ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA DE ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELECTRÓNICO, AUTOMATIZACIÓN Y CONTROL

TEMA: DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL INTELIGENTE PARA UN ROBOT PHOENIX TIPO HEXÁPODO MEDIANTE LA TARJETA STM32F4 DISCOVERY Y SIMULINK DE MATLAB

AUTORES:

ANDINO ALBERCA, CRISTIAN ADRIÁN

RODRÍGUEZ SÁNCHEZ, DIEGO XAVIER

DIRECTOR: ING. PROAÑO, VICTOR MSc.

SANGOLQUÍ

2016

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, "DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL INTELIGENTE PARA UN ROBOT PHOENIX TIPO HEXÁPODO MEDIANTE LA TARJETA STM32F4 DISCOVERY Y SIMULINK DE MATLAB" realizado por los señores CRISTIAN ADRIÁN ANDINO ALBERCA y DIEGO XAVIER RODRÍGUEZ SÁNCHEZ, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a los señores CRISTIAN ADRIÁN ANDINO ALBERCA y DIEGO XAVIER RODRÍGUEZ SÁNCHEZ para que lo sustente públicamente.

Sangolquí, 08 de junio de 2016

ING. VÍCTOR GONZALO PROAÑO ROSERO MSc.

DIRECTOR

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Nosotros, CRISTIAN ADRIÁN ANDINO ALBERCA, con cédula de identidad N° 1722249024 y DIEGO XAVIER RODRÍGUEZ SÁNCHEZ, con cédula de identidad N° 0401193727 declaramos que este trabajo de titulación "DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL INTELIGENTE PARA UN ROBOT PHOENIX TIPO HEXÁPODO MEDIANTE LA TARJETA STM32F4 DISCOVERY Y SIMULINK DE MATLAB" ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas. Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, 08 de junio de 2016

CRISTIAN ADRIÁN ANDINO ALBERCA

C.C: 1722249024

DIEGO XAVIER RODRÍGUEZ SÁNCHEZ

C.C: 0401193727

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Nosotros, CRISTIAN ADRIÁN ANDINO ALBERCA, y DIEGO XAVIER RODRÍGUEZ SÁNCHEZ, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL INTELIGENTE PARA UN ROBOT PHOENIX TIPO HEXÁPODO MEDIANTE LA TARJETA STM32F4 DISCOVERY Y SIMULINK DE MATLAB" cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Sangolquí, 08 de junio de 2016

CRISTIAN ADRIÁN ANDINO ALBERCA

C.C: 1722249024

DIEGO XAVIER RODRÍGUEZ SÁNCHEZ

C.C: 0401193727

DEDICATORIA

Dedico este proyecto de investigación a Dios,

a mis padres Hernán y Otilia, y a mi hijo Joaquín.

A Dios porque ha estado conmigo a cada paso que doy,

cuidándome y dándome fortaleza para continuar,

a mis padres, quienes a lo largo de mi vida han

velado por mi bienestar y educación siendo mi apoyo en todo momento

y a mi hijo que es y siempre será fuente de inspiración para mi vida.

CRISTIAN ANDINO

DEDICATORIA

Este trabajo de investigación está dedicado de manera especial a mis padres Hugo Rodríguez y Nancy Sánchez, quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento.

A mis familiares y amigos por haberme dado su fuerza y apoyo incondicional en esta carrera de Ingeniería que me han ayudado y llevado hasta donde estoy ahora.

Por último a mi compañero Cristian Andino y a nuestro Director Victor Proaño, que con esfuerzo y trabajo en conjunto, se ha logrado la terminación de esta investigación.

DIEGO RODRÍGUEZ

AGRADECIMIENTO

Este presente trabajo de investigación quiero agradecer a mis padres, familiares y amigos porque me brindaron su apoyo moral y emocional, para seguir estudiando y lograr el objetivo principal, obtener el título universitario de tan linda carrera.

A mi amigo Diego Rodríguez con quien he compartido grandes momentos y hemos podido concluir satisfactoriamente este proyecto de investigación.

De igual manera a mis queridos formadores en especial al Ingeniero Víctor Proaño y al Ingeniero Darwin Alulema, por el apoyo y el conocimiento brindado para culminar satisfactoriamente con este trabajo.

CRISTIAN ANDINO

AGRADECIMIENTO

Por el apoyo moral y emocional,

agradezco a las personas más importantes en mi vida, mis Padres.

A todas las personas que hicieron realidad la parte de un sueño

que está por cumplirse.

Gracias por el apoyo de mis profesores,

quienes supieron guiarme y darme las pautas para seguir y

alcanzar una madures profesional e intelectual.

Un agradecimiento de forma especial al Ingeniero Darwin Alulema,

quien nos colaboró con la realización del proyecto, y al Ingeniero Victor Proaño

que nos supo guiar y compartir sus conocimientos,

para la terminación de esta investigación de grado.

DIEGO RODRÍGUEZ

ÍNDICE DE CONTENIDO

ÍNDICE TABLAS

ÍNDICE FIGURAS

RESUMEN

Los robots son una de las tecnologías que han despertado más interés en la aplicación de controladores inteligentes por sus aplicaciones a una gran diversidad de tareas de forma cooperante con el ser humano. En este trabajo de investigación se hace relación a la aplicación de técnicas inteligentes hacia un Robot Phoenix tipo Hexápodo. La programación de los algoritmos de control inteligente se lo realiza en la plataforma de Simulink de Matlab y es embebido en la tarjeta STM32F4 Discovery con la utilización de bloques de programación de la librería denomina Waijung. El proyecto de investigación está dividido en cuatro etapas. En la primera etapa se encuentran los algoritmos de locomoción del Robot Hexápodo, haciendo énfasis a la locomoción tipo trípode con dirección hacia adelante y de realización de giro. Como segundo se tiene la emulación del movimiento de locomoción por medio de un modelo hecho mediante Redes Neuronales Artificiales (RNA) con aprendizaje Supervisado. Esta RNA tiene como entradas la posición PWM inicial de los servomotores y como salidas el valor PWM requerido para el movimiento de locomoción. A continuación se realiza un Controlador Difuso Proporcional Integral (PI) para el equilibrio del Robot Hexápodo. Esto se realiza con el sensor acelerómetro LIS3DSH propia de la tarjeta STM32F4 Discovery, tomando en cuenta como equilibrio el eje *Roll*. Por último se aplica la teoría de Generador Central de Patrones (GCP) para la locomoción del Robot Hexápodo, haciendo que los movimientos se asemejen a los movimientos reales de una araña.

PALABRAS CLAVES:

- **ROBOT HEXÁPODO PHOENIX**
- **STM32F4 DISCOVERY**
- **REDES NEURONALES ARTIFICIALES**
- **CONTROL DIFUSO**
- **GENERADOR CENTRAL DE PATRONES**

ABSTRACT

Robots are one of the technologies that have aroused more interest in the application of intelligent controllers for their applications to a great diversity of tasks cooperatively with humans. This project is made related the application of intelligent techniques to a Robot Hexapod Phoenix. Programming intelligent control algorithms, it takes on the platform of Matlab's Simulink and is embedded in the STM32F4 Discovery board with the use of block programming library called Waijung. The research project is divided into four stages. The first stage is locomotion algorithms of Hexapod Robot, emphasizing the tripod type locomotion forward direction and performing rotation. Later the emulation movement of locomotion through a model based on Artificial Neural Networks (ANN) with supervised learning. This RNA has as inputs the PWM initial position of the servomotors and as outputs the PWM value required for the movement of locomotion. Following is a Proportional Integral (PI) Fuzzy Controller that is performed to balance the Hexapod Robot. This is done with the LIS3DSH accelerometer sensor of STM32F4 Discovery board, considering as the balance of the Roll axis. Finally the theory of Central Pattern Generators (CPG) for Hexapod Robot locomotion is applied, making movements resembling the actual movements of a spider.

KEYWORDS:

• HEXAPOD ROBOT PHOENIX

• STM32F4 DISCOVERY

• ARTIFICIAL NEURAL NETWORKS

• FUZZY CONTROL

• CENTRAL PATTERN GENERATOR

CAPÍTULO I
INTRODUCCIÓN

1.1. Antecedentes

En la Universidad de las Fuerzas Armadas- ESPE se han desarrollado varios prototipos de robots, entre ellos robots del tipo hexápodos. Este tipo de robots resultan muy útiles para la experimentación de algoritmos de control clásico e inteligente. El Control Inteligente ofrece algunas perspectivas interesantes de control, porque es capaz de suministrar metodologías que permiten realizar de forma automática algunas de las tareas realizadas típicamente por los humanos.

Algunos algoritmos de control inteligente son basados en Redes Neuronales Artificiales, las cuales permiten crear Sistemas de Computación Neuronal (Fernando Izaurieta, Carlos Saavedra, 2009). En la Universidad CENIDET de Cuernavaca, México, se desarrolló un trabajo investigativo basado en la simulación de Redes Neuronales Artificiales para la generación de movimientos de locomoción para un robot Hexápodo. Ese trabajo se centra básicamente en la creación de una red neuronal que pueda generar oscilaciones que controlen las extremidades del robot; permitiendo sólo la generación de un tipo de caminar, tipo trípode. (Julio Pérez Machorro, 2009)

Otro tipo de algoritmo de Control Inteligente es la Lógica Difusa, que ha demostrado ser una herramienta útil en el campo de la robótica, caracterizada principalmente por: la imposibilidad de disponer de un modelo matemático fiable del robot, la incertidumbre e imprecisión de los datos proporcionados por los sensores y la presencia de incertidumbre en el conocimiento del entorno. En la Universidad de Querétaro se hizo un Algoritmo Difuso de locomoción libre para un Robot caminante de seis patas (Robot Hexápodo), el cual utiliza técnicas de lógica difusa para la toma de decisiones. La valoración de los algoritmos se realiza mediante la simulación del proceso de locomoción del robot. (Efrén Gorrostieta, Emilio Vargas, 2007)

Existe una nueva rama de la Inteligencia Artificial denominada Generadores Centrales de Patrones (CGP). Un CGP es un complejo sistema compuesto por

subredes de neuronas, capaces de generar patrones de movimiento rítmico en animales, las cuales también coordinan los tiempos precisos para cada tipo de locomoción. Actualmente hay algunos circuitos de CGP realizados. Por ejemplo, en la Universidad Catania, Italia, se realizó un análisis de las estructuras de procesamiento neuronal para la locomoción en un robot hexápodo. La principal inspiración proviene del paradigma biológico del Generador Central de Patrones, que se utiliza para modelar las poblaciones neuronales responsables de la planificación y el control de la locomoción en animales. (Paolo Arena, Luigi Fortuna, 2002)

1.2. Justificación e Importancia

En la actualidad se llevan a cabo investigaciones sobre los robots que presentan extremidades, debido a sus amplias características como son:

- Estos robots pueden navegar por terrenos de difícil acceso para vehículos convencionales como: regiones pantanosas, junglas, minas, volcanes, terremotos, reactores nucleares, en el espacio exterior, etc.
- Este tipo de robots exige un mayor conocimiento del comportamiento de las extremidades de los seres vivos, permitiendo realizar los estudios de locomoción cuya dinámica se ha considerado como las principales manifestaciones de vida.
- El diseño y construcción de estos robots abarca una extensa gama de aplicaciones, entre las que se puede destacar: rescate de seres vivos en desastres naturales, monitoreo de volcanes activos, recolección de datos y muestras en zonas de alto grado de contaminación, etc. (César Fuertes, Romel Llumiquinga, 2005)

Los robots hexápodos presentan un gran inconveniente al ejecutar sus movimientos, debido a la dinámica y estabilidad del robot. Para poder dar una solución al equilibrio en este trabajo se propone la lógica difusa, porque permite realizar control de posición sin la necesidad de un modelo matemático establecido. En este caso se controla mediante “palabras”, las cuales reemplazan a los valores numéricos de las variables y permiten establecer las reglas.

Otro tema aplicado a este robot hexápodo fueron las Redes Neuronales Artificiales, los cuales son estructuras matemáticas que permiten emular sistemas no lineales, debido a su alta capacidad para establecer correlaciones entre conjuntos de

datos de entrada/ salida. Por tanto, si se dispone de los patrones de entrenamiento del movimiento del robot hexápodo se puede entrenar una Red Neuronal Artificial para reproducir el movimiento.

Y por último se aplicará al robot los Generadores centrales de patrones, cuyo propósito principal es generar movimientos más aproximados a los robots basándose en aspectos biológicos de los animales. Este tema es aplicable al robot hexápodo debido a su semejanza a una araña. Las arañas, como muchos otros seres vivos, disponen de un Generador Central de Patrones (GCP) propio del sistema nervioso, cuyo propósito es generar los movimientos de locomoción. Las neuronas artificiales del robot presentan un comportamiento de patrón rítmico que sirve para controlar y coordinar las actividades de locomoción repetitivas del robot. La locomoción basada en GCP tiene la habilidad de generar esquemas de control modular, capaz de adaptarse y generar movimientos complejos de manera coordinada, utilizando estructuras neuronales embebidas en una plataforma.

La Universidad de las Fuerzas Armadas-ESPE, está obligada a realizar y demostrar evidencias de la investigación hecha por la misma. Una forma de hacerlo es a través de la aplicación de las ideas propuestas por la teoría. La Lógica difusa y las Redes Neuronales Artificiales son temas de gran importancia y pertenecen a la materia de Control Inteligente. Con este trabajo se pretende utilizar algoritmos inteligentes de manera práctica. Además en este trabajo se ha incluido un tema nuevo que son los Generadores Centrales de Patrones, un tema que puede ser incluido posteriormente en la materia de Control Inteligente. La utilización de este tipo de algoritmos en un robot resulta de gran importancia para estudios posteriores aplicados tanto a la Robótica como la Inteligencia Artificial.

1.3. Alcance del Proyecto

El proyecto se divide en 5 partes que son:

- Descripción de la Mecánica.
- Diseño Electrónico.
- Control de estabilidad basada en un controlador PI Difuso.

- Emulación de locomoción por Redes Neuronales Artificiales de entrenamiento Supervisado.
- Aplicación de navegación basada en Generadores Centrales de Patrones.

1.3.1. Alcance del Diseño Mecánico

Se realizó un robot móvil tipo hexápodo de modelo Phoenix, con la forma de una araña que consta mecánicamente con las siguientes características:

- Consta de 6 extremidades (patas del robot).
- Cada extremidad tiene tres grados de libertad, es decir 3 servomotores por extremidad para un total de 18 servomotores.
- El movimiento de las articulaciones es mediante la utilización de servomotores que trabajan con una fuente DC de 4 a 7 voltios.
- Consta de una pieza metálica central donde se instala cada extremidad y sirve de soporte para la placa PCB y el controlador del Robot.

1.3.2. Alcance de Diseño Electrónico

El controlador utilizado fue la tarjeta STM32F4 Discovery. Es una placa de evaluación de bajo costo de microcontroladores de STM32F4 ARM Cortex-M4. Incluye una herramienta de depuración incrustada ST-LINK/V2, un acelerómetro LIS3DSH digital y un micrófono digital, un DAC de audio con el controlador de altavoz integrado clase D, LEDs y botones pulsadores y un conector USB OTG micro-AB. La tarjeta se programa usando la herramienta de Simulink de Matlab. Esta tarjeta permite programar códigos de Matlab mediante una librería de Matlab llamada Waijung blockset, en la cual se puede encontrar los periféricos (bloques) del microcontrolador que utilizaron para el sistema de encendido. En los servomotores se utilizaron los bloques de generación PWM, que tiene como función dar pulsos de ancho variable a una frecuencia fija. La tarjeta STM32F4 Discovery utiliza Matlab y herramientas (tools de Matlab) para su programación, que sirven para la implementación de algoritmos de control inteligente de manera más eficiente, tales como:

- Neural: es la librería para el diseño de redes neuronales.
- Newff: permite crear una red neuronal.

- Sim: simula una red neuronal.
- Fuzzy: es la herramienta propia de Matlab para lógica difusa.

1.3.3. Alcance del Control de posición PI Difuso

Utilizando algoritmos de control difuso se realizó un control de posición para las seis extremidades del robot Hexápodo. Se usó como entrada al controlador, el acelerómetro LIS3DSH que entrega los valores de *Roll* del ángulo en el cual se inclina dicho acelerómetro. Fue necesario realizar un controlador PI para reducir el error en estado estacionario y estabilizar al robot en una posición de cero grados de inclinación, permitiendo al robot Hexápodo pararse y equilibrarse automáticamente sobre distintas superficies.

1.3.4. Alcance del Sistema de Redes Neuronales Artificiales

En esta parte se realizó un análisis sobre los distintos movimientos de locomoción que tiene un robot hexápodo y se evaluó cual es el más óptimo sobre distintas superficies. Una vez identificado el movimiento más óptimo se realizó una emulación de los algoritmos de locomoción utilizando diferentes métodos de entrenamiento supervisado de Redes Neuronales Artificiales para un control de movimiento. En esta parte se utilizó como entradas del sistema a los servomotores del robot y las salidas fueron los movimientos de locomoción como caminar hacia adelante y girar mediante la utilización de Redes Neuronales Artificiales.

1.3.5. Aplicación sobre Generadores Centrales de Patrones

Se realizó una investigación acerca de los principios que dan origen a la locomoción basada en GCP y los modelos de interconexión para la generación de múltiples patrones de locomoción. Con esto se recreó otro tipo de locomoción al robot hexápodo, es decir mediante un GCP.

1.4. Objetivos

1.4.1. General

Implementar algoritmos de control inteligente para una plataforma robótica Phoenix tipo hexápodo mediante la tarjeta STM32F4 Discovery y Simulink de Matlab.

1.4.2. Específicos

- Implementar la parte mecánica del Robot Hexápodo (hardware).
- Diseñar e implementar una placa tipo PCB para la adaptación de la tarjeta STM32F4 Discovery y los demás componentes electrónicos necesarios para el robot Phoenix.
- Diseñar e implementar algoritmos de locomoción para los diferentes movimientos del robot Phoenix.
- Implementar algoritmos de Redes Neuronales Artificiales para lograr un aprendizaje supervisado de movimientos de locomoción.
- Implementar un controlador difuso para equilibrio del robot hexápodo.
- Investigar e implementar una aplicación de Generadores Centrales de Patrones para el Robot.
- Evaluar la funcionalidad del robot con todos sus algoritmos implementados en diferentes entornos.

CAPÍTULO II
MARCO TEÓRICO

2.1. Introducción

Este capítulo describe el fundamento teórico de los algoritmos aplicados al Robot Hexápodo. En primera instancia se presenta una breve descripción del Robot Hexápodo Phoenix y de la tarjeta STM32F4 Discovery. La tarjeta STM32F4 tiene un acelerómetro LIS3DSH, que es utilizado para el equilibrio del robot hexápodo en el eje *Roll;* y la librería Waijung Blocket, que permite la programación de la tarjeta STM32F4 mediante la plataforma de Simulink de Matlab.

A continuación se realiza una breve explicación de las Redes Neuronales Artificiales (RNA) y Control Difuso, cuya teoría fue aplicada al Robot Hexápodo para el control de locomoción y equilibrio respectivamente. Por último una amplia explicación del Generador Central de Patrones, desde su principio biológico hasta su modelamiento matemático, para demostrar su principal objetivo que es realizar la locomoción de forma más natural de los robots; en este caso del Robot Hexápodo.

2.2. El Sistema Robótico Hexápodo Phoenix

El robot hexápodo Phoenix (Ver Figura 1) tiene un gran parecido a los arácnidos de tres pares de patas en cuanto a su cuerpo y las articulaciones de sus extremidades, haciéndolo ver lo más natural posible. Los 3 DOF (Grados de libertad) característicos del modelo Phoenix, hacen que el robot pueda caminar en cualquier dirección. El modelo Phoenix presenta 3 servomotores por cada extremidad por lo que da un total de 18 servomotores para el movimiento del robot. Este modelo fue creado por el ingeniero noruego Kåre Halvorsen de la casa comercial Lynxmotion.

El robot está hecho por piezas de aluminio ultra resistente de alta calidad, haciendo que el robot sea muy ligero y fácil de controlar. El controlador comercial del robot Phoenix es el BotBoarduino SSC-32. Este microcontrolador presenta un programa creado por Jeroen Janssen para la generación de impulsos PWM para los servomotores Hitec HS-645 (propias de la casa comercial), controlando la velocidad

de caminata, dirección del robot, rotación en su propio eje y regulación de la altura del robot mediante los servomotores instalados en cada extremidad. Adicionalmente presenta otro controlador denominado RC-01, que permite manipular el robot de forma inalámbrica por medio del control PS2, haciendo que el usuario pueda operar el robot de forma más simple y sencilla.

Por último el robot Phoenix presenta una batería tipo LiPo de 6 voltios Ni-MH. El precio en Estados Unidos del robot Phoenix con controladores y servomotores es de $1247.89. (Robot Shop Inc, 2012)

Tabla 1

Características y Especificaciones del Robot Hexápodo Phoenix

Características	Especificaciones
Control de movimiento de servomotor	Circuito local cerrado
Dirección	Omnidireccional
Número de Patas	6
Grados de Libertad por pata	3
Velocidad de Movimiento	5”/s
Altura Corporal	2”
Altura Máxima	5.25”
Anchura Corporal	5.88”
Anchura Máxima	17”
Longitud Corporal	7.50”
Longitud Máxima	14.50”
Peso sin Baterías	10 lb con 10 oz

Fuente: (Robot Shop Inc, 2012)

Figura 1 Robot Hexápodo Phoenix

Fuente: (Robot Shop Inc, 2012)

2.3. Tarjeta STM32F4 Discovery

La STM32F4 Discovery (Ver Figura 2) es una tarjeta de bajo costo y de fácil programación, teniendo varias opciones en lenguajes de programación como C/C++ y programación de bloques en la plataforma Matlab con ayuda de la librería Waijung.

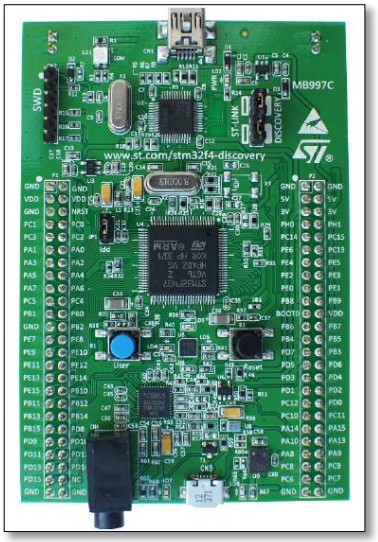

Figura 2 Tarjeta STM32F4 Discovery

Fuente: (STMicroelectronics, 2014)

2.3.1. Características generales de la tarjeta STM32F4

La tarjeta STM32F4 DISCOVERY presenta las siguientes características:

- Microcontrolador STM32F407VGT6 con 1 MB de memoria flash, 192 KB de RAM, encapsulado LQFP100.
- ST-LINK/V2 incorporado con modo selector para usar el kit como un ST-LINK/V2 independiente (con conector SWD para programación y depuración).
- Fuente de alimentación de placa: a través del bus USB o desde una fuente de alimentación externa de 5V.
- Fuente de aplicación exterior: 5v y 3v.
- Sensor de movimiento ST MEMS LIS3DSH, acelerómetro con salida digital de 3 ejes.
- Sensor de audio ST MEMS MP45DT02, micrófono digital omnidireccional.
- Audio DAC CS43L22 con controlador integrado de altavoz clase D.
- Ocho LEDs:
  - LD1 (rojo / verde) para la comunicación USB
  - LD2 (rojo) alimentación 3,3 V
  - Cuatro LEDs de usuario, LD3 (naranja), LD4 (verde), LD5 (rojo) y LD6 (azul)
  - 2 LEDs USB OTG, LD7 (verde) VBus y LD8 (rojo) de sobre corriente.
- Dos pulsadores (usuario y reseteo).
- USB OTG con conector micro-AB. (STMicroelectronics, 2014)

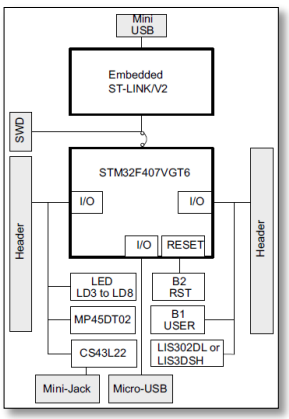

Figura 3 Diagrama de Bloques de Hardware de STM32F4

Fuente: (STMicroelectronics, 2014)

2.3.2. Sensor de Movimiento ST MEMS LIS3DSH

El sensor LIS3DSH se encuentra presente en la versión de placa MB997C de la tarjeta STM32F4 Discovery. El LIS3DSH es un acelerómetro lineal de 3 ejes (ver Figura 4) de baja potencia. Este sensor presenta escalas de ±2g/±4g/±6g/±8g/±16g y una tasa de captación de valores de salida con frecuencia entre 3.125 Hz a 1.6 KHz.

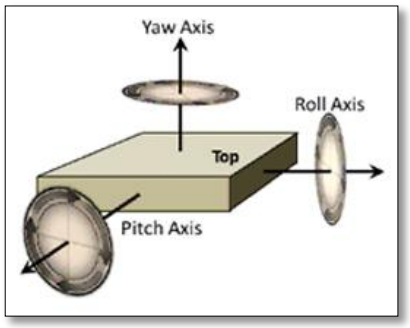

Figura 4 Ejes del sensor LIS3DSH

Fuente: (STMicroelectronics, 2014)

2.3.3. Librería de Programación en Matlab Waijung Blockset

Waijung Blockset es una librería gratuita para Simulink de Matlab, creada en Tailandia por la compañía Aimagin. Esta librería ha sido diseñada especialmente para soportar toda la familia de microcontroladores STM32F4, cuya programación es a base de bloques de simulink, haciéndolo fácil de programar toda la familia de tarjetas STM32F4 existentes en el mercado.

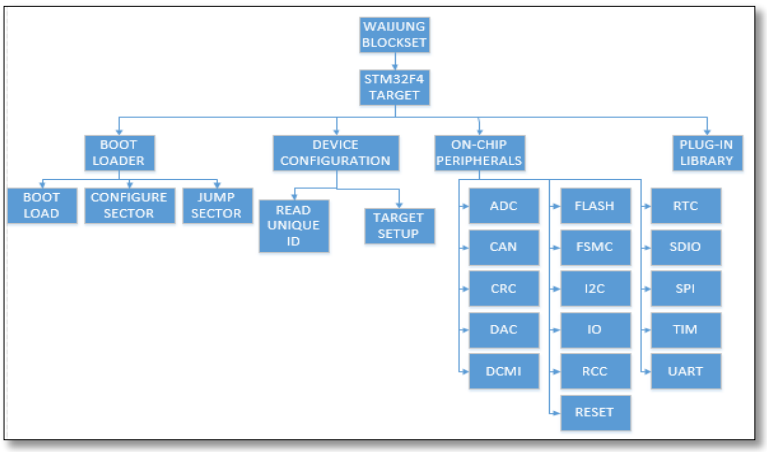

Figura 5 Bloques de Programación de la Librería Waijung

2.4. Movimientos del Robot Hexápodo

Los tipos de movimiento que puede hacer un robot hexápodo para desplazarse hacia adelante se describen en la Figura 6.

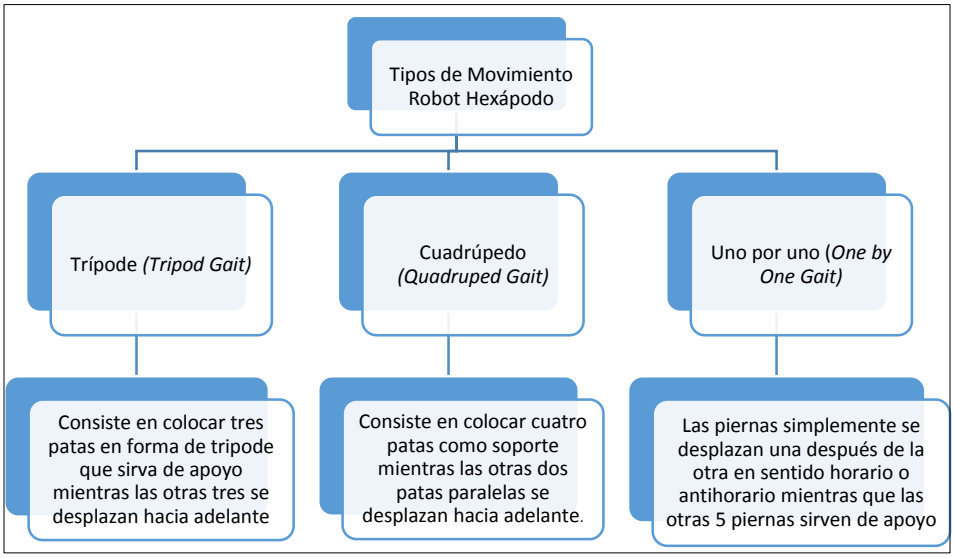

Figura 6 Tipos de Movimientos del Robot Hexápodo

El movimiento más utilizado para este tipo de robots es el trípode, debido a lo sencillo y estable que resulta. Este sistema de tracción permite la ausencia de cualquier sistema de equilibrio. Además brinda mayor velocidad en relación a los otros tipos de movimiento.

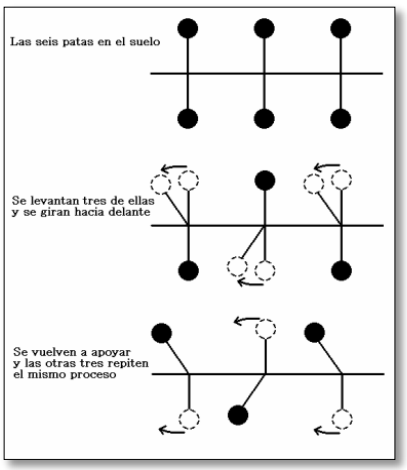

Figura 7 Movimiento Trípode para Robot Hexápodo

Fuente: (Alejandro Rosique, 2006)

La disposición de las articulaciones del robot hexápodo se muestra en la Figura 8. Para poder crear un desplazamiento por medio del movimiento trípode es necesario en primera instancia que se muevan las articulaciones impares (1,3 y 5), mientras las articulaciones pares (2,4 y 6) sirven de soporte para el robot. Al momento que las articulaciones impares terminan su secuencia y llegan al suelo, es el turno de las articulaciones pares realizar el movimiento hacia adelante.

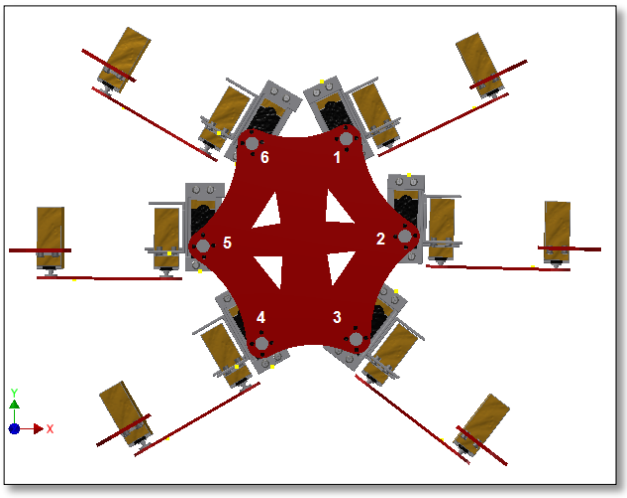

Figura 8 Numeración Articulaciones Robot Hexápodo

2.4.1. Estabilidad Estática

Para mantener el equilibrio del robot se crea un polígono de estabilidad, en este caso es un triángulo el cual se forma primero con las articulaciones pares y luego con las impares (Ver Figura 9).

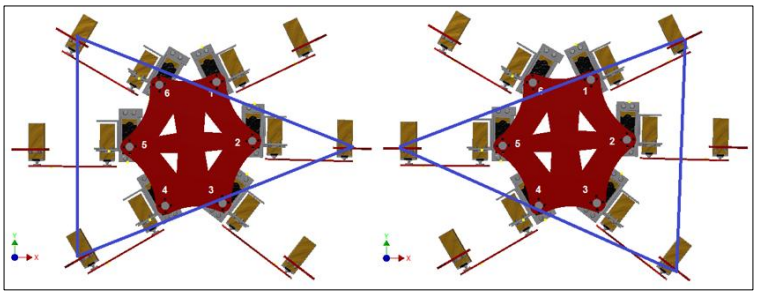

Figura 9 Polígono de Estabilidad

2.5. Redes Neuronales Artificiales (RNA)

2.5.1. Definiciones y Características

Las Redes Neuronales Artificiales (RNA) son aproximaciones matemáticas no lineales a la forma como funciona el cerebro. Las RNA funcionan como un procesador masivo de forma paralela, formado por unidades simples de entrada y salida que tienen una propensión natural para almacenar conocimiento experimental (Ver Figura 10). Las RNA se asemejan al cerebro biológico por dos aspectos muy importantes:

- El conocimiento se adquiere mediante la red dentro de su arquitectura a través de un proceso de aprendizaje.
- Los pesos sinápticos o fuerzas de las conexiones entre neuronas, son utilizadas para el almacenamiento de conocimiento de la red neuronal.

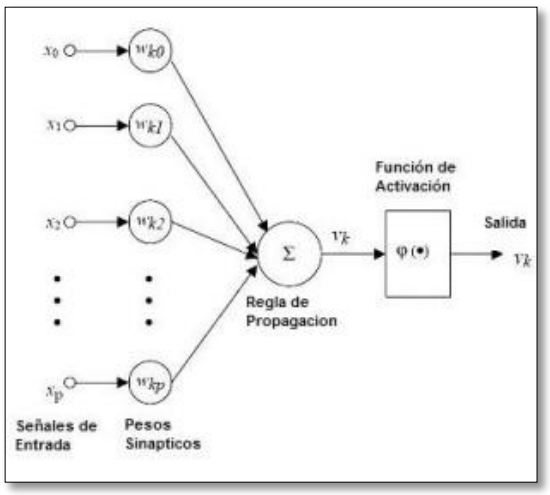

Figura 10 Arquitectura de una Red Neuronal Artificial (RNA)

Fuente: (Jonathan Ocampo, 2011)

El funcionamiento de una RNA en esencia se presenta en la Figura 10. Se aplica un conjunto de entradas "X_p" a la neurona, cada una de ellas representa una salida de otra neurona. Cada entrada se multiplica por sus pesos sinápticos "w_{kp}" correspondiente al grado de conexión de la sinapsis. Todas estas entradas ponderadas se suman y se determina el nivel de activación de cada neurona. Esta salida "v_k" es procesada por su función de activación "φ" para producir la señal de salida de la red

neuronal "$F(v_k)$". Esta función de activación "φ" puede ser una función lineal, umbral o no lineal (Ver Figura 10).

$$F(V_k) = \varphi\left(\sum_{k=0}^{p} x_p w_{kp}\right) \tag{2.1}$$

Las funciones de activación (Ver Figura 11) más usadas son:

- La función Signo
- La función Sigmoidal
- La función Rampa o lineal
- La función Tangente Hiperbólica

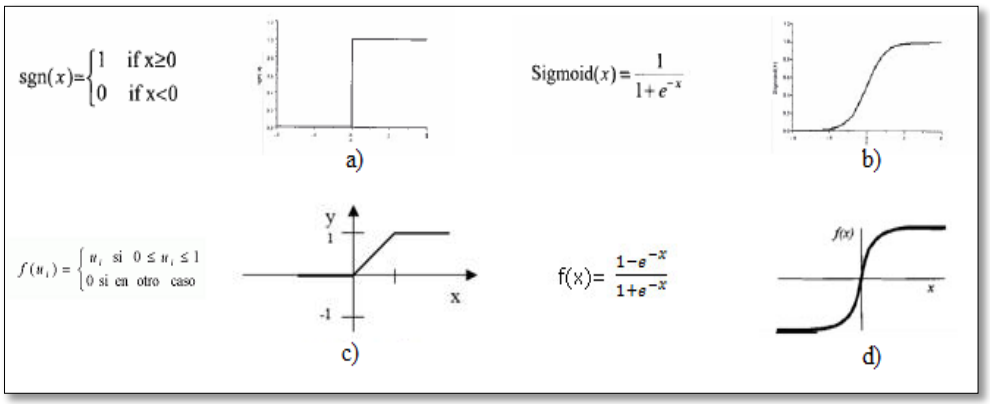

Figura 11 Funciones de Activación

2.5.2. Arquitectura de Red Neuronal

La capacidad de cálculo y potencia de la computación neuronal, depende de la organización y disposición de las neuronas en múltiples conexiones que constituyen las RNA. Cuando se trabaja con grandes cantidades de neuronas, es natural ordenar aquellas que tienen comportamientos similares en capas. Estos pueden ser una red monocapa o multicapa.

2.5.2.1. Red Neuronal Monocapa

La red más simple consiste en un grupo de neuronas ordenadas en una sola capa donde se realizan conexiones laterales entre neuronas, es decir, cada una de sus entradas están conectadas a sus pesos correspondientes de las neuronas.

2.5.2.2. Red Neuronal Multicapa

La red neuronal multicapa se forma a partir del grupo de capas simples conectadas en cascada, es decir, la salida de una capa es la entrada de la siguiente capa. Se ha demostrado que las redes multicapa presentan cualidades y aspectos por encima de las redes de capa simple. (Xabier Basogain Olabe, 2010)

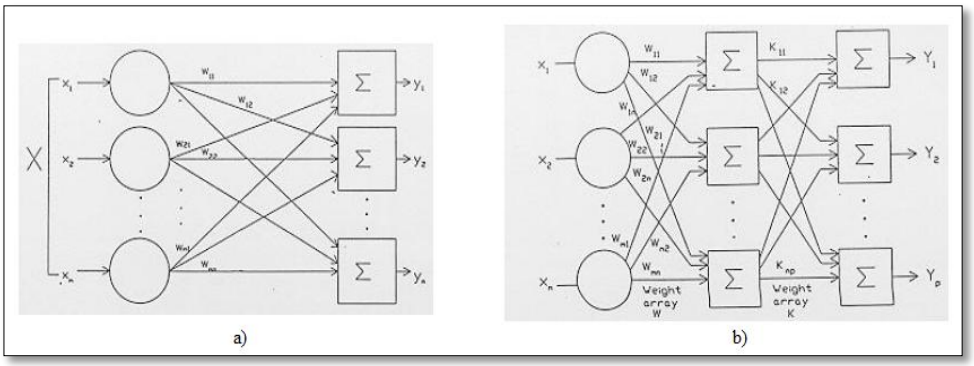

Figura 12 Arquitectura de Red Neuronal

Fuente: (Xabier Basogain Olabe, 2010)

2.5.3. Entrenamiento de Redes Neuronales

El entrenamiento de las redes neuronales es el proceso de presentar los patrones a aprender a la red y a los cambios de los pesos sinápticos de las conexiones, usando una regla de aprendizaje. Existen tres tipos de entrenamiento que son:

- **Entrenamiento Supervisado:** Consiste que la red neuronal dispone de los patrones de entrada y los patrones de salida que se desea para esa entrada; en función de ellos se modifican los pesos sinápticos para ajustar la entrada a esa salida.

- **Entrenamiento No Supervisado:** Consiste en no presentar patrones objetivos, sino solo patrones de entrada, y dejar a la red neuronal clasificar dichos patrones en función de las características comunes de los patrones.
- **Entrenamiento Reforzado:** Este entrenamiento se basa en la regla de Hebb, cuya base es completamente biológica. Se basa en que el supervisor no enseña patrones objetivos si no que solo le dice si acierta (+1) o falla (-1) en su respuesta ante un patrón de entrada para que en función de ello se ajusten sus pesos sinápticos.

2.5.4. Aplicaciones de las Redes Neuronales

Las redes neuronales con esta nueva técnica de cálculo, permite que sea utilizada en una extensa variedad de aplicaciones. Las aplicaciones que son más utilizadas, son aquellas que dan resultados razonables cuando las entradas presentan ruido o son incompletas. Entre las que se pueden enumerar son:

- Análisis y Procesado de Señales
- Reconocimiento de Imágenes
- Filtrado de Ruido
- Robótica
- Procesado del Lenguaje
- Diagnósticos Médicos
- Modelado de Sistemas
- Modelos Económicos y Financieros

2.6. Sistema de Control Difuso (Fuzzy Control)

La lógica difusa fue propuesta por primera vez por Zadeh en 1965 y es basado en el concepto de los conjuntos difusos. La lógica difusa utiliza expresiones que no son ni totalmente ciertas ni totalmente falsas, es decir, la lógica aplica conceptos que pueden tomar un valor de veracidad cualquiera dentro de un conjunto de valores que oscilan entre dos extremos, la verdad absoluta y la falsedad total.

Un sistema de control difuso trabaja de manera muy diferente a un sistema de control convencional. Estos usan el conocimiento experto para generar una base de

conocimientos, que dará al sistema la capacidad de tomar decisiones sobre ciertas acciones que se presentan en su funcionamiento.

Los sistemas de control difuso permiten describir un conjunto de reglas para controlar un proceso y partiendo de estas reglas generar las acciones de control. Los sistemas de control difuso pueden aplicarse en sistemas sencillos como en sistemas cuyos modelos matemáticos son muy complejos. La estructura de un control difuso se observa en la Figura 13.

Figura 13 Estructura de Modelo Difuso

Fuente: (Roberto Supo, 2003)

2.6.1. Fuzzificación

La fuzzificación tiene como objetivo convertir valores reales en valores difusos. En la fuzzificación se asignan grados de pertenencia a cada una de las variables de entrada con relación a los conjuntos difusos previamente definidos, utilizando las funciones de pertenencia asociadas a los conjuntos difusos. Las decisiones de fuzzificación deben ser tomadas por:

- Número de entradas.
- Tamaño de universos en discusión.
- Número y formas de los conjuntos difusos.

2.6.2. Base de Reglas

La base de reglas contiene el conocimiento asociado con el dominio de la aplicación y los objetivos del control. En esta etapa se deben definir las reglas lingüísticas de control que realizarán la toma de decisiones que decidirán la forma en la que debe actuar el sistema.

La lógica difusa se basa en relaciones, las cuales se determinan por medio de cálculo de reglas "SI-ENTONCES" con las cuales se puede modelar aspectos cualitativos del conocimiento humano, así como los procesos de razonamiento sin la necesidad de un análisis cuantitativo de precisión (Pedro Ponce Cruz, 2010). Un ejemplo de una regla sería:

SI la temperatura es alta ENTONCES se debe de encender el ventilador.

2.6.3. Inferencia

La inferencia relaciona los conjuntos difusos de entrada y salida, para representar las reglas que definirán el sistema. La inferencia utiliza la información para generar respuestas mediante el uso de condiciones.

2.6.4. Defuzzificación

La defuzzificación realiza el proceso de adecuar los valores difusos generados en la inferencia a valores reales, que posteriormente se utilizarán en el proceso de control. En la defuzzificación se utilizan métodos matemáticos simples.

2.6.5. Aplicaciones de Lógica Difusa

La lógica difusa presenta varias aplicaciones en áreas multidisciplinarias que van desde la tecnología de electrodomésticos, hasta programas computacionales para tomar decisiones. Entre sus aplicaciones se tiene:

- Cámaras de Video
- Reconocimiento Óptico
- Bases de Datos
- Electrodomésticos
- Controladores (lazo abierto y lazo cerrado)

- Procesos Industriales
- Trayectos de Transporte

2.7. Generador Central de Patrones (GCP)

2.7.1. Principio Biológico de GCP

Los Generadores Central de Patrones GCP (Central Pattern Generators) son circuitos neuronales que sin entradas sensoriales periféricas o centrales producen salidas con patrones rítmicos que se encuentran en los seres vivos, desde animales simples, hasta los mamíferos superiores y los seres humanos. El GCP se localiza generalmente en la espina cordal en los vertebrados o en los ganglios en los invertebrados y son responsables del control rítmico de actividades motoras como caminar, respirar, masticar, entre otros. (Enriques Martínez Peña, 2008)

Para generar este ritmo, las neuronas que los forman se comportan como osciladores dinámicos (Ver Figura 14). Se trata de interneuronas y motoneuronas heterogéneas que trabajan conjuntamente para generar una señal regular.

Las motoneuronas son las encargadas de producir la secuencia rítmica. Que sean heterogéneas significa que cuando están aisladas del resto presentan un comportamiento muy diferente unas de otras y muy diferente al que muestran cuando están conectadas. En este último caso, presentan una actividad rítmica y regular. (Roberto Latorre Camino, 2004)

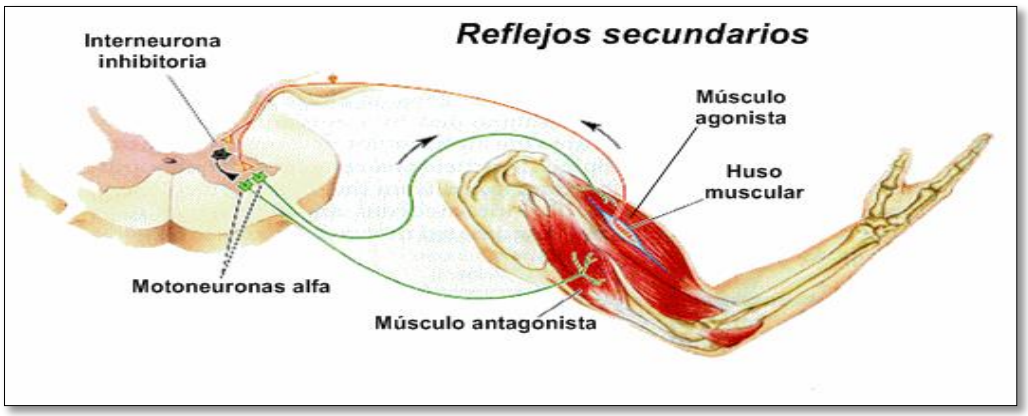

Figura 14 Ejemplo Funcionamiento de Oscilaciones Rítmicas de Neuronas

El GCP es enormemente utilizado para el control de locomoción en robots bípedos, cuadrúpedos y hexápodos. Al usar GCP en el control de movimiento, el problema de control es visto como un flujo continuo de señales análogas manejadas por leyes de difusión. (Julio Pérez Machorro, 2009).

La implementación análoga permite realizar una locomoción más aproximada a la que realizan los seres vivos.

2.7.2. Origen y Fundamento Matemático

El modelo matemático del GCP está basado en las Redes Neuronales Celulares (CNN), las cuales se definen como una red de sistemas no lineales acoplados. El enfoque de las CNN para GCP de locomoción, es considerar un circuito no lineal de segundo orden a uno hecho por dos celdas CNN de primer orden.

2.7.2.1. Redes Neuronales Celulares (CNN)

Las CNN son concebidas en el Laboratorio de la Universidad de California en Berkeley por Leon Chua, que junto a Lin Yang publican en dos artículos científicos la teoría de las CNN y sus primeras aplicaciones (Leon Chua, Lin Yang, 1988). El principal campo de aplicación de las CNN ha sido el procesamiento de imágenes y el reconocimiento de patrones; pero también se abre un enfoque en el control de postura y movimiento de robots biológicamente inspirados.

Los fundamentos en que se basa Leon Chua para generar las CNN son la unión de las Redes Neuronales Artificiales (RNA), y los Autómatas Celulares (AC). De las RNA extrae la capacidad de procesamiento asíncrono en paralelo y la interacción global de los elementos de la red; mientras que de los AC obtiene la estructura, es decir, distribuir sus elementos de procesamiento (células) en rejillas regulares y permitir la comunicación de cada célula con las otras células vecinas.

2.7.2.2. Autómatas Celulares (AC)

Los Autómatas Celulares (AC) surgen en la década de 1940 con John Von Neumann, que intentaba modelar una máquina que fuera capaz de autoreplicarse, llegando así a un modelo matemático de dicha máquina con reglas complicadas sobre una red rectangular. Inicialmente fueron interpretados como conjunto de células que

crecían, se reproducían y morían a medida que pasaba el tiempo. Los AC son un modelo matemático para un sistema dinámico, compuesto por un conjunto de celdas o células que adquieren distintos estados o valores. (David Alejandro Reyes Gómez, 2011)

Un espacio de un AC está compuesto por un arreglo n-dimensional de elementos llamados células que son la unidad básica de los AC. Las células están colocadas en una rejilla de geometría regular (Ver Figura 15); estas muestran varios AC de dos dimensiones expresadas como cuadros y sus relaciones entre ellas (denominados vecinos) como líneas.

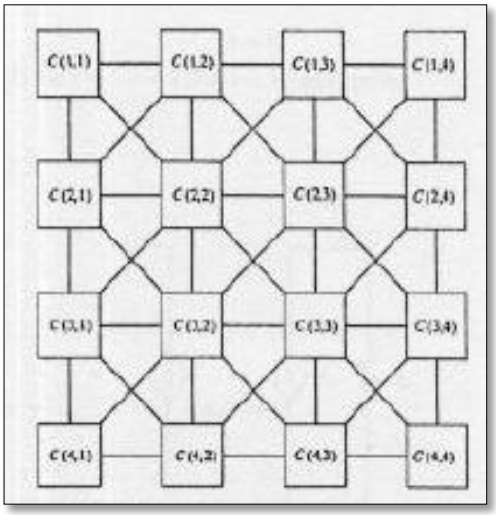

Figura 15 Conexión Bidimensional para el Espacio de AC con 8 vecinos

Fuente: (Leon Chua, Lin Yang, 1988)

En un AC, cada célula interactúa con las demás dentro de su vecindad finita. En un AC bidimensional regular de 8 vecinos (ver Figura 15), se nombra cada célula como $C(i,j)$, dónde i representa la fila y j representa la columna de la célula. El término r representa el radio que existe entre células vecinas y se nombra las células vecinas mediante $Nr(i,j)$, por lo cual queda definida como:

$$Nr(i,j) = \left\{ C(k,l) \middle| \begin{matrix} max\{|k-i|,|l-j|\} \leq r, \\ 1 \leq k \leq M; 1 \leq l \leq N \end{matrix} \right\} \tag{2.2}$$

Donde r es un número entero positivo y k, l son las coordenadas de otra célula donde la magnitud de la diferencia entre i, k y j, l no excede el valor de r para ninguno de los dos casos. En Figura 16 se muestra las vecindades de una célula para $r = 1$.

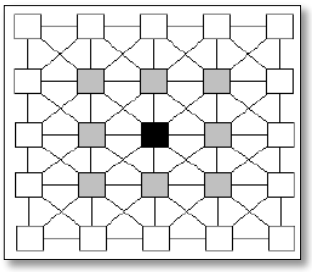

Figura 16 Vecindad con r=1 para una célula central (color negro).

Fuente: (Julio Pérez Machorro, 2009)

2.7.2.3. Modelo Matemático de Redes Neuronales Celulares (CNN)

La Red Neuronal Celular definida por los autores Chua y Yang (Leon Chua, Lin Yang, 1988), consiste en usar circuitos dinámicos no lineales localmente interconectados. Usando un arreglo de 2 dimensiones, en una capa simple, la definición matemáticamente se escribe como (Leon Chua, Lin Yang, 1988):

$$\frac{dx_{ij}}{dt} = -x_{ij} + \sum_{kl \in N_r(ij)} A_{ij;kl}\left(y_{kl}(t), y_{ij}(t)\right) + \sum_{kl \in N_r(ij)} B_{ij;kl}\left(u_{kl}(t), u_{ij}(t)\right) + I_{ij} \tag{2.3}$$

ij Se refiere a la $ij - ésima$ neurona en una malla de 2 dimensiones.

$kl \in N_r(ij)$ Es la $kl - ésima$ neurona con una vecindad de radio r de la $ij - ésima$ neurona.

x_{ij} Es el estado de la $ij - ésima$ neurona.

A Es la plantilla de retroalimentación.

B Es la plantilla de prealimentación, junto con A, llamada también plantilla de control.

u_{ij} Es la entrada del sistema.

I_{ij} Es el término de corriente de polarización (bias), usualmente una constante.

y_{ij} Es la salida de la $ij - ésima$ neurona.

La función de salida se representa como:

$$y_{ij} = f(x_{ij}) \tag{2.4}$$

Donde, $f(x_{ij})$ es una función de salida no lineal o denominada función de activación o umbral. Se dice que si $B = 0$ para cada neurona, entonces la CNN es autónoma.

2.7.2.4 Modelo Matemático para GCP de Locomoción

Para ser llevado a la implementación del GCP, se simplifica el enfoque de CNN considerando al oscilador no lineal como un circuito no lineal de segundo orden, hecho de dos celdas CNN de primer orden (Elizabeth Sedeño Bustos, 2011).

A Partir de (2.3) se escoge $B = 0$, y la función de salida (2.4) será una función lineal con saturación (Función Rampa, ver Figura 16), entonces:

$$y_{ij} = \frac{1}{2}\left(|x_{ij} + 1| - |x_{ij} - 1|\right) \tag{2.5}$$

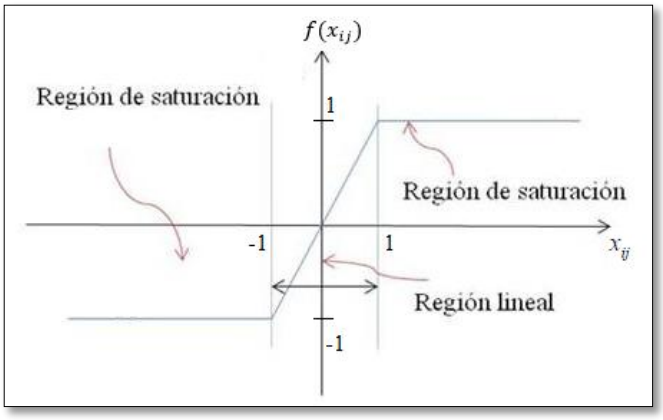

Figura 17 Función de Activación del GCP

De acuerdo a Arena-Fortuna (Paolo Arena, Luigi Fortuna, 2002), se imponen las siguientes condiciones para simplificar las ecuaciones:

$$\frac{dx_1}{dt} = -x_1 + \alpha y_1 - \beta y_2 + I_1 \quad (2.6)$$

$$\frac{dx_2}{dt} = -x_2 + \alpha y_2 - \beta y_1 + I_2 \quad (2.7)$$

Donde,

$$\alpha = (u + 1) \quad (2.8)$$

$$\beta = s \quad (2.9)$$

Por lo tanto las ecuaciones de locomoción para GCP quedan:

$$\frac{dx_1}{dt} = -x_1 + (u + 1)y_1 - sy_2 + I_1 \quad (2.10)$$

$$\frac{dx_2}{dt} = -x_2 + (u + 1)y_2 - sy_1 + I_2 \quad (2.11)$$

Donde $(u + 1)$ representa el ancho de la señal de GCP, y s representa la dinámica de la señal (lento- rápido o rápido- rápido).

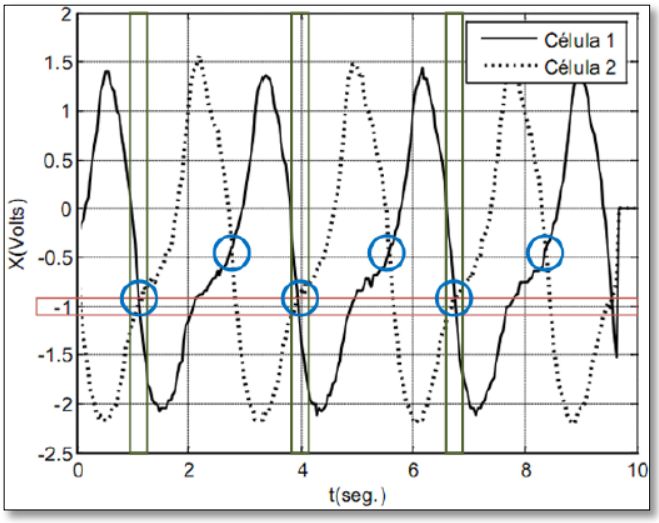

Figura 18 Señal de Locomoción del GCP para un Hexápodo

2.7.3. Aplicaciones del GCP

La misión fundamental del GCP es en generar un patrón de actividad para controlar ciertos músculos que deben moverse de forma rítmica. Entre las aplicaciones más comunes de GCP se tiene:

- Control del movimiento en el campo de la robótica.
- Modelamiento de Plantas por neuromecanismos
- Codificación temporal de Información.
- Reconocimiento de Firmas.

CAPÍTULO III
DISEÑO E IMPLEMENTACIÓN

3.1. Introducción

El propósito de este capítulo es mostrar a detalle todos los dispositivos mecánicos, y electrónicos para el ensamblaje del Robot, y el Diseño de los algoritmos de Control Inteligente embebidos en la tarjeta STM32F4 Discovery. En primera instancia se muestra las medidas de las piezas mecánicas del Robot Hexápodo tipo Phoenix, cuyo diseño viene prefabricado por la casa comercial Lynxmotion; y la descripción detallada de ensamblaje de piezas.

Posteriormente se encuentra el Diseño e Implementación Electrónica, que especifica los materiales necesarios para el funcionamiento del Robot. Además el diseño de una placa PCB para la conexión de la tarjeta STM32F4 Discovery y los servomotores para el control del Robot.

Finalmente el Diseño de algoritmos de Control Inteligente, explica la implementación de algoritmos de Redes Neuronales Artificiales (RNA), Control Difuso (Fuzzy) y Generador Central de Patrones (GCP). Estos algoritmos son programados en la tarjeta STM32F4 Discovery mediante la plataforma Simulink de Matlab.

3.2. Descripción de la Mecánica del Robot Hexápodo

En esta sección se describe las partes mecánicas que conforman el Robot Hexápodo Phoenix y la forma de ensamblaje para cada uno de sus componentes.

3.2.1. Descripción de Piezas Mecánicas del Robot

El Robot Hexápodo Phoenix consta de las siguientes partes:

- Una base donde se acoplan las 6 articulaciones terminadas.

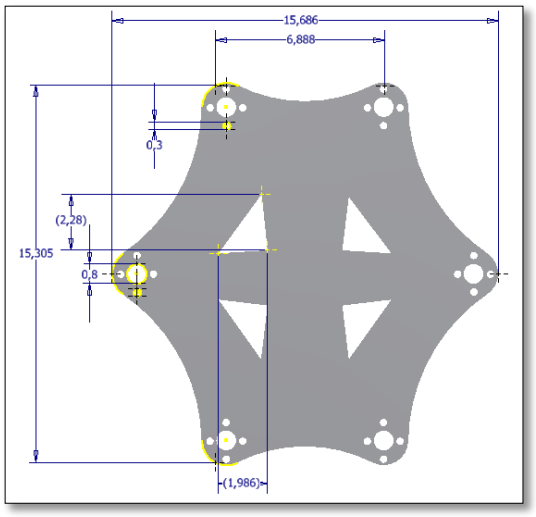

Figura 19 Base del Robot Hexápodo Phoenix

- 6 piezas denominados muslos que sirven de union entre la base y las patas del robot.

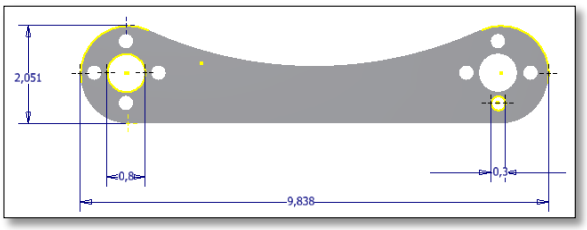

Figura 20 Muslo del Robot Hexápodo Phoenix

- 6 piezas denominadas patas que sirven de apoyo del robot sobre la superficie.

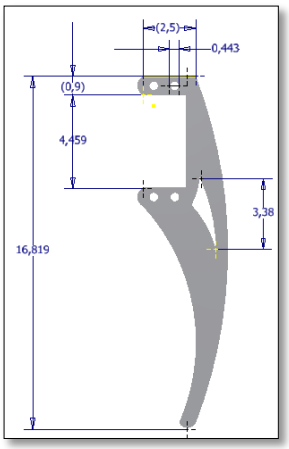

Figura 21 Pata del Robot Hexápodo Phoenix

- 12 piezas de soporte para los servomotores del cuerpo y de los muslos.

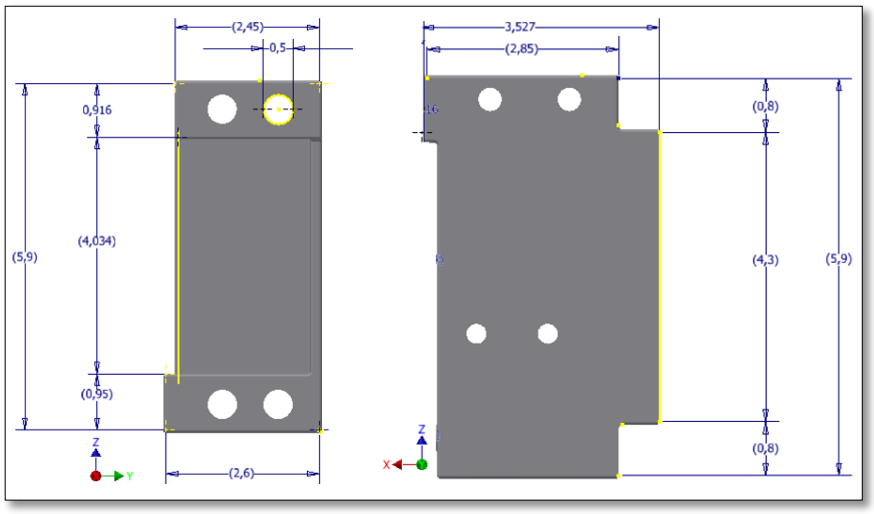

Figura 22 Soportes de servomotores

- 18 servomotores marca TowerPro MG995.

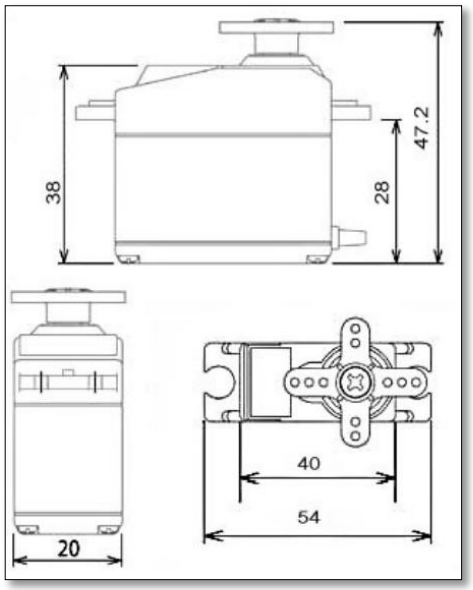

Figura 23 Medidas de servomotor TowerPro MG995

Fuente: (Electronica Caldas, 2011)

El servomotor estándar de alta velocidad MG995 presenta una rotación de 180 grados (90 grados por dirección). Sus especificaciones mecánicas se observan en la Tabla 2.

Tabla 2

Especificaciones Mecánicas del servomotor MG995

Características	Especificaciones Mecánicas
Peso	55 g
Límite de Ángulo	180°
Altura	54 mm
Ancho	20 mm
Largo	38 mm
Rango de Temperatura	0 °C- 55 °C
Material	Poliamida

Fuente: (Electronica Caldas, 2011)

3.2.2. Ensamblaje del Robot.

Los pasos para armar una articulación del robot son los siguientes:

- Paso 1:

Colocar el servomotor sobre la pieza denominada Pata colocando 4 tornillos de ⅝” y tuercas de ⅛”, tal como se indica en Figura 24:

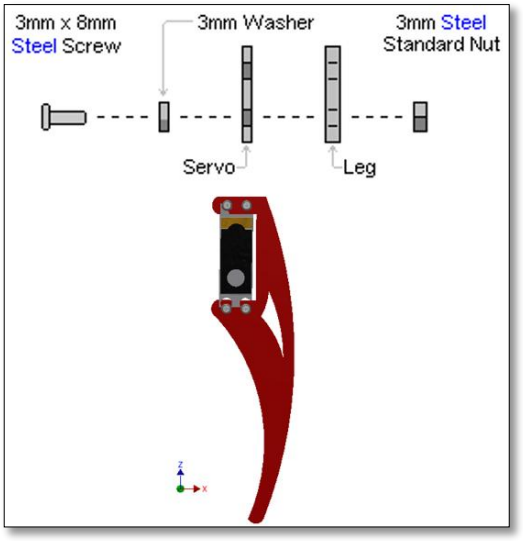

Figura 24 Ensamble de Pata del Robot Hexápodo Phoenix

Fuente: (Robot Shop Inc, 2012)

- Paso 2:

Se colocan los soportes de los motores del cuerpo y muslo mediante tornillos de ¼” y tuercas como se indica en Figura 25:

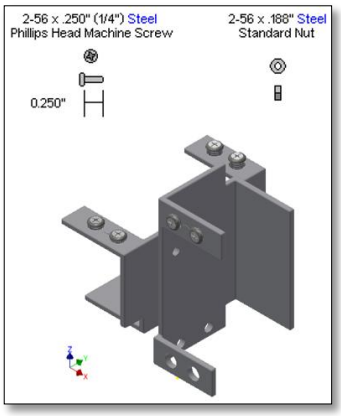

Figura 25 Ensamble de Soportes de servomotores del Robot Phoenix

Fuente: (Robot Shop Inc, 2012)

- Paso 3:

El siguiente paso es insertar los servomotores denominados cuerpo y muslo en los soportes como se indica en Figura 26:

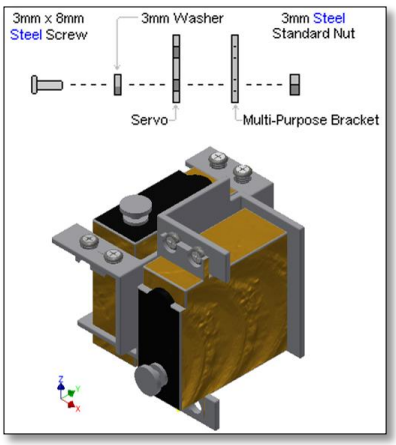

Figura 26 Acople de servomotores con soportes del Robot Phoenix

Fuente: (Robot Shop Inc, 2012)

- Paso 5:

Se acopla la pieza llamada muslo sobre el acople antes realizado con 4 tornillos de ⅛” como muestra la Figura 27:

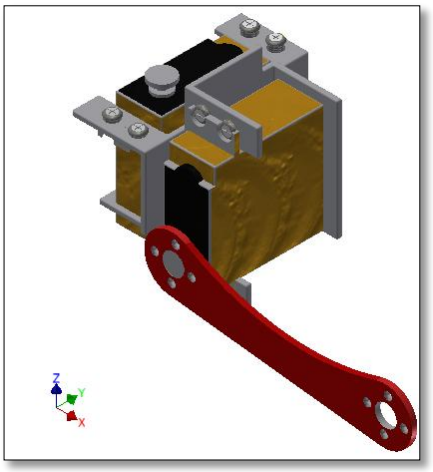

Figura 27 Acople de Pieza Muslo a servomotor

- Paso 6:

Se acopla la parte armada en el paso 1 con la parte del paso 5, igualmente con 4 tornillos ⅛” asi como muestra la Figura 28. De esta forma queda armada la articulación nombrando a los servomotores como cuerpo, muslos y pata.

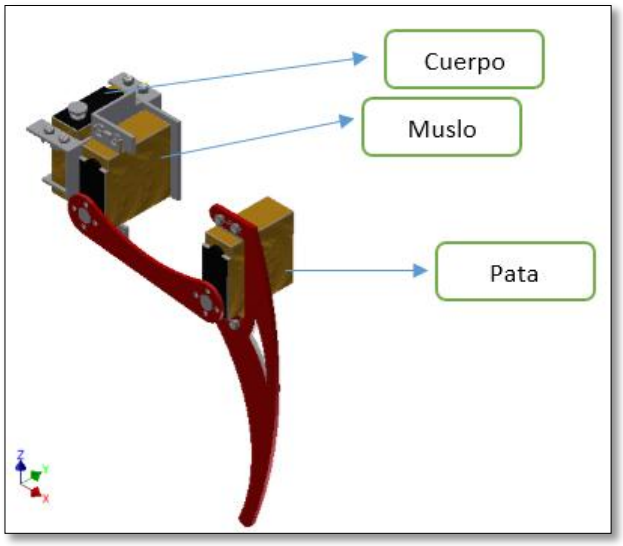

Figura 28 Articulación completa del Robot Hexápodo Phoenix

- Paso 7:

Hay que repetir el mismo procedimiento para las otras 5 articulaciones y termina con el acople de las mismas sobre la pieza de la base.

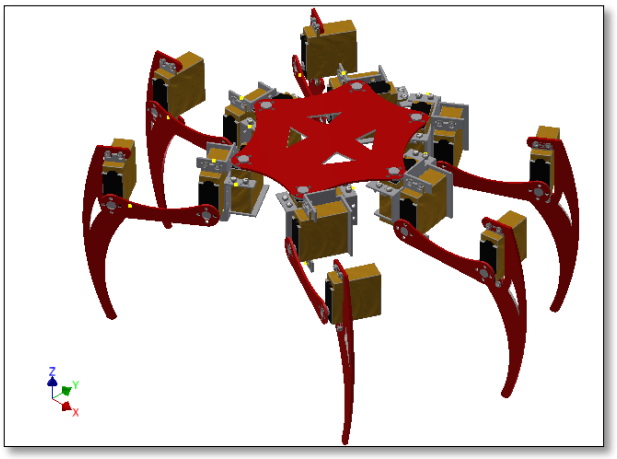

Figura 29 Armado Total del Robot Hexápodo Phoenix

3.3. Descripción de la Electrónica del Robot Hexápodo

3.3.1. Funcionamiento de Servomotores

Como se explicó en la implementación mecánica, los servomotores a controlar son los TowerPro MG995. Estos servomotores son controlados por PWM (Modulación por Ancho de Pulso) con un periodo de 20 milisegundos (Ver Figura 31). Las especificaciones electrónicas se observan en la Tabla 3.

Tabla 3

Especificaciones Electrónicas del servomotor MG995

Características	Especificaciones Electrónicas
Torque	4.8 V (8.5 Kgf.cm), 6 V (10 Kgf.cm)
Velocidad de Operación	4.8 V (0.2 s/60°), 6 V (0.16 s/60°)
Voltaje de Operación	4.8 V a 7.2 V
Ancho de Banda muerta	5 us
Periodo de Funcionamiento	20 ms (50Hz)
Corriente de Operación	500 mA

Fuente: (Electronica ColdFire, 2010)

El servomotor MG995 presenta 3 cables de 30 centímetros cada uno, con un conector de 3 pines universal tipo “S” hembra. Los cables en el conector están distribuidos de la siguiente forma: Rojo = Alimentación (+), Café = Alimentación (-) o Tierra, Naranja = Señal PWM. (Ver Figura 30)

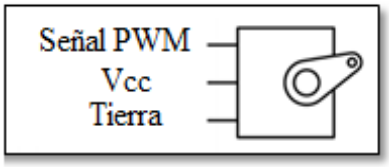

Figura 30 Conexión de Servomotor MG995

Fuente: (Electronica Caldas, 2011)

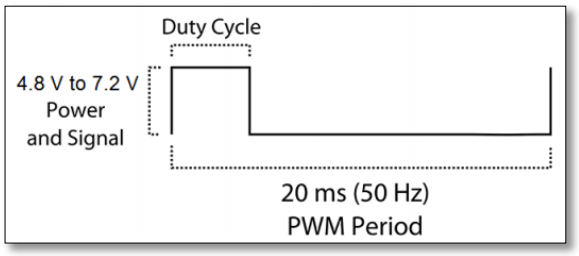

Figura 31 PWM de servomotor MG995

Fuente: (Electronica Caldas, 2011)

En la Figura 32 se presenta la relación de ancho de pulso PWM y el valor del ángulo del servomotor.

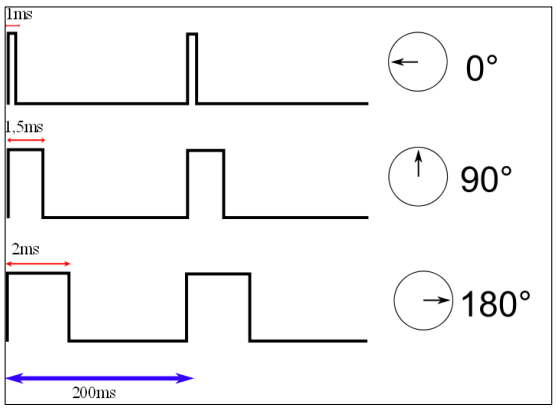

Figura 32 Posición del servomotor

Fuente: (Electronica Caldas, 2011)

La Tabla 4 presenta los valores del ángulo relacionados con el ancho de pulso y los valores PWM que se utiliza en la programación de Simulink de Matlab.

Tabla 4

Relación PWM con ángulo de servomotores.

Valor Ángulo (°)	Ancho del pulso (ms)	Valor PWM (%)
0	1	5
90	1.5	7.5
180	2	10

El control de PWM se realiza mediante Simulink de Matlab con los bloques presentes de la librería Waijung. Esta librería permite embeber el código Simulink de Matlab hacia la tarjeta STM32F4 Discovery. El bloque de programación para PWM es denominado *"Basic PWM"*. (Ver Figura 33)

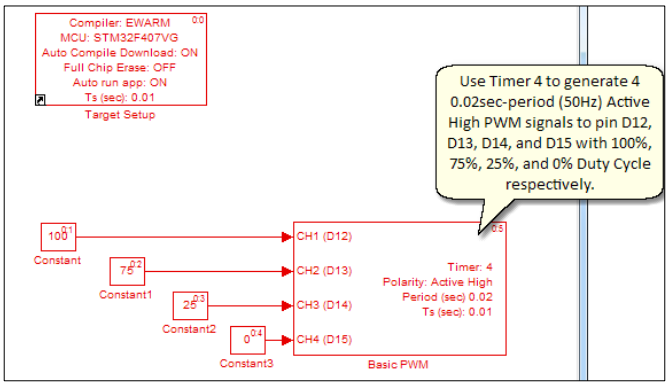

Figura 33 Bloque de Programación para PWM

Fuente: (Aimagin, 2015)

El menú de funcionamiento de *"Basic PWM"* (Ver Figura 34) presenta varios parámetros de programación. Las especificaciones del menú se observa en Tabla 5 .

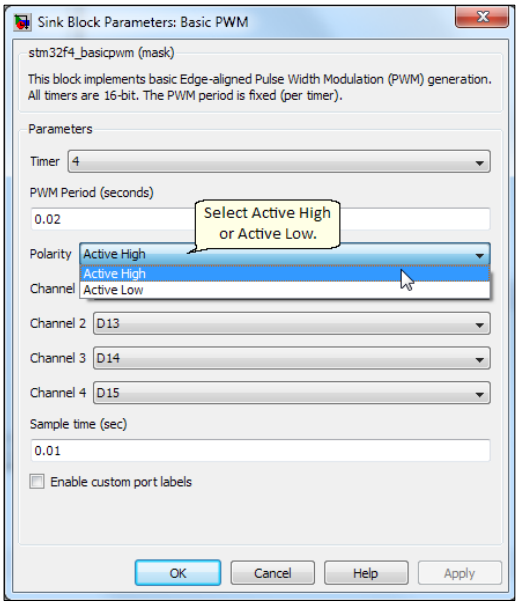

Figura 34 Menú de Funcionamiento del bloque "Basic PWM"

Fuente: (Aimagin, 2015)

El listado de pines PWM de la tarjeta STM32F4 Discovery se encuentra en el **¡Error! No se encuentra el origen de la referencia.**.

Tabla 5

Especificaciones del Menú "Basic PWM"

Item de Configuración	Descripción
Timer	Lista de los temporizadores que permitan realizar señal PWM (1,2,3,14,15,16,17)
PWM Period	Especifica el valor en segundos de periodo de pulso de PWM
Polarity	Activación de Pulso PWM en Positivo o negativo (Alto o Bajo)
Channel 1	Selección de pin PWM del canal 1 (Depende de Temporizador)
Channel 2	Selección de pin PWM del canal 2 (Depende de Temporizador)

CONTINÚA

Channel 3	Selección de pin PWM del canal 3 (Depende de Temporizador)
Channel 4	Selección de pin PWM del canal 4 (Depende de Temporizador)
Sample Time	Periodo de Muestreo en segundos
Enable custom port labels	Activación de etiquetas de canales

Fuente: (Aimagin, 2015)

Los valores de entrada para cada canal son valores de porcentaje entre 0 a 100% al periodo total de PWM, permitiendo el giro del servomotor al ángulo deseado.

3.3.2. Diseño de Placa PCB

El diseño de la placa PCB es basado en la polarización de los 18 servomotores que necesita el Robot Hexápodo, la señal de control de cada servomotor, la implementación de capacitores para evitar el ruido a los cuales son propensos los servomotores, la incrustación de la tarjeta STM32F4 a la placa PCB con sus pines de control PWM para cada servomotor y su polarización respectivamente.

Los 18 servomotores se encuentran distribuidos en 6 grupos, donde cada grupo representa la estructura completa de cada extremidad, es decir, representa los 3 servomotores encargados del movimiento completo de una pata. Su distribución se observa en la Figura 35.

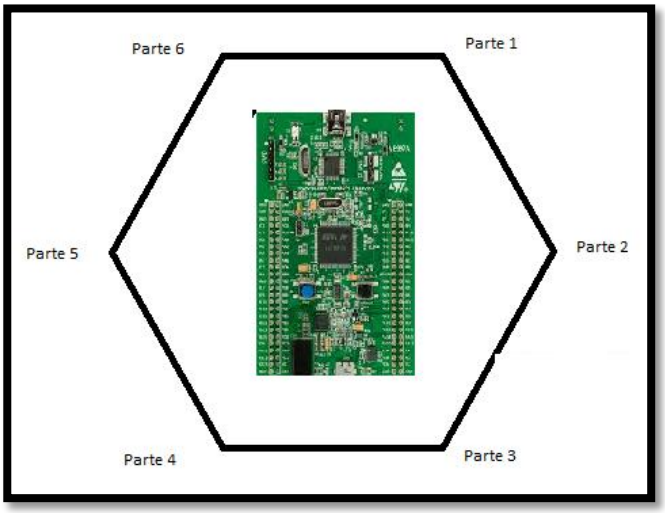

Figura 35 Distribución por grupos para placa PCB

Para cada grupo es necesario distribuir 3 pines de control PWM, dando un total de 18 pines. La distribución de pines se observa en la Tabla 6.

Tabla 6

Distribución de Pines de Control PWM

Parte 1	Parte 2	Parte 3	Parte 4	Parte 5	Parte 6
Pata1	Pata2	Pata3	Pata4	Pata5	Pata6
B4	B5	C9	E9	E11	E13
Muslo1	Muslo2	Muslo3	Muslo4	Muslo5	Muslo6
C6	C7	C8	E5	E6	B9
Cuerpo1	Cuerpo2	Cuerpo3	Cuerpo4	Cuerpo5	Cuerpo6
B6	B7	B8	A5	B10	B11

La conexión de los servomotores y de la tarjeta STM32F4 Discovery se realizó en el programa Proteus ISIS v5.62, cuyo esquema electrónico para la fabricación de la placa PCB se observa en **¡Error! No se encuentra el origen de la referencia.**. La distribución de pines de PWM se observa en Figura 36.

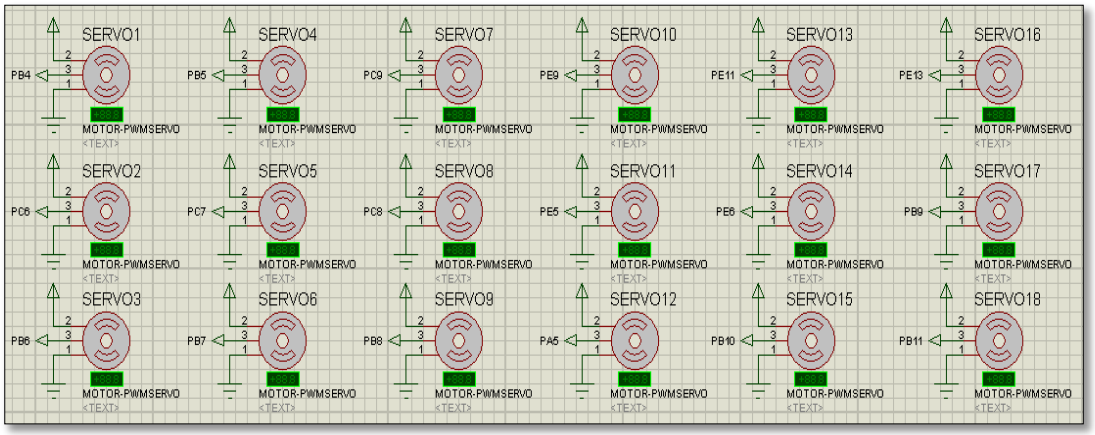

Figura 36 Conexión de Servomotores a Pines de Control

Un problema en el control de servomotores, es su sensibilidad al ruido. Para solucionar el ruido que normalmente proviene de las variaciones en el nivel de tensión de la fuente, se utiliza capacitores de conexión por *"bypass"* (Ver Figura 37). Esta

conexión permite regular las posibles perturbaciones de la fuente y obtener una señal sin ruido que afecte el funcionamiento de los servomotores.

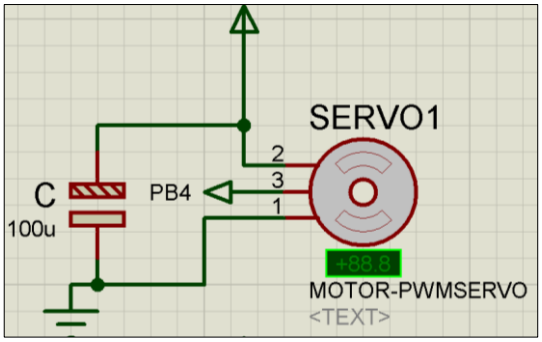

Figura 37 Conexión de Capacitor por Bypass

Para este problema de ruido se instala 1 capacitor electrolítico por cada extremidad. El valor del capacitor es de 100 uF, que es sugerido por el fabricante TowerPro. Los materiales necesarios para la fabricación de la placa PCB se muestran en Tabla 7.

Tabla 7

Lista de Materiales de Placa PCB

Elementos	Cantidad
Capacitor Electrolítico 100uF	6
Borneras	2
Espadines Macho	54
Espadines Hembra	4x24
LED Rojo	2
Resistencias 330 Ohm	4

Para el ajuste de la placa PCB en el Robot, se fabricó la placa en forma de la base del Robot (Ver Figura 19), donde en cada esquina se encuentra las conexiones de los 3 servomotores que representa cada extremidad (Ver Figura 38).

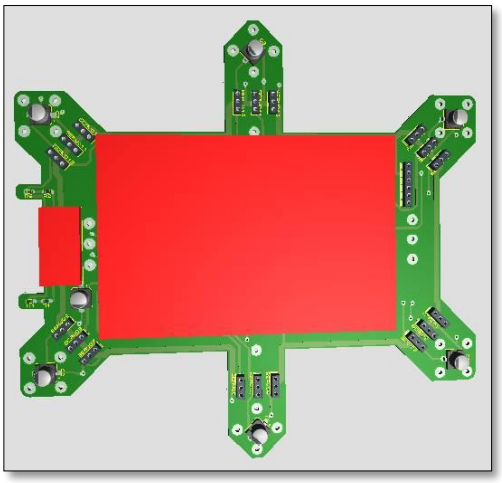

Figura 38 Placa PCB

3.4. Algoritmo de Locomoción

3.4.1. Problemática del Algoritmo de locomoción.

Este tema es primordial para los temas posteriores a este trabajo de investigación, cuyo objetivo principal es encontrar los valores de posición de los 18 servomotores que controlan al robot hexápodo para que se pare y luego camine. La posición inicial del robot sirve para el controlador PI Difuso, mientras que la caminata sirve para los algoritmos de RNA y GCP.

3.4.2. Posición Inicial del Robot

Se realizó el posicionamiento de los triángulos de estabilidad quedando como indica la Figura 39.

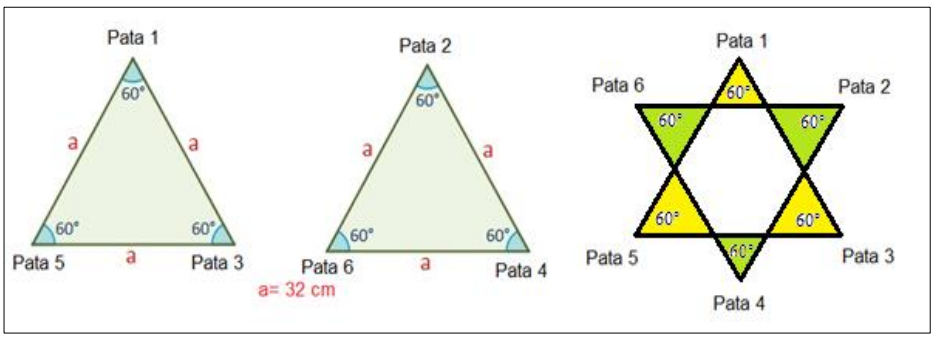

Figura 39 Triángulos de Estabilidad

Para realizar el posicionamiento de estos triángulos de estabilidad se realizaron pruebas y se comprobó el error existente por medio del uso del acelerómetro de la tarjeta STM32F4 Discovery. Si los *leds* indicadores del acelerómetro (*roll* y *pitch*) no se encienden indica que no existe inclinación. El programa del acelerómetro que permite visualizar la inclinación de la tarjeta se encuentra en **¡Error! No se encuentra el origen de la referencia.**.

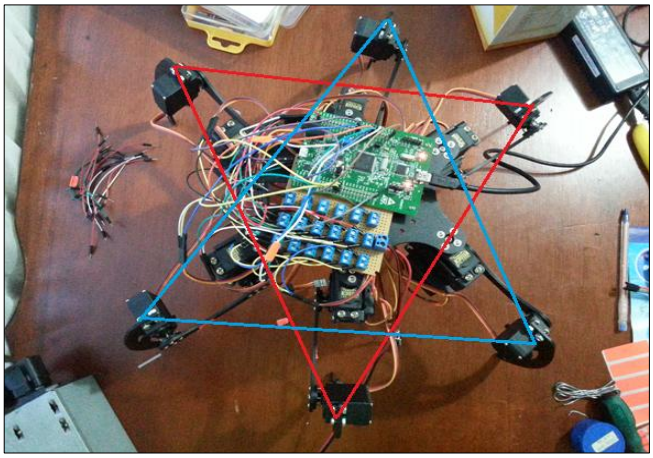

Figura 40 Pruebas de estabilidad del Robot Hexápodo

Los valores de distancia admiten un valor de ± 2 cm para permanecer en una posición estable. Con estos valores se puede colocar al robot en su posición inicial. Primero se posicionan los 6 servomotores descritos como cuerpo1, cuerpo2, cuerpo3, cuerpo4, cuerpo5 y cuerpo6 mediante el uso de los bloques *"Basic PWM"*. Estos valores varían de acuerdo al modo en el que se encuentren anclados los servomotores a la estructura. Las constantes que permiten cumplir el principio de estabilidad se observa en la Figura 41.

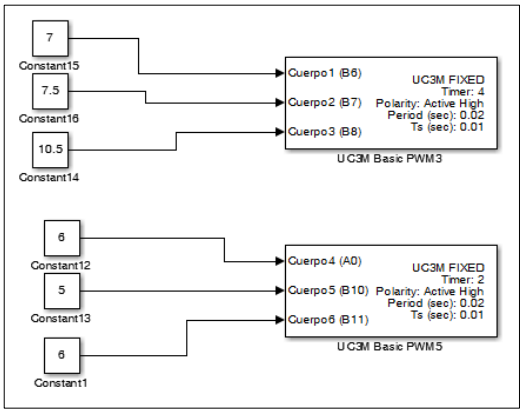

Figura 41 Valores PWM para Posición Inicial de Servomotores del cuerpo

Los siguientes servomotores en posicionar son los denominados muslo1, muslo2, muslo3, muslo4, muslo5 y muslo6. Estos son colocados en una posición de 0˚ de inclinación en el eje x, es decir paralelos a la superficie que se apoya. Mientras que los servomotores denominados pata1, pata2, pata3, pata4, pata5 y pata6 deben estar a 0˚de inclinación en el eje z, perpendiculares a la superficie de apoyo.

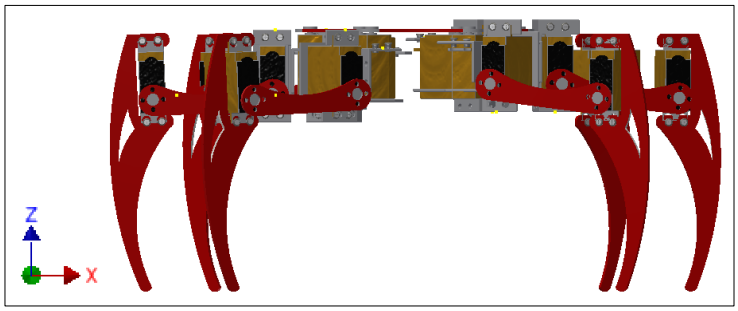

Figura 42 Ejes del Robot Hexápodo

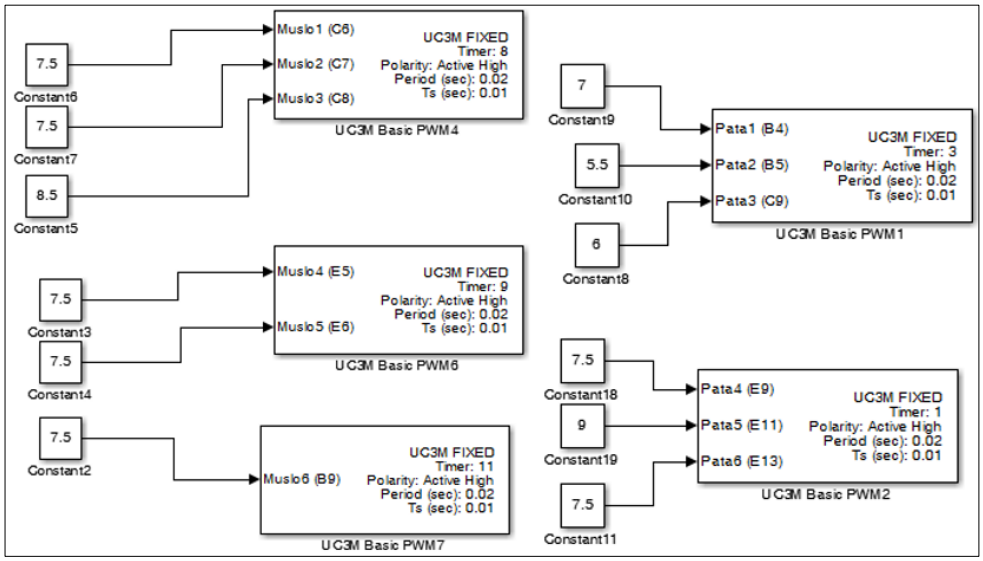

Figura 43 Valores PWM para posición inicial servomotores Muslos y Patas

3.4.3. Secuencia de Movimientos

Para la creación de la secuencia de cada una de las articulaciones es necesario crear dos estados para los servomotores del cuerpo y también de los muslos. Los servomotores del cuerpo son los que permiten desplazar las extremidades hacia adelante y los muslos permiten elevarlas del suelo.

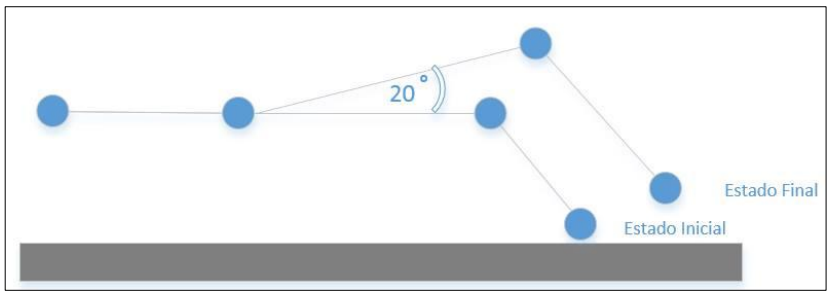

Figura 44 Estados servomotores Muslos

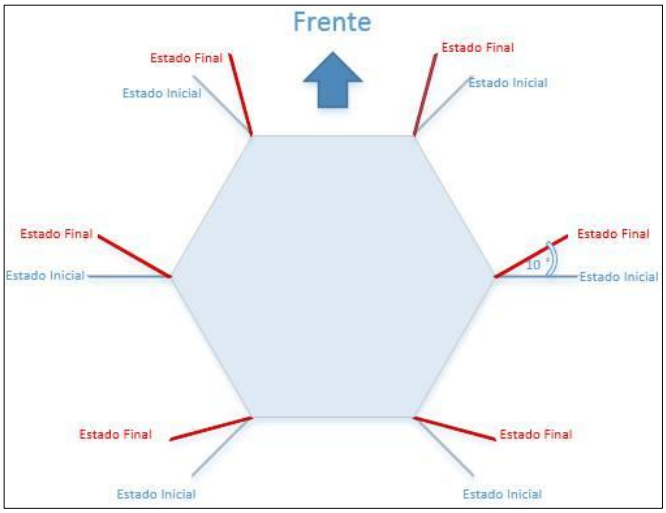

Figura 45 Estados servomotores Cuerpo

Los valores del Estado Final se obtienen aumentando un valor entero a la señal PWM de los servomotores del lado derecho del robot (extremidades 1, 2 y 3), mientras que para los servomotores del lado izquierdo se disminuye el mismo valor (extremidades 4, 5 y 6).

Tabla 8

Estados y Valores PWM para servomotores Cuerpo

CUERPO	Inicial	Final
cuerpo 1	7	8
cuerpo 2	7.5	8.5
cuerpo 3	10.5	11.5
cuerpo 4	6	5
cuerpo 5	5	4
cuerpo 6	6	5

Tabla 9

Estados y Valores PWM para servomotores Muslo

MUSLO	Inicial	Final
muslo 1	7.5	9.5
muslo 2	7.5	9.5
muslo 3	8.5	10.5

muslo 4	7.5	5.5
muslo 5	7.5	5.5
muslo 6	7.5	5.5

La generación del movimiento hacia adelante obedece a la Figura 46, donde el eje *X* representa el tiempo (0.25 segundos cada estado) y el eje *Y* es estado en el cual se encuentra el servomotor.

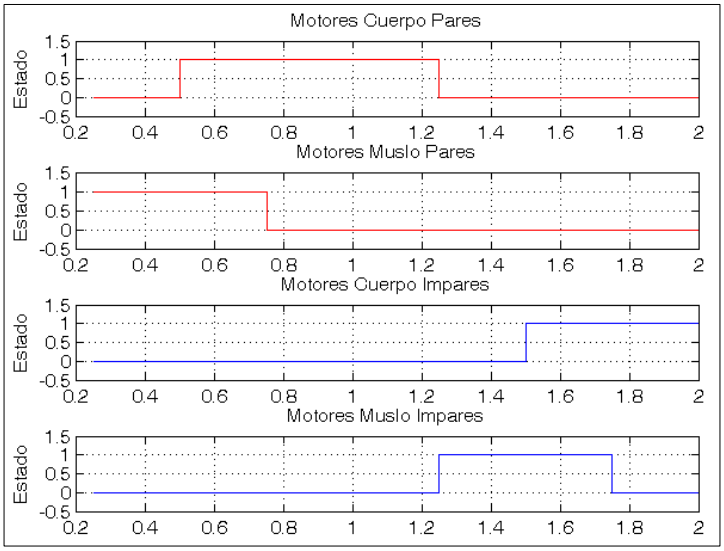

Figura 46 Secuencia caminata hacia Adelante

La secuencia de valores PWM de cada motor para lograr el desplazamiento hacia adelante del robot hexápodo se representan en la Tabla 10.

Tabla 10

Secuencia PWM desplazamiento adelante

Cuerpo	**Valor PWM**	**Muslo**	**Valor PWM**
Cuerpo 1	[7 7 7 7 7 8 8 8]	Muslo 1	[7.5 7.5 7.5 7.5 9.5 9.5 7.5 7.5]
Cuerpo 2	[7.5 8.5 8.5 8.5 7.5 7.5 7.5 7.5]	Muslo 2	[9.5 9.5 7.5 7.5 7.5 7.5 7.5 7.5]
Cuerpo 3	[10.5 10.5 10.5 10.5 10.5 11.5 11.5 11.5]	Muslo 3	[8.5 8.5 8.5 8.5 10.5 10.5 8.5 8.5]

Cuerpo 4	[6 5 5 5 6 6 6 6]	Muslo 4	[5.5 5.5 7.5 7.5 7.5 7.5 7.5 7.5]
Cuerpo 5	[5 5 5 5 5 4 4 4]	Muslo 5	[7.5 7.5 7.5 7.5 5.5 5.5 7.5 7.5]
Cuerpo 6	[6 5 5 5 6 6 6 6]	Muslo 6	[5.5 5.5 7.5 7.5 7.5 7.5 7.5 7.5]

Para realizar el giro del robot hexápodo se cumplen los mismos principios. Lo que varía es el estado final de los motores del cuerpo.

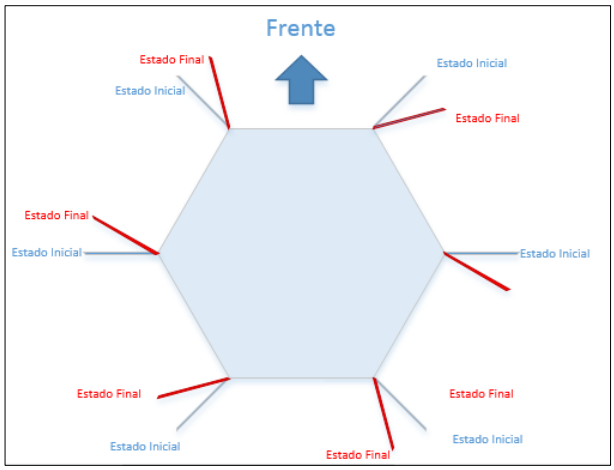

Figura 47 Estados de servomotores cuerpo de giro hacia la Derecha

Tabla 11

Secuencia PWM para giro a la Derecha

Cuerpo	Valor PWM	Muslo	Valor PWM
Cuerpo 1	[7 7 7 7 7 8.5 8.5 8.5]	Muslo 1	[7.5 7.5 7.5 7.5 9.5 9.5 7.5 7.5]
Cuerpo 2	[7.5 9 9 9 7.5 7.5 7.5 7.5]	Muslo 2	[9.5 9.5 7.5 7.5 7.5 7.5 7.5 7.5]
Cuerpo 3	[10.5 10.5 10.5 10.5 10.5 12 12 12]	Muslo 3	[8.5 8.5 8.5 8.5 10.5 10.5 8.5 8.5]
Cuerpo 4	[6 7.5 7.5 7.5 6 6 6 6]	Muslo 4	[5.5 5.5 7.5 7.5 7.5 7.5 7.5 7.5]

Cuerpo 5	[5 5 5 5 5 6.5 6.5 6.5]	Muslo 5	[7.5 7.5 7.5 7.5 5.5 5.5 7.5 7.5]
Cuerpo 6	[6 7.5 7.5 7.5 6 6 6 6]	Muslo 6	[6 6 8 8 8 8 8 8]

3.5. Algoritmos de Redes Neuronales Artificiales (RNA)

3.5.1. Problemática del Entrenamiento RNA.

En este tema se describe los parámetros necesarios para crear una RNA en Matlab y embeberla dentro de la tarjeta STM32F4 DISCOVERY. El objetivo de esta parte es crear una RNA que permita emular los algoritmos de locomoción conociendo los parámetros de entrenamiento. Se usó el error cuadrático medio como medida de calidad del aprendizaje RNA.

3.5.2. Solución de Redes Neuronales Artificiales

El objetivo que se persigue es construir una RNA en la arquitectura de un perceptrón multicapa y aprendizaje supervisado de la secuencia de valores PWM para cada servomotor necesarios para la locomoción del robot hexápodo mediante Matlab. En esta parte se utiliza dos métodos de entrenamiento de RNA: retropropagación y regla delta generalizada.

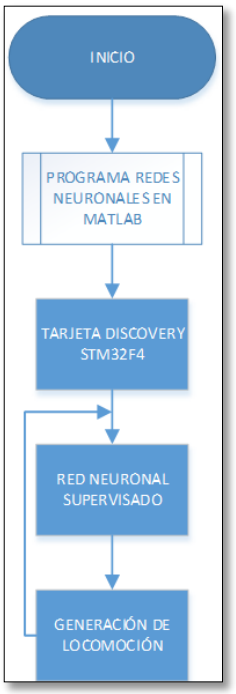

Figura 48 Propuesta de Solución de RNA

Se detalla la creación de la RNA que permite emular los movimientos hacia adelante y giro a la derecha del robot hexápodo.

Para la creación de la RNA se toman en cuenta los siguientes aspectos:

- Patrones de entrenamiento
- Arquitectura de la red
- Función de Activación.
- Mecanismo de aprendizaje.

3.5.3. Patrones de entrenamiento.

El conjunto de entradas viene dado por la posición actual de los servomotores que intervienen en la locomoción del robot hexápodo. Como ya se mencionó la locomoción se realiza mediante el movimiento sincronizado de 12 servomotores distribuidos dos por cada una de las extremidades del robot (cuerpo1-6 y muslo1-6), mientras los últimos 6 servomotores (patas) sirven como soporte del robot y por tanto no serán utilizados para este modelo.

El conjunto de entradas tendrán valores flotantes aleatorios entre 5 y 12, debido a que son valores mínimos y máximos respectivamente, de valores PWM que permiten posicionar a los servomotores. Para generar el movimiento de locomoción es necesario una secuencia sincronizada de 8 movimientos por cada servomotor, por tanto de igual manera el conjunto de entrada será un vector con 8 valores.

Los patrones de entrenamiento se han obtenido a partir de un conjunto aleatorio de posiciones iniciales para el hexápodo. Cada posición inicial está descrita por un valor PWM asociado a cada uno de los 12 servomotores que producen el movimiento del cuerpo y los muslos. Cada patrón de entrenamiento se completa con el valor de *target*, que se ha obtenido de la puesta en funcionamiento del servomotor durante 1 segundo. El *target* representa el valor PWM que se aplicará a cada servomotor en la siguiente iteración después que ha transcurrido 1 segundo.

Por tanto, se han utilizado 8 patrones de entrenamiento, cada uno refleja la posición inicial del hexápodo y el valor PWM requerido para el movimiento desde esa posición. Cada patrón se forma de una matriz de 12x2.

Entonces la RNA producirá en su salida los valores requeridos de PWM para provocar el movimiento siguiente (después de 1 segundo) tomando como información la posición actual.

	Patrón 1		Patrón 2		Patrón 3		Patrón 4	
	inputs	*targets*	*inputs*	*targets*	*inputs*	*targets*	*inputs*	*targets*
cuerpo1	7.38	7.00	10.80	7.00	6.93	7.00	6.75	7.00
cuerpo2	6.23	7.50	7.70	8.50	8.45	8.50	7.29	8.50
cuerpo3	6.49	10.50	8.93	10.50	8.23	10.50	10.20	10.50
cuerpo4	10.12	6.00	7.12	5.00	9.23	5.00	7.27	5.00
cuerpo5	9.47	5.00	9.76	5.00	9.55	5.00	10.07	5.00
cuerpo6	7.59	6.00	7.28	5.00	9.77	5.00	7.22	5.00
muslo1	10.75	7.50	8.53	7.50	7.38	7.50	10.65	7.50
muslo2	6.17	9.50	9.50	9.50	9.40	7.50	7.75	7.50
muslo3	8.19	8.50	10.45	8.50	9.28	8.50	6.98	8.50
muslo4	7.91	5.50	10.80	5.50	6.81	7.50	7.26	7.50
muslo5	9.83	7.50	8.74	7.50	6.59	7.50	9.08	7.50
muslo5	9.98	5.50	6.69	5.50	8.49	7.50	8.37	7.50

Figura 49 Patrones de entrenamiento parte 1

	Patrón 5		Patrón 6		Patrón 7		Patrón 8	
	inputs	*targets*	*inputs*	*targets*	*inputs*	*targets*	*inputs*	*targets*
cuerpo1	7.76	7.00	8.65	8.00	6.83	8.00	8.69	8.00
cuerpo2	10.15	7.50	9.90	7.50	9.01	7.50	10.98	7.50
cuerpo3	8.93	10.50	10.67	11.50	7.31	11.50	6.39	11.50
cuerpo4	8.75	6.00	6.65	6.00	9.27	6.00	8.21	6.00
cuerpo5	10.59	5.00	8.84	4.00	9.45	4.00	6.53	4.00
cuerpo6	7.43	6.00	8.35	6.00	9.74	6.00	10.81	6.00
muslo1	9.79	9.50	6.06	9.50	8.25	7.50	6.02	7.50
muslo2	9.77	7.50	7.69	7.50	6.42	7.50	9.87	7.50
muslo3	7.90	10.50	6.81	10.50	7.14	8.50	10.09	8.50
muslo4	8.84	7.50	9.97	7.50	10.57	7.50	10.34	7.50
muslo5	6.38	5.50	7.56	5.50	6.76	7.50	6.42	7.50
muslo5	6.27	7.50	8.64	7.50	10.13	7.50	8.00	7.50
muslo6								

Figura 50 Patrones de entrenamiento parte 2

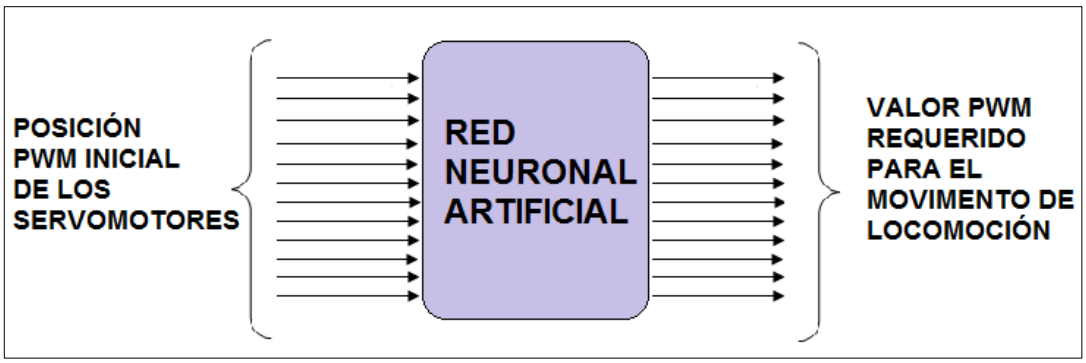

Figura 51 Esquema General de Patrones

3.5.4. Arquitectura de Red

El conjunto de entradas se relaciona con una capa oculta compuesta de 12 neuronas que se unen en forma *feedforward* (conexión hacia adelante), por tanto es una red neuronal multicapa. Son necesarias 12 salidas, una por cada servomotor. En las salidas aparecerá el valor PWM para actuar sobre los servomotores y provocar el desplazamiento. (Ver Figura 52).

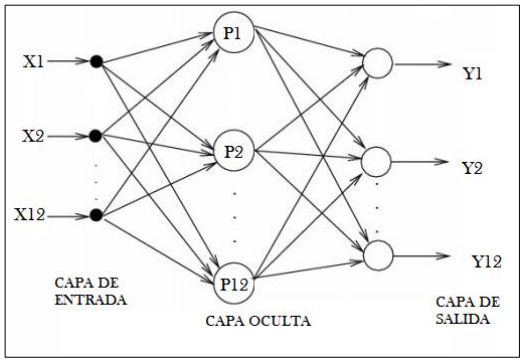

Figura 52 Arquitectura de Red

3.5.5. Función de Activación

La red diseñada presenta dos tipos de funciones de activación: en la capa oculta y en la capa de salida. Matlab permite la elección de las funciones de activación de entre las siguientes: Función sigmoide, sigmoidal logarítmica, lineal, signo o escalón. (Ver Figura 53). Para la capa oculta Matlab ocupa por defecto la función Sigmoide, porque es comúnmente usada en redes multicapa debido a que la mayoría de algoritmos de aprendizaje necesita una función derivable. La función lineal es usada por defecto para la capa de salida para reproducir siempre un valor analógico.

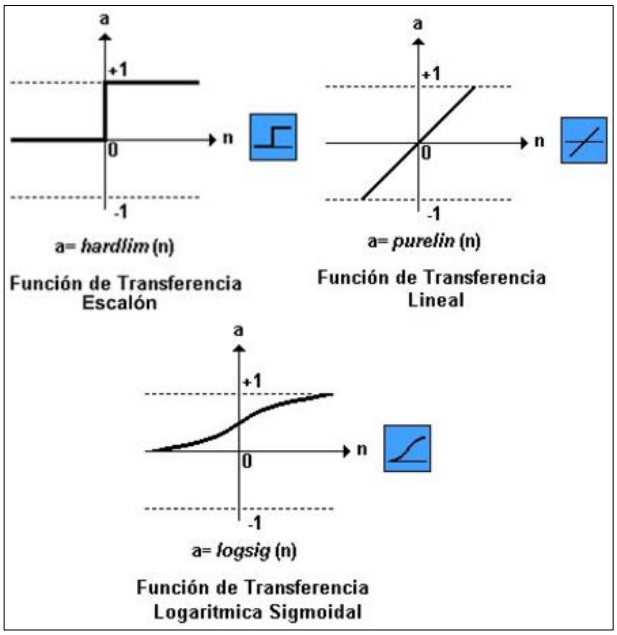

Figura 53 Funciones de transferencia en Matlab

Fuente: (Yinna Bautista, 2012)

Tabla 12

Funciones de transferencia de las capas de RNA

Capa	Función de transferencia
Oculta	Sigmoide
Salida	Lineal

3.5.6. Mecanismo de aprendizaje.

El entrenamiento o aprendizaje de la red neuronal consiste en presentar las entradas y sus respectivos *targets* u objetivos, para que la red vaya realizando un ajuste de sus salidas mediante la modificación de sus pesos y umbrales, todo esto para que el error disminuya. Los métodos de entrenamiento de red que se utilizaron son Retropropagación y Gradiente conjugada.

El método de Retropropagación (propagación del error hacia atrás), es un entrenamiento de redes multicapa que consiste en el aprendizaje de un conjunto predefinido de entradas y salidas dados. Por cada iteración los pesos son ajustados de forma que disminuya el error entre la salida deseada y la respuesta de la red.

Este método como primer paso aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado en las neuronas de salida con la salida que se desea obtener y se calcula un valor de error para cada neurona de salida. Como segundo paso, estos errores se transmiten hacia atrás, partiendo de la capa de salida hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada. (Marco Valencia, Cornelio Yáñez, 2006)

La regla delta generalizada permite acercar al mínimo de la función de error cuadrático mediante el descenso por el gradiente. La función de error suele tener un

gran número de mínimos locales y el algoritmo de aprendizaje puede quedar atrapado en un mínimo local no deseado, puesto que en él, el gradiente vale cero. Para prevenir que el algoritmo de aprendizaje quede atrapado en mínimos locales podemos hacer que los cambios de los pesos sinápticos dependan del gradiente medio de los puntos de un entorno, en lugar de depender del gradiente de un solo punto, pero dicha modificación suele requerir un gran esfuerzo computacional y no resulta eficiente. (J.Ortiz, 2009)

Tabla 13

Algoritmos de entrenamiento RNA

Función	Algoritmo
trainrp	Retropropagación (Gradiente conjugada)
traingd	Regla delta generalizada

3.5.7. Creación de la RNA en Matlab

Una vez determinado los parámetros de la RNA se crea el código en Matlab que representa los parámetros elegidos para este modelo. Para la creación de la red de tipo *feedforward* de dos capas se utiliza el comando *Newff.* La sintaxis de este comando es el siguiente:

$$net = newff(P, T, [S1\ S2 \ldots S(N-1)],\{TF1\ TF2 \ldots TFN\},BTF) \quad (3.1)$$

- P es la Matriz con los patrones de entrada
- T Matriz con las salidas deseadas
- S Tamaño de la capa i (sólo capas ocultas)
- Tfi Función de transferencia de la capa i, por defecto es *'tansig'* para las capas ocultas y '*purelin'* para la capa de salida.
- BTF Función de entrenamiento, por defecto *'trainlm'*.

Para ajustar los parámetros del entrenamiento se utilizan los siguientes comandos:

- *net.trainParam.epochs = 1000:* Permite ingresar el número de épocas para el entrenamiento en este caso está para 1000 épocas.
- *Net.trainParam.show = 50:* Permite visualizar el entrenamiento durante las primeras 50 épocas
- *net.trainParam.goal = 1e-24:* Es el valor del error medio cuadrático

Para realizar el entrenamiento de la red se utiliza el comando *train.* La sintaxis es la siguiente:

$$train(NET,P,T) \tag{3.2}$$

- NET Una red inicializada
- P Los patrones de entrada
- T Las salidas deseadas

3.5.8. Aplicación de RNA al Movimiento de caminata hacia Adelante

Se definen dentro del programa las entradas (matriz 12x8) y los *targets* (matriz 12x8). Posteriormente se crea la red. Los parámetros se configuran de la siguiente manera:

```
net = newff(inputs,targets,12,{‘tansig’ ‘purelin’},’trainrp’);
net.trainParam.show = 50;
net.trainParam.epochs = 1000;
net.trainParam.goal = 1e-24;
net = train(net,inputs,targets);
```

Con esto se crea y se entrena la red. Mediante el *Neural Network Training* se puede observar el entrenamiento de la red con el número de épocas en las cuales llegó al objetivo (Ver Figura 54).

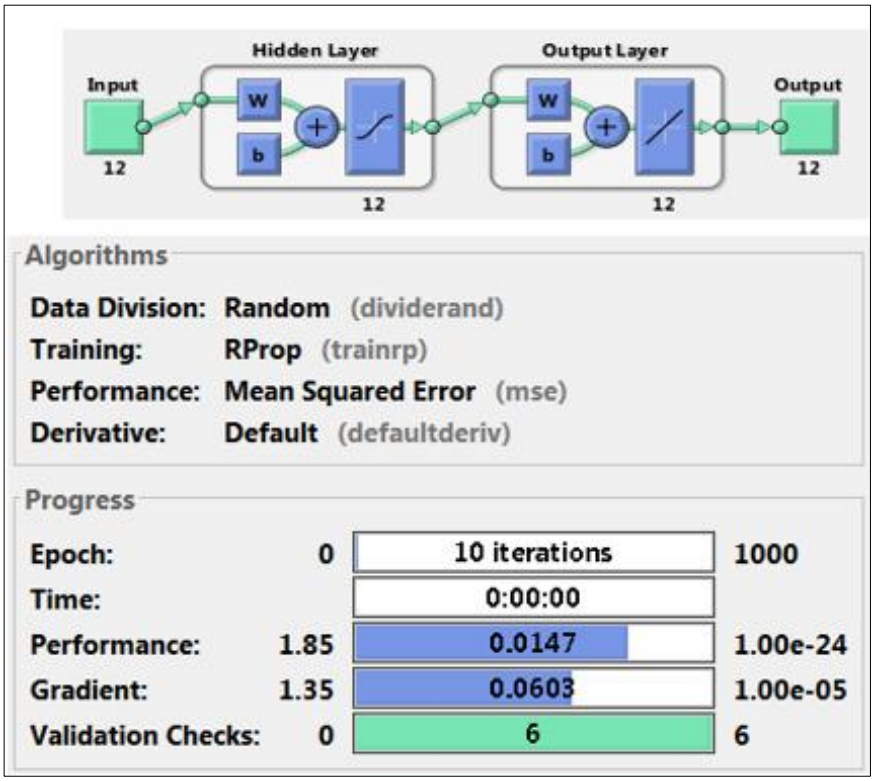

Figura 54 Entrenamiento RNA para locomoción del Robot

Para conocer el rendimiento del entrenamiento se utiliza la siguiente línea de código:

$$perf = perform(net,outputs,targets) \quad (3.3)$$

Este es un factor que indica el error cuadrático medio de las salidas con respecto a los *targets* durante el entrenamiento. Mientras más tienda a cero este valor, habrá realizado un mejor entrenamiento. Para este caso entregó un valor de 0.67.

Para crear un bloque en *Simulink*, que permita recrear el diseño de esta RNA se utiliza *gensim.* A este bloque se ingresan 12 entradas, las cuales contienen una secuencia de entradas aleatorias. Y se enlazan 12 salidas a los generadores PWM de la tarjeta STM32F4Discovery. Todo esto se hace para verificar que el modelo realizado realice el movimiento de locomoción hacia adelante. La programación en Matlab y el modelo hecho en *Simulink* completo se encuentra en **¡Error! No se encuentra el origen de la referencia.**.

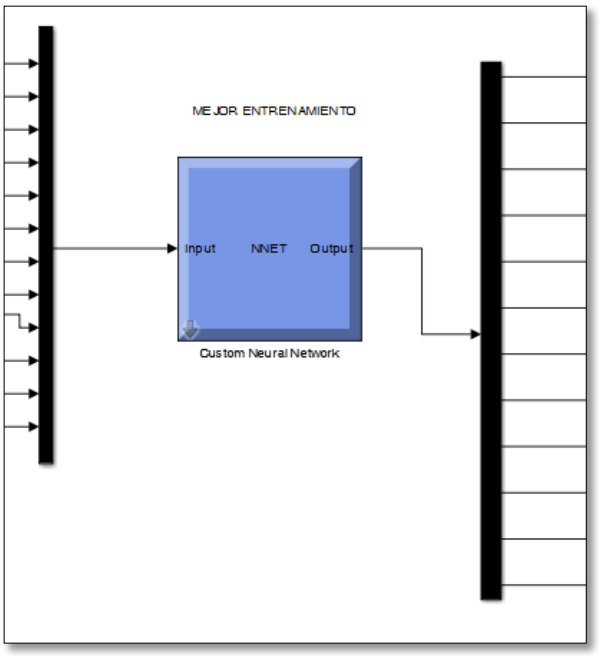

Figura 55 Bloque Neural Network

Tabla 14

Parámetros de red movimiento hacia adelante

Parámetro	**Valor**
Número de Entradas	12
Número de salidas	12
Capas Ocultas	1
Número de Neuronas Capa Oculta	12
Función transferencia capa oculta	*tansig*
Función de transferencia capa salida	*purelin*
Tipo de entrenamiento	*Trainrp* (Retropropagación)
Error máximo permitido	1e-24
Número de épocas	1000

3.5.9. Aplicación de RNA a Movimiento Giro hacia Derecha

Al igual que el movimiento hacia adelante se declara la matriz 12x8 con entradas aleatorias. En este caso se cambia la matriz de *targets*, que igual es una matriz de 12x8. Los valores necesarios para el giro se muestran en Tabla 11.

El código para configurar los parámetros de la red:

net = newff(inputs,targets,12,{'tansig' 'purelin'},'traingd');
net.trainParam.show = 50;
net.trainParam.lr = 50e-5;
net.trainParam.epochs = 10000;
net.trainParam.goal = 1e-24;

En este caso se cambió el tipo de entrenamiento, la regla delta generalizada. Método el cual requirió de más épocas para poder alcanzar un error cuadrático medio similar al entrenamiento de la red anterior. El valor que se obtuvo es de 0.69.

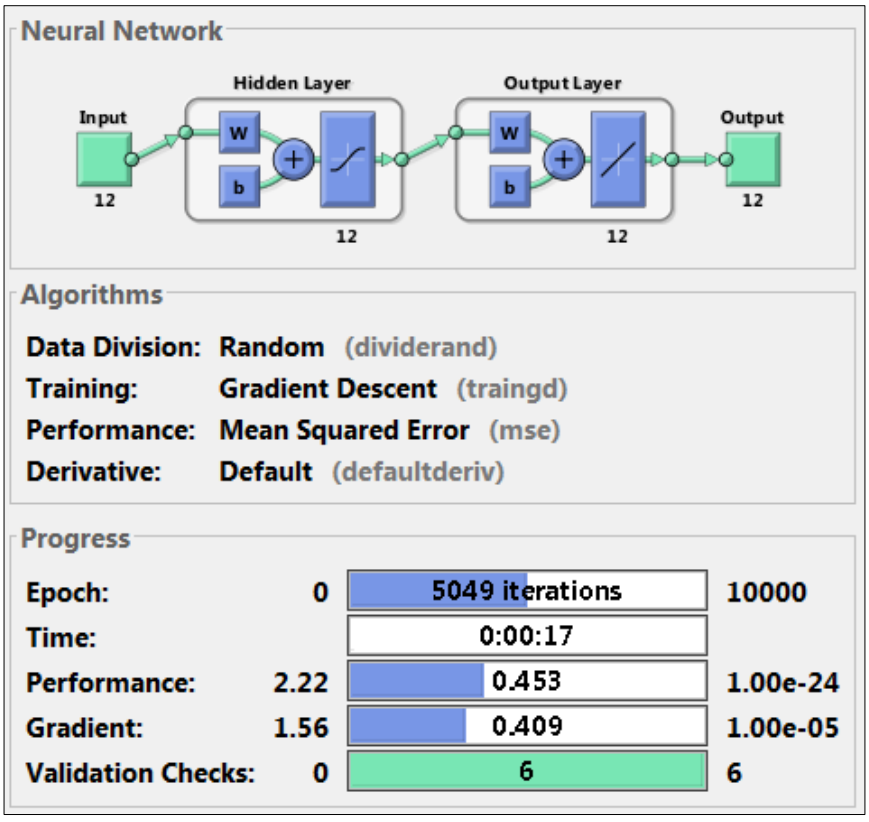

Figura 56 Entrenamiento RNA para Giro hacia Derecha del Robot

Tabla 15

Parámetros de Red Giro hacia la Derecha

Parámetro	**Valor**
Número de Entradas	12
Número de salidas	12
Capas Ocultas	1
Número de Neuronas Capa Oculta	12
Función transferencia capa oculta	*tansig*
Función de transferencia capa salida	*purelin*
Tipo de entrenamiento	*Traingd* (Regla delta generalizada)
Error máximo permitido	1e-24
Número de épocas	10000

3.6. Algoritmo de Control Difuso (Fuzzy Control)

3.6.1. Problemática del Control Difuso

El diseño del controlador difuso se realiza en la plataforma de Simulink de Matlab. El objetivo del controlador es estabilizar la posición de la base del Robot, es decir, utilizar sus seis extremidades para lograr que su base se encuentre en un ángulo de cero grados de inclinación. Se usó como entrada del controlador, el acelerómetro LIS3DSH que entrega los valores de *Roll* del ángulo.

Fue necesario realizar un controlador PI para reducir el error en estado estacionario y estabilizar al robot en una posición inicial, es decir, cero grados de inclinación, permitiendo al robot Hexápodo pararse y equilibrarse automáticamente sobre distintas superficies (Ver Figura 57)

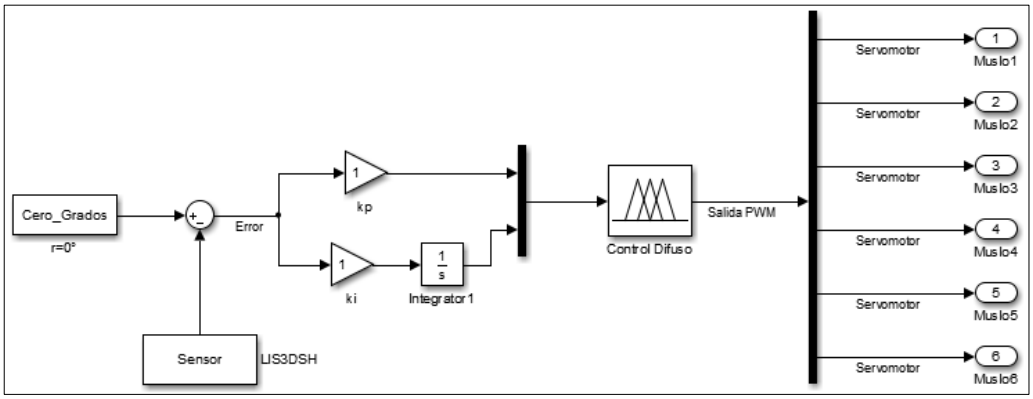

Figura 57 Diagrama del Control PI Difuso

3.6.2. Solución de Control Difuso

Se definen las funciones de pertenencia y la base de reglas para el equilibrio del robot hexápodo en el eje *Roll.* Los datos del eje *Roll* son mediante el acelerómetro LIS3DSH propio de la tarjeta STM32F4 DISCOVERY.

Una de las ventajas en realizar la programación de la tarjeta STM32F4 Discovery, es que presenta varios elementos programables, en el caso del control difuso, se encuentra el acelerómetro LIS3DSH que posee en la librería Waijung existe el bloque de programación respectivamente. Este acelerómetro genera valores numéricos de los ejes *Roll, Pitch y Yaw*. Para el control difuso se realiza el equilibrio del eje *Roll*, es decir, el equilibrio con respecto al eje longitudinal, siendo los valores numéricos positivos una rotación hacia la derecha, y valores numéricos negativos una rotación hacia la izquierda; tomando en cuenta que el punto de equilibrio longitudinal es igual a cero. Con estos valores numéricos de giro, se realiza el control difuso de 6 servomotores (servomotores de Muslos) para controlar el equilibrio en la superficie.

3.6.3. Fuzzificación

Se realiza el proceso por el cual se transforma cada dato de la señal de entrada a variables difusas calculando su grado de pertenencia de los conjuntos difusos en que se divide su rango.

Como entradas se tiene los valores numéricos del acelerómetro (*Roll*) (Ver la Figura 58) y la integral del error del sensor (*errorRoll)*.

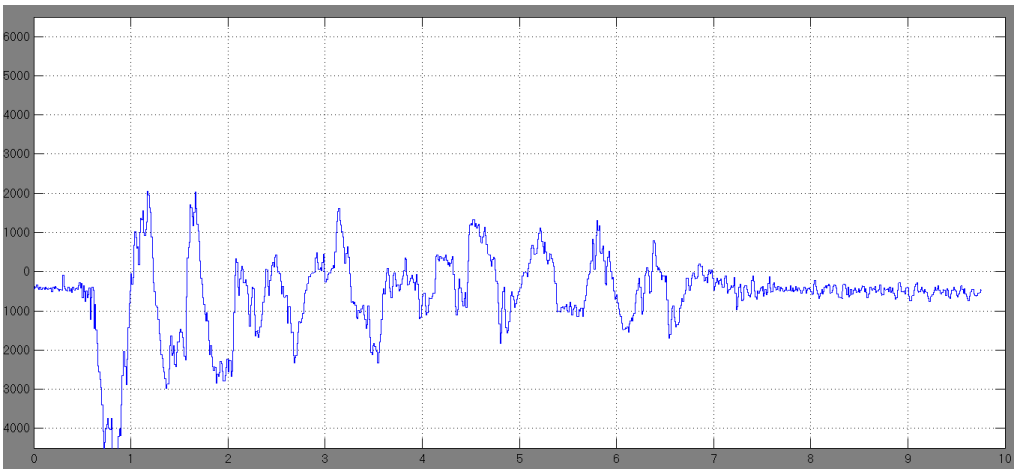

Figura 58 Señal de acelerómetro LIS3DSH

Los valores numéricos que entrega el acelerómetro LIS3DSH se encuentran en el rango de [-15000, 15000], cuya interpretación en ángulos está entre [-90°, 90°], siendo -15000 igual -90° y 15000 igual a 90°. Los valores del acelerómetro son lineales.

En la primera entrada que se denomina *Roll* se tiene como Universo de discurso el rango entre [-15000, 15000], siendo dividido en cinco conjuntos difusos (Ver Figura 59) cuyos valores lingüísticos son:

MI	Muy Izquierda cuya función de pertenencia es del rango [-15000, -8000,-4000], siendo -15000 a -8000 un grado de membresía de 1.
I	Izquierda cuya función de pertenencia es triangular del rango [-8000, -4000, 0].
C	Centro cuya función de pertenencia es trapezoidal del rango [-2000, -1000,1000 2000].
D	Derecha cuya función de pertenencia es triangular del rango [0, 4000, 8000].
MD	Muy Derecha cuya función de pertenencia es del rango [4000, 8000,15000], siendo 8000 a 15000 un grado de membresía de 1.

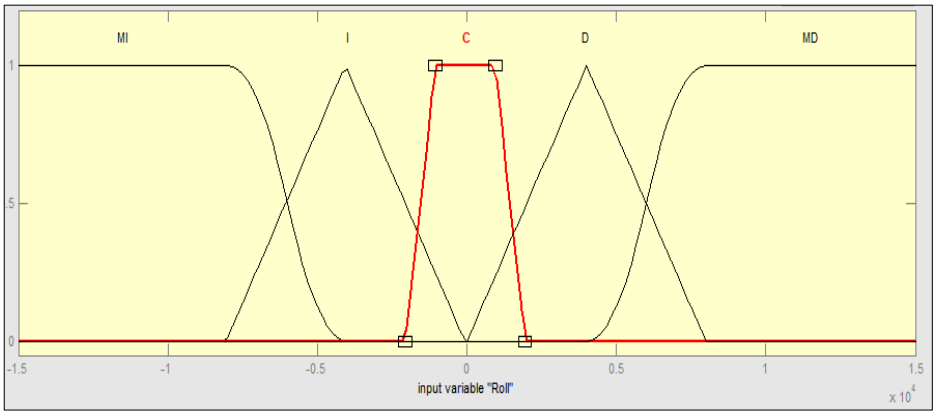

Figura 59 Conjuntos Difusos de la Entrada *Roll*

Como segunda entrada se tiene la integral del error (*errorRoll*) que como Universo de discurso se encuentra entre [-5000, 5000], siendo dividido en tres conjuntos difusos cuyos valores lingüísticos son:

EI Error Izquierda cuya función de pertenencia es del rango [-5000, -2000,0], siendo -5000 a -2000 un grado de membresía de 1.

EC Error Centro cuya función de pertenencia es trapezoidal del rango [-1000, -500, 500, 1000].

ED Error Derecha cuya función de pertenencia es del rango [0, 2000,5000], siendo 2000 a 5000 un grado de membresía de 1.

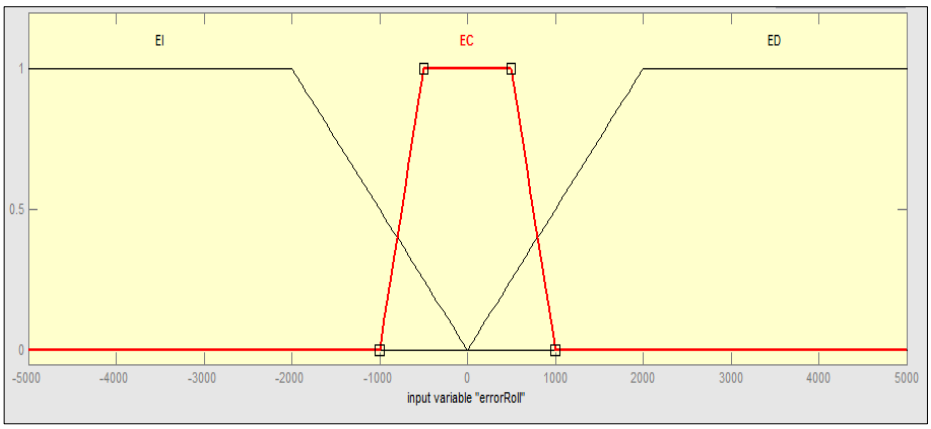

Figura 60 Conjuntos Difusos de la Entrada: Integral del error (*ErrorRoll*)

3.6.4. Base de Reglas

La Tabla 16 muestra la base de reglas para la realización del equilibrio en el eje Roll. Existen 5 conjuntos difusos en la entrada *Roll* y 3 conjuntos difusos en la entrada *ErrorRoll*, por lo que se tiene 5 x 3 = 15 reglas posibles por cada servomotor, es decir que al controlar 6 servomotores se podría obtener un total de 90 reglas.

Tabla 16

Base de Reglas

M*		***ErrorRoll***		
		EI	**EC**	**ED**
Roll	**MI**	MIM*	MIM*	IM*
	I	MIM*	IM*	CM*
	C	IM*	CM*	DM*
	D	CM*	DM*	MDM*
	MD	DM*	MDM*	MDM*

* Representa el número del servomotor (1-6)

3.6.5. Defuzzificación

Luego de la evaluación de la base de reglas para cada servomotor, se obtiene la respuesta para cada una de ellas. Existen 6 salidas donde cada una de ellas controla un servomotor (Muslos) el cual cada salida presenta diferentes universos de discurso con sus respectivos conjuntos difusos.

Cada servomotor presenta diferentes valores PWM que son ángulos entre -20° a 20° con respecto al eje longitudinal. Los valores de cada servomotor en el programa de Matlab se observa en Tabla 17.

Tabla 17

Valores PWM de Servomotores Muslos

ServoMotores	Valores PWM		
	-20°	**0°**	**20°**
Muslo1	5	8.5	12
Muslo2	5	8.5	12
Muslo3	6	9.5	13
Muslo4	3.5	7	10.5
Muslo5	4	7.5	11
Muslo6	4	7.5	11

El objetivo de los valores de salida para cada servomotor de muslos, es permanecer a la base del robot de forma paralela al eje longitudinal sin importar el ángulo de la superficie (Ver Figura 61).

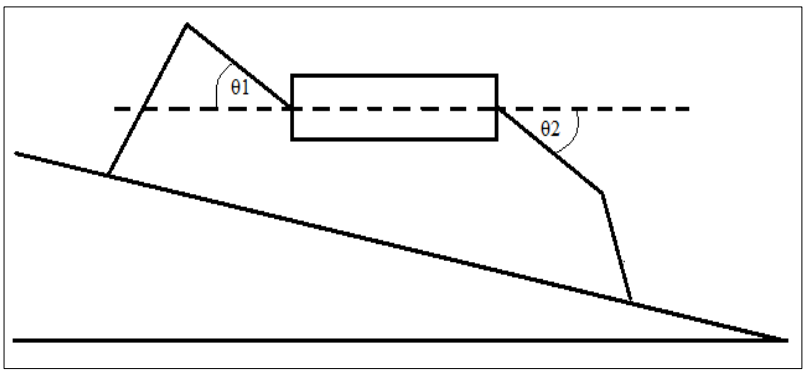

Figura 61 Inclinación del Robot Hexápodo en Superficies

Los valores lingüísticos de los conjuntos difusos para cada servomotor son:

MDM1 Muy Derecha de Muslo1.

DM1 Derecha de Muslo1.

CM1 Centro de Muslo1.

IM1 Izquierda de Muslo1.

MIM2 Muy Izquierda de Muslo2.

MDM2	Muy Derecha de Muslo2.
DM2	Derecha de Muslo2.
CM2	Centro de Muslo2.
IM2	Izquierda de Muslo2.
MIM2	Muy Izquierda de Muslo2.
MDM3	Muy Derecha de Muslo3.
DM3	Derecha de Muslo3.
CM3	Centro de Muslo3.
IM3	Izquierda de Muslo3.
MIM3	Muy Izquierda de Muslo3.
MDM4	Muy Derecha de Muslo4.
DM4	Derecha de Muslo4.
CM4	Centro de Muslo4.
IM4	Izquierda de Muslo4.
MIM4	Muy Izquierda de Muslo4.
MDM5	Muy Derecha de Muslo5.
DM5	Derecha de Muslo5.
CM5	Centro de Muslo5.
IM5	Izquierda de Muslo5.
MIM5	Muy Izquierda de Muslo5.
MDM6	Muy Derecha de Muslo6.
DM6	Derecha de Muslo6.
CM6	Centro de Muslo6.

IM6 Izquierda de Muslo6.

MIM6 Muy Izquierda de Muslo6.

Los conjuntos difusos de cada servomotor se encuentran en **¡Error! No se encuentra el origen de la referencia.**.

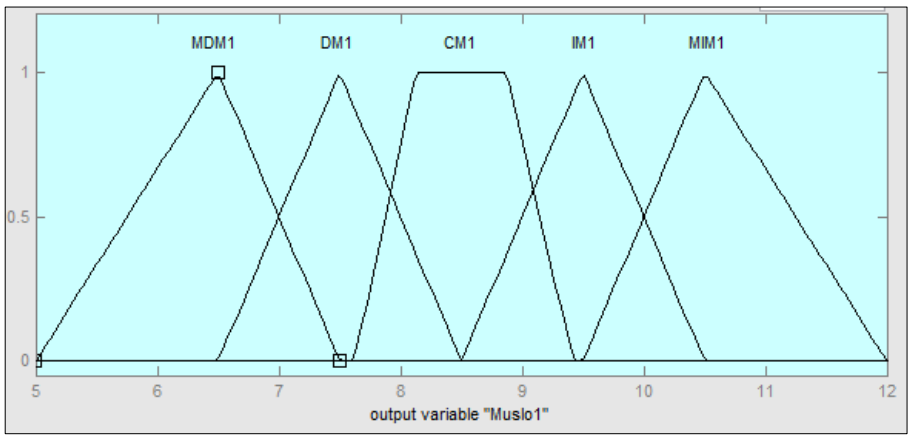

Figura 62 Conjuntos Difusos de Salida de Muslo1

Para la comprobación del ángulo del inclinación *Roll* se utilizan dos LED propios de la tarjeta STM32F4 Discovery que muestran el lado de inclinación, siendo derecha el LED de color Rojo e Izquierda el LED de color Verde. Si el robot se encuentra equilibrado, los LED Rojo y Verde permanecerán apagados. El programa de funcionamiento de los LED con el acelerómetro se observa en la Figura 63.

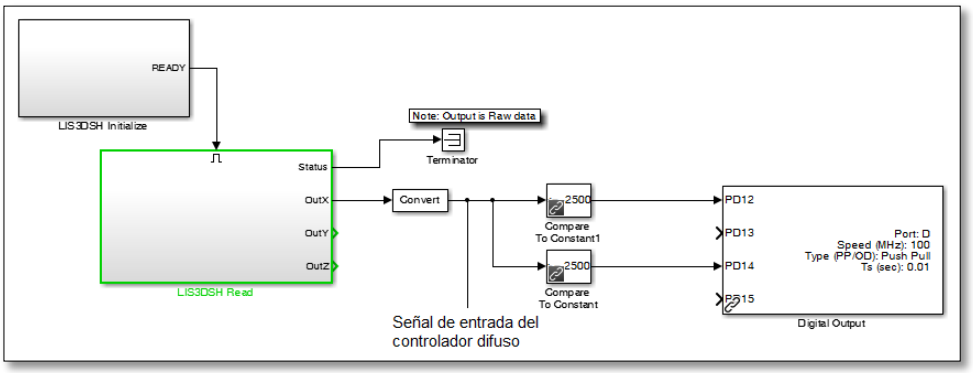

Figura 63 Bloques de Programación de encendido de LEDs

En Figura 65 se observa el diagrama de bloques del control difuso implementado en Matlab. El acelerómetro utiliza un bloque de filtraje digital para que la señal de

entrada del bloque de control difuso (Fuzzy Logic Controller) no capte valores de ruido que podría tener el acelerómetro y obtener un mejor control.

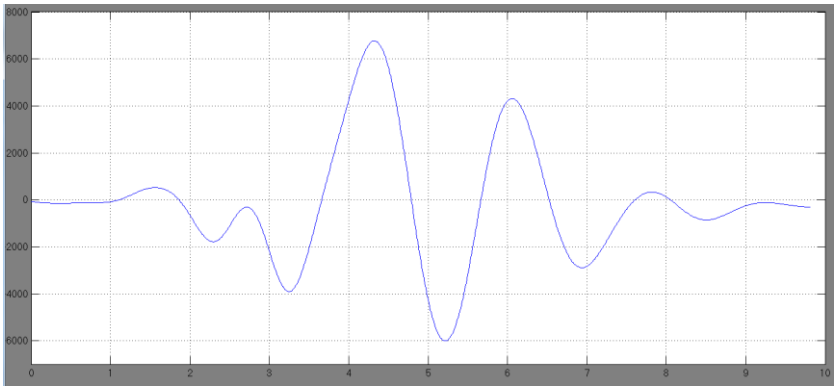

Figura 64 Señal del acelerómetro LIS3DSH con filtraje digital

Para reducir el error en estado estacionario y estabilizar al robot en una posición inicial, es decir, cero grados de inclinación con respecto al eje longitudinal, se realizó un controlador Proporcional Integral Difuso (PI Difuso). Este tipo de controlador permite que el estado estacionario presente un error de cero y así evitar las posibles oscilaciones en la posición de equilibrio o setpoint. La configuración del controlador se observa en Figura 65.

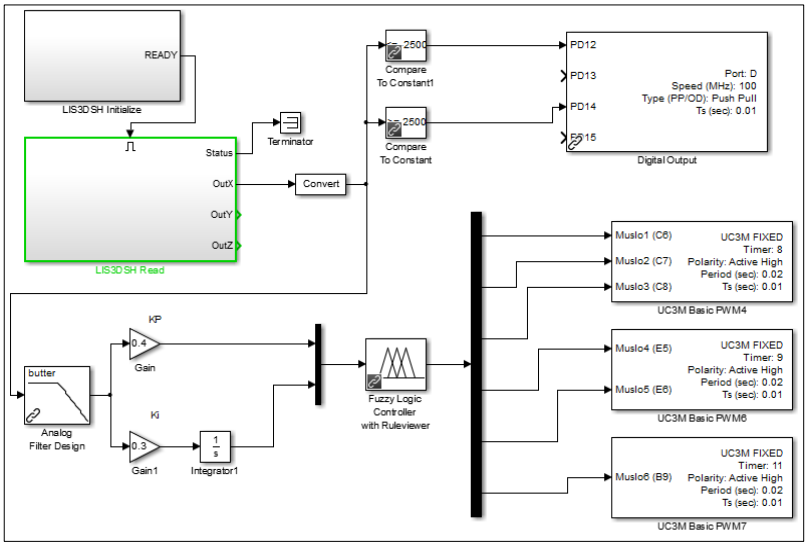

Figura 65 Control Difuso en Simulink de Matlab

Los valores de Ki y Kp se obtuvieron mediante prueba y error. Los resultados de variar estos parámetros se encuentran en el Capítulo IV de Pruebas y Resultados.

3.7. Algoritmo Generador Central de Patrones (GCP)

En esta parte se detalla el modelo matemático de un GCP locomotor de una araña y la forma de aplicarla al robot hexápodo. Además se recreó este GCP locomotor mediante señales sinusoidales.

3.7.1. Problemática del GCP.

La propuesta de los generadores centrales de patrones es la recreación de movimientos basados en la biología de los seres vivos aplicados a robots. Por tanto, se planteó crear nuevos movimientos al robot hexápodo, más realista y muy semejante a los que presentan los seres vivos como en este caso sería una araña.

3.7.2. Solución de Generador Central de Patrones

Construir una CNN con cuatro neuronas mediante Matlab, que permita generar un patrón oscilatorio para la locomoción del robot hexápodo mediante el accionamiento de 18 servomotores.

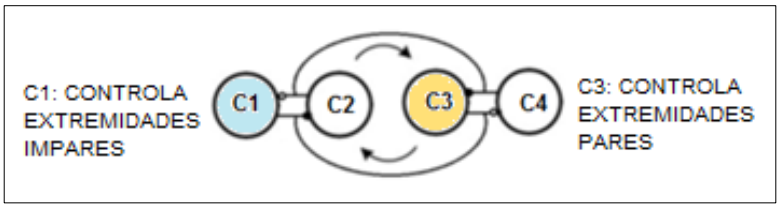

Figura 66 Propuesta de Solución de GCP

3.7.3. Diseño de un Generador Central de Patrones.

Para realizar el generador central de patrones para la locomoción se parte del modelo matemático propuesto por Chua y Yang para una Red Neuronal Celular de dos celdas (CNN), colocando un valor de β=0 para que no exista retroalimentación y la red celular sea autónoma (Elizabeth Sedeño Bustos, 2011). Trasladando esto al espacio de estados queda de la siguiente manera:

$$\dot{x}1 = -x1 + (1 + \mu) * y1 - s * y2 + i1 \tag{3.4}$$

$$\dot{x}2 = -x2 + (1 + \mu) * y2 - s * y1 + i2 \tag{3.5}$$

Y la función de salida será una implementación lineal de la función de saturación.

$$y_{ij}(t) = \frac{1}{2}(|x_{ij}(t) + 1| - |x_{ij}(t) - 1|) \tag{3.6}$$

De donde:

x Es el estado de la neurona

μ Es la entrada del sistema.

i Es el término de corriente de polarización (bias).

y Es la salida de la neurona.

Dadas las ecuaciones en el espacio de estados, se asignan los siguientes valores a las constantes como menciona Paolo Arena y Luigi Fortuna (Paolo Arena, Luigi Fortuna, 2002), para que exista una dinámica lenta-rápida.

$$\mu = 0.5, s = 1, i1 = -0.3, i2 = 0.3 \tag{3.7}$$

Trasladando este modelo matemático a *Simulink*, permite visualizar el comportamiento de las células en función del tiempo.

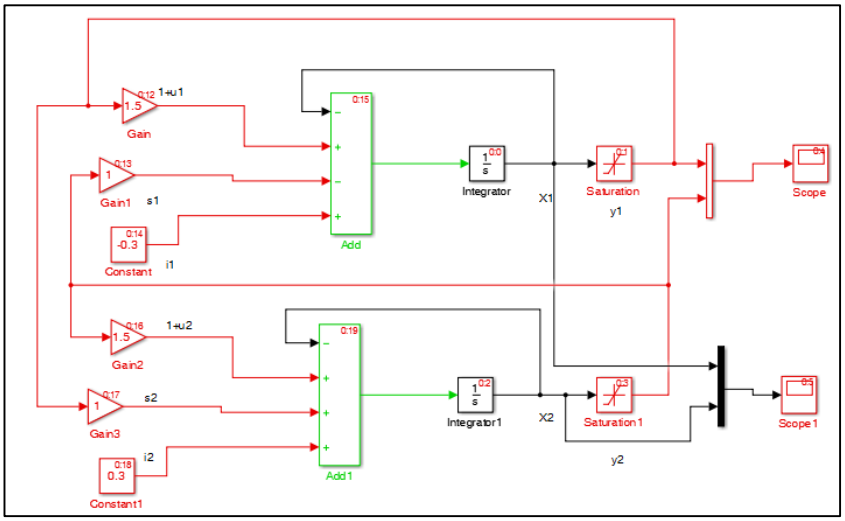

Figura 67 Simulación de Red Neuronal Celular en Simulink

El comportamiento de la célula $x1$ muestra valores $V_p = 1.8\ voltios$ a $V_p = -2.8\ voltios$. Mientras que la célula $x2$ muestra valores $V_p = 2.2\ voltios$ a $V_p = -2\ voltios$.

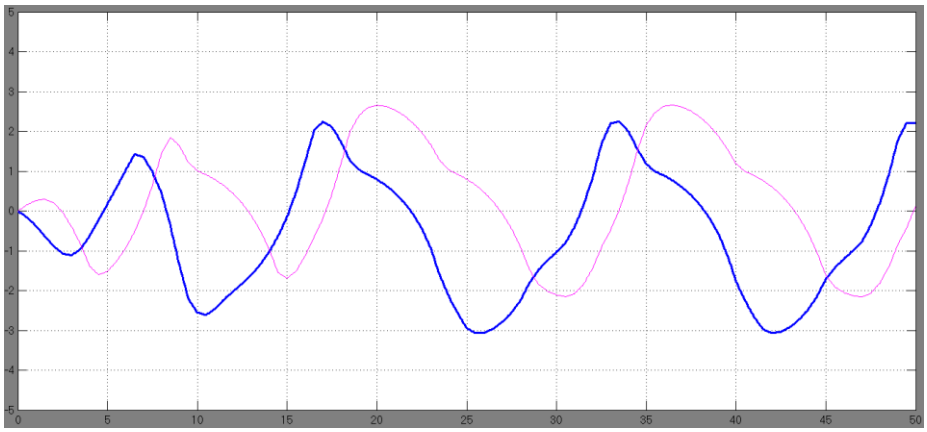

Figura 68 Estados de las neuronas x1 y x2

Las salidas $y1$ y $y2$ son una función de saturación respecto de $x1$ y $x2$, delimitadas en ± 1.

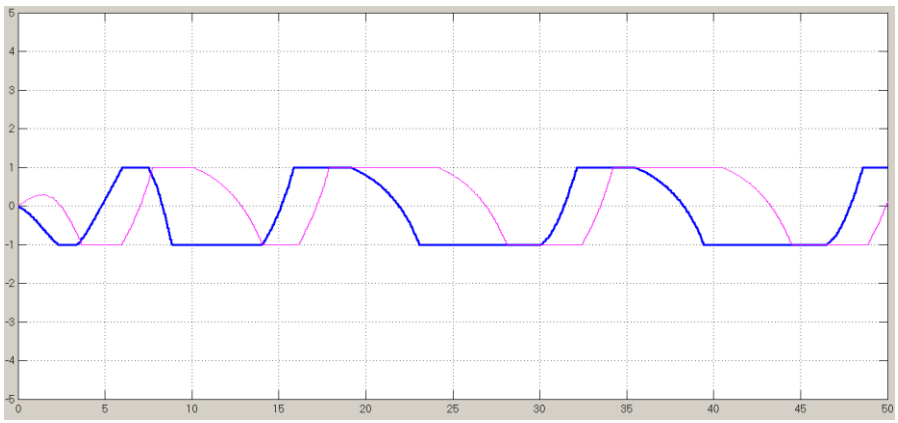

Figura 69 Salidas y1 y y2 de cada celda

3.7.4. GCP Locomotor

Para que las señales de la Red Neuronal Celular (CNN) sirvan como GCP locomotor se debe establecer un desfase de 180º entre cada señal de salida. Esto se debe al tipo de locomoción que posee el robot hexápodo. El movimiento *tripod* necesita tres articulaciones siempre en el suelo para permanecer estable el resto de la

estructura. Por tanto una salida ($y1$) debe controlar los movimientos de las articulaciones pares mientras la otra salida ($y2$) actúa para las impares.

Para crear este desfase entre dos salidas $y1$ y $y2$, Julio Pérez menciona en su tesis, que se puede crear una red neuronal celular a partir de varias redes más pequeñas.

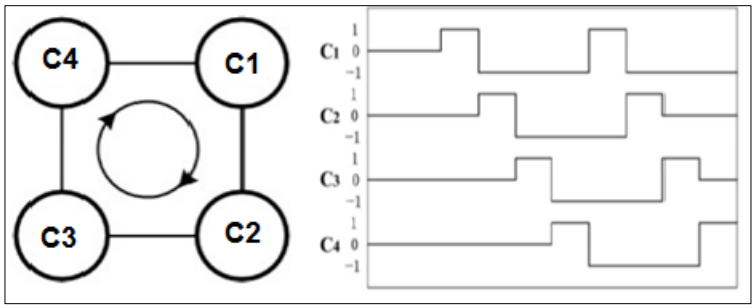

Figura 70 CNN con sus patrones de oscilación.

Fuente: (Julio Pérez Machorro, 2009)

Kraimon Maneesilp, propone la siguiente conexión para lograr dicho desfase:

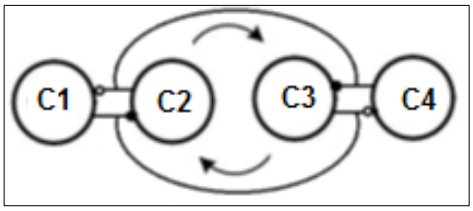

Figura 71 Conexión en anillo

Fuente: (Kraimon Maneesilp, Boonchana Purahong, Pitikhate Sooraksa, 2004)

Las conexiones que existen entre cada CNN (C1 y C2) con (C3 y C4) son sinapsis que permiten hacer una conexión para una CNN más grande. A estas sinapsis se le da una ganancia para crear el desfase (Kraimon Maneesilp, Boonchana Purahong, Pitikhate Sooraksa, 2004). Dicho valor de ganancia para cada sinapsis es de $e_1 = -0.5$ y $e_1 = -0.6$

$$\dot{x}1 = -x1 + (1+\mu) * y1 - s * y2 + i1 - e_2 * y3 \quad (3.8)$$

$$\dot{x}2 = -x2 + (1+\mu) * y2 - s * y1 + i2 \quad (3.9)$$

$$\dot{x}3 = -x3 + (1+\mu) * y3 - s * y4 + i3 - e_1 * y1 \quad (3.10)$$

$$\dot{x}4 = -x4 + (1+\mu) * y4 - s * y3 + i4 \quad (3.11)$$

Trasladando este modelo a *Simulink*, permite observar el desfase de las señales que van a controlar los movimientos del robot hexápodo.

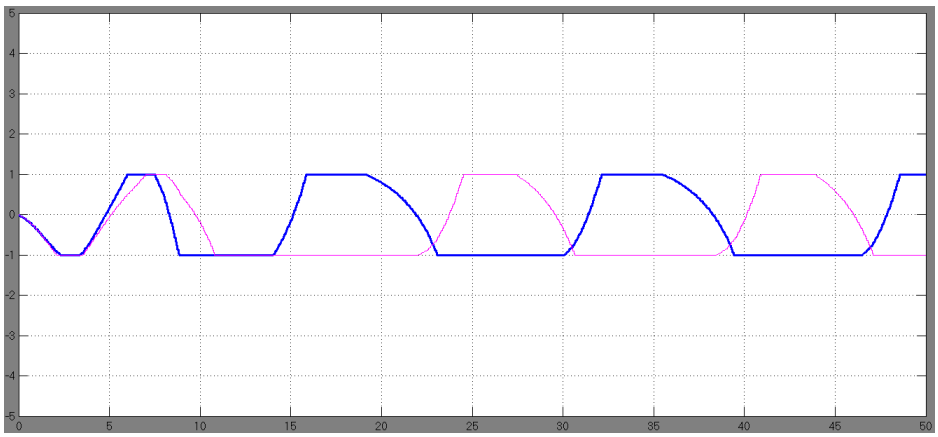

Figura 72 Salidas y1 y y3 de CNN (desfasadas 180°)

3.7.5. Aplicación del Generador de Patrones al Robot Hexápodo.

Para que las señales $y1$ y $y3$ controlen las articulaciones impares y pares respectivamente, se realizó una conversión Analógica a Digital mediante la ecuación de la recta para cada servomotor.

Siendo la ecuación de la recta $y = mx + b$

Los servomotores denominados cuerpo1, cuerpo2 y cuerpo3 poseen los siguientes datos para crear su ecuación:

Tabla 18

Valores para ecuación cuerpo 1-3

	X1 (salida de saturación)	X2 (salida de saturación)	Y1 (valor PWM estado inicial)	Y2 (valor PWM estado adelante)
Cuerpo1	-1	1	7	8
Cuerpo2	-1	1	7.5	8.5
Cuerpo3	-1	1	10.5	11.5

Con estos datos se obtiene el valor de la pendiente:

$$m = \frac{y2-y1}{x2-x1} = \frac{7-8}{1-(-1)} = 0.5 \quad (3.12)$$

Con esto se encuentra el valor de *b*, quedando sus ecuaciones de la siguiente manera:

$$y(cuerpo1) = 0.5x + 7.5 \quad (3.13)$$

$$y(cuerpo2) = 0.5x + 8 \quad (3.14)$$

$$y(cuerpo3) = 0.5x + 11 \quad (3.15)$$

Para los servomotores de cuerpo4, cuerpo 5 y cuerpo6 se tiene:

Tabla 19

Valores para ecuación cuerpo 4-6

	X1 (salida de saturación)	**X2 (salida de saturación)**	**Y1 (valor PWM estado inicial)**	**Y2 (valor PWM estado adelante)**
Cuerpo4	-1	1	6	5
Cuerpo5	-1	1	5	4
Cuerpo6	-1	1	6	5

El valor de la pendiente para estos servomotores es:

$$m = \frac{y2-y1}{x2-x1} = \frac{5-4}{1-(-1)} = -0.5 \quad (3.16)$$

Por tanto sus ecuaciones son:

$$y(cuerpo4) = -0.5x + 5.5 \quad (3.17)$$

$$y(cuerpo5) = -0.5x + 4.5 \quad (3.18)$$

$$y(cuerpo6) = -0.5x + 5.5 \quad (3.19)$$

Tabla 20

Valores para ecuación muslo 1-3

	X1 (salida de saturación)	X2 (salida de saturación)	Y1 (valor PWM estado inicial)	Y2 (valor PWM estado adelante)
Muslo1	-1	1	7.5	9.5
Muslo2	-1	1	7.5	9.5
Muslo3	-1	1	8.5	10.5

El valor de la pendiente para estos servomotores es:

$$m = \frac{y2-y1}{x2-x1} = \frac{9.5-7.5}{1-(-1)} = 1 \quad (3.20)$$

Por tanto sus ecuaciones son:

$$y(muslo1) = x + 8.5 \quad (3.21)$$

$$y(muslo2) = x + 8.5 \quad (3.22)$$

$$y(muslo3) = x + 9.5 \quad (3.23)$$

Y por último los servomotores muslo4, muslo5 y muslo6, tienen estos valores:

Tabla 21

Valores para ecuación muslo 4-6

	X1 (salida de saturación)	X2 (salida de saturación)	Y1 (valor PWM estado inicial)	Y2 (valor PWM estado adelante)
Muslo4	-1	1	7.5	5.5
Muslo5	-1	1	7.5	5.5
Muslo6	-1	1	7.5	5.5

El valor de la pendiente para estos servomotores es:

$$m = \frac{y2-y1}{x2-x1} = \frac{5.5-7.5}{1-(-1)} = -1 \quad (3.24)$$

Por tanto sus ecuaciones son:

$$y(muslo4) = -x + 6.5 \quad (3.25)$$

$$y(muslo5) = -x + 6.5 \quad (3.26)$$

$$y(muslo6) = -x + 6.5 \quad (3.27)$$

Estas ecuaciones son introducidas a continuación de las señales de salida $y1$ y $y2$ según corresponda, y estas a su vez a los bloques de PWM de cada servomotor. Esquema completo se encuentra en el **¡Error! No se encuentra el origen de la referencia.**.

3.7.6. Creación GCP locomotor mediante señales analógicas

Las señales generadas por el modelo propuesto por Chua y Yang son analógicas, por lo que se puede crear otro patrón mediante el uso de señales Sinusoidales. El objetivo de estas señales es que se asemejen a las antes diseñadas y permitan ser modificadas en amplitud y frecuencia sin una matemática compleja.

Las características de estas señales son:

Tabla 22

Características de las Sinusoidales

Sinusoidal	Amplitud		Frecuencia	Fase
Señal 1	2.6 v vp	1.6 vp a -3.6	1 Hz	0°
Señal 2	2.6 v vp	1.6 vp a -3.6	1 Hz	180°

Con estos valores se crea el modelo en *Simulink*, el cual va a ser embebido en la tarjeta STM32F4 Discovery.

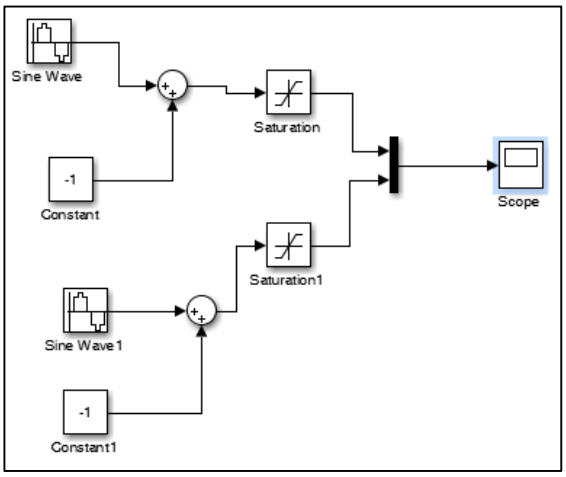

Figura 73 Modelo Simulink sinusoidales

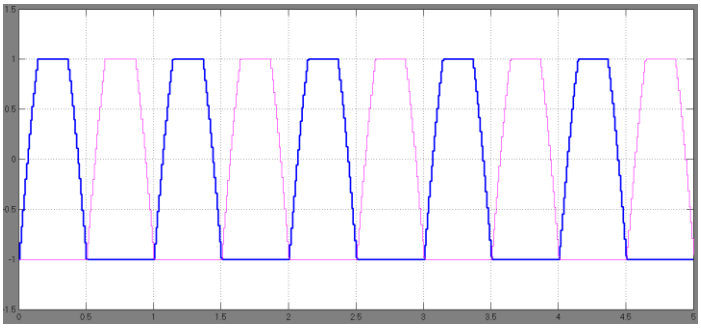

Figura 74 Gráficas voltaje vs tiempo de sinusoidales

Al igual que el modelo anterior, estas señales permiten conectarse con la ecuación de la recta de cada motor para lograr su locomoción:

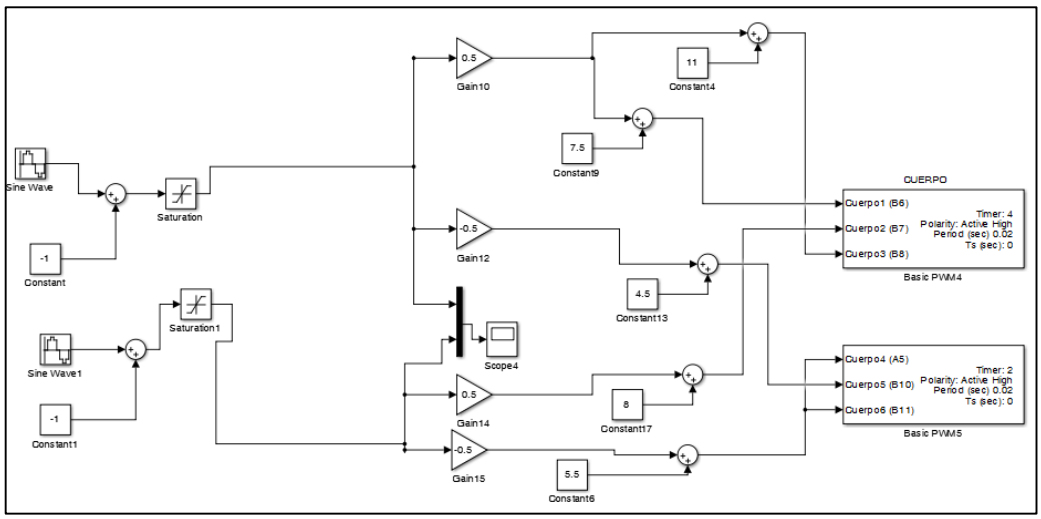

Figura 75 Modelo Simulink GCP mediante sinusoidales

CAPÍTULO IV

PRUEBAS Y RESULTADOS

4.1. Descripción de las pruebas realizadas.

Es este capítulo se especifica las pruebas realizadas y resultados obtenidos del robot hexápodo, usando los algoritmos antes diseñados sobre diferentes entornos.

La primera prueba fue realizar una comparación entre el movimiento de Secuencia hacia Adelante usando solo valores de PWM y el algoritmo de locomoción hecho mediante GCP. La comparación es de velocidad, es decir concluir el algoritmo que realiza más rápido el desplazamiento del robot sobre varias superficies como vidrio, cemento y madera. Otra prueba entre estos algoritmos será comprobar su funcionalidad sobre una superficie inclinada.

Para la parte de RNA, se evaluaron los parámetros de la RNA en función del error cuadrático medio.

En el controlador PI Difuso se varían los coeficientes Ki y Kp para la encontrar la mejor respuesta del controlador.

Por último se realizará una comparación del movimiento de locomoción del robot hexápodo con GCP y con Redes Neuronales.

4.2. Caminata sobre Superficie Lisa.

En esta parte la superficie escogida fue el vidrio, realizando desplazamiento de 150 cm. El período del movimiento de las 6 extremidades es de 2 segundos para ambos algoritmos. Los valores obtenidos se muestran en la Tabla 23.

Tabla 23

Prueba de Velocidad sobre Vidrio

Algoritmo	Superficie	Desplazamiento	Tiempo	Velocidad
Secuencia PWM	Vidrio	150 cm	22 s	6.8 cm/s
GCP	Vidrio	150 cm	20 s	7.5 cm/s

Se puede evidenciar que el algoritmo mediante GCP permite una mayor velocidad para el desplazamiento del robot y el caucho sobre sus patas permite tener un agarre al suelo, caso contrario no podría realizar el movimiento sobre esta superficie.

Figura 76 Vista Frontal de Prueba sobre Vidrio

Figura 77 Vista Superior de Prueba sobre Vidrio

4.3. Caminata sobre superficies irregulares.

El robot hexápodo realizó la caminata sobre una superficie de madera sin inclinación, Se obtuvieron los siguientes resultados:

Tabla 24

Prueba de Velocidad sobre Madera

Algoritmo	Superficie	Desplazamiento	Tiempo	Velocidad
Secuencia PWM	Madera	150 cm	20 s	7.5 cm/s
GCP	Madera	150 cm	17 s	8.8 cm/s

Sobre esta superficie se evidenció un mejor agarre del caucho sobre la madera y de igual manera el algoritmo mediante GCP resultó ser más veloz.

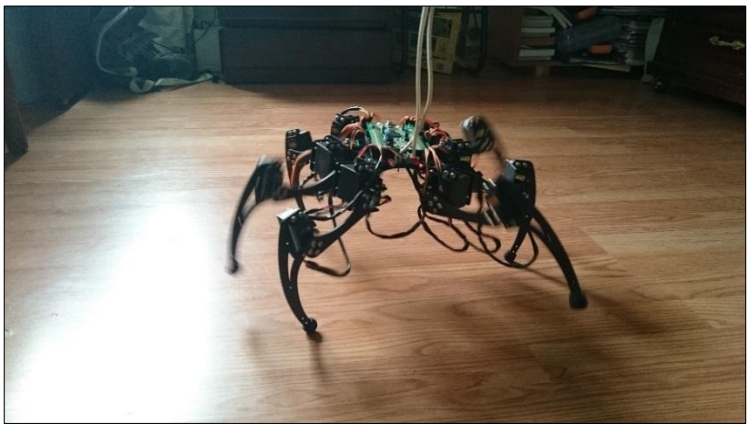

Figura 78 Vista Frontal de Prueba sobre Madera

También realizó la caminata sobre una superficie de cemento con una inclinación obteniendo los siguientes resultados:

Tabla 25

Prueba de Velocidad sobre Cemento

Algoritmo	Superficie	Desplazamiento	Tiempo	Velocidad
Secuencia PWM	Cemento (subida)	150 cm	28 s	5.35 cm/s
GCP	Cemento (subida)	150 cm	27 s	5.55 cm/s
Secuencia PWM	Cemento (bajada)	150 cm	18 s	8.33 cm/s
GCP	Cemento (bajada)	150 cm	15 s	10 cm/s

El robot hexápodo logró concluir las pruebas satisfactoriamente. En la subida se evidencia el peso de las patas traseras, mientras que en la bajada el robot tiende a inclinarse hacia adelante.

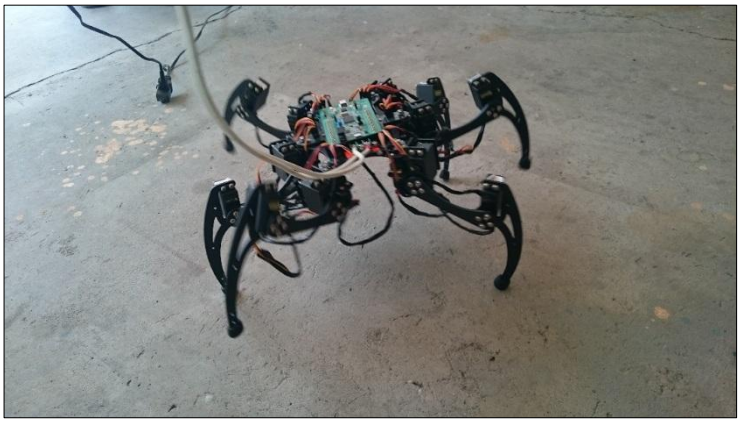

Figura 79 Vista Frontal de Prueba sobre Cemento inclinación Arriba

Figura 80 Vista Frontal de Prueba sobre Cemento inclinación Abajo

4.4. Pruebas de entrenamiento RNA

La primera prueba es variar el número de neuronas de la capa oculta. Utilizando por defecto el método de entrenamiento el de Retropropagación.

Tabla 26

Prueba de número de neuronas en capa oculta

Número de Neuronas	Error cuadrático medio
6	0.3178
12	0.1325
18	0.1286

Esto indica que si se aumenta el número de neuronas de la capa oculta mejora el entrenamiento de la RNA. Una vez determinado una capa oculta de 18 neuronas se procede a verificar los algoritmos de entrenamiento

Tabla 27

Prueba del Método de Entrenamiento

Método de entrenamiento	Error cuadrático medio
Retropropagación	0.1027
Regla delta generalizada	0.9105

El método de Retropropagación tiene un mejor rendimiento que la regla delta generalizada tomando en cuenta el mismo número de épocas (1000 épocas). La siguiente prueba es verificar la respuesta del entrenamiento neuronal variando las funciones de activación.

Tabla 28

Prueba de Funciones de Activación

Capa Oculta	Capa de salida	Error cuadrático medio
tansig	purelin	0.1138
tansig	tansig	0.0904
purelin	purelin	0.3329
hardlim	hardlim	1.0990

La función de activación sigmoide tanto para la capa oculta como para la capa de salida entrega un mejor rendimiento del entrenamiento RNA.

Todas estas pruebas del entrenamiento de red Neuronal permitieron determinar los parámetros necesarios para obtener la mejor respuesta al momento de emular los movimientos de locomoción del robot Hexápodo. Los parámetros se muestran en la Tabla 29.

Tabla 29

Resultados de Pruebas en entrenamiento RNA

Parámetro	**Valor**
Número de Entradas	12
Número de salidas	12
Capas Ocultas	1
Número de Neuronas Capa Oculta	18
Función transferencia capa oculta	*tansig*
Función de transferencia capa de salida	*tansig*
Tipo de entrenamiento	*Trainrp* (Retropropagación)
Error máximo permitido	1e-48
Número de épocas	1000

La prueba final sobre entrenamiento se realizó aplicando el algoritmo sobre el robot hexápodo para que camine sobre una superficie plana de cerámica obteniendo resultados satisfactorios.

Figura 81 Prueba Final de Entrenamiento RNA

4.5. Pruebas del controlador PI difuso

Se realizó varios ajustes de forma experimental a los coeficientes proporcional e integral. En la Tabla 30 muestra los diferentes valores experimentados de los coeficientes y su respectivo resultado.

Tabla 30

Resultados del Controlador Difuso

Kp	Ki	cResultado
0.4	0.3	Sistema estable sin error
1	0.3	Sistema oscilatorio (establece después de un tiempo)
0.4	1	Sistema inestable
0.5	0.5	Sistema estable con error

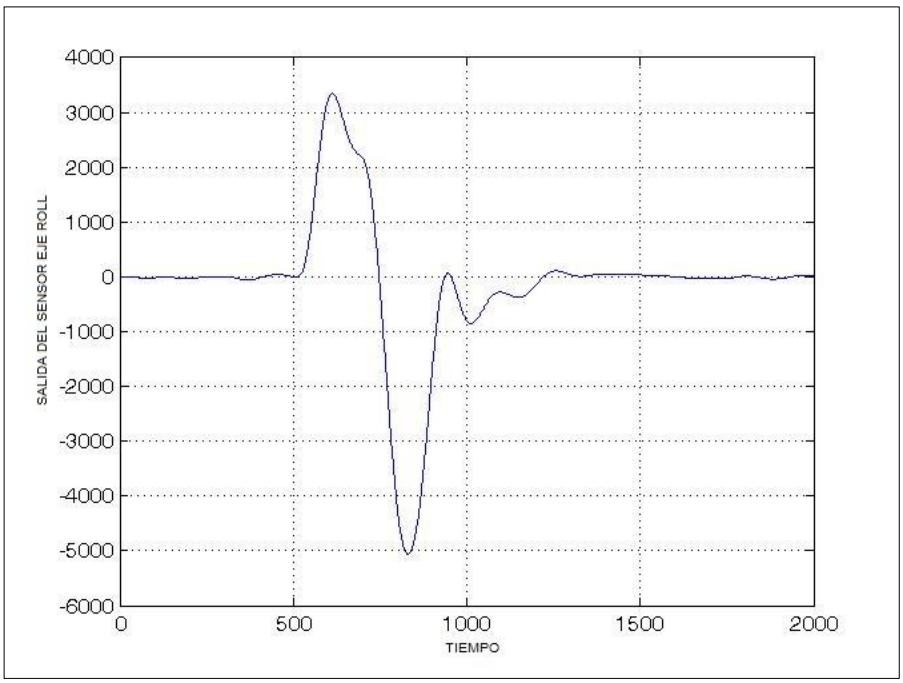

Figura 82 Respuesta del Controlador con Kp=0.4 y Ki=0.3

En la Figura 82 se observa la respuesta del controlador difuso con un error de cero. Estos valores de Kp y Ki fueron los más eficaces para el equilibrio del robot. El

tiempo de muestreo es de 0.01 segundos con un total de 2000 muestras obtenidas por el controlador difuso, lo que representa 100 muestras por segundo.

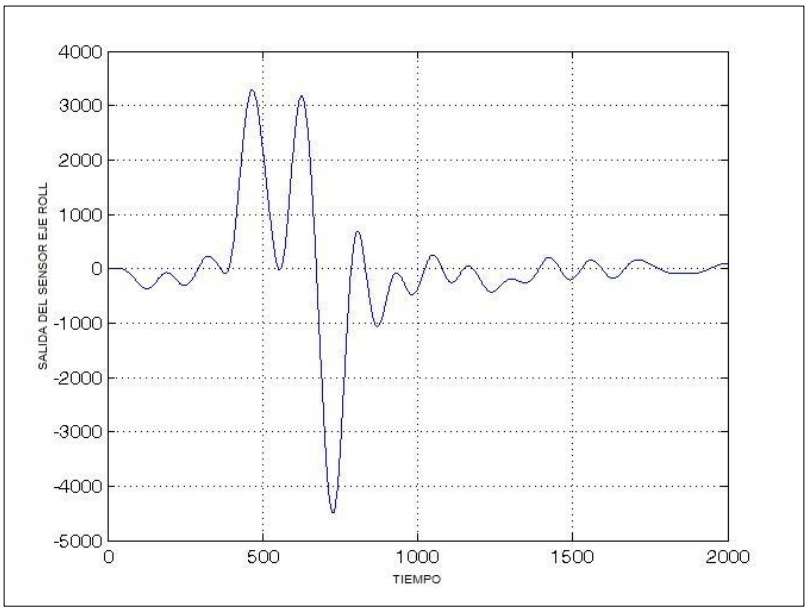

Figura 83 Respuesta del Controlador con Kp=1 y Ki=0.3

En la Figura 83 la respuesta del controlador es oscilatoria, esto se debe a que el Kp afecta a la acción del control, haciéndolo más rápido y oscilatorio.

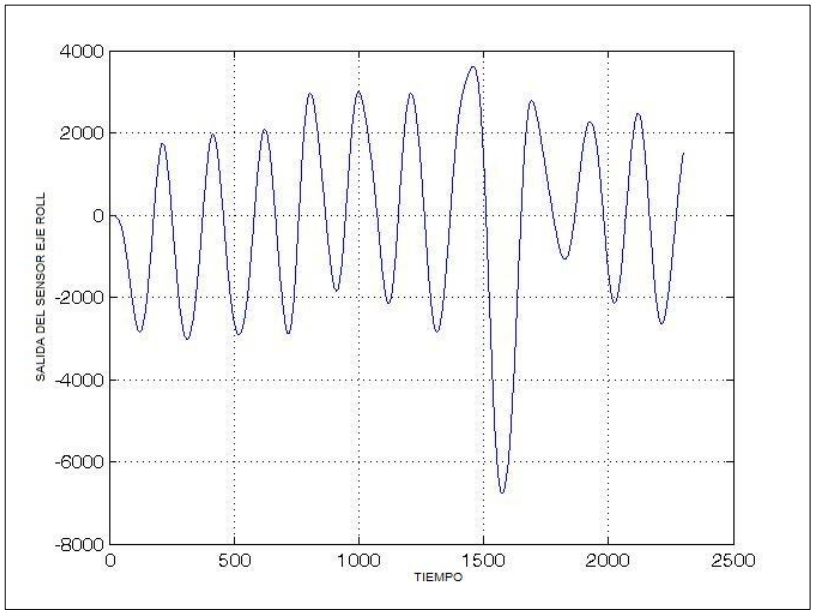

Figura 84 Respuesta del Controlador con Kp=0.4 y Ki=1

En la Figura 84 el controlador es inestable, esto se debe a la acción integral, la cual es demasiado alta, con lo que conlleva a una respuesta muy rápida del controlador y se encuentre en estado oscilatorio.

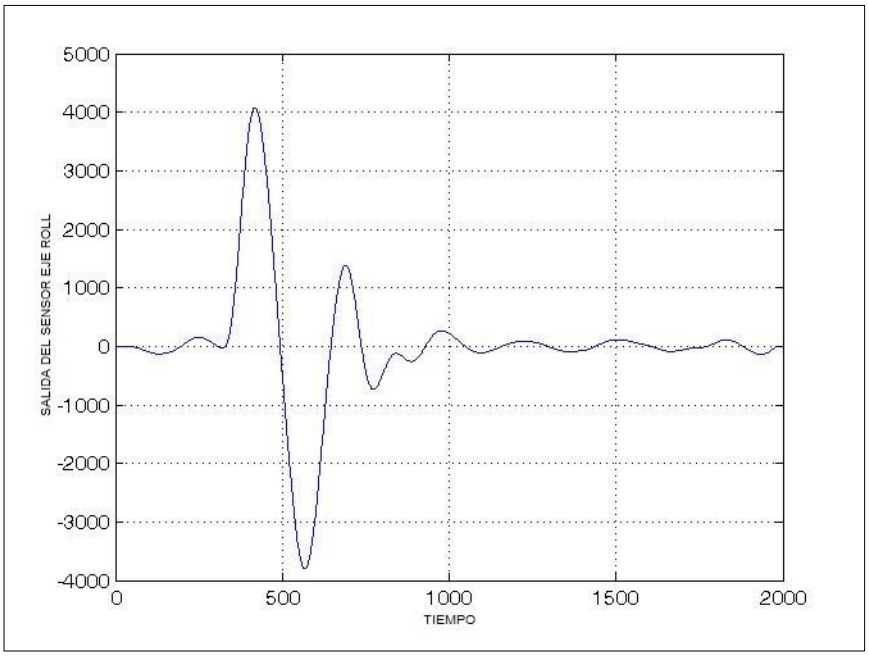

Figura 85 Respuesta del Controlador con Kp=0.5 y Ki=0.5

Por último en la Figura 85 el controlador presenta una buena respuesta con un pequeño error considerable para el equilibrio del robot hexápodo, a diferencia del control de la Figura 82 que es ideal sin la presencia de error.

4.6. Prueba de Movimientos GCP

La mayor ventaja de los movimientos de locomoción del Robot Hexápodo por GCP, es que presenta movimientos mucho más armoniosos y reales que los movimientos realizados de forma manual. Esto se debe que la programación de locomoción sin GCP presenta 8 valores de PWM en cada servomotor para realizar su movimiento hacia adelante. A diferencia de la locomoción con GCP que al utilizar ondas sinusoidales y transformando a valores PWM, se obtiene un total de 400 valores PWM para el movimiento de cada servomotor. Esto hace que los servomotores tenga un movimiento más fluido y real que la locomoción sin GCP. Un ejemplo de los valores PWM de un servomotor se encuentra en **¡Error! No se encuentra el origen de la referencia.**.

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- El robot hexápodo es capaz de caminar sobre superficies lisas e irregulares, utilizando dos métodos diferentes, mediante RNA y GCP, y capaz de equilibrarse automáticamente sobre superficies con 20 ° de inclinación.
- El robot hexápodo presenta gran estabilidad al momento de realizar su desplazamiento, debido a la elección del movimiento trípode. Además este movimiento proporciona mayor velocidad de desplazamiento en relación a otros movimientos. Esta velocidad es variable, depende directamente de la frecuencia de las señales que controlan cada servomotor.
- La tarjeta STM32F4 resulta de gran utilidad como controlador del robot hexápodo, esta permite embeber cualquier código y herramientas de Matlab, tales como: *Neural, Fuzzy Controler*, ecuaciones en el espacio de estados, señales digitales, señales analógicas, etc. Esto facilita la creación de los algoritmos que controlan los movimientos del hexápodo.
- El consumo de corriente del robot hexápodo resulta bastante elevado, debido a que utiliza 18 servomotores para su funcionamiento. Cada servomotor consume aproximadamente 500 mA, para un total de 9 Amperios aproximadamente. Por esta razón se utilizó una fuente de 5 Vdc y 40 A de salida, suficiente para abastecer el nivel de potencia requerido.
- Se logró realizar un modelo de locomoción por medio de Redes Neuronales Artificiales, usando un aprendizaje supervisado para sus patrones de entrenamiento. En este caso los patrones de entrenamiento se basaron en el movimiento de locomoción analizado previamente, donde se conocen los valores de PWM de los servomotores necesarios para posicionar a las extremidades del robot y la secuencia para lograr su desplazamiento.
- El método de entrenamiento de Redes Neuronales Artificiales mediante Retropropagación presentó un mejor resultado al momento de emular los movimientos de locomoción respecto al método de la Regla Delta Generalizada.

Retropropagación es un método más eficiente porque obtiene un menor valor del error cuadrático medio en menos épocas de entrenamiento con respecto a la Regla Delta Generalizada.

- El problema del equilibrio del robot hexápodo fue resuelto satisfactoriamente mediante el uso de un controlador PI difuso. La inexistencia del modelo matemático del hexápodo hace idónea la elección de la lógica difusa como método de control. Es decir, el robot se controla por medio de "proposiciones" que no son necesariamente ciertas o necesariamente falsas.
- Las funciones de pertenencia para el controlador difuso se obtuvieron de manera experimental. El número de estas funciones de pertenencia afecta directamente a la respuesta del controlador, mientras más funciones se desarrollen más fina será su respuesta. Con esto implica un aumento de la base de reglas que interactúan en el controlador.
- Los valores de ganancia del controlador PI difuso se obtuvieron mediante prueba y error. La modificación de estas ganancias cumplen con la teoría de control, donde el Ki es el tiempo integral, el cual regula la velocidad de acción de control, mientras que el Kp afecta tanto a la parte integral como a la parte proporcional de la acción de control.
- Se logró diseñar una Red Neuronal Celular (CNN) basados en una arquitectura analógica para la creación de un Generador Central de Patrones (GCP) locomotor capaz de realizar el control del movimiento del robot hexápodo, sustentado en una base biológica y matemática.
- Los movimientos creados por medio de un GCP otorgan una respuesta más suave o armoniosa con respecto a los movimientos clásicos de los robots. Esta aplicación resulta idónea para robots semejantes a animales, donde se busca movimientos más aproximados a la realidad.

5.2. Recomendaciones

- Para el funcionamiento del Robot Hexápodo Phoenix de forma inalámbrica, se recomienda utilizar una batería tipo LiPo de 6 voltios con capacidad mínima 9000 mAh.

- El problema de ruido en los servomotores es muy común, por lo tanto para evitar alteraciones en las señales de control, es recomendable envolver los alambres con cualquier tipo de cinta para así evitar que los cables interfieran como si se trataran de unas antenas.
- El ensamblaje de piezas es muy importante para el equilibrio del Robot Hexápodo. Se recomienda leer el manual de ensamblaje del Robot Phoenix, que se lo puede obtener en la página web de la casa comercial Lynxmotion.
- El encendido del Robot Hexápodo se lo debe realizar cuando la tarjeta STM32F4 Discovery se encuentre incrustada en su totalidad en la placa PCB. Esto evitará que exista posibles daños tanto en la placa PCB, los servomotores y la tarjeta STM32F4 Discovery.
- Para embeber un nuevo código en la tarjeta STM32F4 Discovery, se recomienda retirar la tarjeta de la placa PCB para evitar posibles daños de la tarjeta STM32F4 Discovery, los servomotores y de la placa PCB.
- La manipulación de valores de los bloques PWM de Simulink para el movimiento de los servomotores, es recomendable utilizar valores entre 1 a 12 en ciclos de porcentaje con un periodo de 20 ms. Esto se debe que el rango de funcionamiento de los servomotores se encuentran en 0 ms a 2.5 ms como máximo.

5.3. Trabajos Futuros

- Manipulación de movimientos del Robot Hexápodo Phoenix de forma remota, sea por Bluetooth o por Wifi. En la placa PCB se introdujo un módulo Wifi para realizar una comunicación serial entre la tarjeta STM32F4 Discovery y una red local. Igualmente se instaló espadines que permiten incrustar cualquier módulo Bluetooth para hacer una comunicación serial con la tarjeta STM32F4 Discovery.
- Realización de movimientos de locomoción más sofisticados y realistas. Esto se debe a que en este proyecto se utilizó 12 servomotores para el movimiento de locomoción, sin tomar en cuenta los 6 servomotores de las patas. La manipulación de estos 6 servomotores restantes hará que el movimiento del Robot se observe mucho más real al movimiento de una araña.
- Obtención de un mejor equilibrio del Robot Hexápodo. El acelerómetro de la tarjeta STM32F4 Discovery presenta 3 ejes: *Roll*, *Pitch* y *Yaw*. El presente

proyecto solo presenta equilibrio en el eje *Roll* haciendo falta como parte de equilibrio el eje *Pitch*. El eje *Yaw* puede ser utilizado como eje de navegación del Robot Hexápodo.

- La teoría de GCP aplicado al Robot Hexápodo es de forma autónoma al ser su retroalimentación $\beta=0$. Esto quiere decir que no necesita de ninguna entrada para su funcionamiento. Para la utilización de esta parte teórica del GCP, se pueden utilizar cualquier tipo de entrada como sensores de presencia para así modificar los movimientos de locomoción del Robot haciéndolo capaz de esquivar obstáculos.
- Una de las grandes ventajas de la tarjeta STM32F4 Discovery, es que presenta varios módulos extras, fácil de programar, entre los que se puede encontrar módulos de: Cámaras, GSM, GPS, Wifi, Audio, etc. Cada módulo presenta su librería de programación en Simulink de Matlab.

BIBLIOGRAFÍA

Aimagin. (21 de Abril de 2015). *Waijung Blockset*. Obtenido de http://waijung.aimagin.com/

Alejandro Rosique. (Enero de 2006). *Seminario de Diseño y Construcción de Microrobots*. Obtenido de http://www.alcabot.com/alcabot/seminario2006/Trabajos/AlejandroRosiqueGomez.pdf

Berkan, Trubatch. (1996). Fuzzy Systems Design Principles. *IEEE Press*.

Bishop. (2002). *The Machatronic Handbook.* Washington D.C, USA: Crc Press.

Campos, Alvarado, Muñoz. (29 de Octubre de 2015). *Accesus*. Obtenido de ue.accesus.com/hexapodo

César Fuertes, Romel Llumiquinga. (2005). *Diseño e Implementación de un Robot Móvil tipo Hexápodo Teledirigido.* Sangolquí: Escuela Politécnica del Ejército.

David Alejandro Reyes Gómez. (2011). *Descripción y Aplicaciones de los Autómatas Celulares.* Puebla: Universidad Autónoma de Puebla.

Efrén Gorrostieta, Emilio Vargas. (2007). *Algoritmo Difuso de Locomoción Libre para un Robot Caminante de seis Patas.* Querétaro: Universidad Autónoma de Querétaro.

Electronica Caldas. (Octubre de 2011). *Electronica Caldas*. Obtenido de http://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf

Electronica ColdFire. (Julio de 2010). *Cold-Fire*. Obtenido de http://www.coldfire-electronica.com/esp/item/26/6/servomotor-towerpro-mg995-15kg-cm

Elizabeth Sedeño Bustos. (2011). *Locomoción de un Robot Cuadrúpedo: Un enfoque a Celdas Neuronales Analógicas.* Morelos, México: CENIDET.

Enriques Martínez Peña. (2008). *Control de un Robot Tipo Puma utilizando celdas Neuronales Analógicas.* Morelos, México: CENIDET.

Fernando Izaurieta, Carlos Saavedra. (2009). *Redes Neuronales Artificiales.* Concepción, Chile: Universidad de Concepción.

G. Patel, J. Holleman. (1998). Analog VLSI model of intersegmental coordination with nearest neighbor coupling. *Georgia Institute of Technology.* Atlanta, USA: Adv. Neural Inform Precessing System, Vol 10.

J.Ortiz. (2009). *Lenguajes y Ciencias de la Computación.* Obtenido de Universidad de Málaga: http://www.lcc.uma.es/~jmortiz/archivos/Tema5.pdf

Jonathan Ocampo. (2011). *Reconocimiento de caracteres de una placa de automovil mediante redes neuronales artificiales utilizando matlab.* Sangolquí, Ecuador: Universidad de las Fuerzas Armadas.

Julio Pérez Machorro. (2009). *Generación de locomoción de un robot hexápodo usando dos células neuronales analógicas.* Morelos, México: CENIDET.

Kevin Passino, Stephen Yurkovich. (1998). *Fuzzy Control.* Ohio, USA: Addison-Wesley.

Kraimon Maneesilp, Boonchana Purahong, Pitikhate Sooraksa. (2004). A new Ananlog Control Circuit Design for Hexapod using Cellular Neural Network. *IEEE*, 2506-2510.

Leon Chua, Lin Yang. (1988). Cellular Neural Networks: Theory. *IEEE Transactions on Circuits and Systems*, 1257-1272.

León, L. A. (2010). *Generación de patrones de marcha de un hexápodo por CPG´s mediante autómatas Mealy.* Ciudad de México, México: Instituto Politécnico Nacional.

Madrid G, Jiménez E, Reyes L. (2010). *Diseño e implementación del sistema de control de un dispositivo móvil evasor de obstáculos usando lógica difusa borrosa.* Puebla, México: 9° Congreso Nacional de Mecatrónica.

Marco Valencia, Cornelio Yáñez. (2006). *Algoritmo Backpropagation para Redes Neuronales: conceptos y aplicaciones.* México DF: Centro de Investigación en Computación del IPN. Obtenido de

http://www.repositoriodigital.ipn.mx/bitstream/handle/123456789/8628/Archivo%20que%20incluye%20portada,%20%C3%ADndice%20y%20texto.pdf?sequence=1

Muhammad Hafiz Kassim, Norzeti Zainal, Mohd Rizal Arshad. (2008). *Central Pattern Generator in Bio-inspired Robot: Simulation using MATLAB.* Pulau Pinang, Malaysia: USM Robotic Research Group (URRG).

Paolo Arena, Luigi Fortuna. (2002). Analog Cellular Locomotion Control of Hexapod Robot. *IEEE Control Systems Magazine*, 21-36.

Passino, Y. (1998). *Fuzzy Control.* California, USA: Addison-Wesley.

Pedro Ponce Cruz. (2010). *Inteligencia Artificial con Aplicaciones a la Ingeniería.* México D.F.: AlfaOmega.

Roberto Latorre Camino. (2004). *ESTUDIO DE LAS FIRMAS NEURONALES EN LOS GENERADORES CENTRALES DE PATRONES.* Madrid, España: Universidad Autónoma de Madrid.

Roberto Supo. (2003). *Lógica Difusa.* Tacna, Perú: Universidad Nacional Jorge Basadre.

Robot Shop Inc. (Agosto de 2012). *Lynxmotion*. Obtenido de http://www.lynxmotion.com/c-117-phoenix.aspx

Samarasinghe, S. (2010). *Neural Networks For Applied Sciences and Engineering.* Auerbach Publications.

STMicroelectronics. (2014). *Discovery Kit for STM32F407/417 Lines.* Obtenido de User Manual: www.st.com/stm32f4-discovery

Xabier Basogain Olabe. (2010). *Redes Neuronales Artificiales y sus Aplicaciones.* Bilbao, España: Escuela Superior de Ingeniería de Bilbao.

Yinna Bautista. (25 de Noviembre de 2012). *Inteligencia Artificial*. Obtenido de http://konocimientointeligenciaartificial.blogspot.com/

Zdenko Kovacic, Stjepan Bogdan. (2006). *Fuzzy Controller Design.* Zagreb, Croacia: Taylor & Francis.