlos en un archivo.

BIBLIOGRAFÍA.

1. ADAMS, B., BREAZEAL, C., BROOKS, R., "Humanoid Robots: A New Kind Of Tool", IEEE Intelligent Systems, Julio – agosto, 2000.
2. ALCANTARA G., M. A., *NEURORED: Simulador y analizador de redes neuronales tipo biológico.* Tesis de Ing. en Computación, Fac. de Ingeniería, UNAM, México, 1992.
3. ALI A MINAI, TIRUNELVELI, "Stimulus-Induced Bifurcations in Discrete-Time Neuronal Oscillators ", *Biological Cybernetics*, Vol. 79, 1998.
4. - BEER R., HILLEL J. CHIEL AND LEON S. STERLING, "An Artificial Insect" *American Scientist*, Vol. 4. septiembre-octubre, 1991.
5. BEER, R. D., QUINN, R. D., CHIEL, H. J. AND RITZMANN, R. E., "Biologically Inspired Approaches to Robotics", *Communications of the ACM*, Vol. 40, No. 3, pgs. 31-38, March 1997.
6. BEER, R. D., RITZMANN, R. D. AND MCKENNA, T., *Biological Neural Networks in Invertebrate Neuroethology and Robotics*, Academic Press, USA 1993.
7. BERNS, K., ILG, W., DECK, M., ALBIEZ, J., DILLMAN, R., "Mechanical Construction and Computer Architecture of the Four-Legged Walking Machine Bisam" *IEEE/ASME Transaction and Mecatronics*, Vol 4. No. 1, pp. 32-38, marzo, 1999.
8. CHAPIN, K. K., MOXON, K. A., MARKOWITZ, R. S., AND NICOLELIS, M. A. L., "Real-Time Control of a Robot Arm Using Simultaneously Recorded Neurons in the Motor Cortex", Nature Neurosciences, Vol. 2, pgs. 664-670, 1999. .
9. CHIEL, H. J., BEER, R. D., QUIN, R. D., ESPENCHIED S. K., "Robustness of a Distributed Neural Network Controller for Locomotion in a Hexapod Robot", *IEEE Transactions and automation*, vol 8, No. 3, pp. 293-303, junio, 1992.
10. ESPINOSA I., "Los Robots Flexibles y las Redes Neuronales", *Ciencias*, No. 5 Especial, pp. 89-103,enero 1991.
11. ESPINOSA, I., "Ambientes Virtuales (Av) para Robots Móviles", *Trabajo Libre para presentar en el XXII Congreso de Ingeniería Biomédica*, septiembre 8, 1999.
12. ESPINOSA, I., "Las Redes Neuronales Artificiales", *Ciencia y Desarrollo*, Vol.12 No. 163,Abril 1990.
13. ESPINOSA, I., GONZÁLEZ, H., QUIZA, J., GONZÁLEZ, J., ARROYO, R., LARA, R,. "Neurodynamics Oscillators", *Daesso Han Goddard Space Fligt Center e Instutito de Investigaciones Aplicadas y en Sistemas*, pp. 255-262, marzo 23 – 25, 1994.

14. ESPINOSA, I., JIMÉNEZ, J., LARA, R., ESPINOSA J., "Simulación de Poblaciones de Osciladores Neuronales Básicos como Memorias Asociativas y Optimizadoras." *Congreso Latinoamericano de Ingeniería Biomédica,* pp. 543-546, noviembre 1998.
15. FRANK C., HOPPENSTEADTD, EUGENE M. IZHIKEVICH "Synaptic Organizations and Dinamical Properties of Weakly Connected Neural Oscillators ". *Biological Cybernetics* **,** Vol. 75,1996.
16. HODGKIN, A. L., HUXLEY, A. F. AND KATZ, B., "Measurement of Current-Voltage Relations in the Membrane of the Giant Axon of Loligo ", *Journal of Physiology*, Vol. 116, pgs. 424-448, 1952.
17. http://www.cs.uu.nl/people/markov/lego/.
18. ITO, S., YUASA H., LUO, Z., ITO, M., YANAGIHARA, D., "A Mathematical Model of Adaptative Behavior in Quadruped Locomotion", *Biological Cybernetics,* Vol. 78, pp. 337-347, 1998.
19. KANDEL, E., HAWKINS, R., "Bases Biológicas del Aprendizaje y de La Individualidad", *Investigación y Ciencia,* pp., Noviembre, 1992
20. KIMURA, S., YANO, M., SHIMIZU, H., "A Self-Organizing Model of Walking Patterns of Insects", *Biological Cybernetics,* pp. 505-512, Vol. 69, 1993.
21. KOSKO, B., "Constructing and Associative Memory", *Byte* no. 137, pp. 137-44, Septiembre 1987.
22. LARA, R., JIMÉNEZ, J., ESPINOSA, I., "Redes neuronales con propiedades de oscilación, memoria y optimización y su posible aplicación en la robótica Flexible", *SOMI XIII Congreso de Instrumentación,* pp. 255-259, octubre 1998.
23. LIPPMANN, R., "An Introduction to Computing With Neural Nets", *IEEE ASSP Magazine*, pp. 5-23, Abril, 1987.
24. LÓPEZ, J., "Cajal y la Estructura Histológica del Sistema Nervioso", *Investigación y ciencia, pp. 6-13* , Febrero, 1993.
25. MACGREGOR, R.J., *Neural and Brain Modeling*, Academic Press, USA, 1987.
26. MATSUOKA, K., "Sustained Oscillators Generated by Mutually Inhibiting Neurons With Adaptation", *Biological Cybernetics,* Vol. 52, pp. 367-376, 1985.
27. Oct. 2, 1999.
28. OVERMARS M., *"Programando Robots Lego usando NQC, version 3.03"*, Department of Computer Science, Utrecht University, pgs. 2-47,
29. QUINN, R., "Biologist and Engineers Create a New Generation of Robots That Imitate Life", *Science,* Vol. 288, pp. 80-83, Abril del 2000.
30. Santos, S., "Introducción a las Redes Neuronales Artificiales", www.