ESCUELA POLITÉCNICA DEL EJÉRCITO

FACULTAD DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TITULO
EN INGENIERÍA ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA BASADO EN UNA RED NEURONAL NO SUPERVISADA PARA EL CONTROL DE MOVIMIENTO DE UN ROBOT MÓVIL

Gustavo Adolfo Moreno Jiménez

QUITO - ECUADOR
2005

CERTIFICACIÓN

Certificamos que el presente proyecto de grado titulado “Diseño e implementación de un sistema basado en una Red Neuronal no supervisada para el control de movimiento de un Robot Móvil“ ha sido desarrollado en su totalidad por el Sr. Gustavo Adolfo Moreno Jiménez, como requisito previo para la obtención del titulo de Ingeniero Electrónico.

Sangolquí, Agosto del 2005.

_____	_____
Ing. Hugo Ortiz	Ing. Víctor Proaño
DIRECTOR	CODIRECTOR

AGRADECIMIENTO

A mi familia y amigos sin los cuales hubiese sido muy difícil llevar a cabo este proyecto, gracias por colaborar conmigo todo este tiempo.

DEDICATORIA

A todas aquellas personas que me han cambiado y han hecho de mi una mejor persona, que han hecho que me esfuerce y me han hecho dar todo de mi, en especial a mi madre, Martha Lucia, que me apoyo e incentivo para realizar este proyecto.

PRÓLOGO

Este trabajo contiene el estudio y diseño de una Red Neuronal Artificial para el control de movimiento de un Robot móvil, el diseño del software y hardware necesarios para simular su comportamiento mediante el uso de microcontroladores y el diseño de la plataforma móvil que será controlada por la red.

En el estudio de las Redes Neuronales Artificiales se hace un breve resumen del funcionamiento biológico de las neuronas, así como de los tipos más conocidos de Redes Artificiales; Red Multicapa, Modelo Kohonen y Modelo Hopfield. Se describen los procesos de aprendizaje, niveles de activación, capas y algunas de las aplicaciones de estas redes.

En el diseño de la Red Neuronal Artificial se explica la topología usada en la red neuronal que será simulada en el proyecto, su forma de aprendizaje, influencia entre neuronas y las señales de error que se pueden producir y como afectan su aprendizaje.

También se presentan todos los elementos necesarios para el diseño de hardware que simulará el comportamiento de la Red Neuronal y el diagrama de conexión, así como el software necesario para esta simulación. Además el esquema de la plataforma, para la cual se diseña la red, con las características físicas como tamaño y forma, y las características eléctricas de los diferentes dispositivos.

Finalmente se hace un análisis de los resultados de la red neuronal, los pesos aproximados en los cuales se estabiliza la red y el comportamiento esperado de la plataforma.

ÍNDICE

CAPITULO 1

INTRODUCCIÓN

1.1. ANTECEDENTES……..…… 1
1.2. JUSTIFICACIÓN E IMPORTANCIA DEL PROYECTO……. 1
1.3. OBJETIVOS ……..……. 2
1.4. DESCRIPCIÓN GENERAL ……..…….. 3

CAPITULO 2

REDES NEURONALES ARTIFICIALES

2.1. REDES NEURONALES BIOLÓGICAS…….…… 7
2.2. DEFINICIÓN DE RED NEURONAL ARTIFICIAL ……..……. 9
2.3. CARACTERÍSTICAS DE LAS REDES NEURONALES ARTIFICIALES.... 10
2.4. ALGUNOS TIPOS DE REDES NEURONALES …….. 13
2.5. APLICACIÓN DE LAS REDES NEURONALES ARTIFICIALES ……. 22

CAPITULO 3

ARQUITECTURA DE LA RED NEURONAL ARTIFICIAL

3.1. DEFINICIÓN DEL PROBLEMA ……. 24
3.2. DISEÑO DE TOPOLOGÍA DE LA RED NEURONAL ……...……. 26
3.3. FORMA DE APRENDIZAJE ……..…….. 29

CAPITULO 4

IMPLEMENTACIÓN DE LA RED NEURONAL ARTIFICIAL

4.1. DISEÑO DEL HARDWARE ……..…….. 34
4.2. DISEÑO DEL SOFTWARE ……..…….. 37

CAPITULO 5

DISEÑO E IMPLEMENTACIÓN DE LA PLATAFORMA

5.1. PARTES ……..…….. 55
5.2. PLATAFORMA ……..……. 63

CAPITULO 6

PRUEBAS Y RESULTADOS

6.1 RESULTADOS DE LA RED NEURONAL ……...…..…… 66
6.2. RESULTADOS DE LA PLATAFORMA ……..……. 71
6.3. POSIBLES CAMBIOS EN LA RED NEURONAL..…….. 73

CAPITULO 7

CONCLUSIONES Y RECOMENDACIONES

7.1. CONCLUSIONES ……..……. 75
7.2 RECOMENDACIONES …… 76

CAPITULO 1

INTRODUCCIÓN

1.1 ANTECEDENTES

El control de movimiento de plataformas se realiza con microcontroladores y algoritmos complejos que hacen un análisis de las condiciones externas y de acuerdo a estas condiciones reaccionan según la programación previa que estos contengan para llevar a cabo la tarea asignada.

Los sistemas de control para el movimiento de plataformas móviles requieren de un complejo análisis de las condiciones de funcionamiento así como el comportamiento de los sensores, la forma como se este controlando el sistema, los motores utilizados para el desplazamiento, la carga de la plataforma móvil y otras condiciones, lo que hace que el diseño sea muy complejo y poco flexible.

Para estos sistemas se usan sensores de proximidad que permiten conocer la distancia entre la plataforma y los objetos para que el controlador evite chocarse contra estos. Estos sensores de proximidad suelen ser de tipo ultrasónico o infrarrojos.

1.2 JUSTIFICACIÓN E IMPORTANCIA DEL PROYECTO

La implementación de robots móviles ha demostrado la importancia de la flexibilidad en el desarrollo de controladores inteligentes, por ello el controlador que se desarrollará basándose en una Red Neuronal tendrá la posibilidad de conectarse con distintos tipos de sensores que pocos sistemas diseñados en nuestro ámbito poseen. Esta iniciativa permite a este proyecto ser uno de los pioneros en la Facultad de Ingeniería Electrónica de la ESPE relacionado con el control de robots móviles basados en redes Neuronales.

La necesidad de encontrar formas sencillas de controlar los movimientos automáticos de los robots debido a la complejidad del medio en que actúan hace indispensable el diseño de nuevos sistemas de control. Las Redes Neuronales son una de las opciones para el desarrollo de estos sistemas ya que ofrecen Simplicidad, flexibilidad y la posibilidad de tener un aprendizaje autónomo.

El rápido avance de la electrónica hace necesario que los nuevos sistemas de control tengan la capacidad de conectarse con dispositivos de características tecnológicas emergentes independiente de los avances en ellos establecidos, por lo que un controlador flexible es un buen camino en la libre modernización de los sistemas.

La importancia radica en la facilidad que para las personas ajenas a la electrónica sería tener sistemas adaptables que no requieran una programación o ajustes complicados para su buen desempeño, sino que aprendan y se adapten por si mismos a las condiciones en las que deben realizar su trabajo.

1.3 OBJETIVOS

1.3.1 Objetivo General

- Diseñar e implementar un sistema basado en una red neuronal no supervisada para el control de movimiento de un robot móvil.

1.3.2 Objetivos Específicos

- Estudiar las características de las condiciones en las que se desempeñan los robots móviles.

- Estudiar y comprender los problemas que presentan las arquitecturas de Redes Neuronales en el control de Robots.

- Diseñar una arquitectura de Red Neuronal flexible para que pueda adaptarse a situaciones en las que cambien las condiciones del entorno.

- Diseñar una arquitectura de Red Neuronal capaz de expandirse fácilmente mediante la conexión con otras Redes Neuronales, permitiendo así agregar nuevos sensores y actuadores que también puedan influir en el comportamiento del proceso.

- Diseñar e implementar una plataforma móvil

- Obtener un controlador que pueda tener redundancia en su funcionamiento mediante neuronas adicionales que entrarían en fase de aprendizaje en el momento en que falle una neurona activa.

- Implementar el controlador para la plataforma móvil

1.4 DESCRIPCION GENERAL

El presente trabajo abre un nuevo campo de investigación en los sistemas de control con redes neuronales basándose en el diseño de un nuevo tipo de arquitectura de red neuronal.

Las redes neuronales son sistemas que mediante elementos electrónicos intentan imitar el comportamiento del cerebro biológico, esto con el fin de obtener las grandes ventajas que estos posee en comparación con los sistemas electrónicos tradicionales. Algunas de estas ventajas son flexibilidad, adaptabilidad, capacidad de aprendizaje, el procesamiento de gran cantidad de información en corto tiempo y la simplicidad en sus conexiones y estructuras.

Las redes neuronales tienen un gran campo de aplicación en todo lo referente a robots que interactúan con los seres humanos. Por ejemplo podemos apreciar los robots móviles que encuentran cada día nuevos lugares en los cuales desempeñarse; es muy importante conocer las características de estos nuevos ambientes y diseñar sistemas que les permitan a los robots tener un buen comportamiento.

La mayor parte de los ambientes en los que se pueden encontrar actividades para ser realizadas por robots son habitados por seres humanos por lo tanto las características mas comunes en estos ambientes serán: las variaciones en la cantidad de luz, variaciones en la distribución de los objetos, paredes y objetos con distintas características de reflexión de luz, sonido, diferencias de colores y formas, y otras.

Las variaciones en la cantidad de luz, se dan porque la mayor parte de los lugares en los que habita el ser humano están iluminados por la luz natural del sol, esta luz natural es propensa a cambios climáticos, estacionales y a la rotación de la tierra.

Variaciones en la distribución de los objetos, al tener humanos y objetos en el mismo entorno es de esperar que estos últimos sean manipulados por los humanos luego de lo cual serán ubicados en un lugar distinto del que se encontraban inicialmente y los robots no tendrán conocimiento de este cambio.

Paredes y objetos con distintas características, las diferencias entre los distintos objetos que se pueden encontrar en cualquier ambiente son fundamentales ya que los robots usan sensores que se pueden ver afectados por la diferencia de color, forma, material u otra característica del objeto y llevar al robot a comportarse de forma errada.

Existen dos formas de solucionar estos problemas de manera que un robot pueda tener un comportamiento aceptable, la primera forma es logrando que el ambiente sea lo suficientemente estable para que los sensores no se vean afectados, y la segunda forma es logrando que el robot pueda cambiar su comportamiento de acuerdo a las condiciones del ambiente.

En la primera se necesitan crear y mantener las condiciones en las que se realizarán las tareas; como áreas libres por las que puedan circular los robots, condiciones estables de luz, temperatura, presión o cualquier otra variable física que pueda afectar a los sensores; marcas en los objetos que encontrará el robot de manera que este pueda reconocerlos y cualquier otra situación necesaria para que no se confunda. Aunque esta es la forma mas

utilizada presenta demasiados inconvenientes para lugares en los que habiten seres humanos. Además se deben utilizar complejos algoritmos que determinarán el comportamiento del robot en cada uno de los casos en los que se tendrá que desenvolver y se tienen que incluir muchos sensores para todas las variables que consideremos puedan afectar el comportamiento. El mas grave problema en esta solución es que si por algún motivo se falla en los sensores, o en el número de ambientes distintos en los que se desempeña el robot o en sus características será necesario rediseñar por completo al robot y sus reglas de comportamiento.

En la segunda forma de controlar robots lo que se pretende es lograr que un robot aprenda por si mismo cuales son las reglas necesarias para su buen desempeño y conozca cuales sensores son necesarios. De esta forma y si se olvida incluir sensores para alguna condición física que pueda afectar al robot bastaría con conectar un sensor que mida esta condición y el robot por si mismo lo integraría a sus propias reglas de comportamiento. Esto representa una gran ventaja respecto a la solución anterior ya que no es necesario ningún estudio previo de las condiciones o ambientes en los que se encontrará el robot. En este tipo de soluciones se pueden reconocer dos etapas fundamentales.

En la primera etapa, de aprendizaje, la red ajusta el valor de las conexiones de entrada (pesos) de acuerdo a la salida deseada y a los estímulos de entrada dados, de manera que se minimice el error. En la segunda etapa la red solo debe responder a los estímulos de entrada con la salida adecuada.

La etapa de aprendizaje puede ser supervisada cuando la red posee un algoritmo de aprendizaje que cuenta con un apoyo externo o “maestro” que corrige la salida de la red de acuerdo con la salida que se considera correcta; o no supervisada cuando no se tiene apoyo externo y la red debe aprender, sin ayuda, de los datos que se suministran.

Las redes neuronales pueden ser realizadas de varias maneras, en hardware mediante transistores de efecto de campo (FET), amplificadores operacionales, etc., o en software mediante computadoras y programas que simulan el comportamiento de las neuronas en una red.

Finalmente en este trabajo se desarrolla una arquitectura de red neuronal que permite a un robot moverse en una habitación y evitar luego de la etapa de aprendizaje chocarse con las paredes y objetos que se encuentren en esta habitación; las condiciones de la habitación como son: luz, color de las paredes y objetos, las texturas, los materiales y la distribución; podrían cambiar sin que esto afecte el desempeño final. Todo esto sin la necesidad de unas reglas fijas impuestas por la persona que programó el robot.

CAPÍTULO 2

REDES NEURONALES ARTIFICIALES

2.1 REDES NEURONALES BIOLÓGICAS

Mediante las Redes Neuronales Artificiales se intenta, muchas veces, imitar los procesos de aprendizaje y control del cerebro biológico, por lo que se facilita su entendimiento si se tienen algunos conocimientos de las Redes Neuronales Biológicas.

Los cerebros biológicos están formados de muchas neuronas conectadas entre si utilizando la información recibida para dar una respuesta a cada situación. La neurona biológica esta compuesta de las partes que se puede observar en la figura 2.1.

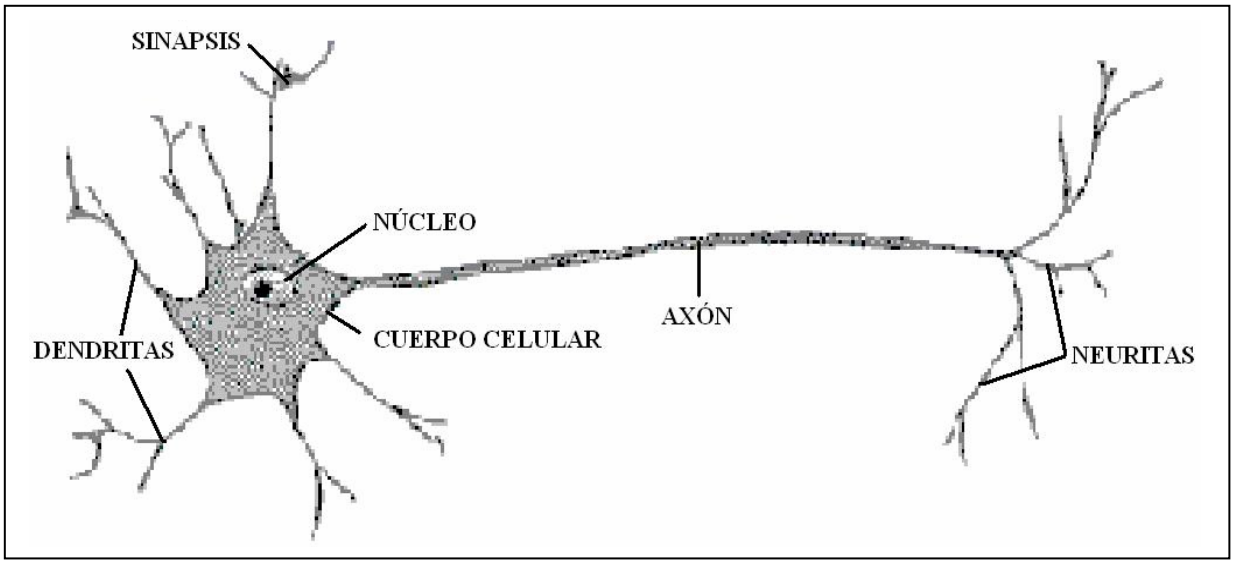


Figura. 2.1 Neurona Biológica

Desde el cuerpo celular se extiende una rama principal llamada Axón que a su vez se ramifica en Neuritas, estructura que emite los impulsos de salida hacia las demás neuronas. También se extienden ramas cortas llamadas dendritas que son las estructuras de

entrada y permiten recibir las señales desde otras neuronas. La conexión entre el Axón de una neurona y la dendrita de otra es conocida como sinapsis. Las neuronas, a través de la sinapsis, reciben señales eléctricas, pequeños impulsos provenientes de otras neuronas o de ellas mismas. Las sinapsis pueden variar en fuerza, unas pueden producir una señal débil y otras una señal fuerte.

Una neurona integra las señales que recibe y puede provocar un impulso de salida que le será transmitido a otras neuronas. La representación matemática aproximada del comportamiento de una neurona sería la suma de cada una de las señales de entrada multiplicada por la fuerza de su sinapsis; si este valor es mayor al nivel de activación, la neurona activará su salida, en caso contrario no habrá ninguna salida. En la figura 2.2 puede observarse la comparación entre una neurona biológica y una neurona artificial.

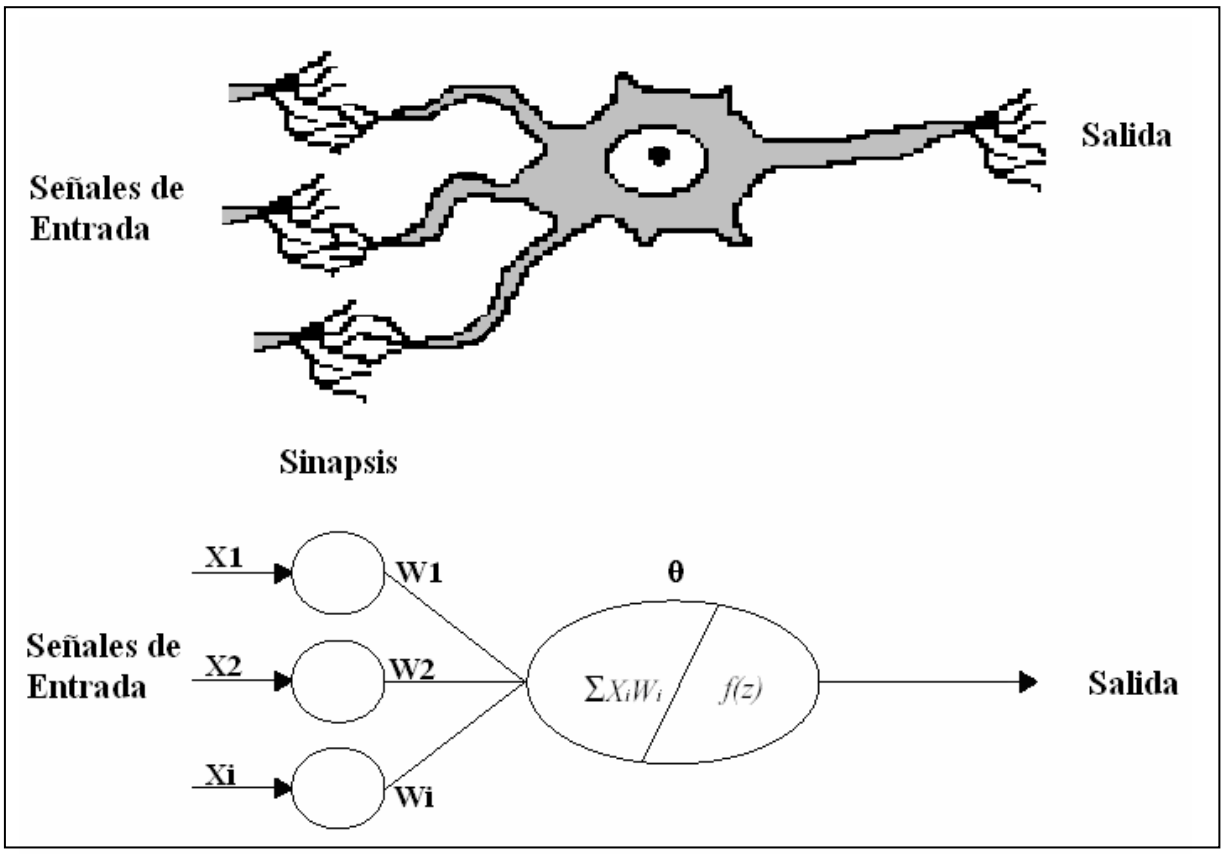


Figura. 2.2. Modelo Matemático Aproximado

$$\sum_{i=1}^{n} X_i \times W_i = W_1 \times X_1 + W_2 \times X_2 + \ldots + W_n \times X_n$$

Si $\sum_i X_i \times W_{i.} > \theta$ (Nivel de Activación) se activa la salida.

Estos conocimientos básicos de una Red Neuronal Biológica permiten comprender mejor los conceptos de Red Neuronal Artificial y los tipos de Redes Artificiales que en su mayoría buscan imitar el comportamiento biológico.

2.2 DEFINICIÓN DE RED NEURONAL ARTIFICIAL

No hay una definición aceptada universalmente de lo que es una Red Neuronal Artificial, sin embargo es aceptado que son redes con varios procesadores simples en paralelo, cada uno con una pequeña cantidad de memoria y altamente interconectados; el conocimiento es almacenado en la fuerza de sus conexiones y es adquirido a través de un aprendizaje.

Haykin[1] en su trabajo "*Neural Networks: A Comprehensive Foundation,* NY: Macmillan.", las define como:

> *"Una Red Neuronal es un procesador masivo paralelo distribuido que tiene propensión natural para almacenar conocimiento experimental, haciéndolo viable para su uso. La Red Neuronal se asemeja al cerebro en dos aspectos:*
>
> 1. *El conocimiento es adquirido por la Red a través de un proceso de aprendizaje.*
>
> 2. *La fuerza de las conexiones entre neuronas -conocidas como pesos sinápticos- es usada para almacenar el conocimiento."*

[1] Haykin, S. Neural Networks, A Comprehensive Foundation, MacMillan Collage Publishing Company. 1994.

2.3 CARACTERÍSTICAS DE LAS REDES NEURONALES ARTIFICIALES

2.3.1 Pesos

En una red neuronal Artificial los pesos representan la fuerza que existe en una sinapsis. Estos pesos pueden ser variables o fijos, en caso de ser fijos la tarea para la cual será utilizada la red neuronal debe estar definida previamente, en el caso de pesos variables estos van adaptándose a medida que la red va aprendiendo la tarea.

2.3.2 Etapa de Aprendizaje

Es la etapa en la que la Red Neuronal Artificial, modifica sus pesos en respuesta a una información de entrada y una salida deseada. La etapa de aprendizaje terminará cuando los pesos no cambien. Existen dos clases de aprendizaje, uno en el que la red aprende de forma supervisada y otro en forma no supervisada.

El aprendizaje supervisado se da mediante un maestro que determina la respuesta que debe tener la red para una información de entrada establecida, el maestro verificará la salida y en caso de tener una respuesta errónea se cambiarán los pesos de conexión**.** Los tipos de aprendizaje supervisado son:

- Por corrección de error, consiste en ajustar los pesos en función del error que se presenta entre la salida deseada y la obtenida. Tipos de redes : Perceptrón, Regla Delta (Madaline), Backpropagation, Counterpropagation.

- Por refuerzo, no se indica cual es la salida deseada, solo se indica a la red si la salida es correcta o incorrecta y en función de esto los pesos se ajustan. Tipos de redes : LRP (Linear Reward Penalty), ARP (Associative Reward Penalty), Adaptive Heuristic Critic.

- Estocástico, consiste en realizar cambios aleatorios en los pesos. Tipos de redes : Máquina de Boltzman, Máquina de Cauchy.

El aprendizaje no supervisado se da sin necesidad de un maestro, no requiere de ninguna influencia externa para ajustar los pesos de las conexiones. La red no recibe información del entorno que le indique si la salida es correcta. La salida en este tipo de redes puede ser una codificación de los datos de entrada que mantiene los datos relevantes de la información o pueden realizar un mapeo de características de forma que si se presentan datos similares se verán afectadas neuronas cercanas entre si. Los tipos de aprendizaje supervisado son:

- Hebbiano, consiste en ajustar los pesos según la correlación, si dos unidades están activas se refuerza la conexión, si una esta activa y la otra no, se debilita la conexión. Tipos de redes: Hopfield, Linear Associative Memory, Fuzzy Associative Memory, Grossberg, Bidirectional.

- Competitivo y cooperativo, en este aprendizaje las neuronas compiten para activarse y llevar a cabo una tarea, con esto se pretende que solo una neurona se active cuando se presente cierta información de entrada. Tipos de redes: Learning Vector Quantizer, Cognitrón / Neocognitrón, Teoria de Resonancia Adaptativa.

2.3.3 Capas

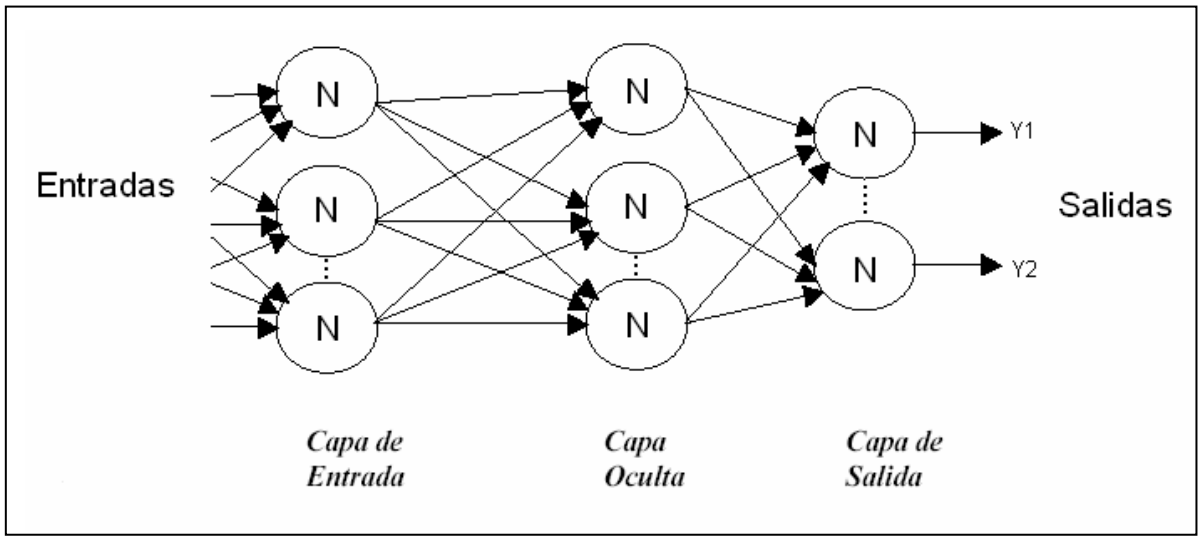


Figura. 2.3. Red Neuronal con varias capas

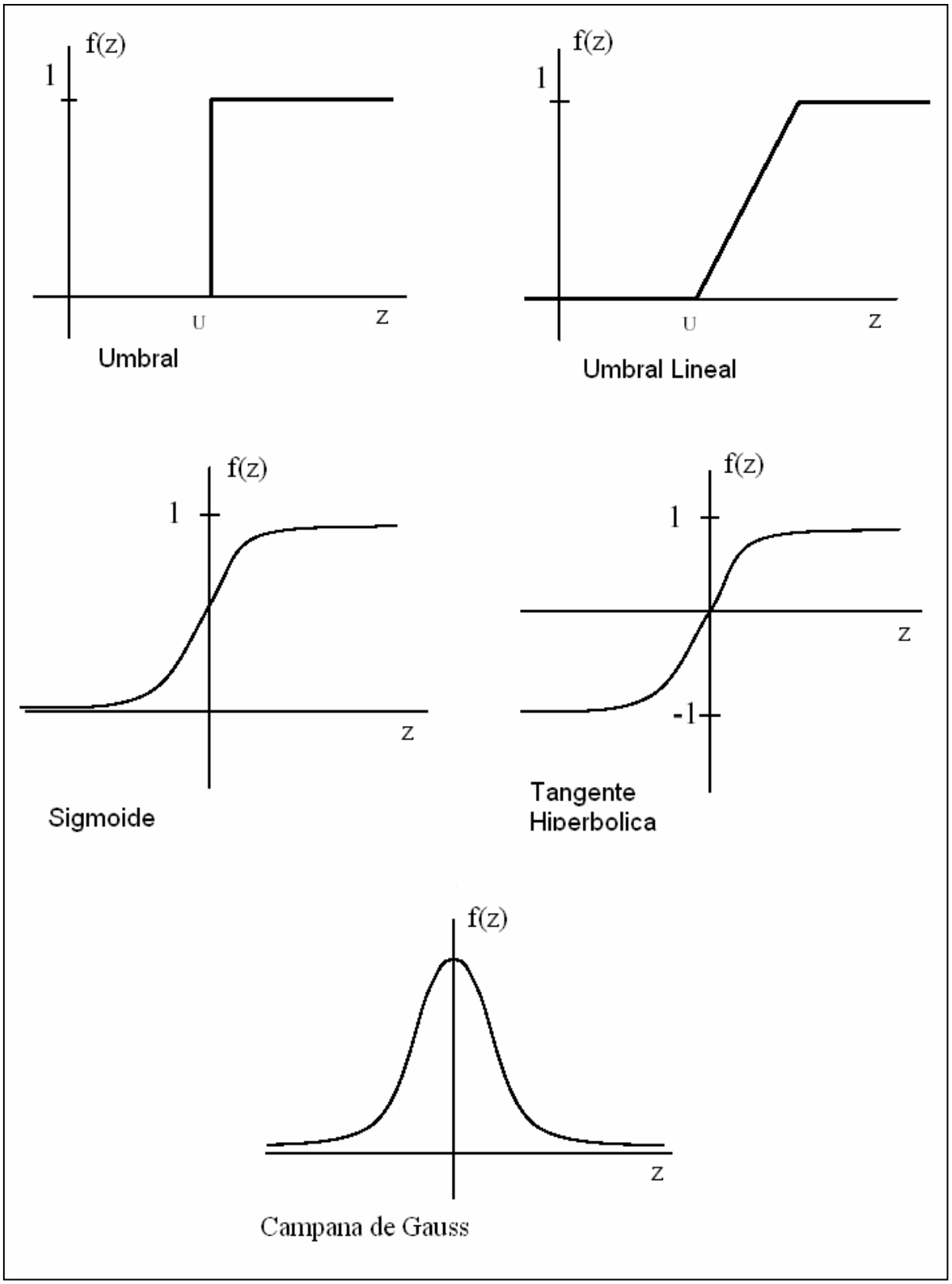


Figura. 2.4. Funciones de Activación

Una capa es un grupo de neuronas cuyas entradas proviene de la misma fuente y su salida se dirige al mismo destino. Las redes neuronales se pueden componer de tres tipos de capas: capa de entrada que son las neuronas que reciben las señales desde el medio, capa oculta son las que poseen sus entradas y salidas dentro de la red y finalmente capa de salida cuyas neuronas envían la señal fuera de la red. Una red neuronal puede tener una o mas capas ocultas. En la figura 2.3 se puede observar una red neuronal con una capa de neuronas de entrada, una capa oculta y una capa de salida.

2.3.4 Nivel de Activación

El nivel de activación o umbral es el valor a partir del cual se activa la salida de la neurona, si la suma de las entradas de la neurona multiplicadas por la fuerza de cada sinapsis (peso) es mayor a este valor umbral la neurona generará la función de activación.

2.3.5 Funciones de Activación

Es la función de salida de la neurona, valor en la salida de la neurona, algunas veces es llamada función de transferencia, esta función usualmente alimenta las entradas de otras neuronas por medio de sinapsis a menos que sea de una neurona de la capa de salida de la red. Generalmente se usan funciones de salida no lineales en redes multicapa con el fin de hacer redes que pueden resolver problemas no lineales. Las funciones de salida pueden tener distintas formas como se indica en la figura 2.4.

2.4 ALGUNOS TIPOS DE REDES NEURONALES

Existe un gran número de redes neuronales y cada día se crean nuevos modelos o modificaciones de los modelos ya existentes, por esta razón es muy difícil incluir un estudio de todas las redes neuronales existentes. A continuación se describirán solo algunas redes neuronales que presentan características interesantes.

2.4.1 Perceptrón

Intenta modelar el comportamiento de la neurona biológica, el perceptrón es la base de las Redes Neuronales En la figura 2.5 se observa el modelo del perceptrón; en el cuerpo de la neurona cada señal de entrada X_i es multiplicada por un peso W_i, los resultados son luego sumados y evaluados, si se supera un umbral θ el perceptrón activa una función de salida $Y=f(z)$.

La primera capa es un grupo de sensores que detectarán las señales específicas, el aprendizaje de la Neurona se da mediante el ajuste de los pesos W y el umbral θ, el conocimiento que se almacena en una neurona se encuentra en el valor de los pesos luego de ser ajustados en la etapa de aprendizaje.

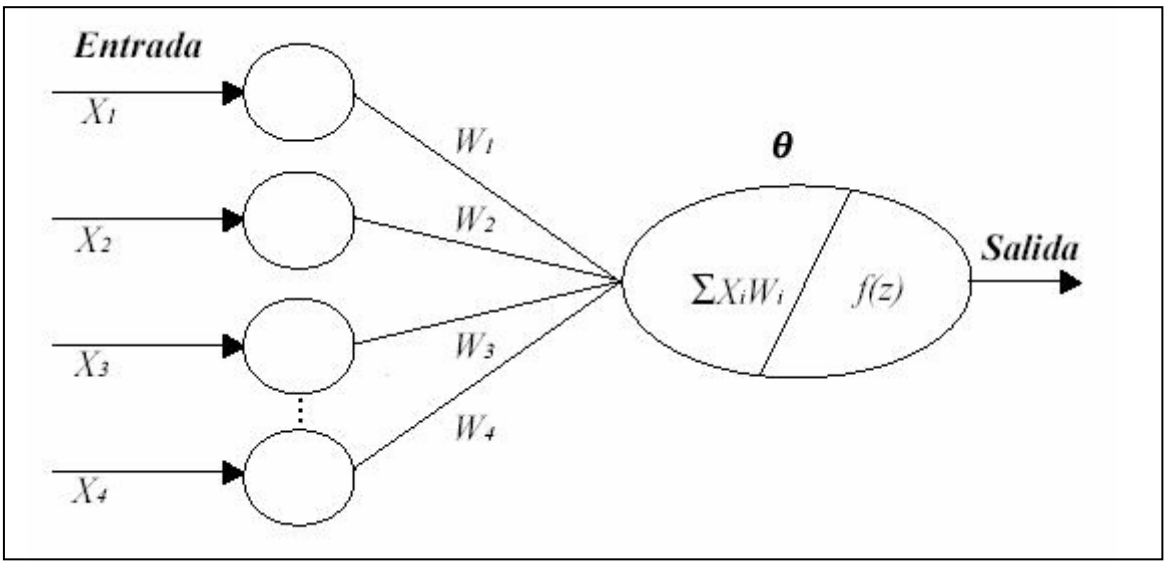


Figura. 2.5. Perceptrón

Ecuaciones.

$$\sum_i X_i \times W_i = W_1 \times X_1 + W_2 \times X_2 + \ldots + W_n \times X_n$$

Si $\sum_i X_i \times W_{i.} > \theta$ (Umbral) se activa la salida $Y = f(z)$

Un perceptrón solo no puede resolver problemas complejos y puede activar una salida, aunque posee la capacidad de aprendizaje y adaptación.

2.4.2 Modelo Kohonen

El modelo de Kohonen es una red neuronal artificial presentada por el profesor Teuvo Kohonen de la universidad de Helsinki. Este modelo se caracteriza por la capacidad de formar mapas de características en una forma similar a lo que ocurre en el cerebro, existe evidencia que las neuronas en el cerebro se organizan en zonas de forma que la información de los sentidos se representan en forma de capas. Esto quiere decir que posee la capacidad de formar mapas de la información recibida.

Es una red de tipo Auto organizada que clasifica conjuntos de datos para los que no se tiene ninguna organización, la red presenta un resultado dependiendo de la similitud entre los patrones de entrada que se tienen. Es una red no supervisada, es decir que no requiere de un maestro para su aprendizaje, competitiva y con una estructura de 2 capas, una de entrada y otra de salida. El aprendizaje no supervisado se da ya que no se tiene como objetivo una salida específica para la red neuronal.

La competitividad en el modelo Kohonen se da en el hecho que cada neurona lucha por ser entrenada e inhibe a las otras neuronas para que estas no lo sean, así la neurona que genera una mayor salida para un patrón de entrada será la que alterará sus pesos de conexión acercándose más al patrón de entrada dado.

El objetivo final de la red neuronal es categorizar los datos que se ingresan a la red, así cuando se introduce información con características similares a la red esta las clasifica formando parte de una categoría que ha sido creada por la misma red neuronal. Por esta capacidad de organizar la información en categorías este tipo de redes son muy útiles en el análisis exploratorio de datos (data mining).

En la versión original del modelo Kohonen no existen conexiones hacia atrás, se caracteriza por tener un número ***i*** de neuronas de entrada que se conectan a través de sus salidas a un número ***j*** de neuronas de salida, entre las neuronas de salida existen

inhibiciones que pueden considerarse como conexiones implícitas que influirán sobre las neuronas vecinas. En la figura 2.6 se puede observar la estructura de una de las redes neuronales presentadas por Kohonen.

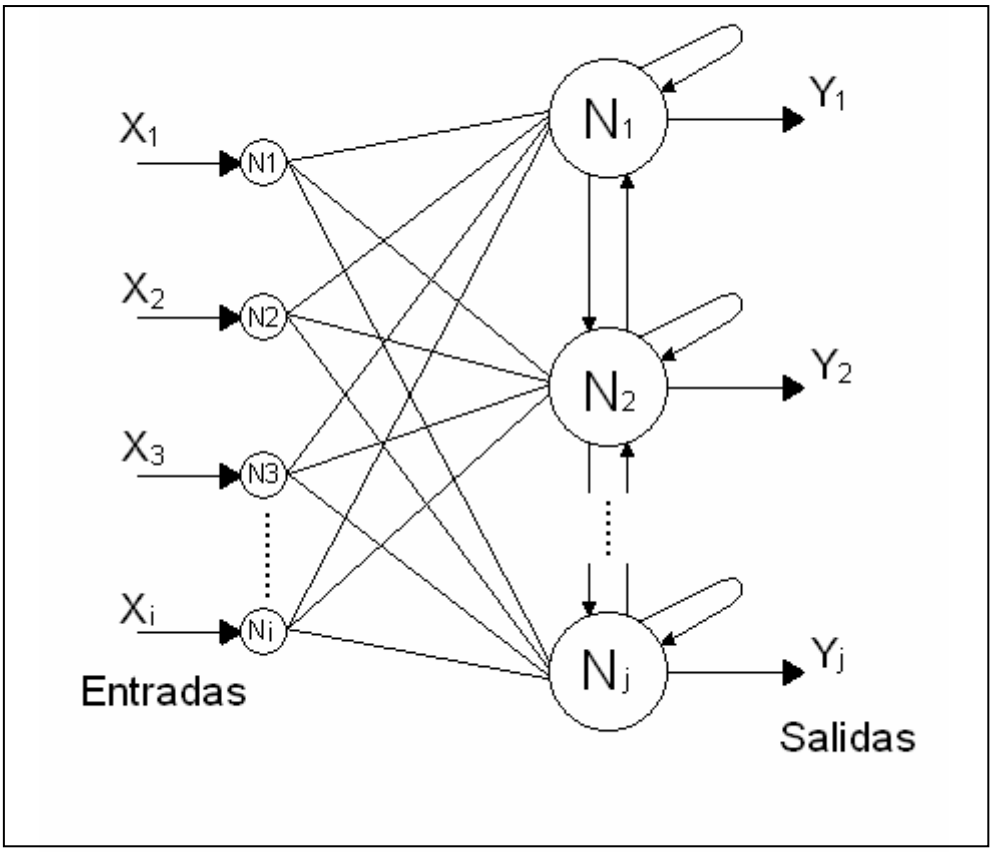


Figura. 2.6. Modelo Kohonen

La influencia que tiene una neurona de salida hacia otra neurona depende de la distancia entre estas, siendo muy usual una función de sombrero mexicano, para esta influencia. Estas inhibiciones entre neuronas se pueden considerar como conexiones con pesos negativos que en lugar de ayudar a la activación de la neurona la desactivan. Ver figura 2.7

La forma de aprendizaje de una red neuronal de este tipo se realiza de la siguiente manera. Una vez que se presenta una entrada a la red neuronal se establece cual es la neurona ganadora, esta neurona es la que tiene el vector de pesos mas parecido al vector de entrada, a continuación se puede observar uno de los criterios para establecer la neurona ganadora.

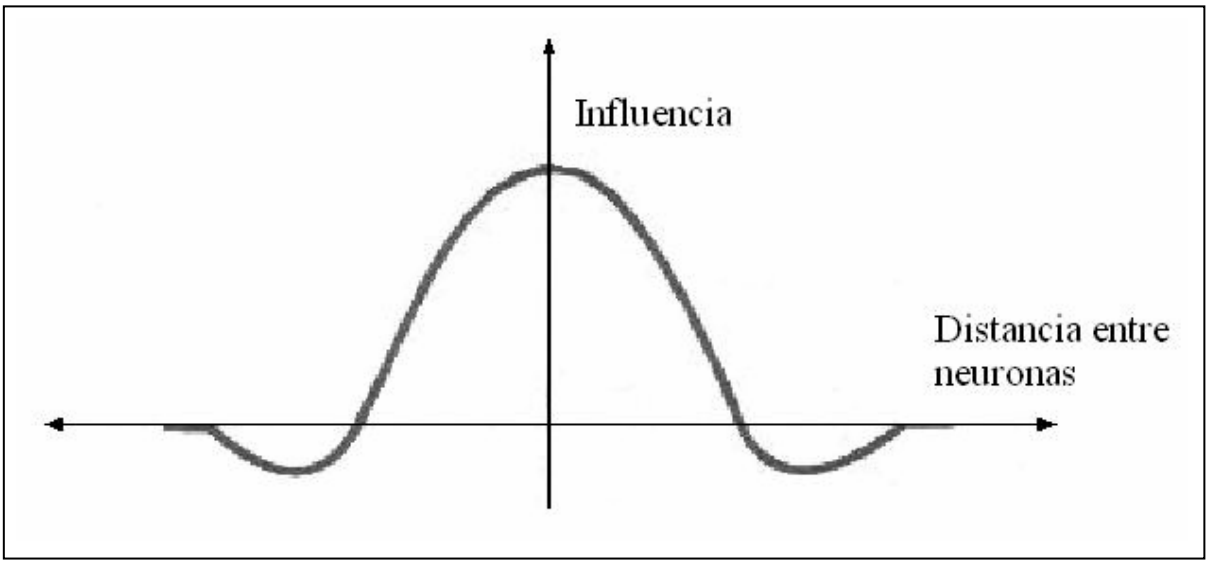


Figura. 2.7. Función de influencia entre neuronas de salida

$$\text{Neurona ganadora} = \min \sum_{i=1}^{N} (X_{pi} - W_{ji})^2$$

Donde,

Xp Es el vector de entrada

Wj Es el vector de pesos de la red

Este criterio establece a la neurona ganadora como aquella que tenga la menor distancia euclidiana entre el vector de entrada y los pesos de la neurona.

Establecida la neurona ganadora el vector de pesos de esta es alterado para que se acerque un poco al vector de entrada de manera que la siguiente vez que se presente la información la neurona reaccione con una salida aun mayor. El proceso se repite con más vectores de entrada de manera que los vectores de pesos se modifiquen hasta que cada neurona represente una categoría o dominio en el espacio de entrada.

En el caso de existir un número mayor de categorías en los vectores de entrada que el número de neuronas de salida, las neuronas podrán tener varias categorías asociadas y no solo una categoría como se daría en el caso de tener un número mayor o igual de neuronas de salida.

Durante el aprendizaje de este tipo de redes se introduce un factor adicional que es la influencia de la neurona ganadora sobre sus vecinas, este valor altera el vector de pesos de las neuronas cercanas a la neurona ganadora acercándolo al vector de entrada, aunque en una medida menor que el acercamiento que sufre el vector de pesos de la neurona ganadora. De esta forma se logra que neuronas que se encuentran cercanas topológicamente respondan a patrones de entrada similares entre si, logrando la auto organización del mapa de características.

2.4.3 Red Multicapa (Multilayer Perceptrón)

La red multicapa es uno de los modelos más comunes de red neuronal, se caracteriza por tener varias capas de neuronas, una capa de entrada, una capa de salida y una o más capas intermedias ocultas; para resolver la mayor parte de los problemas solo se requiere de una capa oculta. En esta red cada neurona de una capa proporciona una entrada para cada una de las neuronas que existen en la siguiente capa, como se puede observar en la figura 2.8.

Es una red de tipo supervisado ya que se requiere de una salida esperada para poder determinar el error y realizar el aprendizaje, esta salida será determinada previamente por un maestro; uno de los principales inconvenientes en este tipo de red es precisamente el aprendizaje, la dificultad en los métodos de entrenamiento hizo decrecer el interés en este tipo de redes neuronales.

El aprendizaje de este modelo de red se realiza usualmente mediante la propagación hacia atrás (backpropagation), aunque también se pueden usar métodos como el aprendizaje constructivo o los algoritmos genéticos.

El método de propagación hacia atrás consiste en presentar a la red un patrón de entrada con el cual es calculada la salida de la red, una vez que se obtiene la salida de la red esta se compara con la salida deseada para dicho patrón y se obtiene el valor de error, luego se transmite los errores hacia atrás, partiendo de la capa de salida hacia las neuronas de la capa intermedia y luego de capa en capa hasta que todas las neuronas reciben la parte

relativa del error que aportan al error total, con este valor de error se alteran los pesos de las neuronas de acuerdo al error presente en cada una, de forma que la siguiente vez que se presente el mismo patrón de entrada se tenga un error menor. La facilidad en el proceso de aprendizaje, radica en la capacidad de la red de propagación de errores de adaptar los pesos de las capas intermedias aprendiendo la relación entre los patrones de entrada y las salidas de la red neuronal, de forma que al presentarse nuevos patrones, luego de la etapa de aprendizaje, esta pueda presentar una salida correcta.

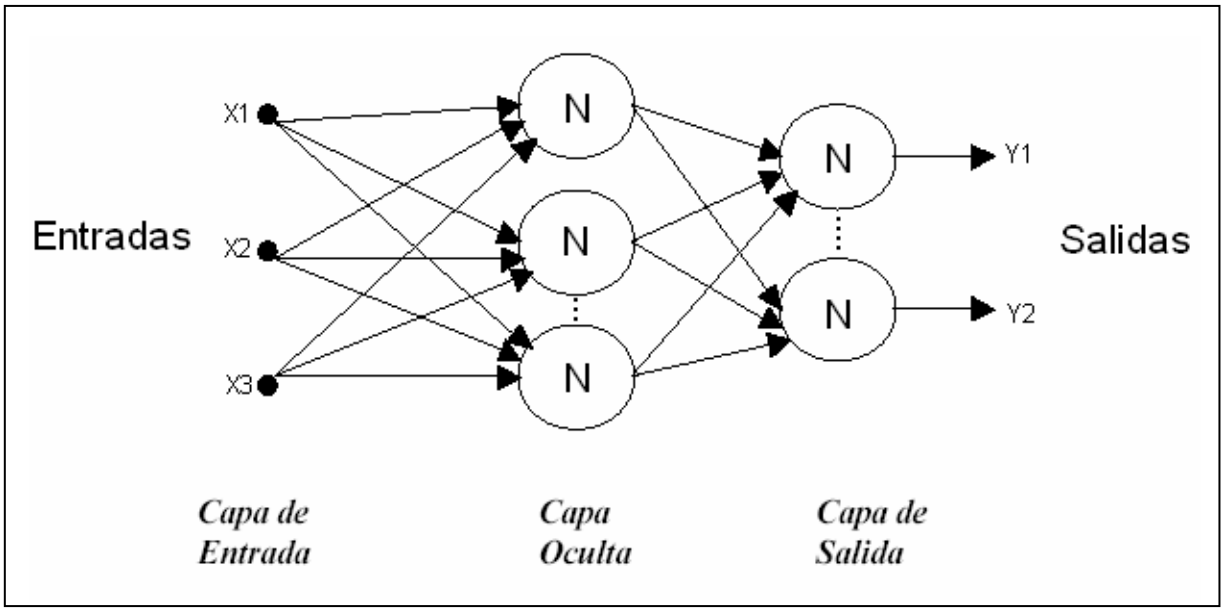


Figura. 2.8. Red Multicapa

El método del aprendizaje constructivo comienza con una red sin unidades ocultas, esta red es entrenada por un tiempo y luego son agregadas unidades ocultas sin variar los pesos de las neuronas ya existentes y se continúa el entrenamiento continuando el proceso hasta tener todas las neuronas necesarias en la red. Los algoritmos constructivos son efectivos para escapar de los mínimos locales (valores en los que la red neuronal parece tener una respuesta óptima). Este tipo de algoritmos de entrenamiento son usados para funciones de activación de las neuronas que no son continuas como lo es la función escalón.

El método de los algoritmos genéticos es usado en problemas difíciles en los que no se conocen muchas de sus características. La idea es codificar una red neuronal por cromosomas, tanto su topología como sus pesos de conexión, y luego aplicar operadores

genéticos como apareamiento y mutación, después los cromosomas son decodificados y evaluados en la realización de una tarea específica definiendo los cromosomas de mejor adaptación. Este método permite una búsqueda paralela, puede definir al mismo tiempo topología y pesos de la red y no necesita de funciones continuas. Ver figura 2.9.

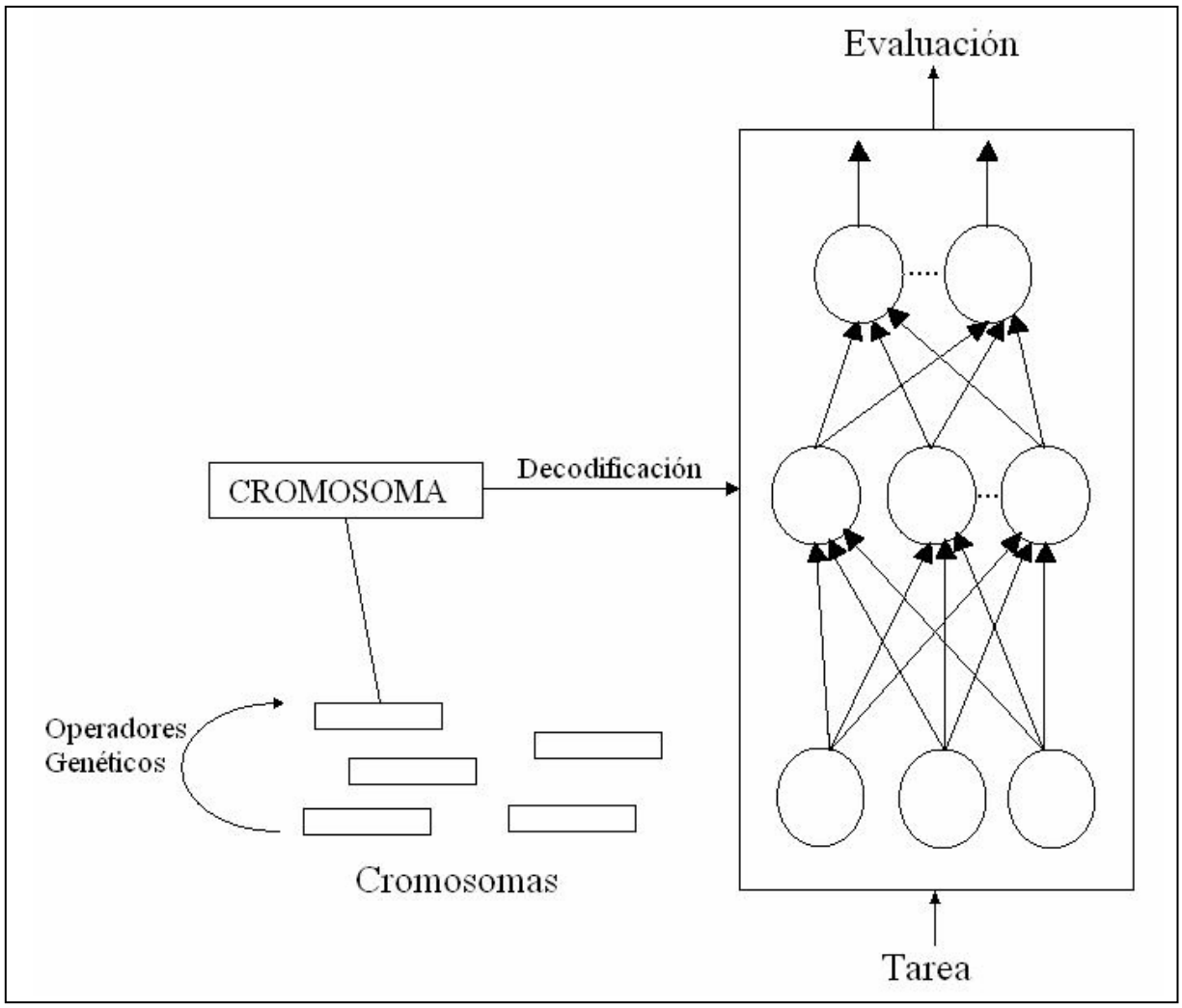


Figura. 2.9. Método Algoritmos Genéticos

La capacidad de la red neuronal de presentar salidas correctas ante entradas desconocidas se le llama generalización, aunque para obtener una respuesta correcta las entradas deben tener características que le permitan a la red relacionarla con los casos en los que fue entrenada.

2.4.4 Modelo Hopfield

El modelo de Hopfield es un modelo de red neuronal presentado por John Hopfield del Instituto de Tecnología de California en los años 80. Son redes que aprenden a reconstruir los patrones de entrada que memorizan durante su entrenamiento (memorias auto asociativas). Poseen funciones de activación boleanas de umbral, esto quiere decir que si una unidad recibe una estimulación total mayor a un umbral establecido la salida será 1, en caso contrario será un 0. La principal característica de esta red es la capacidad de almacenar información y luego recuperarla aunque esta se encuentre incompleta.

Este modelo consiste en un número j de elementos de procesamiento interconectados que cambian sus valores de activación independientemente, la salida de cada uno de los elementos de la red esta conectada a una de las entradas de todos los otros elementos; todos los elementos son a su vez entradas y salidas (red de una sola capa), ver figura 2.10.

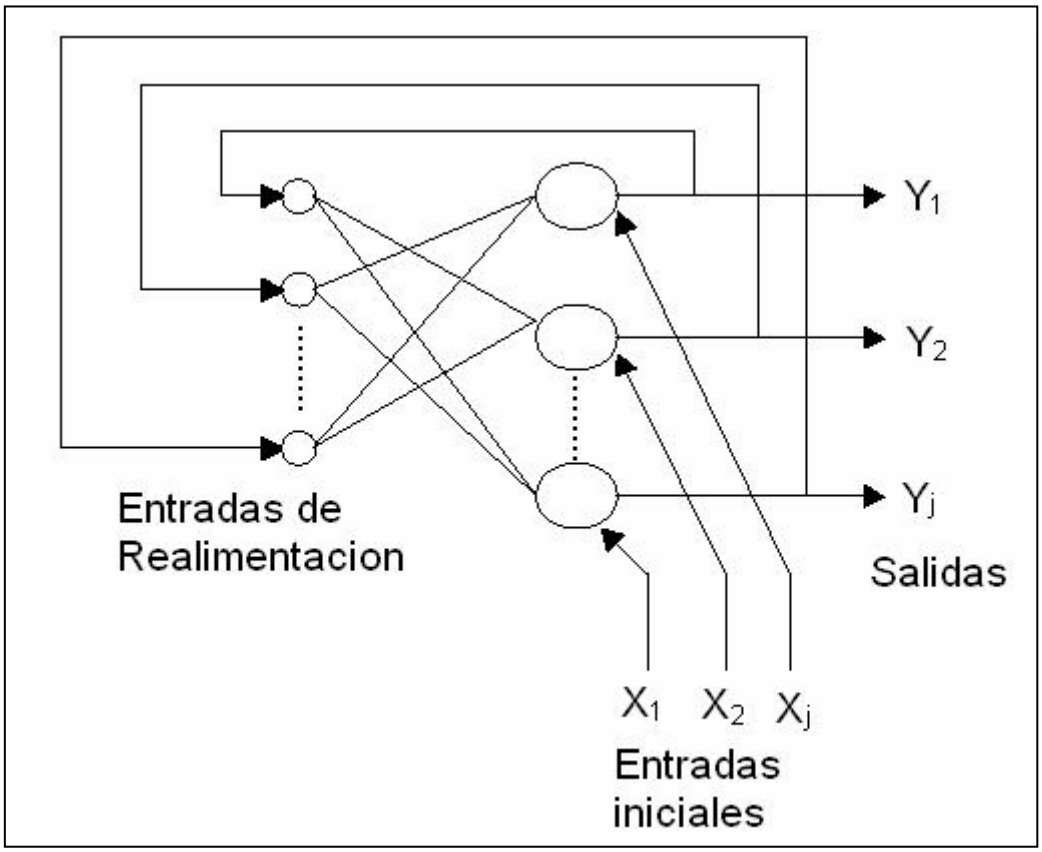


Figura. 2.10. Modelo Hopfield

El funcionamiento de esta red empieza con la aplicación de un patrón de entrada el cual produce un estado inicial que a su vez es entregado a la red, a través de las conexiones entre sus elementos, lo que produce un nuevo estado que también será alimentado, produciéndose de esta forma una sucesión hasta llegar a una estabilidad en la que los elementos de la red ya no sufren cambios en su activación, finalmente la salida de la red será la combinación de las salidas de todos sus elementos en el estado estable.

Al presentar una nueva entrada a la red esta tomará una configuración inicial parecida a alguno de los estados de un estímulo almacenado, luego comenzará a cambiar hasta llegar al estado que representa a la señal almacenada en la etapa de aprendizaje. Si el nuevo estímulo es muy diferente a cualquier de los que ya están almacenados se alcanzará un punto que no representa ningún recuerdo. Otro caso que podría darse es que la red comience a presentar estados que se repiten periódicamente y no llegue a estabilizarse nunca.

2.5 APLICACIONES DE LAS REDES NEURONALES ARTIFICIALES

Las aplicaciones para las Redes Neuronales se dan en una gran cantidad de áreas. Ante tantas aplicaciones a continuación se citarán solo algunas de estas áreas y algunas aplicaciones en cada una de ellas.

Electrónica y Telecomunicaciones. Las aplicaciones en la electrónica incluyen el control de procesos, reconocimiento de voz, visión artificial, análisis de fallas, modelos no lineales, sistemas de piloto automático, control dinámico de trayectoria en robots, sistemas ópticos, compresión de datos e imágenes, translación en tiempo real de lenguaje hablado y otros.

Bancos y Finanzas. Asesoria de préstamos, evaluación de créditos, evolución de precios, reconocimiento de firmas, identificación de falsificaciones, criptografía, códigos de seguridad adaptativos, seguimiento de hipotecas y otros.

Manufactura. Control de procesos y producción, evaluación de fallas en máquinas y procesos, sistemas de control de calidad visuales, modelamiento de procesos, procesos de mantenimiento, sistemas de seguimiento de materiales y productos terminados entre otros.

Medicina. Análisis de enfermedades, análisis de electroencefalograma, diseño de prótesis, reconocimiento de células portadoras de cáncer, diagnóstico y tratamiento a partir de síntomas y otros, por lo tanto reducción de gastos.

CAPÍTULO 3

ARQUITECTURA DE LA RED NEURONAL ARTIFICIAL

3.1 DEFINICIÓN DEL PROBLEMA

Diseñar una red neuronal artificial que pueda controlar el movimiento de un Robot (plataforma móvil) dentro de un lugar con condiciones variables de luz, temperatura, distribución de objetos, color y forma de las paredes, de manera que al final logre moverse sin chocar con los objetos o paredes.

3.1.1 Condiciones del robot

3.1.1.1 Sensores

El robot tendrá tres sensores infrarrojos que miden la proximidad de un objeto y entregan una señal análoga asociada a la distancia medida, estos sensores trabajan en el rango de 4 a 30 cm. con una señal de 3 a 0.024 Voltios respectivamente.

3.1.1.2 Motores

El robot cuenta con dos motores para su movimiento, cada uno de los cuales esta unido a una rueda a un lado de la plataforma, con el fin de lograr el giro del robot sobre su eje con solo detener uno de los dos motores cambiando la trayectoria.

3.1.2 Condiciones del ambiente

El ambiente en el que se desenvolverá la plataforma será uno con características comunes a los lugares que son usualmente habitados por seres humanos con lo que se pueden esperar variaciones en las siguientes condiciones de trabajo.

3.1.2.1 Condiciones de luz

La luz en el cuarto en el cual se desplazará el robot puede variar desde no tener ninguna luz hasta estar plenamente iluminada, lo importante será que los sensores no reciban luz directamente ya que esto puede cambiar el valor entregado por el sensor y confundir a la red neuronal.

3.1.2.2 Condiciones de las paredes y objetos

Las paredes y objetos deben tener un tamaño mínimo de 4 cm. en cada una de sus dimensiones y tener una superficie que refleje la longitud de onda de los sensores, 850 nm ± 70 nm, para que puedan ser detectados por la red neuronal.

3.1.2.3 Condiciones del suelo

El suelo del cuarto debe ser plano, sin inclinación, no debe tener obstáculos o irregularidades grandes, no mayores a 5 mm, que puedan afectar el desplazamiento de la plataforma.

Probablemente existen más condiciones que afectan el comportamiento del robot que podrían ser utilizadas en el diseño del mismo, pero siendo el objetivo de este trabajo el diseño y prueba de la red neuronal no se extenderá más en el estudio de estas condiciones.

3.2 DISEÑO DE TOPOLOGÍA DE LA RED NEURONAL

Para el diseño de la red neuronal artificial se usan algunos conceptos de otras redes neuronales, sin embargo no se usa ninguno de los modelos en su totalidad ya que presentan inconvenientes para resolver el problema. Es por esto que se plantea un nuevo modelo de red neuronal y una nueva teoría para el aprendizaje de la red.

Al ser un problema bastante simple, que solo requiere como respuestas de la red neuronal la activación o desactivación de los motores, solo se utilizarán dos neuronas cada una de las cuales se encargará del manejo de uno de los motores. Como el fin es que la plataforma no choque con ningún objeto las neuronas se encargarán de desactivar los motores en el caso que las condiciones de movimiento puedan llevar a una colisión, es decir que las neuronas se activarán para detener a los motores y evitar de esta forma los objetos. Solo se tendrán tres entradas normales en cada neurona ya que este es el número de sensores de proximidad con el que contará el robot.

Para conocer las condiciones en las cuales se producen los choques, la red neuronal tendrá una señal adicional que será llamada señal de error, ésta se encargará de indicar a la red cuando se producen equivocaciones en el comportamiento. En el caso de querer compararla con un modelo biológico estaríamos hablando de una señal parecida al dolor, el cual causa un cambio en la red y por ende en el comportamiento. Esta señal será usada en la etapa de aprendizaje para ayudar en el ajuste de los pesos de las conexiones de entrada en las neuronas.

Además se tiene otra señal que permite a las neuronas tener una influencia sobre las demás, lo cual ayudará en el aprendizaje de la red y en su comportamiento. En una primera etapa del aprendizaje las neuronas tendrán pesos muy parecidos o iguales para todas sus conexiones lo que hace que estas tengan el mismo comportamiento ante las mismas señales de entrada, esto llevaría a tener dos neuronas que actúan de igual manera con lo cual no lograríamos ningún avance. Sin embargo al tener una forma de influir una neurona sobre la otra y al llegar a una condición de activación, la neurona que primero se active, que será llamada en adelante neurona ganadora, enviará una señal que influirá sobre la otra neurona cambiando las condiciones para esta última con lo que no tendrán la misma

activación y aprendizaje, luego de un tiempo los pesos de sus conexiones se diferenciarán lo suficiente para que cada neurona tenga un comportamiento distinto. Esta señal será llamada señal de influencia. Además esta señal de influencia podría servir para cambiar el comportamiento y aprendizaje a través de otras neuronas u otro tipo de sistema controlador conectado a esta entrada de la neurona. En este trabajo la señal de influencia se presenta en una entrada con un peso fijo negativo, sin embargo se podría lograr que este peso fuera cambiando a medida que transcurre el aprendizaje de la neurona.

Cumpliendo con las ideas expuestas la topología de la red neuronal podría ser la que se observa en la figura 3.1. En este grafico la señal "E" de color azul es la señal de error y la señal "I" de color rojo es la señal de influencia entre neuronas.

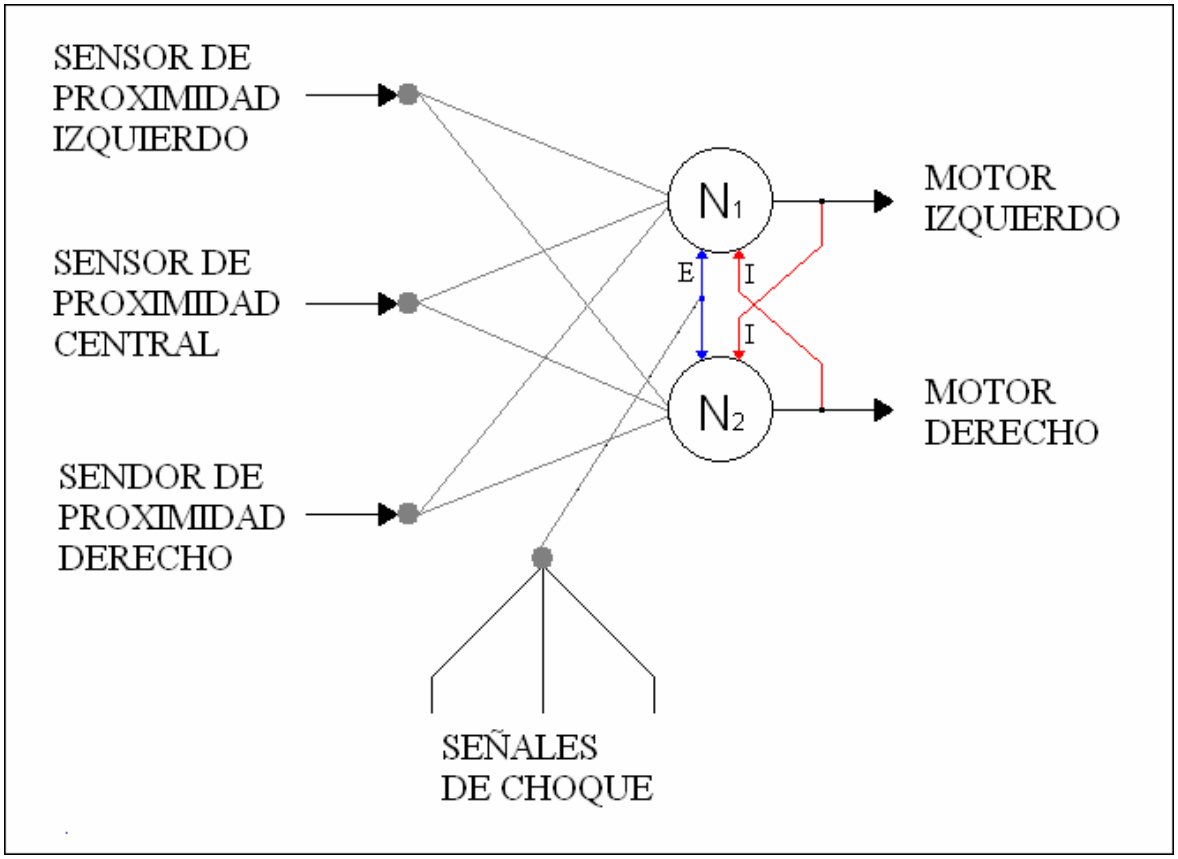


Figura. 3.1 Red Neuronal Propuesta 1

Sin embargo la topología se podría cambiar un poco para tener una mayor facilidad en las conexiones y evitar la necesidad de tener una entrada adicional por cada neurona en una red con este tipo de arquitectura, ya que cada neurona necesitaría una conexión de entrada

por cada una de las otras neuronas que estén trabajando con ella. Por lo que se propone una arquitectura diferente que puede verse en la figura 3.2.

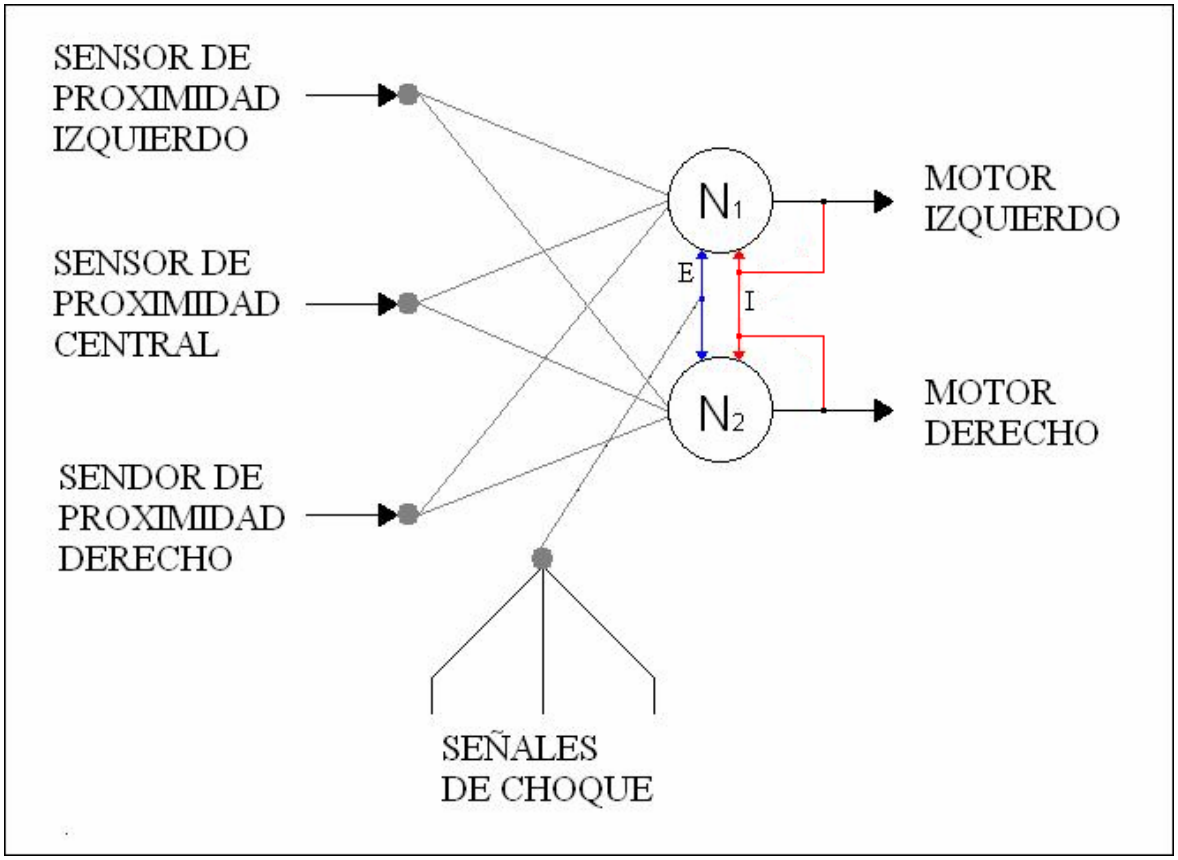


Figura. 3.2 Red Neuronal Propuesta 2

En la nueva arquitectura se tiene una señal que puede ser usada por cualquier neurona para influir sobre el resto. A esta señal se podrían conectar las salidas de todas las neuronas y aunque una neurona estaría realimentándose, dando un pequeño período entre una activación y otra, ella nunca llegaría a influir en su propia activación o aprendizaje ya que su propia influencia desaparecería antes. Lo que se debería tener en cuenta si se utiliza esta arquitectura es garantizar que las señales solo puedan ir desde la salida de las neuronas hasta las entradas de las otras neuronas para que la salida de una neurona no interfiera en la señal de salida de otra directamente. Es decir que la señal no debe ir desde una salida hasta la salida de otra neurona.

Finalmente en este proyecto se utilizará la red neuronal de la figura 3.1 “Red Neuronal Propuesta 1” por simplicidad en el diseño y conexión.

3.3 FORMA DE APRENDIZAJE

Para el aprendizaje se debe considerar las señales de error, la influencia entre las neuronas y las activaciones correctas de una neurona, es decir, sin que se produzcan señales de error durante su activación. La finalidad del aprendizaje será llevar a la red a alcanzar unos pesos para cada una de sus conexiones que permitan a la plataforma moverse cerca de las paredes y objetos sin llegar a chocar.

3.3.1 Memorización de pesos

Al ser un modelo de red neuronal artificial diferente a los encontrados y debido a que este modelo no puede tener una etapa de aprendizaje bien definida, sino que aprende constantemente, se desarrolló un nuevo concepto que permite a la red mantener constante los pesos que dan un buen resultado en su funcionamiento y variables aquellos pesos que no influyen de forma apreciable en su comportamiento, para juzgar si se tiene un buen resultado se usan las señales de error.

Esta memorización será un tiempo durante el cual la neurona mantiene estable el peso de una conexión, este tiempo será diferente para cada una de las conexiones y estará influenciado por la señal de entrada, de forma que una señal de entrada con un valor muy grande provocará una memorización a largo plazo. El valor de la memorización se establecerá cada vez que se produzca una activación de la neurona y también se verá influido por su valor previo, esto para lograr que entre más activaciones sufra la neurona, mayor sea el tiempo que recuerda el peso de la conexión. A medida que transcurra el tiempo sin que se produzcan nuevas activaciones de la neurona, este valor irá disminuyendo hasta llegar a cero, en este momento el peso será libre para variar de nuevo.

$$Memorización = Memorización + Señal \quad Entrada$$

También se propone que llegado un punto muy alto de memorización esta se vuelva permanente.

3.3.2 Cálculo de pesos

Para el calculo de los pesos se debe tener en cuenta que estos varian en forma diferente para distintas etapas del aprendizaje, primero se debe considerar una etapa en la que no se tiene ninguna memorización.

En esta etapa sin memorización los pesos puedan cambiar de acuerdo a la señal que se recibe en cada entrada de forma que se pueda tener un aprendizaje constante. Estos pesos aumentarán o disminuirán su valor como un porcentaje de la diferencia entre el valor del peso y la señal de entrada como se muestra en la siguiente ecuación.

Sin memorización.

$$Peso \quad Siguiente = Peso \quad Anterior + (Señal \quad Entrada - Peso \quad Anterior) * Porcentaje$$

En el presente trabajo se usará un porcentaje del 25% quedando la ecuación de la siguiente forma.

Sin memorización.

$$Peso \quad Siguiente = Peso \quad Anterior + (Señal \quad Entrada - Peso \quad Anterior) / 4$$

Este porcentaje se usa para que los pesos no cambien exactamente igual a la señal de entrada y mantenga una pequeña influencia de sus valores anteriores. Esto evitará que la neurona se vea afectada por señales repentinas que cambien demasiado.

Otra etapa en la que los pesos varían su valor esta dada cuando la neurona se activa, en esta etapa ya se tiene una memorización y la neurona cambia sus pesos en forma diferente

a las ecuaciones descritas anteriormente. Esta etapa se divide en dos posibles cambios para los pesos. La primera posibilidad es que y no se produzca señal de error, en este caso tenemos que los pesos varían según la siguiente ecuación.

Con memorización y sin error.

$$\text{Si,} \quad PESO < SEÑAL\_DE\_ENTRADA$$

$$\text{Entonces,} \quad PESO = PESO + 1$$

Con esto los pesos se incrementarán hasta lograr un valor similar a aquellas señales de entrada que activan las neuronas, no se incluye un decremento de los pesos ya que al poderse activar la neurona con diferentes señales podríamos eliminar lo aprendido en otra situación y en otro peso.

La segunda posibilidad, en caso de tener una activación, es que se produzca una señal de error, con lo cual los pesos deberían variar según la siguiente ecuación.

Con memorización y error.

$$Peso = Peso - Decremento$$

El valor del decremento será explicado un poco mas adelante.

3.3.3 Influencia entre neuronas

La influencia entre neuronas se da como una forma de lograr que las neuronas no reaccionen ante las mismas señales de entrada. A través de esta señal de influencia la neurona ganadora cambiará las condiciones en las otras neuronas, en este trabajo será una conexión inhibitoria que dificultará la activación de la otra neurona. Esta conexión inhibitoria será un valor fijo. También se agrega un pequeño retardo que dará un pequeño tiempo para que la neurona ganadora intente resolver la situación por si sola, si no lo logra se podrá activar la segunda neurona si las condiciones no han cambiado.

3.3.4 Señales de error

Las señales de error permiten a la red conocer cuales acciones u omisiones son erradas y cambiar así su comportamiento de forma que la siguiente vez que se presente una situación el comportamiento no sea el mismo. En este proyecto la señal estará dada por sensores en la parte frontal de la plataforma que determinarán si la plataforma se ha chocado.

La señal de error se verifica siempre que una neurona ha estado activada, esto para determinar si su activación fue la que produjo el error en el comportamiento. Si la neurona no ha estado activa y se produce la señal de error esta no sufrirá refuerzo negativo, aunque se produzca su activación mientras continua activa la señal de error.

En caso de tener una señal de error.

$$Peso = Peso - Decremento$$

Donde, Decremento es el valor en el que decrecen los pesos cada vez que se produce una señal de error

En el presente proyecto el valor en que decrecen los pesos será igual a 8, variando este valor se logra que la neurona aprende un poco mas rápido o mas lento, aunque se debe tener cuidado ya que un valor muy grande puede producir, en caso de una señal de error equivocada, un mal entrenamiento que será muy difícil de cambiar; en lugar de esto un valor pequeño puede ser cambiado con refuerzos positivos fácilmente pero se demorará mas en entrenar la red. Este valor estará detallado en el diseño del software que simulará el comportamiento de cada neurona.

3.3.5 Activación de la neurona

Las neuronas se activarán al momento que el nivel de activación supere al umbral de activación, una vez que esto sucede la neurona permanecerá activa y recibirá un refuerzo positivo como recompensa a su activación.

La ecuación que permite conocer el nivel de activación es la siguiente.

$$Nivel = P_0 * S_0 + P_1 * S_1 + P_2 * S_2 - I$$

Donde:

P_0 es el peso en la entrada 0

S_0 es la señal en la entrada 0

P_1 es el peso en la entrada 1

S_1 es la señal en la entrada 1

P_2 es el peso en la entrada 2

S_2 es la señal en la entrada 2

I valor constante que se presenta cuando existe Influencia.

CAPÍTULO 4

IMPLEMENTACION DE LA RED NEURONAL ARTIFICIAL

4.1 DISEÑO DEL HARDWARE

Para la implementación de la red neuronal artificial se utilizarán microcontroladores que simularán el comportamiento de una neurona con todas las condiciones vistas en el capitulo anterior.

Se usarán dos microcontroladores cada uno de los cuales estará conectado a los sensores de proximidad con entradas análogas, además estarán conectados a la entrada de error y a la entrada de influencia mediante entradas digitales y tendrán una sola salida conectada a un integrado operador de los motores. Cada uno de estos microcontroladores realizará todas las operaciones correspondientes a la neurona y guardará el valor de los pesos y memorizaciones para cada una de las conexiones de entrada así como el valor del umbral de activación.

El diagrama de conexión de los elementos se puede observar en la figura 4.1.

4.1.1 Microcontrolador

El microcontrolador que se usará en la implementación de la red neuronal es el PIC12F675 el cual tiene las siguientes características básicas:

- 6 entradas/salidas 4 de las cuales pueden funcionar como entradas análogas.
- Reloj interno de 4 Mhz.
- Memoria de 1024 bytes de programa, 64 bytes de SRAM y 128 bytes de EEPROM

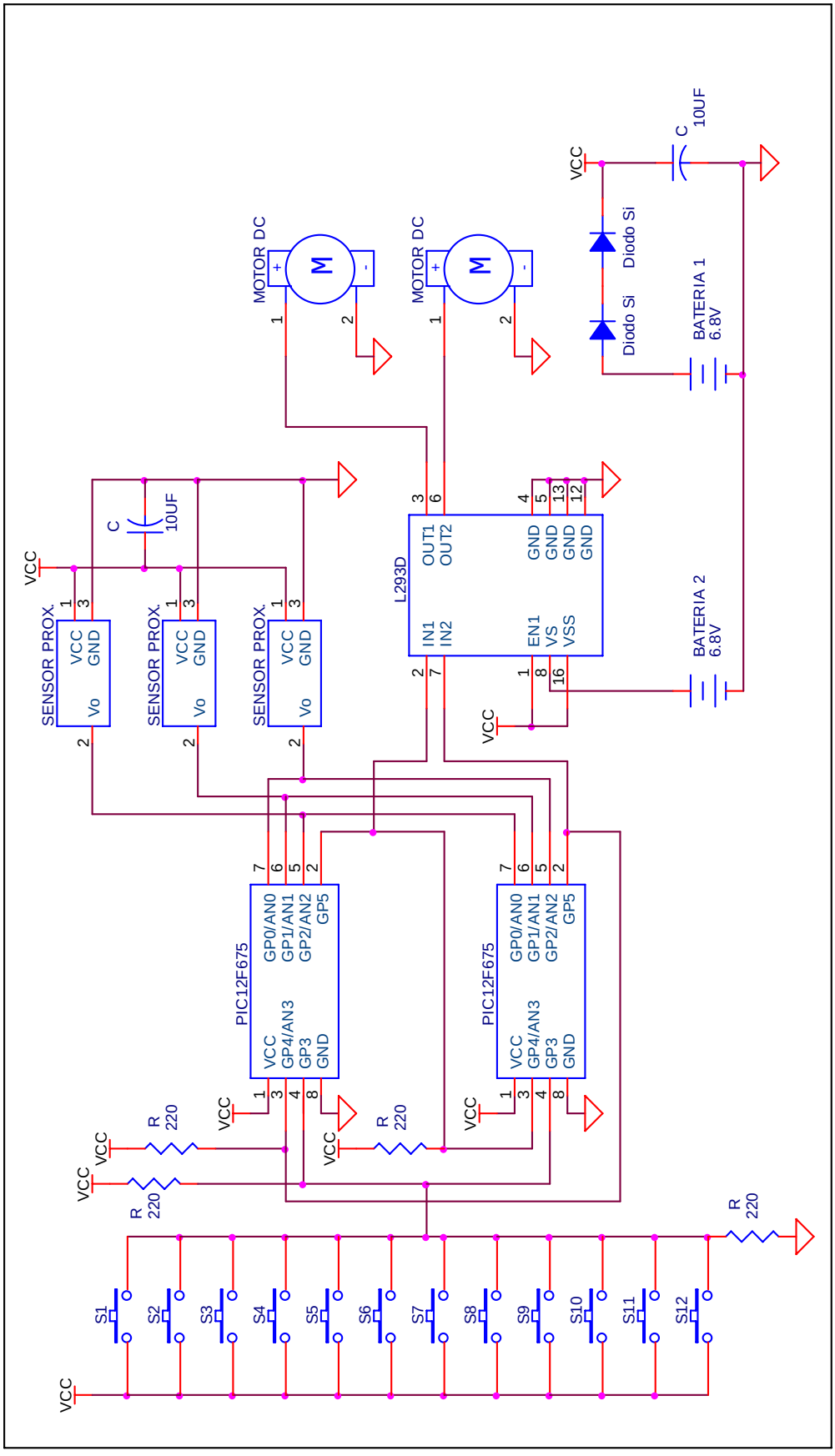


Figura. 4.1 Diagrama de conexión de elementos

Estas características básicas lo hacen un microcontrolador capaz de cumplir con este trabajo y al mismo tiempo es pequeño, fácil de conectar y usar. Para ver las otras características de este microcontrolador refiérase a los data sheets.

4.1.2 Manejador de motor

El manejador de los motores (Driver de motor) es el L293D, el cual posee las siguientes características básicas:

- Voltaje de salida 4.5 a 36 V.
- Salida de 600mA por canal.
- Salida pico de 1.2 A por canal
- Diodos Clamping
- Fuente separada para las entradas lógicas.

Otras características en el data sheet.

4.1.3 Fuente de voltaje

Para la fuente de voltaje se usarán dos baterías recargables Panasonic LC-R065P de 6.8 voltios y 5Ah, una de las cuales estará encargada de alimentar los motores, mientras la otra alimentará los microcontroladores y sensores de proximidad, esta segunda batería es conectada a través de dos diodos que harán caer el voltaje hasta aproximadamente 5 Voltios.

4.1.4 Sensores de proximidad

Los sensores con los que se detectará la proximidad de objetos son los sensores Sharp GP2D120 cuyas características principales son :

- Baja influencia del color de los objetos.
- Rango de detección 4 a 30 cm.

- Voltaje de entrada 4.5 a 5.5 V.
- Voltaje de salida 0.25 a 2.9 V.
- Señal no lineal.

Otras características en el data sheet.

4.1.5 Sensores de Choque

Para los sensores de choque se usarán pulsadores conectados en paralelo, distribuidos en la parte frontal de la plataforma de forma que si se produce un choque uno de ellos sea pulsado y envié una señal de 5 Voltios a la entrada correspondiente de los microcontroladores.

4.1.6 Otros elementos

Además de estos elementos se usaran resistencias para conectar los pulsadores, diodos de silicio, con el fin de tener una pequeña caída de voltaje en la fuente que permita obtener aproximadamente 5 a 5.5V, y capacitores necesarios para estabilizar el voltaje de alimentación en los integrados y sensores.

4.2 DISEÑO DEL SOFTWARE

El software necesario para el funcionamiento de los microcontroladores se realizó en código de máquina para optimizar su funcionamiento y lograr que se ejecute rápidamente.

4.2.1 Inicialización del microcontrolador

El microcontrolador es inicializado con tres entradas análogas, dos entradas digitales y una salida digital, además es programado para que use un reloj interno (cristal) en su funcionamiento, son deshabilitadas todas las interrupciones y el comparador.

En la inicialización del microcontrolador se reservan registros de memoria para realizar las operaciones y registros para guardar datos de la siguiente manera.

4.2.1.1 Registros de operaciones

Estos registros son necesarios para realizar todas las operaciones que permitan simular el comportamiento de la neurona, más no representan el aprendizaje de la misma.

AUX, AUX+1, AUX+2. Usados en los retardos y en operaciones donde no se quiere perder el valor inicial de un registro.

CANAL_0. Este registro contiene el valor digital de la señal de entrada 0. Esta señal varia en valor digital entre 13 y 154 debido a que la señal del sensor varia entre 0.25 y 2.9 Voltios.

CANAL_1. Este registro contiene el valor digital de la señal de entrada 1. Esta señal varia en valor digital entre 13 y 154 debido a que la señal del sensor varia entre 0.25 y 2.9 Voltios.

CANAL_2. Este registro contiene el valor digital de la señal de entrada 2. Esta señal varia en valor digital entre 13 y 154 debido a que la señal del sensor varia entre 0.25 y 2.9 Voltios.

MULT_0, MULT_0+1. Almacenan el valor de la multiplicación realizada entre el peso 0 y la entrada 0.

MULT_1, MULT_1+1. Almacenan el valor de la multiplicación realizada entre el peso 1 y la entrada 1.

MULT_2, MULT_2+1. Almacenan el valor de la multiplicación realizada entre el peso 2 y la entrada 2.

NIVEL, NIVEL+1. Almacenan el valor de la suma de los registros de multiplicación y la resta en caso de haber influencia de otras neuronas. Estos registros de Nivel determinarán si se activa o no la neurona al ser comparados con el umbral establecido.

CONT +1. Este registro sirve para realizar la grabación de datos, en la memoria EEPROM, que permiten analizar el comportamiento de la neurona. Este registro permite que se realice una grabación cada 127 veces que pasa el programa. Esto con el fin de obtener datos de un periodo más extenso de tiempo, de no hacerlo se obtienen valores que cambian muy poco entre si haciendo muy difícil el análisis.

CONT. Este registro permite determinar en que lugar de la memoria EEPROM se grabará un dato.

DIREC. A este registro se le envía la dirección en la que se desea guardar un dato en la memoria EEPROM

DATO. A este registro se le envía el valor que se desea guardar en la memoria EEPROM.

BANDERA. A este registro se le da un valor de uno en su bit menos significativo cuando al momento de activarse una neurona ya existía una señal de error, esto para que la neurona no modifique su comportamiento por refuerzo negativo ya que su influencia no fue la que produjo el error.

En algunos casos son necesarios dos o mas registros ya que el valor de las operaciones puede fácilmente sobrepasar el 255 que es el valor máximo que se puede almacenar en uno solo.

4.2.1.2 Registros de Datos

Estos registros representarán el aprendizaje realizado por la neurona y del valor almacenado en ellos dependerá su comportamiento.

PESO_0. Este registro almacena el peso para la entrada 0. Este registro varia entre 0 y 154, ya que este ultimo es el máximo valor que puede entregar el sensor.

PESO_1. Este registro almacena el peso para la entrada 1. Este registro varia entre 0 y 154, ya que este ultimo es el máximo valor que puede entregar el sensor.

PESO_2. Este registro almacena el peso para la entrada 2. Este registro varia entre 0 y 154, ya que este ultimo es el máximo valor que puede entregar el sensor.

MEMO_0, MEMO_0+1, MEMO_0+2. Estos tres registros sirven para guardar el valor de la memorización del peso para la entrada 0.

MEMO_1, MEMO_1+1, MEMO_1+2. Estos tres registros sirven para guardar el valor de la memorización del peso para la entrada 1.

MEMO_2, MEMO_2+1, MEMO_2+2. Estos tres registros sirven para guardar el valor de la memorización del peso para la entrada 2.

UMBRAL. Este registro contiene el valor del nivel al cual la neurona debe activarse. Se inicializa con un valor de 08h (Hexadecimal) esto para lograr una activación de las neuronas cuando se encuentre un objeto a 15 cm. aproximadamente.

El código para la inicialización del microcontrolador puede observarse en el anexo 1.

4.2.2 Programa Principal

El Programa principal se encarga de comprobar si debe activarse la neurona. Además llama a una función “Contador” que, aunque no es necesaria para el funcionamiento de la neurona, ayuda en el análisis del comportamiento de la red neuronal ya que permite mantener un registro de los últimos datos en la memoria EEPROM.

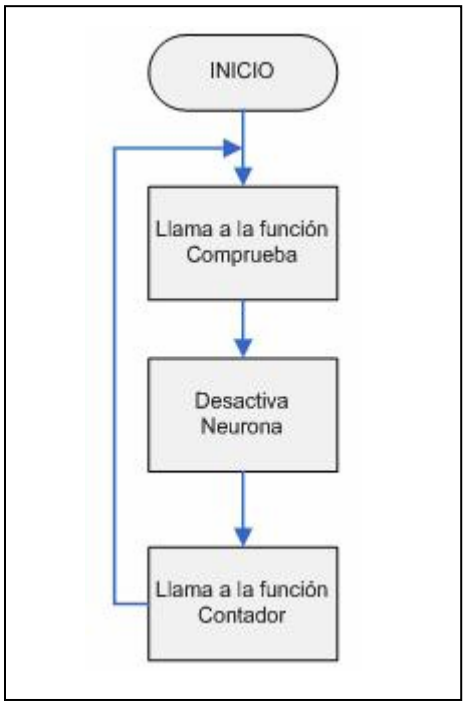


Figura. 4.2 Diagrama de flujo programa principal

4.2.3 Subrutinas

En las subrutinas se encuentran todas las funciones que permitirán realizar la simulación de la neurona. Dentro de estas subrutinas se encuentra funciones para la conversión análoga digital, la actualización de pesos, el aprendizaje positivo y negativo, retardos necesarios para simular el comportamiento real de las neuronas y otras. El código para las subrutinas pueden observarse en el anexo 1.

4.2.3.1 Comprueba

La función “Comprueba” es la encargada de verificar si la neurona debe o no activarse, para ello lo primero que se hace en esta función es verificar si existe una señal de influencia entre las neuronas, de existir se produce la resta y un pequeño retardo que permite a la neurona ganadora activarse sola por un tiempo, tras esto se llama a la función “Conversion” para obtener los valores digitales de las entradas, luego se realizan las multiplicaciones de los valores de entrada con sus pesos respectivos, estos valores son almacenados en los registros “MULT”, a su vez estos valores que se obtuvieron de la

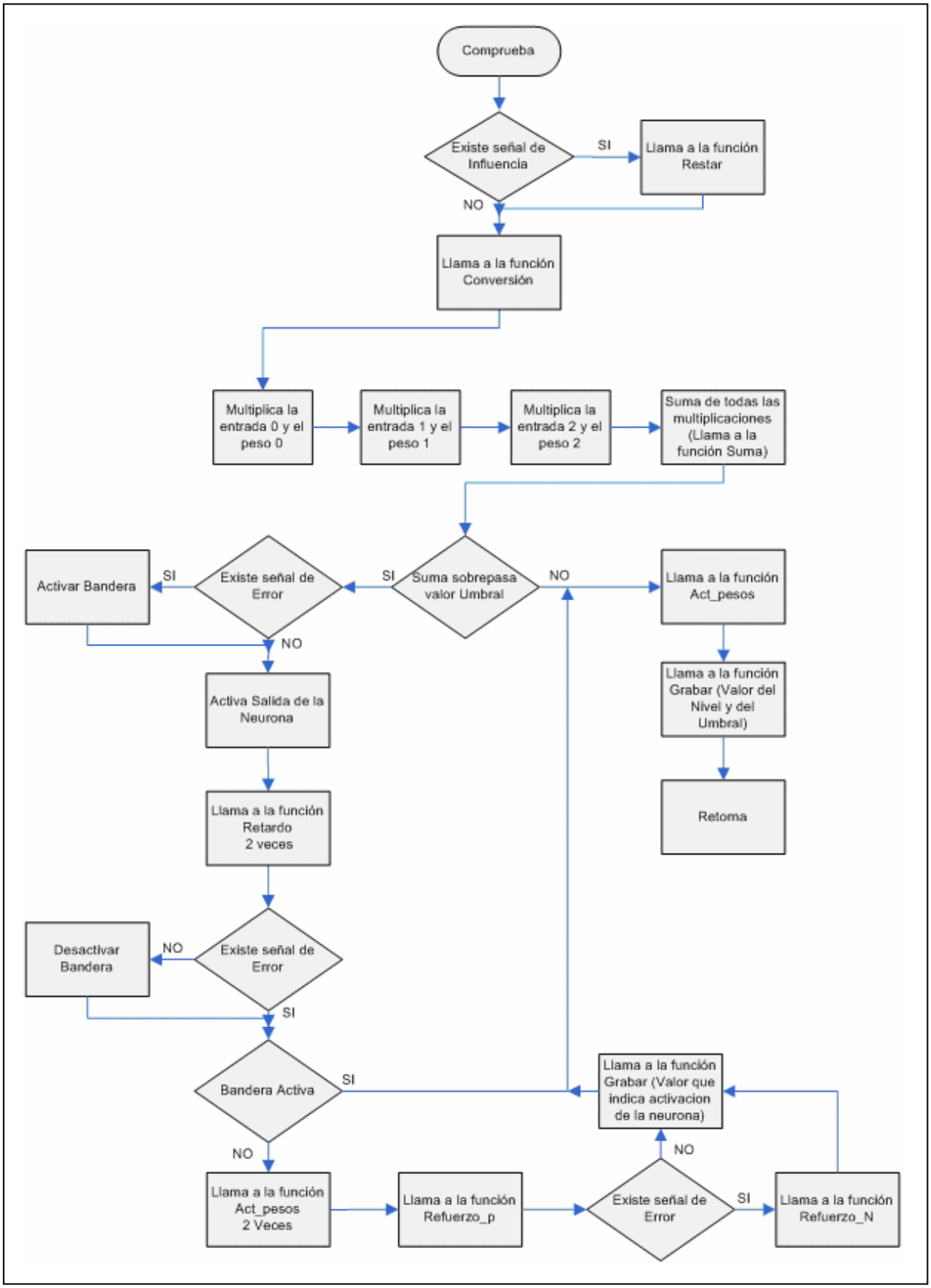


Figura. 4.3 Diagrama de flujo función Comprueba

multiplicación son sumados entre si y su valor, la parte mas significativa de él ya que el byte menos significativo es descartado, es almacenada en los registros "NIVEL". Una vez se obtiene el valor total del nivel se compara con el valor del Umbral de activación y si el primero es mayor se procede a la activación de la neurona.

En el proceso de activación se verifica si existe una señal de error antes de activarse la neurona, si es así y se mantiene esta señal hasta el final de su activación la neurona no debe recibir ningún tipo de refuerzo, para esto se usa el registro "BANDERA", luego se cambia el valor de la salida de la neurona, de un uno lógico a un cero lógico ya que se esta usando lógica inversa para la activación, se da un retardo que permite a la neurona permanecer activada durante un tiempo, luego de este tiempo se comprueba nuevamente si existe señal de error, en caso de existir y si esta señal se encontraba presente desde el comienzo de la activación no se produce ningún refuerzo, si no existe señal de error se realiza un refuerzo positivo llamando a la función "Refuerzo_P" y si la señal de error es nueva, es decir que se produjo durante el tiempo de activación de la neurona se realiza además un refuerzo negativo llamando a la función "Refuerzo_N", finalmente se graba una señal, que indica que la neurona fue activada, en la memoria EEPROM, después pasa al proceso normal de salida de la función.

En caso de no tener un nivel suficiente para la activación se da un proceso de salida en el cual se actualizan los pesos y luego se guardan los datos del nivel y el umbral en la memoria EEPROM, tras lo cual termina la función.

La señal de error se comprueba repetidamente para evitar que una neurona que no influyó en el comportamiento que llevo a la situación en la que se produce esta señal no se vea reforzada negativamente.

4.2.3.2 Actualizar pesos

En esta subrutina se hace el cálculo de los nuevos pesos según las reglas vistas en el capitulo anterior, para realizar la actualización de cada uno de los pesos se debe verificar

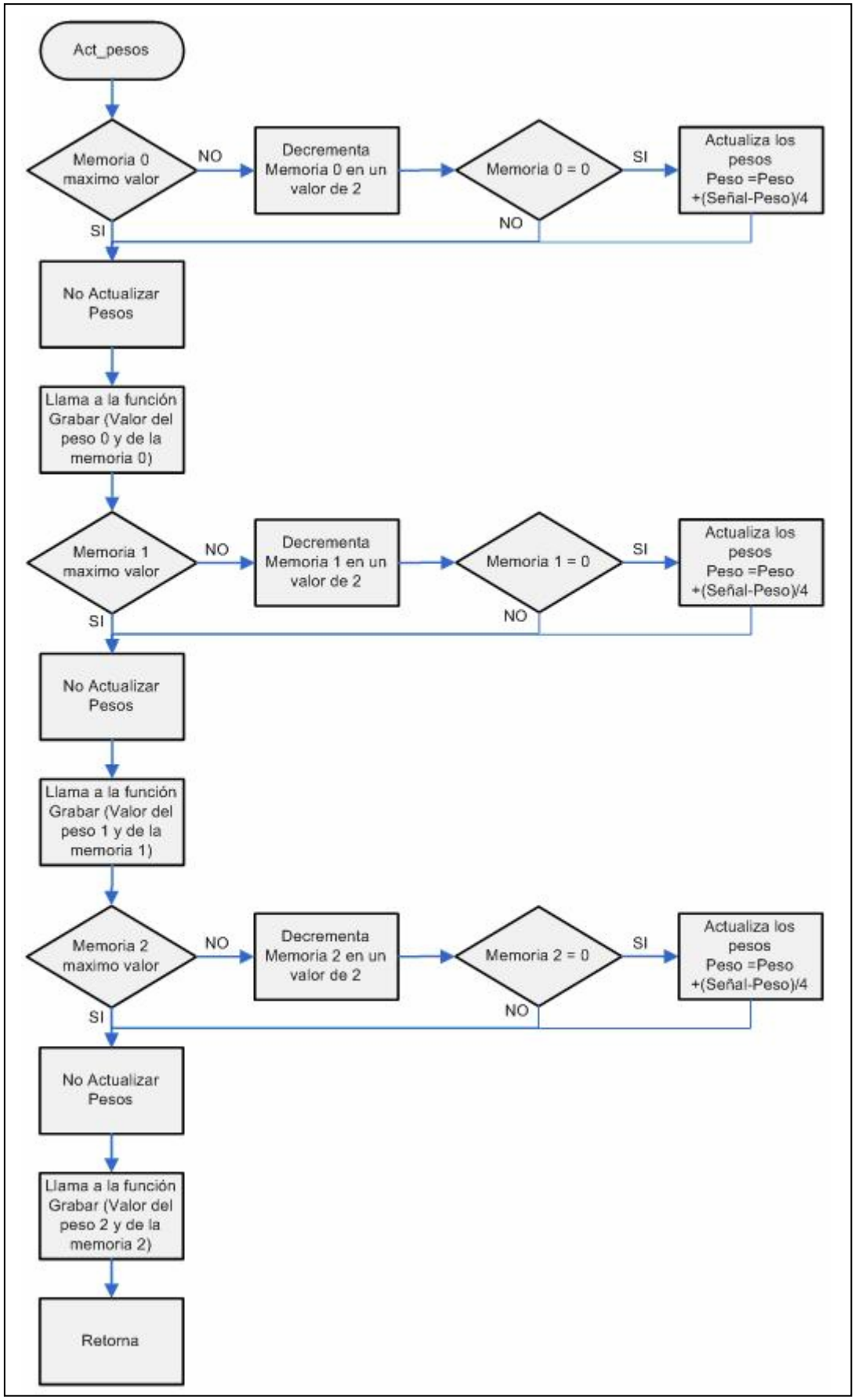


Figura. 4.4 Diagrama de flujo función Actualización de pesos

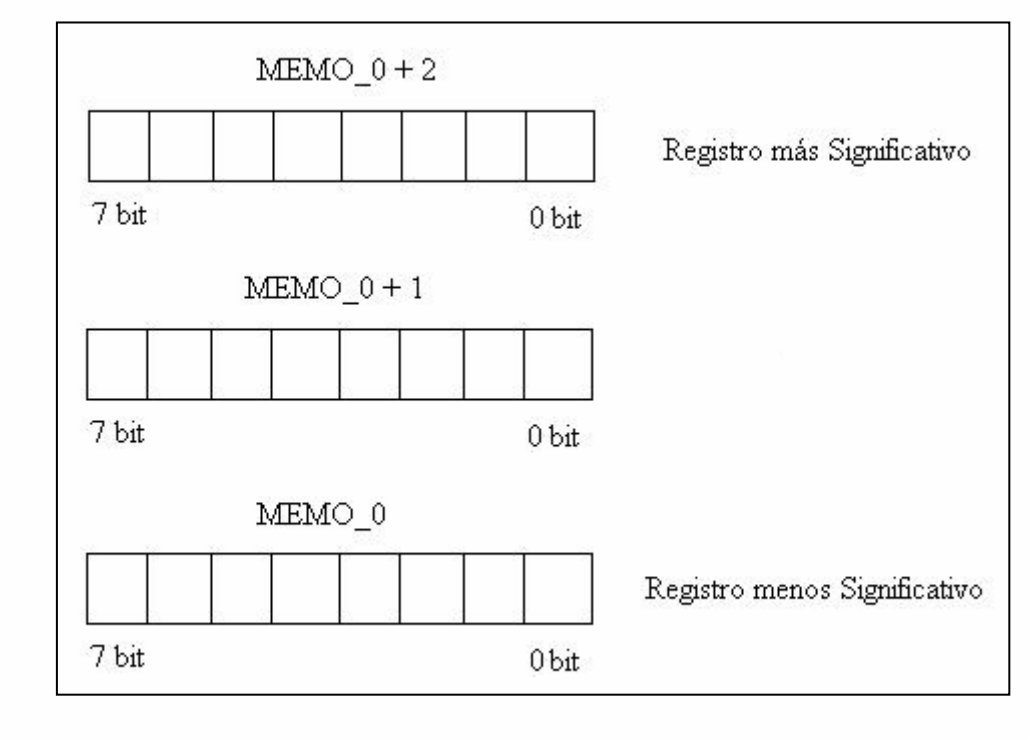


Figura. 4.5 Registros de memoria

primero si no existe un valor de memoria, de existir este valor el peso permanecerá invariable.

Si el valor de la memoria para un peso cualquiera alcanza un valor mayor o igual a 10000000b (binario) en el registro más significativo, dicho peso será memorizado permanentemente. Figura 4.5 Registros de memoria.

En caso de tener un número menor al anteriormente mencionado pero mayor a cero el valor total de la memoria debe ser disminuido en 2 y el peso debe permanecer igual; solo en el caso de tener un valor de memoria igual a cero se debe cambiar el valor del peso correspondiente.

En el proceso de actualización se verifica si la señal de entrada es mayor o menor al valor actual del peso correspondiente. Luego se realiza la suma o resta, dependiendo el caso, de la señal y el peso; este valor es almacenado en un registro que es rotado dos veces

a la derecha, lo que significa que dividimos para 4 su valor descartando los bits decimales, y este valor es sumado o restado del peso actual obteniendo así el nuevo peso.

Finalmente se llama a la función “Grabar” para almacenar los datos de los pesos y los tres registros de memorización, por cada entrada, en la memoria EEPROM.

4.2.3.3 Contador

La función “Contador” se usa con el fin de grabar la información necesaria en la memoria EEPROM con un cierto intervalo de tiempo. Al tener una memoria bastante reducida, solo 128 bytes, no se pueden grabar todos los datos que se generan en la neurona por lo que se hace necesario hacer un muestreo de los datos generados.

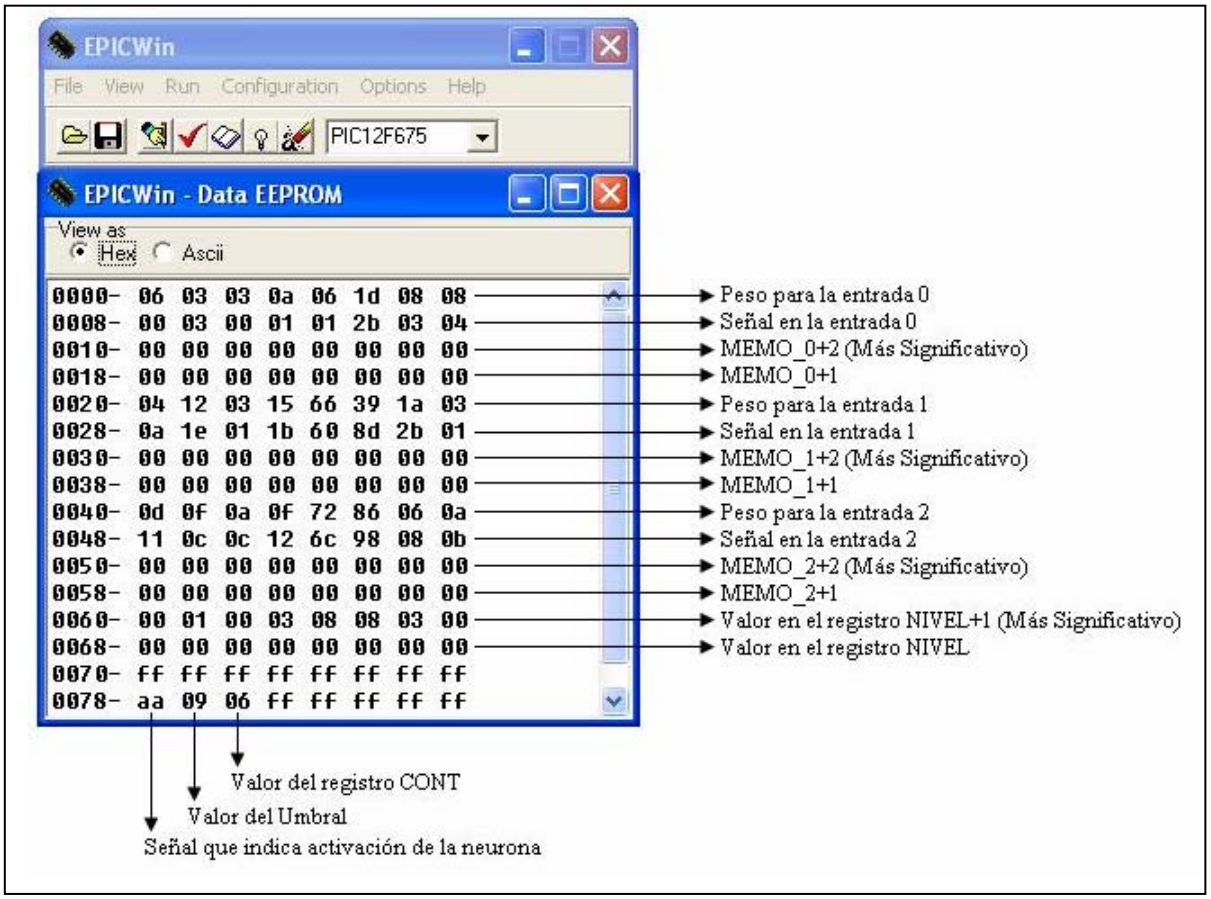


Figura. 4.6 Distribución de memoria

En esta función se tiene un registro (CONT+1) que permite contar cuantas veces se ha ejecutado el programa principal, una vez que este registro alcanza el valor de 128 permite al programa de grabación escribir los valores de las señales, pesos, memorias y otros. Estos valores serán necesarios para el análisis del comportamiento de la neurona, también se incrementa el contador de posición (registro CONT) que nos da la ubicación de los datos en la memoria EEPROM, una vez este contador llega al valor de 8 vuelve al valor de 0 inmediatamente para grabar de nuevo en la primera columna de la memoria. Los datos se almacenan en la memoria de la forma que se muestra en la figura 4.6.

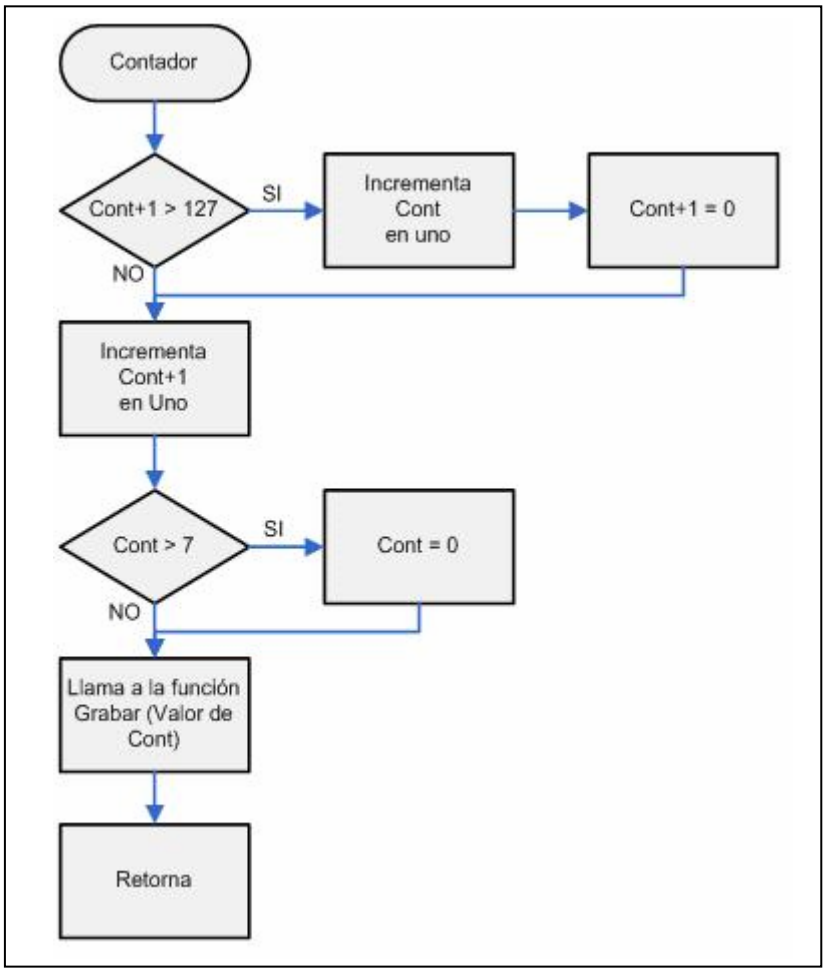


Figura. 4.7 Diagrama de flujo función Contador

El valor del registro “CONT” nos sirve para saber la columna en la que se encuentran grabados los últimos pesos, señales, memorias y niveles de activación. Las otras siete columnas contienen estos mismos datos pero en momentos anteriores. En el ejemplo de la

figura podemos observar que la última columna de datos grabados es la sexta y los datos más antiguos que se poseen se encuentran en la séptima columna.

Los datos que se encuentran grabados en la última fila corresponden a una señal que indica que existió por lo menos una activación de la neurona, el valor del umbral de activación y el valor del registro “CONT” respectivamente. Las direcciones restantes de la memoria no son utilizadas.

4.2.3.4 Suma y Resta

Las funciones “Suma” y “Resta” se encargan de realizar la suma entre el valor almacenado en el registro “AUX” y el valor en el registro “NIVEL”, para la suma en caso de existir un valor mayor a 255 se debe incrementar el registro “NIVEL+1” en uno, y para la resta en el caso de obtener un valor menor a 0 se debe disminuir el registro “NIVEL+1” en uno.

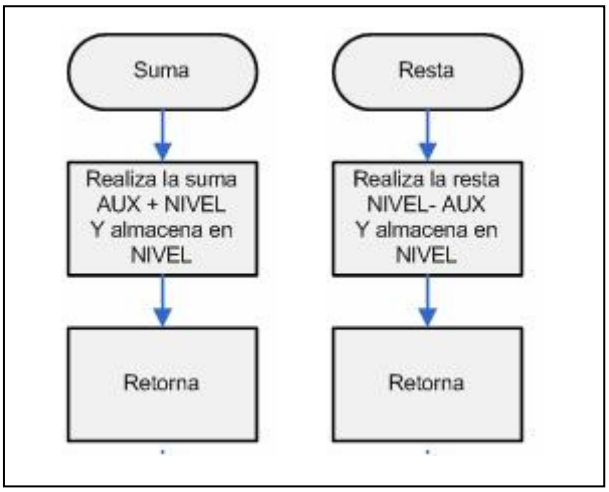


Figura. 4.8 Diagramas de flujo funciones Suma y Resta

4.2.3.5 Conversión Análoga Digital

La función “Conversión” halla los valores digitales en las entradas del microcontrolador y almacena estos valores en las variables “CANAL_0”, ”CANAL_1” y “CANAL_2”.

Dentro de este proceso se usa la función “Convertir” que mantiene al programa en un bucle hasta que se termine la conversión análoga digital para cada canal. Una vez se obtiene los valores para las tres entradas se termina el proceso.

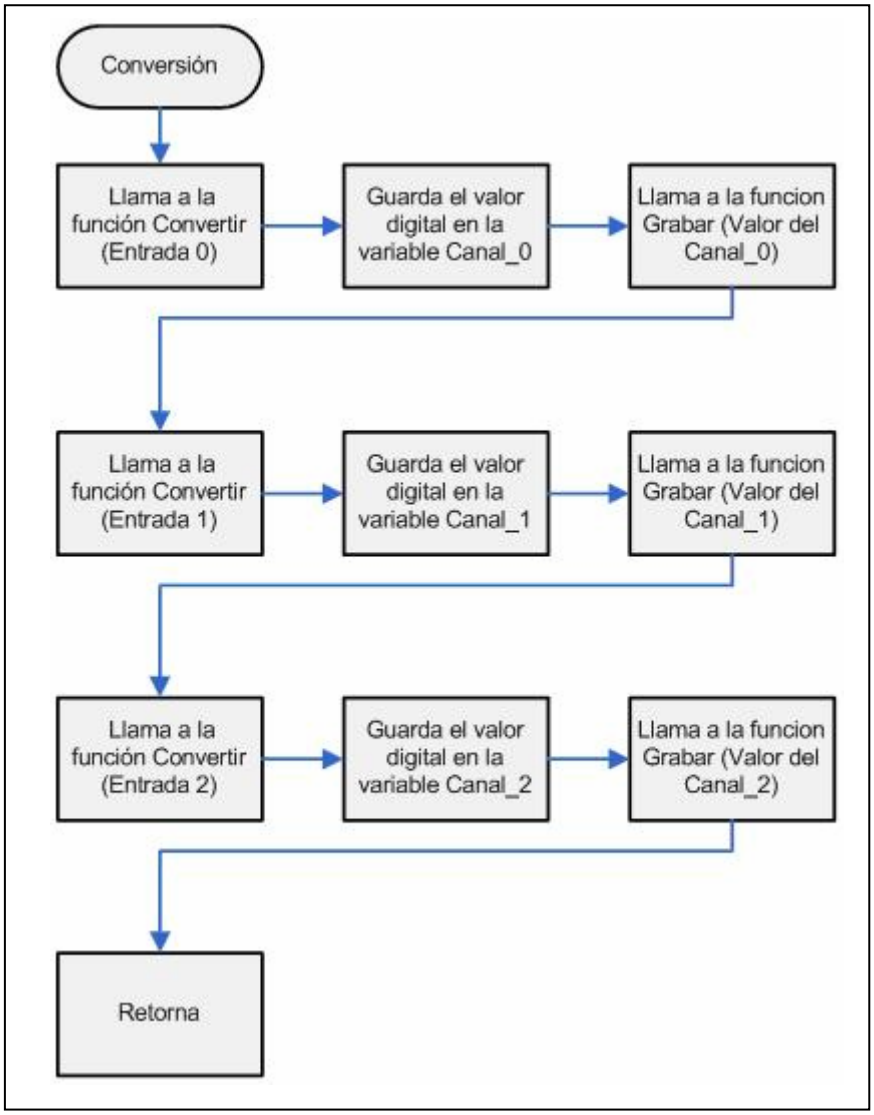


Figura. 4.9 Diagrama de flujo función Conversión

La conversión análoga digital se realiza con 10 bits aunque los dos últimos bits del valor de conversión son descartados dejando solo los 8 bits mas significativos que son luego transferidos a los registros.

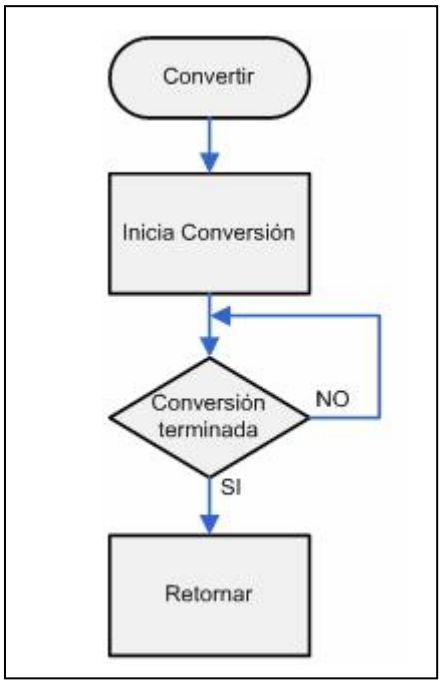


Figura. 4.10 Diagrama de flujo función Convertir

4.2.3.6 Grabación en memoria EEPROM

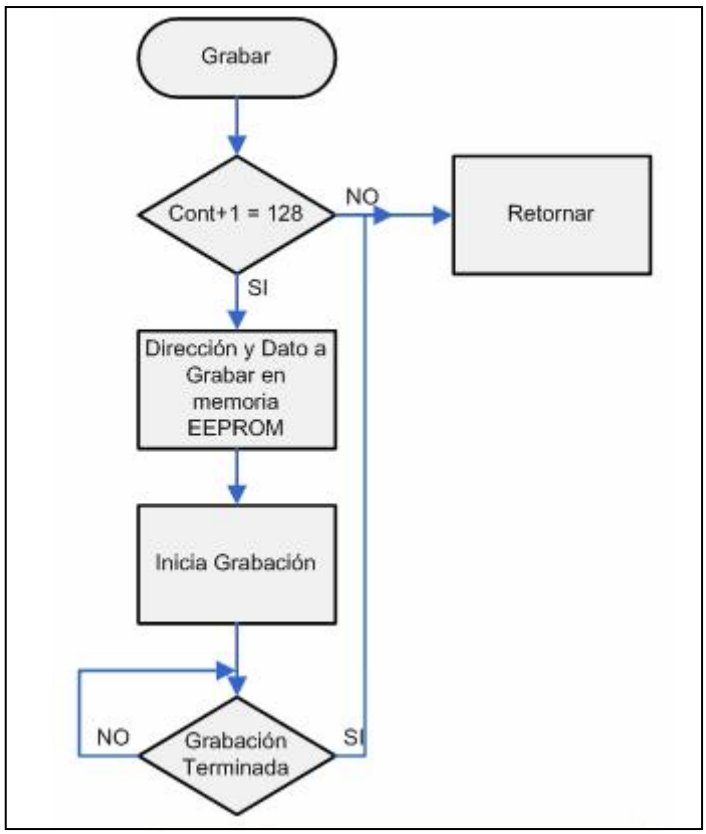


Figura. 4.11 Diagrama de flujo función Grabar

La función “Grabar” lleva la información más importante a la memoria EEPROM de manera que pueda ser leída después de haber desconectado la red neuronal. Como se indicó anteriormente solo se graban los datos cuando el registro “CONT+1” sea igual a 128. Se debe tener en cuenta que el valor que se desea grabar debe ser transferido al registro “DATO” antes de llamar a esta función, al igual que la dirección al registro “DIREC”.

4.2.3.7 Aprendizaje Positivo

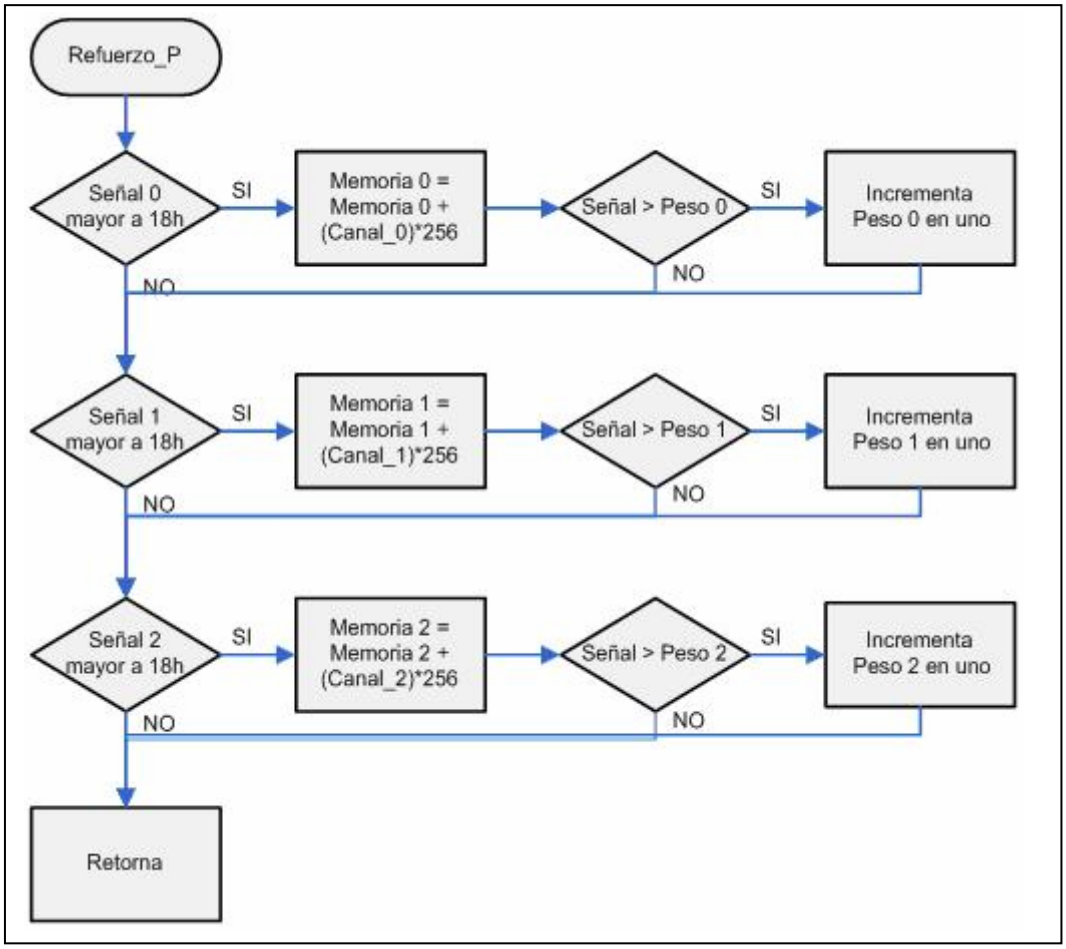


Figura. 4.12 Diagrama de flujo función Refuerzo Positivo

La función “Refuerzo_P” aumenta el nivel de memoria para cada peso de acuerdo a la señal que se recibe en su respectiva entrada cuidando no superar el valor máximo de memorización. La señal de entrada es sumada con el valor del segundo registro más

significativo ya que cuando se agrega este valor en el registro menos significativo de la memoria se pierden muy rápido las señales aprendidas. Es por esta razón que en el diagrama de flujo se indica que la memoria es aumentada en un valor igual a la señal de entrada multiplicada por 256 ya que al ser almacenada la señal en el segundo registro y no en el primero se tiene el mismo efecto que si se realizará la multiplicación.

También se realiza un cambio en los pesos para que estos se acerquen más a la señal que produjo la activación incrementando su valor en uno si la señal es mayor que el peso respectivo.

En esta función se realizó un filtro para que las señales muy bajas no afecten el aprendizaje de la neurona y solo aquellas señales representativas de la situación se puedan memorizar, esto se realiza comparando la señal, solo aquellas que tengan un valor mayor a 18h tendrán un refuerzo.

4.2.3.8 Retardos

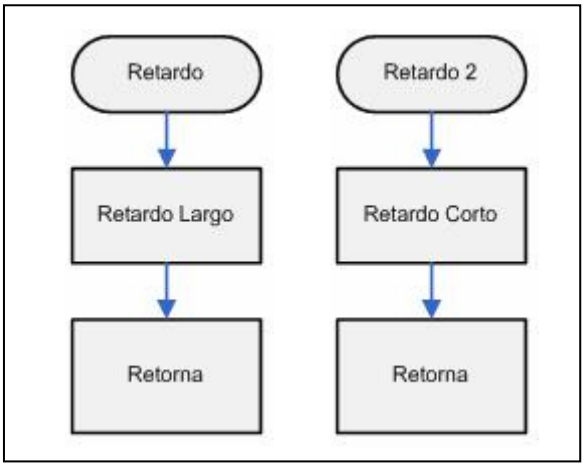


Figura. 4.13 Diagramas de flujo funciones de Retardo

Las funciones “Retardo” y “Retardo2” permiten simular ciertos tiempos que se dan entre las activaciones de las neuronas. Están implementadas en un bucle con una variable que se va disminuyendo hasta llegar a cero, se crearon dos funciones diferentes ya que para algunos casos es necesario un retardo largo y para otros un retardo corto.

4.2.3.9 Aprendizaje Negativo

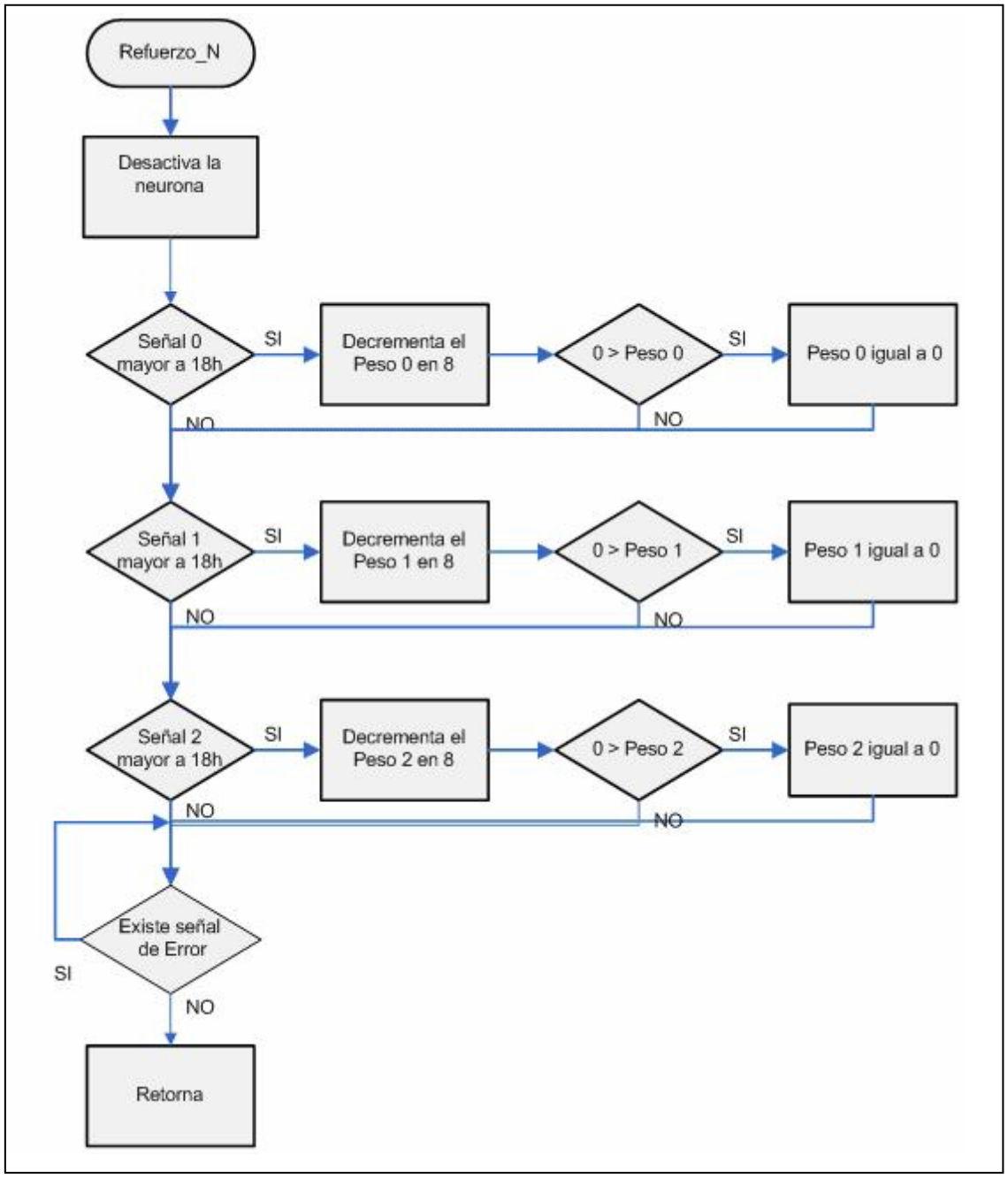


Figura. 4.14 Diagrama de flujo función Refuerzo Negativo

En la función “Refuerzo_N” se desactiva la señal de la neurona para que no se siga forzando el error, luego se decrementan los pesos para las señales mayores a 18h, si los pesos llegan a un valor menor a 0 se les da un valor igual a 0. Una vez que se cambia los

pesos se procede a un bucle en el que se mantiene la neurona inactiva mientras se mantenga la señal de error activa, esto con el fin de evitar una nueva activación de la neurona, que lleve a la misma situación de error.

El código del software creado para realizar la simulación de la Red Neuronal Artificial con el microcontrolador PIC12F675 se encuentra en forma detallada en el anexo 1 de este proyecto.

CAPÍTULO 5

DISEÑO E IMPLEMENTACION DE LA PLATAFORMA

5.1 PARTES

A continuación se presentan los gráficos de los elementos con los que se construyo la plataforma móvil.

5.1.1 Ruedas

5.1.1.1 Ruedas Laterales

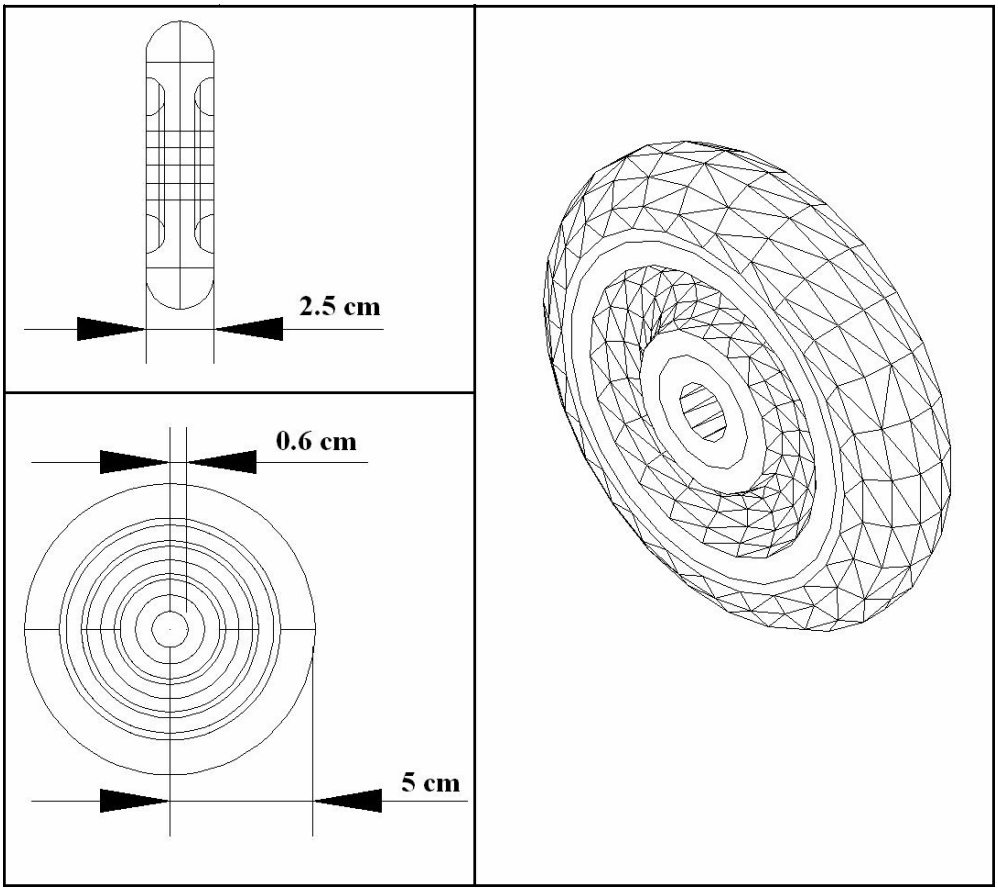


Figura. 5.1. Ruedas Laterales

Las ruedas laterales son plásticas, su radio es de 5 cm. para superar el radio de los motores, estas ruedas se encuentran unidas directamente al eje de cada motor y se ubican a los lados de la base de la plataforma.

5.1.1.2 Rueda Delantera

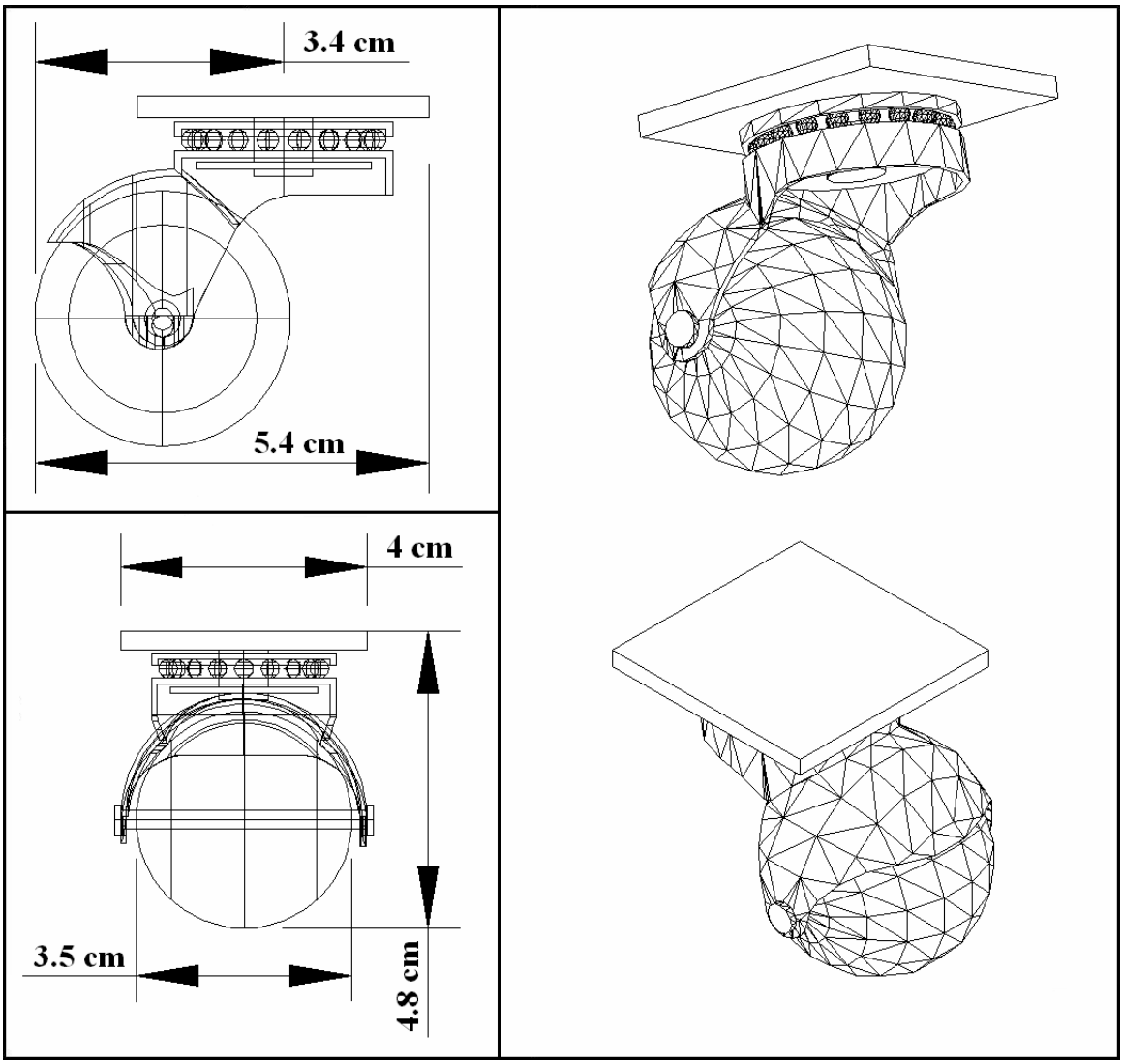


Figura. 5.2. Rueda Pivotal

La rueda delantera es una esfera de caucho que puede girar en cualquier dirección lo que permite a la plataforma tener una buena movilidad. La estructura a la cual se encuentra unida la esfera es metálica. A través de esta estructura metálica la rueda se une a

la base en su parte inferior. La base posee una pequeña parte de madera que sirve para darle a la llanta delantera la misma altura de las llantas laterales nivelando la plataforma.

5.1.2 Motores

Los motores usados son motores de corriente continua con reducción de velocidad mecánica a través de piñones lo que les da un mayor torque y una menor velocidad haciéndolos muy útiles para el trabajo requerido. El rango de trabajo de estos motores es de 5 a 30 Voltios. Consumen una corriente de 500 mA. nominal con un voltaje de alimentación de 6 Voltios y una corriente pico de 1.5 Amperios.

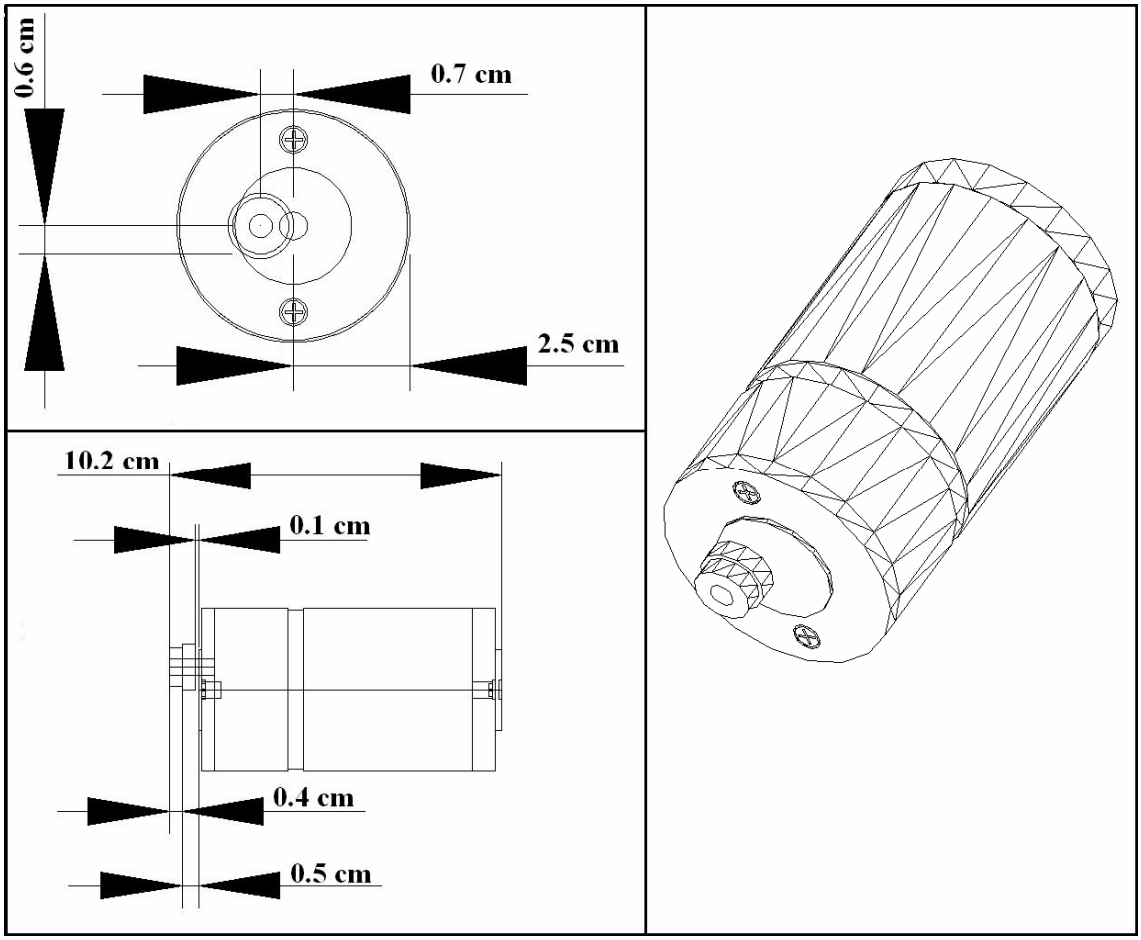


Figura. 5.3. Motores

5.1.3 Baterías

Se usan baterías recargables, de 6 voltios y 5 Amperios hora con un voltaje de carga de 7.25 a 7.45 Voltios y una corriente menor a 2 Amperios. Se tiene dos baterías, una que entregará la energía para el movimiento de los motores y otra que se encargará de alimentar todos los circuitos eléctricos, esto con el fin de evitar que se tenga mucho ruido eléctrico en los circuitos de control que pueda afectar el funcionamiento de la red neuronal.

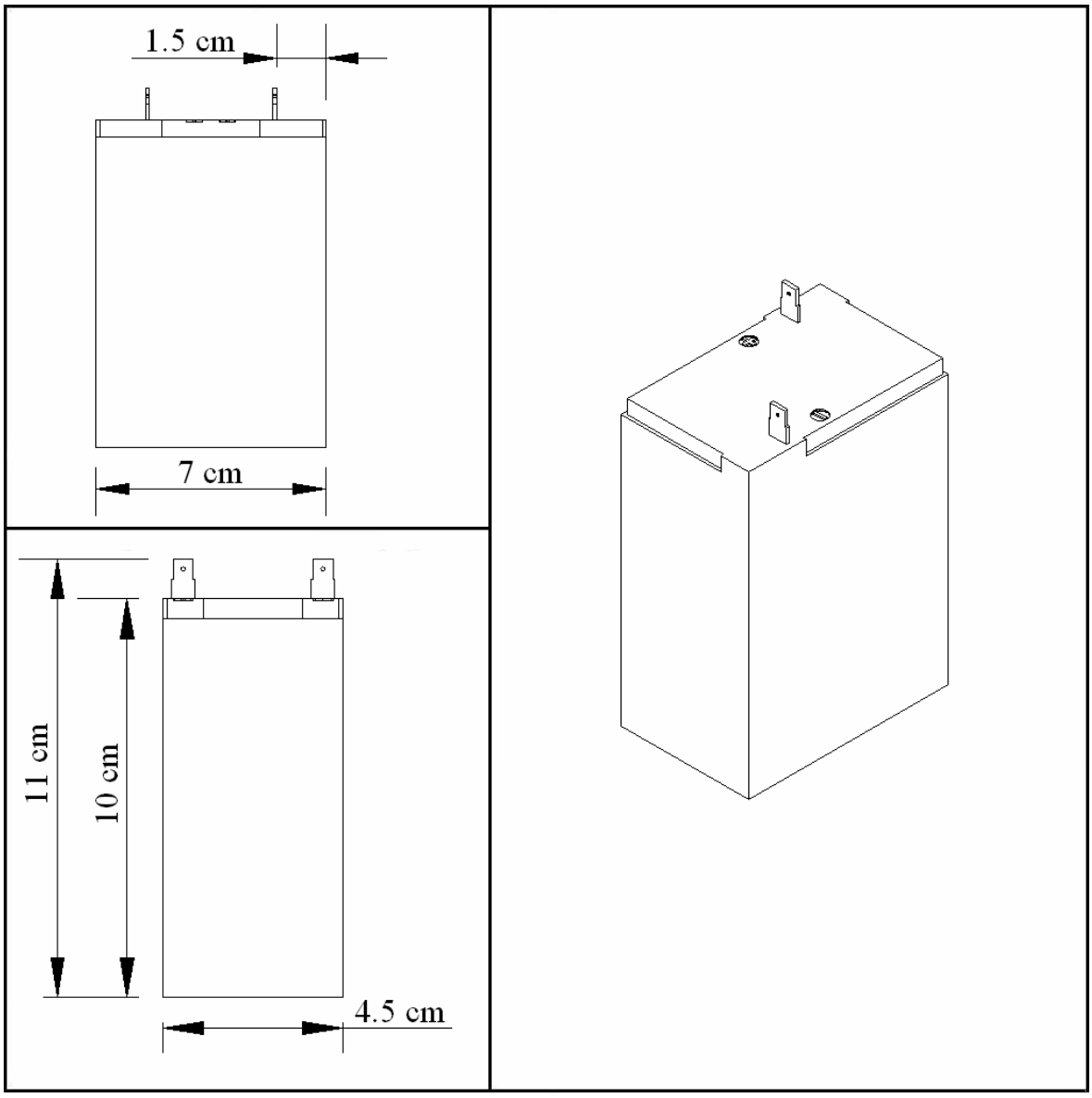


Figura. 5.4. Baterías

5.1.4 Sensores de Proximidad

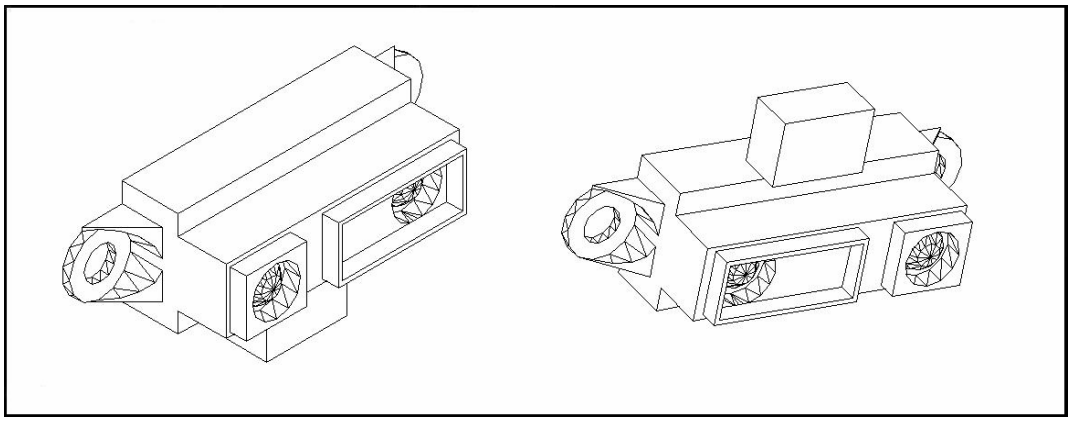


Figura. 5.5. Sensores e proximidad

Para obtener la respuesta de los sensores se tomaron los siguientes datos usando una hoja de papel blanco tamaño oficio:

Distancia (Centímetros)	Voltaje (Voltios)
3,5	2,96
5	2,2
10	1,2
15	0,8
20	0,61
25	0,45
30	0,33
35	0,25
40	0,17
45	0,086
50	0,024

Tabla. 5.1. Valores de respuesta de los sensores

Una vez conseguidos se encontró la ecuación que define la línea de tendencia mas cercana a estos puntos, esta ecuación será usada en el capitulo 6 para encontrar los valores aproximados en los cuales debe activarse una neurona y facilitar la interpretación de los datos obtenidos luego de activar la red neuronal durante algún tiempo.

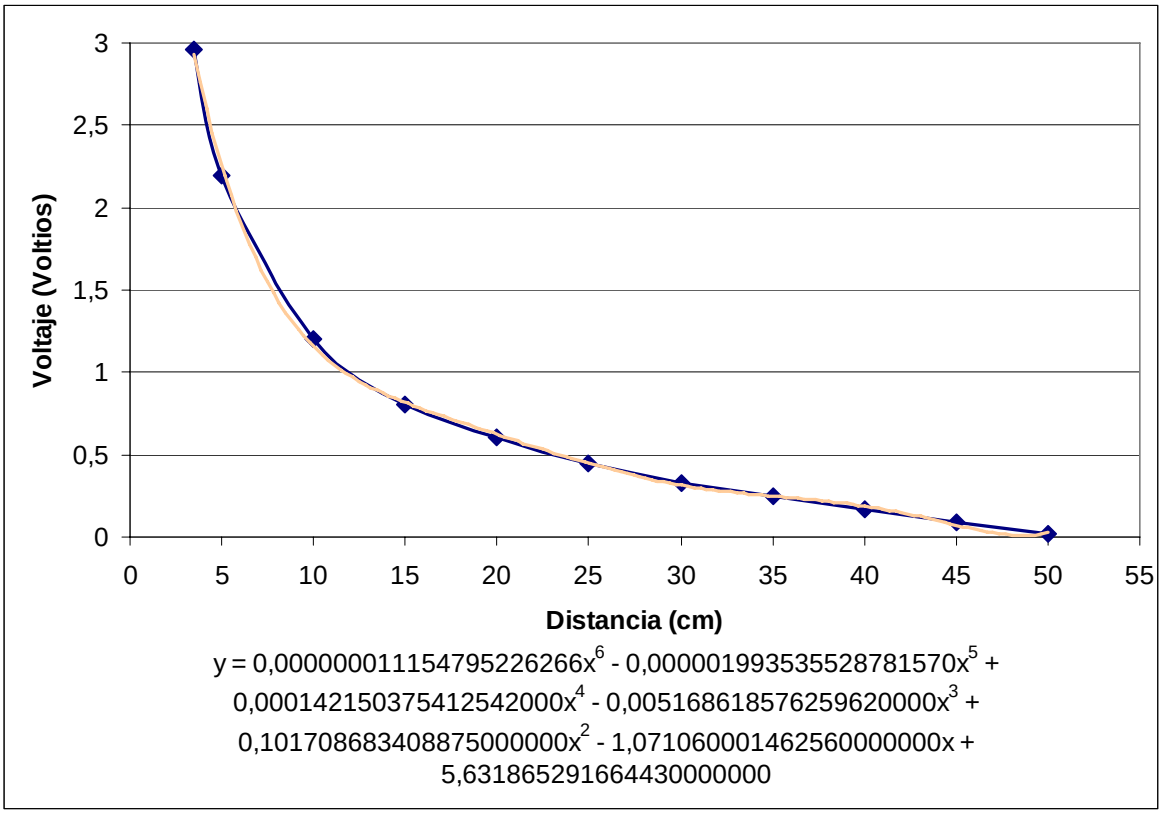


Figura. 5.6. Ecuación aproximada de respuesta de los sensores

Las otras características de los Sensores, como son tamaño, forma, principios de funcionamiento, tiempos de actualización de la señal y otras, se encuentran descritas en el data sheet por lo cual no serán detalladas en este capítulo.

5.1.5 Base

La base de la plataforma es principalmente de madera, aunque se le han agregado rieles metálicos y tornillos que permitirán ajustar las piezas fácilmente. Los rieles inferiores de la base ayudarán a fijar los motores mientras que los rieles superiores permiten fijar las baterías, la caja en la parte inferior ayuda a nivelar la rueda delantera y el paralelogramo en la parte superior sirve para fijar los sensores de proximidad con ángulos de 60º entre si para dar una mejor cobertura.

Además de fijar todos los elementos, la base contiene 12 pulsadores y una banda plástica entre ellos que sirven para determinar la señal de choque en la plataforma.

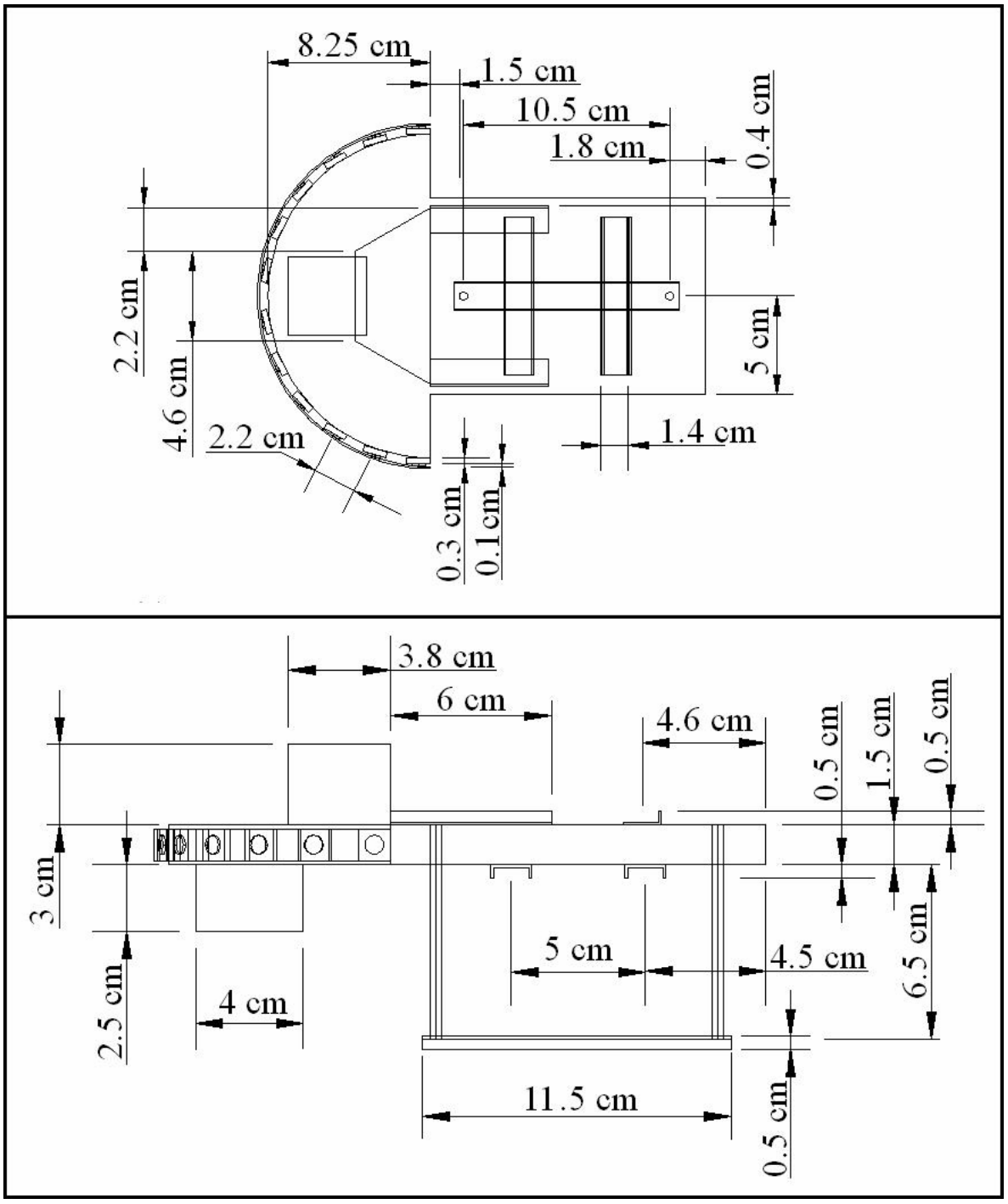


Figura. 5.7. Base Vista 2D

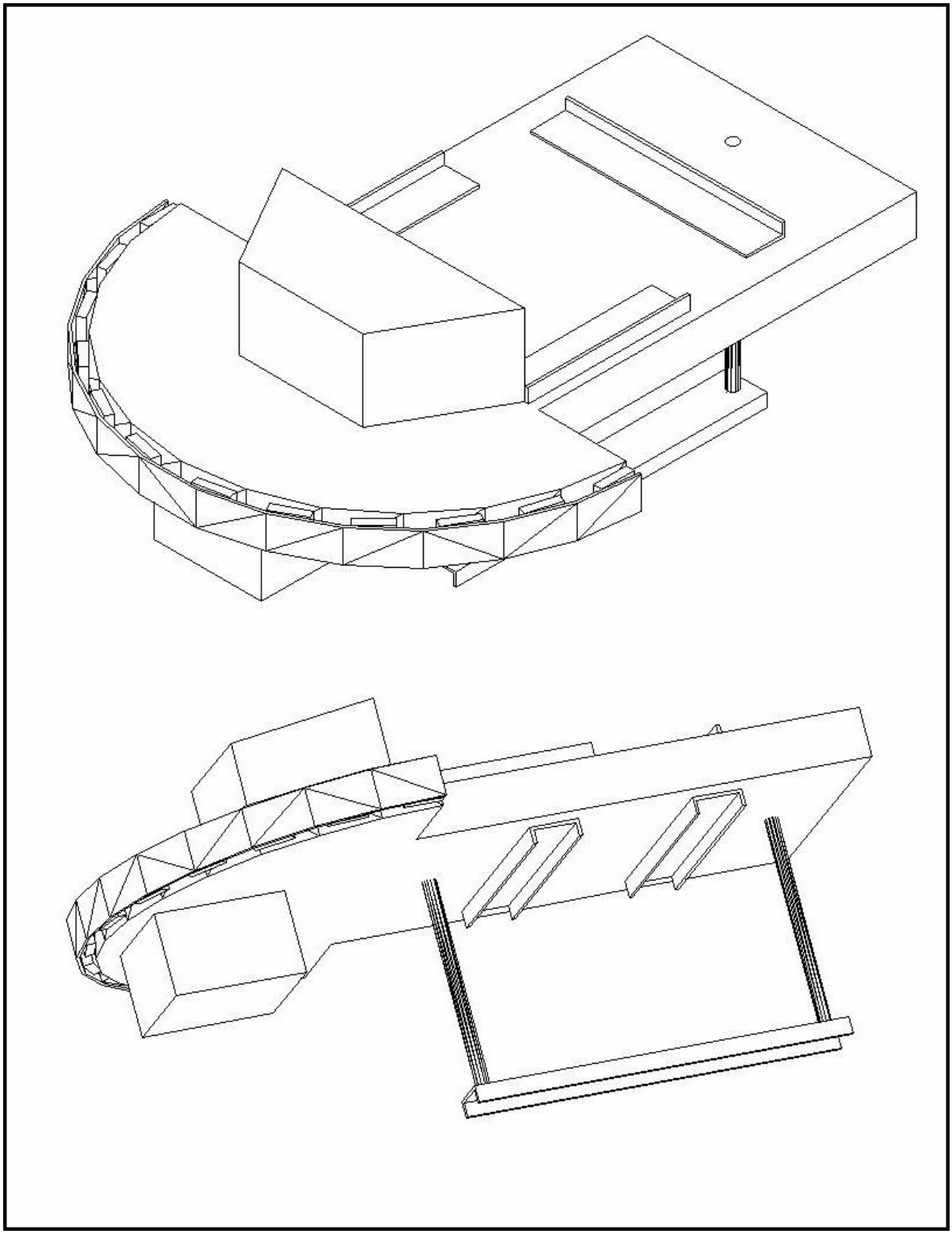


Figura. 5.8. Base Vista 3D

5.2 PLATAFORMA

En las figuras 5.9 a la 5.12 se puede observar la plataforma completa con todos los elementos en su respectivo lugar. Se ofrece la vista lateral, superior y frontal así como dos puntos de vista tridimensionales para una mejor comprensión de la forma en la que se encuentra construida la plataforma de pruebas.

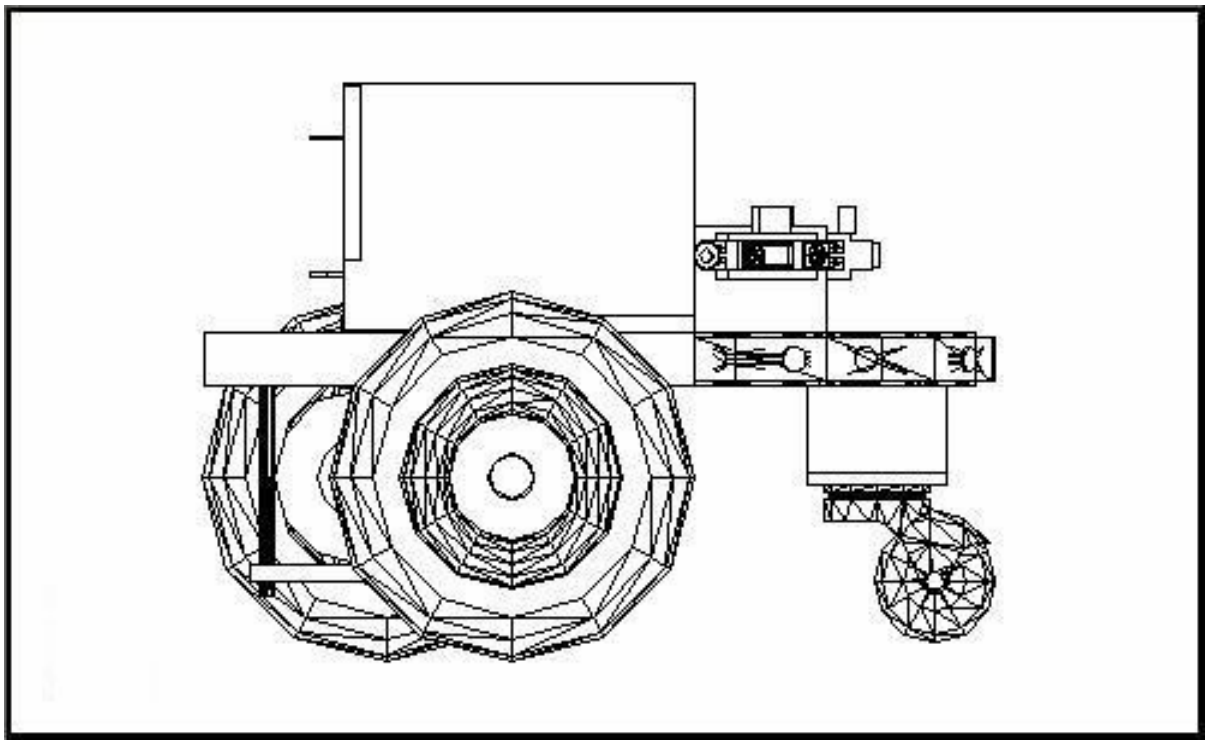


Figura. 5.9. Plataforma Completa Vista Lateral

En estas figuras se puede observar la disposición de las llantas laterales, la cual no es simétrica y la ubicación de los sensores de proximidad tratando de cubrir al máximo el espacio frontal de la plataforma de prueba, la llanta frontal solo sirve para dar estabilidad ya que no posee tracción y los sensores de choque, pulsadores, se encuentran distribuidos simétricamente sobre la parte frontal de la base de la plataforma.

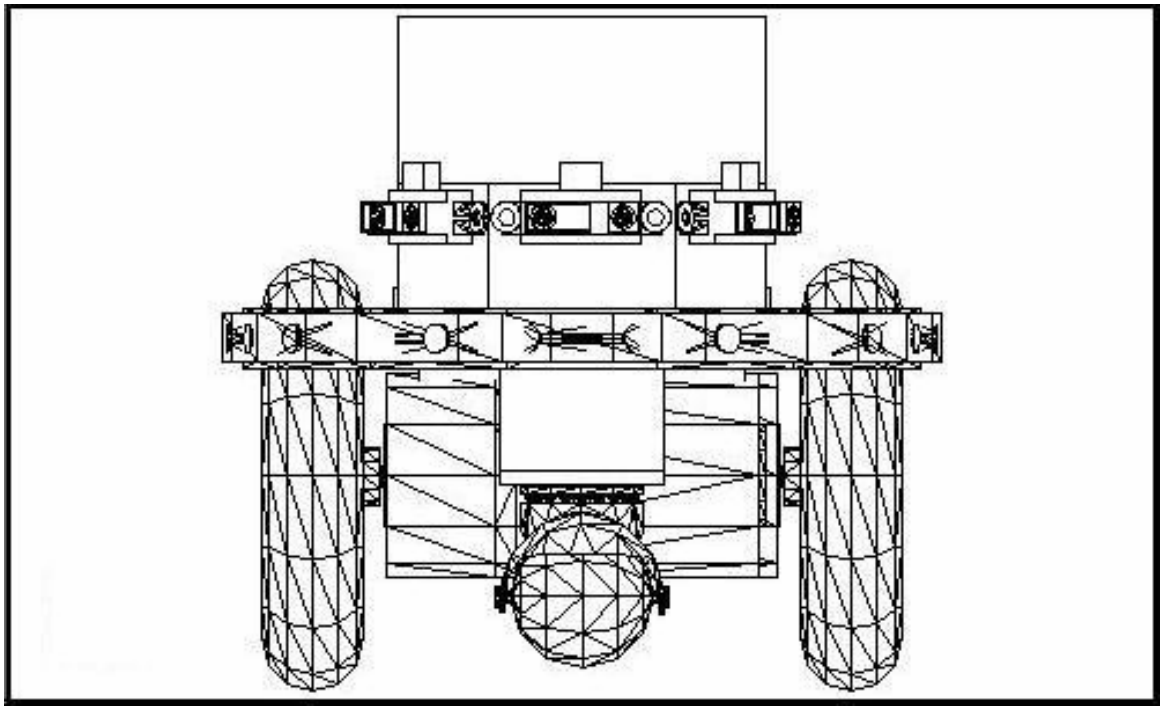


Figura. 5.10. Plataforma Completa Vista Frontal

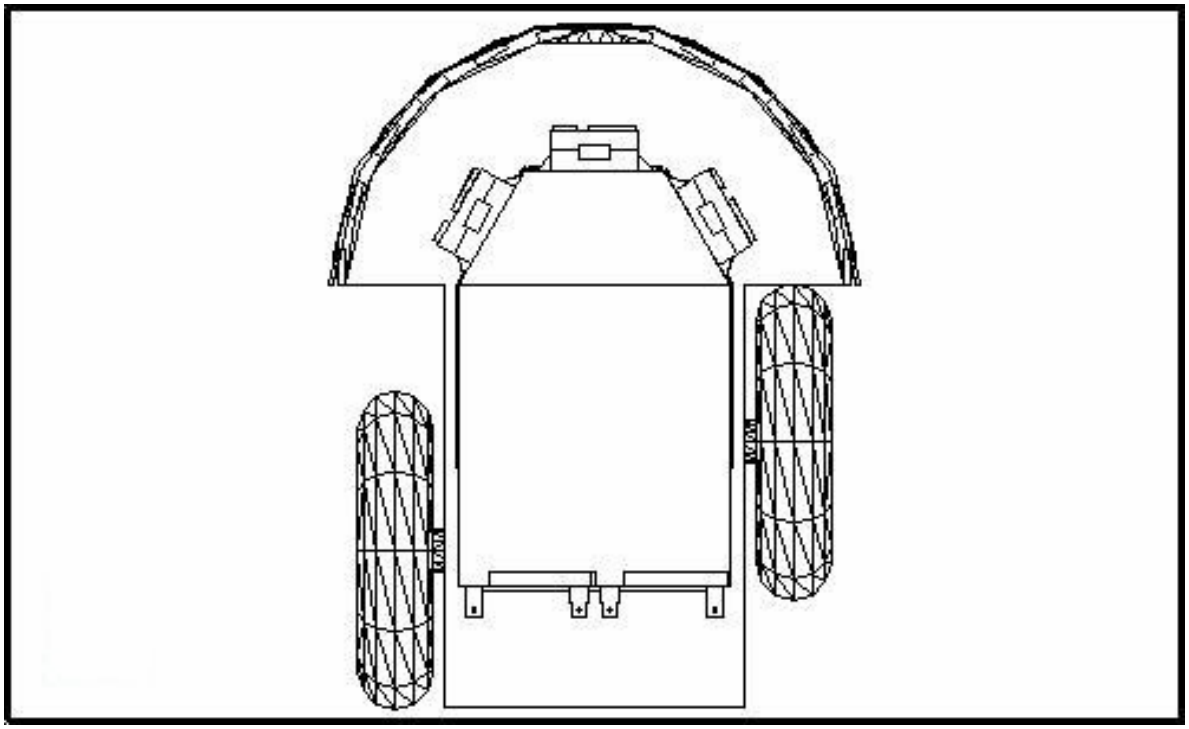


Figura. 5.11. Plataforma Completa Vista Superior

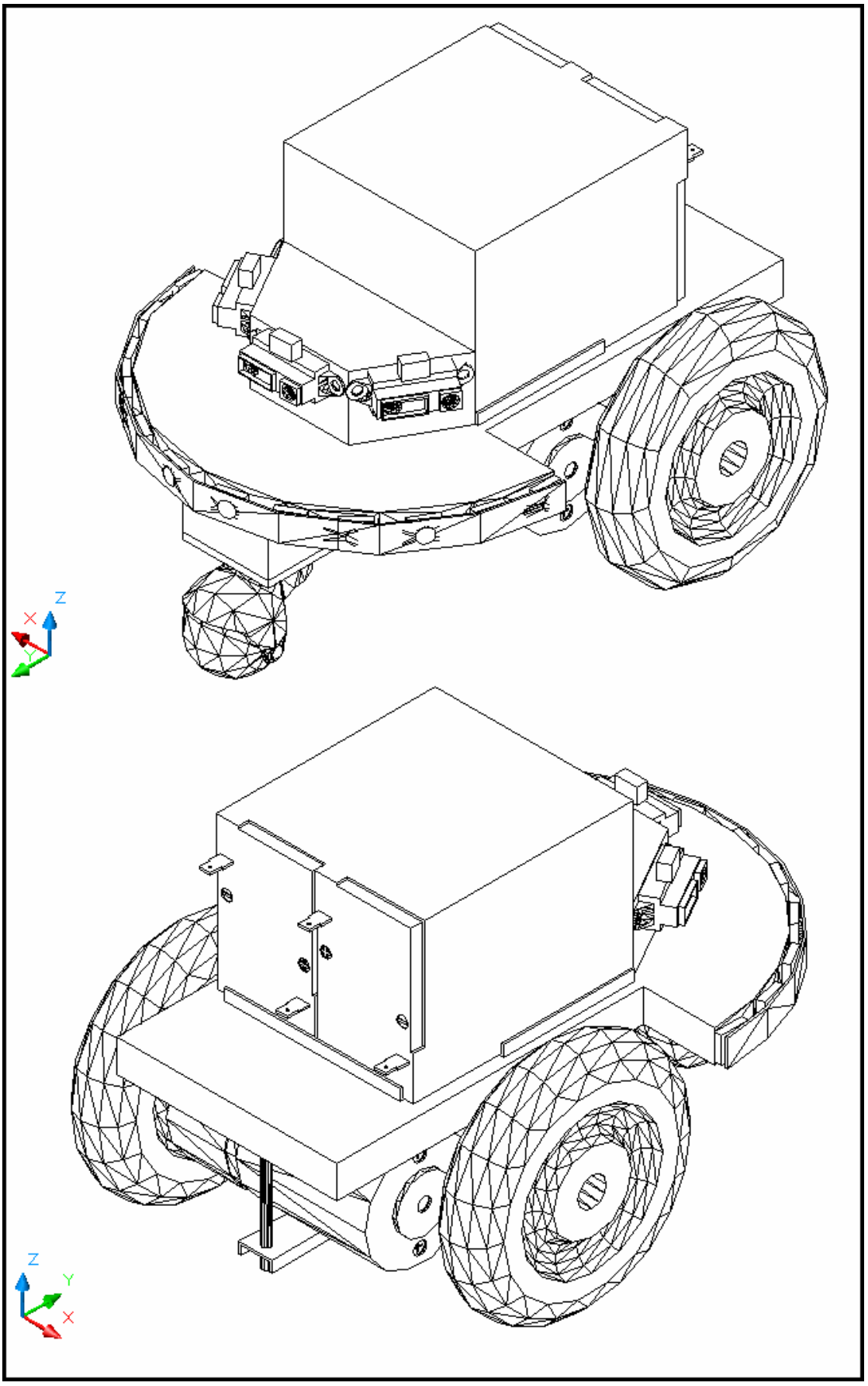


Figura. 5.12. Plataforma Completa Vista 3D

CAPÍTULO 6

PRUEBAS Y RESULTADOS

6.1 RESULTADOS DE LA RED NEURONAL

La red neuronal diseñada logró adaptarse a las condiciones de movimiento, consiguiendo después de algunos errores, alrededor de 4 a 5 equivocaciones, ajustar los pesos de conexión para cada una de sus entradas. Luego de esta etapa en la que la red comete algunos errores se continúa el aprendizaje positivo que lleva a los pesos que influyen en su activación a valores cercanos a las señales de entrada que reciben.

Una vez que se tienen estables los pesos de conexión se conoce cual será el comportamiento de cada neurona ante distintas situaciones; este comportamiento solo podrá alterarse con señales de error o cuando transcurra mucho tiempo sin que se reciban refuerzos de memorización. Los refuerzos de memorización servirán para eliminar cualquier señal que se pudo haber presentado en un caso específico durante el aprendizaje y que no sea característica de los casos en que la neurona debe activarse, es decir que elimina la influencia de señales que se presenten por casualidad.

Los pesos de estabilización así como el valor de las señales necesarias para la activación de la neurona pueden ser alterados mediante la influencia negativa entre neuronas de modo que requiera de señales más fuertes para la activación o mediante una entrada cualquiera que lleve a la neurona a un nivel cercano a activarse, con lo cual se requerirá de señales mas pequeñas para generar una respuesta.

A continuación se explicarán los valores obtenidos en la memoria EEPROM tras una de las pruebas realizadas. Se debe considerar que todas las operaciones realizadas se harán con números hexadecimales.

6.1.1 Análisis de la Neurona 1.

En la figura 6.1 se tienen los valores correspondientes a la memoria EEPROM de la neurona numero uno luego de un periodo de aprendizaje, esta neurona está conectada al motor que mueve a la rueda del lado izquierdo. La conexión de los sensores de proximidad para esta neurona es la siguiente:

- Sensor de Proximidad Izquierdo conectado al Canal de entrada 0.
- Sensor de Proximidad Central conectado al Canal de entrada 1.
- Sensor de Proximidad Derecho conectado al Canal de entrada 2.

Para la interpretación de los datos en la memoria EEPROM se debe observar la información de la figura 4.6 “Distribución de memoria”.

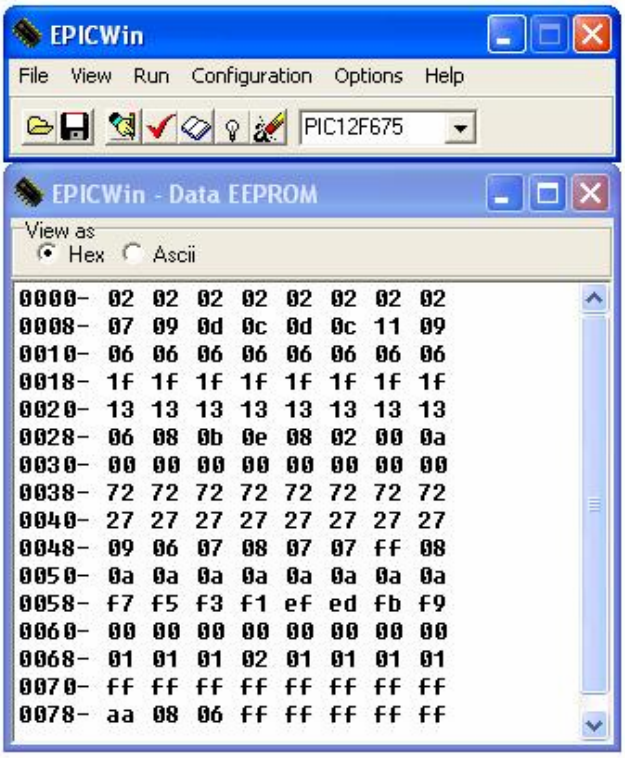


Figura. 6.1 Valores memoria EEPROM neurona 1

Con los datos conseguidos en la lectura de la memoria, se puede conocer cual es el comportamiento de la neurona cuando se presenten ciertas señales en los sensores de

proximidad. Así para los valores de esta neurona en especial podemos observar lo siguiente.

- Cuando se presente un obstáculo a la plataforma solamente en su lado izquierdo esta neurona no se activará aun sin la existencia de señales de inhibición ya que el peso que posee para esta conexión es muy pequeño. Considerando la existencia de señales iguales a cero en los sensores frontal y derecho, la señal necesaria para la activación sería:

$$Umbral\_Activación = 800$$
$$Peso\_0 = 2$$
$$800 \leq (Senal\_entrada) * (PESO\_0)$$
$$Senal\_entrada = 400$$

Como la máxima señal que pueden entregar los sensores es igual a A6, según los datos tomados, sería imposible alcanzar la activación.

- Cuando se presente un obstáculo solamente en su lado derecho esta neurona se activará fácilmente ya que el peso que posee para esta conexión es alto. Considerando de igual manera señales de cero en los otros sensores, la señal necesaria para la activación sería:

$$Umbral\_Activación = 800$$
$$Peso\_0 = 27$$
$$800 \leq (Senal\_entrada) * (PESO\_0)$$
$$Senal\_entrada = 34$$

Es decir que la neurona se activaría al detectar un objeto a una distancia aproximada de 12 cm. según la línea de tendencia vista en la figura 5.6. “Ecuación aproximada de respuesta de los sensores”.

- Cuando se presente un obstáculo a la plataforma en la parte frontal sin que se detecte señal a ninguno de los dos lados la neurona se activará con una señal bastante alta ya que el peso en esta conexión no es muy grande. La señal necesaria para la activación seria:

$$Umbral\_Activación = 800$$
$$Peso\_0 = 13$$
$$800 \leq (Senal\_entrada) * (PESO\_0)$$
$$Senal\_entrada = 6B$$

Es decir que la neurona se activaría al detectar un objeto a una distancia aproximada de 5,6 cm. según la línea de tendencia vista en la figura 5.6. “Ecuación aproximada de respuesta de los sensores”. Esto en el caso de no existir influencia negativa de otra neurona, y como se verá a continuación es muy probable que esta influencia exista ya que la neurona número dos tiene un mayor peso para esta conexión.

6.1.2 Análisis de la Neurona 2.

En la figura 6.2 se encuentran los valores correspondientes a la memoria EEPROM de la neurona número dos luego de un periodo de aprendizaje, esta neurona está conectada al otro motor, es decir que controla la rueda del lado derecho. La conexión de los sensores de proximidad para esta neurona es la siguiente:

- Sensor de Proximidad Derecho conectado al Canal de entrada 0.
- Sensor de Proximidad Central conectado al Canal de entrada 1.
- Sensor de Proximidad Izquierdo conectado al Canal de entrada 2.

Para la interpretación de los datos en la memoria EEPROM se debe observar la información de la figura 4.6 “Distribución de memoria” como ya se observó en la parte anterior.

Además se debe tener en cuenta que la conexión de los sensores es distinta para la neurona número dos. Para verificar la diferencia en las conexiones de las dos neuronas se puede observar la figura. 4.1 “Diagrama de conexión de elementos”.

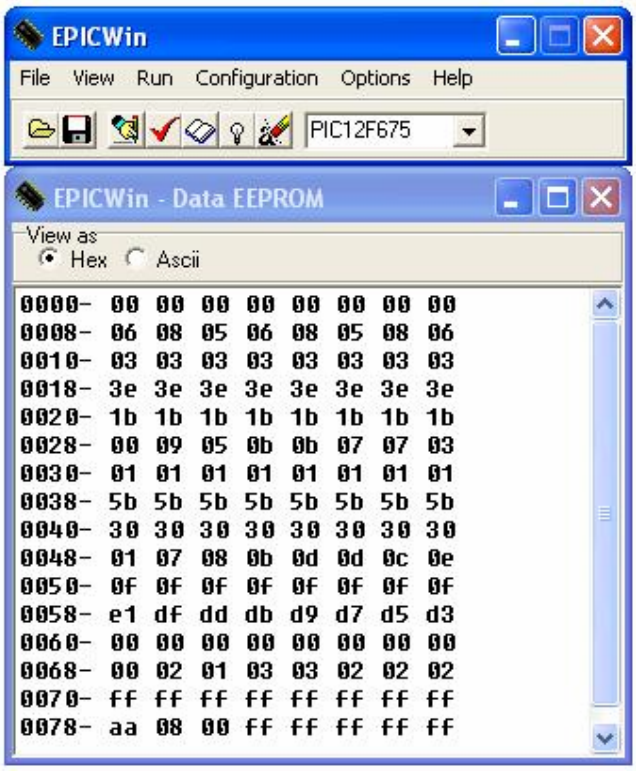


Figura. 6.2 Valores memoria EEPROM neurona 2

Con estos datos de la memoria EEPROM se interpreta el siguiente comportamiento de la neurona.

- Cuando se presente un obstáculo a la plataforma solamente en su lado derecho esta neurona no se activará ya que el peso que posee para esta conexión es igual a cero.

- Cuando se presente un obstáculo solamente en su lado izquierdo esta neurona se activará fácilmente ya que el peso que posee para esta conexión es alto. Considerando señales de cero en los otros sensores, la señal necesaria para la activación sería:

$$Umbral\_Activación = 800$$
$$Peso\_0 = 30$$
$$800 \leq (Senal\_entrada) * (PESO\_0)$$
$$Senal\_entrada = 2A$$

Es decir que la neurona se activaría al detectar un objeto a una distancia aproximada de 15,5 cm. según la línea de tendencia vista en la figura 5.6. “Ecuación aproximada de respuesta de los sensores”.

- Cuando se presente un obstáculo a la plataforma en la parte frontal sin que se detecte señal a ninguno de los dos lados la neurona se activará con una señal alta, pero menor que la señal necesaria en la neurona número uno con lo cual la plataforma tenderá a girar, siempre que tenga un obstáculo al frente, hacia su lado derecho. La señal necesaria para la activación sería:

$$Umbral\_Activación = 800$$
$$Peso\_0 = 1B$$
$$800 \leq (Senal\_entrada) * (PESO\_0)$$
$$Senal\_entrada = 4B$$

Es decir que la neurona se activaría al detectar un objeto a una distancia aproximada de 8,1cm. según la línea de tendencia vista en la figura 5.6. “Ecuación aproximada de respuesta de los sensores”.

6.2 RESULTADOS PLATAFORMA

La plataforma utilizada para las pruebas tiene un buen comportamiento exceptuando los sensores que permiten determinar los choques contra los obstáculos; los otros elementos de la plataforma no presentan ningún problema permitiendo cumplir con los objetivos.

Los motores utilizados permiten un buen desplazamiento en suelo plano y permiten superar obstáculos pequeños que se encuentren en el piso; aunque los motores se

encuentren ubicados asimétricamente, es decir uno más adelante que otros, esto no influye en los resultados ya que no se necesita tener giros precisos o giros iguales hacia ambos lados.

Las baterías permiten mantener a la plataforma en funcionamiento por aproximadamente 2 horas, se debe tener en consideración que una de las baterías alimentará a los motores y otra a los circuitos electrónicos por lo que una de las baterías se descargará mas rápido que la otra, por lo general la primera batería en descargarse es la de los motores.

La base de la plataforma resiste bien el peso y asegura los motores de forma que estos no se muevan hacia fuera o roten causando rozamiento de las llantas con la base, además permite un fácil desensamblaje de los motores; la sujeción de la batería permite retirarla de manera sencilla haciendo muy cómoda su manipulación.

Los sensores de proximidad tuvieron un rango de funcionamiento aun mayor a lo esperado siendo este entre 4 y 40 cm., además tienen una gran inmunidad al ruido tanto eléctrico como lumínico; los tres sensores cubren completamente la parte frontal de la plataforma pero en algunos casos los objetos pequeños no son detectados hasta que se encuentran cerca de la plataforma por lo que seria aconsejable usar mas sensores para cubrir el área frontal.

Los sensores de choque no funcionaron apropiadamente, la plataforma debe ser ayudada en la etapa de aprendizaje ya que al momento de producirse un choque contra las paredes, en algunos casos, los pulsadores que deben sensar el choque no se oprimen totalmente con lo cual no se produce la señal de error para la red neuronal, otras veces estos pulsadores se quedan enganchados en los objetos o provocan demasiado rozamiento por el cual la plataforma no logra separarse de la pared.

6.3 POSIBLES CAMBIOS EN LA RED NEURONAL

6.3.1 Pesos Negativos

Los pesos negativos presentarían grandes ventajas sobre los pesos que solo tienen un valor positivo. Un peso negativo puede representar la influencia inversa que puede tener una señal en la activación de una neurona, dentro de este trabajo se llegaría a establecer pesos negativos para los casos en los que una neurona no debe activarse, por ejemplo cuando se acerque a un objeto por el lado izquierdo la neurona que controla al motor de la rueda izquierda le correspondería sufrir una influencia negativa y entre mas próximo este el objeto la influencia será mayor, ya que si se activa esta neurona produciría un giro hacia el lado izquierdo que llevaría a un choque. Incluso teniendo pesos negativos la señal que lleva la influencia entre las neuronas podría ser una entrada igual a las otras y adaptaría su peso con el aprendizaje de la red.

6.3.2 Cálculo de pesos con el cambio de la señal

Usando un método de actualización de los pesos en el cual se calcule el valor del mismo de acuerdo al cambio presentado entre una señal de entrada y la señal que se presentó anteriormente se podrían generar pesos más grandes para aquellas entradas que tuviesen señales con cambios rápidos y muy fuertes, además se lograría tener los pesos negativos, anteriormente descritos, cuando una señal disminuya su valor ante un evento. Una ecuación para el cálculo de estos pesos podría ser.

$$Peso = Señal\_Actual - Señal\_Anterior$$

6.3.3 Umbral de activación variable.

Si se implementara un umbral de activación para la neurona que fuera variable se tendría la ventaja de poder utilizar sensores y señales de entrada en distintos rangos de voltaje, también se podrían tener umbrales negativos combinados con pesos negativos que lograrían utilizar señales con lógica inversa, señales que ante un evento disminuyan su valor en lugar de aumentarlo.

6.3.4 Multiplicación simultanea para todas las entradas

Uno de los mayores inconvenientes para la simulación del comportamiento de la red neuronal con los microcontroladores es que estos procesan y ejecutan las instrucciones secuencialmente, esto dificulta imitar exactamente el comportamiento de las neuronas, ya que estas obtienen la información de la señal en todas sus entradas al mismo tiempo, Si se logra, cambiando el software de simulación, imitar este comportamiento se tendría un mejor resultado de la interacción entre neuronas.

CAPÍTULO 7

CONCLUSIONES Y RECOMENDACIONES

7.1 CONCLUSIONES

- Se concluye que aunque se intente imitar el comportamiento de las neuronas con los microcontroladores estos no pueden realizar el procesamiento de todas las señales al mismo tiempo, por lo cual será mucho más difícil obtener la respuesta deseada de la red neuronal. Este procesamiento en paralelo es muy importante en el momento de interacción de las neuronas ya que permite una diferenciación del comportamiento de las mismas ante entradas iguales.

- Se destaca la gran flexibilidad en el comportamiento de la red neuronal ya que al no establecer reglas de comportamiento fijas la red puede aprender sus propias reglas durante el funcionamiento y puede cambiarlas en cualquier momento lo cual le permite adaptarse fácilmente a cualquier situación. Es decir que no posee unos pesos fijos en su fase de trabajo.

- También se destaca la adaptabilidad de la red neuronal ya que al no tener unos pesos fijos desde el comienzo de su activación permite el intercambio entre los sensores y motores e incluso se admite el intercambio entre las neuronas ya que estas poseen exactamente el mismo programa de funcionamiento. En caso de sistemas más grandes esta adaptabilidad ayudaría mucho en el momento de la conexión de la red neuronal.

- Se concluye que aunque las redes neuronales no son los sistemas de control mas óptimos para situaciones bien definidas e invariables, como es el caso de evitar choques contra las paredes, pueden ser inmejorables en las situaciones que

contengan cambios en el ambiente ya que poseen una gran capacidad de adaptación.

- Las redes neuronales son muy útiles en campos donde se posee gran cantidad de información y no se tiene una clara idea de la forma en la que esta se encuentra relacionada.

- Esta red neuronal es un sistema de control que permiten ser ampliado de manera sencilla ya que se puede simplemente conectar nuevos sensores a otras entradas y la red por si misma se encargará de integrar la influencia de estos en sus reglas de comportamiento sin la necesidad de intervención humana, evitando tener que rediseñar todo el sistema de control.

- Las redes neuronales funcionan mejor que otros sistemas de control en problemas con grandes cantidades de datos que deban ser procesados ya que cada neurona se encargará de procesar una parte de la información. Este procesamiento en paralelo de todas las neuronas las hace mucho más rápidas para este tipo de tareas.

7.2 RECOMENDACIONES

- Se recomienda utilizar más sensores en la parte frontal que le permitan a la plataforma identificar plenamente el lugar del cual proviene la señal más alta, con esto se lograría que la red neuronal tenga un mejor criterio para decidir la dirección en la cual debe girar.

- Se recomienda cambiar el sistema de errores ya que los pulsadores colocados en la parte delantera de la plataforma no funcionaron adecuadamente y provocaron que la plataforma se atascara contra algunos obstáculos.

- Se recomienda utilizar microcontroladores con un mayor número de puertos, y usar señales en estos puertos que permitan conocer lo que esta ocurriendo con la red

neuronal ya que el sistema de grabar en la memoria EEPROM solo permite conocer los últimos datos de la red pero no los cambios que estos sufrieron a través del entrenamiento.

REFERENCIAS BIBLIOGRAFICAS

- Neural Networks: A Comprehensive Foundation, Haykin, NY, Macmillan.

- Inteligencia Artificial, Un Enfoque Moderno, Stuart Russell, Prentice Hall, Edición 1995.

- Redes de Neuronas Artificiales, Un Enfoque Practico, Pedro Isasi Viñuela, Prentice Hall, Edición 2004.

- http://electronica.com.mx/neural, Red Neurona tipo Hopfield y tipo Kohonen.

- http://www.tdcat.cesca.es/TESIS_UPC/ AVAILABLE/TDX-0416102-075520/ 26ApendiceD.PDF, Redes Neuronales y Teoria de los Conjuntos Difusos.

- http://www.lfcia.org/~cipenedo/cursos/scx/scx.html, Redes Neuronales.

- http://www.answermath.com/neural_networks.htm, Conceptos Basicos Redes Neuronales.

- ftp://ftp.sas.com/pub/neural/FAQ.html, Preguntas Frecuentes sobre Redes Neuronales.

- http://www.microchip.com, Datasheet PIC12F675.

- http://www.comp.nus.edu.sg/~pris/ArtificialNeuralNetworks/Perceptrons.html, Conceptos básicos Perceptrón.

ANEXO 1

CODIGO DE SIMULACIÓN

INICIALIZACION, PROGRAMA PRINCIPAL Y SUBRUTINAS PARA EL MICROCONTROLADOR PIC12F675

Código de Inicialización

```
;===================================================================
    list p=12f675
    include "p12f675.inc"
    RADIX       DEC
   __CONFIG     3FC4H       ;Configuracion
                            ;Reloj Interno sin salida
                            ;Power-up Timer (ACTIVADO)
                            ;Brown-out Reset (ACTIVADO)
;*******************************************************************
;Reserva de Registros
;*******************************************************************
      org   20h             ;Reservar memoria RAM
AUX         res   3         ;Registros auxiliares para operaciones
CANAL_0     res   1         ;Valor de la conversión A/D del Canal 0
CANAL_1     res   1         ;Valor de la conversión A/D del Canal 1
CANAL_2     res   1         ;Valor de la conversión A/D del Canal 2
PESO_0      res   1         ;Peso para la entrada 0
PESO_1      res   1         ;Peso para la entrada 1
PESO_2      res   1         ;Peso para la entrada 2
MEMO_0      res   3         ;Memoria para el ajuste del peso 0
MEMO_1      res   3         ;Memoria para el ajuste del peso 1
MEMO_2      res   3         ;Memoria para el ajuste del peso 2
MULT_0      res   2         ;Valor de la multiplicación del peso_0 y el canal_0
MULT_1      res   2         ;Valor de la multiplicación del peso_1 y el canal_1
MULT_2      res   2         ;Valor de la multiplicación del peso_2 y el canal_2
NIVEL       res   2         ;Suma de los tres canales multiplicados por los pesos
UMBRAL      res   1         ;Valor del Umbral de Activación de la Neurona
CONT        res   2         ;Contador que indica en que posición grabar
DIREC       res   1         ;Dirección EEPROM en la que se va a grabar
DATO        res   1         ;Valor que se grabará en la memoria EEPROM
BANDERA     res   1         ;bit 0 = 1 existe señal de error
```

```
;=================================================================
        org     0x000
        nop
        goto  Inicializacion
;=================================================================

;*****************************************************************
;Inicialización de las Entradas y Salidas
;*****************************************************************
        org     0x08
Inicializacion
        call    Retardo2        ;Tiempo hasta lograr la estabilización del cristal
        bsf     STATUS,RP0      ;Banco 1
        movlw   0DFh
        movwf   OPTION_REG      ;setea reloj interno
        clrf    VRCON           ;Voltaje Vref (comparador) apagado (Ahorro energía)
        movlw   37h
        movwf   ANSEL           ;GP<2:0> como Entradas Análogas. Oscilador interno 500
KHz
        movlw   1Fh
       movwf    TRISIO          ;GP 5 Salida, GP<4:3> Entradas Digitales, GP<2:0>
                                ;Entradas Análogas
;-----------------------------------------------------------------------------------------
        bcf     STATUS,RP0      ;Banco 0
        clrf    GPIO            ;Inicializa Entradas/Salidas con valor 0
        clrf    INTCON          ;Deshabilita las interrupciones
        movlw   07h
        movwf   CMCON           ;Deshabilita el comparador GP<2:0>
;-----------------------------------------------------------------------------------------

;*****************************************************************
;Inicialización de variables
;*****************************************************************
        clrf    PESO_0          ;Pesos Iniciales en todas las conexiones iguales a cero
```

```
        clrf    PESO_1
        clrf    PESO_2
        clrf    MEMO_0+2        ;Memoria igual a cero para todos los pesos
        clrf    MEMO_1+2
        clrf    MEMO_2+2
        clrf    MEMO_0+1
        clrf    MEMO_1+1
        clrf    MEMO_2+1
        clrf    MEMO_0
        clrf    MEMO_1
        clrf    MEMO_2
        clrf    DIREC
        clrf    CONT
        clrf    CONT+1
        movlw   08h             ;Valor inicial del Umbral de activación (Aprox a 15 cm)
        movwf   UMBRAL
        goto    Inicio          ;Envía al programa principal
;=================================================================
```

Código del Programa Principal

```
;*****************************************************************************
;Programa Principal
;*****************************************************************************
Inicio
        call    Comprueba       ;Función que comprueba si debe activarse la neurona
        bsf     GPIO,GP5        ;desactiva la neurona, se desactiva la salida
                                ;(LOGICA INVERSA)
        call    Contador        ;Función para realizar muestreo
        goto    Inicio          ;Vuelve al inicio
;=================================================================
```

Código Subrutina Comprueba

```
;++++++++++++++++++++++++++++++++++++++++++++++++
;Comprueba
;++++++++++++++++++++++++++++++++++++++++++++++++
Comprueba
        clrf    NIVEL
        clrf    NIVEL+1
        movlw   10h             ;Valor de influencia entre neuronas
        movwf   AUX
        btfss   GPIO,GP4        ;Comprueba si existe influencia (Lógica Inversa)
        call    Restar

        call    Conversion      ;Obtiene los valores Digitales en las entradas
;-----------------------------------------------------------------------------------------------
        clrf    MULT_0          ;Realiza la multiplicación del peso 0 y el valor del canal 0
        clrf    MULT_0+1        ;El valor de la operación se almacenará en los registros
                                ;MULT_0 y MULT_0+1
        movf    CANAL_0,0
        movwf   AUX
        incf    AUX,1           ;Se asegura que aux es mayor a 0
mul0
        movf    PESO_0,0
        addwf   MULT_0,1
        btfsc   STATUS,C
        incf    MULT_0+1,1
        decfsz  AUX,1
        goto    mul0
;-----------------------------------------------------------------------------------------------
        clrf    MULT_1          ;Realiza la multiplicación del peso 1 y el valor del canal 1
        clrf    MULT_1+1        ;El valor de la operación se almacenará en los registros
                                ;MULT_1 y MULT_1+1
        movf    CANAL_1,0
        movwf   AUX
```

```
        incf    AUX,1           ;Se asegura que aux es mayor a 0
mul1
        movf    PESO_1,0
        addwf   MULT_1,1
        btfsc   STATUS,C
        incf    MULT_1+1,1
        decfsz  AUX,1
        goto    mul1
;-------------------------------------------------------------------------------------------------
        clrf    MULT_2          ;Realiza la multiplicación del peso 2 y el valor del canal 2
        clrf    MULT_2+1        ;El valor de la operación se almacenará en los registros
                                ;MULT_2 y MULT_2+1
        movf    CANAL_2,0
        movwf   AUX
        incf    AUX,1           ;Se asegura que aux es mayor a 0
mul2
        movf    PESO_2,0
        addwf   MULT_2,1
        btfsc   STATUS,C
        incf    MULT_2+1,1
        decfsz  AUX,1
        goto    mul2
;-------------------------------------------------------------------------------------------------
        movf    MULT_0+1,0
        movwf   AUX             ;Se mueve el valor que se va a sumar al registro AUX
        call    Sumar
        movf    MULT_1+1,0
        movwf   AUX             ;Se mueve el valor que se va a sumar al registro AUX
        call    Sumar
        movf    MULT_2+1,0
        movwf   AUX             ;Se mueve el valor que se va a sumar al registro AUX
        call    Sumar
;-------------------------------------------------------------------------------------------------
        btfsc   NIVEL+1,7       ;Si el valor es negativo debe salir sin activar la neurona
```

```
        goto    Salir_c
        btfsc   NIVEL+1,2
        goto    Activa          ;Activa la salida de la neurona
        btfsc   NIVEL+1,1       ;Si el valor es mayor a 130560
        goto    Activa          ;Activa la salida de la neurona
        btfsc   NIVEL+1,0       ;Si el valor es mayor a 65280
        goto    Activa          ;Activa la salida de la neurona
        movf    UMBRAL,0
        subwf   NIVEL,0
        btfsc   STATUS,C        ;Si el valor es mayor al Umbral
        goto    Activa          ;Activa la salida de la neurona
        goto    Salir_c
;-------------------------------------------------------------------------------------------
Activa
        bcf     BANDERA,0
        btfsc   GPIO,GP3        ;Comprueba si existe señal de error
        bsf     BANDERA,0       ;Si existe señal de error antes de la activación se activa
bandera
        bcf     GPIO,GP5        ;Activa la neurona (Lógica Inversa)
        call    Retardo         ;Tiempo durante el cual la neurona permanece activa
        call    Retardo         ;Tiempo durante el cual la neurona permanece activa
        btfss   GPIO,GP3        ;Comprueba si existe señal de error
        bcf     BANDERA,0       ;Si luego de la activación no existe señal de error se
                                ;desactiva la bandera
        btfsc   BANDERA,0       ;Comprueba si existe señal de error
        goto    Salir_c         ;Si la bandera esta activa no se realiza ningún aprendizaje
                                ;y termina la activación de la neurona
        call    Act_pesos       ;Actualiza el valor de los pesos
        call    Act_pesos       ;Actualiza el valor de los pesos
        call    Refuerzo_P      ;Llama a la función de refuerzo positivo en los pesos
        btfsc   GPIO,GP3        ;Comprueba si existe señal de error
        call    Refuerzo_N      ;Si hay señal de error luego de la activación de la neurona
                                ;se da un Refuerzo Negativo
        movlw   0AAh
```

```
        movwf   DATO
        movf    CONT,0
        sublw   78h
        movwf   DIREC
        call    Grabar
;-------------------------------------------------------------------------------------------------
Salir_c
        movf    NIVEL+1,0       ;Se graban los valores de NIVEL y NIVEL+1
        movwf   DATO
        movlw   60h
        movwf   DIREC
        call    Grabar
        movf    NIVEL,0
        movwf   DATO
        movlw   68h
        movwf   DIREC
        call    Grabar
        movf    UMBRAL,0
        movwf   DATO
        movf    CONT,0
        sublw   79h
        movwf   DIREC
        call    Grabar
        call    Act_pesos       ;Actualiza el valor de los pesos
        return
;================================================================
```

Código Subrutina Actualizar Pesos

```
;++++++++++++++++++++++++++++++++++++++++++++++++
;Actualizar Pesos
;++++++++++++++++++++++++++++++++++++++++++++++++
Act_pesos
```

```
        btfsc   MEMO_0+2,7      ;Si el valor de memoria es mayor a 32512 se memorizará
                                ;el peso permanentemente y no necesitará actualizaciones
        goto    No_act_0
        movlw   01h
        subwf   MEMO_0,1        ;Decrementa el valor de la memoria en 1
        btfsc   STATUS,C
        goto    No_act_0        ;Si valor es diferente de 0 no se necesitan actualizaciones
        movlw   01h
        subwf   MEMO_0+1,1      ;Decrementa el valor del byte mas significativo de la
                                ;memoria en 1
        btfsc   STATUS,C
        goto    No_act_0        ;Si el byte mas significativo es mayor a 0 no se actualiza
        movlw   01h
        subwf   MEMO_0+2,1      ;Decrementa el valor del byte mas significativo de la
                                ;memoria en 1
        btfsc   STATUS,C
        goto    No_act_0        ;Si el byte mas significativo es mayor a 0 no se actualiza
        clrf    MEMO_0
        clrf    MEMO_0+1
        clrf    MEMO_0+2
act_0                           ;ACTUALIZA EL VALOR DEL PESO 0
                ;Peso actual = Peso anterior + (Señal Actual - Peso anterior)/2
        movf    CANAL_0,0
        movwf   AUX+1
        movf    PESO_0,0
        movwf   AUX
        subwf   AUX+1,0
        btfss   STATUS,C
        goto    Neg_0
        movwf   AUX+2           ;AUX = (Señal Actual - Peso anterior)
        bcf     AUX+2,0
        bcf     STATUS,C
        rrf     AUX+2,1         ;AUX = (Señal Actual - Peso anterior)/2
        rrf     AUX+2,0         ;AUX = (Señal Actual - Peso anterior)/4
```

```
        addwf   PESO_0,1        ;PESO_0  =  Peso actual.
        goto    No_act_0
Neg_0                           ;Si la señal es menor que el valor actual del peso
        movf    AUX+1,0
        subwf   AUX,0
        movwf   AUX+2           ;AUX = (Peso anterior - Señal Actual)
        bcf     AUX+2,0
        bcf     STATUS,C
        rrf     AUX+2,1         ;AUX = (Peso anterior - Señal Actual)/2
        rrf     AUX+2,0         ;AUX = (Señal Actual - Peso anterior)/4
        subwf   PESO_0,1        ;PESO_0  =  Peso actual.
No_act_0
        movf    PESO_0,0        ;Graba el valor del Peso 0
        movwf   DATO
        movlw   00h
        movwf   DIREC
        call    Grabar
        movf    MEMO_0+2,0      ;Graba valor de la memoria 0 (registro mas significativo)
        movwf   DATO
        movlw   10h
        movwf   DIREC
        call    Grabar
        movf    MEMO_0+1,0      ;Graba el valor de la memoria 0
        movwf   DATO
        movlw   18h
        movwf   DIREC
        call    Grabar
;-----------------------------------------------------------------------------------------------
        btfsc   MEMO_1+2,7      ;Si el valor de memoria es mayor a 32512 se memorizará
                                ;el peso permanentemente y no necesitará actualizaciones
        goto    No_act_1
        movlw   01h
        subwf   MEMO_1,1        ;Decrementa el valor de la memoria en 1
        btfsc   STATUS,C
```

```
        goto    No_act_1        ;Si  valor es diferente de 0 no se necesitan actualizaciones
        movlw   01h
        subwf   MEMO_1+1,1      ;Decrementa el valor del byte mas significativo de la
                                ;memoria en 1
        btfsc   STATUS,C
        goto    No_act_1        ;Si el byte mas significativo es mayor a 0 no se actualiza
        movlw   01h
        subwf   MEMO_1+2,1      ;Decrementa el valor del byte mas significativo de la
                                ;memoria en 1
        btfsc   STATUS,C
        goto    No_act_1        ;Si el byte mas significativo es mayor a 0 no se actualiza
        clrf    MEMO_1
        clrf    MEMO_1+1
        clrf    MEMO_1+2
act_1                           ;ACTUALIZA EL VALOR DEL PESO 1
                        ;Peso actual = Peso anterior + (Señal Actual - Peso anterior)/2
        movf    CANAL_1,0
        movwf   AUX+1
        movf    PESO_1,0
        movwf   AUX
        subwf   AUX+1,0
        btfss   STATUS,C
        goto    Neg_1
        movwf   AUX+2           ;AUX = (Señal Actual - Peso anterior)
        bcf     AUX+2,0
        bcf     STATUS,C
        rrf     AUX+2,1         ;AUX = (Señal Actual - Peso anterior)/2
        rrf     AUX+2,0         ;AUX = (Señal Actual - Peso anterior)/4
        addwf   PESO_1,1        ;PESO_1  =  Peso actual.
        goto    No_act_1
Neg_1                           ;Si la señal es menor que el valor actual del peso
        movf    AUX+1,0
        subwf   AUX,0
        movwf   AUX+2           ;AUX = (Peso anterior - Señal Actual)
```

```
        bcf     AUX+2,0
        bcf     STATUS,C
        rrf     AUX+2,1         ;AUX = (Peso anterior - Señal Actual)/2
        rrf     AUX+2,0         ;AUX = (Señal Actual - Peso anterior)/4
        subwf   PESO_1,1        ;PESO_1  =  Peso actual.
No_act_1
        movf    PESO_1,0        ;Graba el valor del Peso 1
        movwf   DATO
        movlw   20h
        movwf   DIREC
        call    Grabar
        movf    MEMO_1+2,0      ;Graba valor de la memoria 1 (registro mas significativo)
        movwf   DATO
        movlw   30h
        movwf   DIREC
        call    Grabar
        movf    MEMO_1+1,0      ;Graba el valor de la memoria 1
        movwf   DATO
        movlw   38h
        movwf   DIREC
        call    Grabar
;-------------------------------------------------------------------------------------------
        btfsc   MEMO_2+2,7      ;Si el valor de memoria es mayor a 32512 se memorizará
                                ;el peso permanentemente y no necesitará actualizaciones
        goto    No_act_2
        movlw   01h
        subwf   MEMO_2,1        ;Decrementa el valor de la memoria en 1
        btfsc   STATUS,C
        goto    No_act_2        ;Si  valor es diferente de 0 no se necesitan actualizaciones
        movlw   01h
        subwf   MEMO_2+1,1      ;Decrementa el valor del byte mas significativo de la
                                ;memoria en 1
        btfsc   STATUS,C
        goto    No_act_2        ;Si el byte mas significativo es mayor a 0 no se actualiza
```

```
        movlw   01h
        subwf   MEMO_2+2,1      ;Decrementa el valor del byte mas significativo de la
                                ;memoria en 1
        btfsc   STATUS,C
        goto    No_act_2        ;Si el byte mas significativo es mayor a 0 no se actualiza
        clrf    MEMO_2
        clrf    MEMO_2+1
        clrf    MEMO_2+2
act_2                           ;ACTUALIZA EL VALOR DEL PESO 2
                        ;Peso actual = Peso anterior + (Señal Actual - Peso anterior)/2
        movf    CANAL_2,0
        movwf   AUX+1
        movf    PESO_2,0
        movwf   AUX
        subwf   AUX+1,0
        btfss   STATUS,C
        goto    Neg_2
        movwf   AUX+2           ;AUX = (Señal Actual - Peso anterior)
        bcf     AUX+2,0
        bcf     STATUS,C
        rrf     AUX+2,1         ;AUX = (Señal Actual - Peso anterior)/2
        rrf     AUX+2,0         ;AUX = (Señal Actual - Peso anterior)/4
        addwf   PESO_2,1        ;PESO_2  =  Peso actual.
        goto    No_act_2
Neg_2                           ;Si la señal es menor que el valor actual del peso
        movf    AUX+1,0
        subwf   AUX,0
        movwf   AUX+2           ;AUX = (Peso anterior - Señal Actual)
        bcf     AUX+2,0
        bcf     STATUS,C
        rrf     AUX+2,1         ;AUX = (Peso anterior - Señal Actual)/2
        rrf     AUX+2,0         ;AUX = (Señal Actual - Peso anterior)/4
        subwf   PESO_2,1        ;PESO_2  =  Peso actual.
No_act_2
```

```
        movf    PESO_2,0        ;Graba el valor del Peso 2
        movwf   DATO
        movlw   40h
        movwf   DIREC
        call    Grabar
        movf    MEMO_2+2,0      ;Graba valor de la memoria 2 (registro mas significativo)
        movwf   DATO
        movlw   50h
        movwf   DIREC
        call    Grabar
        movf    MEMO_2,0        ;Graba el valor de la memoria 2
        movwf   DATO
        movlw   58h
        movwf   DIREC
        call    Grabar
        return
;=================================================================
```

Código Subrutina Contador

```
;++++++++++++++++++++++++++++++++++++++++++++++++
;Contador
;++++++++++++++++++++++++++++++++++++++++++++++++
Contador                        ;La función Contador permite realizar un muestreo de las
                                ;condiciones de pesos, señales, niveles de activación y
                                ;otros, Variando el máximo valor al que puede llegar el
                                ;registro CONT+1
        btfsc   CONT+1,1
        incf    CONT,1          ;se puede alterar el timepo entre las muestas de las señales.
        btfsc   CONT+1,1        ;El registro CONT ayuda a determinar el lugar de la
                                ;memoria EEPROM
        clrf    CONT+1          ;en el cual se guardará la informacion
        incf    CONT+1,1
```

```
        btfsc   CONT,3
        clrf    CONT
        movf    CONT,0
        movwf   DATO
        movf    CONT,0
        sublw   7Ah
        movwf   DIREC
        call    Grabar
Salir_Cont
        return
;=================================================================
```

Código Subrutinas Suma y Resta

```
;+++++++++++++++++++++++++++++++++++++++++++++++
;Sumar
;+++++++++++++++++++++++++++++++++++++++++++++++
Sumar
        movf    AUX,0           ;Suma la variable existente en AUX y el valor del NIVEL
        addwf   NIVEL,1         ;En caso de desbordamiento se suma 1 a la variable
                                ;NIVEL+1
        btfsc   STATUS,C
        incf    NIVEL+1,1
        return
;=================================================================
;+++++++++++++++++++++++++++++++++++++++++++++++
;Restar
;+++++++++++++++++++++++++++++++++++++++++++++++
Restar
        call    Retardo         ;Permite que una neurona actué sola por un tiempo sin que
        call    Retardo         ; las otras se activen
        movf    AUX,0           ;Resta la variable existente en AUX y el valor del NIVEL
```

```
        subwf   NIVEL,1         ;En caso de desbordamiento se resta 1 a la variable
                                ;NIVEL+1
        btfss  STATUS,C
        decf   NIVEL+1,1
        return
;=================================================================
```

Código Conversión Análoga Digital

```
;+++++++++++++++++++++++++++++++++++++++++++++++
;Conversion A/D
;+++++++++++++++++++++++++++++++++++++++++++++++
Conversion
        movlw   01h
        movwf   ADCON0          ;habilita la lectura del canal 0
        call    Convertir       ;Llama a la función de conversión A/D
        movf    ADRESH,0        ;Obtiene el valor de la conversión A/D
        movwf   CANAL_0         ;Almacena el valor en la variable CANAL_0
        movwf   DATO            ;Graba el dato de la señal de entrada 0
        movlw   08h
        movwf   DIREC
        call    Grabar
;-----------------------------------------------------------------------------------------------
        movlw   05h
        movwf   ADCON0          ;habilita la lectura del canal 1
        call    Convertir       ;Llama a la función de conversión A/D
        movf    ADRESH,0        ;Obtiene el valor de la conversión A/D
        movwf   CANAL_1         ;Almacena el valor en la variable CANAL_1
        movwf   DATO            ;Graba el dato de la señal de entrada 1
        movlw   28h
        movwf   DIREC
        call    Grabar
```

```
;-------------------------------------------------------------------------------------------------
        movlw   09h
        movwf   ADCON0          ;habilita la lectura del canal 2
        call    Convertir       ;Llama a la función de conversión A/D
        movf    ADRESH,0        ;Obtiene el valor de la conversión A/D
        movwf   CANAL_2         ;Almacena el valor en la variable CANAL_2
        movwf   DATO            ;Graba el dato de la señal de entrada 2
        movlw   48h
        movwf   DIREC
        call    Grabar
        return
;================================================================
;++++++++++++++++++++++++++++++++++++++++++++++++
;Convertir Inicia conversion A/D
;++++++++++++++++++++++++++++++++++++++++++++++++
Convertir
        bsf     ADCON0,GO       ;inicia la conversión
Convirtiendo
        btfsc   ADCON0,GO
        goto    Convirtiendo
        return
;================================================================
```

Código Subrutina Grabar Memoria EEPROM

```
;++++++++++++++++++++++++++++++++++++++++++++++++
;Grabar en la memoria EEPROM
;++++++++++++++++++++++++++++++++++++++++++++++++
Grabar
        btfss   CONT+1,1        ;Se graba cuando el registro CONT+1 llega a su máximo
        goto    No_Grabar       ;valor
        movf    CONT,0
```

```
        addwf   DIREC,1
        bsf     STATUS,RP0      ;Banco 1
        movf    DATO,w
        movwf   EEDATA
        movf    DIREC,w
        movwf   EEADR
        bsf     EECON1,WREN     ;Habilita Escritura
        movlw   55h
        movwf   EECON2
        movlw   0AAh
        movwf   EECON2
        bsf     EECON1,WR       ;Comienza la Escritura
Grabando
        btfsc  EECON1,WR
        goto    Grabando
        bcf     STATUS,RP0      ;Vuelve al Banco 0
No_Grabar
        return
;=================================================================
```

Código Subrutina Aprendizaje Positivo

```
;++++++++++++++++++++++++++++++++++++++++++++++++
;APRENDIZAJE POSITIVO
;++++++++++++++++++++++++++++++++++++++++++++++++
Refuerzo_P
        movf    CANAL_0,0       ;Filtra las señales para que solo se produzca aprendizaje en
        sublw   18h             ;la entrada 0 si esta posee una señal considerable.
        btfsc   STATUS,C
        goto    No_ref_0
        movf    CANAL_0,0
        addwf   MEMO_0+1,1      ;Aumenta el tiempo que el peso permanece en la memoria
```

```
        btfsc   STATUS,C
        incf    MEMO_0+2,1
        btfsc   MEMO_0+2,7      ;Si el valor de memoria es mayor a 32512 se memorizará
        decf    MEMO_0+2,1      ; produzcan el peso permanentemente a menos que se
                                ; señales de error esto se hace para que no se produzca
                                ; un desbordamiento y regreso a cero del registro
        movf    PESO_0,0
        subwf   CANAL_0,0
        btfss   STATUS,C
        goto    No_ref_0
        incf    PESO_0,1
No_ref_0
;-------------------------------------------------------------------------------------------------
        movf    CANAL_1,0       ;Filtra las señales para que solo se produzca aprendizaje en
        sublw   18h             ;la entrada 1 si esta posee una señal considerable.
        btfsc   STATUS,C
        goto    No_ref_1
        movf    CANAL_1,0
        movwf   PESO_1
        addwf   MEMO_1+1,1      ;Aumenta el tiempo que el peso permanece en la memoria
        btfsc   STATUS,C
        incf    MEMO_1+2,1
        btfsc   MEMO_1+2,7      ;Si el valor de memoria es mayor a 32512 se memorizará
        decf    MEMO_1+2,1      ;el peso permanentemente a menos que se produzcan
                                ;señales de error esto se hace para que no se produzca un
                                ;desbordamiento y regreso a cero del registro
        movf    PESO_1,0
        subwf   CANAL_1,0
        btfss   STATUS,C
        goto    No_ref_1
        incf    PESO_1,1
No_ref_1
;-------------------------------------------------------------------------------------------------
        movf    CANAL_2,0       ;Filtra las señales para que solo se produzca aprendizaje en
```

```
        sublw   18h             ;la entrada 2 si esta posee una señal considerable.
        btfsc   STATUS,C
        goto    No_ref_2
        movf    CANAL_2,0
        movwf   PESO_2
        addwf   MEMO_2+1,1      ;Aumenta el tiempo que el peso permanece en la memoria
        btfsc   STATUS,C
        incf    MEMO_2+2,1
        btfsc   MEMO_2+2,7      ;Si el valor de memoria es mayor a 32512 se memorizará
        decf    MEMO_2+2,1      ;el peso permanentemente a menos que se produzcan
                                ;señales de error esto se hace para que no se produzca un
                                ;desbordamiento y regreso a cero del registro
        movf    PESO_2,0
        subwf   CANAL_2,0
        btfss   STATUS,C
        goto    No_ref_2
        incf    PESO_2,1
No_ref_2
        return
;===============================================================
```

Código Subrutinas de Retardo

```
;++++++++++++++++++++++++++++++++++++++++++++++++
;RETARDO
;++++++++++++++++++++++++++++++++++++++++++++++++
Retardo
        clrf    AUX+1
        clrf    AUX
Delay
        NOP
        NOP
```

```
        NOP
        NOP
        decfsz  AUX,f
        goto    Delay
        decfsz  AUX+1,f
        goto    Delay
        return
;+++++++++++++++++++++++++++++++++++++++++++++++++
;RETARDO 2
;+++++++++++++++++++++++++++++++++++++++++++++++++
Retardo2
        movlw   30h
        movwf   AUX+1
        clrf    AUX
Delay2
        decfsz  AUX,f
        goto    Delay2
        decfsz  AUX+1,f
        goto    Delay2
        clrf    AUX
        return
;================================================================
```

Código Subrutina Aprendizaje Negativo

```
;+++++++++++++++++++++++++++++++++++++++++++++++++
;APRENDIZAJE NEGATIVO
;+++++++++++++++++++++++++++++++++++++++++++++++++
Refuerzo_N
        bsf     GPIO,GP5        ;Desactiva la salida de la neurona
        movlw   08h             ;Valor en el que serán decrementados los pesos de las
        movwf   AUX             ;conexiones de esta neurona
```

```
Decr
        movf    CANAL_0,0       ;Filtra las señales para que solo se produzca aprendizaje en
        sublw   18h             ;la entrada 0 si esta posee una señal considerable.
        btfsc   STATUS,C
        goto    No_ref_neg_0
        decf    PESO_0,1        ;Decrementa peso 0 hasta un valor mínimo de cero
        btfsc   PESO_0,7
        clrf    PESO_0
No_ref_neg_0
        movf    CANAL_1,0       ;Filtra las señales para que solo se produzca aprendizaje en
        sublw   18h             ;la entrada 1 si esta posee una señal considerable.
        btfsc   STATUS,C
        goto    No_ref_neg_1
        decf    PESO_1,1        ;Decrementa peso 1 hasta un valor mínimo de cero
        btfsc   PESO_1,7
        clrf    PESO_1
No_ref_neg_1
        movf    CANAL_2,0       ;Filtra las señales para que solo se produzca aprendizaje en
        sublw   18h             ;la entrada 2 si esta posee una señal considerable.
        btfsc   STATUS,C
        goto    No_ref_neg_2
        decf    PESO_2,1        ;Decrementa peso 2 hasta un valor mínimo de cero
        btfsc   PESO_2,7
        clrf    PESO_2
No_ref_neg_2
        decfsz  AUX,f
        goto    Decr
No_sal
        Btfsc   GPIO,GP3        ;Comprueba si existe señal de error
        goto    No_sal
        return
;===============================================================
        end
```

ANEXO 2

DATA SHEETS

MICROCONTROLADOR PIC12F675

DRIVER DE MOTOR L293D

SENSOR DE PROXIMIDAD SHARP GP2D120

PIC12F629/675

8-Pin FLASH-Based 8-Bit CMOS Microcontroller

High Performance RISC CPU:

- Only 35 instructions to learn
  - All single cycle instructions except branches
- Operating speed:
  - DC - 20 MHz oscillator/clock input
  - DC - 200 ns instruction cycle
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect, and Relative Addressing modes

Special Microcontroller Features:

- Internal and external oscillator options
  - Precision Internal 4 MHz oscillator factory calibrated to ±1%
  - External Oscillator support for crystals and resonators
  - 5 μs wake-up from SLEEP, 3.0V, typical
- Power saving SLEEP mode
- Wide operating voltage range - 2.0V to 5.5V
- Industrial and Extended temperature range
- Low power Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Detect (BOD)
- Watchdog Timer (WDT) with independent oscillator for reliable operation
- Multiplexed $\overline{\text{MCLR}}$/Input-pin
- Interrupt-on-pin change
- Individual programmable weak pull-ups
- Programmable code protection
- High Endurance FLASH/EEPROM Cell
  - 100,000 write FLASH endurance
  - 1,000,000 write EEPROM endurance
  - FLASH/Data EEPROM Retention: > 40 years

Low Power Features:

- Standby Current:
  - 1 nA @ 2.0V, typical
- Operating Current:
  - 8.5 μA @ 32 kHz, 2.0V, typical
  - 100 μA @ 1 MHz, 2.0V, typical
- Watchdog Timer Current
  - 300 nA @ 2.0V, typical
- Timer1 oscillator current:
  - 4 μA @ 32 kHz, 2.0V, typical

Peripheral Features:

- 6 I/O pins with individual direction control
- High current sink/source for direct LED drive
- Analog comparator module with:
  - One analog comparator
  - Programmable on-chip comparator voltage reference (CV_{REF}) module
  - Programmable input multiplexing from device inputs
  - Comparator output is externally accessible
- Analog-to-Digital Converter module (PIC12F675):
  - 10-bit resolution
  - Programmable 4-channel input
  - Voltage reference input
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Option to use OSC1 and OSC2 in LP mode as Timer1 oscillator, if INTOSC mode selected
- In-Circuit Serial Programming™ (ICSP™) via two pins

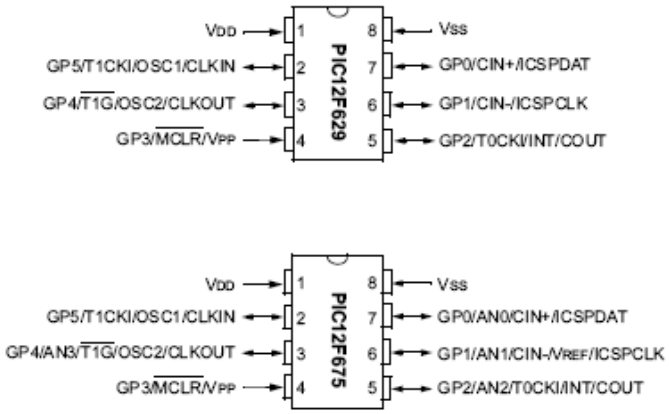
Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	Comparators	Timers 8/16-bit
	FLASH (words)	SRAM (bytes)	EEPROM (bytes)				
PIC12F629	1024	64	128	6	–	1	1/1
PIC12F675	1024	64	128	6	4	1	1/1

* 8-bit, 8-pin devices protected by Microchip's Low Pin Count Patent: U.S. Patent No. 5,847,450. Additional U.S. and foreign patents and applications may be issued or pending.

PIC12F629/675

Pin Diagrams

8-pin PDIP, SOIC, DFN-S



3.0 GPIO PORT

There are as many as six general purpose I/O pins available. Depending on which peripherals are enabled, some or all of the pins may not be available as general purpose I/O. In general, when a peripheral is enabled, the associated pin may not be used as a general purpose I/O pin.

Note:	Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023)

3.1 GPIO and the TRISIO Registers

GPIO is an 6-bit wide, bi-directional port. The corresponding data direction register is TRISIO. Setting a TRISIO bit (= 1) will make the corresponding GPIO pin an input (i.e., put the corresponding output driver in a Hi-impedance mode). Clearing a TRISIO bit (= 0) will make the corresponding GPIO pin an output (i.e., put the contents of the output latch on the selected pin). The exception is GP3, which is input only and its TRISIO bit will always read as '1'. Example 3-1 shows how to initialize GPIO.

Reading the GPIO register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified, and then written to the port data latch. GP3 reads '0' when MCLREN = 1.

The TRISIO register controls the direction of the GP pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISIO register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

Note:	The ANSEL (9Fh) and CMCON (19h) registers (9Fh) must be initialized to configure an analog channel as a digital input. Pins configured as analog inputs will read '0'. The ANSEL register is defined for the PIC12F675.

EXAMPLE 3-1: INITIALIZING GPIO

```
bcf     STATUS,RP0      ;Bank 0
clrf    GPIO            ;Init GPIO
movlw   07h             ;Set GP<2:0> to
movwf   CMCON           ;digital IO
bsf     STATUS,RP0      ;Bank 1
clrf    ANSEL           ;Digital I/O
movlw   0Ch             ;Set GP<3:2> as inputs
movwf   TRISIO          ;and set GP<5:4,1:0>
                        ;as outputs
```

3.2 Additional Pin Functions

Every GPIO pin on the PIC12F629/675 has an interrupt-on-change option and every GPIO pin, except GP3, has a weak pull-up option. The next two sections describe these functions.

3.2.1 WEAK PULL-UP

Each of the GPIO pins, except GP3, has an individually configurable weak internal pull-up. Control bits WPUx enable or disable each pull-up. Refer to Register 3-3. Each weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset by the $\overline{\text{GPPU}}$ bit (OPTION<7>).

REGISTER 3-1: GPIO — GPIO REGISTER (ADDRESS: 05h)

U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
bit 7							bit 0

bit 7-6: **Unimplemented**: Read as '0'

bit 5-0: **GPIO<5:0>**: General Purpose I/O pin.

1 = Port pin is >VIH
0 = Port pin is <VIL

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 3-2: TRISIO — GPIO TRISTATE REGISTER (ADDRESS: 85h)

U-0	U-0	R/W-x	R/W-x	R-1	R/W-x	R/W-x	R/W-x
—	—	TRISIO5	TRISIO4	TRISIO3	TRISIO2	TRISIO1	TRISIO0
bit 7							bit 0

bit 7-6: **Unimplemented**: Read as '0'

bit 5-0: **TRISIO<5:0>**: General Purpose I/O Tri-State Control bit
1 = GPIO pin configured as an input (tri-stated)
0 = GPIO pin configured as an output.
Note: TRISIO<3> always reads 1.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 3-3: WPU — WEAK PULL-UP REGISTER (ADDRESS: 95h)

U-0	U-0	R/W-1	R/W-1	U-0	R/W-1	R/W-1	R/W-1
—	—	WPU5	WPU4	—	WPU2	WPU1	WPU0
bit 7							bit 0

bit 7-6 **Unimplemented**: Read as '0'

bit 5-4 **WPU<5:4>**: Weak Pull-up Register bit
1 = Pull-up enabled
0 = Pull-up disabled

bit 3 **Unimplemented**: Read as '0'

bit 2-0 **WPU<2:0>**: Weak Pull-up Register bit
1 = Pull-up enabled
0 = Pull-up disabled

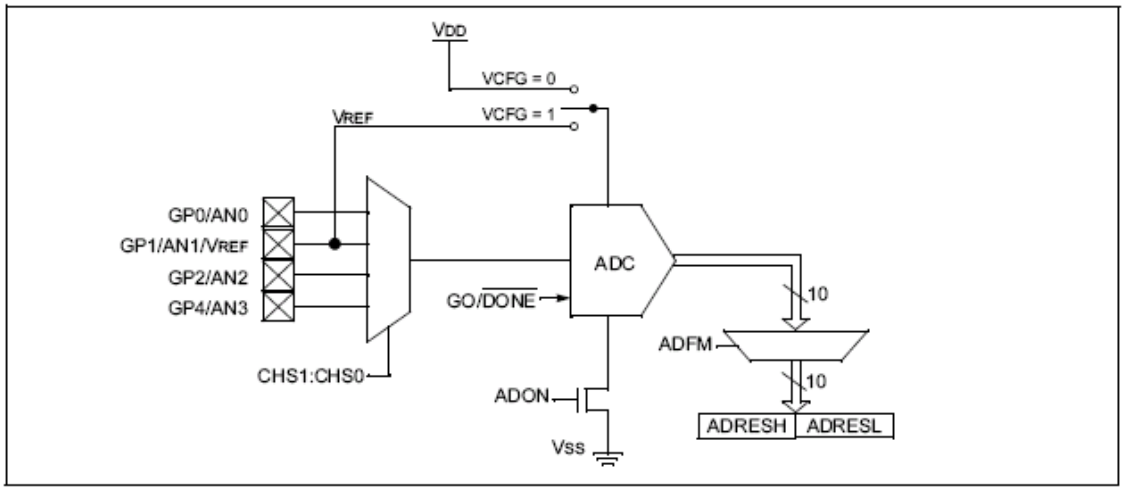
Note 1: Global $\overline{\text{GPPU}}$ must be enabled for individual pull-ups to be enabled.
2: The weak pull-up device is automatically disabled if the pin is in Output mode (TRISIO = 0).

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

7.0 ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE (PIC12F675 ONLY)

The analog-to-digital converter (A/D) allows conversion of an analog input signal to a 10-bit binary representation of that signal. The PIC12F675 has four analog inputs, multiplexed into one sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a binary result via successive approximation and stores the result in a 10-bit register. The voltage reference used in the conversion is software selectable to either VDD or a voltage applied by the VREF pin. Figure 7-1 shows the block diagram of the A/D on the PIC12F675.

FIGURE 7-1: A/D BLOCK DIAGRAM



7.1 A/D Configuration and Operation

There are two registers available to control the functionality of the A/D module:

1. ADCON0 (Register 7-1)
2. ANSEL (Register 7-2)

7.1.1 ANALOG PORT PINS

The ANS3:ANS0 bits (ANSEL<3:0>) and the TRISIO bits control the operation of the A/D port pins. Set the corresponding TRISIO bits to set the pin output driver to its high impedance state. Likewise, set the corresponding ANS bit to disable the digital input buffer.

Note:	Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

7.1.2 CHANNEL SELECTION

There are four analog channels on the PIC12F675, AN0 through AN3. The CHS1:CHS0 bits (ADCON0<3:2>) control which channel is connected to the sample and hold circuit.

7.1.3 VOLTAGE REFERENCE

There are two options for the voltage reference to the A/D converter: either VDD is used, or an analog voltage applied to VREF is used. The VCFG bit (ADCON0<6>) controls the voltage reference selection. If VCFG is set, then the voltage on the VREF pin is the reference; otherwise, VDD is the reference.

7.1.4 CONVERSION CLOCK

The A/D conversion cycle requires 11 TAD. The source of the conversion clock is software selectable via the ADCS bits (ANSEL<6:4>). There are seven possible clock options:

- Fosc/2
- Fosc/4
- Fosc/8
- Fosc/16
- Fosc/32
- Fosc/64
- FRC (dedicated internal RC oscillator)

For correct conversion, the A/D conversion clock (1/TAD) must be selected to ensure a minimum TAD of 1.6 μs. Table 7-1 shows a few TAD calculations for selected frequencies.

TABLE 7-1: TAD vs. DEVICE OPERATING FREQUENCIES

A/D Clock Source (TAD)		Device Frequency			
Operation	ADCS2:ADCS0	20 MHz	5 MHz	4 MHz	1.25 MHz
2 Tosc	000	100 ns(2)	400 ns(2)	500 ns(2)	1.6 µs
4 Tosc	100	200 ns(2)	800 ns(2)	1.0 µs(2)	3.2 µs
8 Tosc	001	400 ns(2)	1.6 µs	2.0 µs	6.4 µs
16 Tosc	101	800 ns(2)	3.2 µs	4.0 µs	12.8 µs(3)
32 Tosc	010	1.6 µs	6.4 µs	8.0 µs(3)	25.6 µs(3)
64 Tosc	110	3.2 µs	12.8 µs(3)	16.0 µs(3)	51.2 µs(3)
A/D RC	x11	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1,4)

Legend: Shaded cells are outside of recommended range.

Note 1: The A/D RC source has a typical TAD time of 4 µs for VDD > 3.0V.

2: These values violate the minimum required TAD time.

3: For faster conversion times, the selection of another clock source is recommended.

4: When the device frequency is greater than 1 MHz, the A/D RC clock source is only recommended if the conversion will be performed during SLEEP.

7.1.5 STARTING A CONVERSION

The A/D conversion is initiated by setting the GO/$\overline{\text{DONE}}$ bit (ADCON0<1>). When the conversion is complete, the A/D module:

- Clears the GO/$\overline{\text{DONE}}$ bit
- Sets the ADIF flag (PIR1<6>)
- Generates an interrupt (if enabled).

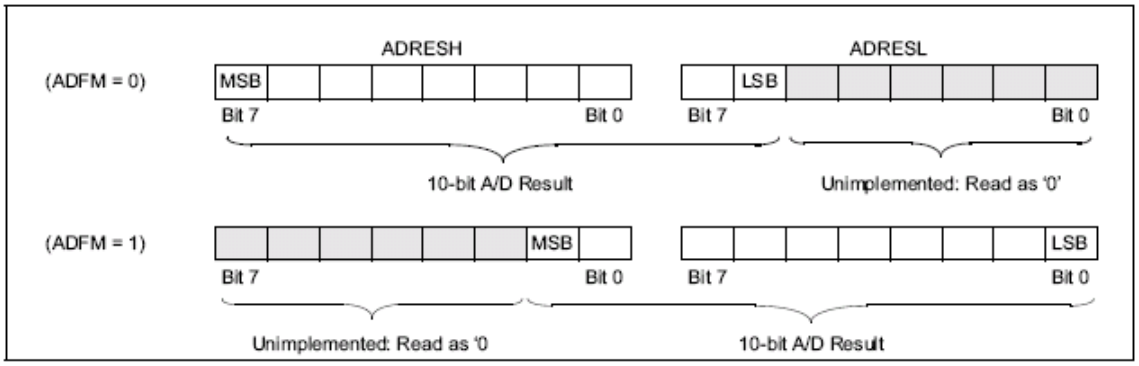
If the conversion must be aborted, the GO/$\overline{\text{DONE}}$ bit can be cleared in software. The ADRESH:ADRESL registers will not be updated with the partially complete A/D conversion sample. Instead, the ADRESH:ADRESL registers will retain the value of the previous conversion. After an aborted conversion, a 2 TAD delay is required before another acquisition can be initiated. Following the delay, an input acquisition is automatically started on the selected channel.

Note:	The GO/$\overline{\text{DONE}}$ bit should not be set in the same instruction that turns on the A/D.

7.1.6 CONVERSION OUTPUT

The A/D conversion can be supplied in two formats: left or right shifted. The ADFM bit (ADCON0<7>) controls the output format. Figure 7-2 shows the output formats.

FIGURE 7-2: 10-BIT A/D RESULT FORMAT



REGISTER 7-1: ADCON0 — A/D CONTROL REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	—	—	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7 **ADFM**: A/D Result Formed Select bit
1 = Right justified
0 = Left justified

bit 6 **VCFG**: Voltage Reference bit
1 = V_{REF} pin
0 = V_{DD}

bit 5-4 **Unimplemented**: Read as zero

bit 3-2 **CHS1:CHS0**: Analog Channel Select bits
00 = Channel 00 (AN0)
01 = Channel 01 (AN1)
10 = Channel 02 (AN2)
11 = Channel 03 (AN3)

bit 1 **GO/$\overline{\text{DONE}}$**: A/D Conversion Status bit
1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.
This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress

bit 0 **ADON**: A/D Conversion STATUS bit
1 = A/D converter module is operating
0 = A/D converter is shut-off and consumes no operating current

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 7-2: ANSEL — ANALOG SELECT REGISTER (ADDRESS: 9Fh)

U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1
—	ADCS2	ADCS1	ADCS0	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0

bit 7 **Unimplemented**: Read as '0'.

bit 6-4 **ADCS<2:0>**: A/D Conversion Clock Select bits

000 = Fosc/2
001 = Fosc/8
010 = Fosc/32
x11 = FRC (clock derived from a dedicated internal oscillator = 500 kHz max)
100 = Fosc/4
101 = Fosc/16
110 = Fosc/64

bit 3-0 **ANS3:ANS0**: Analog Select bits
(Between analog or digital function on pins AN<3:0>, respectively.)
1 = Analog input; pin is assigned as analog input(1)
0 = Digital I/O; pin is assigned to port or special function

Note 1: Setting a pin to an analog input automatically disables the digital input circuitry, weak pull-ups, and interrupt-on-change. The corresponding TRISIO bit must be set to Input mode in order to allow external control of the voltage on the pin.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

7.2 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 7-3. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), see Figure 7-3. **The maximum recommended impedance for analog sources is 10 kΩ.** As the impedance is decreased, the acquisition time may be decreased.

After the analog input channel is selected (changed), this acquisition must be done before the conversion can be started.

To calculate the minimum acquisition time, Equation 7-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

To calculate the minimum acquisition time, TACQ, see the PICmicro™ Mid-Range Reference Manual (DS33023).

EQUATION 7-1: ACQUISITION TIME

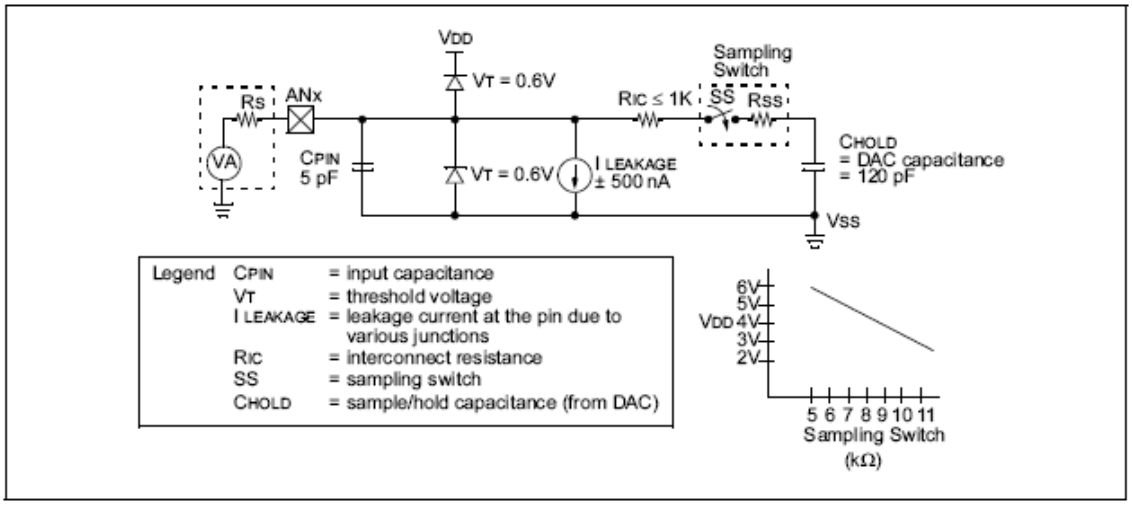
$$
\begin{aligned}
T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\
&= T_{AMP} + T_C + T_{COFF} \\
&= 2\mu s + T_C + [(\text{Temperature} - 25^\circ C)(0.05\mu s/^\circ C)] \\
T_C &= C_{HOLD}(R_{IC} + R_{SS} + R_S)\ \ln(1/2047) \\
&= -120pF(1k\Omega + 7k\Omega + 10k\Omega)\ \ln(0.0004885) \\
&= 16.47\mu s \\
T_{ACQ} &= 2\mu s + 16.47\mu s + [(50^\circ C - 25^\circ C)(0.05\mu s/^\circ C) \\
&= 19.72\mu s
\end{aligned}
$$

Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

FIGURE 7-3: ANALOG INPUT MODEL



8.0 DATA EEPROM MEMORY

The EEPROM data memory is readable and writable during normal operation (full VDD range). This memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers. There are four SFRs used to read and write this memory:

- EECON1
- EECON2 (not a physically implemented register)
- EEDATA
- EEADR

EEDATA holds the 8-bit data for read/write, and EEADR holds the address of the EEPROM location being accessed. PIC12F629/675 devices have 128 bytes of data EEPROM with an address range from 0h to 7Fh.

The EEPROM data memory allows byte read and write. A byte write automatically erases the location and writes the new data (erase before write). The EEPROM data memory is rated for high erase/write cycles. The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature as well as from chip to chip. Please refer to AC Specifications for exact limits.

When the data memory is code protected, the CPU may continue to read and write the data EEPROM memory. The device programmer can no longer access this memory.

Additional information on the Data EEPROM is available in the PICmicro™ Mid-Range Reference Manual, (DS33023).

REGISTER 8-1: EEDAT — EEPROM DATA REGISTER (ADDRESS: 9Ah)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
bit 7							bit 0

bit 7-0 **EEDATn**: Byte value to write to or read from Data EEPROM

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 8-2: EEADR — EEPROM ADDRESS REGISTER (ADDRESS: 9Bh)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	EADR6	EADR5	EADR4	EADR3	EADR2	EADR1	EADR0
bit 7							bit 0

bit 7 **Unimplemented**: Should be set to '0'

bit 6-0 **EEADR**: Specifies one of 128 locations for EEPROM Read/Write Operation

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

8.1 EEADR

The EEADR register can address up to a maximum of 128 bytes of data EEPROM. Only seven of the eight bits in the register (EEADR<6:0>) are required. The MSb (bit 7) is ignored.

The upper bit should always be '0' to remain upward compatible with devices that have more data EEPROM memory.

8.2 EECON1 AND EECON2 REGISTERS

EECON1 is the control register with four low order bits physically implemented. The upper four bits are non-implemented and read as '0's.

Control bits RD and WR initiate read and write, respectively. These bits cannot be cleared, only set, in software. They are cleared in hardware at completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental, premature termination of a write operation.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ Reset, or a WDT Time-out Reset during normal operation. In these situations, following RESET, the user can check the WRERR bit, clear it, and rewrite the location. The data and address will be cleared, therefore, the EEDATA and EEADR registers will need to be re-initialized.

Interrupt flag bit EEIF in the PIR1 register is set when write is complete. This bit must be cleared in software.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the Data EEPROM write sequence.

REGISTER 8-3: EECON1 — EEPROM CONTROL REGISTER (ADDRESS: 9Ch)

U-0	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
—	—	—	—	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7-4 **Unimplemented**: Read as '0'

bit 3 **WRERR**: EEPROM Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{\text{MCLR}}$ Reset, any WDT Reset during normal operation or BOD detect)
0 = The write operation completed

bit 2 **WREN**: EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the data EEPROM

bit 1 **WR**: Write Control bit
1 = Initiates a write cycle (The bit is cleared by hardware once write is complete. The WR bit can only be set, not cleared, in software.)
0 = Write cycle to the data EEPROM is complete

bit 0 **RD**: Read Control bit
1 = Initiates an EEPROM read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set, not cleared, in software.)
0 = Does not initiate an EEPROM read

Legend:			
S = Bit can only be set			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

8.3 READING THE EEPROM DATA MEMORY

To read a data memory location, the user must write the address to the EEADR register and then set control bit RD (EECON1<0>), as shown in Example 8-1. The data is available, in the very next cycle, in the EEDATA register. Therefore, it can be read in the next instruction. EEDATA holds this value until another read, or until it is written to by the user (during a write operation).

EXAMPLE 8-1: DATA EEPROM READ

```
bsf    STATUS,RP0     ;Bank 1
movlw  CONFIG_ADDR    ;
movwf  EEADR          ;Address to read
bsf    EECON1,RD      ;EE Read
movf   EEDATA,W       ;Move data to W
```

8.4 WRITING TO THE EEPROM DATA MEMORY

To write an EEPROM data location, the user must first write the address to the EEADR register and the data to the EEDATA register. Then the user must follow a specific sequence to initiate the write for each byte, as shown in Example 8-2.

EXAMPLE 8-2: DATA EEPROM WRITE

```
bsf    STATUS,RP0     ;Bank 1
bsf    EECON1,WREN    ;Enable write
bcf    INTCON,GIE     ;Disable INTs
movlw  55h            ;Unlock write
movwf  EECON2         ;
movlw  AAh            ;
movwf  EECON2         ;
bsf    EECON1,WR      ;Start the write
bsf    INTCON,GIE     ;Enable INTS
```

Required Sequence

The write will not initiate if the above sequence is not exactly followed (write 55h to EECON2, write AAh to EECON2, then set WR bit) for each byte. We strongly recommend that interrupts be disabled during this code segment. A cycle count is executed during the required sequence. Any number that is not equal to the required cycles to execute the required sequence will prevent the data from being written into the EEPROM.

Additionally, the WREN bit in EECON1 must be set to enable write. This mechanism prevents accidental writes to data EEPROM due to errant (unexpected) code execution (i.e., lost programs). The user should keep the WREN bit clear at all times, except when updating EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. The WR bit will be inhibited from being set unless the WREN bit is set.

At the completion of the write cycle, the WR bit is cleared in hardware and the EE Write Complete Interrupt Flag bit (EEIF) is set. The user can either enable this interrupt or poll this bit. The EEIF bit (PIR<7>) register must be cleared by software.

8.5 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the Data EEPROM should be verified (see Example 8-3) to the desired value to be written.

EXAMPLE 8-3: WRITE VERIFY

```
bcf    STATUS,RP0     ;Bank 0
:                     ;Any code
bsf    STATUS,RP0     ;Bank 1 READ
movf   EEDATA,W       ;EEDATA not changed
                      ;from previous write
bsf    EECON1,RD      ;YES, Read the
                      ;value written
xorwf  EEDATA,W
btfss  STATUS,Z       ;Is data the same
goto   WRITE_ERR      ;No, handle error
:                     ;Yes, continue
```

8.5.1 USING THE DATA EEPROM

The Data EEPROM is a high-endurance, byte addressable array that has been optimized for the storage of frequently changing information (e.g., program variables or other data that are updated often). Frequently changing values will typically be updated more often than specifications D120 or D120A. If this is not the case, an array refresh must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in FLASH program memory.

8.6 PROTECTION AGAINST SPURIOUS WRITE

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built in. On power-up, WREN is cleared. Also, the Power-up Timer (72 ms duration) prevents EEPROM write.

The write initiate sequence and the WREN bit together help prevent an accidental write during:

- brown-out
- power glitch
- software malfunction

10.0 INSTRUCTION SET SUMMARY

The PIC12F629/675 instruction set is highly orthogonal and is comprised of three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

Each PIC12F629/675 instruction is a 14-bit word divided into an **opcode**, which specifies the instruction type, and one or more **operands**, which further specify the operation of the instruction. The formats for each of the categories is presented in Figure 10-1, while the various opcode fields are summarized in Table 10-1.

Table 10-2 lists the instructions recognized by the MPASM™ assembler. A complete description of each instruction is also available in the PICmicro™ Mid-Range Reference Manual (DS33023).

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator, which selects the bit affected by the operation, while 'f' represents the address of the file in which the bit is located.

For **literal and control** operations, 'k' represents an 8-bit or 11-bit constant, or literal value.

One instruction cycle consists of four oscillator periods; for an oscillator frequency of 4 MHz, this gives a normal instruction execution time of 1 µs. All instructions are executed within a single instruction cycle, unless a conditional test is true, or the program counter is changed as a result of an instruction. When this occurs, the execution takes two instruction cycles, with the second cycle executed as a NOP.

Note:	To maintain upward compatibility with future products, do not use the OPTION and TRIS IO instructions.

All instruction examples use the format '0xhh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

10.1 READ-MODIFY-WRITE OPERATIONS

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the instruction, or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

For example, a CLRF GPIO instruction will read GPIO, clear all the data bits, then write the result back to GPIO. This example would have the unintended result that the condition that sets the GPIF flag would be cleared.

TABLE 10-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

FIGURE 10-1: GENERAL FORMAT FOR INSTRUCTIONS

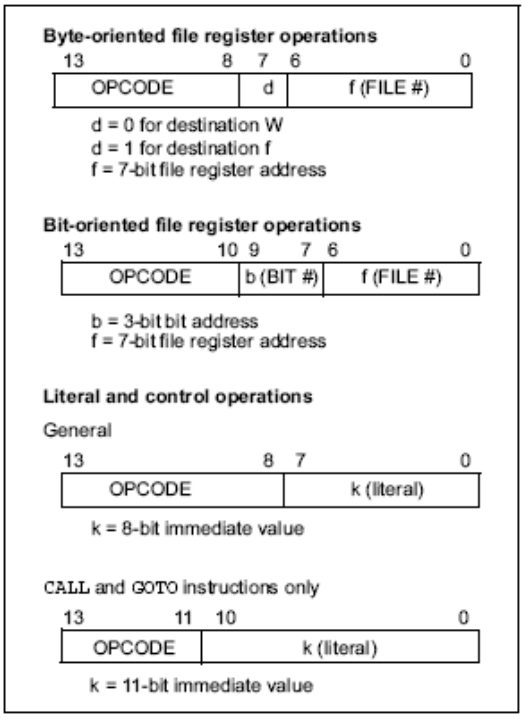


TABLE 10-2: PIC12F629/675 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF GPIO, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

Note: Additional information on the mid-range instruction set is available in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

10.2 Instruction Descriptions

ADDLW	Add Literal and W
Syntax:	[*label*] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the eight-bit literal 'k' and the result is placed in the W register.

ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

ANDLW	AND Literal with W
Syntax:	[*label*] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.

ANDWF	AND W with f
Syntax:	[*label*] ANDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

BCF	Bit Clear f
Syntax:	[*label*] BCF f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	0 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is cleared.

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	1 → (f<b>)
Status Affected:	None
Description:	Bit 'b' in register 'f' is set.

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	$0 \le f \le 127$ $0 \le b < 7$
Operation:	skip if (f<b>) = 1
Status Affected:	None
Description:	If bit 'b' in register 'f' is '0', the next instruction is executed. If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.

BTFSC	Bit Test, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	$0 \le f \le 127$ $0 \le b \le 7$
Operation:	skip if (f<b>) = 0
Status Affected:	None
Description:	If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.

CALL	Call Subroutine
Syntax:	[*label*] CALL k
Operands:	0 ≤ k ≤ 2047
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Description:	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

CLRWDT	Clear Watchdog Timer
Syntax:	[*label*] CLRWDT
Operands:	None
Operation:	00h → WDT 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. STATUS bits $\overline{TO}$ and $\overline{PD}$ are set.

CLRF	Clear f
Syntax:	[*label*] CLRF f
Operands:	0 ≤ f ≤ 127
Operation:	00h → (f) 1 → Z
Status Affected:	Z
Description:	The contents of register 'f' are cleared and the Z bit is set.

COMF	Complement f
Syntax:	[*label*] COMF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	($\bar{f}$) → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'.

CLRW	Clear W
Syntax:	[*label*] CLRW
Operands:	None
Operation:	00h → (W) 1 → Z
Status Affected:	Z
Description:	W register is cleared. Zero bit (Z) is set.

DECF	Decrement f
Syntax:	[*label*] DECF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination)
Status Affected:	Z
Description:	Decrement register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

DECFSZ	Decrement f, Skip if 0
Syntax:	[*label*] DECFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) - 1 → (destination); skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, then a NOP is executed instead, making it a 2TCY instruction.

GOTO	Unconditional Branch
Syntax:	[*label*] GOTO k
Operands:	0 ≤ k ≤ 2047
Operation:	k → PC<10:0> PCLATH<4:3> → PC<12:11>
Status Affected:	None
Description:	GOTO is an unconditional branch. The eleven-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two-cycle instruction.

INCF	Increment f
Syntax:	[*label*] INCF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination)
Status Affected:	Z
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

INCFSZ	Increment f, Skip if 0
Syntax:	[*label*] INCFSZ f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) + 1 → (destination), skip if result = 0
Status Affected:	None
Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead, making it a 2TCY instruction.

IORLW	Inclusive OR Literal with W
Syntax:	[*label*] IORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .OR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are OR'ed with the eight-bit literal 'k'. The result is placed in the W register.

IORWF	Inclusive OR W with f
Syntax:	[*label*] IORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .OR. (f) → (destination)
Status Affected:	Z
Description:	Inclusive OR the W register with register 'f'. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.

MOVF	Move f
Syntax:	[*label*] MOVF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) → (destination)
Status Affected:	Z
Description:	The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register, since status flag Z is affected.

MOVLW	Move Literal to W
Syntax:	[*label*] MOVLW k
Operands:	$0 \le k \le 255$
Operation:	k → (W)
Status Affected:	None
Description:	The eight-bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

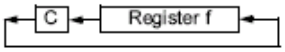
MOVWF	Move W to f
Syntax:	[*label*] MOVWF f
Operands:	$0 \le f \le 127$
Operation:	(W) → (f)
Status Affected:	None
Description:	Move data from W register to register 'f'.

NOP	No Operation
Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.

RETFIE	Return from Interrupt
Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None

RETLW	Return with Literal in W
Syntax:	[*label*] RETLW k
Operands:	$0 \le k \le 255$
Operation:	k → (W); TOS → PC
Status Affected:	None
Description:	The W register is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

RLF	Rotate Left f through Carry
Syntax:	[*label*] RLF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'.

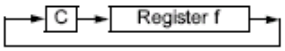


SLEEP	
Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT, 0 → $\overline{\text{WDT}}$ prescaler, 1 → $\overline{\text{TO}}$, 0 → $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Description:	The power-down STATUS bit, $\overline{\text{PD}}$ is cleared. Time-out STATUS bit, $\overline{\text{TO}}$ is set. Watchdog Timer and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped.

RETURN	Return from Subroutine
Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

SUBLW	Subtract W from Literal
Syntax:	[*label*] SUBLW k
Operands:	$0 \le k \le 255$
Operation:	k - (W) → (W)
Status Affected:	C, DC, Z
Description:	The W register is subtracted (2's complement method) from the eight-bit literal 'k'. The result is placed in the W register.

RRF	Rotate Right f through Carry
Syntax:	[*label*] RRF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'.



SUBWF	Subtract W from f
Syntax:	[*label*] SUBWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - (W) → (destination)
Status Affected:	C, DC, Z
Description:	Subtract (2's complement method) W register from register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

SWAPF	Swap Nibbles in f
Syntax:	[*label*] SWAPF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<3:0>) → (destination<7:4>), (f<7:4>) → (destination<3:0>)
Status Affected:	None
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed in register 'f'.

XORLW	Exclusive OR Literal with W
Syntax:	[*label*] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Description:	The contents of the W register are XOR'ed with the eight-bit literal 'k'. The result is placed in the W register.

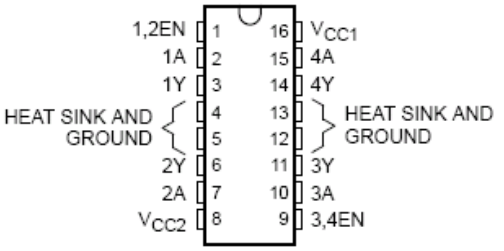
XORWF	Exclusive OR W with f
Syntax:	[*label*] XORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .XOR. (f) → (destination)
Status Affected:	Z
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'.

L293, L293D
QUADRUPLE HALF-H DRIVERS

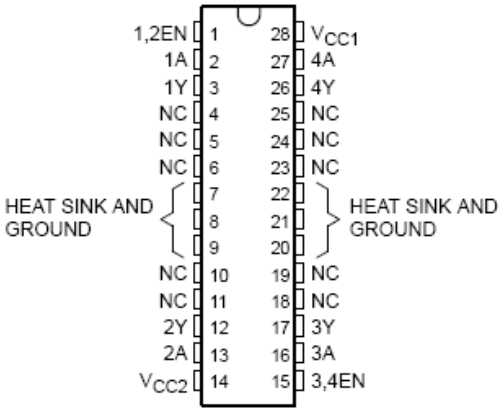
SLRS008B – SEPTEMBER 1986 – REVISED JUNE 2002

- Featuring Unitrode L293 and L293D Products Now From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functional Replacements for SGS L293 and SGS L293D
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

N, NE PACKAGE
(TOP VIEW)



DWP PACKAGE
(TOP VIEW)



description

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

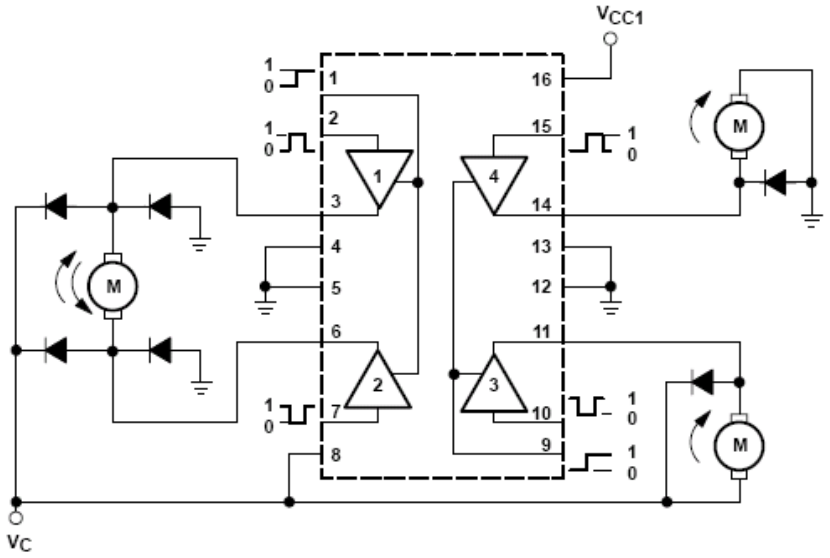
All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression.

A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation.

The L293and L293D are characterized for operation from 0°C to 70°C.

block diagram



NOTE: Output diodes are internal in L293D.

TEXAS INSTRUMENTS
AVAILABLE OPTIONS

T_A	PACKAGE
	PLASTIC DIP (NE)
0°C to 70°C	L293NE L293DNE

Unitrode Products from Texas Instruments

AVAILABLE OPTIONS

T_A	PACKAGED DEVICES	
	SMALL OUTLINE (DWP)	PLASTIC DIP (N)
0°C to 70°C	L293DWP L293DDWP	L293N L293DN

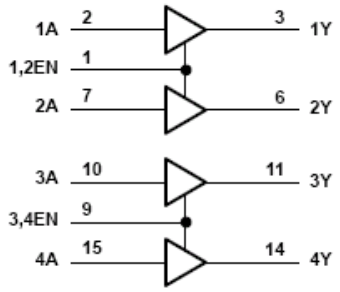
The DWP package is available taped and reeled. Add the suffix TR to device type (e.g., L293DWPTR).

FUNCTION TABLE
(each driver)

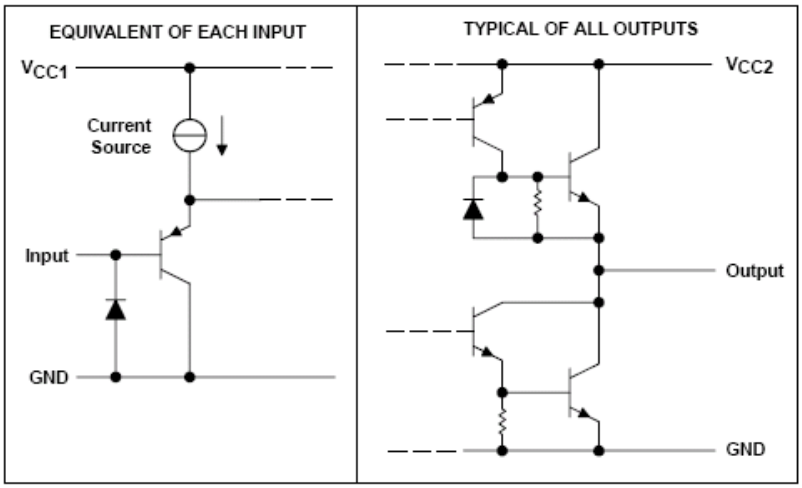
INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)
† In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.

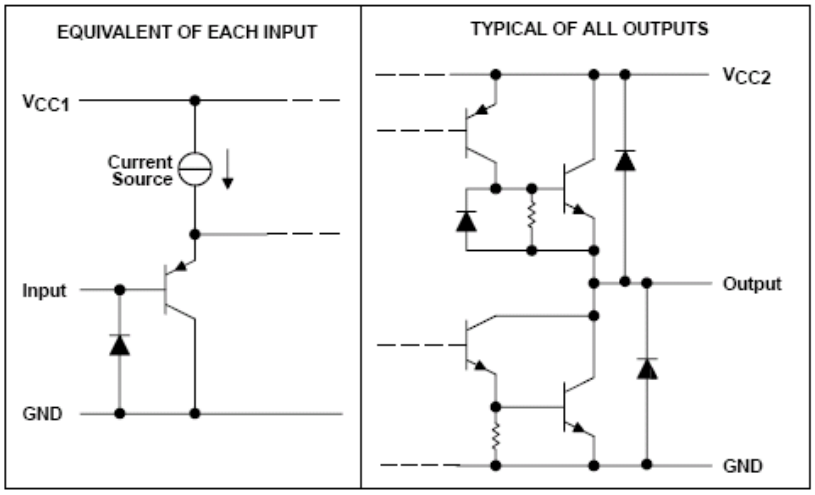
logic diagram



schematics of inputs and outputs (L293)



schematics of inputs and outputs (L293D)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{CC1} (see Note 1)	36 V
Output supply voltage, V_{CC2}	36 V
Input voltage, V_I	7 V
Output voltage range, V_O	–3 V to V_{CC2} + 3 V
Peak output current, I_O (nonrepetitive, t ≤ 5 ms): L293	±2 A
Peak output current, I_O (nonrepetitive, t ≤ 100 µs): L293D	±1.2 A
Continuous output current, I_O: L293	±1 A
Continuous output current, I_O: L293D	±600 mA
Continuous total dissipation at (or below) 25°C free-air temperature (see Notes 2 and 3)	2075 mW
Continuous total dissipation at 80°C case temperature (see Note 3)	5000 mW
Maximum junction temperature, T_J	150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	260°C
Storage temperature range, T_{stg}	–65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltage values are with respect to the network ground terminal.
2. For operation above 25°C free-air temperature, derate linearly at the rate of 16.6 mW/°C.
3. For operation above 25°C case temperature, derate linearly at the rate of 71.4 mW/°C. Due to variations in individual device electrical characteristics and thermal resistance, the built-in thermal overload protection may be activated at power levels slightly above or below the rated dissipation.

recommended operating conditions

			MIN	MAX	UNIT
	Supply voltage	V_{CC1}	4.5	7	V
		V_{CC2}	V_{CC1}	36	
V_{IH}	High-level input voltage	$V_{CC1} \le 7$ V	2.3	V_{CC1}	V
		$V_{CC1} \ge 7$ V	2.3	7	V
V_{IL}	Low-level output voltage		–0.3†	1.5	V
T_A	Operating free-air temperature		0	70	°C

† The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

electrical characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		L293: I_{OH} = –1 A L293D: I_{OH} = –0.6 A		V_{CC2}–1.8	V_{CC2}–1.4		V
V_{OL}	Low-level output voltage		L293: I_{OL} = 1 A L293D: I_{OL} = 0.6 A			1.2	1.8	V
V_{OKH}	High-level output clamp voltage		L293D: I_{OK} = –0.6 A			V_{CC2} + 1.3		V
V_{OKL}	Low-level output clamp voltage		L293D: I_{OK} = 0.6 A			1.3		V
I_{IH}	High-level input current	A	V_I = 7 V			0.2	100	µA
		EN				0.2	10	
I_{IL}	Low-level input current	A	V_I = 0			–3	–10	µA
		EN				–2	–100	
I_{CC1}	Logic supply current		I_O = 0	All outputs at high level		13	22	mA
				All outputs at low level		35	60	
				All outputs at high impedance		8	24	
I_{CC2}	Output supply current		I_O = 0	All outputs at high level		14	24	mA
				All outputs at low level		2	6	
				All outputs at high impedance		2	4	

switching characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	L293NE, L293DNE			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low-to-high-level output from A input	C_L = 30 pF, See Figure 1		800		ns
t_{PHL}	Propagation delay time, high-to-low-level output from A input			400		ns
t_{TLH}	Transition time, low-to-high-level output			300		ns
t_{THL}	Transition time, high-to-low-level output			300		ns

switching characteristics, V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	L293DWP, L293N L293DDWP, L293DN			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low-to-high-level output from A input	C_L = 30 pF, See Figure 1		750		ns
t_{PHL}	Propagation delay time, high-to-low-level output from A input			200		ns
t_{TLH}	Transition time, low-to-high-level output			100		ns
t_{THL}	Transition time, high-to-low-level output			350		ns

GP2D12/GP2D15

General Purpose Type Distance Measuring Sensors

■ Features

1. Less influence on the color of reflective objects, reflectivity
2. Line-up of distance output/distance judgement type
   Distance output type (analog voltage) : **GP2D12**
   Detecting distance : 10 to 80cm
   Distance judgement type : **GP2D15**
   Judgement distance : 24cm
   (Adjustable within the range of 10 to 80cm)
3. External control circuit is unnecessary
4. Low cost

■ Applications

1. TVs
2. Personal computers
3. Cars
4. Copiers

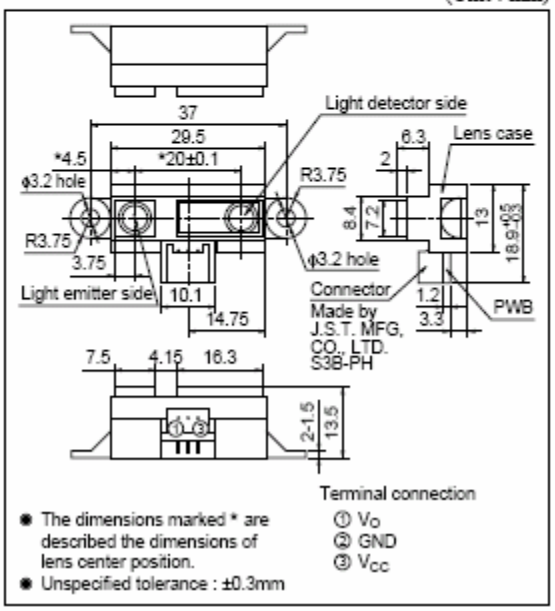
■ Absolute Maximum Ratings

(Ta=25°C, Vcc=5V)

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	−0.3 to +7	V
Output terminal voltage	V_O	−0.3 to Vcc +0.3	V
Operating temperature	T_{opr}	−10 to +60	°C
Storage temperature	T_{stg}	−40 to +70	°C

■ Outline Dimensions

(Unit : mm)



■ Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
Operating supply voltage	V_{CC}	4.5 to +5.5	V

■ Electro-optical Characteristics

(Ta=25°C, V_{CC}=5V)

Parameter		Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range		ΔL	*1 *3	10	–	80	cm
Output terminal voltage	GP2D12	V_O	L=80cm *1	0.25	0.4	0.55	V
	GP2D15	V_{OH}	Output voltage at High *1	Vcc –0.3	–	–	V
		V_{OL}	Output voltage at Low *1	–	–	0.6	V
Difference of output voltage	GP2D12	ΔVo	Output change at L=80cm to 10cm *1	1.75	2.0	2.25	V
Distance characteristics of output	GP2D15	V_O	*1 *2 *4	21	24	27	cm
Average Dissipation current		I_{CC}	L=80cm *1	–	33	50	mA

Note) L : Distance to reflective object.

*1 Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 · white face, reflective ratio ; 90%).
*2 We ship the device after the following adjustment : Output switching distance L=24cm±3cm must be measured by the sensor.
*3 Distance measuring range of the optical sensor system.
*4 Output switching has a hysteresis width. The distance specified by Vo should be the one with which the output L switches to the output H.

Fig.1 Internal Block Diagram

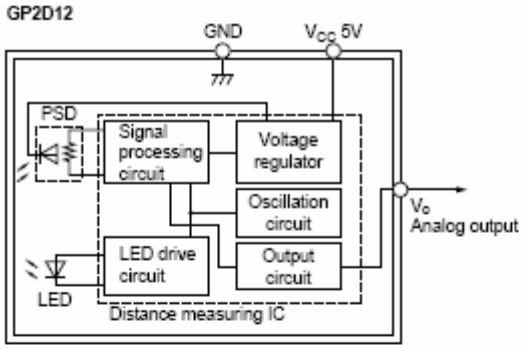


Fig.2 Internal Block Diagram

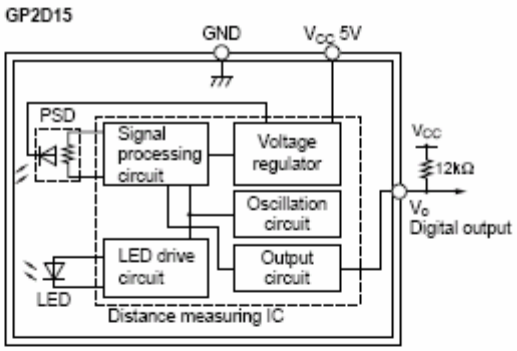


Fig.3 Timing Chart

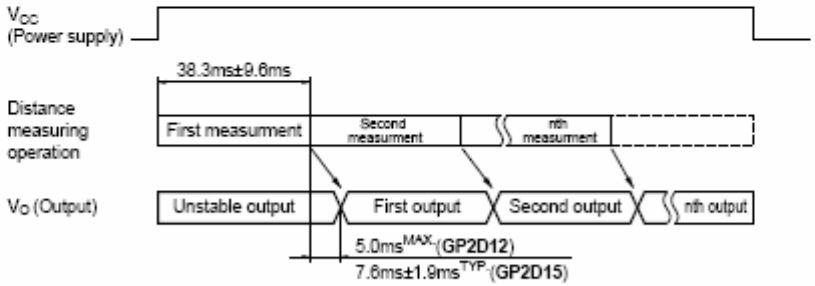


Fig.4 Distance Characteristics

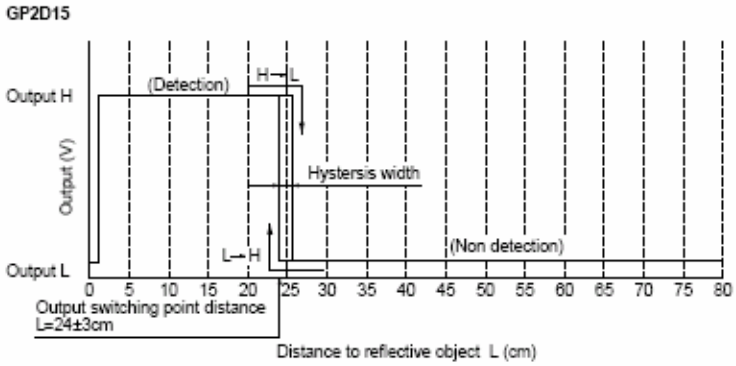


Fig.5 Analog Output Voltage vs. Surface Illuminance of Reflective Object

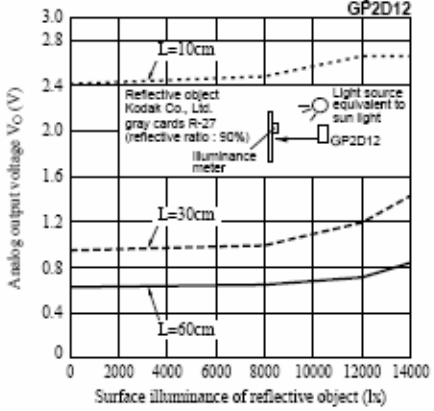


Fig.6 Analog Output Voltage vs.Distance to Reflective Object

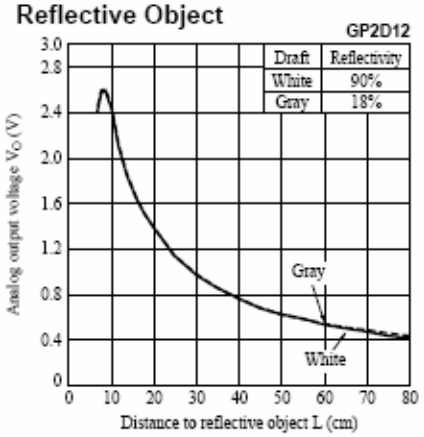


Fig.7 Analog Output Voltage vs.Ambient Temperature

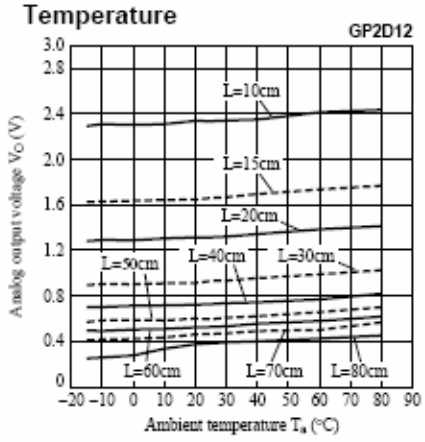
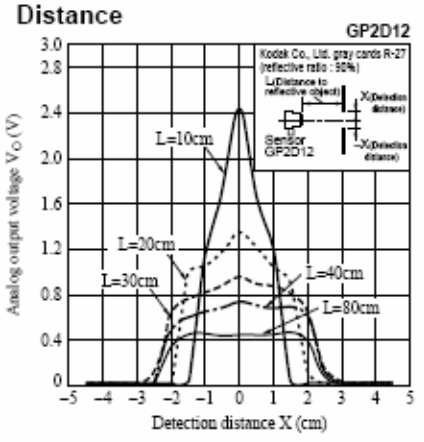


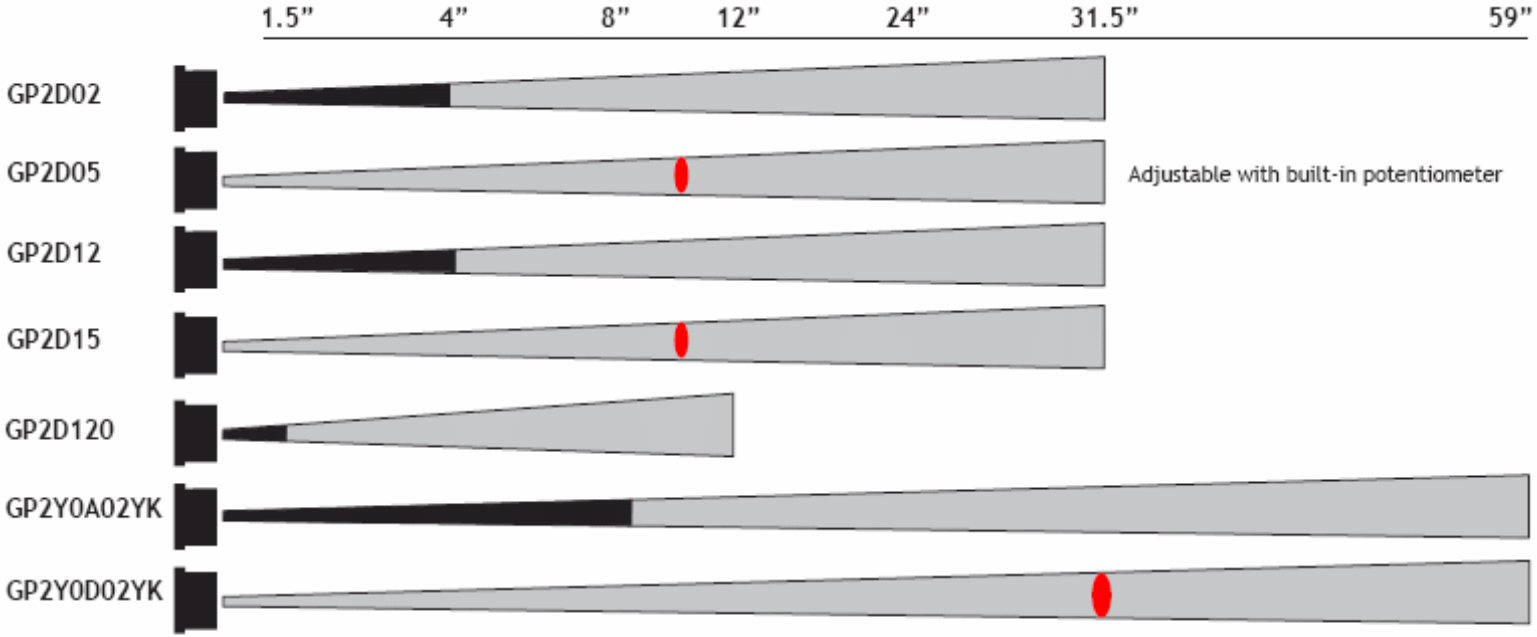
Fig.8 Analog Output Voltage vs.Detection Distance



Acroname
easier robotics

Acroname Quick Comparison Chart for Sharp IR Rangers

Ranger	Output	Minimum Distance	Maximum Distance	On Current	Off Current
GP2D02	Serial	10cm	80cm	~22mA	~3mA
GP2D05	Digital	-	fixed at 24cm	~10mA	~3mA
GP2D12	Analog	10cm	80cm	~33mA	always on
GP2D15	Digital	-	fixed at 24cm±3cm	~33mA	always on
GP2D120	Analog	4cm	30cm	~33mA	always on
GP2Y0A02YK	Analog	20cm	150cm	~33mA	always on
GP2Y0D02YK	Digital	-	fixed at 80cm	~33mA	always on



The end of the black notes the minimum detectable range. The end of the grey notes the maximum range. The red oval notes a fixed range. Objects closer than the noted minimum range will give incorrect readings. For more information please see the product page for each sensor, and checkout *Demystifying the Sharp IR Detectors* in our articles section.

Minimum Range Maximum Range Fixed Range

ÍNDICE DE TABLAS

Tabla 5.1. Datos de los sensores de proximidad…… 58

ÍNDICE DE FIGURAS

Figura 2.1. Neurona Biológica 7
Figura 2.2. Modelo Matemático Aproximado 8
Figura 2.3. Red Neuronal con varias capas 11
Figura 2.4. Funciones de Activación 12
Figura 2.5. Perceptrón 14
Figura 2.6. Modelo Kohonen 16
Figura 2.7. Función de influencia entre neuronas de salida 17
Figura 2.8. Red Multicapa 19
Figura 2.9. Método Algoritmos Genéticos 20
Figura 2.10. Modelo Hopfield 21
Figura 3.1. Red Neuronal Propuesta 1 27
Figura 3.2. Red Neuronal Propuesta 2 28
Figura 4.1. Diagrama de conexión de elementos 35
Figura 4.2. Diagrama de flujo programa principal 41
Figura 4.3. Diagrama de flujo función Comprueba 42
Figura 4.4. Diagrama de flujo función Actualización de pesos 44
Figura 4.5. Registros de memoria 45
Figura 4.6. Distribución de memoria 46
Figura 4.7. Diagrama de flujo función Contador 47
Figura 4.8. Diagramas de flujo funciones Suma y Resta 48
Figura 4.9. Diagrama de flujo función Conversión 49
Figura 4.10. Diagrama de flujo función Convertir 50
Figura 4.11. Diagrama de flujo función Grabar 50
Figura 4.12. Diagrama de flujo función Refuerzo Positivo 51
Figura 4.13. Diagramas de flujo funciones de Retardo 52
Figura 4.14. Diagrama de flujo función Refuerzo Negativo 53

Figura 5.1. Ruedas Laterales ……..…… 55
Figura 5.2. Rueda Pivotal ……....…… 56
Figura 5.3. Motores……..…… 57
Figura 5.4. Baterías ……...……...…… 58
Figura 5.5. Sensores de proximidad ……..…… 59
Figura 5.6. Ecuación Aproximada de respuesta de los sensores ……...…… 60
Figura 5.7. Base Vista 2D …….……...…… 61
Figura 5.8. Base Vista 3D ……...……...…….…… 62
Figura 5.9. Plataforma Completa Vista Lateral ……..…….…… 63
Figura 5.10. Plataforma Completa Vista Frontal ……...…… 64
Figura 5.11. Plataforma Completa Vista Superior ……...…… 64
Figura 5.12. D Plataforma Completa Vista 3D ……...…….…… 65
Figura 6.1. Valores memoria EEPROM neurona 1 …….....…… 67
Figura 6.2. Valores memoria EEPROM neurona 2 ……...……...…...…… 70

GLOSARIO

RNA	Red Neuronal Artificial
Algoritmo	Conjunto de instrucciones que sigue el procesador
Microcontrolador	Dispositivo que integra procesador, memoria y puertos análogos y digitales en un solo elemento.
Memoria EEPROM	Memoria que puede ser escrita y borrada eléctricamente por el microcontrolador.
Sinapsis	Conexión entre dos neuronas
Cromosoma	Parte de la célula que contiene la información genética.
Información Genética	Información que describe todos los rasgos de un físicos de un ser.
Driver de Motor	Dispositivo electrónico encargado de alimentar con energía a los motores, se activa con señales digitales.
Registros	Lugares definidos en la memoria en los que se almacenan los datos.
Subrutinas	Fragmentos de programa que realizan funciones especiales dentro de la programación.

Sangolquí, Agosto de 2005

ELABORADO POR:

_____
Sr. Gustavo Adolfo Moreno Jiménez

AUTORIDADES:

_____
Tcrn.. Ing. Marcelo Gómez Cobos
Decano de la Facultad de Ingeniería Electrónica

_____
Sr. Dr. Jorge Carvajal
Secretario Académico de la Facultad de Ingeniería Electrónica