UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

IMPLEMENTACIÓN DE UNA RED NEURONAL ARTIFICIAL EN UN MICROCONTROLADOR DE 8 BITS PARA CONTROL DE DIRECCIÓN DE UN ROBOT MÓVIL

TESIS
PARA OBTENER EL GRADO DE

INGENIERO EN REDES

PRESENTA
MISAEL BAEZA PADILLA

DIRECTOR DE TESIS
DR. VÍCTOR MANUEL SÁNCHEZ HUERTA

ASESORES
DR. JAVIER VÁZQUEZ CASTILLO
DR. JAIME SILVERIO ORTEGÓN AGUILAR
DR. FREDDY IGNACIO CHAN PUC
DR. HOMERO TORAL CRUZ

CHETUMAL QUINTANA ROO, MÉXICO, AGOSTO DE 2018

UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

TRABAJO DE TESIS TITULADO
"IMPLEMENTACIÓN DE UNA RED NEURONAL ARTIFICIAL EN UN MICROCONTROLADOR DE 8 BITS PARA EL CONTROL DE DIRECCIÓN DE UN ROBOT MÓVIL"

ELABORADO POR
MISAEL BAEZA PADILLA

BAJO SUPERVISIÓN DEL COMITÉ DE ASESORÍA Y APROBADO COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:

INGENIERO EN REDES

COMITÉ DE TESIS

DIRECTOR: _____
DR. VÍCTOR MANUEL SÁNCHEZ HUERTA

ASESOR: _____
DR. JAVIER VÁZQUEZ CASTILLO

ASESOR: _____
DR. JAIME SILVERIO ORTEGÓN AGUILAR

ASESOR: _____
DR. FREDDY IGNACIO CHAN PUC

ASESOR: _____
DR. HOMERO TORAL CRUZ

Universidad de Quintana Roo
DCI DIVISIÓN DE CIENCIAS E INGENIERÍA

CHETUMAL QUINTANA ROO, MÉXICO, AGOSTO DE 2018

Agradecimientos

Antes de todo quiero agradecer a Dios, porque durante los años que estuve en la Universidad me ha dado las fuerzas que necesitaba. Que todo lo que haga pueda honrar a Dios.
A mis queridos padres don Cheto y doña Silvia, que me han dado la oportunidad de seguir estudiando. A pesar de todos los problemas que como familia hemos tenido, siempre han estado dándome su apoyo. A mi querida hermanita Lesly, a Rubiel y al pequeño Alex, porque siempre estuvieron impulsándome. Sin duda a mis abuelos; Faustino, Raúl, Fabiana y Obdulia, qué sería de mí sin ellos, que me dieron consejos en cada momento.

A mi tío Rodolfo y a toda su familia; tía Chesca, Carlos, Pocho y Mirsa, porque desde el primer día estuvieron conmigo, de no ser por ustedes esto no sería posible.

A mis tías Dámaris y Saraí por haber sido como madres para mí, siempre aconsejándome y apoyándome.
A mis tíos Efraín y Azael, gracias por apoyarme durante estos años.

A Lidia Santos por apoyarme, aconsejarme, y alegrarme los días.

Gracias a mis compañeros; Aracely, John, Eneldo, Carlos y Diego Go Go. Por toda su ayuda durante los años en la Universidad, por los momentos que pasamos.

A todos mis profesores, por haberme ayudado siempre que lo necesitaba. En especial al maestro Rubén González, al Dr. Javier Vázquez, al Dr. Jaime Ortegón y sin duda al Dr. Víctor Sánchez, por haber sido estricto como tutor y por guiarme de la mejor manera durante mis años como su tutorado.
¡A todas las personas que han sido parte de esta aventura que empieza a escribirse, GRACIAS!

Dedicatoria

Nadie se merece más esto que mis padres, porque sin su apoyo no hubiera sido posible. Es por ustedes y para ustedes.

Lely es gracias a tu apoyo que termino esta etapa.

Abuelitos es para ustedes y es solo el inicio.

Tío Rodolfo, sin su ayuda no estaría escribiendo esto.

Lilo sin tu ayuda y tu cariño no sería posible.

Resumen

El siguiente trabajo de tesis presenta el desarrollo de una Red Neuronal Artificial (RNA) B-Spline, utilizada para el control de dirección de un robot móvil. El robot Zumo 32U4 recibe información de un sensor de movimiento y la procesa para determinar qué tanto ha sido desviado de su trayectoria. El resultado de procesar la información es un escalar utilizado para corregir la desviación del robot, dicho valor sirve para compensar la velocidad de los motores del robot si éste ha sufrido cambios en su trayectoria inicial.

En el Capítulo 1 analizamos la problemática, establecemos razones por las cuales realizar un proyecto de tesis basado en el análisis de ésta es válido. Definimos el objetivo general y los objetivos particulares.

La teoría fundamental del proyecto se expone en el Capítulo 2. Realizamos una recolección de todos los datos útiles sobre el robot Zumo 32U4, su estructura, sus componentes y sobre la interfaz I2C necesaria para la adaptación de un sistema de posición. Exponemos la teoría sobre la técnica de las B-Splines y cómo utilizarlas en una red neuronal.

En el Capítulo 3 analizamos las características de las Redes Neuronales Artificiales (RNA), así como también se establece el funcionamiento de la RNA B-Spline y las ventajas de trabajar con una RNA de este tipo.

El desarrollo de la RNA es el tópico del Capítulo 4. En este Capítulo analizamos cómo implementar con detenimiento el algoritmo elegido.

Las pruebas experimentales y sus debidos resultados son el tópico del Capítulo 5. En este Capítulo analizamos el comportamiento de la RNA en sus diferentes configuraciones.

Tabla de Contenido

Tabla de Figuras

Capítulo 1. Control de la Velocidad en los Motores del Zumo Robot 32U4

1.1 Introducción

Las computadoras actuales pueden realizar miles de cálculos por segundo y debido a esta capacidad de procesamiento podemos implementar Redes Neuronales Artificiales (RNA) complejas. El problema surge cuando queremos implementar una RNA en un microcontrolador de 8 bits. La memoria del microcontrolador es limitada, no puede almacenar valores muy grandes, además de esto, tampoco cuenta con el poder de procesamiento que tienen las computadoras actuales. Muchos sistemas de control en la actualidad precisan dos cosas; espacios más compactos y menos consumo de energía. La energía suministrada a estos compactos sistemas de control proviene de baterías, por lo que los microcontroladores son la opción más viable para estas situaciones, debido al bajo consumo de energía.

1.2 Justificación

Los microcontroladores de 8 bits no tienen la misma potencia de trabajo que un procesador de 32 bits, por lo que la red neuronal implementada en este proyecto de tesis se basa en funciones de activación. Con las funciones de activación reducimos la cantidad de operaciones necesarias en el procesamiento. Si bien en la actualidad contamos con procesadores de alta capacidad de procesamiento, los microcontroladores siguen siendo ampliamente utilizados. Debido a que son pequeños y consumen menos energía que un procesador, encontramos microcontroladores en máquinas industriales, en drones, en juguetes, en aparatos electrodomésticos, etc. En esto radica la importancia del presente trabajo de tesis, en implementar una red neuronal en un microcontrolador de 8 bits y encontrar más soluciones a problemas cotidianos de control. Actualmente no se dispone de suficiente información sobre la implementación de redes neuronales en controladores de 8 bits, basta con realizar una búsqueda en la Red para hacer notar esta situación.

1.3 Definición del problema

El detalle más importante por considerar cuando se trabaja con microcontroladores de 8 bits es el uso de la memoria. Los registros de memoria pueden ser sobre cargados con facilidad.

Cuando requerimos de precisión en la recolección de los datos, es fundamental tener en cuenta el tipo de variables que se necesitan en el programa. Por ejemplo, si declaramos una variable de tipo entera, y la utilizamos para almacenar valores de tipo coma flotante, perderemos información que el sistema precisa. Otro ejemplo, si declaramos variables de tipo coma flotante y solamente guardamos en ellas valores enteros, estamos desperdiciando espacio de memoria, que puede afectar notablemente al microcontrolador. La implementación de una red neuronal por lo tanto se ve limitada por los espacios de memoria. En este proyecto de tesis, nos proponemos implementar una red neuronal que no requiera tanto poder de procesamiento y de identificar los problemas que surjan en el proceso.

1.4 Objetivo general

Implementar y programar una red neuronal utilizando la tecnología B-Spline en el robot Zumo 32U4 para realizar una corrección en la velocidad de sus motores y con ello mantener una dirección determinada.

1.5 Objetivos particulares

- Adaptar un nuevo sensor de movimiento en la estructura del robot Zumo 32U4.
- Determinar el rango óptimo en la detección de movimiento basado en la información entregada por el sensor.
- Implementar el algoritmo de la Red Neuronal B-Spline.
- Programar la Red Neuronal B-Spline en el IDE de Arduino.
- Utilizar el valor obtenido de la red neuronal para ajustar la velocidad de los motores y con ello corregir la dirección del robot.
- Comparar el rendimiento de la Red Neuronal B-Spline con un algoritmo de Control Proporcional.

Capítulo 2. Marco Teórico

2.1. Zumo 32U4

Zumo 32U4 es un pequeño pero completo robot controlado por un microcontrolador ATmega32U4, debido a todas las características que posee se convierte en un robot apto para peleas de robots Mini-Sumo. El robot cuenta con motores de Puente-H duales; estos motores pueden cambiar su polaridad y de esa manera girar hacia adelante o hacia atrás. Para percibir el ambiente el robot cuenta con sensores de movimiento y con un sistema de sensores de proximidad. El Zumo 32U4 cuenta con "Push-Buttons" que sirven de interfaz para el usuario y con una pantalla LCD en la cual el usuario puede ver en tiempo real la información que el robot está generando [1].

Figura 1: Robot Zumo 32U4

2.1.2 Funcionamiento de los motores del robot Zumo 32U4

Los motores del Zumo32U4 se pueden controlar de forma relativamente sencilla desde el código, esto es posible porque Pololu nos facilita las librerías necesarias para el código. La velocidad de los motores se puede configurar con valores desde uno hasta cuatrocientos, este valor es un escalar y cada motor puede recibir un valor diferente. Mientras un motor recibe un valor de 100, el otro puede recibir -200 lo cual significa que éste motor se mueve hacia atrás.

2.1.3 Motores CD del Zumo 32U4

Los motores de Corriente Directa del Zumo 32U4 utilizan un controlador DRV883x que provee una solución integrada para aplicaciones donde el voltaje es bajo. El bloque de control de salida consiste en un MOSFET de canal N configurado como Puente-H, con esta configuración el Puente-H puede invertir la polaridad de la corriente y así el motor puede girar hacia adelante o hacia atrás. El robot no cuenta con ruedas normales, es decir, sus cuatro ruedas no tocan el piso. Cuenta con correas para pista o mejor conocidas como "orugas", las ruedas giran hacia adelante o hacia atrás, pero el contacto con el piso es por medio de las "orugas". Si queremos hacer girar al robot hacia la derecha podemos hacerlo de dos maneras; la primera es que la oruga del lado izquierdo es la que se moverá hacia adelante o bien la oruga del lado derecho se moverá hacia atrás, es decir, mientras la oruga de un lado se mueve, la del otro lado debe de estar fija[2].

Los motores del robot móvil Zumo 32U4 se encuentran en la parte frontal del robot como se puede apreciar en la Fig. 2 por lo tanto, actúan solamente en las dos ruedas frontales.

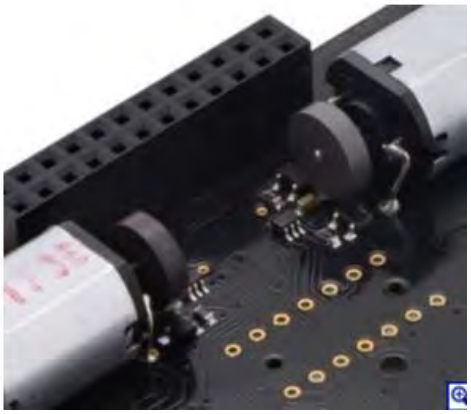


Figura 2: Motores Zumo 32U4

Los siguientes pines son los utilizados para controlar los motores.

- Los pines digitales 15 y 16, controlan la dirección de los motores derecho e izquierdo respectivamente. Un voltaje alto mueve los motores hacia atrás y un voltaje bajo los mueve hacia adelante.
- Los pines digitales 9 y 10, controlan la velocidad de los motores derecho e izquierdo respectivamente.

Para lograr esto se utiliza la Modulación por Ancho de Pulso (PWM)[1].

2.2 Modulación PWM

La Modulación por Ancho de Pulso o PWM por sus siglas en inglés, es una técnica que permite replicar el comportamiento de una señal analógica, esto se logra manipulando la frecuencia y el ciclo de trabajo de la señal digital. El ciclo de trabajo se define por el tiempo en el que la señal se encuentra en un valor alto, es decir, en un uno lógico divido entre el periodo. La frecuencia se define por el total de veces que un ciclo se repite en un segundo, por ejemplo, cuando se dice que una señal tiene una frecuencia de 1KHz, significa que la señal se repite 1000 veces o completó 1000 ciclos en un segundo. Cuando la señal cambia entre el valor alto y el valor bajo con una alta frecuencia, la señal parecerá tener un comportamiento similar al de una señal analógica constante en su salida.

La modulación PWM es ampliamente utilizada para el control de velocidad en motores de Corriente Directa, como es el caso del robot Zumo 32U4.

2.3 Control Proporcional

El Control Proporcional (P) es la acción proporcional al error de control. El error mide la diferencia entre el valor actual y el valor que se define en el punto de control, cuando se han realizado estos cálculos, entonces aplica el cambio. Este control puede definirse como el valor requerido del error para alcanzar un cambio en nuestra salida. En la Fig. 3 se puede observar el funcionamiento de un controlador proporcional.

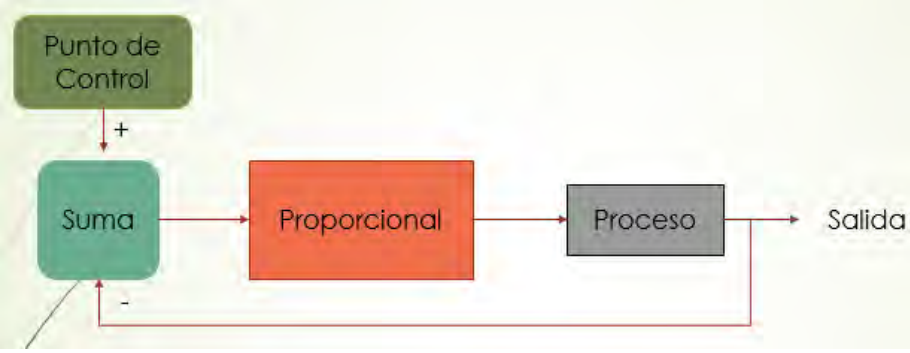


Figura 3. Diagrama a bloques de un Controlador Proporcional

En el robot Zumo 32U4 se probó un algoritmo de Control Proporcional para compensar la velocidad de los motores y con ello corregir la dirección si ésta sufrió algún cambio, sin embargo, cuando usamos el Control Proporcional para corregir la dirección del robot móvil, ésta se ve afectada por perturbaciones exteriores o cambios en los parámetros[3].

2.7 Controladores de 8 bits

Un controlador de 8 bits es un sistema autónomo que cuenta con una memoria, un procesador y periféricos. La mayoría de los microcontroladores de 8 bits de la actualidad son usados como sistemas embebidos en productos como teléfonos, automóviles, domótica, etc.[4]. Los microcontroladores de 8 bits fueron desarrollados en la década de 1970, sin embargo, siguen siendo bastante populares.

Los microcontroladores de 8 bits tienen limitaciones en cuanto al uso de la memoria y el uso de los registros se ve afectado por el tamaño de los datos que pueden almacenar. Claramente estos microcontroladores están en desventaja en comparación con los nuevos sistemas de 32 bits o 64 bits, sin embargo cuando queremos realizar tareas que no requieren tanto poder de procesamiento, pero sí menos consumo de energía los microcontroladores de 8 bits son los más adecuados.

El robot móvil Zumo 32U4 cuenta con un microcontrolador ATMega32U4 de 8 bits, el cual será el encargado de ejecutar el Neuro-Controlador basado en la RNA B-Spline.

2.8 Problema de cinemática inversa

La cinemática del robot estudia el movimiento con respecto a un sistema de referencia. En la cinemática existen dos problemas fundamentales, la cinemática directa y la inversa.
La cinemática inversa determina las coordenadas articulares en función de la posición del robot. En el desarrollo del presente proyecto, surge un problema de cinemática inversa conocido como "pitch" o "cabeceo".[5]

En la Fig. 4 se muestra cómo este problema afecta al robot móvil Zumo32U4. Cuando la perturbación que el robot percibe es muy grande, la RNA no converge y se pierde el control sobre la memoria del robot. Cuando se presenta este "cabeceo" los valores de salida de la RNA son elevados, lo cual significa que los valores de compensación en la velocidad son muy grandes. Este problema ocasiona que el robot esté oscilando de un lado a otro sin control.

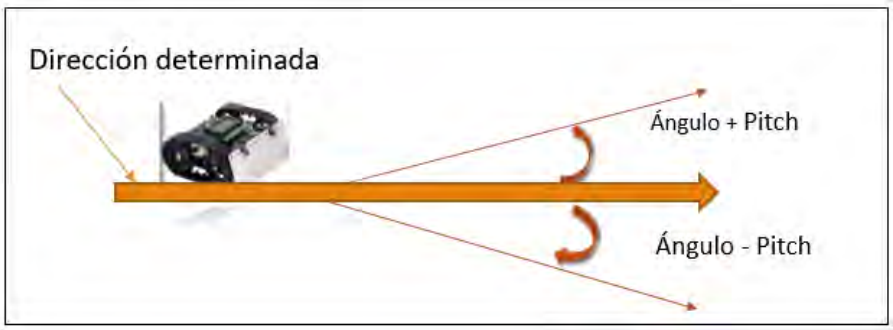


Figura 4: Problema de Cinemática Inversa "Pitch"

Debido a los problemas de cinemática en los robots, mantener la dirección cuando el robot está en movimiento puede ser muy complicado. El problema crece cuando el robot no cuenta con gran poder de procesamiento para poder procesar información de múltiples sensores. Procesar información de la menor cantidad de sensores es lo más viable para robots pequeños, sin embargo, las correcciones realizadas pueden no ser lo suficientemente precisas.

Capítulo 3. Red Neuronal B-Spline

3.1 Redes neuronales artificiales

El cerebro humano realiza cálculos de gran precisión, es un sistema no lineal, con capacidad de realizar cálculos en paralelo y, sobre todo, tiene la capacidad de reorganizarse. Cuando nos enfrentamos a situaciones desconocidas o de las cuales tenemos poca información disponible, tenemos la habilidad de aprender; las conexiones cerebrales cambian, algunas se deshacen o se forman nuevas conexiones.

Una Red Neuronal Artificial (RNA) es la unión masiva de pequeñas unidades de procesamiento que tienen la capacidad de guardar pequeñas porciones de información y hacerlas disponibles para usos posteriores. El conocimiento de la RNA es adquirido mediante un proceso de aprendizaje influenciado por el ambiente. Para este proceso es muy importante definir las conexiones Inter-Neuronales también conocidas como los “pesos”, pueden amplificar o minimizar el valor que devuelve la neurona.

El algoritmo de aprendizaje tiene la función de modificar los pesos, al hacer esto podemos reducir el error de aproximación al valor real y acercarnos lo más posible al objetivo deseado. Las RNA más avanzadas tienen la capacidad de modificar su propia topología, sin embargo, en el presente proyecto se utiliza una de las técnicas más simples; las redes basadas en funciones de activación.
Es importante señalar que existen RNA´s que no utilizan funciones de activación. Las RNA´s utilizadas en procesadores de alta capacidad están basadas en funciones exponenciales y logarítmicas.

3.1.2 Elementos de una red neuronal artificial

Las entradas de las RNA reciben información del ambiente. Para el presente proyecto se utiliza solo una entrada o variable, la cual será proporcionada por el sensor de movimiento. El sensor de movimiento detecta los cambios en los valores del “eje z”. En configuraciones de una RNA donde las entradas son múltiples, como se puede observar en la Fig. 5, cada entrada tiene un “peso” relativo. Este peso indica la prioridad que tiene la entrada en la neurona como también indica el conocimiento adquirido de la red mediante el algoritmo de aprendizaje.

Las neuronas son unidades de procesamiento, es decir, son la base de una compleja red neuronal. Realizan cálculos no tan complejos, pero al resolver cada unidad una pequeña parte del problema y al estar todas unidas, el resultado es una aproximación al valor real de mucha precisión. El "bias" o "sesgo" es un componente externo que puede aumentar o disminuir la salida de una función de activación.

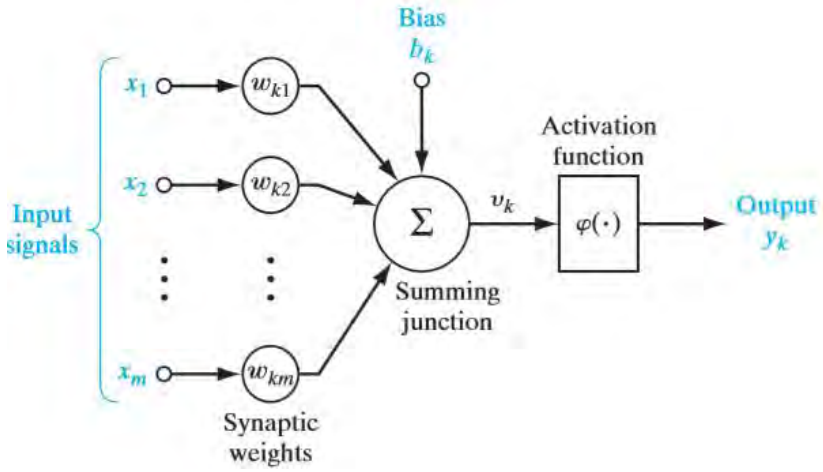


Figura 5: Diagrama de conexiones de una RNA

Las funciones de activación fueron desarrolladas en la década de 1940 y a ese modelo se le nombró "McCulloch-Pitts"[6]. En este modelo la función tomaba el valor de 1 cuando el campo local inducido era mayor o igual que cero. Si el campo local inducido es menor que cero, entonces el valor de la función es cero. Para el presente proyecto, estos modelos matemáticos toman un valor importante. Como se ha mencionado, trabajar con funciones de activación en el área de ingeniería representa ventajas en la optimización. Utilizar funciones sigmoidales es uno de los métodos más comunes, como ventaja es que este tipo de funciones ofrecen aproximaciones muy precisas. Pero en nuestro proyecto no es viable utilizarlas porque realizar todos los cálculos que requieren sería una tarea demasiado exigente que el microcontrolador no podría ejecutar.

La función de salida es la última parte que compone a la neurona, cada neurona tiene una única salida que determina el valor que se transfiere a las neuronas vinculadas, si la función de activación está por debajo de los umbrales determinados entonces el valor de salida es cero.[7]

3.1.3 Aprendizaje no supervisado

Las RNA con aprendizaje no supervisado son aquellas que no necesitan un asesor o maestro externo para realizar el proceso de aprendizaje. La red no recibe información por parte del entorno que le indique si la salida generada en respuesta a una entrada es o no correcta. El aprendizaje no supervisado consiste en que la red descubra por sí misma características, regularidades, correlaciones o categorías en los datos de entrada y se obtenga de forma codificada en la salida.

Con las RNA de aprendizaje no supervisado podemos analizar los componentes principales. Se trata de detectar cuales de los componentes del conjunto de entrada caracterizan en mayor grado el conjunto de datos.
En el presente proyecto se utiliza una configuración de RNA basada en funciones B-Spline debido a que podemos implementar recursividad con estas funciones los cálculos requeridos en el proceso son relativamente sencillos. El microcontrolador utilizado en este proyecto no cuenta con gran poder de procesamiento, en esto radica la importancia de reducir la complejidad de las operaciones.

3.2 Funciones B-Spline

Las Splines son curvas polinomiales a trozos que permiten realizar representaciones matemáticas de superficies tomando en cuenta algunos de los puntos de dicha superficie. Para obtener la representación matemática de la superficie necesitamos construir una función de interpolación que pase por la mayor cantidad de puntos de la superficie como sea posible.

En la actualidad las funciones B-Spline han tomado mucha importancia en el desarrollo de nuevo software. Estas funciones se construyen a partir operaciones matemáticas muy sencillas, pero al usarlas en forma recursiva podemos obtener funciones de grados superiores de mucha precisión. Esta ventaja que ofrecen las funciones B-Spline, más el aumento en la capacidad de procesamiento de las computadoras actuales, permite a los animadores realizar trabajos de mayor calidad.

3.3 Recursividad con B-Splines

Las funciones B-Spline de grado cero son la base para una definición recursiva de todas las funciones B-Spline de orden superior. La relación de recurrencia (3.1) describe una spline de orden 0. Los cálculos son relativamente sencillos para el microcontrolador que posee el robot utilizado en este proyecto. Al tener la salida de la función de grado cero, posteriormente la utilizaremos como entrada en otra función y esa salida se convierte en grado uno. Utilizando este método de recursividad se llega a obtener una función de grado 2 que nos ofrece una aproximación satisfactoria sin tener que exigir mucho poder de procesamiento al microcontrolador. Las spline de orden superior se obtienen a partir de la relación de recurrencia (3.2).

$$B_{i,1}(x) = \begin{cases} 1 & T_i \geq x \geq T_{i+1} \\ 0 & otros \end{cases} \qquad (3.1)$$

Y

$$B_{i,k}(x) = \frac{x - T_i}{T_{i+k-1} - T_i} B_{i,k-1}(x) + \frac{T_{i+k} - x}{T_{i+k} - T_{i+1}} B_{i+1,k-1}(x) \quad \text{para } k \geq 2 \qquad (3.2)$$

Donde ($T_i, \ldots, T_{i+k}$) son los puntos de control o nuestras lambdas, k representa el orden del B-Spline. La importancia de utilizar B-Splines en aproximaciones numéricas se debe principalmente al descubrimiento de las relaciones recursivas, como podemos observar en la relación de recurrencia (3.2) los B-Splines de orden k pueden ser generados a partir de B-Splines de orden k-1.

Utilizar interpolación con funciones básicas o de bajo orden hace que los cálculos sean más simples, es algo de suma importancia cuando trabajamos con un microcontrolador y sobre todo cuando esperamos respuestas en cortos periodos de tiempo. La recursividad también nos permite mayor precisión en las

operaciones matemáticas y nos permite incluir múltiples puntos de control; con esto logramos estabilidad numérica y lo más importante es que las aproximaciones tienden a ser bastante precisas.

3.4 Red neuronal B-Spline

Mediante las RNA B-Spline existe la posibilidad de limitar el espacio de entrada debido a las funciones de activación. En la Fig. 6 podemos observar la arquitectura de esta RNA.
La característica más importante de las RNA B-Spline es la minimización en los datos de salida debido a las funciones de activación. Una función base se construye a partir de un conjunto de puntos de control llamados “nodos”, “umbrales” o “lambdas. Estos valores definen el tamaño y la forma de las funciones base que el usuario establece.[8]

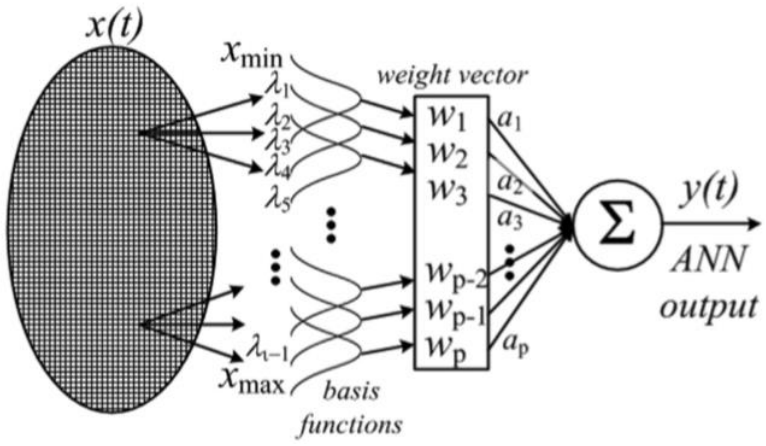


Figura 6: Diagrama de conexión de RNA B-Spline

Las RNA B-Spline permiten construir funciones de grado superior comenzando con una sencilla operación aritmética. Las relaciones de recurrencia (3.1) y (3.2) nos permiten observar este proceso, lo cual reduce las operaciones y permite que este tipo de RNA sea adecuado para sistemas embebidos.

Las Redes Neuronales Artificiales pueden clasificarse en dos categorías principales. Las redes que requieren de un proceso de entrenamiento y las que no requieren de este proceso, las primeras son conocidas como “off-line” y las demás se clasifican como “on-line”. Las RNA B-Spline no requieren un proceso de entrenamiento, por lo que se conocen como redes de aprendizaje “on-line”.

La salida de la RNA consiste en la sumatoria del producto de los pesos por la salida de las funciones de activación. En la salida únicamente participan un número de funciones base determinado, por lo que los pesos no son calculados necesariamente en cada paso. La característica mencionada anteriormente reduce el tiempo de ejecución y requiere menor poder de procesamiento. Las RNA B-Spline son por lo tanto adecuadas para un control adaptable “on-line”.[9]

Las RNA B-Spline pueden adaptarse para detectar cambios en el estado de operación del sistema y para compensar perturbaciones externas. Esto se debe al vector de los pesos, que se actualizan “on-line” en cada muestreo de los datos.

Capítulo 4. Desarrollo del Neuro-Controlador B-Spline

La implementación del Neuro-Controlador B-Spline en el presente proyecto se basa en el algoritmo elegido. El diseño del algoritmo utilizado fue tomado del artículo “On-line air supply control of PEM fuel cell by an adaptive neural network”[8]. La Fig. 7 nos permite observar el algoritmo a detalle. Se pueden apreciar las conexiones entre los procesos y finalmente la acción sobre los motores. En fases posteriores de este capítulo analizamos con detenimiento cada componente del algoritmo.

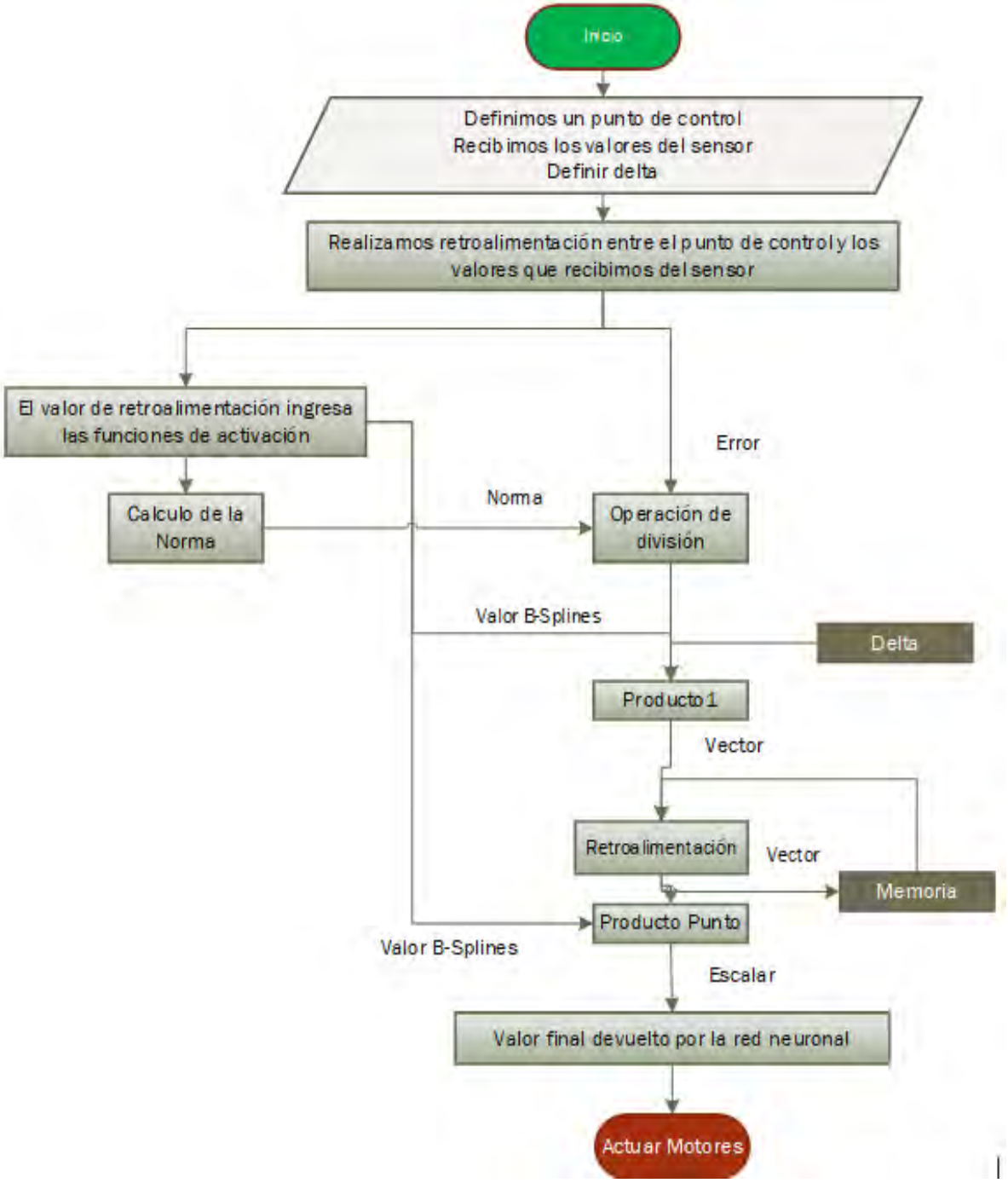


Figura 7: Algoritmo de la RNA B-Spline

4.1 Programar en el Arduino IDE

El código en el cual se programa para Arduino está basado en C++. La sintaxis es muy parecida lo que permite a un programador con cocimientos de C++ adaptarse con facilidad al IDE de Arduino.
Los programas escritos en Arduino se llaman sketches y se guardan con la extensión (.ino). El IDE (Fig. 8) identifica errores en el código y en la pantalla en la esquina inferior izquierda muestra un pequeño resumen de la causa de dicho problema[10]. En el entorno podemos identificar diversas notificaciones, en la esquina inferior derecha nos dice el nombre de la placa para la cual estamos programando y el nombre del puerto serial donde reconoció la placa.

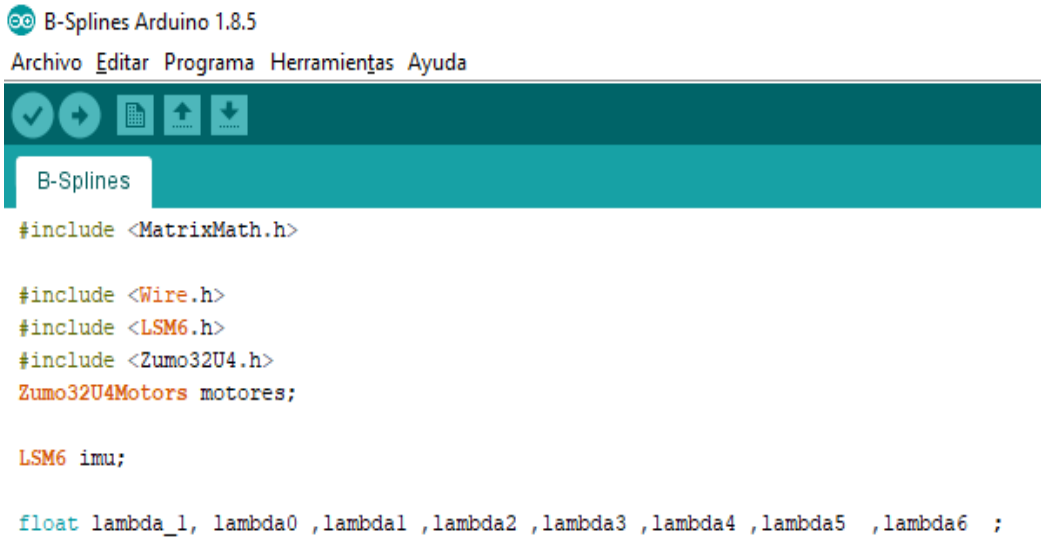


Figura 8: Librerías Arduino IDE

En la parte superior del área de trabajo, es donde incluimos las librerías que necesitaremos a lo largo de nuestro programa, todas las librerías con las que trabajamos en Arduino están escritas en C++, como se muestra en la Fig. 8. Cuando incluimos una librería respetando la sintaxis del IDE, debemos notar cambios de color en las líneas de código, esto indica que la inclusión de la librería es correcta.

Sin embargo, algunas veces el nombre de la librería no cambia de color, como se muestra en la Fig. 8 con la librería del Zumo 32U4, esto se debe a un error del IDE que probablemente se corrija más adelante. En

esta área también podemos definir variables globales como las lambdas definidas en el código mostrado; igualmente podemos asignar nombres a funciones de alguna librería o declarar componentes como el del sensor LSM6.

El cuerpo de un programa en Arduino tiene dos métodos principales. El primero es el método setup (). En este método se establecen configuraciones estáticas. Por ejemplo, en este método podemos definir la velocidad de la interfaz serial, podemos de igual manera crear una estructura de control que permita verificar si los sistemas que necesitamos están funcionando al momento del inicio y con ello dar paso al siguiente método.

El método loop () es el primer método en ser llamado cada vez que el sistema inicia después de llamar al método setup (). Dentro de este método podemos escribir las instrucciones que definan el comportamiento general de nuestro programa; podemos definir estructuras de control, definir variables locales, asignar valores a los programas e incluso llamar a otros métodos. Cuando se han completado todas las instrucciones dentro del método loop () incluyendo el tiempo programado de espera, el método vuelve a iniciar una y otra vez mientras el sistema esté en funcionamiento. Las estructuras de los métodos mencionados se presentan en la Fig. 9.

Figura 9: Métodos Principales Arduino IDE

Para administrar el sketch que estamos escribiendo, el IDE nos ofrece diversas herramientas que podemos observar en la Fig. 10. La primera de ellas es la opción "verificar"; al elegir esta opción el IDE buscará errores en el código y de presentarse alguno, en la esquina inferior izquierda nos mostrará el mensaje correspondiente. Si el sketch no tiene errores entonces se compila.

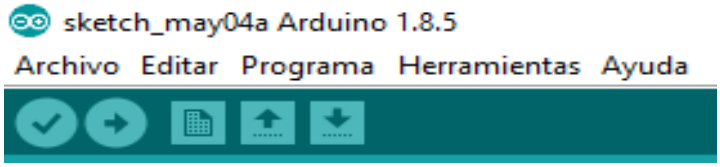


Figura 10: Herramientas Arduino IDE

La opción "subir" nos permite cargar el programa al microcontrolador, como medida extra de seguridad, al elegir "subir" el IDE compila el código por si el usuario olvidó hacerlo y de esa manera evita cargar códigos erróneos al microcontrolador. Antes de "subir" el programa, es importante verificar la parte inferior derecha del IDE, porque ahí podemos comprobar que el microcontrolador está conectado, en caso de no detectar nada, el IDE nos mostrará un mensaje detallando porqué falló.

4.1.1 Programar al Zumo 32U4

Debemos descargar las librerías necesarias para el Zumo 32U4 e instalarlas en el IDE, igual de necesario es instalar el controlador del puerto USB. Cuando la instalación de las librerías y del controlador se ha completado, en la sección de herramientas elegimos la placa "Pololu A-Star 324". En la Fig. 11 observamos en qué sección del IDE se muestran las placas.

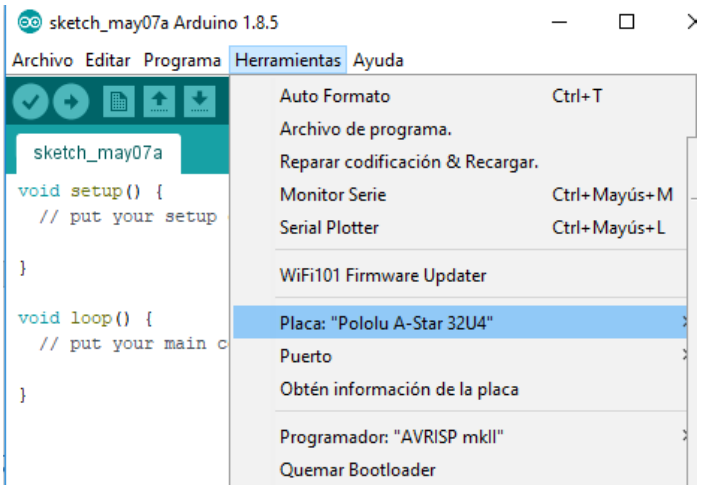


Figura 11: Placas Arduino IDE

En la sección de “Programa” podemos verificar que la librería se instaló correctamente, buscándola con el nombre de “Zumo 32U42”. La Fig. 12 nos permite observar el proceso mencionado anteriormente.

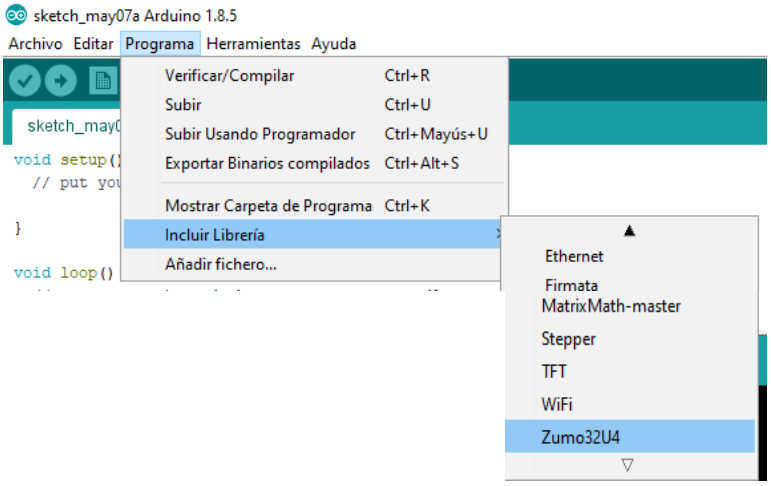


Figura 12: Programar Placa Arduino IDE

Las librerías mencionadas anteriormente se pueden encontrar en el sitio web de Pololu, del mismo modo en el sitio web Pololu provee a los usuarios una guía donde se muestra el proceso de instalación de las mismas.[11]

4.2 Sensores de movimiento

4.2.1 Interfaz I2C

El nombre I2C significa “Circuito Inter-Integrado”, es un protocolo de comunicación serial síncrono. En este protocolo la información es enviada bit por bit en un solo canal de comunicación. El protocolo I2C combina lo mejor de la tecnología UART y de la tecnología SPI. El bus I2C consiste únicamente en dos líneas; una es la SDA (Serial Data Line) la cual transporta los bits de datos y la otra línea es SCL (Serial Clock Line) la cual es usada para definir las señales de reloj. Cuando se envía un mensaje a través de la interfaz I2C, éste debe contener la dirección que identifique al dispositivo al cual es dirigido[12].

Los dispositivos conectados a la interfaz I2C deben tener direcciones únicas, los dispositivos que utilicen la interfaz pueden ser de dos tipos. El maestro es el que inicia la transferencia de datos y también define la señal de reloj, dependiendo de la configuración del circuito el dispositivo maestro puede cambiar. El dispositivo esclavo recibirá un mensaje de inicio del maestro, compara la dirección de destino que contiene y si coincide con la suya entonces inicia la transmisión de datos.

4.2.2 Sensor LSM303D MiniMU-9 v3

El robot Zumo 32U4 cuenta con un sensor de aceleración/magnetómetro de alta capacidad, que puede ser usado en diversas aplicaciones, una de ellas es detectar colisiones y determinar qué cambios ha sufrido en su orientación, estas aplicaciones se logran realizando una medición a las unidades inerciales devueltas por el sensor. El sensor LSM303D con dimensiones de 3 x 3 x 1mm, es un sistema que permite realizar mediciones de aceleración lineal en 3D. Este sensor puede trabajar en dos frecuencias de operación; en su modo estándar trabaja con frecuencias de 100kHz y en su modo rápido trabaja a 400kHz.

El sensor LSM303D puede ser configurado de múltiples formas; incluyendo la selección de sensibilidad dinámica para el acelerómetro, elegir diferentes rangos para los datos de salida y también es posible programar interrupciones inerciales externas. El sensor debe ir conectado a una fuente de voltaje de 2.5V a 5V. Las terminales SDA y SCL deben ir conectados a puertos que tengan el mismo voltaje que el terminal de VIN.

Cuando el sensor está conectado en el modo I2C, los siete bits de la dirección del dispositivo tienen los últimos dos bits menos significativos determinados por el voltaje en el puerto SA0.

4.2.3 Sensor LSM6D33 MiniMU-9 v5

Este sensor con dimensiones de 3x3x0.86mm, mostrado en la Fig.13, cuenta con un acelerómetro digital en 3D. El LSM6D33 se mantiene habilitado en modo de bajo consumo y de esta manera ofrece detección de movimiento con el menor consumo de recursos posibles. Este sensor puede ser utilizado como contador de pasos, como monitor de vibraciones, en el ahorro de energía inteligente, por mencionar algunos ejemplos de aplicaciones típicas.

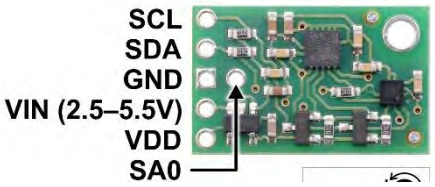


Figura 13: MiniMU-9 v5

En el presente proyecto se utilizó este sensor en lugar del sensor IMU con el que cuenta el robot por defecto, debido a su mayor resolución y menor consumo de energía. Además, este sensor también ofrece mejor capacidad en la reducción de ruido y mejores compensaciones Tasa-Cero.

En el presente proyecto se utiliza el giroscopio para detectar la rotación del robot, a partir de definir un punto fijo, si el robot sufre cambios en su dirección se utiliza la lectura realizada por el sensor para definir un valor de compensación en la velocidad que los motores recibirán y con ello corregir la dirección del robot móvil.

4.3 Conexión del sensor MiniMU-9 v5

Para poder utilizar este sensor en el Zumo 32U4 es necesario conectar el puerto SA0 a tierra, al hacer esto el bit menos significativo en la dirección del sensor cambia y es direccionado físicamente. El robot Zumo 32U4 utiliza por defecto el sensor de movimiento MiniMU-9 v3, el cual tiene la dirección 1101011b y dicha dirección coincide con la dirección por defecto del MiniMU-9 v5. Al conectar el puerto SA0 del MiniMU-9 v5 a tierra, la dirección del sensor cambia a 1101010b, por lo que ya no hay conflicto entre las direcciones y podemos utilizar este sensor. En la Fig. 14 se muestran las conexiones necesarias para utilizar el sensor.

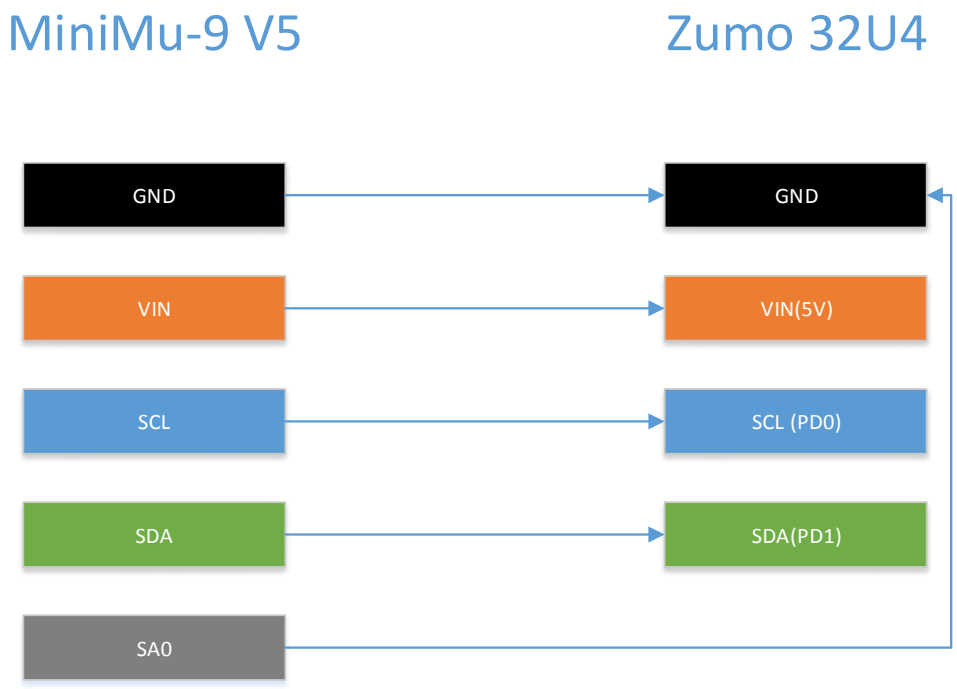


Figura 14: Conexión Zumo 32U4 y MiniMU-9 v5

4.4 Rango del sensor en la detección del movimiento.

En la gráfica de la Fig. 15 podemos observar el rango, que va desde -4000 a 4000, estos valores corresponden a los movimientos esperados en el proyecto, es decir, movimientos no tan abruptos. La gráfica de la Fig. 16 presenta movimientos abruptos y el rango de éstos.

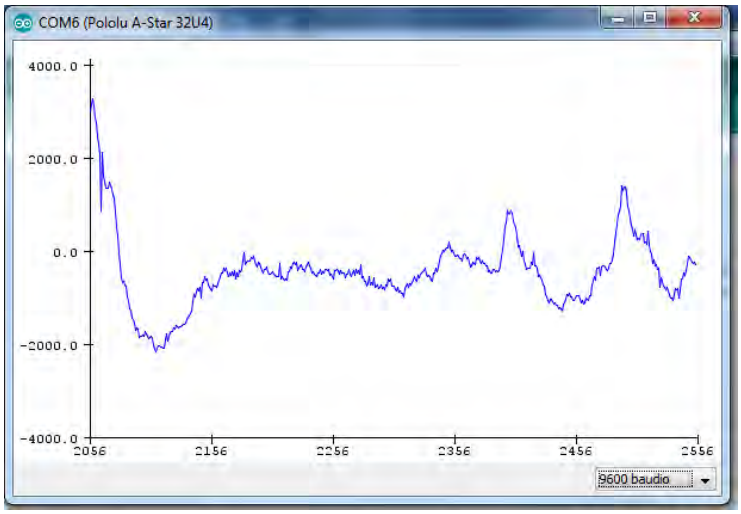


Figura 15: Primeras lecturas del sensor

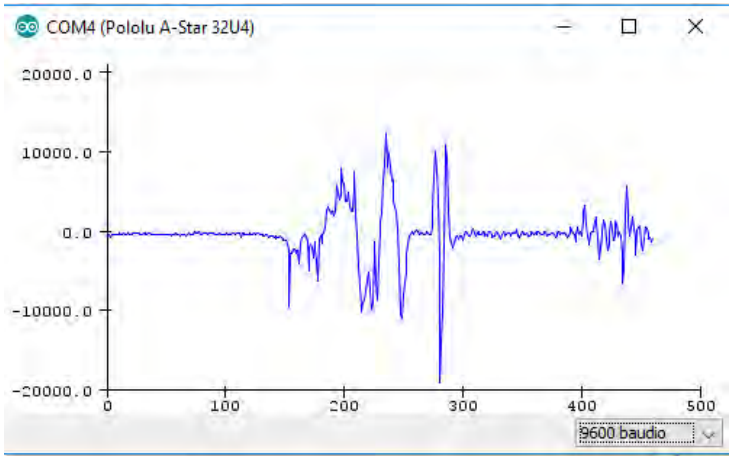


Figura 16: Movimientos abruptos del sensor

En el presente proyecto los movimientos esperados no son abruptos, con apoyo de las gráficas, hemos decidido utilizar umbrales asimétricos. Con base en pruebas experimentales se definieron los umbrales como asimétricos, no hay un método que nos permita asegurar el mejor diseño en la definición de los umbrales. En el proyecto se realizaron diversas pruebas para definir los rangos de trabajo de cada umbral.

En el proyecto necesitamos definir umbrales, lambdas o puntos de control que definirán los rangos en los cuales una función se activará o no. Analizando los rangos de operación del sensor de movimiento definiremos entonces los umbrales óptimos, cabe señalar que los movimientos o perturbaciones esperadas sobre el robot no deberían ser tan abruptos. Es importante analizar esto, porque si en los umbrales consideramos un movimiento máximo de 1000, por dar un ejemplo, pero el movimiento registrado es de 20000 entonces ninguna función se activará y por supuesto ese valor perdido afectará el resultado final devuelto por la red neuronal.

Como se puede apreciar en la Fig. 17, el valor que devuelve el sensor nunca es cero, esto se debe a los movimientos pequeños y la presencia de ruido electrónico. Cuando el robot sigue la trayectoria definida, los valores que el sensor devuelve son menores a mil, oscilan entre 300 y 800.

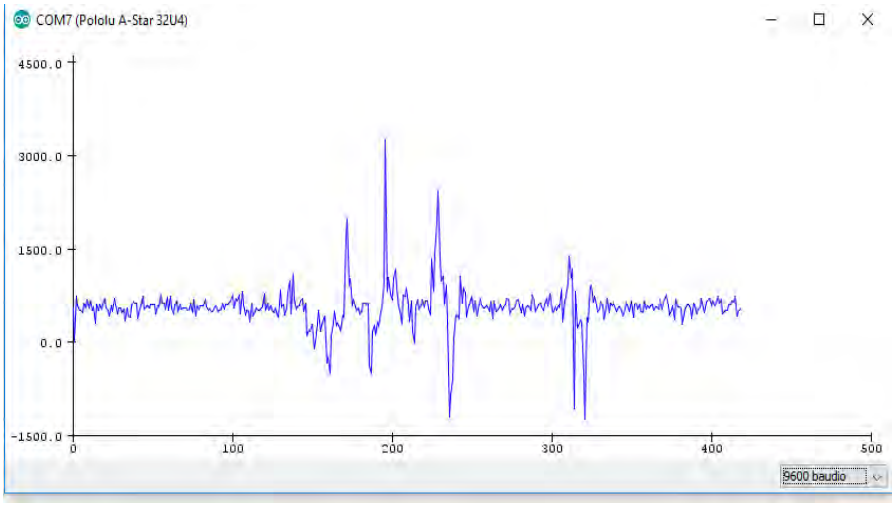


Figura 17: Lectura estable del sensor

El punto de control almacena valores de tipo entero, al calcular la diferencia del punto de control menos el valor actual del sensor, el valor resultante es muy pequeño. Al almacenarse en la memoria se guarda

como cero y de esa manera logramos estabilidad. Sin embargo, esta característica de la memoria nos presenta un problema importante que analizaremos en fases posteriores del proyecto.

4.5 Configuración del punto de control

Cuando el sensor MiniMU-9 v5 es energizado detecta la posición en la que se encuentra y la define como su origen. El cambio en la detección de los movimientos es a partir de que el robot ha definido las coordenadas de su origen. Tomando ventaja de este principio, realizamos un promedio de las primeras 5 lecturas del eje "z" tomadas por el sensor y dicho promedio será nuestro punto de control. Estas 5 lecturas se realizan en el método void setup () y el promedio se define como un valor constante; únicamente cambiará si el robot se apaga o si se reinicia. El procedimiento mencionado anteriormente podemos observarlo en la Fig. 18.

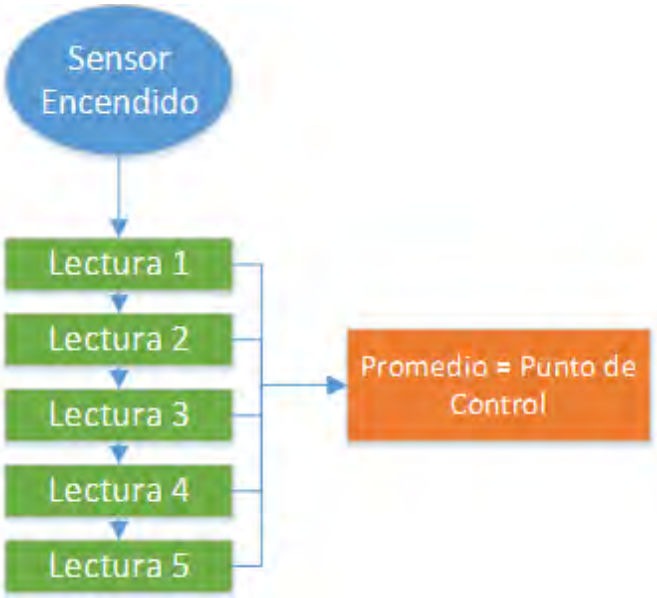


Figura 18: Punto de Control

4.6 Funciones de activación

Como se puede apreciar en la Fig. 19, primero debemos calcular el error, que es la diferencia entre el punto de control y el valor actual registrado por el sensor.

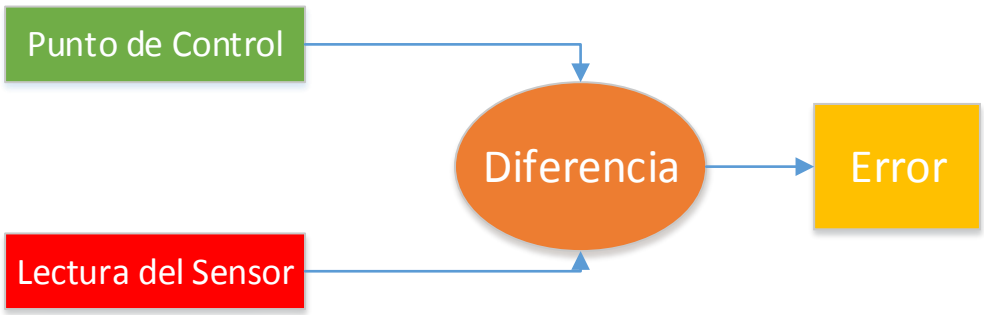


Figura 19: Cálculo del error

Antes de ingresar a las funciones de activación hacemos un escalamiento del error; los valores se reducen de 1000 a 1, con esto las funciones de activación logran realizar los cálculos en menos tiempo. La memoria del microcontrolador recibe valores pequeños, como resultado el microcontrolador requiere de menor poder de procesamiento.

En la Fig. 20 podemos analizar el comportamiento de la RNA de dos funciones de activación. El diseño de estas dos funciones está basado en la experimentacion, con este diseño se obtuvieron óptimos resultados de funcionamiento. Posteriormente, en el proyecto se analizan las diferentes configuraciones tanto de la RNA con dos funciones de activación como de la RNA con cuatro funciones.

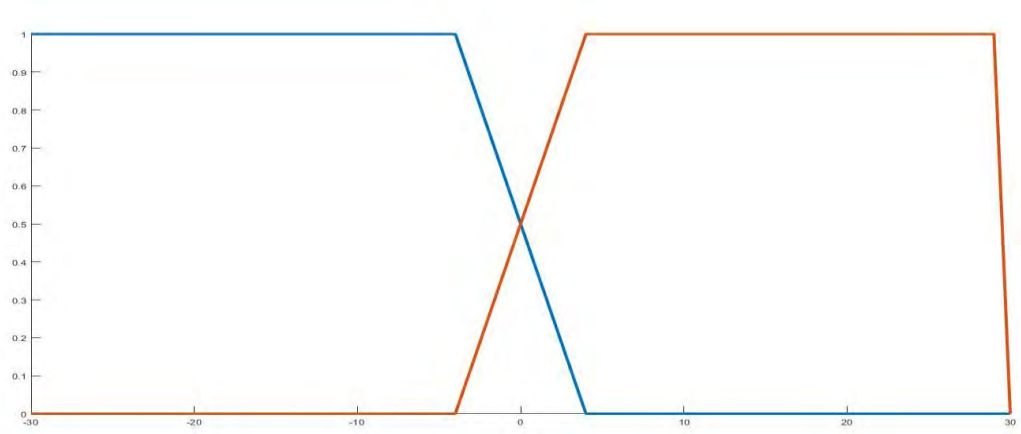


Figura 20: Gráfica de la RNA con dos funciones de activación

En la Fig. 21 podemos analizar el comportamiento de la RNA con cuatro funciones de activación. Las B-Splines son asimétricas. El diseño de las funciones se determinó con base en pruebas experimentales y fueron con las que se obtuvo mejor desempeño en el seguimiento de la trayectoria. Las funciones de activación mapean el valor de la variable de entrada a un valor correspondiente a la función de activación que se active.

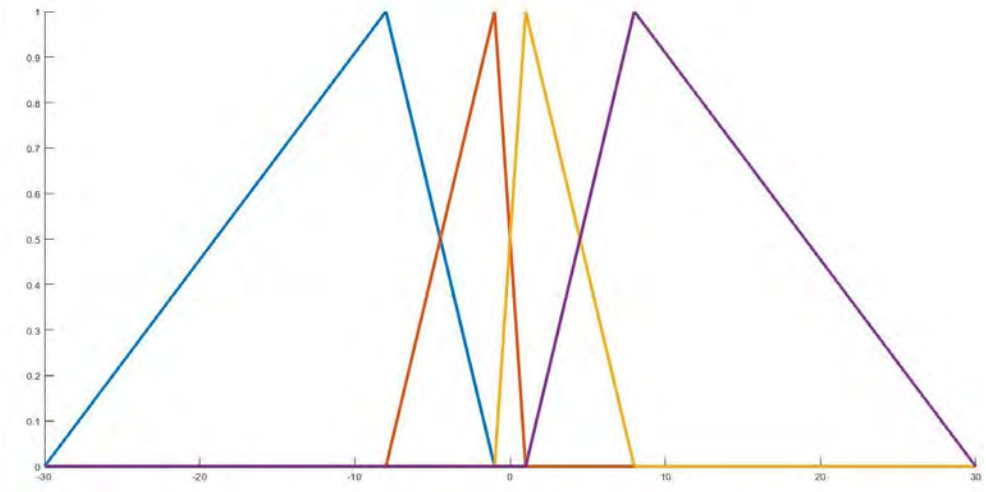


Figura 21: Gráfica de la RNA con cuatro funciones de activación

En el caso de que una función de activación no se active por la variable de entrada se asigna el valor de 0. De esta forma, si la configuración de la RNA cuenta con cuatro funciones de activación, entonces en la salida tendremos un vector de cuatro componentes. Lo esencial de este proceso es que solamente se pueden activar como máximo dos funciones en todo momento, de esta manera reducimos los datos necesarios para el procesamiento.

4.7 Cálculo del tamaño del vector resultante

Las funciones de activación nos devuelven como resultado un vector. El número de componentes del vector puede no ser el mismo número que el de las funciones de activación. Es decir, el error puede estar ubicado como máximo en el rango de dos funciones de activación. El siguiente paso es calcular el tamaño del vector; el tamaño lo obtenemos al calcular la norma. Posteriormente dividimos el error entre la norma, este proceso se muestra en la Fig. 22.

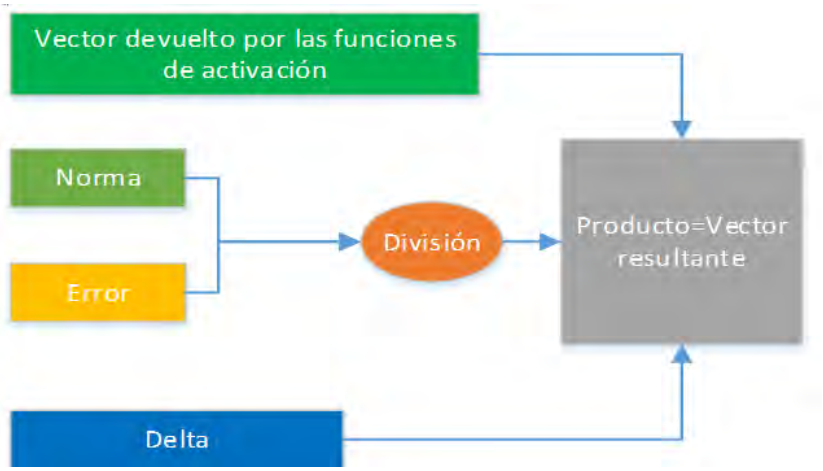


Figura 22: Cálculo del vector resultante

El valor del "delta" que es la regla de aprendizaje basado en el gradiente descendente el cual actualiza el valor de los pesos con cada iteración para minimizar el error, nos permite que el aprendizaje en línea de la RNA converja en una solución. Con base en la experimentación definimos el delta con un valor de 0.5, con un valor superior el algoritmo de la RNA no converge. El producto entre el vector devuelto por las funciones de activación, el delta y la división, es el vector resultante.

4.8 Aprendizaje de la red

La Fig. 23 corresponde a la actualización de los pesos. La memoria está compuesta por 2 arreglos. Los componentes de los dos arreglos utilizados en este proceso se inicializan en ceros. Los arreglos almacenan los componentes del vector resultante, un arreglo almacena el valor actual y los valores pasados son almacenados en el segundo arreglo.

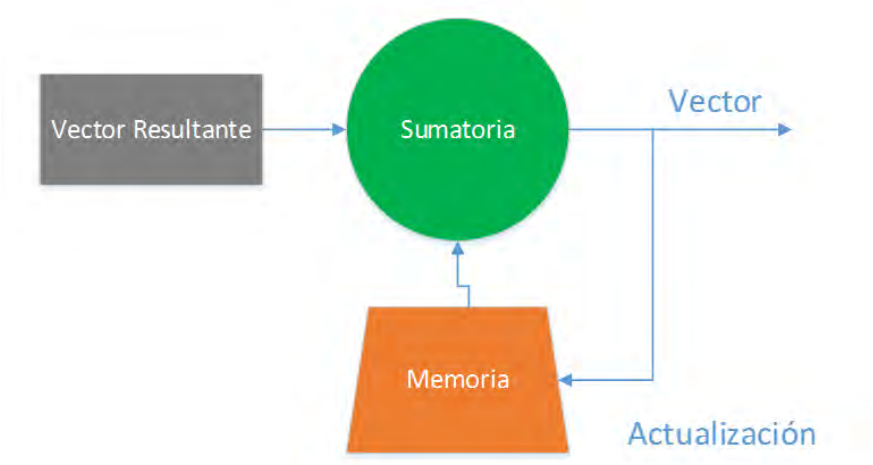


Figura 23: Aprendizaje de la RNA

En el proceso de actualización de los pesos o aprendizaje, la RNA en cada iteración busca regresar a los valores iniciales. Como la memoria fue inicializada en cero, esto porque al iniciar el algoritmo el valor del error es cero. Cada vez que el error es diferente de cero la RNA ajusta los valores intentando regresar la memoria a sus valores iniciales.

4.9 Cálculo de la salida de la red neuronal

El último proceso es calcular la salida de la RNA, proceso que podemos observar en la Fig. 24. El valor final devuelto por la RNA es un escalar, este es utilizado para realizar la compensación en la velocidad de los motores. La salida de la RNA es el producto punto del vector devuelto por las funciones de activación por el vector resultante del aprendizaje de la red neuronal.

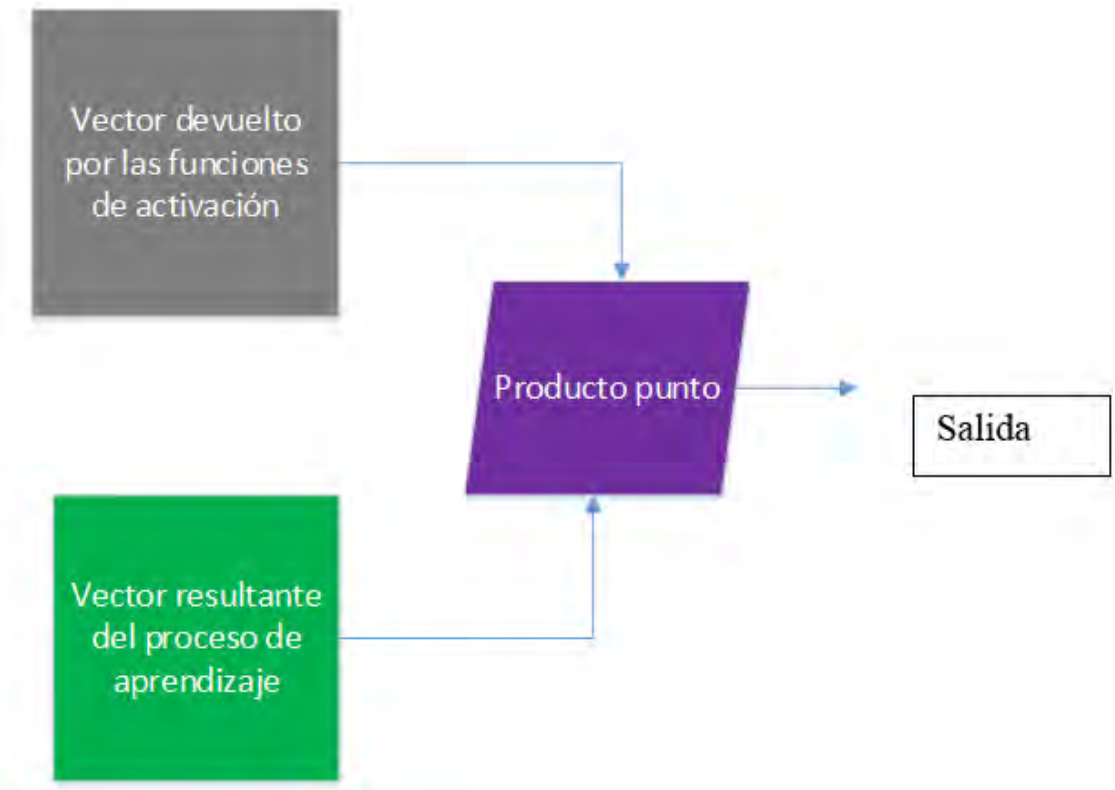


Figura 24: Salida de la RNA B-Spline

Capítulo 5. Resultados Experimentales

En el robot se realizaron pruebas con un algoritmo de Control Proporcional, en esta sección analizaremos el funcionamiento del Control Proporcional y el funcionamiento de la Red Neuronal. Analizaremos las mejoras en el control proporcionadas por la RNA y los errores o deficiencias encontradas.

5.1 Pruebas de la RNA B-Spline de dos funciones de activación

Esta configuración de la RNA B-Spline es la más sencilla utilizada en el proyecto, en la Fig. 25 observamos su comportamiento. El error puede encontrarse entre 3 intervalos: el intervalo creado por el umbral negativo más lejano de 0 y el umbral negativo más cercano a 0, el segundo intervalo puede encontrarse en el umbral negativo más cercano a 0 y el umbral positivo más cercano a 0 y finalmente el tercer intervalo puede encontrarse el error en el intervalo creado por el umbral positivo más cercano a 0 y el umbral positivo más lejano de 0.

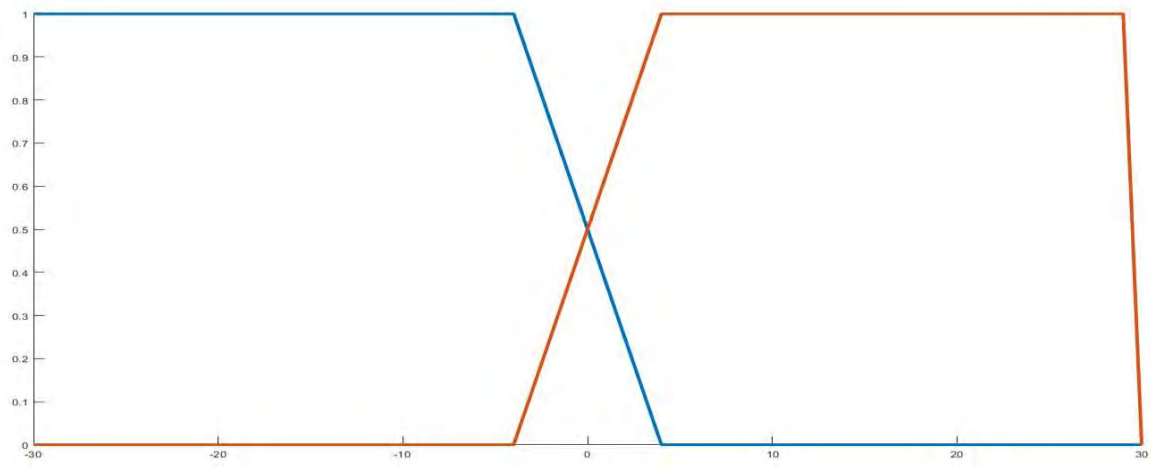


Figura 25: Configuración de la RNA con dos funciones de activación

En la gráfica dibujada en la Fig. 26 observamos las correcciones más pequeñas realizadas por esta RNA en la salida. Las lambdas o umbrales más cercanos a 0 son -4 y 4.

Los valores devueltos por la RNA son altos, considerando una velocidad inicial de 100. Son valores superiores a 10, esto significa que un motor tendrá una velocidad mayor a 110 y el segundo motor tendrá una velocidad menor a 90. El análisis revela que ésta configuración de la RNA no es lo suficientemente sensible para realizar ajustes más pequeños en la velocidad de los motores

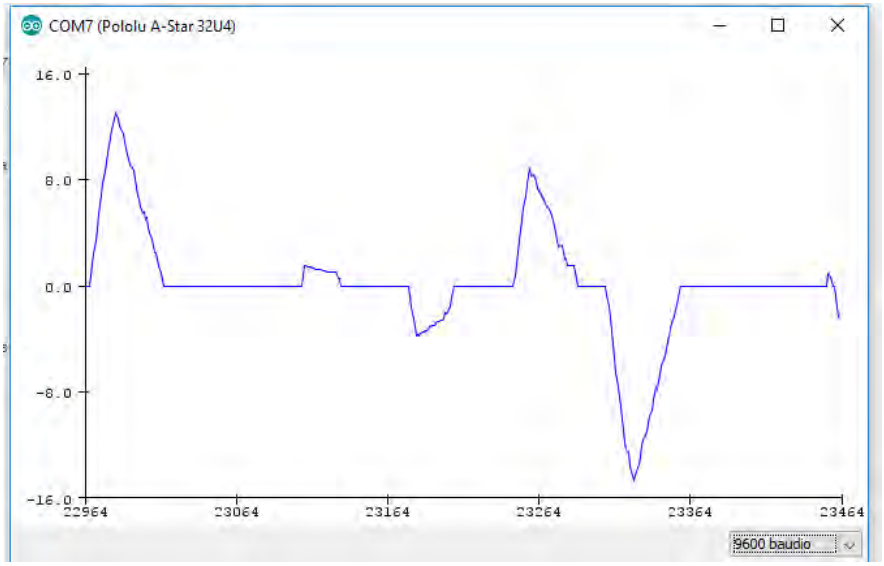


Figura 26: Valor de salida de la primera configuración de la RNA con dos funciones de activación

Cuando cambiamos el valor de las lambdas más cercanas a cero, y las definimos en -1 y 1, obtenemos más sensibilidad en la RNA, en la Fig. 27 podemos observar el comportamiento de esta configuración.

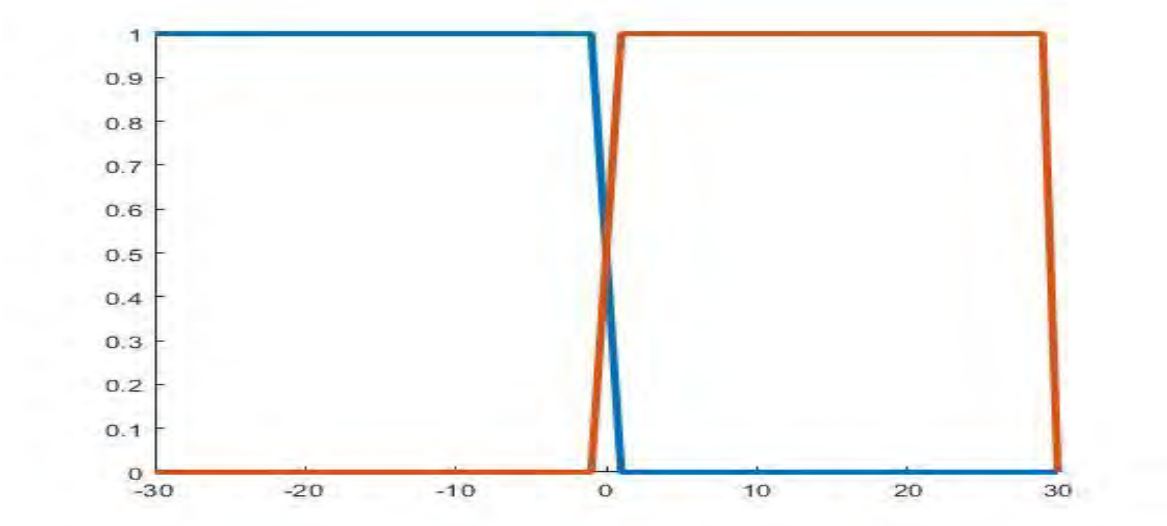


Figura 27: Comportamiento de la RNA con umbrales en -1 y 1

La Fig. 28 corresponde a los valores más pequeños en la salida de la RNA. El escalar devuelto por la RNA es menor en esta configuración, en la configuración anterior los valores eran mayores a 10. En esta configuración son menores a 10, es decir, ahora podemos realizar ajustes más pequeños en la velocidad de los motores.

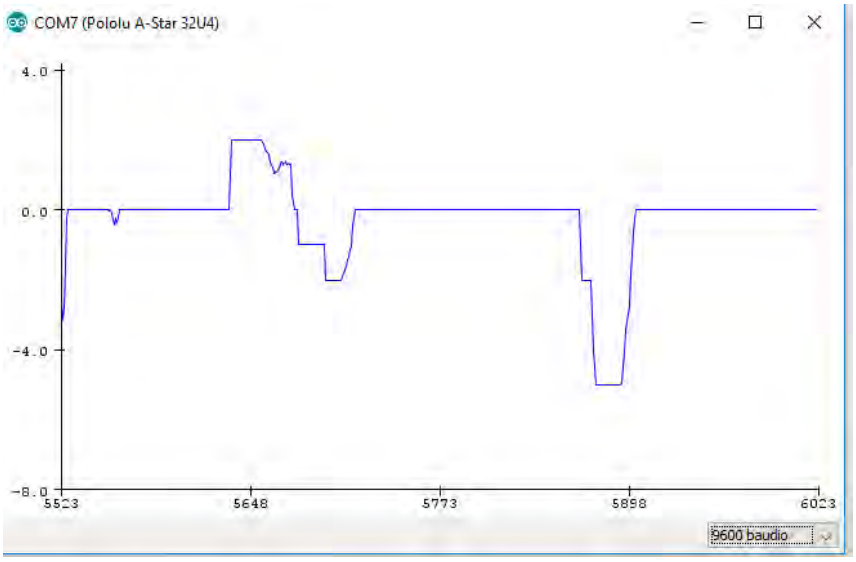


Figura 28: Salida de la RNA con umbrales en -1 y 1

La configuración más básica de la RNA funciona bien cuando el robot experimenta perturbaciones moderadas. En la Fig. 29 se presenta el comportamiento de la RNA cuando las lambdas más cercanas a 0 son -4 y 4. En la Fig. 30 observamos los valores de compensación en la salida de la RNA con esta configuración.

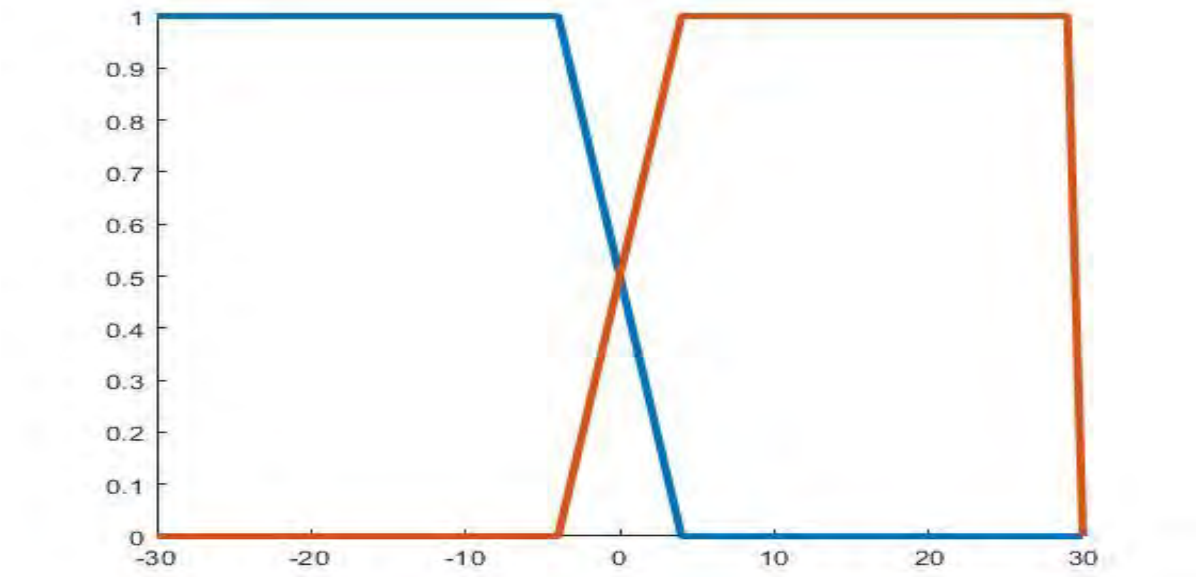


Figura 29: Comportamiento de la RNA con umbrales en -4 y 4

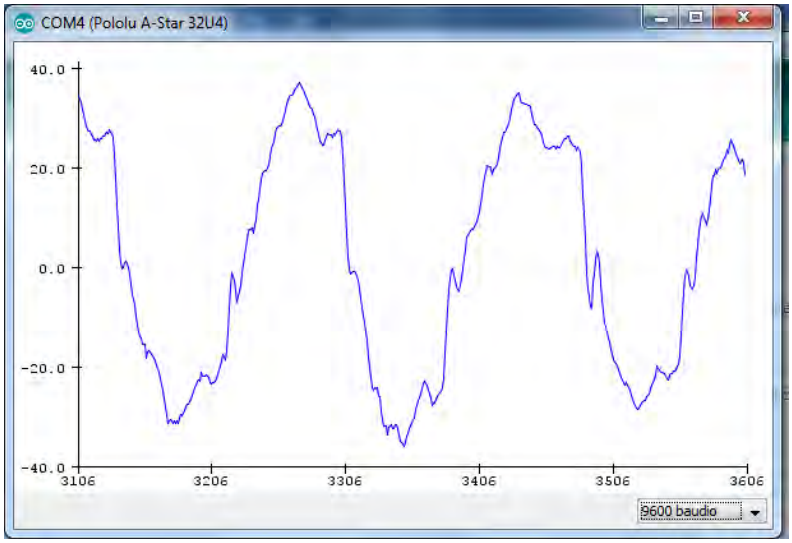


Figura 30: Salida de la RNA con umbrales en -4 y 4

La Fig. 31 corresponde al comportamiento de la RNA cuando los umbrales más cercanos a 0 son -1 y 1. El comportamiento de la RNA en estas dos configuraciones no difiere mucho cuando las perturbaciones son moderadas, esto puede notarse al analizar la salida de la RNA en la Fig. 32.

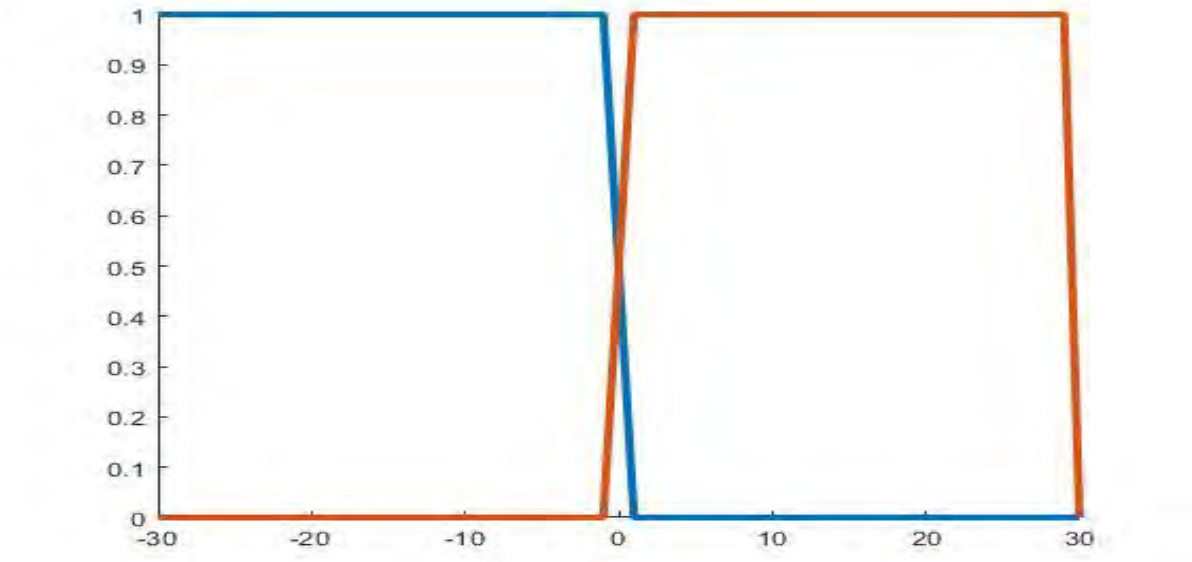


Figura 31: Comportamiento de la RNA con umbrales en -1 y 1

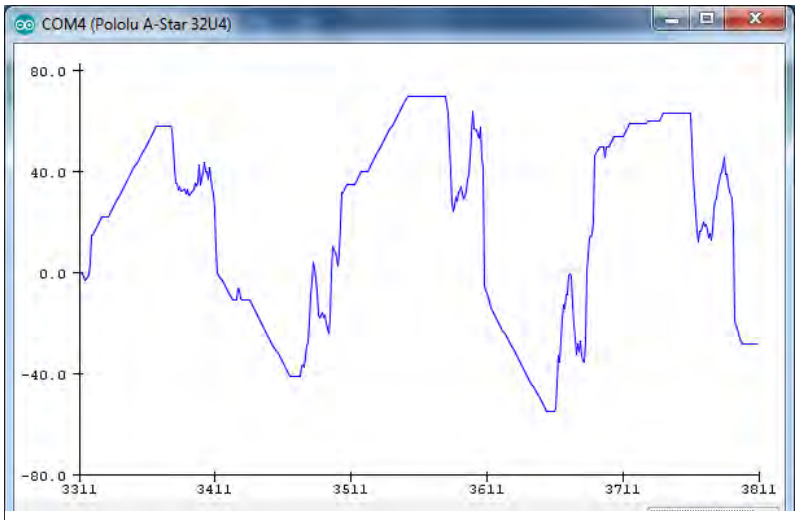


Figura 32: Salida de la RNA con umbrales en -1 y 1

Cuando las perturbaciones se vuelven mayores, la RNA con dos funciones de activación se ve afectada. El problema surge cuando los valores que deben ser almacenados en la memoria se vuelven demasiado grandes, entonces la memoria se vuelve inestable y perdemos el control sobre la RNA. Las perturbaciones generan que el valor de compensación sea el mayor posible, como podemos observar en la Fig. 33.

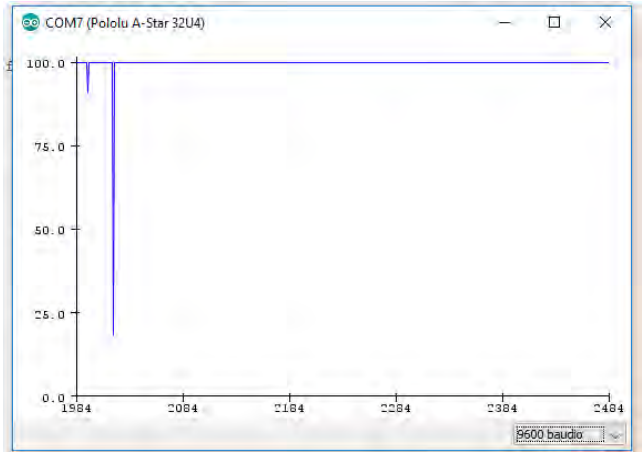


Figura 33: Salida de la RNA con perturbaciones fuertes

Cuando esto sucede, perdemos el control sobre la RNA, la Fig. 34 nos permite observar que la salida de la RNA es el valor máximo de compensación. En este momento el problema de "cabeceo" está fuera de control.

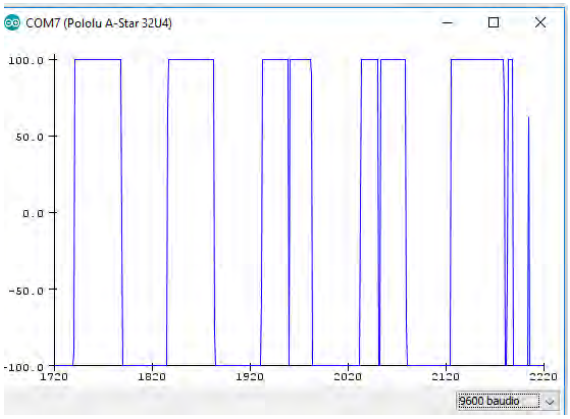


Figura 34: Problema de "cabeceo" en la RNA de dos funciones de activación

5.1.2 Pruebas de la RNA B-Spline de dos funciones de activación a velocidades diferentes.

Las velocidades que se pueden configurar en el robot Zumo 32U4 van desde cero hasta cuatrocientos. Para iniciar con las pruebas en esta sección, el parámetro de velocidad será de doscientos.

La RNA funciona correctamente aun cuando la velocidad es de 200, cuando aumentamos la velocidad a 300 el funcionamiento sigue siendo correcto. Cuando la velocidad es de 400 el robot tiene algunos problemas para corregir su dirección, a esta velocidad el robot experimenta un ligero “cabeceo”. El “cabeceo” experimentado dura pocos segundos y luego el robot logra estabilizarse.

No es posible realizar las mismas pruebas con una sola configuración del Control Proporcional. Lo que se ajusta en el control Proporcional es su ganancia. Si este valor es pequeño, la respuesta en el tiempo del controlador es lento. Si es grande el valor, es más rápido, pero se arriesga estabilidad. El problema del “cabeceo” en el Control Proporcional se presenta incluso cuando la velocidad es baja.

La RNA B-Spline puede ser configurada una sola vez y funcionar a diferentes velocidades. El problema del “cabeceo” en esta configuración de la RNA surge cuando las perturbaciones son muy fuertes o cuando las perturbaciones son moderadas pero la velocidad es de 400.

5.2 Pruebas de la RNA B-Spline de cuatro funciones de activación

La RNA B-Spline más compleja utilizada en el proyecto está compuesta por cuatro funciones de activación. Los rangos de detección de las funciones de activación son asimétricos, el diseño de las funciones de determinó mediante experimentación.

Como sucede con la RNA de dos funciones de activación, perdemos sensibilidad. Este problema es debido a que los datos que recibe la memoria son pequeños. La memoria guarda datos de tipo entero, al recibir valores menores a uno los guarda como cero.

La Fig. 35 corresponde al comportamiento de la RNA cuando las lambdas más cercanas a cero son -4 y 4. En la Fig. 36 podemos observar los valores de compensación devuelto en la salida de esta configuración.

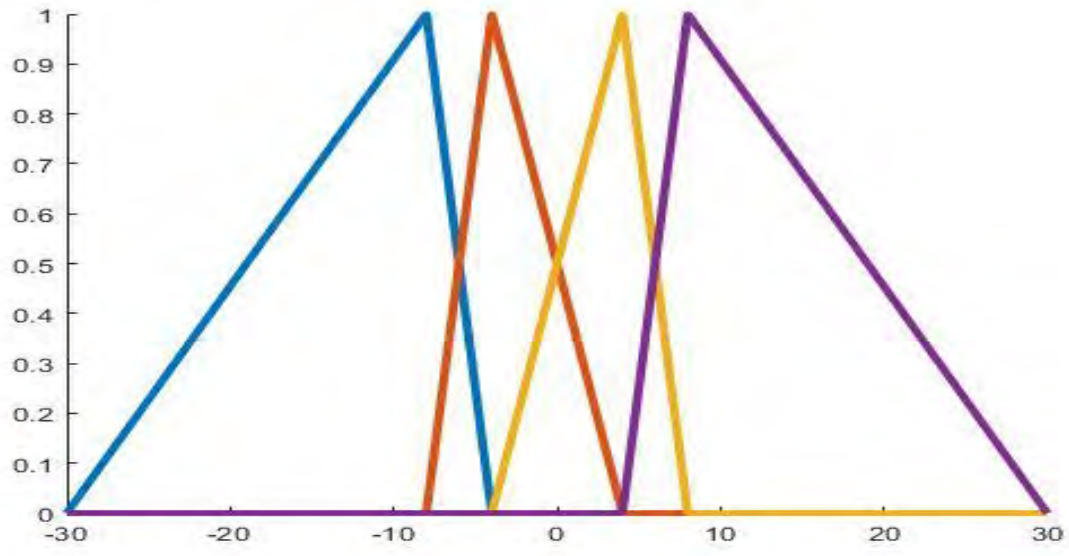


Figura 35: Comportamiento de la RNA con umbrales en -4 y 4

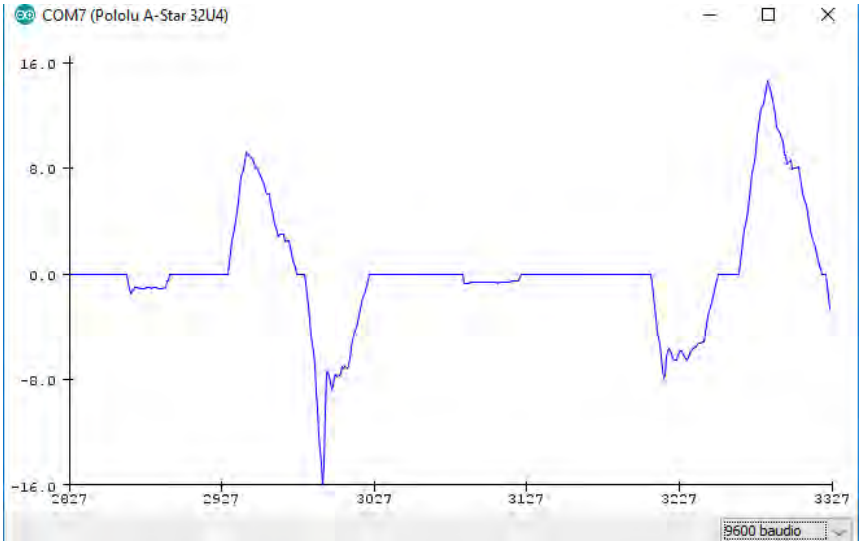


Figura 36: Salida de la RNA con umbrales en -4 y 4

En la Fig. 37 podemos observar el comportamiento de la RNA con cuatro funciones de activación cuando los umbrales más cercanos a 0 son -1 y 1. La salida devuelta por la RNA en esta configuración podemos observarla en la Fig. 38.

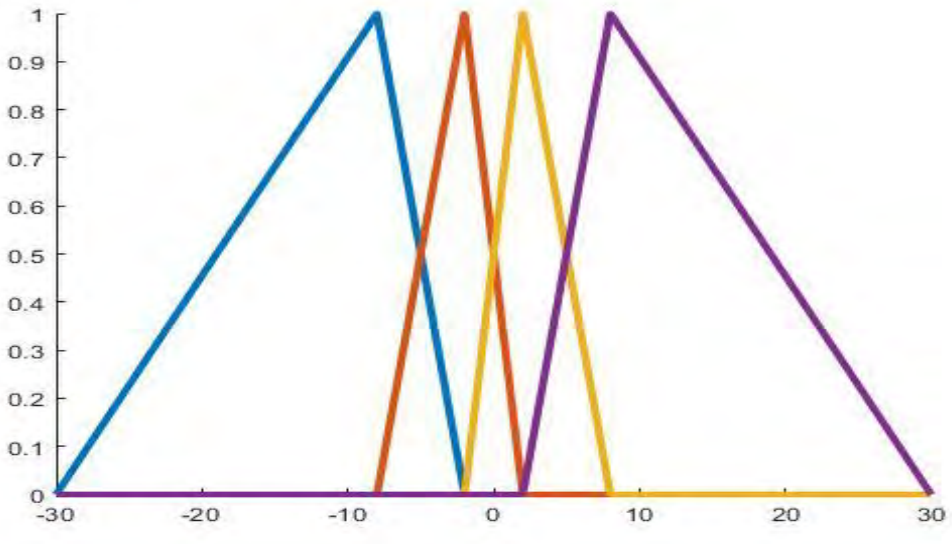


Figura 37: Comportamiento de la RNA con umbrales en -1 y 1

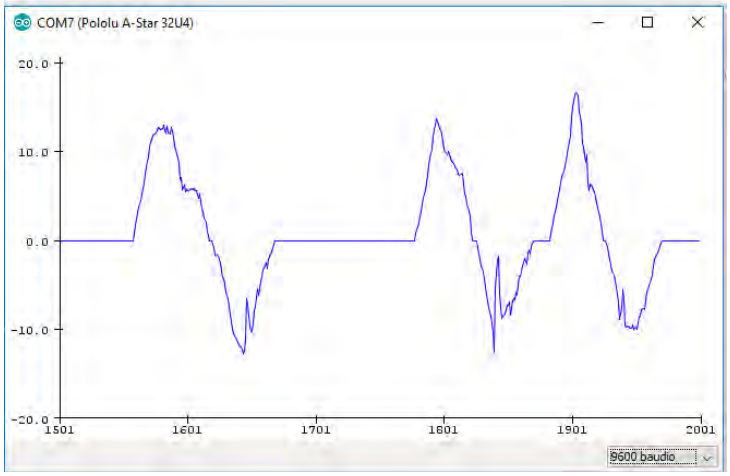


Figura 38: Salida de la RNA con umbrales en -1 y 1

Cuando realizamos ajustes en la RNA, definiendo las cuatro lambdas más cercanas a cero en un rango de -2 y 2, obtenemos la mayor sensibilidad que la RNA nos permite. En la Fig. 39 podemos observar que los valores de compensación más pequeños en esta configuración son mayores que los observados en la figura anterior.

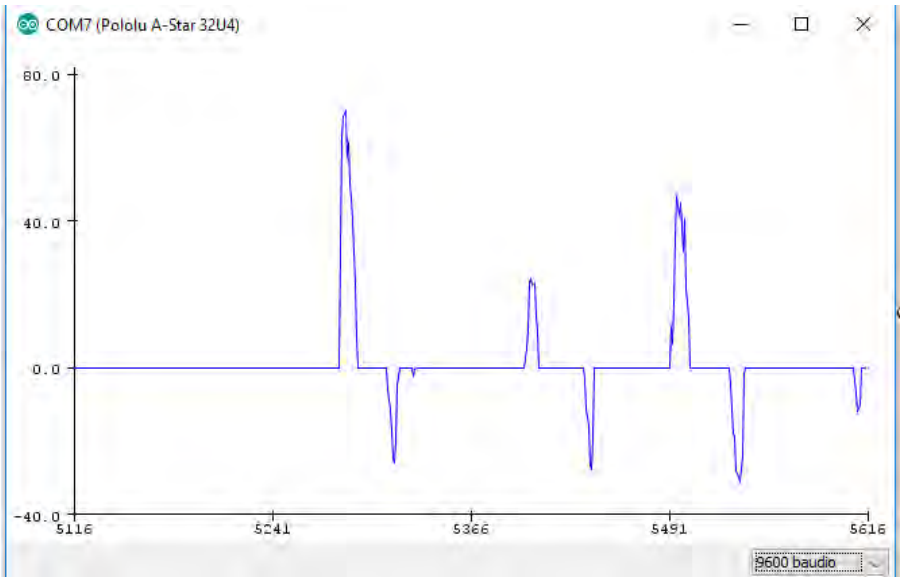


Figura 29: Salida de la RNA con umbrales en -2 y 2

La RNA con cuatro neuronas es más resistente a las fuertes perturbaciones. En la RNA con dos neuronas perdemos el control cuando las perturbaciones superan un cierto nivel de intensidad, este problema ocurre porque la memoria recibe valores muy grandes. La Fig. 40 es una representación de lo que ocurre en la memoria. En esta representación "Dato1" corresponde a los valores que recibe la memoria cuando las perturbaciones son moderadas; "Dato2" corresponde a los valores que recibe la memoria cuando las perturbaciones son fuertes. "Dato2" es mayor a los datos que la memoria puede almacenar, este "desbordamiento" causa la pérdida parcial del control de la memoria.

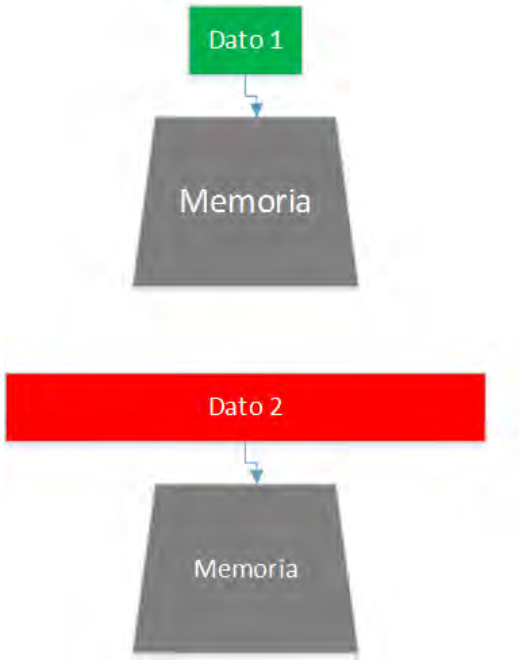


Figura 40: Representación del problema de memoria

La Fig. 41 corresponde al funcionamiento de la RNA de cuatro neuronas cuando percibe fuertes perturbaciones. La RNA puede mantener la memoria controlada, sin embargo, después de experimentar fuertes movimientos la red neuronal cuenta con solamente dos neuronas para el control.

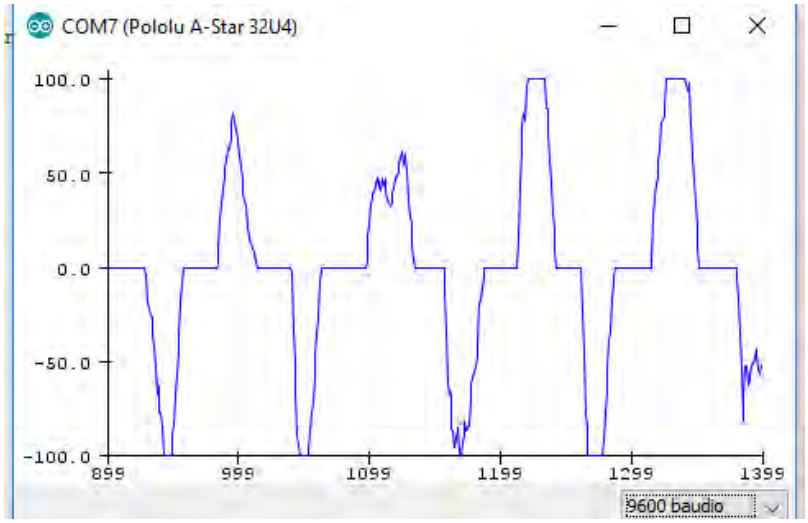


Figura 41: Salida de la RNA experimentando fuertes perturbaciones

En la Fig. 42 podemos observar el comportamiento de la memoria cuando las perturbaciones son fuertes. Las líneas de color azul y naranja representan el comportamiento de las neuronas que se mantienen en control. Las líneas de color verde y rojo representan el comportamiento de las neuronas que ya no están bajo control y por ello oscilan los valores calculados por estas funciones de activación.

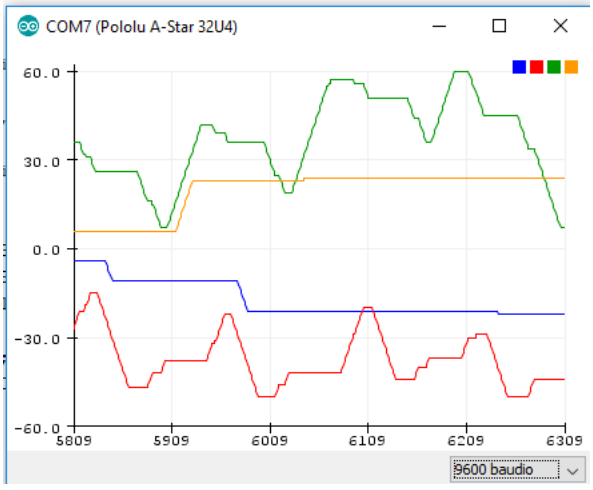


Figura 42: Comportamiento de la memoria

La Fig. 43 corresponde al comportamiento de la RNA de cuatro funciones de activación cuando tenemos control sobre las cuatro neuronas.

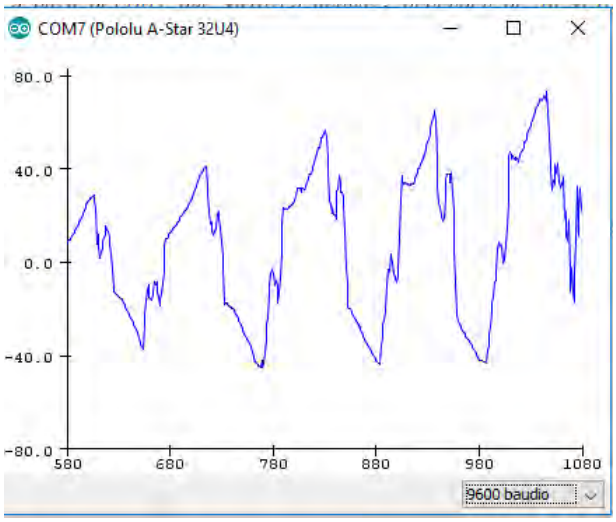


Figura 43: Comportamiento de la RNA cuando las cuatro neuronas están bajo control

En la Fig. 44 se puede observar el comportamiento de la RNA cuando perdemos el control sobre dos de las cuatro neuronas. Los valores que actúan sobre los motores son mayores, sin embargo, no son los valores máximos en todo momento.

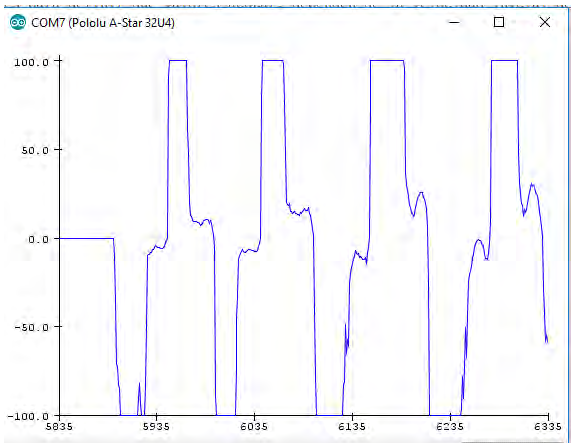


Figura 44: Comportamiento de la RNA cuando se pierde el control de dos neuronas

5.2.2 Pruebas de la RNA B-Spline de cuatro funciones de activación a velocidades diferentes

Las pruebas realizadas con esta configuración de la RNA fueron con velocidades de 200,300 y 400. Con esta configuración la RNA no presenta el problema de "cabeceo" que surge en la RNA con dos funciones de activación cuando la velocidad es de 400.

Al igual que la RNA con dos funciones de activación esta configuración no es lo suficientemente sensible para reaccionar cuando los movimientos son suaves. La ventaja que tiene la RNA con cuatro funciones de activación sobre la configuración que cuenta con dos funciones es que es más resistente a los movimientos abruptos.

Mantener la RNA bajo control no depende de la velocidad de los motores del robot en esta configuración con cuatro neuronas o en la configuración de la RNA con dos neuronas. Los movimientos abruptos producen en la RNA valores muy grandes que no pueden ser almacenados en la memoria. Independientemente de la velocidad, la RNA más resistente es la que cuenta con cuatro neuronas, sin embargo, el control depende solamente de dos de las cuatro neuronas después de experimentar fuertes perturbaciones. Perdemos de forma permanente el control sobre las dos neuronas restantes.

Conclusiones

Los resultados experimentales han demostrado que la RNA B-Spline es capaz de lograr el control de dirección del robot móvil. La red neuronal fue implementada en un microcontrolador de 8 bits con lo que se demostró que las funciones recursivas permiten realizar cálculos complejos con la menor capacidad de procesamiento.

Entre los objetivos particulares se encuentra la conexión de un modelo del sensor de movimiento más reciente y con características enfocadas en la eficiencia del consumo de energía. La adaptación de este sistema de control permite aprender sobre su funcionamiento, así como la estructura de la interfaz I2C. Al realizar la implementación del algoritmo de control elegido y el estudio de su funcionamiento permitió conocer las características del aprendizaje no supervisado. Lo anterior se logró mediante la implementación de la RNA B-Spline, logrando tiempos de respuesta relativamente cortos.

La RNA B-Spline es capaz de adaptarse a situaciones donde la velocidad inicial del robot es diferente, esta capacidad no se encuentra en el algoritmo de Control Proporcional. La configuración de la RNA B-Spline se define solo una vez y puede funcionar eficazmente aún si cambiamos el parámetro de la velocidad inicial. El algoritmo de Control Proporcional debe ser modificado en cada ocasión de manera que el parámetro de la velocidad inicial cambie, dependiendo del rango de los movimientos esperados. La adaptabilidad de la RNA B-Spline ofrece una ventaja notable sobre el algoritmo de Control Proporcional.

El funcionamiento de la RNA B-Spline es óptimo cuando las perturbaciones son moderadas, sin embargo, presenta problemas cuando las perturbaciones son muy pequeñas o son muy grandes. Cuando las perturbaciones son muy pequeñas la RNA B-Spline no es lo suficientemente sensible para detectarlas o en el caso contrario se pierde el control parcialmente sobre la memoria de la RNA.

Los valores que recibe la memoria son menores a uno, al ser de tipo entero, todos estos se guardan como cero. Lo cual puede ser un problema y este se puede solucionar en un trabajo futuro, ya que pueden emplearse técnicas para manipulación de datos enteros como datos fraccionarios.

La pérdida de control de la memoria se presenta cuando los datos exceden su capacidad de almacenamiento, esto le ocasiona problemas a la misma. Por lo que la configuración más sensible a éste es la RNA con dos funciones de activación; al tener esta RNA dos neuronas, los valores son mayores que los capturados en la configuración con cuatro neuronas. La RNA con cuatro neuronas es más resistente. Cuando las perturbaciones se vuelven abruptas se pierden dos neuronas y el control se mantiene en las dos restantes; ciertamente el control no es tan preciso como cuando aún las cuatro neuronas funcionan correctamente.

La implementación de Redes Neuronales Artificiales en microcontroladores de 8 bits es posible gracias al uso de funciones recursivas que no requieren gran capacidad de procesamiento. Para poder obtener mejores resultados se sugiere seguir explorando las técnicas que permitan manipular diferentes tipos de datos y crear librerías especiales para lograr este objetivo.

Bibliografía

[1] "Pololu - Zumo 32U4 Robot." [Online]. Available: https://www.pololu.com/category/170/zumo-32u4-robot. [Accessed: 07-May-2018].

[2] "DRV883x Low-Voltage H-Bridge Driver."

[3] J. Redrejo, "Desarrollo de sistemas de regulación y control," 2004.

[4] "8 bit microcontroller, what is an 8 bit microcontroller? Embedded, flash, etc - Future Electronics." [Online]. Available: http://www.futureelectronics.com/en/microcontrollers/8-bit-microcontroller.aspx. [Accessed: 31-May-2018].

[5] F. H. R. Leyva, "Robótica 2 . Modelado Cinemática de Robots Sistema de Coordenadas," 2012.

[6] S. Haykin, *Neural Networks and Learning Machines*, vol. 3. 2008.

[7] C. A. Ruiz, M. Susana, B. Autor, : Damián, and J. Matich, "Universidad Tecnológica Nacional – Facultad Regional Rosario Departamento de Ingeniería Química Grupo de Investigación Aplicada a la Ingeniería Química (GIAIQ) Redes Neuronales: Conceptos Básicos y Aplicaciones," vol. 1, p. 55, 2001.

[8] V. M. Sanchez, R. Barbosa, L. G. Arriaga, and J. M. Ramirez, "Real time control of air feed system in a PEM fuel cell by means of an adaptive neural-network," *Int. J. Hydrogen Energy*, vol. 39, no. 29, pp. 16750–16762, 2014.

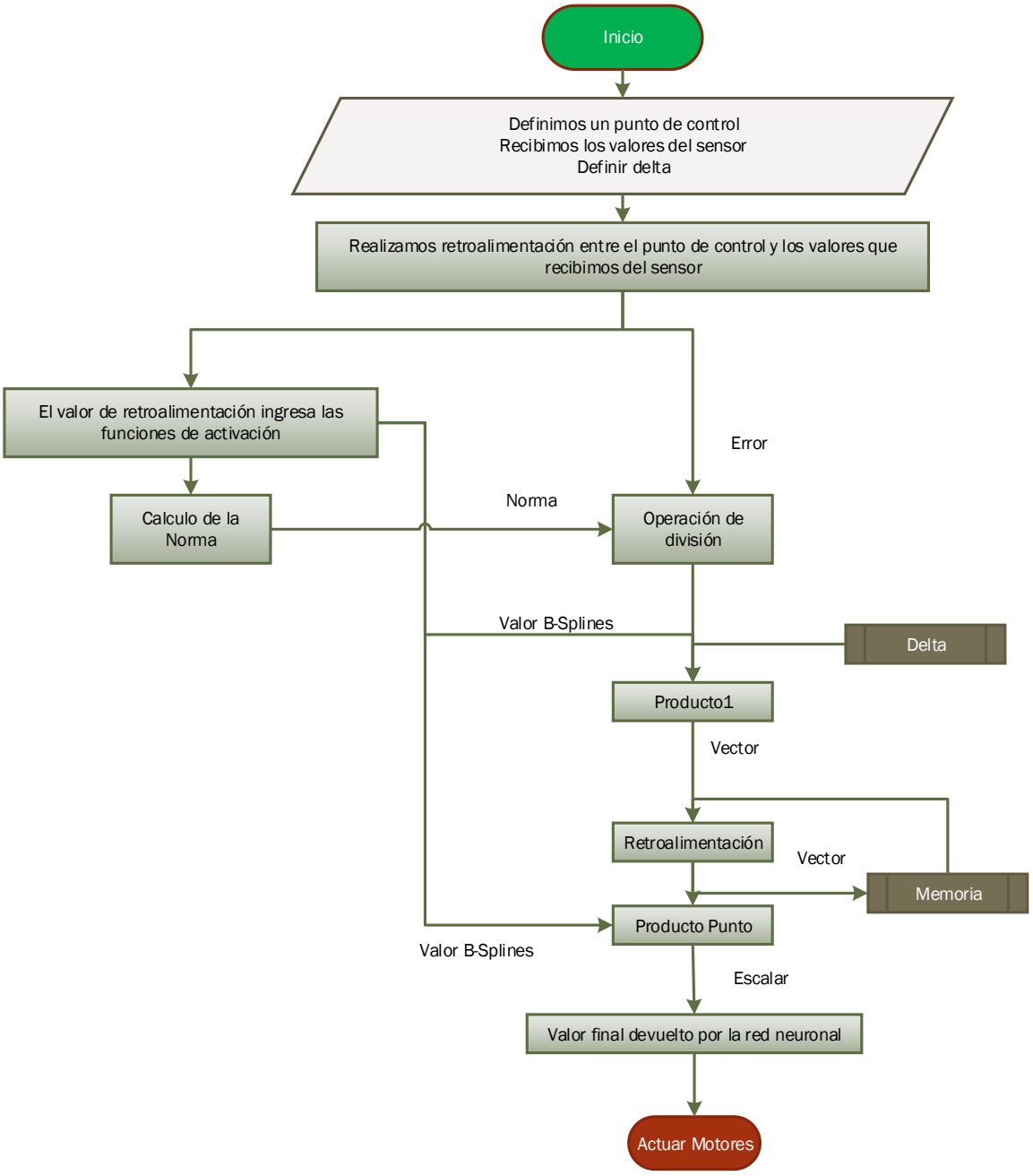
[9] V. Sánchez, J. M. Ramírez, and G. Arriaga, "On-line air supply control of PEM fuel cell by an adaptive neural network," *North Am. Power Symp. 2010, NAPS 2010*, pp. 1–6, 2010.

[10] "Arduino - Environment." [Online]. Available: https://www.arduino.cc/en/Guide/Environment. [Accessed: 04-May-2018].

[11] Pololu Corporation, "Pololu Zumo 32U4 Robot User's Guide," 2015.

[12] "I2C | Aprendiendo Arduino." [Online]. Available: https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/. [Accessed: 19-Jun-2018].

Anexo A. Algoritmo Principal



Anexo B. Código Fuente Arduino, para dos funciones de activación.

```
#include <Wire.h>
#include <LSM6.h>
#include <Zumo32U4.h>

Zumo32U4Motors motores;

//con este arreglo guardamos las primeras 5 lecturas del sensor en el
método setup();
float punto_control_arreglo[5];

//Variable donde se guarda el promedio de las 5 lecturas
float punto_control;


//umbrales para medir el error
int  lambda_1, lambda0, lambda1,lambda2;


//Los arreglos utilizados como los pesos, deben de ser de tipo entero.

int w[]= {0,0};     //Los pesos de la red, los inicializamos con 0
int w1[]= {0,0};   //con este arreglo actualizamos los pesos de la red

float B,C;         //variables para las B-Spline


LSM6 sensor;         //Declaramos el MiniMu-9 V5 con el nombre "sensor"
char report[80];
int vel;              //Es la velocidad que reciben los motores


void setup()
{
  Serial.begin(9600);
  Wire.begin();

//La siguiente condición nos sirve para verificar que el sensor funciona
correctamente
  if (!sensor.init())
  {

    Serial.println(¡” Ha fallado la conexión con el sensor!”);

  }

//El sensor inicia con su configuración por defecto, pero podemos cambiar
registros si se necesita en el programa
  sensor.enableDefault();


  for(int i=0; i <5; i++){

  sensor.read();        //iniciamos la lectura de los datos
```

```
  float lectura_inicial = (int)sensor.g.z;

   punto_control_arreglo[i]=lectura_inicial;


  }


//Calculamos el promedio de las primeras 5 lecturas, lo guardamos en
punto_control; que es nuestro "setpoint"
punto_control=(const int) ((punto_control_arreglo[0] +
punto_control_arreglo[1] + punto_control_arreglo[2] +
punto_control_arreglo[3]+punto_control_arreglo[4]+punto_control_arreglo[5])
/6);


delay (3000);
}

void loop()
{

  sensor.read();  //iniciamos las lecturas del sensor

//Leemos los datos del giroscopio del eje "z" y los asignamos a la variable
lectura_sensor
float lectura_sensor = (int)sensor.g.z;
//Serial.println(lectura_sensor,DEC);

//El error es la diferencia entre el setpoint y el valor que se recibe
//del sensor
float error = ((punto_control)  - (lectura_sensor)) ;
//Serial.println(error,DEC);


float err = error/1000;           //Escalamiento del error
//Serial.println(err,DEC);
int a,b,c,d,e;

//Serial.println(err,DEC);   //sirve para ver el valor del error antes de
todo el procesamiento

// el error ingresa a las funciones de activación B-Spline

//las lambdas son los umbrales necesarios para detectar los movimientos y
nos sirven para construir las B-Spline
lambda_1 =-30;
lambda0 = -4;
lambda1= 4;
lambda2 = 30;


//Si el error se encuentra en el umbral entonces se activa la función
if (err >= lambda_1 && err < lambda0){
  b=1;
} else{
  b=0;
```

```
}

if (err >= lambda0 && err < lambda1){
  c=1;
}else{
  c=0;
}
if (err >= lambda1 && err <= lambda2){
  d=1;
}else{
  d=0;
}


//Obtenemos un vector de 2 componentes, son dos splines de segundo orden

B = b + ((lambda1 - err)/(lambda1 - lambda0))*c;
C = ((err - lambda0)/(lambda1 - lambda0))*c + d;


//calcular la norma del vector que devuelven las B-Spline
float Y= (B*B) + (C*C) ;
float norma = (sqrt(Y)*1000);
//Serial.println(norma,DEC);



//dividir el error entre la norma
float division1= (error) / (norma);
//Serial.println((division1)/2,DEC);


// el siguiente producto recibe la norma, el vector de la b-splines y un
delta
float delta = 0.5;
/** Cuando delta tiene un valor de 1, salida de la red neuronal que puede
regresar a 0, es decir, tenemos inestabilidad
 *  Cuando delta tiene un valor de 1.5, también tenemos inestabilidad en la
red neuronal
**/

//producto1 es un vector con 2 componentes
//imprime 0 sin movimiento. Con movimiento imprime valores negativos
float producto1b= division1*B*delta;

//Imprime 0 sin movimiento. Con movimiento imprime valores positivos
float producto1c= division1*C*delta;

//Serial.println(producto1c,DEC);

//proceso de aprendizaje de la red neuronal

/**Debido a que en la memoria solo se guardan números enteros, los valores
pequeños que corresponden al procesamiento
 * de los movimientos suaves, se guardan como 0, esto implica que perdemos
sensibilidad
**/
```

```
w1[0]=w[0] + producto1b;
w1[1]=w[1] + producto1c;

//Las siguientes instrucciones son para analizar el comportamiento de la
memoria
/**
snprintf(report, sizeof(report), "w1[0]: %6d   w1[1]:%6d",
    w1[0],w1[1]);

  Serial.println(report);
//Serial.println(w1[1]);
**/

//el siguiente paso es calcular el producto punto
float   producto_punto =   ((B)*(w1[0])) + ((C)*(w1[1])) ;

//con las siguientes condiciones acotamos el valor final devuelto por la
red
//los limites en las acotaciones dependen de la velocidad inicial de los
motores

if(producto_punto>100){
  producto_punto =100;
}
if (producto_punto < -100){
  producto_punto =-100;
}
Serial.println(producto_punto,DEC);

//actualizacion de los pesos
w[0]=w1[0];
w[1]=w1[1];

//Con las siguientes instrucciones realizamos la acción correspondiente en
cada motor
  vel = 100 + (producto_punto);
  motores.setLeftSpeed((int) vel );

  vel = 100 - (producto_punto);
  motores.setRightSpeed ((int) vel);


}
```

Anexo C. Código Fuente Arduino para cuatro Funciones de activación.

```
#include <Wire.h>
#include <LSM6.h>
#include <Zumo32U4.h>

Zumo32U4Motors motores;

//con este arreglo capturamos las 5 primeras lecturas del sensor
float punto_control_arreglo[5];

//en esta variable guardamos el promedio de las ultimas 5 lecturas
float punto_control;

//umbrales para medir el error
int lambda_2, lambda_1, lambda0, lambda1,lambda2, lambda3;

int w[]= {0,0,0,0};
int w1[]={0,0,0,0};

//variables para las B-Spline, es un vector con 4 componentes
float A,B,C,D;

char report[80];
LSM6 sensor;
int vel;          //velocidad que reciben los motores



void setup()
{
  Serial.begin(9600);
  Wire.begin();

//con la siguiente condición nos aseguramos que el sensor inicia de forma
correcta
  if (!sensor.init())
  {

    Serial.println(¡” Ha fallado la conexión con el sensor!”);

  }
  sensor.enableDefault();



  for(int i=0; i <5; i++){
  sensor.read();            //iniciamos las lecturas del sensor
  float lectura_inicial = (int)sensor.g.z;

   punto_control_arreglo[i]=lectura_inicial;


  }
```

```
//Calculamos el promedio de las primeras 5 lecturas, lo guardamos en
punto_control; que es nuestro setpoint

punto_control=(const int) ((punto_control_arreglo[0] +
punto_control_arreglo[1] + punto_control_arreglo[2] +
punto_control_arreglo[3]+punto_control_arreglo[4]+punto_control_arreglo[5])
/6);


delay (3000);
}

void loop()
{

/**iniciamos las lecturas inerciales **/
  sensor.read();


//Leemos los datos del sensor y los asignamos a la variable lectura_sensor
float lectura_sensor = (int)sensor.g.z;

//El error es la diferencia entre el setpoint(punto de control) y el valor
que se recibe del sensor
float error = ((punto_control)  - (lectura_sensor)) ;



/**En el escalamiento del error se hicieron pruebas con reducción de 100 a
1 y de 10 a 1, al usar estas proporciones en el escalamiento
 * el robot es sensible a los movimientos suaves. El problema se encuentra
en que la red neuronal es inestable, le toma mucho tiempo
 * ajustar el error a 0, debido a estas razones el escalamiento utilizado
es de 1000 a 1
**/

float err = error/1000;           //Escalamiento del error
//Serial.println(err,DEC);

int a,b,c,d,e;                //son los splines lineales


// el error ingresa a las funciones de activación B-Spline
lambda_2 =-30;
lambda_1 =-8;
lambda0 = -2;
lambda1= 2;
lambda2 = 8;
lambda3 = 30;

if ( err >= lambda_2 && err < lambda_1){
  a=1;
}else {
  a=0;
}
if (err >= lambda_1 && err < lambda0){
  b=1;
```

```
} else{
  b=0;
}

if (err >= lambda0 && err < lambda1){
  c=1;
}else{
  c=0;
}
if (err >= lambda1 && err < lambda2){
  d=1;
}else{
  d=0;
}
if (err >= lambda2 && err < lambda3){
  e =1;
} else {
  e=0;
}

//Obtenemos un vector con 4 componentes A,B,C y D
A = ((err - lambda_2)/(lambda_1 - lambda_2))*a + ((lambda0 - err)/(lambda0
- lambda_1))*b;
B = ((err - lambda_1)/(lambda0 - lambda_1))*b + ((lambda1 - err)/(lambda1 -
lambda0))*c;
C = ((err - lambda0)/(lambda1 - lambda0))*c + ((lambda2 -  err)/(lambda2 -
lambda1))*d;
D = ((err - lambda1)/(lambda2 - lambda1))*d + ((lambda3 - err)/(lambda3 -
lambda2))*e;


//calcular la norma del vector que devuelven las B-Spline
float Y= (A*A) + (B*B) + (C*C)+ (D*D) ;

float norma = (sqrt(Y)*1000);   //cálculo de la norma y reescalamiento

//dividir el error entre la norma
float division1= (error) / (norma);

// el siguiente producto recibe la norma, el vector de la b-splines y un
delta
float delta = 0.5;



//producto1 es un vector con 4 componentes
float producto1a= division1*A*delta;
float producto1b= division1*B*delta;
float producto1c= division1*C*delta;
float producto1d= division1*D*delta;


//proceso de aprendizaje de la red neuronal
w1[0]=w[0] + producto1a;
w1[1]=w[1] + producto1b;
w1[2]=w[2] + producto1c;
w1[3]=w[3] + producto1d;
```

```
/**
//Instrucciones para observar el comportamiento de la memoria
snprintf(report, sizeof(report), "w1[0]: %6d   w1[1]:%6d  w1[2]:%6d
w1[3]:%6d",
    w1[0],w1[1],w1[2],w1[3]);

  Serial.println(report);
**/

//Al calcular el producto punto entre dos vectores obtenemos un escalar,
este valor es la salida de la red neuronal
 float producto_punto = ((A)*(w1[0])) +  ((B)*(w1[1])) + ((C)*(w1[2])) +
((D)*(w1[3]));


 //Con las siguientes condiciones acotamos el valor final que devuelve la
red neuronal.
 //Los límites para definir que valores acotar, dependen de la velocidad
inicial de los motores
 if(producto_punto>100){
  producto_punto =100;
}
if (producto_punto < -100){
  producto_punto =-100;
}



//Actualizacion de los pesos
w[0]=w1[0];
w[1]=w1[1];
w[2]=w1[2];
w[3]=w1[3];


//Con las siguientes instrucciones realizamos la accion correspondiente
sobre cada motor
  vel = 100 + (producto_punto);
  motores.setLeftSpeed((int) vel );

  vel = 100 - (producto_punto);
  motores.setRightSpeed ((int) vel); }
```

Anexo D. Código Matlab para dos funciones de activación.

```
%Prueba splines: proyecto Zumo32U4
%Universidad de Quintana Roo
%Misael Baeza

clear all, clc;

%Definimos los umbrales del arreglo y calculamos su longitud
x  = [-30:1:30];     longitud = length(x);

for i = 1:longitud


 lambda_1  = -30;
 lambda0  = -4;
 lambda1  =4;
 lambda2  = 30;

%
  if x(i)>=lambda_1 && x(i)<lambda0
      b(i) = 1;
   else
      b(i) = 0;
  end
%
  if x(i)>=lambda0 && x(i)<lambda1
      c(i) = 1;
   else
      c(i) = 0;
  end
%
   if x(i)>=lambda1 && x(i)<lambda2
      d(i) = 1;
   else
      d(i) = 0;
   end

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %Vector resultante de las funciones de activación, es un vector de dos
  %componentes

 B(i)   =  b(i)+ ((lambda1-x(i))./(lambda1-lambda0)).*c(i);
 C(i)   = ((x(i)-lambda0)./(lambda1-lambda0)).*c(i) + d(i);


end

%graficas de las b-splines
figure(1)
hold on,
plot( x,B, x,C,'linewidth',3);
```

Anexo E. Código Matlab para cuatro funciones de activación.

```
%Prueba splines: proyecto Zumo32U4
%Universidad de Quintana Roo
%Misael Baeza

clear all, clc;

%Definimos los umbrales del arreglo y calculamos su longitud
x  = [-30:0.001:30];     longitud = length(x);

for i = 1:longitud

 lambda_2 = -30;
 lambda_1  = -8;
 lambda0  = -1;
 lambda1  =1    ;
 lambda2  = 8;
 lambda3  = 30;

%
  if x(i)>=lambda_2 && x(i)<lambda_1
       a(i) = 1;
    else
       a(i) = 0;
   end
%
  if x(i)>=lambda_1 && x(i)<lambda0
      b(i) = 1;
   else
      b(i) = 0;
  end
%
  if x(i)>=lambda0 && x(i)<lambda1
      c(i) = 1;
   else
      c(i) = 0;
  end
%
   if x(i)>=lambda1 && x(i)<lambda2
      d(i) = 1;
   else
      d(i) = 0;
   end
%
    if x(i)>=lambda2 && x(i)<lambda3
      e(i) = 1;
   else
      e(i) = 0;
    end



%Vector resultante de las funciones de activación, es un vector de
```

```
%cuatro componentes

A(i)   = ((x(i)-lambda_2)./(lambda_1-lambda_2)).*a(i) + ((lambda0 -
x(i))/(lambda0 - lambda_1))*b(i);

B(i)   = ((x(i)-lambda_1)./(lambda0 - lambda_1)).*b(i) + ((lambda1-
x(i)).(lambda1-lambda0)).*c(i);

C(i)   = ((x(i)-lambda0)./(lambda1-lambda0)).*c(i) + ((lambda2-
x(i)).(lambda2-lambda1)).*d(i);

D(i)   = ((x(i)-lambda1)./(lambda2-lambda1)).*d(i) + ((lambda3-
x(i)).(lambda3-lambda2)).*e(i);

end

%graficas de las b-splines
figure(1)
hold on,
plot(x,A, x,B, x,C, x, D,'linewidth',3);
```