Formando **líderes** para la construcción de un nuevo **país en paz**

Desarrollo De Sistema Embebido Para Aplicar Control Adaptativo Basado En Redes Neuronales A Diferentes Variables Físicas

Trabajo Fin de Grado

Autor:
Nick Brayan Duarte Acevedo
Tutor/es:
Juan Martin Caceres Tarazona

Marzo 2021

Desarrollo De Sistema Embebido Para Aplicar Control Adaptativo Basado En Redes Neuronales A Diferentes Variables Físicas

Autor
Nick Brayan Duarte Acevedo

Tutor/es
Juan Martin Caceres Tarazona
MSc. Controles Industriales

Grado en Ingeniería Mecatrónica

PAMPLONA, Marzo 2021

Preámbulo

Los sistemas embebidos siempre vienen desarrollados para cumplir una prestación o realizar algún tipo de tarea en especifico, incorporando en su núcleo un microcontrolador de gama variable permitiendo ser su centro de operaciones lógicas activando algún puerto de control, en dependencia de la aplicación. debido a ello se busco si hay sistemas embebidos para realizar control a lazo cerrado de algún proceso de una planta sin importar la dinámica o variable a controlar, que aparte de ello incorporara como algoritmo de control el uso de las redes neuronales, y la información encontrada, con estas exigencias la información encontrada iba asociada a control de variable solo en tarjetas de desarrollo mas no en un dispositivo como sistemas embebidos.
Teniendo en cuenta estas consideraciones se ideo realizar un sistema embebido como dispositivo de control de múltiple variable con algoritmos de control neuronal, siendo este el tema que abarcara el trabajo, desde el desarrollo del dispositivo, hasta la aplicación de los algoritmos y validación del control.

Agradecimientos

Agradezco a familiares amigos que estuvieron presentes en mi paso por la universidad apoyándome con la fe y esperanza que podía culminar mis estudios, agradezco al reverendo padre Juan Carlos Garcia, por recibirme en su hogar como uno más de la familia y aportando de su sabiduría en mi día a día. agradezco al profeso Juan Cáceres, por guiar mi proyecto, y las horas extensas de asesorías, que aportaron gran cantidad de conocimiento en mi formación como profesional. Agradezco a mis compañeros de carrera, por su apoyo colaboración y amistad que brindaron en esta etapa, en especial a Fernando y Wilkinfader que son personas de admirar y me nutrían de conocimiento y como persona, Agradezco a cuerpo docente de la carrera por ser excelentes como personas y como profesional.

Dedico este proyecto...
A mi hermano Jonny Duarte y hermana Laura Duarte ...
Que gracias a su guía y apoyo, me permiten ir sumando logros en mi vida...
Siendo ellos mi luz y fuerza de seguir adelante en esta vida llena de retos.

La ciencia no sólo es una disciplina de la razón
sino también del romance y de la pasión

Stephen Hawking

Los espíritus mediocres
Condenan generalmente
todo aquello que no esta a su
alcance.

françois de la rochefoucauld

Índice general

Índice de figuras

Índice de tablas

Índice de Códigos

Resumen

Los sistemas de control convencional han dado grandes prestaciones, en torno a lo que se trata controlar un sistema de algún proceso en específico, siendo robusto en el mayor de sus casos y cumpliendo su cometido, estos controladores convencionales como el PID, tiene que ser sintonizados de manera adecuada en base al modelado matemático de la planta y presentan un buen desempeño en el punto de operación para el cual fue sintonizado, son controladores para sistemas lineales. El desarrollo de las nuevas tecnologías a permitido desarrollar un conjunto de técnicas de Identificación y Control en línea utilizando algoritmos inteligentes, mediante la hibridación de técnicas convencionales de identificación y control con las potencialidades de la computación emergente, específicamente, Redes Neuronales y Lógica Difusa, El control adaptativo es de gran utilidad para la solución de una variedad de problemas de control no lineal donde el control convencional no resulta eficiente, esta característica provee de atractivo a estas técnicas para una gran cantidad de aplicaciones de control en la industria.

En este trabajo se desarrollara un sistema embebido para aplicar control adaptativo basado en redes neuronales a diferentes variables físicas, se seleccionara el microcontrolador con la arquitectura adecuada que permita computar algoritmos complejos y los puertos entrada-salida con una alta resolución, se diseñara y fabricaran las placas del circuito impreso, se programaran e implementaran los algoritmos de control adaptativo basado en redes neuronales en el sistema embebido, se realizara una interfaz que estará incluida en una pantalla que permita interacción para la configuración de parámetros importantes (variable a controlar, selección y configuración de puertos entradas-salida, taxa de aprendizaje, entre otros) y por último se validara el funcionamiento del sistema embebido controlando 3 variables con características diferentes.

1. Introducción

El siguiente trabajo de investigación, aborda varias disciplinas de las que un ingeniero en mecatrónica esta capacitado en trabajar. De las cuales se pueden combinar y llegar obtener resultados innovadores, como teoría de sistemas de control, uso de lenguajes de programación, diseño de circuitos electrónicos.
La aplicación de la teoría de control durante más de medio siglo ha llevado a denominar de formas distintas a las diversas estrategias de control que han ido desarrollándose a lo largo de décadas, atendiendo a los objetivos que se persiguen, la información disponible en cada caso, la metodología empleada en el diseño del sistema de control, etc. Así por ejemplo al control clásico le sucedió años después el denominado control moderno.(Santos, 2011) En donde Todos los sistemas físicos son no lineales en su comportamiento, debido a que ningún sistema de la naturaleza es ideal. Existen regiones donde debe escogerse un área de trabajo específica con el fin de linealizarla y trabajar en ella.(Reyes y Montaña, 2010)
Los sistemas de control convencional como el PID (controlador proporcional, integral y derivativo) es una técnica de control tradicional más utilizada en la industria, donde más de un 90% de los lazos de control utilizan la acción proporcional combinada con la acción integral(Acedo Sanchez, 2003).
Causando desventajas al usar este tipo de control, el sistema está presente bajo perturbaciones o cambio de alguna variable interna de la planta, el controlador no operaria de manera óptima dando respuestas erróneas o con sobre picos a su salida, sin tener la capacidad de aprender que es lo que sucede en la salida para corregir esa respuesta, cambiar parámetros del controlador, como los controles clásicos presentan múltiples dificultades en temas de control automático y auto sintonización, se idea trabajar con controles no convenciones basados en inteligencia artificial en donde actualmente la IA es una ciencia que abarca multitud de áreas, como podría ser en el ámbito automovilístico, usos militares, asistencia sanitaria, la agricultura, el cambio climático, la gestión de riesgo financiero, el consumo actual de los "Smartphone" o incluso en sectores laborales. Por lo que se puede decir que cada día se puede ver que va sufriendo una evolución simbiótica entre las máquinas y los humanos.(Peñafiel y Ing. Ávila, 2020) estos tipos de control demuestran ser robustos en múltiples aplicaciones, la estructura de este tipo de control es simple, siendo esto mismo una falencia de este controlador lo que limita el control y no pueda operar de manera satisfactoria. también existen otro tipos de control, siendo técnicas de control no convencionales o también conocidas como las redes de neuronas artificiales son un paradigma de aprendizaje y procesamiento automático inspirado en el funcionamiento del sistema nervioso central de los animales. Han tenido una gran evolución desde que en 1943 McCulloch y Walter Pitts introdujeran el concepto de neurona artificial, gracias en gran medida a modelos y algoritmos más complejos publicados con posterioridad como el modelo de Hopfield y el algoritmo Backpropagation. (Doctoral y cols., 2015)
las Redes Neuronales Artificiales (ANN) han recibido un interés particular como una tecno-

logía para minería de datos, puesto que ofrece los medios para modelar de manera efectiva y eficiente problemas grandes y complejos, Los modelos de ANN son dirigidos a partir de los datos, es decir, son capaces de encontrar relaciones (patrones) de forma inductiva por medio de los algoritmos de aprendizaje basado en los datos existentes más que requerir la ayuda de un modelador para especificar la forma funcional y sus interacciones. Estas redes son un método de resolver problemas, de forma individual o combinadas con otros métodos, para aquellas tareas de clasificación, identificación, diagnóstico, optimización o predicción en las que el balance datos/conocimiento se inclina hacia los datos y donde, adicionalmente, puede haber la necesidad de aprendizaje en tiempo de ejecución y de cierta tolerancia a fallos. En estos casos las RNAs se adaptan dinámicamente reajustando constantemente los "pesos" de sus interconexiones. (Caicedo y Lopez, 2010) Una contribución importante para acercar la potencialidad de las RNA a la solución de problemas prácticos, es la propuesta en la que se propone un esquema de control neuronal directo y se demuestra que para la mayoría de los casos prácticos, la red se puede entrenar directamente retropropagando el error de control en lugar de utilizar el error de salida de la red, como se hace generalmente.(Noriega y cols., 2008) para poder hacer uso de una arquitectura a base de redes neuronales para este trabajo se debe tener un sistema de procesamiento de datos bastante potente para que pueda ejecutar las instrucciones de manera eficaz, se implementará un micro controlador con estructura ARM están diseñados para ser lo más eficiente posible, aceptando solo instrucciones que se puedan lograr en un único ciclo de memoria. El proceso común para los procesadores es buscar, decodificar y ejecutar instrucciones, y como las unidades RISC son de 32 bits limita la cantidad de instrucciones.(Alonso, 2018)

2. Objetivos

2.1. General

Desarrollo de un Sistema Embebido Para Aplicar Control Adaptativo Basado En Redes Neuronales a diferentes variables físicas.

2.2. Específicos

- Seleccionar los componentes necesarios para implementar un sistema adaptativo embebido.
- Fabricar el hardware del sistema embebido.
- Desarrollar una interfaz para acceder al sistema embebido y configurar parámetros importantes.
- Programar los algoritmos de control adaptativo basado en redes neuronales e implementarlos en el sistema embebido.
- Validar el funcionamiento del sistema embebido controlando 3 variables con características diferentes.
- Redactar la ficha técnica del sistema embebido.

3. Marco Teórico

3.1. Listas de definición

Para una mejor compresión del vocabulario técnico que se usara en el trabajo, se darán algunas definiciones que serán mas usadas.

Variable Controlada Y Señal De Control O Variable Manipulada: La variable controlada es la cantidad o condición que se mide y controla. La señal de control o variable manipulada es la cantidad o condición que el controlador modifica para afectar el valor de la variable controlada. Normalmente, la variable controlada es la salida del sistema. Controlar significa medir el valor de la variable controlada del sistema y aplicar la variable manipulada al sistema para corregir o limitar la desviación del valor medido respecto del valor deseado. (Cruz Avilés y cols., 2018)

Planta: se designará como planta a cualquier objeto físico que pueda ser controlado. Puede ser un equipo, quizás simplemente un juego de piezas de una máquina funcionando juntas, cuyo objetivo es realizar una operación determinada. Ejemplos de plantas son: horno de calentamiento, reactor químico, etc(Perez y cols., 2007)

Proceso: se definirá como una operación o conjuntos de pasos con una secuencia determinada, que producen una serie de cambios graduales que llevan de un estado a otro, y que tienden a un determinado resultado final. Se denominará proceso a cualquier operación que se vaya a controlar. Ejemplos de procesos son: químicos, económicos, biológicos, etc.(Perez y cols., 2007)

Sistemas Un sistema es una combinación de componentes que actúan juntos y realizan un objetivo determinado. Un sistema no está necesariamente limitado a los sistemas físicos. El concepto de sistema se puede aplicar a fenómenos abstractos y dinámicos, como los que se encuentran en la economía. Por tanto, la palabra sistema debe interpretarse en un sentido amplio que comprenda sistemas físicos, biológicos, económicos y similares.(Cruz Avilés y cols., 2018)

Palabra (informática) una palabra es una cadena finita de bits que son manejados como un conjunto por la máquina. El tamaño o longitud de una palabra hace referencia al número de bits contenidos en ella, y es un aspecto muy importante al momento de diseñar una arquitectura de ordenadores.

3.2. Tipos de sistemas de control

Los sistemas de control se se pueden encontrar en dos clasificaciones, los sitemas de control a lazo abierto y los sitemas de control a lazo cerrado.

3.2.1. Sistema de control de lazo abierto

Los sistemas de control de lazo abierto u Open Loop son aquellos cuya señal de salida no influye en la señal de entrada. Es decir, la señal de salida no afecta al proceso.(Queralt, 2004)

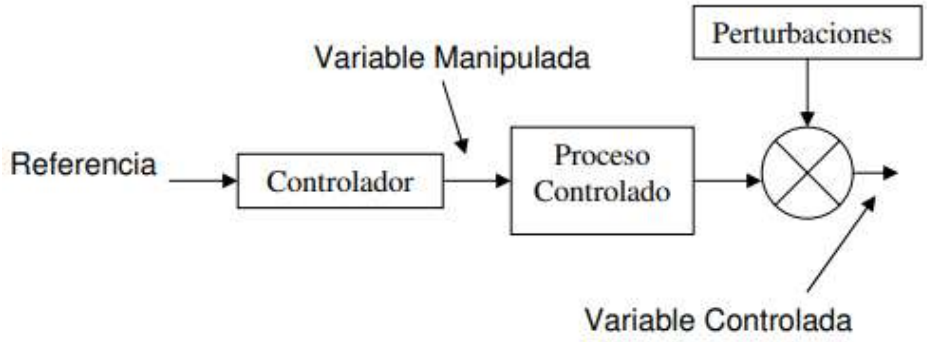


Figura 3.1: Sistema de control en lazo abierto
Fuente:(Asesorado, 2007)

3.2.2. Sistemas de control de lazo cerrado

Los sistemas de control de lazo cerrado o Closed Loop son aquellos en los que la señal de salida influye en la señal de entrada. ver figura 3.2

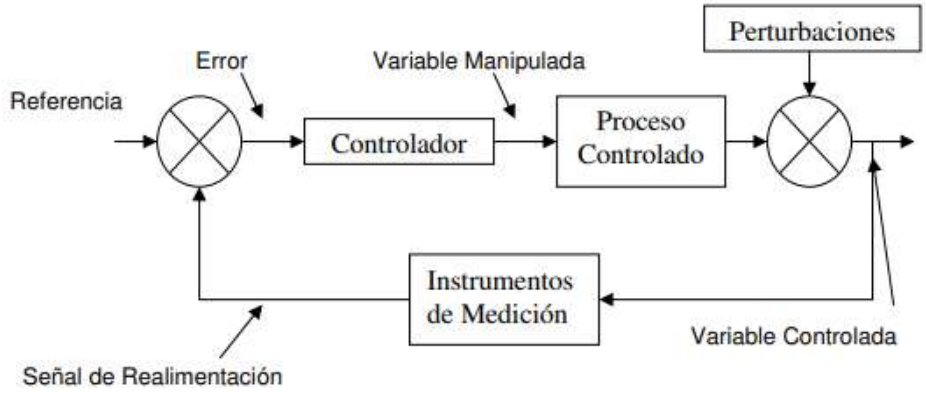


Figura 3.2: Esquema del control por retroalimentación o Feedback
Fuente:(Asesorado, 2007)

- Control por retroalimentación o Feedback: Este tipo de control mide la diferencia entre la variable controlada y el punto de consigna con el objetivo de minimizarla.(Queralt, 2004)

3.3. Técnicas De Control Convencional

3.3.1. Teoría De Control Clásico

Las teorías de control que se utilizan habitualmente son la teoría de control clásica (también denominada teoría de control convencional), la teoría de control moderno y la teoría de control robusto El control automático ha desempeñado un papel vital en el avance de la ingeniería y la ciencia. El control automático se ha convertido en una parte importante e integral en los sistemas de vehículos espaciales, en los sistemas robóticos, en los procesos modernos de fabricación y en cualquier operación industrial que requiera el control de temperatura, presión, humedad, flujo, etc. Es deseable que la mayoría de los ingenieros y científicos estén familiarizados con la teoría y la práctica del control automático.Cruz Avilés y cols. (2018) entre los controladores clásicos podemos encontrar:

- Control PID
- Compensador de adelanto.
- Compensador de retardo.
- Compensador de retardo-adelanto.

El sistema embebido con redes neuronales se va a comparar con el controlador convencional clásico PID, dando una definición del controlador PID clasico.

3.3.1.1. Control PID

Un controlador o regulador PID es un dispositivo que permite controlar un sistema en lazo cerrado para que alcance el estado de salida deseado. El controlador PID está compuesto de tres elementos que proporcionan una acción Proporcional, Integral y Derivativa. Estas tres acciones son las que dan nombre al controlador PID. ver figura 3.3 Pardo (2020)

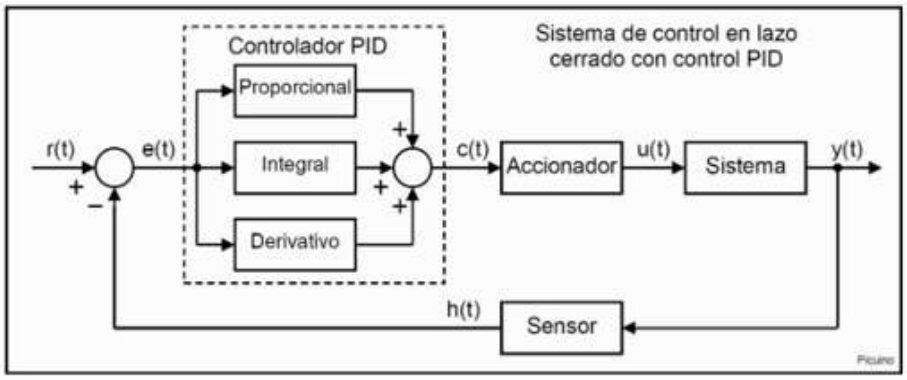


Figura 3.3: lazo de control PID
Fuente: (Pardo, 2020)

Los miembros de la familia de controladores PID, incluyen tres acciones: proporcional (P), integral (I) y derivativa (D). Estos controladores son los denominados P, I, PID.(Mazzone, 2002)

3.3.1.2. Control PID en tiempo continuo

El denomina control PID en tiempo continuo, debido al hecho que el tipo de controlador que se utiliza en este momento es del tipo de señales analógicas, osea el controlador o sistema no deja de percibir información sobre el proceso en ningun momento del mismo(Asesorado, 2007) obersando su comportamiento en la figura

- P: acción de control proporcional, da una salida del controlador que es proporcional al error, es decir: (Mazzone, 2002)

$$P = K_p e\left(t\right) \rightarrow C_p\left(s\right) = K_p \tag{3.1}$$

- I: acción de control integral: da una salida del controlador que es proporcional al error acumulado, lo que implica que es un modo de controlar lento.Mazzone (2002)

$$I = K_i \int_0^t e\left(t\right) dt \rightarrow C_i\left(s\right) = \frac{K_i}{S} \tag{3.2}$$

- D: un control con acción derivativa se hace una corrección que es proporcional a la derivada del error respecto al tiempo. (Asesorado, 2007)

$$D = K_d * e\frac{d\left(t\right)}{dt} \rightarrow C_d\left(s\right) = K_d S \tag{3.3}$$

PID: acción de control proporcional-integral-derivativa, esta acción combinada reúne las ventajas de cada una de las tres acciones de control individuales. La ecuación de un controlador con esta acción combinada se obtiene mediante:(Mazzone, 2002)

$$u(t) = k_p e(t) + \frac{K_p}{T_i} \int_0^t e(\tau) d\tau + k_p T_d \frac{de(t)}{dt} \tag{3.4}$$

y su función transferencia resulta:

$$C_{PID}(s) = k_p \left(1 + \frac{1}{\tau_i s} + T_d s\right) \rightarrow C_{PID}(s) = k_p + k_d s + \frac{k_i}{s} \tag{3.5}$$

3.3.1.3. Control PID discreto

el diseño de sistema de control en tiempo discreto es similar al principio al diseño de sistemas de control en tiempo continuo. EL objetivo del diseño es básicamente, determinar el controlador para que el sistema tenga un desempeño de acuerdo a las especificaciones. De hecho, en la mayoría de situaciones, el proceso controlado es el mismo excepto que en sistemas en tiempo discreto el controlador esta diseñado para procesar datos digitales que en sistemas en tiempo discreto el controlador está diseñado para procesar datos digitales o muestreados.(BASURTO, 1991)

La figura 3.4 indica un clásico esquema de control a lazo cerrado con compensación digital. D esquematiza el controlador digital, la llave L al convertidor análogo-digital (A/D)

y H al convertidor digital-analógico (D/A) o reconstructor de señal. El controlador digital es básicamente un procesador digital que cada T segundos resuelve un algoritmo recursivo (ecuación de diferencias). El diseño del controlador digital suele ser realizado empleando técnicas propias de los sistemas muestreados. Potencialmente, estas técnicas permiten obtener controles más versátiles que los que se pueden conseguir con compensadores analógicos. Sin embargo, existen aplicaciones donde los controladores analógicos han demostrado trabajar satisfactoriamente, razón por la cual en muchas de estas aplicaciones se prefiere diseñar los controladores digitales directamente como una aproximación de los controladores analógicos. Este es, por ejemplo, el caso del controlador PID cuya implementación digital es solo una aproximación numérica de su ecuación integro-diferencial.(Tacconi y cols., 2005)

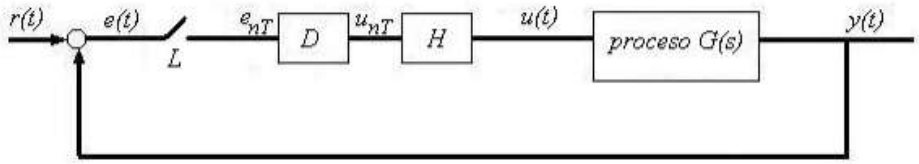


Figura 3.4: Esquema básico de control digital
Fuente: (Tacconi y cols., 2005)

el controlador PID en el dominio del tiempo continuo se describe como en la ecuacion 3.5.
el componente proporcional Kp se implementa en forma digital mediante una ganancia constante Kp ya que una computadora digital o procesador tiene una longitud de palabra finita, la contante Kp no puede realizarse con resolución infinita.(BASURTO, 1991)
Las aproximaciones numéricas mas comúnmente empleadas para discretizar la ecuación diferencial de un sistema analógico son:

1. a) regla rectangular hacia adelante o de Euler

   b) regla rectangular hacia atrás

   c) regla trapezoidal o de Tustin

Estas reglas pueden interpretarse directamente en relación con las acciones integral y derivativa del controlador PID.(Tacconi y cols., 2005)
Por simplicidad se emplea la siguiente notación: e_{nT} =e(nT) , u_{nT}=u(nT) con n entero y T el período de muestreo.

a)"regla rectangular hacia adelante ", según esta regla, la integral 3.2. es decir el área bajo la curva K1.e(t) (figura 3.5, puede ser aproximada por la sumatoria 3.6.Tacconi y cols. (2005)

$$u(nT) \cong K_i T \sum_{i=0}^{n-1} e_{iT} \tag{3.6}$$

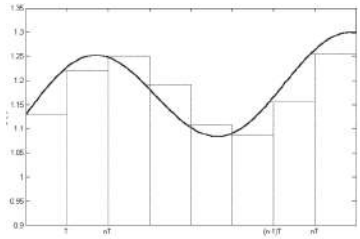


Figura 3.5: Aproximación de la integral por la "regla rectangular a izquierda
Fuente: (Tacconi y cols., 2005)

b) "regla rectangular hacia atrás", según esta regla, la integral 3.2 es aproximada por la ecuación 3.7 (figura 3.6)Tacconi y cols. (2005)

$$u(nT) \cong u_{(n-1)T} + K_i T e_{nT} \tag{3.7}$$

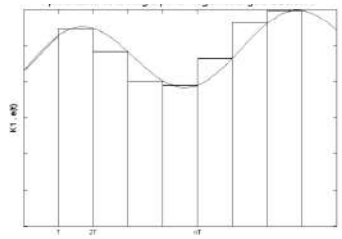


Figura 3.6: aproximación de la integral por "la regla rectangular a derecha
Fuente: (Tacconi y cols., 2005)

c) "regla trapezoidal", según esta regla, la integral 3.2 es aproximada por la ecuación 3.8 (figura 3.7)Tacconi y cols. (2005)

$$u(nT) \cong u_{(n-1)T} + (T/2)\left[e_{nT} + e_{(n-1)T}\right] \tag{3.8}$$

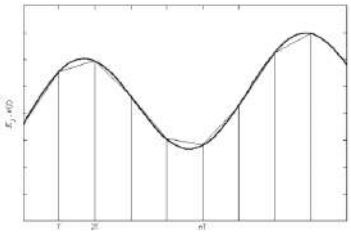


Figura 3.7: Aproximación de la integral por la "regla del trapecio
Fuente: (Tacconi y cols., 2005)

3.3.2. Teoría Control Moderno

En general, los sistemas de control moderno se refieren a la implementación física y lógica de dichos principios fundamentales encontrados. En la actualidad es difícil encontrar un sistema que no contenga algún tipo de sistema de control retroalimentado. Consideremos el caso de un automóvil moderno, el cual tiene entre 200 a 3000 sensores (dependiendo de la marca y el tipo de vehículo). Estos vehículos modernos permiten una conducción más segura y cómoda del vehículo al proporcionar herramientas integradas como ABS, asistencia de frenado, seguimiento de ruta, corrección de errores humanos, entre otros. Dichos avances son posibles gracias a la aplicación de leyes cada vez más precisas de control de la dinámica global del vehículo. Hoy por hoy el 40% del precio de un vehículo corresponde a la electrónica embarcada. Puedes revisar el video que ejemplifica en las ligas de interés.(Mazzone, 2002) los controladores modernos que podemos encontrar:

- Retroalimentación de estados.
- Retroalimentación con estados observados.
- Regulador optimo cuadrático.

3.4. Técnicas De Control No Convencional

El control adaptativo nació en 1950, al diseñar el control automático de un avión donde la dinámica era variable en el rango de operación deseado. Por este motivo, el control convencional no era capaz de abarcar el rango de operación y se llega a una técnica de esquema de ganancias por rangos (Gain scheduling). Una de las aplicaciones más importantes en la actualidad es el control de vuelo de aeronaves no tripuladas, debido a los cambios en la dinámica de vuelo, como se ha mencionado antes.(Embention, 2018)

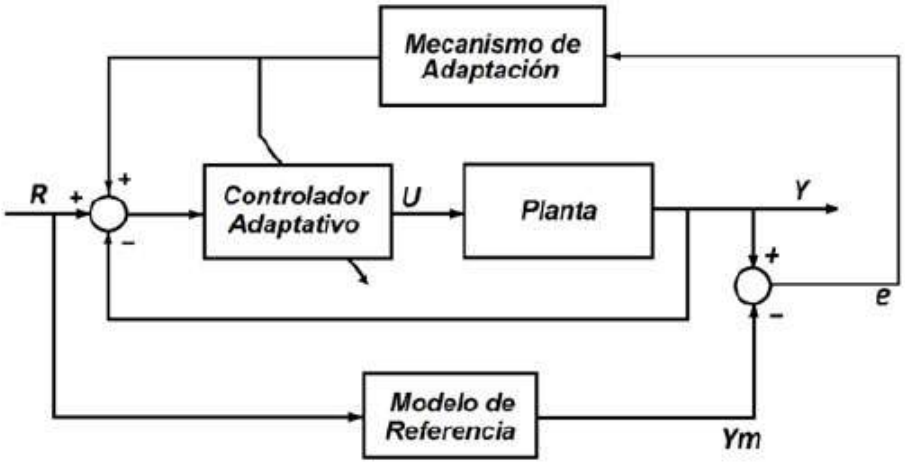


Figura 3.8: lazo de control adaptativo
Fuente: (Esparza y Núñez, 2014)

Específicamente el control adaptativo es un conjunto de técnicas que permiten ajustar en tiempo real el valor de los parámetros de control, permitiendo un buen seguimiento de las variables controladas, aunque se desconozca los parámetros de la planta o cambien en el tiempo. Este control es un tipo especial de control no lineal, y el proceso puede ser separado en dos escalas de tiempo: tiempo rápido (bucle de la realimentación) y tiempo lento (variación de los parámetros de control, el cual afecta a los automatismos).(Embention, 2018)entre los existentes de control no convencional encontramos:

- Control difuso
- Redes neuronales artificiales
- Algoritmos genéticos
- Sistema expertos y planeamiento
- Aprendizaje automático.

Para lo que en este trabajo de grado esta centrado en las redes neuronales artificiales.

3.4.1. Redes Neuronales Artificiales.

Una neurona típica posee el aspecto y las partes que se muestran en la figura 1. Sin embargo, debemos observar que el dibujo no está a escala, el axón alcanza un largo típico de centímetros y a veces de varios metros, las dendritas también y las terminales sinápticas, son más largas, numerosas y tupidas Típicamente, las neuronas son 6 ó 5 órdenes de magnitud más lentas que una compuerta lógica de silicio, los eventos en un chip de silicio toman alrededor de nanosegundos (10 ¡9 s), mientras que en una neurona este tiempo es del orden de los milisegundos (10 ¡3). Sin embargo, el cerebro compensa en forma excepcional la lentitud relativa en el funcionamiento neuronal con un número inmenso de neuronas con interconexiones masivas entre ellas. Se estima que el número de neuronas en el cerebro es del orden de 1010, y que el número de conexiones sinápticas es 6 £ 1013. (Izaurieta y Saavedra, 1999)

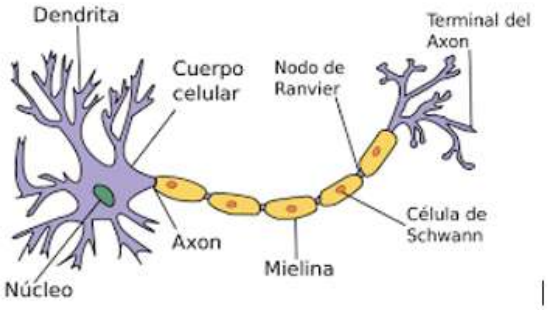


Figura 3.9: neurona y sus partes
Fuente: (Clara Bolonia, 2017)

3.4.1.1. Que Son Las Redes Neuronales Artificiales

Las redes neuronales artificiales (RNA) Es otro tipo de método bioinspirado que trata de emular el funcionamiento de un cerebro humano. Son sistemas que se basan en la interconexión de numerosas unidades básicas de procesamiento, neuronas, con capacidad de aprendizaje. Es un buen ejemplo de procesamiento paralelo. Las neuronas tienen la capacidad de estar activadas o desactivadas, en función de las señales que reciban a la entrada. Cada conexión tendrá una influencia menor o mayor según el peso que tenga asociado. Estos pesos se modifican con el objeto de que la RNA aprenda. Este proceso de aprendizaje es continuado ya que se basa en la información extraída del medio.(Izaurieta y Saavedra, 1999)

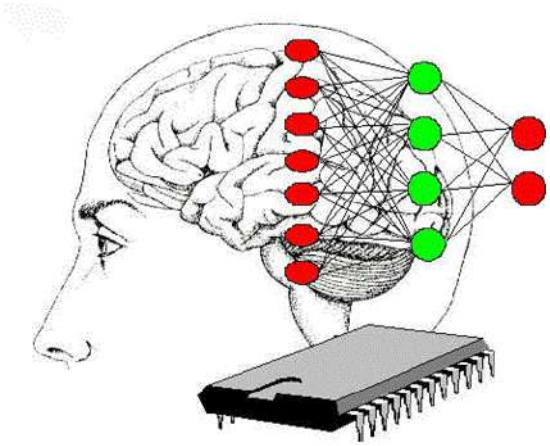


Figura 3.10: analogía red neuronal
Fuente: May (2015)

Las ventajas de las redes neuronales cuando se utilizan para control son:

- Capacidad para aproximar mapeados no lineales mejor que otros esquemas (polinomios, etc).
- Disponibilidad de hardware optimizado para la utilización con redes.
- Capacidad para generalizar con secuencias de entrada no presentadas durante el entrenamiento.
- Capacidad para operar con datos numéricos y simbólicos, simultáneamente.
- Son aplicables a sistemas MIMO.

3.4.1.2. Modelo Neuronal

Una neurona recibe una serie de entradas, representadas por el vector P. En primer lugar, suman estas entradas multiplicada cada una de ellas por un peso diferente. Al resultado se le suma un valor constante denominado polarización (bias) y representado por b, y finalmente se le aplica una función de activación g. Esto queda reflejado en la figura 3.11.(May, 2015)

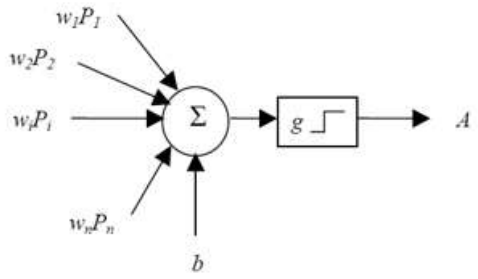


Figura 3.11: modelo neuronal
Fuente: (May, 2015)

Una neurona biológica puede estar activa (excitada) o inactiva (no excitada); es decir, que tiene un "estado de activación". Las neuronas artificiales también tienen diferentes estados de activación; algunas de ellas solamente dos, al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado.(Martínez y cols., 2000) Funciones de activación típicamente usadas puede observarse en la tabla 3.1

3.4.1.3. Redes Neuronales Multicapa

Se denomina arquitectura a la topología, estructura o patrón de conexionado de una red neuronal. En una estructura neuronal los nodos se conectan por medio de sinapsis, esta estructura de conexiones sinápticas determina el comportamiento de la red. Las conexiones sinápticas son direccionales, es decir, la información solamente puede propagarse en un único sentido (desde la neurona presinaptica a la postsinaptica, Figura 3.11). En general, las neuronas se suelen agrupar en unidades estructurales que denominaremos capas. Las neuronas de una capa pueden agruparse, a su vez, formando grupos neuronales (clusters). Dentro de un grupo, o de una capa si no existe este tipo de agrupación6n, las neuronas suelen ser del mismo tipo. Finalmente, el conjunto de una o más capas constituye la red neuronal.(Martín del Brio y Sanz, 2001)

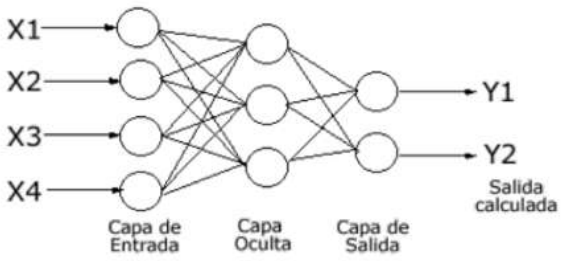


Figura 3.12: Topología redes neuronales multi-capa
Fuente: (M. López, 2017)

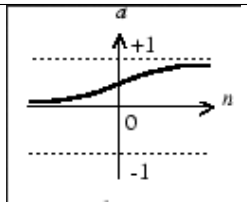
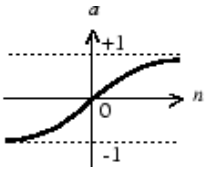
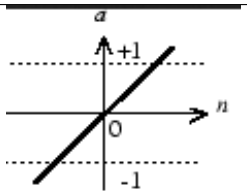
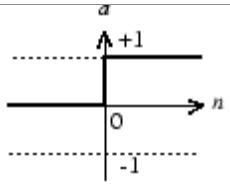
Funciones	**Relación Entrada/salida Modelo matemático**	**Gráfica**
Logsig	$a = \frac{1}{1+e^{-n}}$ (3.9)	
Tansig	$a = \frac{2}{1+e^{-2n}} - 1$ (3.10)	
Purelin	$a = n$ (3.11)	
Hardlim	$a = 0 \rightarrow n < 0$ $a = 1 \rightarrow n \geqslant 0$ (3.12)	

Tabla 3.1: funciones de activación

3.4.1.4. Aprendizaje De Una Red Neuronal

la red procesa los registros en los datos de entrenamiento usando los pesos, bias es y las funciones en las capas ocultas, luego compara las salidas resultantes con las salidas deseadas, los errores se propagan a través del sistema, lo que hace que el sistema ajuste los pesos y biases que serán procesados en la siguiente iteración de entrenamiento. Este proceso ocurre una y otra vez sobre un conjunto de datos a medida que los pesos se ajustan continuamente. Para realizar el proceso, existe un método de optimización muy popular denominado descenso de gradiente (Távara Idrogo, 2019)

3.4.1.5. Descenso En Gradiente

El descenso en gradiente pertenece a un tipo de algoritmos llamados de búsqueda de línea. Estos algoritmos buscan en cada iteración una dirección y actualizan el valor actual de acuerdo a esa dirección. Es decir, en la k-ésima iteración, se tiene un valor θ_k, y se busca una dirección pk para actualizar al valor $\theta_k + 1 = \theta_k + \alpha_k p_k$. donde k > 0 es la 'distancia' que se recorre en la dirección p_k, y es llamada la longitud de paso. Una vez que se actualizó el valor de la iteración, se encuentra una nueva dirección de avance y se actualiza el valor. Esto se hace

hasta que se cumpla cierto criterio de paro, siendo este usualmente que la norma del gradiente sea menor a un escalar pequeño y positivo.En el caso del descenso en gradiente, la dirección de avance p_k es la del máximo descenso, es decir, el negativo del gradiente en la iteración actual $-\nabla L(\theta_k)$, por lo que en cada iteración se hace (Humberto y Contreras, 2017), según la ecuación.

$$\theta_{k+1} = \theta_k + \alpha_k \nabla L\left(\theta_k\right) \tag{3.13}$$

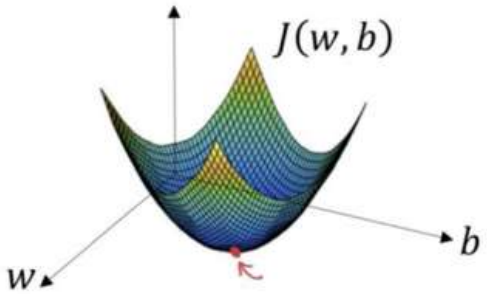


Figura 3.13: descenso del gradiente, ajuste de pesos y bias
Fuente: (Távara Idrogo, 2019)

3.5. Microcontroladores

Los microcontroladores (abreviado μC, UC o MCU) son circuitos integrados que son capaces de ejecutar ordenes que fueron grabadas en su memoria. Su composición esta dada por varios bloques funcionales, los cuales cumplen una tarea específica, son dispositivos que operan uno o más procesos, por lo general los microcontroladores están basados en la arquitectura de Harvard, la cual consiste en dispositivos de almacenamiento separados (memoria de programa y memoria de datos).(Sánchez., 2013) en la figura 3.14 se muestra un esquema general básico de un microcomputador. Se supone de tres bloques fundamentales: la CPU (central processing Unit), la memoria, y la entrada y salida. Los bloques se conectan entre sí mediante grupos de líneas eléctricas denominados buses. Los buses pueden ser direcciones (si transportan direcciones de memoria o de entrada y salida), de datos (si transportan datos o instrucciones) o de control (si transportan señales de control diversas).(Fernando Valdes, 2007)

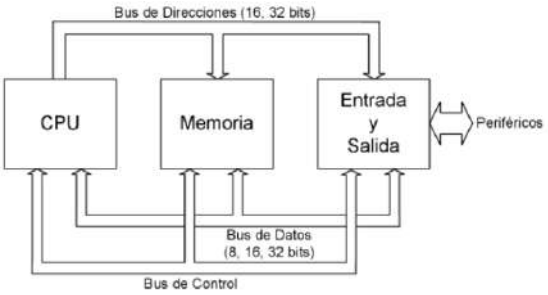


Figura 3.14: esquema básica general de un microcontrolador
Fuente: (Fernando Valdes, 2007)

- **Microcontrolador:** Integrado que incluye un microprocesador, memoria (de programa y datos) y unidades de entrada/salida (puertos paralelo, temporizadores, conversores A/D, puertos serie, etc)(Serrano, 2012)
- **Sistema Embebido (Embedded systems):** Sistema que incorpora microcontroladores (o microprocesadores) para una tarea específica pero que no es "visible" ni "programable" directamente por el usuario (celular, lavarropas, MP3, etc)(Serrano, 2012)

Componentes que conforman un microcontrolador:

- Procesador o CPU (Unidad Central de Proceso).
- Memoria RAM para contener los datos.
- Memoria para el programa tipo ROM/EPROM/EEPROM/Flash.
- Líneas de E/S para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (temporizadores, puertos serie y paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, etc.).
- Generador de impulsos de reloj que sincronizan el funcionamiento de todo el sistema. (Fernando Valdes, 2007)

3.5.1. Tipo de señales

los autómatas recibe y transfiere señales eléctricas, expresando así variables físicas finitas (temperatura, presión etc.). De este modo es necesario incluir en el sistema un convertidor de señal para recibir y cambiar los valores a variables físicas.Perez Molina y cols. (2013) existen 3 tipos de señales en los

Señales binarias , señal de un bit con dos valores posibles ("0" – nivel bajo, falso o "1" – nivel alto, verdadero), que se codifican por medio de un botón o un interruptor. Una activación, normalmente abre el contacto correspondiendo con el valor lógico "1", y una no-activación con el nivel lógico "0". Los límites de tolerancia se definen con interruptores sin contacto. Así el IEC 61131 define el rango de -3 - +5 V para el valor lógico "0", mientras que 11 - 30 V se definen como el valor lógico de "1" (para sensores

sin contacto) a 24 V DC (Fig.12). Además, a los 230 V AC, la IEC 61131 define el rango de 0 – 40 V para el valor lógico de “0”, y 164 – 253 V para el valor lógico “1”. Perez Molina y cols. (2013)

Señales digitales , se trata de una secuencia de señales binarias, consideradas como una sola. Cada posición de la señal digital se denomina un bit. Los formatos típicos de las señales digitales son: tetrad – 4 bits (raramente utilizado), byte – 8 bits, word – 16 bits, double word – 32 bits, double long word – 64 bits (raramente utilizado).Perez Molina y cols. (2013)

Señales analógicas , son aquellas que poseen valores continuos, es decir, consisten en un número infinito de valores (ej. en el rango de 0 – 10 V). Hoy en día, los PLCs no pueden procesar señales analógicas reales. De este modo, estas señales deben ser convertidas en señales digitales y vice-versa. Esta conversión se realiza por medio de sistemas analógicos, que contienen ADC. La elevada resolución y precisión de la señal analógica puede conseguirse utilizando más bits en la señal digital. Por ejemplo, una señal analógica típica de 0 – 10 V puede ser con precisión (pasos para la conversión en una señal digital) desde 0.1 V, 0.01 V o 0.001 V de acuerdo al número de bits que vaya a tener la señal digital.Perez Molina y cols. (2013)

4. Metodología

4.1. Selección de Componentes

Un sistema embebido, también conocido como sistema embarcado o empotrado (integrado, incrustado) es un sistema de computación creado para realizar una algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real. Al contrario de lo que ocurre con los ordenadores de propósito general (como ejemplo un computador personal) que esta proyectado para cumplir un amplio rango de necesidades, los sistemas embebidos.(Carolina Roman Bueno y Johanna Gonzalez Mantilla, 2017).

Los sistemas embebidos están conformados por diversidad de componentes electrónicos, teniendo funciones de acondicionamiento de circuito del microcontrolador, sistemas de protección para entradas de voltaje o contra corrientes, amplificador de señales y dispositivos de acondicionamiento de voltajes de funcionamiento de los dispositivos,esto siempre va de la mano con las exigencias de las condiciones establecidas por el fabricante y dadas en el datasheet del dispositivo. una de los componentes mas importantes para implementar un sistema embebido va asociado al escoger el dispositivo mas eficiente que sera el corazón y cerebro para la ejecución de los algoritmos, eligiendo así el microcontrolador con mejor características y prestaciones que se encuentre y de un acceso fácil al momento de la adquisición entre otras características, siendo un sistema embebido que trabaja con voltajes según el estándar 0-10V en entradas y salidas se deben emplear los circuitos respectivos.

Figura 4.1: Sistemas embebidos
Fuente:(Facultad de Tecnología y Ciencias Aplicadas - UNCA, 2016)

4.1.1. Microcontrolador

En la aplicación de sistemas embebidos, existe un componente fundamental para su operación, gestionando de las tareas, cálculos matemáticos, instrucciones de control, adquisición en la toma de datos o señales de entrada, y control de salida en elementos como efectores finales, el microcontrolador es un dispositivo que a través de instrucciones establecidas por una lógica de programación puede brindar prestaciones muy amplias en muchos campos, como robótica,control y análisis de señales, para la selección de un microcontrolador va asociada a la tarea en la que se va implementar este dispositivo, por lo que se debe tener en cuenta las principales elementos que conforman un microcontrolador

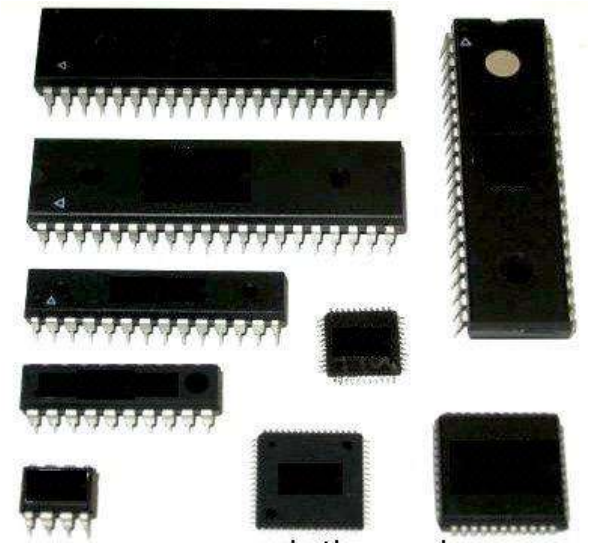


Figura 4.2: Microcontroladores
Fuente:(Gómez, 2017)

4.1.1.1. Elementos de un microcontrolador

Capacidad de procesamiento: El tamaño de la palabra de un procesador es la longitud de la base del número binario que manipula el procesador. Normalmente, los tamaños de palabra son potencias de 2 y la mayoría de los microcontroladores tienen un 8 bits, 16 bits o tamaño de palabra de 32 bits. tamaño de la palabra es un importante factor de rendimiento, ya que afecta a la cantidad de datos que el microcontrolador puede manipular durante un ciclo de instrucción individual. También afecta a la gama de números que se pueden manejar. Un tamaño más grande palabra no es necesariamente mejor para el rendimiento. Por ejemplo, si un microcontrolador sólo se manipulando pequeños números que pueden ser representados en 8 bits o menos, y luego tener un microcontrolador de 32 bits puede ser un desperdicio de recursos y puede no ser el dispositivo de mejor rendimiento en una aplicación particular. Puede ser mejor tener un microcontrolador de 8 bits que puede ser ajustado a una velocidad más alta que la de 32 bits. (Seabrookewindows, 2020)

Memoria: Entendemos por memoria los diferentes componentes del microcontrolador que se

emplean para almacenar información durante un periodo determinado de tiempo. La información que necesitaremos durante la ejecución del programa será, por un lado, el propio código, y por otro, los diferentes datos que usemos durante la ejecución del mismo. Hablaremos por tanto de memoria de programa y de memoria de datos, respectivamente. La diferente naturaleza de la información que hay que almacenar hace necesario el uso de diferentes tipos memorias. Sin hacer especial énfasis en este apartado, sí habrá que tener en cuenta una clasificación básica, que distingue entre memoria volátil y no volátil(Trecedb, 2009)

Entradas y Salidas: los periféricos de un microcontrolador van asociados a los componentes internos que lo componen,estos componentes generan algún tipo de señal salida de control o codificación de señales de entradas, comunicación como los que se van a mencionar:

- conversor analógicos digitales (ADC).
- Conversor digital analógico (DAC).
- Salidas de modulación por ancho de pulso (PWM).
- Entradas y salidas Digital. (I/O)
- Timers
- comunicación por USB.
- protocolo de comunicación serial I2C.
- estándar de comunicaciones SPI.

oscilador: Todo microprocesador o microcontrolador requiere de un circuito que le indique a que velocidad debe trabajar. Este circuito es conocido como un oscilador de frecuencia.(Jonathan López, 2015) la elección de un oscilador se asocia mas a la tarea que debe emplear el microcontrolador en la ejecución de tareas o velocidades de compilación del código,lo mas importante asegurándose de las restricciones dada por el fabricante, encontrada en la hoja de datos.

Encapsulado: al momento de desarrollar un placa de circuito impreso, es importante la elección del encapsulado a usar, esto puede ir muchas veces en criterios del diseñador, o restricciones por el mismo elemento a usar, permitiendo así facilidad a la hora de diseñar o ahorra en el mayor de los casos ahorrar espacio, los dos tipos de encapsulados mas comunes son:

- DIP (Dual in-line package): es una forma de encapsulamiento, común en la construcción de circuitos integrados que consiste en un bloque con dos hileras paralelas de pines.(daniel, 2011) ver figura 4.3 a.
- SMD : Un componente tipo SMD (Surface Mounting Device) se suelda de forma directa a la superficie de la PCB a través de los pads. (Surtel, 2019) ver figura 4.3 b.

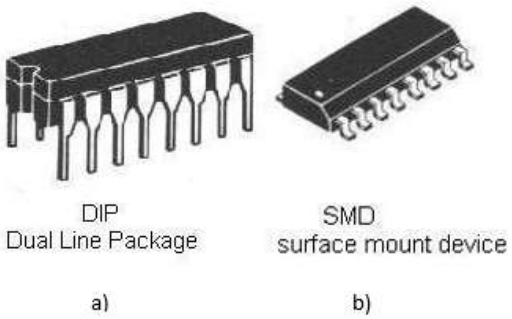


Figura 4.3: Tipo de encapsulado
Fuente:(González García, 2012)

Entorno de Desarrollo: al momento de seleccionar un microcontrolador es importante mirar los entornos de desarrollo con los que permite realizar la ejecución de los algoritmos, algunos entornos permiten la compilación y verificación de errores del código. estos entornos prestan la facilidad de librerías o plataforma de apoyo por otros usuarios, haciendo que al momento de programar ahorremos tiempo y optimización de código. otro punto a tener en cuento es si el entorno de desarrollo es open source, o es un entorno el cual nos exija pagar licencia.

Costos: Los costos de un microcontrolados va asociados siempre a la gama del microcontrolador, rendimiento, las entradas y salidas con las que cuenta, por eso cuando escogemos un microcontrolador, basamos el presupuesto a la necesidad que debemos suplir, y buscar el dispositivo con mejor relación costo-beneficio.

4.1.1.2. Arquitectura Von Neumann La arquitectura tradicional

Neumann La arquitectura tradicional de computadoras y microcontroladores se basa en el esquema propuesto por John Von Neumann, en el cual la unidad central de proceso, o CPU, esta conectada a una memoria única que contiene las instrucciones del programa y los datos.(Sánchez., 2013)

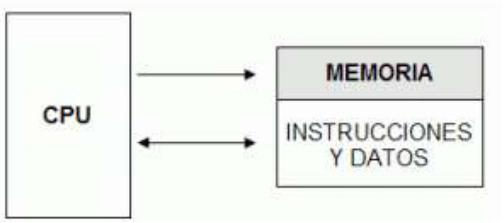


Figura 4.4: arquitectura Neuman
Fuente:(Sánchez., 2013)

Las principales limitaciones que nos encontramos con la arquitectura Von Neumann son:

- La limitación de la longitud de las instrucciones por el bus de datos, que hace que el microprocesador tenga que realizar varios accesos a memoria para buscar instrucciones complejas.
- La limitación de la velocidad de operación a causa del bus único para datos e instrucciones que no deja acceder simultáneamente a unos y otras, lo cual impide superponer ambos tiempos de acceso. (Rosas Oscar, 2017)

4.1.1.3. Arquitectura Harvard

La arquitectura conocida como Harvard, consiste simplemente en un esquema en el que el CPU está conectado a dos memorias por intermedio de dos buses separados. Una de las memorias contiene solamente las instrucciones del programa, y es llamada Memoria de Programa. La otra memoria solo almacena los datos y es llamada Memoria de Datos. Ambos buses son totalmente independientes y pueden ser de distintos anchos. Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instrucción Set Computer), el set de instrucciones y el bus de la memoria de programa pueden diseñarse de manera tal que todas las instrucciones tengan una sola posición de memoria de programa de longitud. (Sánchez., 2013)

Figura 4.5: arquitectura Harvard
Fuente:(Rosas Oscar, 2017)

Ventajas de la arquitecta harvard

- El tamaño de las instrucciones no está relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.
- El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad en cada operación (Rosas Oscar, 2017)

4.2. Circuitos de acondicionamiento

4.2.1. Reguladores de Voltaje

Los sistemas de regulación de voltaje, son necesarios en dispositivos donde se debe limitar el paso de tensión y así evitar algún mal fallo o quema del elemento electrónico, por lo que

es importante tener en cuenta el voltaje a regular, en dependencia al dispositivo a usar y el datasheet ofrecido por el fabricante, sea elementos que operen a una tensión baja y su fuente de alimentación es relativamente alta, se regulan en voltajes como por ejemplo de 5V o 3,3V un esquema de relación de voltaje se puede observar en la figura4.6

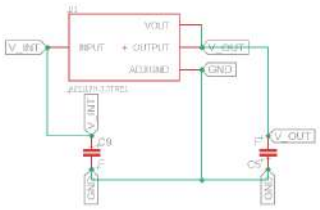


Figura 4.6: esquema de regulación de voltaje
Fuente: Autor

4.2.2. Regulación Voltaje en la Señal Entrada

La captura de datos de señales de entrada el los microcontroladores esta dada en señales de voltaje analógico,en el mayor de sus casos, en rangos dados por el fabricante del microcontrolador. los voltajes mas comunes oscilan entre valores de 0V-10V, 0V-5V, 0V-3.3V. pero cuando la tensión es superior no se puede hacer uso de la señal de voltaje análogo, causando daños en la integridad de algunos puertos de entrada o en todo el dispositivo, la solución mas simple es plantear un esquema de circuito para poder mantener un voltaje de entrada limitado y regulable, la manera mas practica de hacerlo es usando un divisor de tensión, un amplificador operacional en modo seguidor de tensión esto aísla la impedancia de la entrada de ADC, y unos diodos tener de protección, las resistencias a usar se calculan en dependencia del voltaje de entrada que permite el microcontrolador, obsérvese el esquema en la figura 4.7

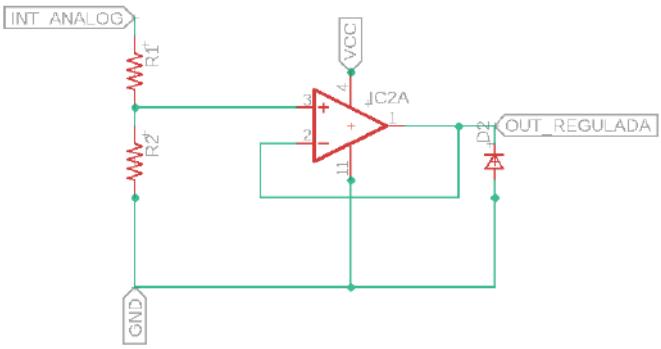


Figura 4.7: esquema de regulación de entrada análoga
Fuente: Autor

como se menciono anteriormente para calcular el voltaje de salida, respecto al voltaje de

entrada se hace uso de un divisor de tensión vista en la ecuación 4.1

$$V_{OUT} = \frac{V_{INT} * R2}{R1 + R2} \tag{4.1}$$

4.2.3. Convertidor de salida Digital analógico DAC

Los microcontroladores por lo general no tienen una salida DAC incorporada en su interior,Mucho de los elementos finales de control encontrados en la industria operan con el estándar de Voltaje 0V-10V se implementa el circuito de acondicionamiento de la señal.
pero dependiendo de la resolución requerida la solución pasa por aprovechar la modulación por ancho de impulso PWM, cuyos generadores si integra(Soriano, 2012)
En una señal PWM la frecuencia base está fijada, pero el ancho del pulso es variable, o sea, que el ciclo de trabajo se puede hacer oscilar entre el 0% y el 100% de acuerdo a la amplitud de la señal original. En la figura siguiente se observa una señal PWM y su transformación como una señal modulada.

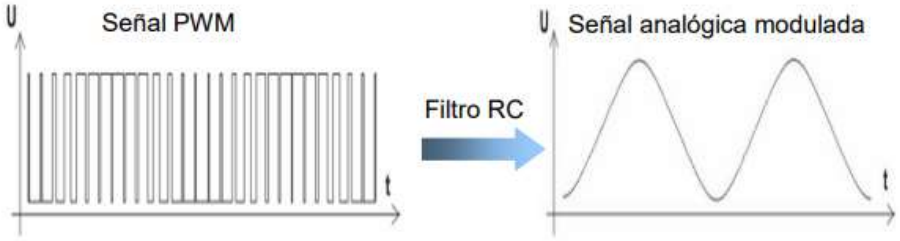


Figura 4.8: Conversión de una señal PWM en analógica
Fuente:(Soriano, 2012)

Un análisis de Fourier de una señal PWM muestra la existencia de un pico principal a la frecuencia Fn = 1/T, mostrando además la existencia de otros picos destacables a F = K/T, donde K es un entero. Los picos con K $\geq$ 2 son armónicos de la componente principal, los cuales son innecesarios y deben ser eliminados. Esto requiere que la señal PWM debe ser filtrada mediante filtros pasa bajos que puedan cancelar este ruido inherente a la propia señal. En la figura 4.9 se muestra el espectro de frecuencia de una señal PWM.(Soriano, 2012)

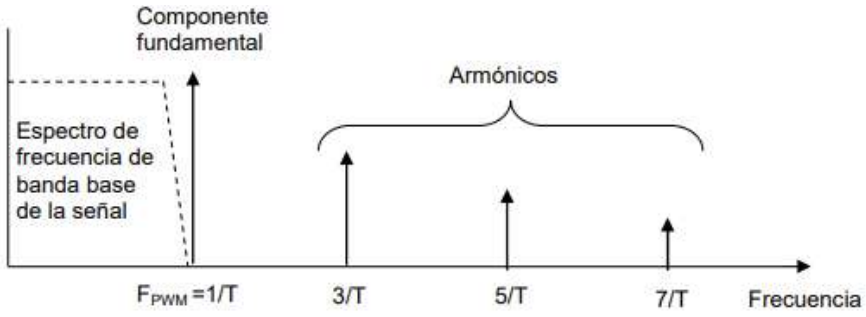


Figura 4.9: Espectro de frecuencia de la señal PWM
Fuente:(Soriano, 2012)

Este espectro muestra que el ancho de banda de la señal buscada podría ser $f_{bw} \leq (F_{PWM} = 1/T)$. Pero esto requeriría el diseño de un filtro complejo y caro pues requiere la respuesta de una función rectangular. En la práctica se escoge F_{PWM} con un valor más reducido. O sea $F_{PWM} = K \cdot F_{PWM}$, donde K » 1. Este valor de K será elegido dependiendo del valor en dB que se quiera rechazar, del propio ruido, de la componente fundamental.Por tanto, para filtrar el ruido con un elevado dB en una señal PWM, se podría optar por filtros pasa bajos complejos de gran pendiente, pero lo más práctico y barato es que el µC module la señal PWM a una frecuencia elevada. Soriano (2012)

Para realizar el filtrado paso bajo con un elevado corte, o sea con una pendiente pronunciada, se implementará un filtro Butterworth de 2º orden (Soriano, 2012), tal y como se muestra en el siguiente esquema.

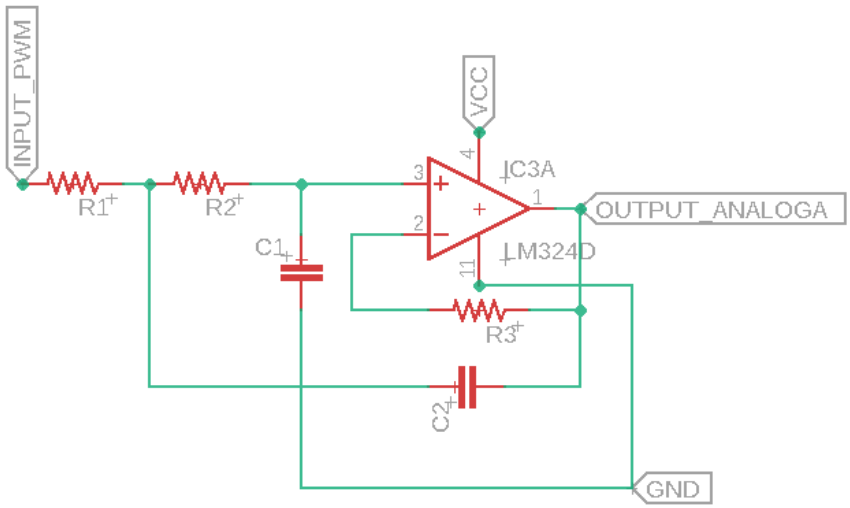


Figura 4.10: Esquema filtro paso bajo para PWM

Fuente: Autor

4.3. Interfaz

las interfaz de usuario es el medio de comunicación eficaz entre el dispositivo y la persona que lo manipule, permitiendo tener un control al momento de la asignación o cambio de parámetros, que a su vez permite visualizar el comportamiento teniendo entradas dinámicas reflejadas en la interfaz.

los diseños de la interfaz se desarrollara acorde a los parámetros mas relevantes que se necesiten alterar al momento de realizar algún tipo de control,sea control clásico o control adaptativo, en estos tipos de controladores existen variables que se deben asignar valores que en dependencia de la planta a controlar, tienen a ser diferentes valores, por ello es importantes una interfaz interactiva acorde a las exigencias del controlador a usar.

los sistemas de interfaz usados en microcontroladores se deben realizar todo los comando movimientos en el menú, todo por código, lo que quiere decir que cada linea o asignación

se debe realizar por el diseñador. lo que se vuelve importante diseñar un modelo que pueda ser implementado de manera sencilla y sea este optimizado, porque el mal desarrollo de una interfaz puede sacrificar espacio en la memoria del microcontrolador o un mal funcionamiento.

Figura 4.11: Interfaz de usuario
Fuente:(COPA-DATA, 2019)

4.4. algoritmo de control discreto

Muchos controladores actuales utilizan microcontroladores digitales.Los reguladores digitales sustituyen varios elementos en un sistema de control tradicional por cálculos en un sistema programado. En la figura 4.12 siguiente puede verse un esquema de un regulador controlado por un microcontrolador.(pardo Carlos, 2014)

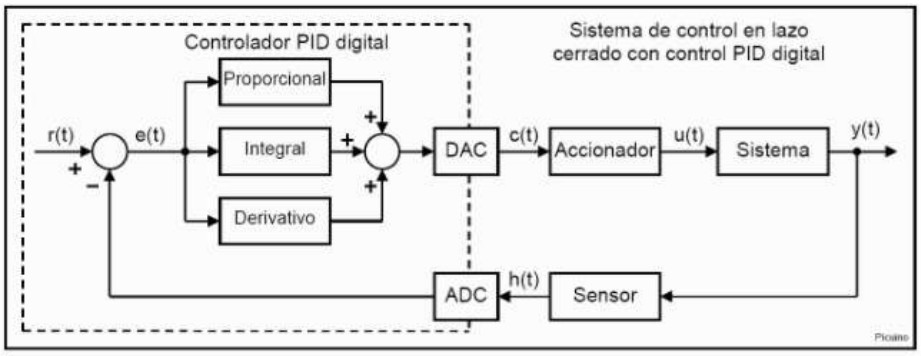


Figura 4.12: Sistema de control en lazo cerrado con control PID DIGITAL
Fuente:(pardo Carlos, 2014)

este controlador es del tipo ”parámetros optimizados” y sus tres parámetros se ajustan con métodos ya muy conocidos. la implementación común de este controlador es en un procesador digital, capaz de calcular una ecuación en diferencias en cada intervalos de muestreo o lo que también se llama, ”en tiempo real”. este es justamente el objetivo con que se desarrollaron los controladores discretos, estos controladores con solo tres parametros se pueden ajustar para tener un comportamiento PID,PI,PD,P e I.(Carelli, 2014)

los sistemas de control análogo últimamente no se implementan, dado esto por el uso de microcontroladores o microprocesadores, que suelen trabajar por señales discretas, por dicha

razón se tienden a utilizar mas estos tipos de control, aunque la señales provenientes de sensores o la realimentación, son señales analógicas continuas en el tiempo ver figura 4.13 que puede ser ejemplo de comportamiento de una señal análoga de un sensor de temperatura, nivel entre otras, como bien se ha mencionado los microcontroladores no pueden trabajar con este tipo de señales y a partir de ese motivo, la señal de se discretea, en otras palabras pasar de señal analógica a un equivalente aproximado de señal análoga esto se realiza a través de un tiempo de muestreo, este tiempo va en dependencia en que tanto querernos que lo tome la data de la señal, sea para procesos de cambios rápidos o cambios lentos.

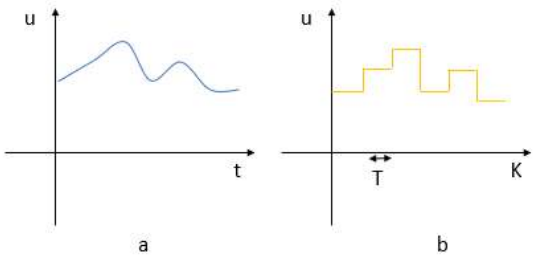


Figura 4.13: figura a) señal analógica, figura b) señal discreta
Fuente: Autor

para hacer el control discreto, se trabajara en base al controlador clásico mas conocido el controlador PID en la ecuación se puede ver la función de transferencia.

$$\frac{u(t))}{e(t)} = \frac{k_p s + k_i + k_d s^2}{s} \tag{4.2}$$

Discretización utilizando la transformada z Los lazos de control continuo, estan formados de tal forma que los componentes del sistema, siempre tienen informacion sobre la variable controlada, la cual es comparada en todo momento con la consigna y en base a esto realizar una acción correctiva o de compensación. Esto no sucede en los sistemas de control discreto, ya que en éstos la información de la variable controlada o el error, entre esta variable y la consigna, solo se obtiene durante el instante de muestreo. La transformada z es utilizada para el análisis y diseño de sistemas discretos, en los que asumimos un periodo de muestreo constante y que sera el mismo para cualquier cantidad de muestre adores en nuestros sistema, además de poseer la misma fase. Asesorado (2007), para hacer la conversión se hace uso de la siguiente ecuación 4.3

$$s = \frac{1 - Z^{-1}}{T} \tag{4.3}$$

ahora Reemplazando s o la ecuación 4.3 en la ecuación 3.5 se obtiene:

$$\frac{U(z)}{E(z)} = \frac{K_p\left(\frac{1-z^{-1}}{T}\right) + K_i + K_d\left(\frac{1-z^{-1}}{T}\right)^2}{\left(\frac{1-z^{-1}}{T}\right)} \tag{4.4}$$

simplificando la anterior ecuación da el siguiente resultado.

$$\frac{U(z)}{E(z)} = \frac{\left(K_p + K_i T + \frac{K_d}{T}\right) + z^{-1}\left(-2\frac{K_d}{T} - K_p\right) + z^{-2}\left(\frac{K_d}{T}\right)}{(1 - z^{-1})} \tag{4.5}$$

esta la versión discreta del controlador,sin embargo de este modo no podemos implementar para realizar algún control, para ello se hace uso de la transformada de la z inversa , pasando de tiempo continuo a tiempo discreto.

$$z^{-1}\left\{\frac{U(z)}{E(z)} = \frac{\left(K_p + K_i T + \frac{K_d}{T}\right) + z^{-1}\left(-2\frac{K_d}{T} - K_p\right) + z^{-2}\left(\frac{K_d}{T}\right)}{(1 - z^{-1})}\right\} \tag{4.6}$$

Resolviendo se obtiene lo siguiente.

$$z^{-1}\left\{U(z)\left(1 - z^{-1}\right) = E(z)\left(\left(K_p + K_i T + \frac{K_d}{T}\right) + z^{-1}\left(-2\frac{K_d}{T} - K_p\right) + z^{-2}\left(\frac{K_d}{T}\right)\right)\right\} \tag{4.7}$$

$$z^{-1}\left\{U(z) - U(z)z^{-1}) = E(z)\left(K_p + K_i + \frac{K_d}{T}\right) + E(z)z^{-1}\left(-2\frac{K_d}{T} - K_p\right) + E(z)z^{-2}\left(\frac{K_d}{T}\right)\right\} \tag{4.8}$$

de este modo ya se puede aplicar la trasformada de la z inversa

$$u\,[k] - u\,[k-1] = e\,[k]\left(K_p + K_i T + \frac{K_d}{T}\right) + e\,[k-1]\left(-2\frac{K_d}{T} - K_p\right) + e\,[k-2]\left(\frac{K_d}{T}\right) \tag{4.9}$$

como salida final para así poder aplicar en el microcontrolador el resultado final es:

$$u\,[k] = e\,[k]\left(K_p + K_i T + \frac{K_d}{T}\right) + e\,[k-1]\left(-2\frac{K_d}{T} - K_p\right) + e\,[k-2]\left(\frac{K_d}{T}\right) + u\,[k-1] \tag{4.10}$$

se simplifica la ecuación haciendo igualdades en los siguientes términos:

$$A_1 = \left(K_p + K_i T + \frac{K_d}{T}\right) \tag{4.11}$$

$$A_2 = \left(-2\frac{K_d}{T} - K_p\right) \tag{4.12}$$

$$A_3 = \left(\frac{K_d}{T}\right) \tag{4.13}$$

Resultado de la salida del controlador:

$$u\,[k] = e\,[k]\,A_1 + e\,[k-1]\,A_2 + e\,[k-2]\,A_3 + u\,[k-1] \tag{4.14}$$

como este controlador trabajar con el error actual y los retardos del error anterior, se puede

decir y hacer la comparativa de la similitud que tiene con la red neuronal adaline. ver figura 4.14

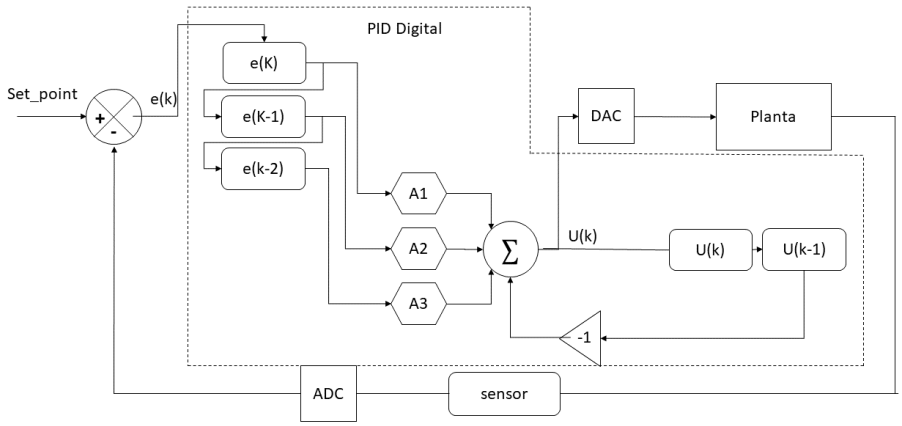


Figura 4.14: Lazo de control PID Discreto
Fuente: Autor

4.5. Algoritmo De Backpropagation (Bp) Para El MLP

Este algoritmo lo publicó Rumelhart en 1986. Aquí se expone el algoritmo básico. En la figura 4.15 es observa un MLP de dos capas ocultas que nos va a servir de ejemplo para la deducción del algoritmo.(Sanchez medina, 1998)

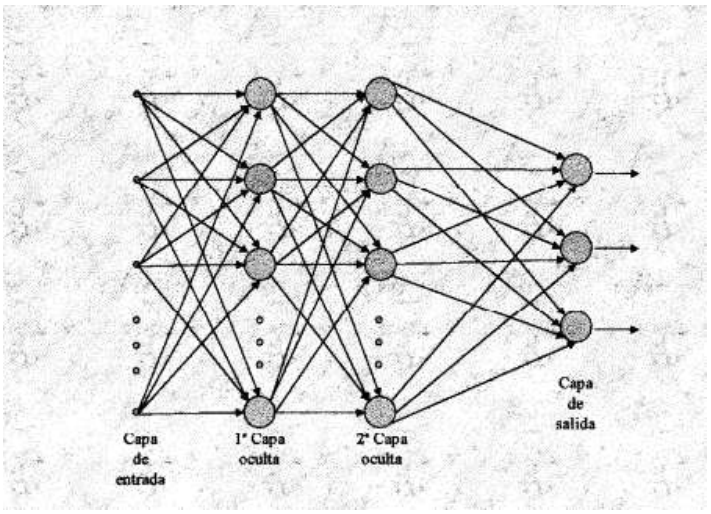


Figura 4.15: Topología de Redes Neuronales Multi-capas
Fuente:(Sanchez medina, 1998)

En cualquier tipo de NN se necesita algún tipo de ponderación o alguna constante que se asocia a cada conexión. Sobre estos elementos propios de cada conexión, mediante su ajuste, es precisamente sobre los cuales se va a efectuar el entrenamiento. En el caso de que estos eiementos antes citados sean de ponderación, es decir que cada señal de entrada a la neurona debe ir multiplicada por ellos(Sanchez medina, 1998), se denominan pesos y la notación que se va a usar es la siguiente:

$$W_{l,i,j}$$

l=Nivel hacia el que va la conexión;
i=nodo destino del nivel l;
j=nodo origen del nivel l-1;

$$y_{l,i}$$

y=es la salida de la neurona.

Como en la Red neuronal adaline, esta red multi-capa tiene un parecido similar en la salida de la neurona, la salida de la red neuronal multi capa, esta dada por la sumatoria de la multiplicación de las entradas por los pesos sinápticos, se dispone de un conjuntos de pares K de entrenamiento que se denota $\{X_k, d_k\}$, donde X es la entrada y d la salida deseada(Sanchez medina, 1998).
Se define la señal de error para el nodo j de la capa de salida y el par de entrenamiento k de la siguiente ecuación.(Sanchez medina, 1998)

$$e_j(k) = d_k(k) - y_j(k) \tag{4.15}$$

se define el valor del error cuadratico de la neurona j como $1/2\left(e_j^2\right)(k)$ Consecuentemente se define el valor instantáneo $\varepsilon(k)$ de la suma de los errores cuadráticos como la suma en todas las neuronas en la capa de salida. La suma instantánea del error cuadrático(Sanchez medina, 1998), esta dada por la ecuación

$$\xi(k) = \frac{1}{2}\sum_{j\xi N_L} e_j^2(k) \tag{4.16}$$

donde el conjunto de N_L incluye a todos las neuronas en la capa de salida de la red. el error cuadrático medio se obtiene sumado $\varepsilon(k)$ para todos los pares de entrenamiento y normalizado, dividiendo por K tal y como se muestra en la ecuación 4.17.(Sanchez medina, 1998)

$$E = \frac{1}{K}\sum_{j\xi N_L} \varepsilon(k) \tag{4.17}$$

la corrección de los pesos sinápticos se realiza mediante un incremento en estos como se indica en la ecuación 4.18.(Sanchez medina, 1998)

$$w'_{L,i,j} = w_{L,i,j} + \Delta w_{L,i,j} \tag{4.18}$$

El algoritmo de backpropagation aplica una corrección $\Delta w_{L,i,j}(k)$ a los pesos sinápticos $\Delta w_{L,i,j}(k)$, la cual es proporcional al gradiente $\partial\xi(k)/\partial w_{L,i,j}(k)$. de acuerdo a la regla de la cadena se puede expresar el gradiente según la ecuación 4.19.(Sanchez medina, 1998)

$$\frac{\partial\xi(k)}{\partial w_{L,i,j}(k)} = \frac{\partial\xi(k)}{\partial e_j(k)}\frac{\partial e_j(k)}{\partial y_j(k)}\frac{\partial y_j(k)}{\partial v_j(k)}\frac{\partial v_j(k)}{\partial w_{L,i,j}(k)} \tag{4.19}$$

el gradiente $\partial\xi(k)/\partial w_{L,i,j}(k)$ representa la dirección de bus quedad en el espacio de los pesos para el peso de la conexión sináptica $w_{L,i,j}$(Sanchez medina, 1998) se deriva en ambos lados la ecuación 4.16, obteniendo:

$$\frac{\partial\xi(k)}{\partial e(k)} = e_j(k) \tag{4.20}$$

derivando en ambos lados la siguiente ecuación se tiene

$$y_i = f(v_i) \tag{4.21}$$

$$\frac{\partial y_i(k)}{\partial v_i(k)} = f_j^{'}(v_j(k)) \tag{4.22}$$

donde la prima significa derivada respecto de su argumento. Finalmente derivando 4.21 a con respecto a $w_{i,j}$, se tiene:(Sanchez medina, 1998)

$$\frac{\partial v_i(k)}{\partial w_{i,j}(k)} = y_i(k) \tag{4.23}$$

se sustituye la ecuación 4.22 a la ecuación 4.23 obteniendo como resultado:

$$\frac{\partial\xi(k)}{\partial w_{i,j}(k)} = -e_i(k)f_j^{'}(v_j(k))y_i(k) \tag{4.24}$$

la corrección $\Delta w_{l,i,j}(k)$ aplicada a $w_{i,j}$ es definida por la regla delta:

$$\Delta w_{l,i,j}(k) = -\eta\frac{\partial\xi(k)}{\partial w_{l,i,j}(k)} \tag{4.25}$$

Donde η es una constante que determina la velocidad del aprendizaje;se denomina parámetro de velocidad de aprendizaje del algoritmo de backpropagation.(Sanchez medina, 1998) dadas las ecuaciones 4.25 y 4.26

$$\Delta w_{l,i,j}(k) = -\eta\delta_j(k)y_i(k) \tag{4.26}$$

el gradiente local $\delta_i(k)$ es definido por:

$$\delta_j(k) = -\frac{\partial\xi(k)}{\partial e_j(k)}\frac{\partial e_j(k)}{\partial y_j(k)}\frac{\partial y_j(k)}{\partial v_j(k)} \tag{4.27}$$

$$\delta_j(k) = -e_j(k)f_j^{'}(v_j(k)) \tag{4.28}$$

El gradiente indica los cambios requeridos en los pesos sinápticos. De la ecuación 4.28 se

deduce que un elemento clave envuelto en el cálculo del ajuste en los pesos $\Delta W_{i,j}(k)$ es la señal de error $e_i(k)$ en la neurona de salida j.En este contexto se pueden identificar dos casos distintos dependiendo de en que parte de la red se encuentre I a neurona j. Está el caso I en el que la neurona j está en la capa de salida. Por otra parte está el caso II en el que la neurona está en cualquier otra capa que no sea la capa de salida.(Sanchez medina, 1998)

Caso I: La neurona j es un nodo de salida. Cuando la neurona j está ubicada en la capa de salida de la red, debe ser contrastada la salida natural con la respuesta deseada en sí misma. De ahí que se use la ecuación 4.26 para calcular la señal de error $e_j(k)$ asociada a la neurona. Con $e_j(k)$ es un problema trivial el cómputo del gradiente local$e_j(k)$ usando la ecuación 4.28 (Sanchez medina, 1998)

Caso II. La neurona j es un nodo oculto. Cuando la neurona j está ubicada en una capa oculta de la red no hay una respuesta deseada especificada para esta neurona. Para calcular la señal de error de esta neurona se determina recursivamente en términos de ¡as señales de error de todas las neuronas a las que está conectada la neurona en cuestión. Aquí es donde el algoritmo de Backpropagation se complica. De acuerdo con la ecuación 4.28 se puede redefinir el gradiente local $\delta_j(k)$ para la neurona oculta j como:Sanchez medina (1998)

$$\delta_j(k) = -\frac{\partial \xi(k)}{\partial y_j(k)} f_j^{'}(v_j(k)) \tag{4.29}$$

Se toma la ecuación 4.16 y se reescribe cambiando el índice j por m. Esto se hace para evitar confusión con el índice j que se usa ahora para la neurona oculta. Sanchez medina (1998)

$$\xi(k) = \frac{1}{2} \sum_{m \xi N_L} e_m^2(k) \tag{4.30}$$

la ecuacion de deriva con respecto a la señal $y_i(k)$ obteniendo:

$$\frac{\partial \xi(k)}{\partial y_i(k)} = \sum_m e_m \frac{\partial e_m(k)}{\partial y_i(k)} \tag{4.31}$$

con esta ecuación se puede hace uso de la regla de la cadena para el calculo de la derivada parcial y se puede reescribir la ecuación de la siguiente manera:

$$\frac{\partial \xi(k)}{\partial y_j(k)} = \sum_m e_m \frac{\partial e_m(k)}{\partial v_m(k)} \frac{\partial v_m(k)}{\partial y_j(k)} \tag{4.32}$$

como la neurona m es una neurona que viene de la capa de salida, se procede con lo siguiente:

$$\frac{\partial e_m(k)}{\partial v_m(k)} = -f_m^{'}(v_m(k)) \tag{4.33}$$

el nivel de actividad de interna de la red para neura k es:

$$v_m(k) = \sum_{j=0}^{q} w_{m,j}(k) y_i(k) \tag{4.34}$$

donde q es el número total de entradas (sin contar el offset) aplicadas a la neurona k. Aquí de nuevo, el peso sináptico $w_{m,0}(k)$ es igual al offset 0,(k) aplicado a la neurona k y la correspondiente entrada t_0 se fija al valor 1. Para cualquier caso, derivando la ecuación 4.34 con respecto a $y_i(k)$ conduce a(Sanchez medina, 1998):

$$\frac{\partial v_m(k)}{\partial y_j(k)} = w_{m,j}(k) \tag{4.35}$$

a partir de esto, se usa la ecuacion 4.33,4.35 y 4.32 da como resultado:

$$\frac{\partial \xi_m(k)}{\partial y_j(k)} = -\sum_{m} \delta_m(k) * w_{m,j}(k) \tag{4.36}$$

donde en la segunda línea, se ha usado la definición del gradiente local δ_m(k) dada en la ecuación 4.28 con el índice m sustituido por j. Finalmente, usando ¡a ecuación 4.36 en 4.26, se obtiene e¡ gradiente local δ_m(k) para una neurona oculta j, tras la reorganización de términos como sigue(Sanchez medina, 1998):

$$\delta_j(k) = f_j^{'}(v_j(k)) \sum_{m} \delta_m(k) w_{m,j}(k) \tag{4.37}$$

siendo la neurona j oculta. El factor $f_v^{'}(v_j(k)$ envuelto en la computación del gradiente local δ_j ecuación 4.37 solamente de la función de activación asociada a la neurona oculta j. el otro actor, el sumatorio depende de dos conjuntos de términos. el primer conjunto de términos, δ_m, requiere del conocimiento de todas las señales de error $e_m(k)$ de todas las neuronas de la capa siguiente a las neurona oculta j, que están directamente conectadas a esta. el segundo conjunto términos $w_m(k)$, consiste en los pesos sinápticos asociados a la conexiones.

4.5.1. Ejemplo de implementación del algoritmo

para el ejemplo se denota que se uso la función de activación sigmoide, por la derivada que aparece en los cálculos da como resultado.

$$f'(x) g.f(x)(1 - f(x)) \tag{4.38}$$

ALGORITMO DE BACKPROPAGATION

Procedimiento BACKPROPAGATION;
  Inicializar los pesos aleatoriamente;
  Repetir
    poner entrada de entrenamiento;
    FEED_FORWARD;
    COMPUTO_GRADIENTE;
    ACTUALIZA_PESOS;
  hasta encontrar condición final;
fin BACKPROPAGATION

Procedimiento FEED_FOWARD ;
  for capa=1 to L do
    for nodo=1 to N_{capa} do
      $y_{capa,nodo} = f(\sum_{i=1}^{N_{capa-1}} W_{capa,nodo,i} \times y_{capa-1,i})$;
    end
  end
end FEED_FOWARD

Procedimiento COMPUTO_GRADIENTE ;
  for capa=1 to L do
    for nodo=1 to N_{capa} do
      if (capa=L) **then**
        $e_{L,nodo} = y_{L,nodo} - d_{nodo;}$
      else
        $e_{capa,nodo} = \sum_{m=1}^{N_{capa-1}} e_{capa+1,m} y_{capa+1,m}(1 - y_{capa+1,m}) \times W_{capa+1,nodo}$
    end
    for todos los pesos en todas las capas **do**
      $g_{capa,j,i} = e_{capa,j} y_{capa,j}(1 - y_{capa,j}) y_{capa-1,i}$;
    end
  end
end COMPUTO_GRADIENTE

Procedimiento ACTUALIZA_PESOS;
  for todos los pesos w_{lji} **do**
    $w_{l,j,i}(k+1) = w_{l,j,i}(k) - \eta * g_{l,j,i}$;
  end
end ACTUALIZA_PESOS;

Tabla 4.1: CODIGO BACKPROPAGATION

Fuente:(Sanchez medina, 1998)

4.6. Selección De Las Plantas Con Diferente Variable Física

haciendo continuidad con lo establecido en este trabajo, poder realizar la implementación del control adaptativo neuronal a diferentes variables físicas, usando el sistema embebido, se necesitan elegir las variables a controlar estableciendo algunos criterios de selección,siendo los siguientes:

- El sistema embebido solo es un sistema de control,este genera una señal de salida dada en el estándar 0-10v o una salida digital de 10V, por lo que la parte de potencia se debe acondicionar a la señal de control.
- la dinámica física de la variable a controlar deben ser diferentes.
- El sistema de control de todas las variables deben ser en lazo cerrado.
- la instrumentación sensorial, se utilizaran dispositivos académicos por la facilidad de su adquisición y costo. de igual manera aplica par ala parte de potencia.

con los criterios anteriores se pueden elegir 3 los tipos de variables físicas que se le realizara un control adaptativo con el sistema embebido, esto con el fin de comprobar que con un mismo controlador se puede realizar control a las variables físicas, por lo que se deben realizar los montajes de las 3 tipos de variable usando instrumentación académica, por su facilidad y costo al querer acceder a ellos en el mercado. las plantas a controlar son las siguientes.

- Planta para el control de Temperatura.
- Planta para el control de Velocidad.
- Planta para el control de nivel de Liquido.

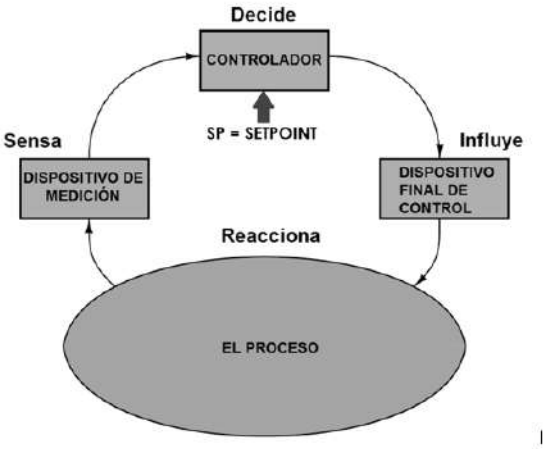


Figura 4.16: Lazo de controlador general
Fuente:(Tony R, 2006)

4.6.1. Planta de Temperatura

La planta de temperatura, se desea controlar la temperatura ambiente de un sistema semicerrado, como los vistos en incubadoras, invernaderos entre otros sistemas simulares. se utilizara un sensor LM35, este proporciona un voltaje de salida análogo en la escala de milivoltio (mV) codificando la señal obteniendo el valor de temperatura, y para la salida del efector se utilizara un bombillo incandescente que por medio de un dimmer se le regulara la potencia, por medio de una etapa de potencia acondicionada a recibir el voltaje variable de 0-10v y duplicada la señal para que se pueda controlar el angulo de disparo en una escala de 0-100% en dependencia del voltaje suministrado por el sistema embebido, su respectivo lazo de control se puede apreciar en la siguiente figura 4.17

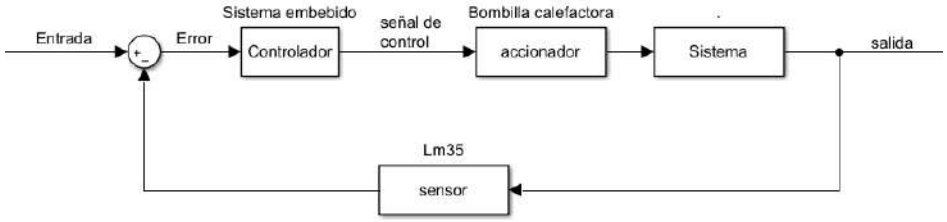


Figura 4.17: Lazo de control para la planta de control Temperatura
Fuente: Autor

4.6.1.1. Instrumentación

4.6.1.1.1. Sensor de Temperatura Lm35 El sensor lm35 es uno de los sensores mas comunes de encontrar, este sensor tiene una calibración de 1°c, por lo general puede abarcar temperaturas en un rango de medición entre -40 y 110°c, el rango de temperatura va limitado por la cantidad de voltaje variable que puede tener a su salida que va desde -400mv hasta los 1100mV. un resumen de la ficha técnica se puede observar en la siguiente tabla.

LM35

voltaje de operación: 4-30 V
Rango de temperatura (°C): -55 hasta 150
corriente de suministro max: 114μA

Tabla 4.2: Sensor Lm35

4.6.1.1.2. Dimmer el dimmer es un dispositivo que permite controlar el voltaje suministrado por la red 110v, y así obtener voltajes intermedios, así de esta forma se puede variar la intensidad de brillo de bombillas incandescentes, la potencia de un calefactor resistivo. las especifica iones del dimmer se puede ver en la siguiente tabla.

Dimmer

voltaje de operación: 110-200 V (AC)
voltaje de entrada digital: 5V
Pines analógicos de entrada: 6
Corriente de salida DC por pin: 40mA
detección: cruce por cero

Tabla 4.3: Dimmer de dos canales

4.6.1.1.3. Arduino uno el uso de este microcontrolador solo sera codificar la señal analógica dada por el sistema embebido y replicada en un porcentaje de 0-100% al bombillo incandescente, porque al incorporar el cruce por cero al sistema embebido, puede sacrificar tiempos de muestreo o retardos en el controlador, por eso se opta en acondicionar la señal, en una etapa de potencia, aclarando que el control se hace solo del sistema embebido y este acondicionamiento, solo actua como etapa de potencia.

Arduino uno

Pines digitales entradas/salidas: 14 (6 con PWM)
Frecuencia de reloj: 16Mhz
Corriente de salida máx: 18A
potencia máx por canal: 1200w
detección: cruce por cero

Tabla 4.4: Arduino uno

4.6.1.1.4. bombilla incandescente el efector final de esta planta y fuente de suministro de energía calorífica esta dada por una bombilla incandescente, que al alimentar con energía de la red la bombilla pasa corriente por un filamento de tungsteno, un material que al calentarse dentro del vidrio emite una luz amarillenta.

Bombilla incandescente

alimentación: 110V (AC)
consumo: 100 W

Tabla 4.5: Bombilla incandescente

4.6.2. Planta de Velocidad

En este tipo de planta se va a querer controlar el valor de Revoluciones Por Minuto(RPM) de un motor eléctrico, el método de obtención de la velocidad estará dado por un encoder giratorio, transformando el movimiento mecánico en señales eléctricas y codificando estos impulsos eléctricos,se puede dar el valor de RPM reales del motor en el sistema embebido, el accionador o etapa de potencia de esta planta, se establece usando un arduino uno y el uso de un mosfet, el arduino uno cumplirá la función de tomar la señal análoga y replicarla pero esta vez en valor PWM, que sera la señal de control para un circuito puente H, que controlara la velocidad del motor establecida principalmente por la señal suministrada por el sistema embebido. su lazo de control se puede apreciar en la figura 4.18

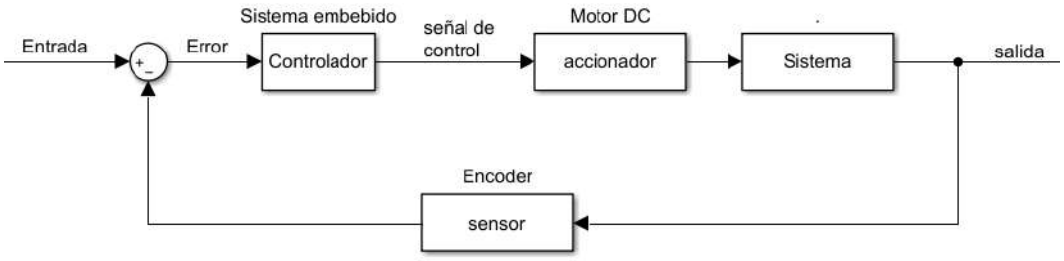


Figura 4.18: Lazo de control para la planta de control Velocidad

Fuente: Autor

4.6.2.1. Instrumentación

4.6.2.1.1. Sensor Encoder los encoder son dispositivos eficaz de medición de PRM de un motor o también usado para control de posición, para esta planta se va a controlar los RPM, con un dispositivo que convierte un movimiento mecánico rotatorio a impulsos o señales eléctricas, con el uso de un microcontrolador estos impulsos son codificados en relación a la resolución por vuelta que pueda brindar el encoder, el dispositivo que se implementara en esta planta cuenta con una resolución de 360 pulsos por vuelta, las características se pueden observar en la tabla 4.6

Encoder LPD3806-360BM-G5-24C

Tipo de sensor: Sensor óptico
voltaje de operación: 5-24 V
corriente de consumo: 40mA
Resolución del encoder: 360 pulsos por vuelta
Frecuencia de respuesta maxima: 20Khz
Tipo de salida: Sensor Digital
Fase de Salida: Fase A y B
Tipo salida: NPN colector abierto
máxima velocidad mecánica: 5000 RPM

Tabla 4.6: Sensor Encoder LPD3806-360BM-G5-24C

4.6.2.1.2. Motor DC El motor de corriente directa sera el elemento final de control, que generara un movimiento mecánico por medio de una voltaje eléctrico, por principio de control de velocidad que se puede dar a un motor, se uso el método de modulación por ancho de pulso (PWM) y asi poder regular los RPM's , el la siguiente tabla 4.7 se podran ver algunas especificaciones del motor eléctrico, se resalta que se tiene pocos datos del motor ya que es reciclado de una impresora láser no se encontró mucha información.

Motor Eléctrico DC

alimentación: 12-24 V(DC)
RPM: ≈ 5000 RPM

Tabla 4.7: Motor Electrico

4.6.2.1.3. etapa de potencia la etapa de potencia para aplicar un control por modulación de ancho de pulso de un motor eléctrico, se realizara por medio un Mosfet IRF740N, permitiendo poder regular la velocidad del motor eléctrico con el PWM codificado por el arduino, en la tabla se ve las especificaciones del circuito a usar.

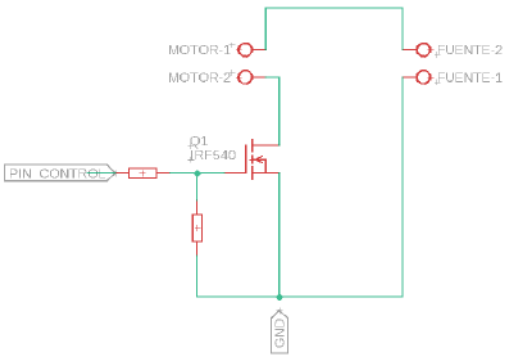
Control velocidad de motor

Tensión disruptiva : 100V (DC)
corriente drenaje: 30A
Disipación de Calor: 85W

Tabla 4.8: Circuito de control Mosfet

4.6.3. Planta de Nivel de liquido

la planta de control de nivel de un liquido, controlara el nivel de un fluido este caso agua, en cm, esta medición se realizara por medio de un sensor de distancia sharp, permitiendo saber cual es el nivel real, el control de llenado del recipiente se utilizara una moto-bomba de 12v, que suministrara el liquido para hacer subir el nivel, se generara una salida incorporando una llave regulable a la salida y poder generar una perturbación al sistema, el lazo de control se puede apreciar en la figura 4.19

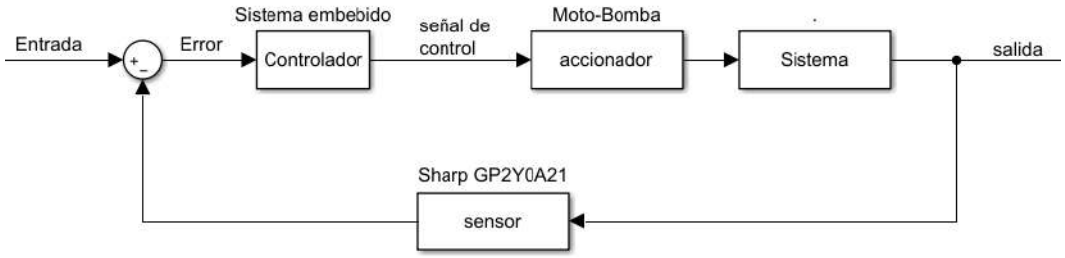


Figura 4.19: Lazo de control para la planta de control Nivel
Fuente: Autor

4.6.3.1. Instrumentación

4.6.3.1.1. Sensor Sharp El sensor Sharp es un sensor de naturaleza óptica, este mide la distancia entre el objeto y el sensor, para poder realizar la medición el sensor incorpora un emisor infrarrojo y un receptor que miden la distancia mediante triangulación, el comportamiento del sensor en relación al a la distancia del objeto se vera en la figura

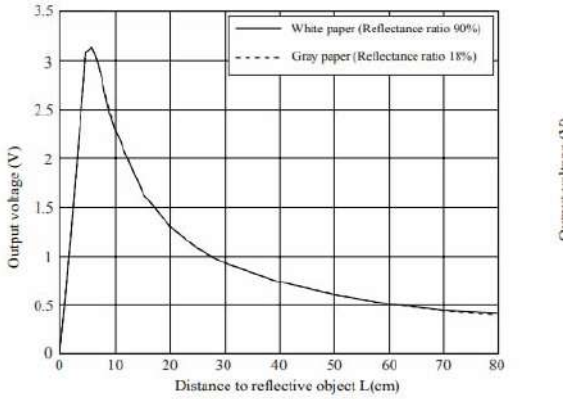


Figura 4.20: caracterización del sensor
Fuente:datasheet del sensor

El modelalado del comportamiento de la gráfica que se usara en el sistema embebido puede observarse en la ecuación 4.39, las características basicas del sensor se pueden ver en la tabla 4.9

$$x = \left(\frac{3027.4}{lectura}\right) e^{1.2134} \tag{4.39}$$

Sensor sharp

Voltaje de Funcionamiento : 4.5 -5.5 V
Salida: Voltaje Analógico (1v-3.3V)
distancia de medición: 10 cm- 80 cm
corriente Nominal: 30mA

Tabla 4.9: SHARP GP2Y0A21

4.6.3.1.2. Bomba Sumergible la moto-bomba actuara como el efector final de esta planta, cumpliendo la labor de llenado del recipiente regulando el flujo del liquido de entrada, en relación al dutty del PWM, la características de la moto bomba se observaran en la tabla 4.7

Bomba sumergible

Voltaje de Funcionamiento : 6-12V
corriente Nominal: 0.35A
Caudal máximo: 280 L/H
RPM: 3500

Tabla 4.10: Bomba Sumergible JT-160

4.6.3.1.3. Puente H El puente H, se usara como la etapa de potencia,y como se explico anteriormente en el control de temperatura por la salida análoga generada por el sistema embebido, no puede controlar directamente, debe ser codificada y replicada al accionador se usa un arduino que solo duplique la salida análoga del sistema embebido y genere una salida PWM, el circuito puente H a usar se puede apreciar en la tabla 4.11

Puente H

Voltaje de Funcionamiento :12V
corriente máx: 3A
señal de entrada: PWM

Tabla 4.11: Puente H

5. Desarrollo

5.1. Selección de Componentes

El seleccionar los componentes indicados en tarjetas electrónicas de desarrollo es de vital importancia, a partir de una adecuada selección podemos garantiza el funcionamiento requerido por algunos componentes. que dependen de voltajes de funcionamiento, o de otros elementos externos que acondiciónenlo simplemente para que el dispositivo comience a operar.
según la metodología se enunciaron 4 circuitos principales de los que consta el sistema embebido de los cuales son:

- **microcontrolador:** la selección del microcontrolador y componentes periféricos para su funcionamiento.
- **Regulador de voltaje:** Importancia de colocar reguladores de voltaje cuando se opera con dispositivos que opera con tensiones bajas
- **Regulación Voltaje en la Señal Entrada** la regulación de voltaje de entrada, mas que un reductor de voltaje, también es una protección de sobre voltajes y permite tener impedancias de entrada altas.
- **Convertidor de salida Digital analógico DAC** como la gran mayoría de microcontroladores solo tiene salida digital, se le acondicionar un circuito para así poder obtener una salida analógica real y ser implementada en mas dispositivos

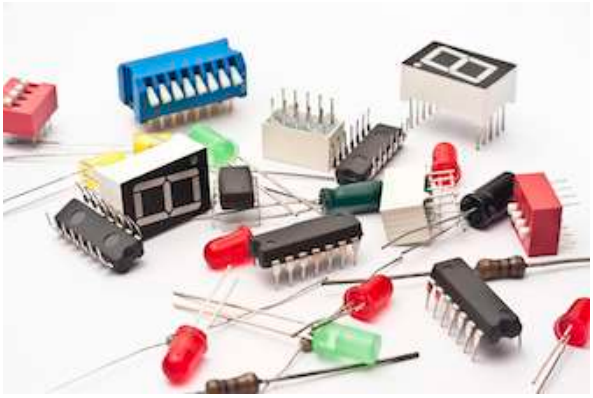


Figura 5.1: componentes electrónicos
Fuente: legaelectronica.com.ar

5.1.1. Microcontrolador

En este apartado se realizo la selección del microcontrolador, se buscaron las características de acuerdo a la metodología establecida, y se comparan parámetros de funcionamiento entre los microcontroladores más comunes y comerciales. para así proceder al diseño electrónico y selección demás componentes que conforman el dispositivo, los datos son tomados directamente de la hoja de datos que ofrece el fabricante de cada dispositivo. la comparación se hará con el microcontrolador ATMEGA-328, aATMEGA-2560, pic-16f84a, PIC18f2550, y el ATSAMD21G18-AU, MK66FX1M0VMD18, algunos de estos microcontroladores ha sido común encontrar en algunas tarjetas de desarrollo como arduino y teensy.

Dispositivo/Descripción	**PIC16F84A**	**PIC18F2550**
Imagen		
voltaje operación	2,0V a 5,5V	2,0V 5,5V
Consumo de corriente típico	< 2mA a 5V, 4mhz	5,8uA modo inactivo
Memoria FLASH	2 KB	2 KB
Memoria EEPROM	64 Bytes	256 Bytes
Memoria Ram	68 Bytes	2KB
Numero de Pines	18	28
Contadores	1 de 8 bits	1 de 8 bits 3 de 16 bits
Canales PWM	No soportado	2 de 8 bits
Convertidos analógico digital	No soportado	10 canales de 10 bits
Comparadores lógicos	No soportado	2
Velocidad de oscilador externo	20 Mhz	48 Mhz
Velocidad oscilador interno	No soportado	8 Mhz
Comunicación	No soportado	USB v2.0, USART, SPI, I2c
ancho de bus	8 bits	8 bits
convertidor digital analógico	No soportado	No soportado
Precio	$ 12.000 Cop	$ 28.000 Cop

Tabla 5.1: especificaciones de los microcontroladores 1

una vez recopilado los datos de los microcontroladores teniendo los datos principales que permiten ver sus características mas fundamentales, escogiendo los dispositivos mas comunes usados en tarjetas de desarrollo que muchos hemos podido entrar en contacto y trabajar con ellos, se le hace la inspección a las tablas 5.1,5.2,5.3, se comienza excluyendo al microcontrolador MK66FX1M0VMD18 por su complejidad al momento de soldar y su alto costo. aunque cuenta con las mejores características y prestaciones, siendo un dispositivo de alta gama. para así llegar a elegir el microcontrolador ATSAMD21G18-AU por las siguientes razones:

- es un microcontrolador de 32 bits, permitiendo mayor escritura de palabra.

Dispositivo/Descripción	**ATMEGA-328P (arduino uno)**	**ATMEGA2560 (arduino Mega)**
Imagen		
voltaje operación	1,8V-5,5V	1,8V - 5,5V
Consumo de corriente típico	300μA modo activo	800μA modo activo
Memoria FLASH	32 KB	256 KB
Memoria EEPROM	1 KB	4KB
Memoria Ram	2 KB	8 KB
Numero de Pines	28	100
Contadores	2 de 8 bits 1 de 16 bits	2 de 8 y 4 de 16 Bits
Canales PWM	6 de 8 bits	14 de 8 bits
Convertidos analógico digital	8 canales de 10 bits	16 canales de 10 bits
Comparadores lógicos	1	1
Velocidad de oscilador externo	16 Mhz	16 Mhz
Velocidad oscilador interno	8 Mhz	8 Mhz
Comunicación	USB emulated, usart, SPI, I2C	USB emulated, usart, SPI, I2C
ancho de bus	8 bits	8 bits
convertidor digital analógico	No soportado	No soportado
Precio	$ 10.000 Cop	$ 42.000 Cop

Tabla 5.2: especificaciones de los microcontroladores 2

- cuenta con una memoria flash de mayor capacidad que la mayoría de microcontroladores seleccionados, aportando mayor almacenamiento.
- Cuenta cuenta con una gran cantidad de convertidores analógicos digitales
- cuenta con una salida digital- analógica
- cuenta con una gran cantidad de canales PWM
- la resolución del DAC y PWM son de 12 bits, lo que permite tener un rango de trabajo de 0 - 4095.
- la velocidad por ciclo de reloj es relativamente alta
- incorpora un puerto nativo de usb, mayor tipos de comunicaciones.
- Cuenta con un encapsulado de 48 pines, permite que la separación entre pines no sea mucha, permitiendo una fácil manipulación al momento de soldar.
- es un microcontrolador que bajo sus prestaciones sigue siendo de bajo costo.

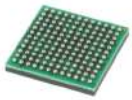
Dispositivo/Descripción	**ATSAMD21G18-AU (Genuino Zero)**	**MK66FX1M0VMD18 (teensy 36)**
Imagen		
voltaje operación	1,62V- 3,6V	1,71-3,6 V
Consumo de corriente típico	360μA modo activo	560μA modo activo
Memoria FLASH	256KB	1,25MB
Memoria EEPROM	16KB	256 KB
Memoria Ram	32KB	256 KB
Numero de Pines	48	144
Contadores	5 de 16bits y 4 de 24 bits	Información incompleta
Canales PWM	20 de 12 bits	22 de 12 bits
Convertidos analógico digital	14 canales de 12 bits	25 canalaes de 12 bits
Comparadores lógicos	2	4
Velocidad de oscilador externo	32kHz hasta 96 Mhz fracional	32Kz hasta 180 MHz con DSP
Velocidad oscilador interno	48 Mhz	48 Mhz
Comunicación	sercom, uart/usart, spi , i2c	Ethernet controller, USB ,SPI,I2C,SDHC,I2S
ancho de bus	32 bits	32 bits
convertidor digital analógico	1 de 12 bits	2 de 12 bits
Precio	$ 23.000 Cop	$ 60.000 Cop

Tabla 5.3: especificaciones de los microcontroladores 3

5.1.1.1. componentes de acondicionamiento

Para la selección de los componentes del microcontrolador ATSAMD21G18-AU, se hizo una búsqueda de la hoja de datos del microcontrolador genuino zero, utilizar el esquemático como guía para la elaboración del diseño, esta tarjeta de desarrollo implemente el mismo microcontrolador por lo que seria de gran ayuda y dará confiabilidad al momento de querer utilizar este microcontrolador, además se realizo una búsqueda de foros donde se haya implementado su circuito para así poder tener los componentes necesarios par así operación.

se tomaron las imágenes de los circuitos esquemáticos para seleccionar los componentes, que a su vez serán de gran ayuda para la elaboración del sistema embebido de este microcontrolador. la siguiente imagen 5.2 muestra el esquemático del microcontrolador del genuino zero.

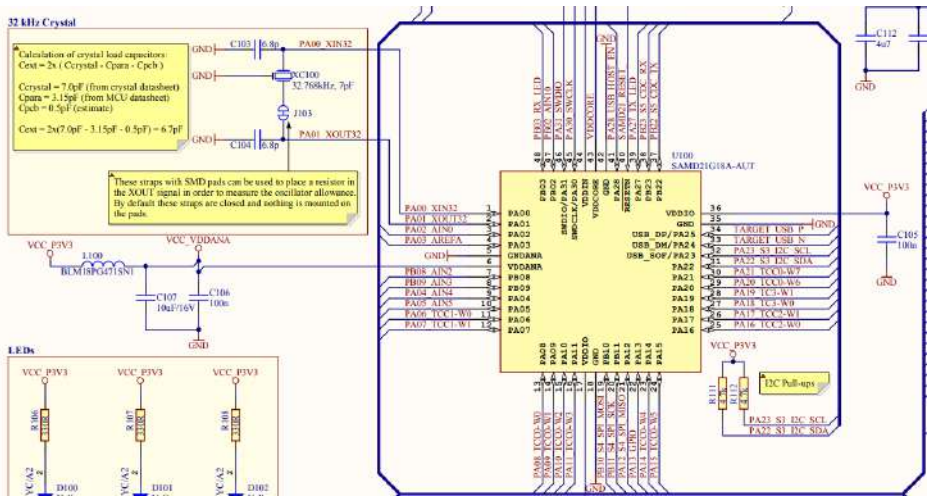


Figura 5.2: esquemático hoja de datos del genuino zero

Fuente: datasheet de genuino zero

según el esquemático anterior, suelen quedar ciertas inquietudes acerca de algunos pines de conexión si son necesarios usar y que hay componentes como la bobina de ferrita, que no son tan fáciles de encontrar en el mercado, se comparo este esquemático con uno encontrado en un foro que implementan también el microcontrolador, a partir de estos dos esquemas se comparan y se asocian los pines necesarios a usar, para poder acondicionar el microcontrolador para su operación el siguiente imagen se puede ver el esquemático de manera mas entendible para su implementación.

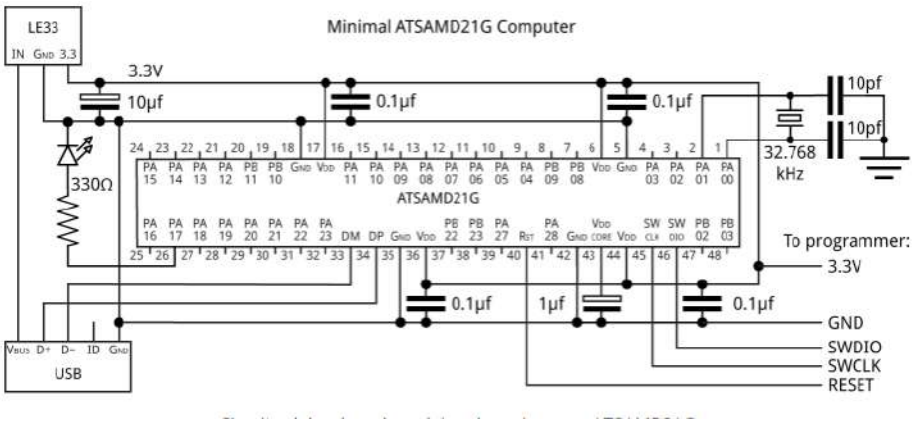


Figura 5.3: esquemático ATSAMD21G18-AU

Fuente: (David Johnson-Davies, 2019)

con el esquema de la imagen 5.3, se aprecian mejor los valores de referencia de los componentes de capacitores y cristal de cuarzo, además vincula los pines de micro usb y programación del microcontrolador.

5.1.2. Regulador de Voltaje

una vez sabiendo con que microcontrolador se va a trabajar,se puede sacar los valores de voltaje de operación establecidos por la hoja de datos, queda seleccionar el regulador e implementar el circuito dado en la metodología visto en la figura 4.6. como el microcontrolador trabaja con voltajes de 3.3V, se busca un regulador de esas características y con un encapsulado pequeño para la optimización del espacio. el regulador escogido presenta las siguientes características.

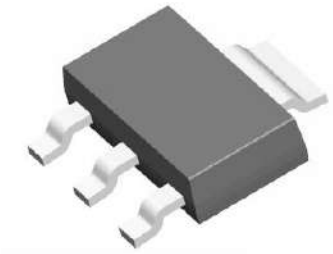
Regulador AZ1117H-3.3TRE1

Voltaje de salida: 3.3 V
Corriente de salida: 800 mA
Número de salidas: 1 Output
Corriente de reposo: 4 mA
Voltaje de entrada MÁX.: 15 V

Tabla 5.4: Características de regulador con encapsulado SOT-223

la referencia de los condensadores a usar, se maneja bajo un valor estándar de capacitores entre los rangos de (0.1 - 100)μF, colocando un condensador a la entrada como filtro para la fuente externa de entrada, y un condensador a la salida del regulador, igualmente para filtrar cualquier ruido o señal de alta frecuencia.

5.1.3. Regulación Voltaje en la Señal Entrada

en la regulación voltaje de la señal, se manejan valores de entrada entre 0-10V, lo que se vuelve importante implementar un circuito de reducción, para obtener un valor de voltaje de 3.3V máximo a la entrada del microcontrolador cuando el voltaje máximo sea de 10V, siguiendo el esquema del circuito de la figura 4.7, se calculan las respectivas resistencias, con la ecuación 4.1. voltaje de entrada= 10V; voltaje de salida= 3.3V, con un valor de resistencia R1=1k.

$$R_2 = \frac{R_1 \left(V_{int} - V_{out}\right)}{V_{out}} \tag{5.1}$$

Reemplazando los valores anteriores da el siguiente resultado. como la intensidad de corriente de entrada es poca, porque es una señal de control, se pueden utilizar este tipo de resistencias

$$R_2 = 2K\Omega \tag{5.2}$$

la alta impedancia de entrada es favorable para el circuito amplificador ayuda a tener una buena amplificación de la señal de entrada, al seleccionar un amplificador operacional aparte de fijarse en la ganancia que ofrece, se analiza los valores de voltaje de funcionamiento y la corriente que puede entrar cada canal del amplificador operacional y el tipo de encapsulado y numero de pines, por lo que se selecciono el siguiente amplificador operacional siguiendo el criterio anterior.

Amplificador operacional LM324

Voltaje de alimentación Min-Máx: 3V - 32 V
Corriente de salida por canal: 800 mA
Número de salidas: 4 Output
Corriente de suministro operativa: 175 μA

Tabla 5.5: Características de Amplificador Operacional LM324

5.1.4. Convertidor de salida Digital analógico DAC

como es bien sabido los microcontroladores rara vez cuentan con salidas o no cuentan con salidas analógicas Reales bueno acondicionar una salidas y así poder ampliar su aplicación para uso de otros dispositivos de control, siguiendo la metodología 4.10 se tiene el circuito modelo y se hace es calcular los valores de resistencias y condensandores que los conforman.

básicamente el circuito es un amplificador operacional seguidor de voltaje, teniendo una ganancia unitaria, el voltaje pasa por el C1 y es igual al voltaje de salida, el procedimiento de diseño para este tipo de filtro es:

- Seleccionar la frecuencia de corte ω_c o bien F_c. En este caso se escoge una frecuencia de corte de 16.6 Hz, para tener la seguridad de filtrar la frecuencia de 50 Hz en el tercer armónico, pues al ser la frecuencia de suministro de la red eléctrica es fácil que provoque ruido por inducción.(Soriano, 2012)
- Escoger C1, seleccionando un valor entre 100 pF y 0.1 μF. Al ser la frecuencia de corte tan baja se escoge el valor más alto de 0.1 μF, para evitar obtener un valor muy alto de R.(Soriano, 2012)
- El valor del condensador C2 = 2*C1. Por tanto, C2 = 0.2 μF
- El valor de R se calcula mediante la siguiente expresión:

$$R = \frac{0.707}{\omega_c * C} = \frac{0.707}{2\pi * 16.6 * 0.1 * 10^{-6}} = 67.784\Omega = 68K\Omega \tag{5.3}$$

- Se selecciona el valor de las resistencias cumpliendo estas condiciones: R = R1 = R2 y R3 = 2R, por tanto: R1 = R2 = 68 KΩ y R3 = 136 KΩ

como este circuito opera como un seguidor de tensión, lo que quiere decir que el voltaje máximo alcanzado y dado por el microcontrolador es de 3.3V se debe aumentar ese voltaje a un valor de 10V máximo, tomando la ecuación de un amplificador de voltaje de configuración no inversor vista en la siguiente ecuación, se realizan los cálculos de resistencias:

$$V_{out} = \left(1 + \frac{R_2}{R_1}\right) V_{int} \tag{5.4}$$

sabiendo los siguientes valores de voltaje de entrada V_{int}=3.3V, V_{out}=10V y un valor iterativo de resistencia R1=1k.

$$R_2 = \left(\frac{V_{out}}{V_{int} - 1}\right) * R_1 \tag{5.5}$$

reemplazando los valores se obtiene como resultado un valor de resistencia aproximado R2= 2 KΩ.

5.1.5. Periféricos de conexión, visualización y control

cuanto se trata de un sistema embebido de control, muchas veces es bueno tener control sobre algún, tipo de variable. debido a esto es importante tener una pantalla que permita visualizar la variable, así ver en tiempo real la variable. lo que se convierte en requisito tener algún tipo de pantalla. se ha optado en usar un tipo de pantalla oled, presentando ventajas, comunicación, espacio de trabajo y tamaño. vistos en la siguiente tabla.

Pantalla OLED

Voltaje de alimentación Min-Máx: 3.3V - 5V
Tamaño: 0.96”
Interfaz: I2C
Resolución: 128x64 píxeles
Monocromo: Píxeles Blancos

Tabla 5.6: Características de Pantalla OLED

otro ítem importante para la interacción con el sistema embebido, es poder movernos por

medio de un menú ser mas dinámico y tener mayor control sobre los procesos, lo que es importante implementar un elemento que permita cumplir esos requerimiento y el mejor con esas prestaciones es el uso de un encoder rotativo con pulsador interno, reduce espacio y permite mayor libertad,las características de este dispositivo se ven en la siguiente tabla.

Encoder rotativo

Voltaje de alimentación Min-Máx: 3.3V - 5V
Rotación: 360°
Número de pulsos: 15
Vida útil: 30.000 ± 200 ciclos
Dimensiones: 33.8mm x 22.4mm

Tabla 5.7: Características encoder

este sistema embebido de control, va a trabajar con señales eléctricas, por lo que el uso de un conector rápido, robusto y de dimensiones favorables, se vuelve importante al momento de seleccionarlo, por ello se opto en usar el siguiente conector visto en la siguiente tabla.

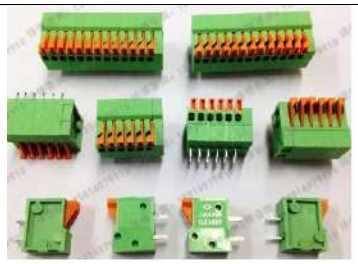
Bornera KF141R

Numero de polos: 10
Separación entre pines: 2.54mm
Tipo: Through Hole

Tabla 5.8: Bornera KF141R

5.2. Diseño De Placa Electrónica y Case

Una vez definido los materiales y tener los bosquejos o modelos de algunos circuitos electrónicos. El diseño de una placa de circuito impreso, tiende ir mucho a criterio del diseñador, sin embargo se mantienen en cuenta las normas de diseño, NORMA IPC 2220 para diseño de PCB- PCB Design. a parte de las reglas de diseño, se debe tener en cuenta el lugar donde se pretende mandar a fabricar la PCB, como algunos aspectos importantes que se debe estar pendiente, como ancho de linea en dependencia de la corriente que va a pasar, espaciado entre lineas y tipo de material.
la elaboración de los diseños se realizo en el software EAGLE de autodesk con licencia de estudiante , el diseño esta realizado en dos etapas, el diseño del esquemático en donde se seleccionan los respectivos elementos, aquellos elementos con los que no cuenta propios del programa, se hizo ayuda de foros, que brindan las librerías de los elementos, y que a su vez se realizaron las librerías de elementos que eran muy difícil de encontrar en la web.
el esquemático se mostrara de manera seccionada, si se muestra de manera completa no se lograría apreciar bien los elementos que lo conforma. los esquemáticos de estos circuitos van en una carpeta llama **diseño_sistema** para abrir en archivos de eagle.

5.2.1. Esquemático de Fuente de Alimentación

El circuito de alimentación esta conformado por el integrado AZ1117H-3.3TRE1, toma la el voltaje de entrada de 5V o 12, y lo reduce a 3.3 V, que sera la alimentación interna del microcontrolador y demás elementos que operen a este nivel de tensión, cuenta también con un conector jack universal, para que pueda ser usado con cualquier tipo de fuente que cumpla estos niveles de voltaje, y un diodo de protección en la alimentación USB

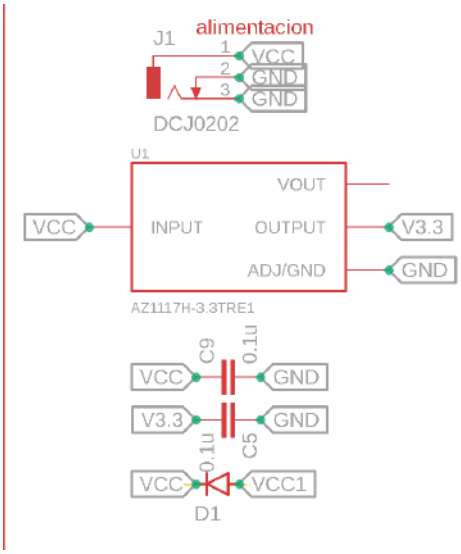


Figura 5.4: esquemático alimentación
Fuente: AUTOR

5.2.2. Esquemático Del Microcontrolador

Para el circuito de acondicionamiento del microcontrolador se baso bajo el esquema visto en la metodología del microcontrolador, haciendo las respectivas conexiones y colocando los elementos para su operación, de igual manera se demarcan los tags para las demás conexiones a los elementos, como señales de entrada analógica-digitales, salidas, pines del encoder, pulsador de entrada, botón de reset, cristal de cuarzo y pantalla oled este esquemático se vera en la figura 5.5

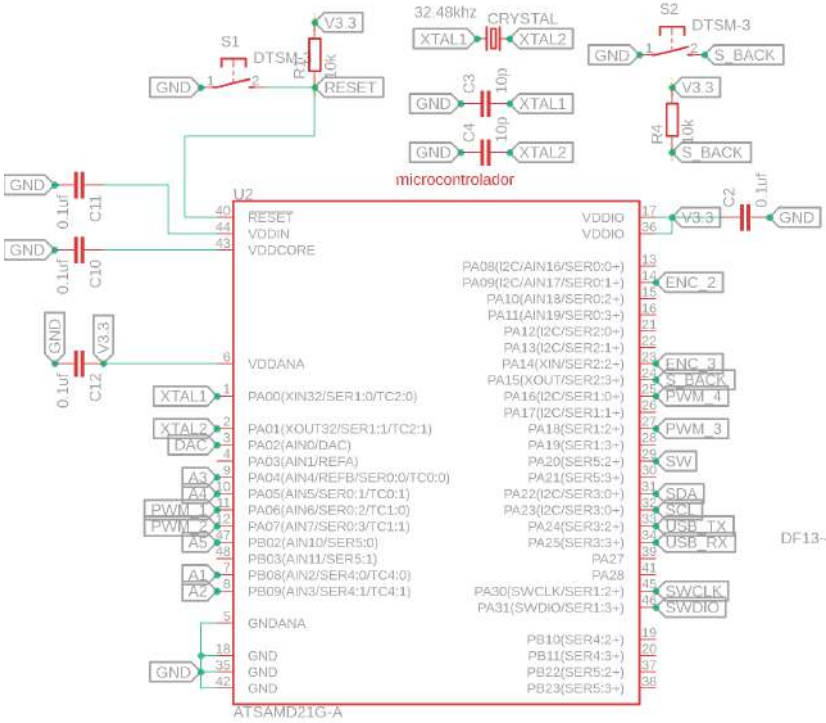


Figura 5.5: Esquemático Microcontrolador
Fuente: AUTOR

5.2.3. Esquemático Para El Acondicionamiento De Señales De Entrada

En la etapa de entrada viene conectada a un divisor de tensión,la entrada física esta dada por los conectores kf141R de 10 polos asignando a cada entrada un GND, se asignaron 5 entradas en los puertos analógicos-digitales del microcontrolador, estos mismos puertos asignados, son los utilizados en el genuino zero, para su facilidad al momento de querer usarlos en el ide de programación. una vez reducido el voltaje en una proporción del 3.3, y con un voltaje de entrada de 10 voltios y genere un voltaje de 3.3V en los puertos de entradas analógicos del microcontrolador. con este circuito suple la necesidad de reducir el voltaje, sin embargo el circuito puede sufrir si el voltaje aplicada llegar ser mayor, que también puede ser afectado por el ruido en la entrada, por ello se le coloca un amplificador operacional en modo seguidor de tensón y un diodo de protección, dado el caso que el voltaje de entrada llegue a ser mayor a los 10V, se menciono que se dispusieron de 5 entradas el esquemático

y se esta presentado un esquema general, debido a que las demás salida operan en base al mismo circuito descrito en la figura 5.6

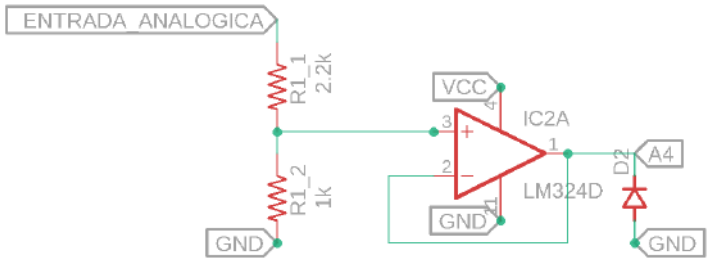


Figura 5.6: Esquemático Divisor De Voltaje
Fuente: AUTOR

5.2.4. Esquemático Del Circuito De Acondicionamiento De Señales De Salida

La etapa de salida van asociados los pines digitales del microcontrolador, estos pines que deben estar configurados para operar como PWM, de este modo generar la salida analógica, ya que esta señal pasa por un filtro Butterworth de 2º orden, convirtiendo ese PWM de 3.3v a un valor analógico real, pero como se desea una salida en la escala 0-10V se asigna un elevador de voltaje a cada puerto de salida analógico, y esa salida ya elevada, va un conector kf141R, se enseñaran dos puertos de salida, pues para los demas puertos se mantiene la misma configuración, solo cambian los dos puertos de asocian de entrada y salida al conector de igual manera con un GND asociado a cada puerto de salida, el esquemático se puede visualizar en la figura 5.7

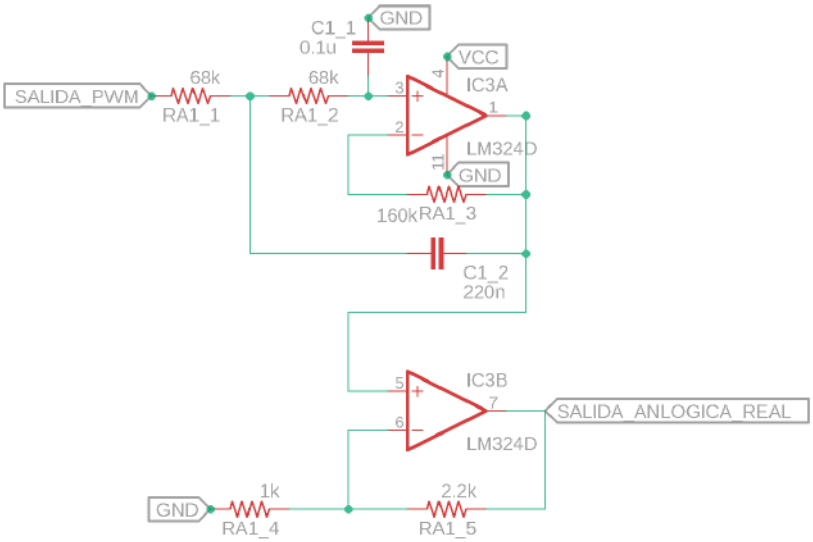


Figura 5.7: Esquemático Salida Análoga (DAC)
Fuente: AUTOR

en la figura 5.8, se presenta un circuito utilizando un amplificador lm358 de dos canalaes, ya que los amplificadores lm324, solo tienen 4, por diseño y espacio se utiliza este para implementar los circuitos faltantes, de entrada anlogica, y salida real del microcontrolador, por ello no lleva filtro de segundo orden, se conecta directamente para elevar el voltaje.

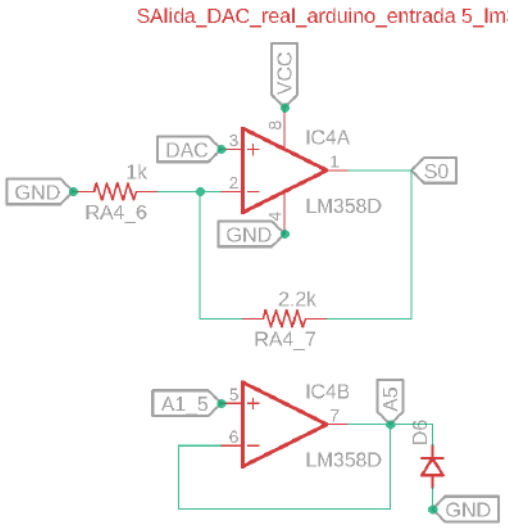


Figura 5.8: Esquemático Entrada ADC Con Salida DAC

Fuente: AUTOR

5.2.5. Conector para periféricos

en los periféricos se mostraran los tags de asignación acorde a los componentes usados, en este caso los conectores van asociados al uso de la pantalla OLED, el Pulsador de entrada, y el encoder rotativo, visto en la figura 5.9

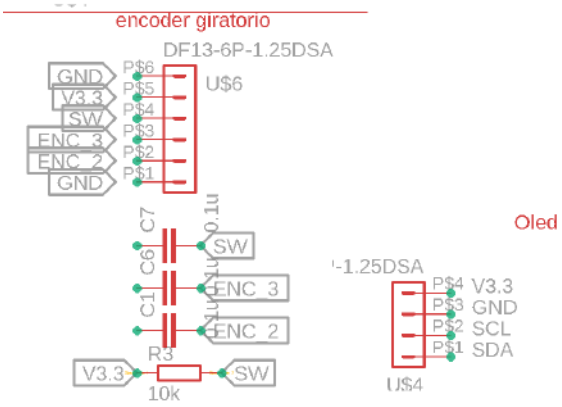


Figura 5.9: Esquemático De Puertos de Oled Encoder

Fuente: AUTOR

otro conector es el usb, que incorpora el microcontrolador facilitando el uso de este mismo,

y no tener que utilizar un cargador, aunque se dejan los pines de programación del cargador para poder subir el boolader, o gestor de arranque del microcontrolador, visto en la figura 5.10

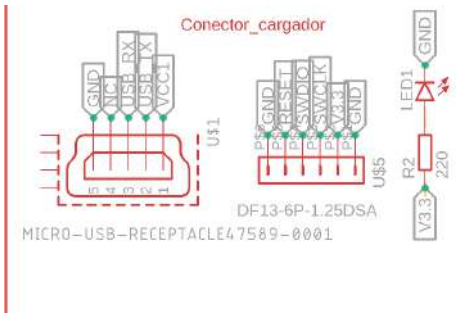


Figura 5.10: Esquemático De micro USB y Programador
Fuente: AUTOR

el esquemático general donde se aprecian todo los elementos en una sola imagen estará ubicado en la sección de anexos, en la figura A.1

5.2.6. Diseño De La PCB Board

una vez teniendo todos los elementos a usar en el sistema embebido, se realiza la unión de cada uno de los caminos asociados para generar la placa de circuito impreso, como se menciono se tienen en cuenta las normas IPC al momento de diseñar, el resultado del diseño sera seccionado en varias etapas donde se visualizara con mayor detalle, para poder realizar los diseños de la board se tuvo en cuenta el lugar de fabricación de la PCB, por las exigencias de requisitos mínimos, en este caso se tenia optado en fabricar en la empresa nacional Col-circuitos ubicada en Medellín, para el diseño de la board del circuito se emplearon componentes de montaje superficial SMD, por el motivo que permiten ahorrar espacio, reducen el ruido eléctrico y dan presentaciones mas profesionales al dispositivo .

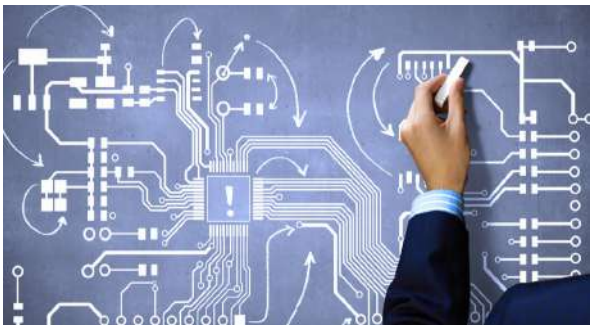


Figura 5.11: Diseño de Board
Fuente: (david, 2017)

5.2.6.1. capa Top

la capa Top es la vista superior de las pista y elementos que conforman esta cara de la pcb vista en la figura 5.12, lo que se puede observar las pistas que salen del microcontrolador a los demás elementos, las puentes que son las perforaciones que permiten comunicar con la otra capa, se puede hacer una dimensión de los elementos como los conectores kf141r, el conector jack y USB.

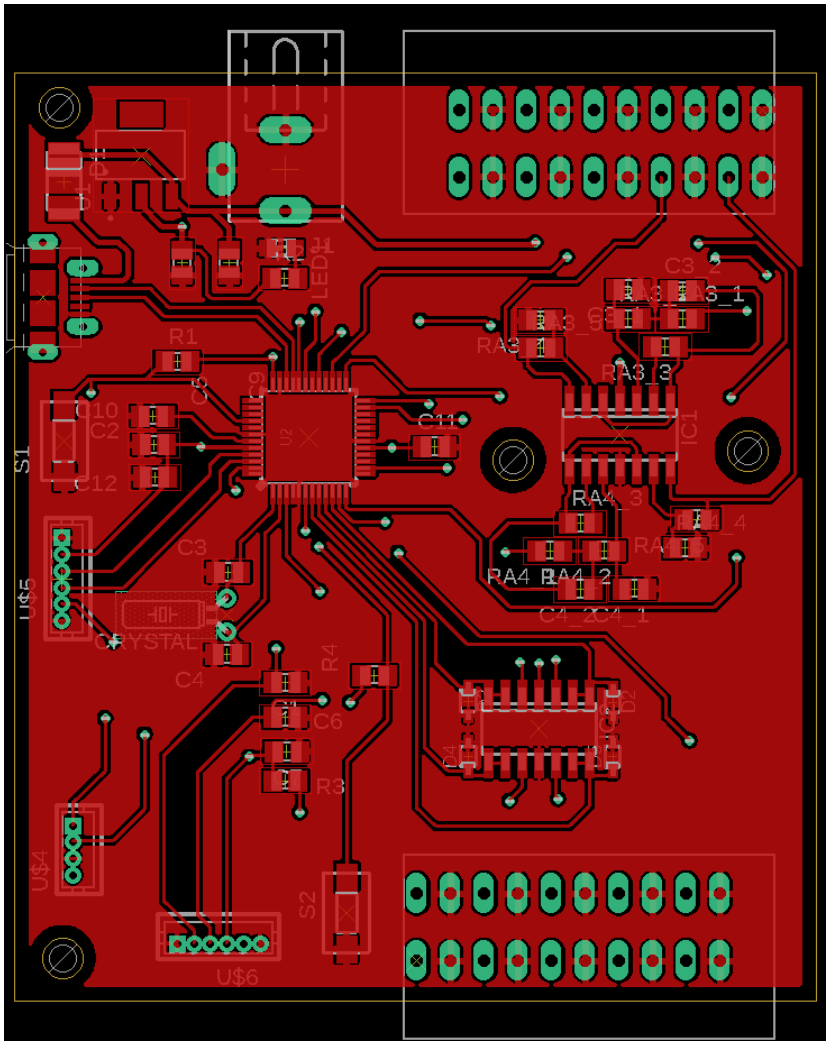


Figura 5.12: Board capa top

Fuente: AUTOR

5.2.6.2. capa button

la capa button es la vista por la parte inferior de la pcb, donde aprecian los elementos que por espacio o facilidad de diseño, no se encuentran en la capa Top, tal como se ve en la figura 5.13, es esta capa se puede observar las resistencias del divider de voltaje y el amplificador lm324 y el lm358.

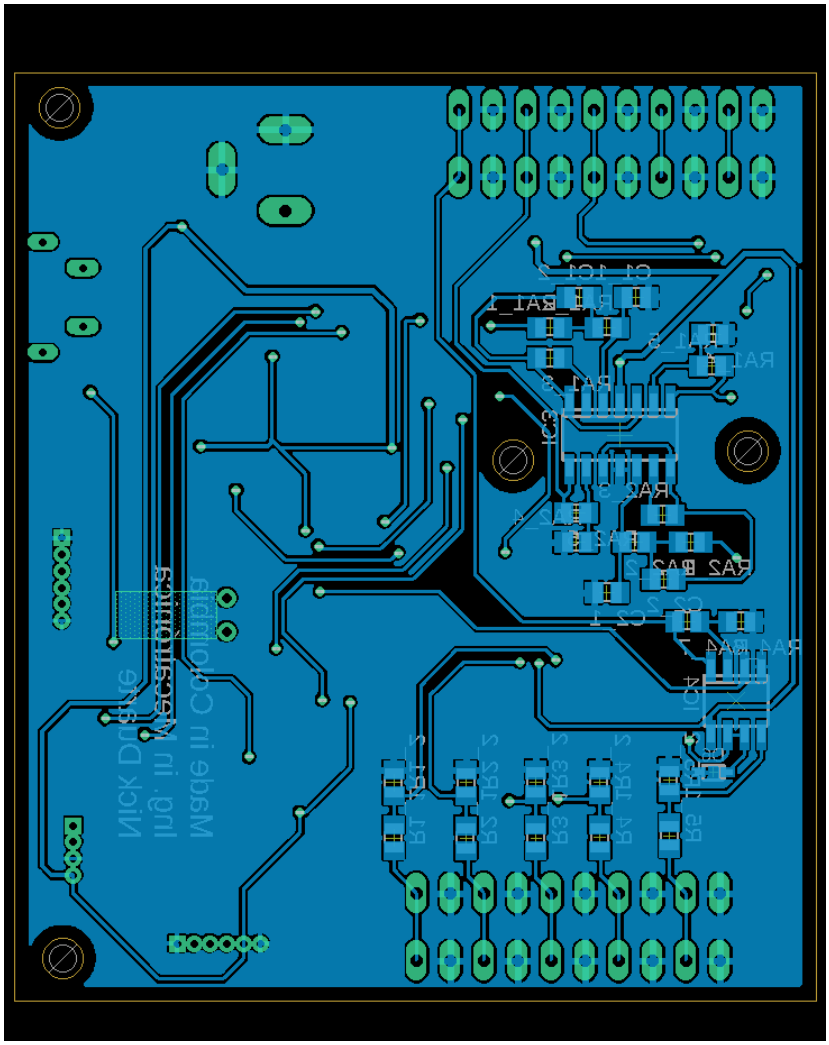


Figura 5.13: Board capa Button

Fuente: AUTOR

5.2.6.3. Modelo 3D de la PCB

la plataforma de eagle permite enlazar con el programa de diseño 3D de fusión 360, y teniendo las librerías completas se puede dar un bosquejo a modo de simulación de como quedara la placa en su acabado final, tal como se pueden ver en la figura 5.14, el modelo en la capa top de la PCB.

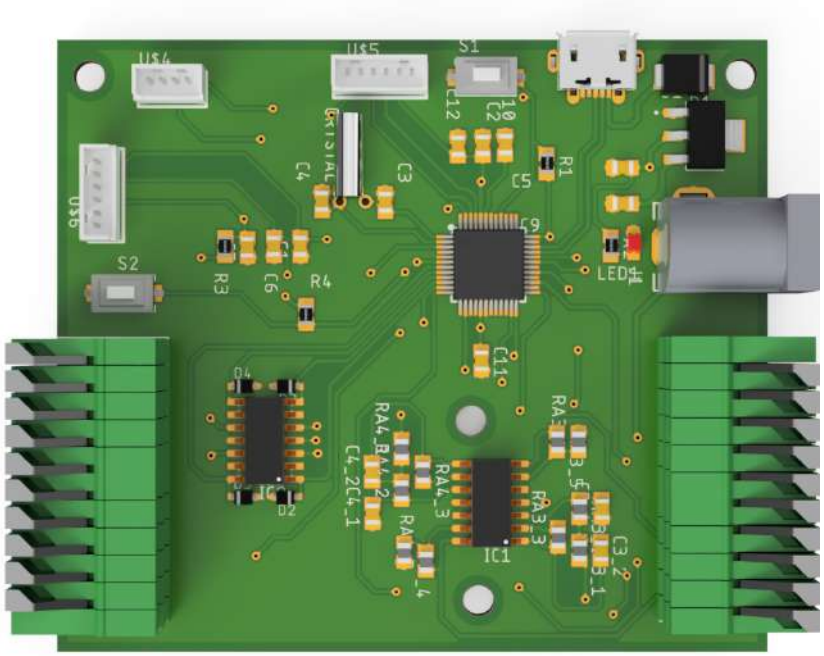


Figura 5.14: Board capa TOP
Fuente: AUTOR

en la figura 5.15 se puede ver el circuito en la capa button, y en la figura 5.16 el isometrico en 3d de la perspectiva de la PCB.

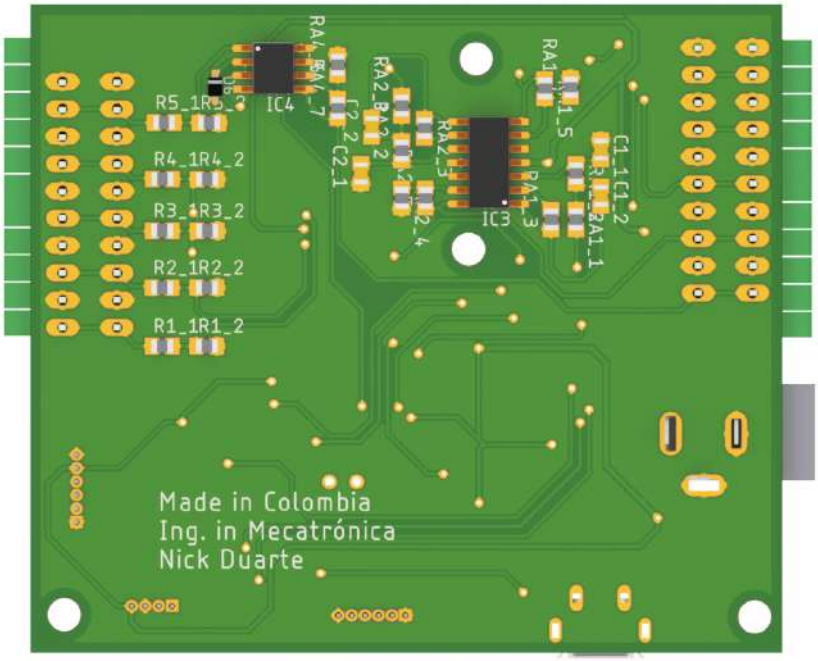


Figura 5.15: Board capa Button
Fuente: AUTOR

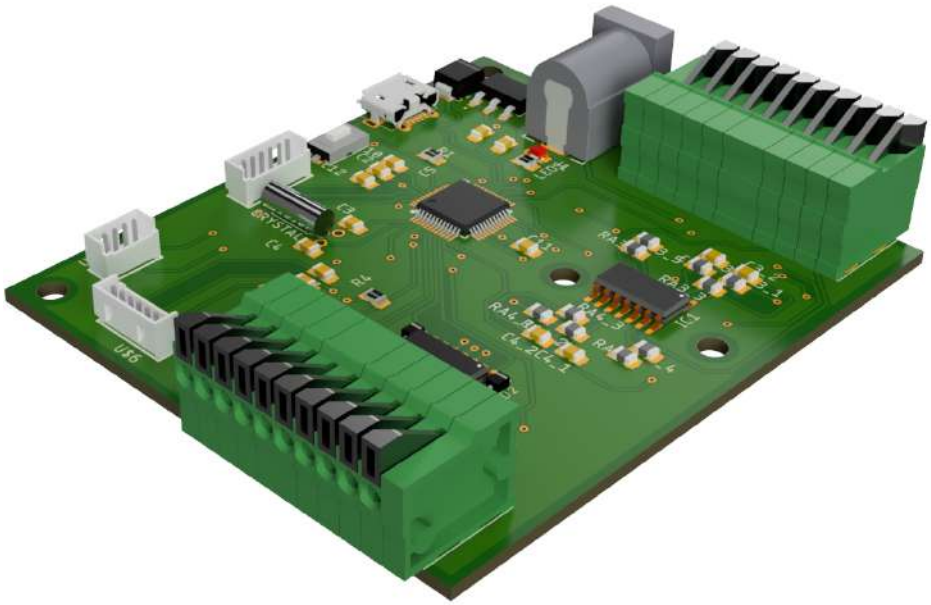


Figura 5.16: Board isometrico
Fuente: AUTOR

5.2.7. Diseño del case

la facilidad que ofrece tener un modelado 3d de la PCB, es al momento de diseñar el case, ya que permite tener las medidas reales de la PCB las dimensiones de la placa y agujeros como los espacios que ocupan los elementos. se trato de mantener un diseño simétrico del sistema embebido y que fuese un diseño agradable a la vista.
el case esta compuesto por 5 elemento:

- tapa inferior, verse en la figura 5.17
- tapa superior,verse en la figura 5.18
- Asegurador de la pantalla oled,verse en la figura 5.21
- tapa del cargador,verse en la figura 5.20
- Pulsador,verse en la figura 5.22

5.2.7.1. Diseño Tapa Inferior

la tapa inferior cuenta con unos agujeros pasantes que permiten asegurar la PCB y asegurar con la tapa superior.

Figura 5.17: Tapa inferior
Fuente: AUTOR

5.2.7.2. Diseño de la Tapa superior

El diseño de esta tapa, se le debió tener en cuenta que la pantalla oled debería asegurarse a este y de igual manera el encoder rotativo, y darle un espacio al pulsador asignado en la PCB, también se debía tener en cuenta el espacio para poder conectar el cargador y este espacio fuese re-movible.

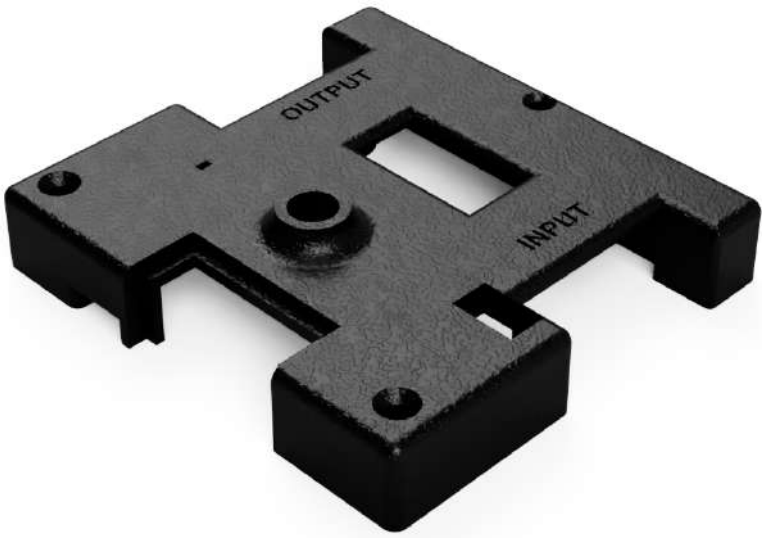


Figura 5.18: Tapa superior vista Top
Fuente: AUTOR

Figura 5.19: Tapa superior vista Button
Fuente: AUTOR

5.2.7.3. Diseño tapa cargador

Esta pequeña tapa permite liberar y ocupar un espacio en la tapa superior, siendo la encargada de ser retirada cuando se necesite cargar el programa y cuando ya se haya terminado, se coloca en su lugar tapando el espacio visto en la tapa superior del case. ver figura

Figura 5.20: Tapa Cargador
Fuente: AUTOR

5.2.7.4. Asegurador Oled

este asegurador oled es un elemento que permite mantener fija la pantalla OLED en la tapa superior del case, siendo atornillada a unos extremos donde tiene sus respectivos orificios.

Figura 5.21: Asegurador OLED
Fuente: AUTOR

5.2.7.5. Pulsador

Este pequeño pulsador, cumple con la tarea de retornar en el menu, y como el pulsador de la PCB queda retirado de la parte suprior del case, se le debe acondicionar un elemento que permita accionarlo.

Figura 5.22: Asegurador OLED
Fuente: AUTOR

5.2.7.6. Ensamblaje

En el ensamblaje se dara aun pre-visualización del producto final como debería quedar al momento de armarlo en físico, si todas las medidas de diseño están bien, ya se podría pasar a la etapa de fabricación, los componentes totales a usar en el ensamble se podrán ver en la tabla 5.9

Numeración	Descripción
1	Tapa cubierta inferior
2	Tapa cubierta superior
3	Circuito sistema embebido
4	Pantalla Oled 128x64 I2c
5	Módulo encoder de rotación EC11
6	Pulsador
7	Tornillo M3 con tuercas
8	Asegurador de pantalla Oled
9	Tapa cargador
10	Perilla encoder

Tabla 5.9: Componentes del dispositivo sistema embebido

en la figura 5.23, se visualizara el componente asociado a la etiqueta de la tabla anterior.

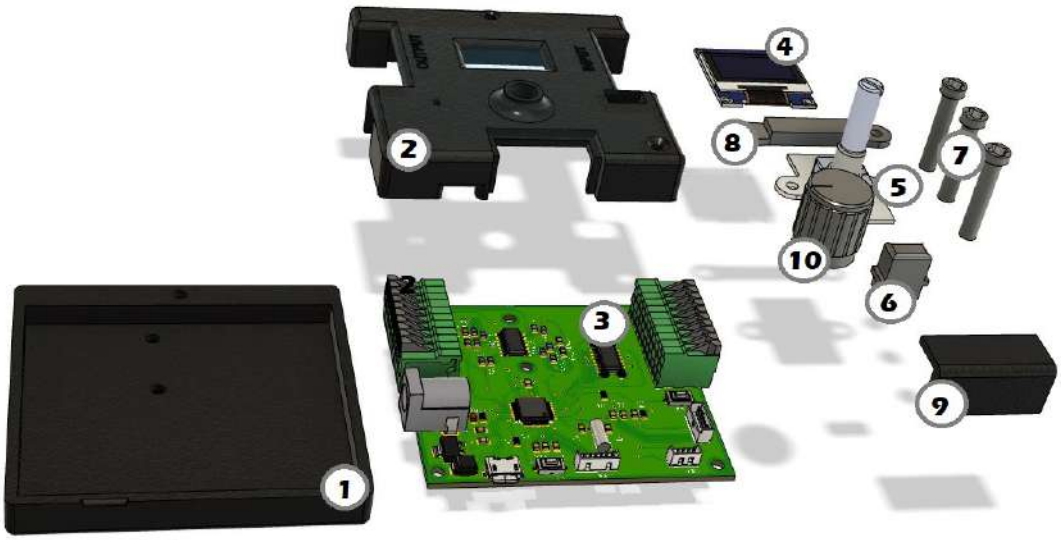


Figura 5.23: Modulo Sistema Embebido explosionado

Fuente: AUTOR

Para dar como resultado el por medio de simulación en ensamblaje visto en la figura 5.24 de una pre-visualización del dispositivo sistema embebido usado como sistema de control

Figura 5.24: Modulo Sistema Embebido
Fuente: AUTOR

5.3. Diseño de Interfaz

El desarrollo de una buena interfaz de usuario, se convierte en un punto vital en este tipo de dispositivos para la comunicación entre el sistema embebido y la persona que lo opere. por lo que elaborar esta interfaz debe llevar unos lineamientos a las exigencias de uso o la aplicación que se le va a dar. Por empezar llevara un menú interactivo y dar cumplimiento a la validación de 3 variables física de dinámica diferente, permitiendo seleccionar la variable a controlar, sea Temperatura, nivel o velocidad, una vez se seleccione se desplegara un sub-menú, pidiendo los siguientes parámetros:

- Valor del Setpoint
- Valor Máximo
- Valor mínimo
- Selección de puertos de entrada y salidas
- Sguiente, desplegara la otra etapa del menu.

abra un indicador que permita saber cual es la posición en la que estamos ubicados y poder seleccionar bien la casilla que se quiere acceder, las posiciones del setpoint, resolución máximo, resolución minina, llevara un sub-menu interno en que se podrá ver 3 casillas y un titulo diciendo incremento viendo el diagrama en la figura 5.25, queriendo decir cual sera la resolución en la que vamos a variar el valor seleccionado los incrementos visto en la sección de incremento son los siguientes:

- incremento en **100** unidades.
- incremento en **10** unidades.
- incremento en **1** unidades.

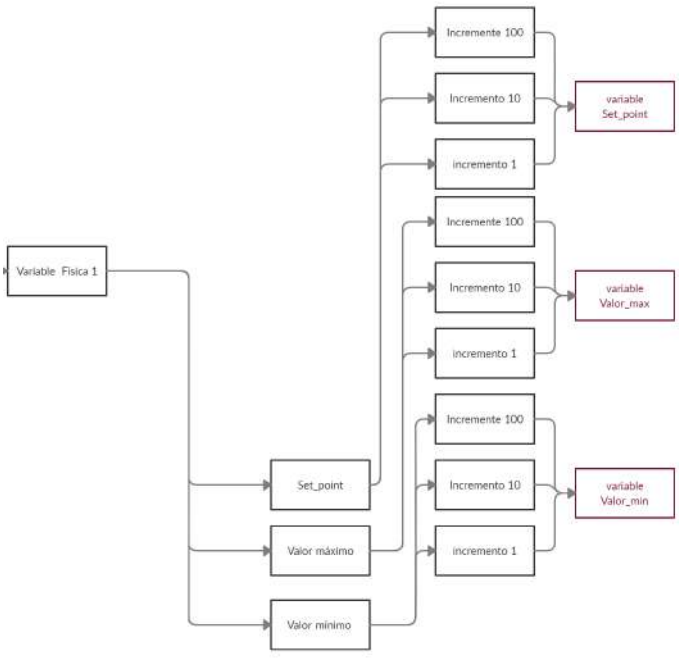


Figura 5.25: Selección de variables con incremento regulable
Fuente: AUTOR

en la selección del sub-menu principal de cada variable se encuentra una casilla de selección de puertos de entrada y salidas, solo selecciona un puerto como entrada, entre puerto 1 al 5, aplica de la misma forma para la selección del puerto de salida a usar, ver firura 5.26

Figura 5.26: Selección de Puertos entrada y salida
Fuente: AUTOR

se encontrada una ultima casilla denominada **Siguiente**, la cual desplegara un nuevo menú, una vez seleccionando nos indicara que tipo de controlador desea usar:

- Control PID
- Control Neuronal

al llegar a seleccionarse el controlador clásico PID, se despliega un ultimo menú indicando las constantes del PID permitiendo ser alteradas accediendo en el mismo menú, visualizando el error y la señal de control registrada. Al seleccionarse el controlador Neuronal se pedirán los datos del tipo de topología que se quiere usar, registrando esos datos accede al menú de visualización, del valor del error y el valor de entrada registrado, y la taxa de aprendizaje puede ser regulable tal como se ve en la figura 5.27.

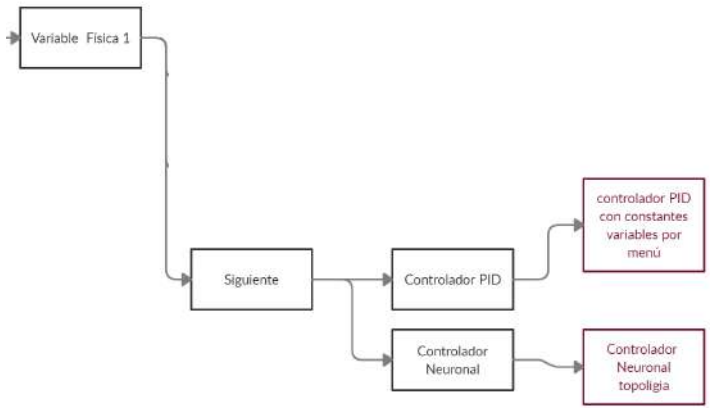


Figura 5.27: Modulo Sistema Embebido
Fuente: AUTOR

los menús internos de todas las variables físicas, serán los mismos, lo que va a cambiar es el tipo de forma o sensor en como se capturan los valores de entrada.
la arquitecta en como se va a manejar el menú se podrá ver en la figura 5.28, solo se explico el comportamiento de una sola variable, por lo que los demás sub-menús de interfaz van a ser los mismos, en cualquier variable a controlar, de tal forma se va a llevar esta estructura para la realización del código incorporado en el sistema embebido.

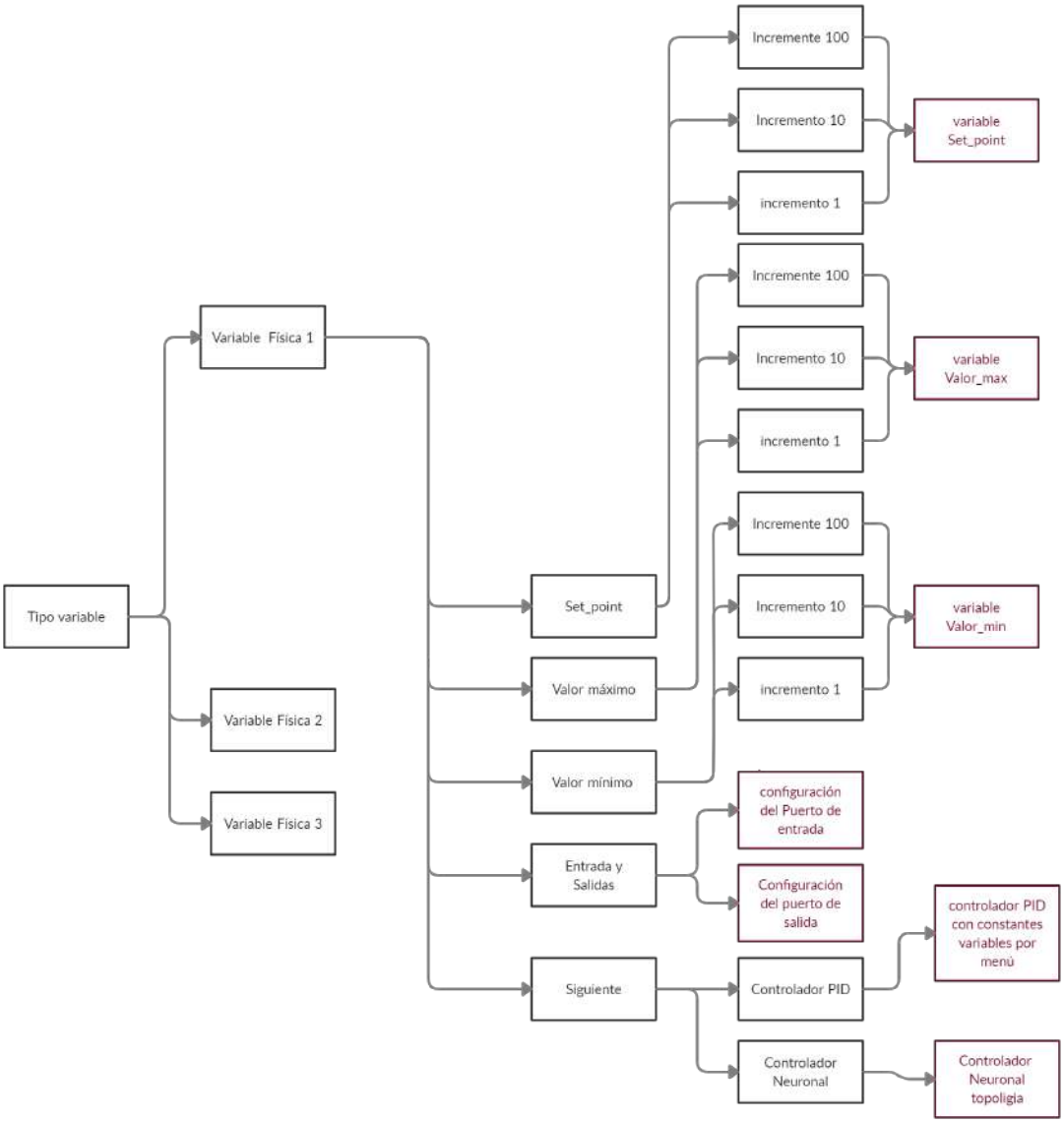


Figura 5.28: Modulo Sistema Embebido
Fuente: AUTOR

5.4. Acondicionamiento e Identificación Del Modelo De Planta

En los acondicionamientos de las plantas,se realizaran las respectivas conexiones de control, alimentación respectivos de los elementos que se deben utilizar y se emplearan los lazos de control vistos en la metodología para poder aplicar el controlador adaptativo neuronal al sistema. Además para dar cumplimiento al objetivo de validación del control neuronal, se debe utilizar un controlador de referencia. se decidió escoger controlador clásico PID o derivado de este mismo, por su facilidad, eficacia al aplicarlo en alguna planta, es pertinente hacer una identificación del modelo de la planta para hallar las constantes del Controlador.

5.4.1. Planta Temperatura

En el acondicionamiento de la planta de temperatura según la metodología se hace uso del lazo de control establecido ver figura 4.17.para obtención de la señal de entrada, se emplea el uso del sensor LM35 que cumple con la lectura de la temperatura ambiente en tiempo real que es codificada esta señal por el sistema embebido por un puerto de entrada, este modulo solo opera con este tipo de sensor.Cuando el sistema embebido ejecute los algoritmos de control adaptativo o control clásico, envía una señal de salida por un puerto de salida del sistema en el rango de tensión de 0-10V en relación al valor de entrada.
Esta señal es enviada a un divisor de voltaje que regula la tensión a la mitad, después de reducirla es recibida por el arduino ya que solo opera sus entradas analógicas a un rango máximo de 5V, el arduino ya viene con algoritmo precargado de control de fase y detección de cruce por cero, codificando la señal de entrada y siendo replicado el valor de salida en un rango de 0-100%, controlado el dispositivo final de control del proceso en lazo cerrado ver figura.5.29

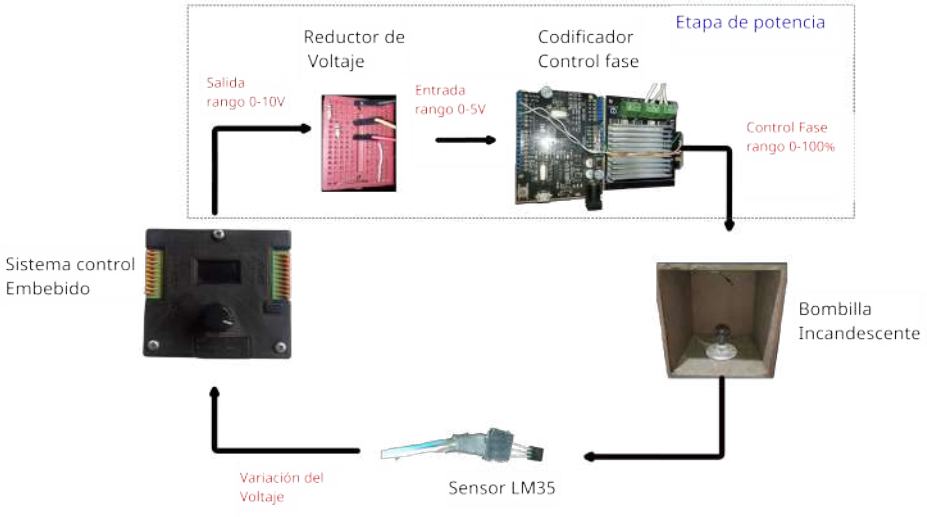


Figura 5.29: Lazo de control Planta Temperatura
Fuente: AUTOR

En la figura 5.30 se puede apreciar el montaje real de la planta de Temperatura, con las

conexiones al sistema embebido.

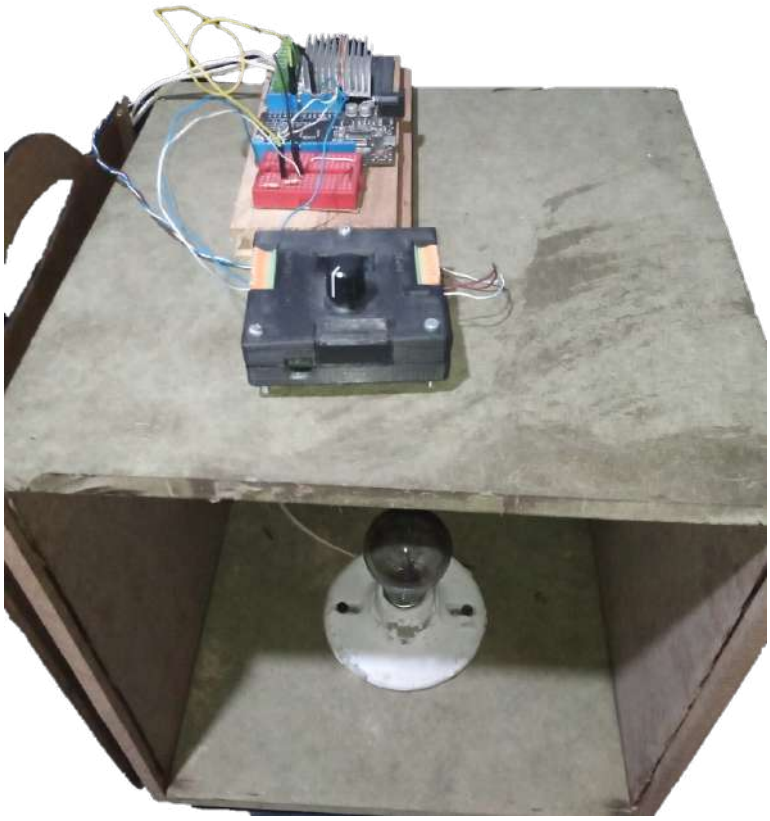


Figura 5.30: Montaje real Lazo de control Planta Temperatura
Fuente: AUTOR

5.4.1.1. Identificación Del Modelo De Planta

Para la identificación del modelo de la planta de Temperatura, se envía una señal en alto al elemento final de control, en este caso el bombillo incandescente, colocando el sistema de tal modo que pueda conservar mejor mejor la energía emitida por este efector y tener condiciones que no sufran de variación por la temperatura externa, cuando se envía la señal, al mismo tiempo los datos registrados por el sensor de temperatura, son tomado y guardados en un archivo de excel por el puerto serial, esto se puede realizar con una tarjeta de desarrollo cualquiera o en este caso se utliza el sistema embebido, los datos registrados del comportamiento de la planta a lazo abierto son los visto en la figura 5.31, la señal enviada es la máxima permitida por el dispositivo, por lo que se analizo cual fue el valor máximo de temperatura emitido por la bombilla, se tomo como valor máximo alcanzado por este elemento de unos aproximadamente 70°C. para hacer una debida identificación se envió una señal hasta que el valor de tempera ruta fuera estable, en este caso un valor de temperatura de 30°c y a partir de ello de envió la señal en alto y toma de datos.

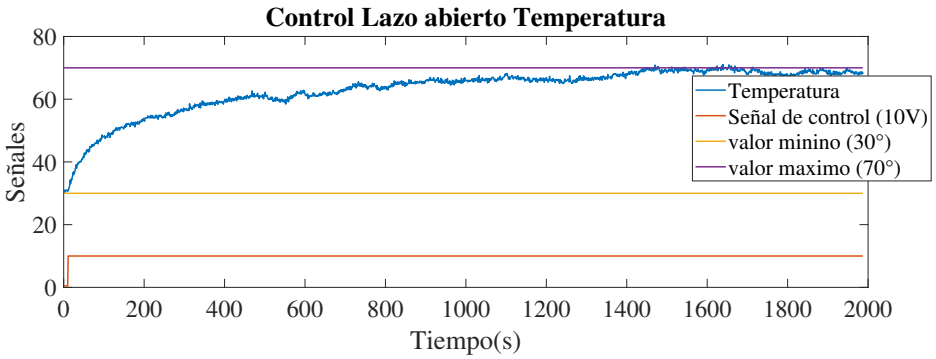


Figura 5.31: control a lazo abierto planta temperatura ante una entrada escalón
Fuente: AUTOR

ya con estos valores de salida obtenidos del comportamiento bajo una entrada escalón, se procede a procesar los datos y encontrar la ecuación característica que modela la planta, no de forma exacta pero da una aproximación cercana, esto se hace mediante la ayuda de un programa, en este caso matlab. este programa es usado para para sintonización o modelados de plantas entre muchas mas funciones y aplicaciones.La herramienta que se utilizara de este programa sera el **indent** para encontrar la función de transferencia de la planta.
para realizar la estimación se cargan los datos de excel al workspace de matlab, en la ventada de commadnd windows se escribe ident, el despliega una ventana, donde se proceden a subir los datos y realizar la estimación de la función de transferencia en tiempo discreto para realizar una simulación mas aproximada a la aplicación real, por el motivo que se opera con un microcontrolador el solo procesa señales discretas, los datos tomados y la estimaciones echas se pueden ver en la figura 5.32

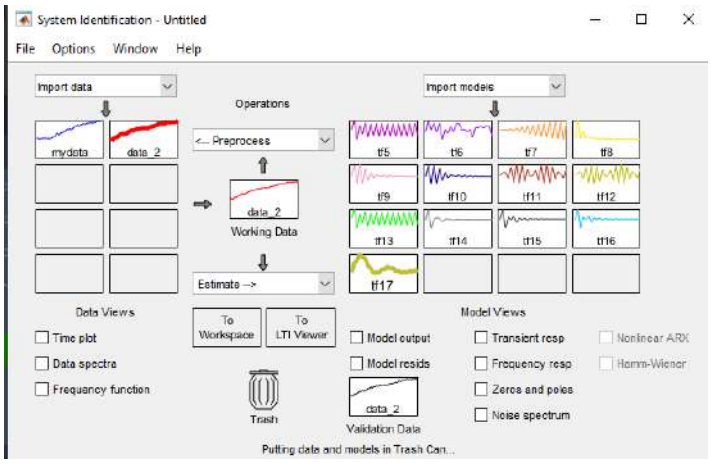


Figura 5.32: Estimación del Modelo
Fuente: AUTOR

se exportan los datos de la ventana al workspace, dando la función de transferencia en

tiempo discreto vista en la ecuación 5.6

$$G(z) = \frac{0.001422z^3 - 0.001302z^2}{1z^4 - 0.1691z^3 - 0.1838z^2 - 0.3192z - 0.3207} \tag{5.6}$$

5.4.2. Planta Velocidad

En el acondicionamiento de la planta de velocidad, se emplea un encoder con una resolución de 360 pulsos por vuelta,siendo el dispositivo que convierte el movimiento rotativo en pulsos eléctricos, y son codificados por el sistema embebido.
En el control de velocidad se asignaron dos puertos definidos por código,para la lectura del encoder, el puerto de entrada 1 y el puerto de entrada 2. por el motivo de que se están utilizando interrupciones externas y esos puertos deben ser configurados antes de ejecutarse el código principal.
cuando se tiene el valor de rpm registrado en el motor, es computado por el algoritmo de control enviando una señal de salida por algún puerto del sistema embebido que se configuro para emitir la señal en el rango de 0-10V. como esta señal va ser codificada por un arduino, de igual manera como el control de temperatura este lleva un reductor de voltaje, para ser leída esta señal por un arduino, que solo cumple con la tarea de adquirir esta señal y reflejarla en modulación de ancho de pulso, siendo enviada a la etapa de potencia y controlado la velocidad de esta manera del motor eléctrico, en la figura 5.33 se aprecia el lazo de control con los dispositivo empleados.

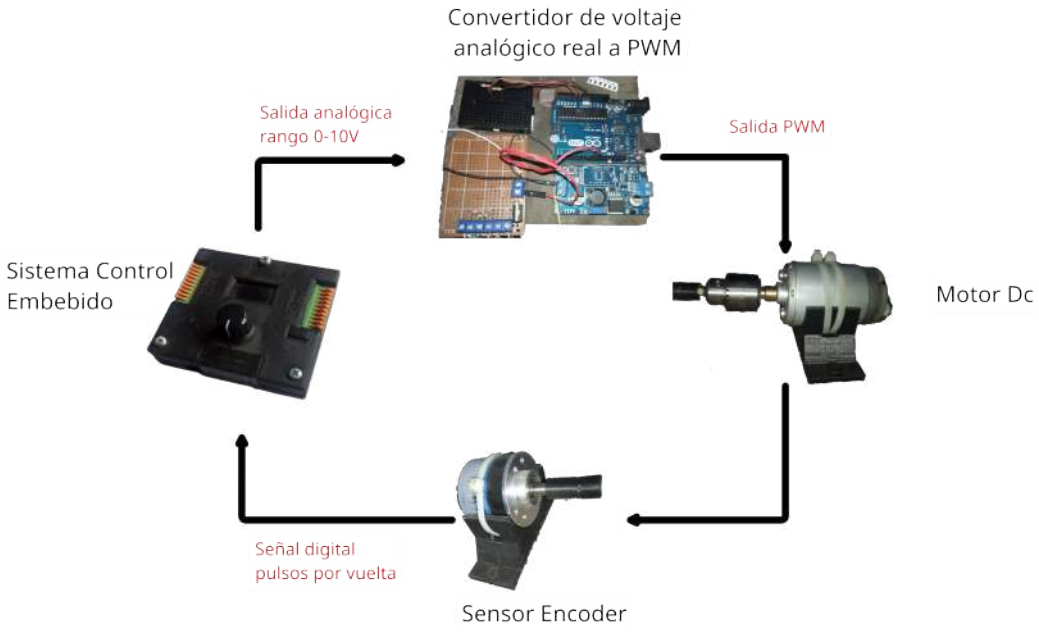


Figura 5.33: Lazo Control Planta Velocidad
Fuente: AUTOR

El montaje real de la planta de control de velocidad, se puede apreciar en la figura tal

Figura 5.34: Lazo Control Planta Velocidad
Fuente: AUTOR

5.4.2.1. Identificación Del Modelo De Planta

La identificación del modelo de la planta de velocidad, se realizo el mismo procedimiento para encontrar el modelo de la planta de temperatura. para tener el comportamiento de la planta se hizo una identificación por switch, queriendo decir que se habilitaba un alto y un ajo en periodos de tiempo y tener el comportamiento de la planta como se puede observar en la figura 5.35

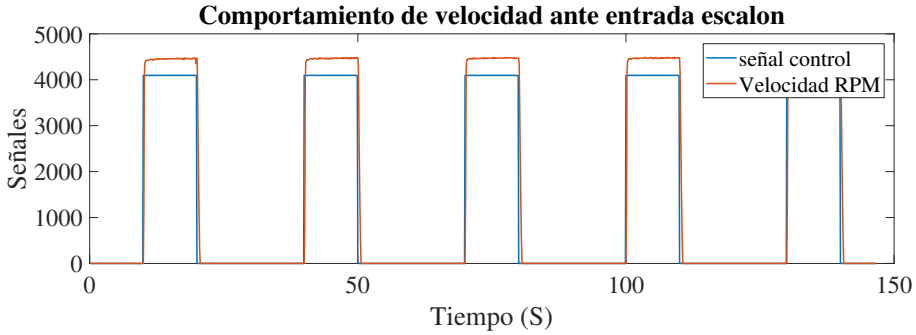


Figura 5.35: control a lazo abierto planta temperatura ante una entrada
Fuente: AUTOR

en la gráfica se ven varias toma de la velocidad máxima alcanzada a una resolución de 4095 que permite la salida del sistema embebido, queriendo decir que la señal de control es de 10V a su salida, en varios intervalos de tiempo, tomando el comportamiento de la planta cuando se le da un alto, se puede tener el comportamiento de la planta cuando se le da un bajo a la salida del sistema embebido,se utiliza la herramienta **ident** de matlab para encontrar la

aproximación de modelo de la planta en tiempo discreto tal y como se ve en la ecuación 5.7.

$$G(z) = \frac{0.3976}{z - 0.6395} \tag{5.7}$$

5.4.3. Plante Nivel

En el acondicionamiento de la planta de nivel, el elemento que da la posición del nivel de agua, es el sensor sharp, este sensor para el modelo de esta planta esta restringido hasta cierto nivel, máximo 15 cm, como el sensor sharp calcula la distancia con un voltaje de salida analógico y siendo codificado por el sistema embebido para tener la posición actual, para llenar el recipiente, se encuentra una bomba sumergible, que es la que por modulación de ancho de pulso, se regulara el caudal de agua entrante al recipiente, además cuenta con una llave de paso generando un flujo continuo a la salida, y poder generar una perturbación, la llave de paso puede ser regulado el caudal de salida, el diagrama del lazo de control de la planta se puede ver en la figura 5.36, con los elementos que se emplearan en su uso.

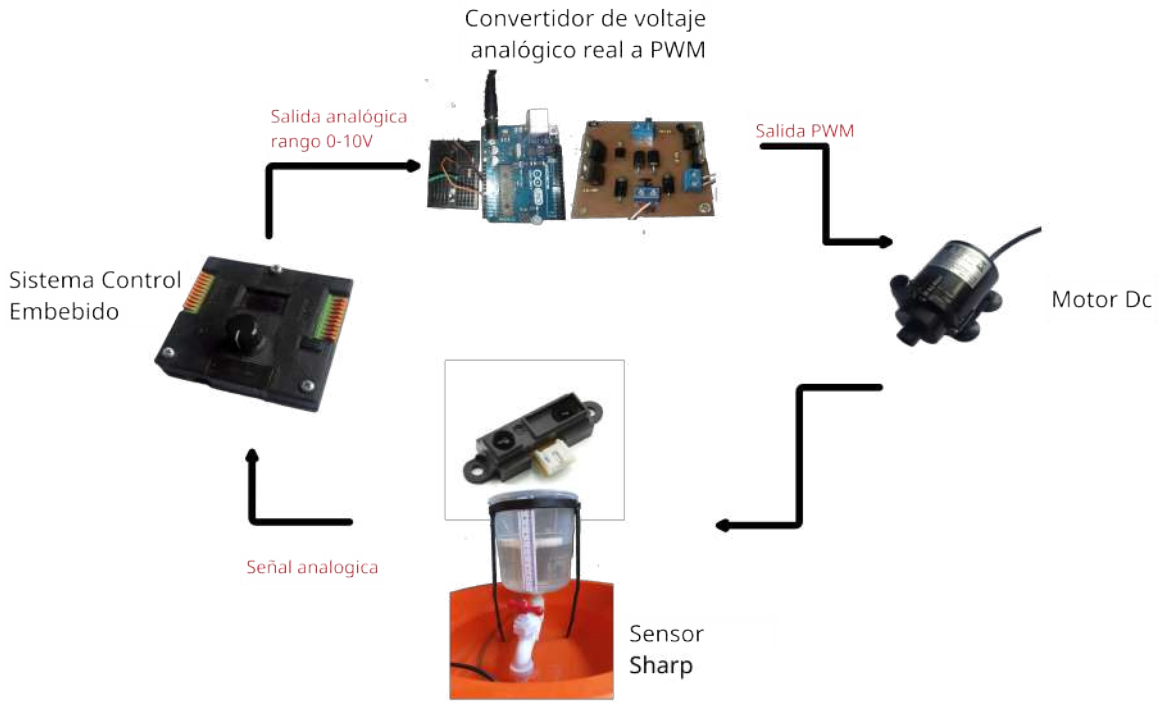


Figura 5.36: Lazo de control planta Nivel

Fuente: AUTOR

en la figura 5.37 se puede visualizar la ubicación del sensor sharp en la parte superior del recipiente, tambien tiene una cinta métrica para medir el nivel de agua en relación al llenado, incorpora un tapa de icopor de un espesor de 1 cm, para que este sea el medio que la luz emitida por el sensor sharp pueda ser rebotada y recibida, cuenta con una llave de paso regulable a su salida, e ingresando el liquido por la parte superior del recipiente, se puede apreciar el sistema embebido, capturando la señal del sensor sharp, y conectado a la etapa de potencia correspondiente a controlar el flujo de agua de la bomba sumergible.

Figura 5.37: Montaje de la planta de control de NIvel
Fuente: AUTOR

5.4.3.1. Identificación Del Modelo De Planta

En la identificación del modelo de planta de nivel, se mantienen los mismos pasos explicados en la planta de temperatura, en esta planta se tomaron datos del comportamiento de la planta ante una entrada escalón a lazo abierto, así como se observa en la figura 5.38

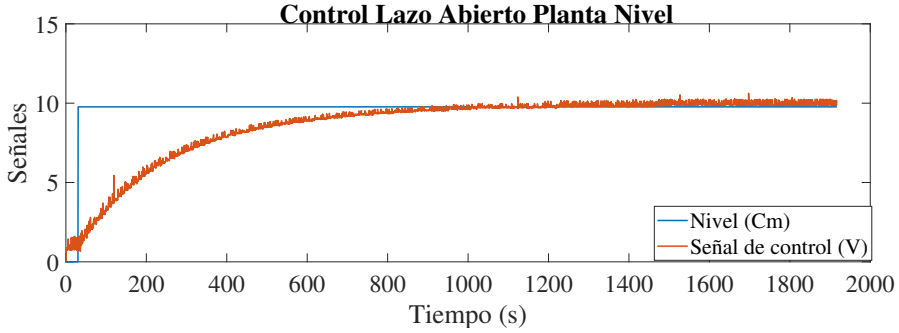


Figura 5.38: Comportamiento de la planta de nivel ante una entrada escalón,
Fuente: AUTOR

para calcular la aproximación del modelo de la planta , se pasan estos datos al identificador de funciones de la herramienta de matlab, generando la función de transferencia vista en la ecuación 5.8

$$G(z) = \frac{1.151e - 5z}{z^2 - 0.001164z - 0.9837} \tag{5.8}$$

5.5. Controlador Clásico

en el cumplimiento del objetivo de validación del controlador neuronal primero se debe tener referencia de respuestas de las plantas ante un controlador ya conocido y poder realizar una comparación así de esta manera validar su funcionamiento ante un controlador clásico. por ello se hallan las constantes del PID o controlador clásico derivados del PID, para cada planta según su modelo en función de transferencia.
se vuelve hacer ayuda de la herramienta matlab, esta vez se utilizara la toolbox simulink y la función de auto tuner para la sintonización del controlador.

5.5.1. Sintonización del Controlador Planta temperatura

La sintonización del controlador clásico para la planta de temperatura, sera un controlador PI, ya que la planta no tiene cambios tan abruptos en periodo de tiempos cortos, el método de sintonización se utilizara ayuda de la herramienta del PID tuner de matlab y facilitar el proceso, sin embargo para poder realizar este procedimiento era necesario tener la función de transferencia de nuestra planta y se utiliza un lazo de control cerrado tal como se ve en la figura 5.39

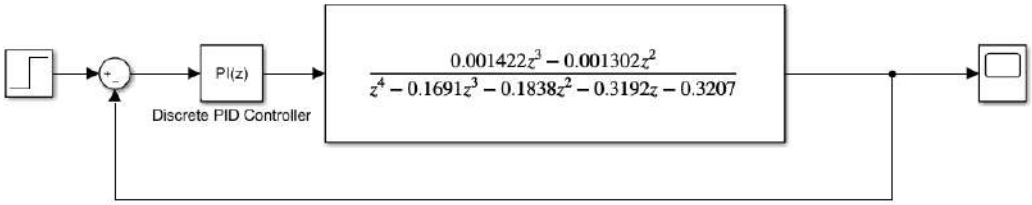


Figura 5.39: Lazo control PI Temperatura
Fuente: AUTOR

una vez sintonizado del controlador PI se pueden extraer del siendo los siguientes valores correspondientes a cada constante:

- Kp=295.4249
- Ki=25.6966

para comprobar el funcionamiento del controlador en simulación de la planta de temperatura, se le ingresan valores de setpoint en intervalos de tiempo de 1000 segundo y demostrar su control en simulación, en la figura 5.40 se aprecia el lazo de control con varios valores de setpoint's, y en la figura 5.41 se se visualiza el comportamiento de la respuesta generada por la planta con esas constantes del control PI.

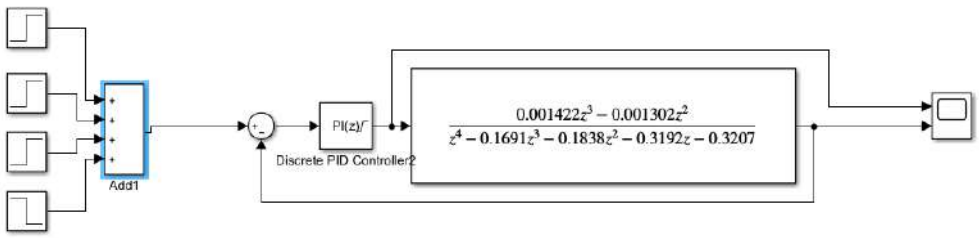


Figura 5.40: Lazo control PI Temperatura ante varias entradas en diferente intervalos de tiempo
Fuente: AUTOR

en la figura 5.41, se visualiza el comportamiento la respuesta de salida de la planta, permitiendo apreciar que la dinámica es lenta en el cambio de la variable controlada, esto debido a que puede tener una constante Kp baja y ki bajas, aunque se hicieron varias pruebas para obtener mejores respuestas del controlado sin generar sobre picos o oscilaciones en la respuesta, siendo estos los valores de constantes que generaron una mejor salida del sistema.

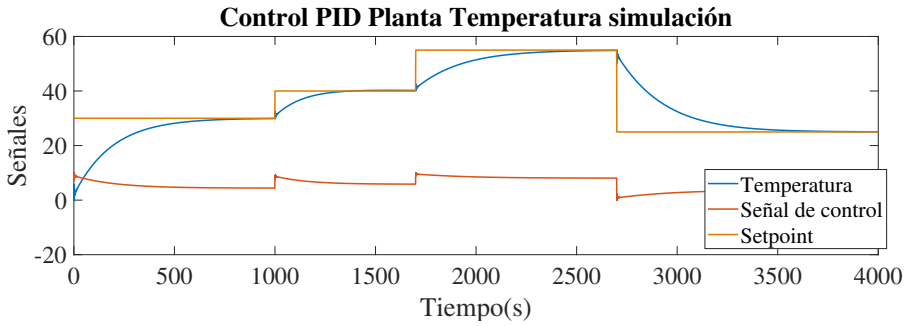


Figura 5.41: Respuesta del control de la planta temperatura ante varios valores entradas
Fuente: AUTOR

5.5.2. Sintonización del Controlador de la Planta Velocidad

La sintonización del controlador clásico de la planta de control de velocidad, se establece un lazo de control con realimentación ver figura 5.42. utilizando la función de transferencia aproximada calculada en la ecuación 5.7, y vista en el diagrama de bloques generado en la herramienta simulink de matlab, para la sintonización se uso la toolbox de PID tuner para obtener unas constates del PID aproximadas y ser aplicadas en el sistema embebido.

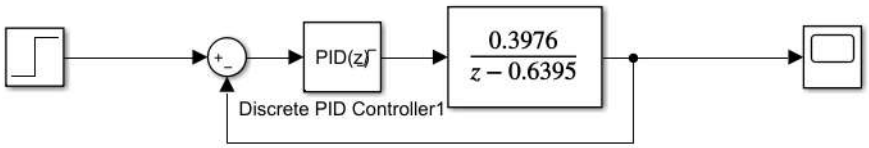


Figura 5.42: Lazo control PID Velocidad
Fuente: AUTOR

las constantes del PID que mostraron mejor respuesta, por la herramienta de PID Tuner son las siguientes :

- Kp= 0.04966
- Ki= =0.19933
- Kd= 0.0000458

para comprobar la respuesta de salida obtenida por las constantes de PID, en la figura 5.43 se ingresaron varios bloques de step activándose en intervalos diferentes de tiempo y así poder ver la eficacia del controlador y ver el comportamiento de la planta ante diferentes valores de Setpoint.

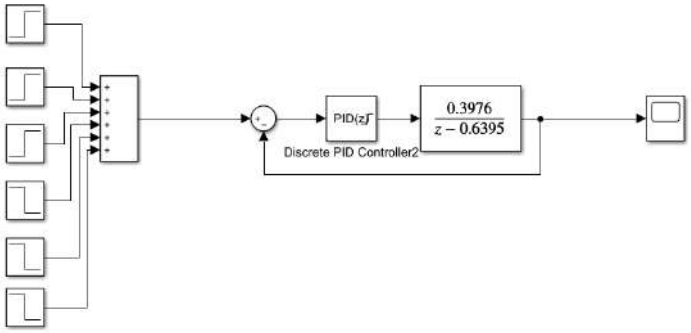


Figura 5.43: Lazo control PID Velocidad ante varias entradas escalón
Fuente: AUTOR

En la figura 5.44 se observa el comportamiento de la señal de salida ante diversa magnitud de entradas, demostrando ser un buen control en la simulación, sin llegar a tener sobre-picos en la señal de salida, la señal de control se escalo en una relación de 400:1, para poder apreciar las dos señales en una misma gráfica, debido a los valores de rpm elevados.

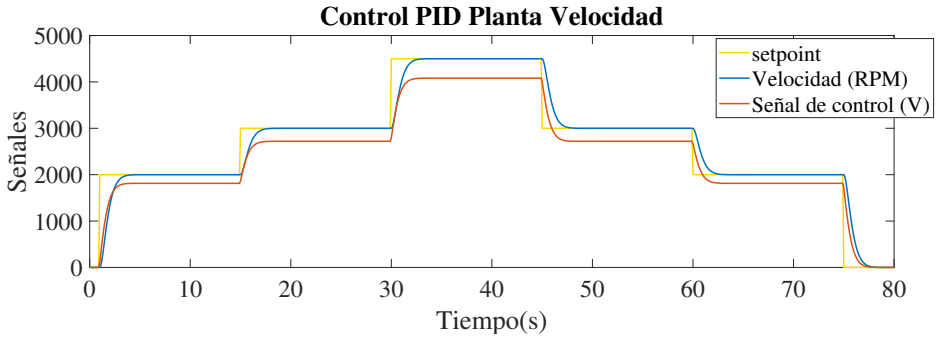


Figura 5.44: Respuesta del control PID de la planta Velocidad ante diferentes entradas
Fuente: AUTOR

5.5.3. Sintonización del Controlador de la Planta Nivel

la sintonización del controlador clásico para la planta de nivel, se utiliza un lazo de control con realimentación, con un control clásico PI, ver figura 5.45 .haciendo uso la función de transferencia calculada anteriormente correspondiente a esta planta.

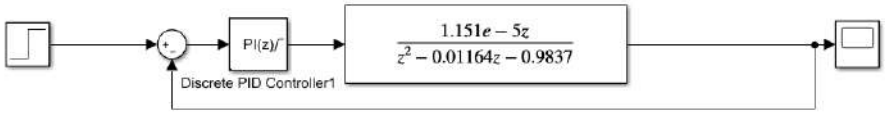


Figura 5.45: Lazo control PI Velocidad entrada aleatoria
Fuente: AUTOR

con la ayuda de la herramienta de matlab se sintoniza el controlador PI generando las siguientes constantes:

- Kp=990.928
- Ki=400.56

para comprobar la eficacia del controlador se sometió a pruebas bajo simulación. en setpoints de diferentes valores, usando varios bloques de setpoint's en intervalos de tiempos distintos , la respuesta de salida puede apreciarse en la figura 5.46, se puede apreciar que el comportamiento del nivel requerido llega al valor establecido, aunque en tiempos algo extensos, siendo esto algo natural de la planta, ya que esta tiene una dinámica lenta en el cambio de su variable controlada.

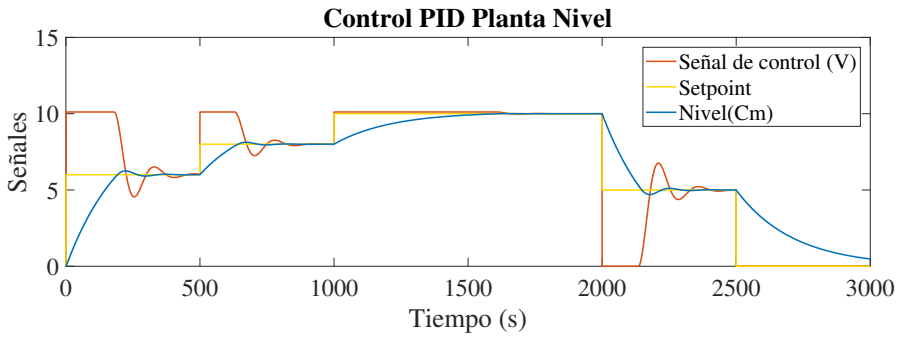


Figura 5.46: Respuesta del control de la planta temperatura varios Setpoint's
Fuente: AUTOR

5.6. Implementación Del Algoritmo Neuronal En Microcontroladores

En la implementación del algoritmo neuronal en microcontroladores, se desarrollaron dos librerías compatibles con la tarjetas de desarrollo arduino en código c,se utilizo de guia la lógica de programación vista en la metodología en la tabla 4.1, para poder generar los algoritmos respectivos que conllevan la implementación de una red neuronal en código. las librerías creadas fueron **Dynamic_Array**, que consiste en crear arreglos unidimensional de n posiciones, arreglos bidimensional de n numero de filas,columnas y arreglos tridimensionales de n filas, columnas y profundidad, forma dinámica, queriendo decir, que estos arreglos pueden cambiar su tamaño de filas, columna o profundidad en cualquier momento, sin tener que generar un arreglo al inicio de la ejecución del código, que debe estar ya definido sus tamaños, siendo denominadas como variables estáticas, no pudiendo cambiar su forma o tamaño en la ejecución de código a libertad. la segunda librería creada llamada **Neural_Networks_**FF, en esta librería contendrá todo el algoritmo que corresponde al red neuronal, desde la creación de la topología, entrenamiento de la red, y la evaluación para así obtener una salida deseada.

5.6.1. Uso De Librería Dynamic_Array y Ejemplo

la realizo esta librería en el entorno de arduino para el sistema embebido, debido a la necesidad de generar arreglos que puedan alterar su tamaño en el cualquier linea de ejecución del código,los arreglos dinámicos permitan alterar su tamaño, siendo variables de vital importancia para implementación de algoritmos neuronales en microcontroladores, por lo que es importante el saber que es datos estática, datos dinámica y arrays dinámicos.

Datos estática su tamaño y forma es constante durante la ejecución de un programa y por tanto se determinan en tiempo de compilación. El ejemplo t´ıpico son los arrays. Tienen el problema de que hay que dimensionar la estructura de antemano, lo que puede conllevar desperdicio o falta de memoria.

Datos Dinámica: su tamaño y forma es variable (o puede ser lo) a lo largo de un programa, por lo que se crean y destruyen en tiempo de ejecución. Esto permite dimensionar la estructura de datos de una forma precisa: se va asignando memoria en tiempo de ejecución según se va necesitando. (P. López, 2006)
Cuando el sistema operativo carga un programa para ejecutarlo y lo convierte en proceso, le asigna cuatro partes lógicas en memoria principal: texto, datos (estáticos), pila y una zona libre. Esta zona libre (o heap) es la que va a contener los datos dinámicos, la cual, a su vez, en cada instante de la ejecución tendrá partes asignadas a los mismos y partes libres que fragmentaran esta zona, siendo posible que se agote si no se liberan las partes utilizadas ya inservibles.(P. López, 2006)
(La pila también varıa su tamaño dinámicamente, pero la gestiona el sistema operativo, no el programador): Para trabajar con datos dinámicos necesitamos dos cosas:

1. Subprogramas predefinidos en el lenguaje que nos permitan gestionar la memoria de forma dinámica (asignación y liberación).
2. Algún tipo de dato con el que podamos acceder a esos datos dinámicos (ya que con los tipos vistos hasta ahora solo podemos acceder a datos con un tamaño y forma ya determinados).

Arreglo dinámico: En programación, un array dinámico, más apropiadamente llamado arreglo dinámico, también llamado inapropiadamente matriz dinámica o tabla dinámica, es un arreglo de elementos que crece o mengua dinámicamente conforme los elementos se agregan o se eliminan. Se suministra como librerías estándar en muchos lenguajes modernos de programación. Un arreglo dinámico no es lo mismo que un arreglo asignado dinámicamente, que es un arreglo de tamaño fijo, pero cuyo tamaño se fija cuando se asigna por primera vez. Sirve exclusivamente para realizar arreglos que no tienen una definición fija por ende su nombre. (Dinámico).(PAÚL, 2013)

en tabla 5.10 se encuentran funciones que fueron realizadas, a partir de la teoría de los arreglos dinámicos, para el uso correcto de la librería Neural_Networks_FF

N°	Funciones	Descripción
1	var.Dynamic_Array()	Crea el constructor o variable asociado como arreglo dinámico, no recibe ningún parámetro
2	var.NewArray_1D(tamaño)	Crea el tamaño del arreglo dinámico unidimensional, en variable tipo float
3	var.IntNewArray_1D(tamaño)	Crea el tamaño del arreglo dinámico unidimensional, en variable tipo int
4	var.CharNewArray_1D(Tamaño)	Crea el tamaño del arreglo dinámico unidimensional, en variable tipo char
5	var.NewArray_2D(filas,Columnas)	Crea el arreglo dinámico bidimensional, con n numero de filas y n numero de columnas
6	var.NewArray_3D(fila,columna, profundidad)	Crea el arreglo dinámico bidimensional, con n numero de filas ,n numero de columna y n numero de profundidad.
7	var.DeleteArray_1D()	Elimina el arreglo dinámico de una dimensión asociado a la variable.
8	var.IntDeleteArray_1D()	Elimina el arreglo dinámico de una dimensión asociado a la variable tipo int
9	var.CharDeleteArray_1D()	Elimina el arreglo dinámico de una dimensión asociado a la variable tipo char
10	var.DeleteArray_2D()	Elimina el arreglo dinámico bidimensional asociado a la variable
11	var.DeleteArray_3D()	Elimina el arreglo dinámico tridimensional asociado a la variable
12	var.WriteArray_1D (posición,valor)	Escribe un valor en la posición establecida.
13	var.IntWriteArray_1D (posición,valor)	Escribe un valor en la posición establecida.
14	var.CharWriteArray_1D (posición,valor)	Escribe una cadena de caracteres en la posición establecida.
15	var.WriteArray_2D (fila,columna,valor)	Escribe el valor en la posición asociado al numero indicado en la fila y columna.
16	var.WriteArray_3D (fila,columna, profundidad,valor)	Escribe el valor en la posición asociado al numero indicado en la fila,columna,profundidad.

Tabla 5.10: Funciones de librería Dynamic_Array 1
Fuente: AUTOR

otras funciones importantes que componen la librería son las vistas en la tabla 5.11

N°	Funciones	Descripción
17	var.ReadArray_1D(posición)	Lee el parámetro asociado a la posición.
18	var.CharReadArray_1D(posición)	Lee el parámetro asociado a la posición.
19	var.ReadArray_2D(fila,columna)	Lee el parámetro asociado a la posición de fila y columna.
20	var.ReadArray_3D(fila,columna, profundidad)	Lee el parámetro asociado a la posición de fila, columna y profundidad.
21	var.GetRows()	Lee numero de filas del arreglo dinámico.
22	var.GetColumns()	Lee numero de columnas del arreglo dinámico
23	var.GetDepth()	Lee numero de Profundidad del arreglo dinámico.
24	var.PrintArray_1D()	Imprime en el puerto serial el arreglo dinámico de 1 dimensión float.
25	var.IntPrintArray_1D()	Imprime en el puerto serial el arreglo dinámico de 1 dimensión int.
26	var.CharPrintArray_1D()	Imprime en el puerto serial el arreglo dinámico de 1 dimensión char.
27	var.PrintArray_2D()	Imprime en el puerto serial el arreglo dinámico bidimensional.
28	var.PrintArray_3D()	Imprime en el puerto serial el arreglo dinámico tridimensional
29	freeRam ()	calcula el valor disponible de memoria SRAM del microcontrolador.
30	var.MaxArray_1D()	Calcula el valor máximo en el arreglo.
31	var.IntMaxArray_1D()	Calcula el valor máximo en el arreglo.
32	var.MaxArray_2D()	Calcula el valor máximo en el arreglo.
33	var.ToinitializeArray_1D(valor)	Inicializa todo el arreglo dinámico con un valor.
34	var.IntToinitializeArray_1D(valor)	Inicializa todo el arreglo dinámico con un valor.
35	var.CharToinitializeArray_1D(valor)	Inicializa todo el arreglo dinámico con un con una cadena de caracteres.
36	var.ToinitializeArray_2D(valor)	Inicializa todo el arreglo dinámico con un valor.
37	var.ToinitializeArray_3D(valor)	Inicializa todo el arreglo dinámico con un valor.

Tabla 5.11: Funciones de librería Dynamic_Array 2
Fuente: AUTOR

5.6.1.1. Ejemplo del uso Librería

cuando la librería es cargada al entorno de desarrollo de arduino. En la sección de ejemplos viene con 3 ejemplos, de como se le hace llamada, lectura, escritura y eliminación de un

arreglo dinámico, unidimensional, bidimensional y tridimensional, en el código 5.1,se aprecia el uso de la librería Dynamic_Array con el ejemplo de arreglo tridimensional, explicando una de las funciones importantes utilizadas en esta librería.

```
1  //Implementacion de un array dinamico 3D en arduino
2  //Incluimos la libreria desarrollada
3  #include<Dynamic_Array.h>
4  //Creamos el constructor, este no recibe ningun parametro
5  // variable asociada como arreglo dinamico, en el ejemplo es W
6  Dynamic_Array W=Dynamic_Array();
7  void setup() {
8    Serial.begin(9600);
9    //Cosultamos la capacidad de memoria SRAM Disponible en bytes
10   // se consulta la capacida de memoria SRAM, para asegurar que exista espacio
11   // en el microcontrolador para genererar la variable dinamica
12    int cap=W.freeRam();
13    //Imprimimos la capacidad de memoria SRAM Disponible en bytes
14    Serial.print("Capacidad de memoria SRAM Disponible: ");
15    Serial.println(cap);
16    //Creamos un nuevo arreglo tridimensional 4 filas,5 columnas, 7 en profundidad
17    // la funcion "variable.NewArray_3D(x1,x2,x3)" se usa para crear un arreglo  tridimensional con ↩
        ↪ las
18    // especificaciones que se le tengan que dar en su tamaño.
19    // w.NewArray_3D(filas,Columnas,Profundidad).
20    W.NewArray_3D(4,5,7);
21    //Cosultamos nuevamente la capacidad de memoria disponble
22    //para calcular la memoria consumida por el arreglo tridimensional creado
23    int cap_array=cap-W.freeRam();
24    //Imprimimos capacidad de memoria consumida por el array
25    Serial.print("Capacidad de memoria consumida por el array: ");
26    Serial.println(cap_array);
27    // Escribimos en el array creado
28    // arduino no cuenta con funciones que calculen el tamaño de
29    // filas, columnas o profundidad de un vector, por ello se cuenta
30    // con la funcion "variable.GetDepth()" que calcula tamaño de produndidad del arreglo dinamico.
31    // "variable.GetColumn()" calcula el tamaño de columnas de arreglo dinamico.
32    // "variable.GetRows()" calcula el tamaño de filas del arreglo dinamico.
33    // con la funcion "variabble.WriteArray_3D(fila, columna, profunidad, valor_escribir)"
34    // se escribe el valor en el arreglo dinamico en una posicion definida.
35   for (double k = 0; k < W.GetDepth(); k++)
36   {
37     for (double i = 0; i < W.GetColumns(); i++)
38        {
39          for (double j = 0; j < W.GetRows(); j++)
40            {
41              W.WriteArray_3D(j,i,k,5);
42            }
43        }
44   }
45   //Leemos e imprimimos el dato guardado en
46   //fila 1 columna 1 profundidad 1 del array
47   // con la funcion "variable.ReadArray_3D(fila,columna,profudidad)"
48   // se lee el valor de la posicion del arreglo dinamico.
49    Serial.print("Dato guardado en fila 0 columna 0 profundidad 0 del array:  ");
50    Serial.println(W.ReadArray_3D(0,0,0));
```

```
51    //Imprimimos el array por puerto serial
52     Serial.println("Datos del array:  ");
53     //W.PrintArray_3D();
54     //Leemos e imprimimos el numero de filas del array
55     Serial.print("Numero de filas del array:  ");
56     Serial.println(W.GetRows());
57     //Leemos e imprimimos el numero de columnas del array
58     Serial.print("Numero de columnas del array:  ");
59     Serial.println(W.GetColumns());
60     //Leemos e imprimimos la profundidad del array
61     //(para array de 3D)
62     Serial.print("Profundidad del array:  ");
63     Serial.println(W.GetDepth());
64     //eliminamos el arrray para liberar memoria
65     W.DeleteArray_3D();
66     //Cosultamos e imprimimos la capacidad de memoria SRAM Disponible en bytes
67     //Para verificar que la memoria se libero correctamente
68     Serial.print("Memoria SRAM Disponible despues de eliminar el array :");
69     Serial.println(W.freeRam());
70   }
71
72 void loop() {
73 }
```

Código 5.1: Ejemplo arreglo dinamico tridimencional

5.6.2. Uso de la librería Neural_Networks_FF y ejemplos

La librería de Neural_Networks_FF desarrollada, sera el elemento principal para la implementación de algoritmos neuronales en un sistema embebido, para poder hacer uso de esta librería era importante el poder declarar arreglos de varios tamaños a disposición, siendo esto suplido por la librería de Dynamic_Array. La librería Neural_Networks_FF cuenta con una gran versatilidad para adaptarse a varias topologías que se le requiera, y que el microcontrolador soporte en su capacidad de memoria SRAM, además permite seleccionar las funciones de activación por capas de la red neuronal, variar el valor de aprendizaje a disposición.
la librería cuenta con dos modos de entrenamiento de la red, la función de **TRAIN_NET** que permite realizar el entrenamiento de la red neuronal por una entrada de datos de patrones y así calcular los pesos, para realizar una clasificación, aproximación de funciones entre mas aplicaciones que se le puedan dar.
el segundo modo el cual sera el aplicada a al sistema embebido es la función abordada por la librería **TRAIN_NET_ONLINE** permitiendo esta función realizar un entrenamiento, calculo de pesos, calculo de error, y generando en valor de salida en linea con el procesos, operando así de este modo como controlador adaptativo neuronal.
en la tabla 5.12 se enunciaran las funciones correspondientes a la librería creada de Neural_Networks_FF explicando, que realiza la funcion, el llamado de dicha función en el entorno de desarrollo.

5.6.2.1. Ejemplo Uso de la librería

Para un mejor entendimiento de como implementar la librería, como una red neuronal de entrenamiento de patrones o una red neuronal como controlador, se se darán un ejemplo

de cada uno, en el código 5.6, se puede apreciar la aplicación de la red como aproximador de función que incorpora de ejemplo esta librería, y el código 5.7 se utiliza la red como controlador.

N°	Funciones	Descripción
1	Neural_Networks_FF()	Constructor de la red neuronal
2	var.DATA_CENTERING (Arreglo_dinámico)	Realiza un centrado de los datos, para que la media de cada entrada sea cercana a 0
3	var.NORMALIZE_DATA (Arreglo_dinámico,Valor_máximo)	Normaliza los datos de entrada, en valores entre 0-1 por el valor máximo de los datos.
4	var.FEED_FORWARD_NET (Topología,funciones_activación)	Crea la topología de la Red, con funciones de activación
5	var.TRAIN_NET (Entradas,Salidas,Error_mínimo, Numero_de_iteraciones)	Realiza el proceso de entrenamiento de la red neuronal, con error minino establecido o numero de iteraciones para terminar el proceso de entrenamiento.
6	var.TRAIN_NET_ONLINE (Entradas,Salidas,lectura)	Realiza el proceso de entrenamiento de la red neuronal en linea, en las entradas los errores en n intervalos de tiempo, la salida deseada o setpoint, la lectura del valor del estado de la variable a controlar
7	var.SIM_NET(Entradas)	Evalúa la Red, asociado al numero de patrones de entrada.
8	logsig(valor)	evalúa el valor en la Función de activación logisg
9	tansig(valor)	evalúa el valor en la Función de activación tanisg
10	purelin(valor)	evalúa el valor en la Función de activación purelin
11	hardlim(Valor)	evalúa el valor en la Función de activación hardlim
12	poslin_lim (valor,Valor máximo, valor mínimo)	evalúa la función poslin_lim, esta función es similar a la función pureline, solo que con el valor de entrada, restringe la salida entre el valor máximo y el valor mínimo
13	dposlin_lim (valor,valor máximo, valor mínimo)	evalúa la derivada de la función poslin limitada entre el valor máximo y mínimo
14	dlogsig(valor)	evalúa el valor en la derivada de la función de activación logsig.

Tabla 5.12: Funciones de librería Neural_Networks_FF

Fuente: AUTOR

en la tabla 5.13 se podrán ver las otras funciones y variables pertenecientes a esta librería desarrollada.

N°	Funciones	Descripción
15	dtansig(Valor)	evalúa el valor en la derivada de la función de activación tansig.
16	var.LearningRat=valor	Asigna un valor de taza de aprendizaje de la red neuronal.
17	var.Ysim	Permite consultar las salidas de red en la ultima capa.
18	var.EPOCHS	Consulta el numero de epocas realizadas en el entrenamiento
19	var.ERROR_C_M	Consulta el valor del error cuadrático medio.
20	var.W	Consulta el valor de los pesos.
21	var.E	Consulta el valor de los Error global.
22	var.y	consulta el valor de salida de la red neuronal online.

Tabla 5.13: Funciones de librería Neural_Networks_FF
Fuente: AUTOR

se tomara la funcion var.FEED_FORWARD_NET(Topología,funciones_activación) numero 5 de la tabla, para explicar el modo correcto, el como introducir los parámetros. según la función, recibe dos parámetros topología y función de activación, los parámetros que recibe solo pueden ser introducidos en arreglo dinámicos, para una mejor comprensión se usara la topología de una red neuronal vista en la figura 5.47

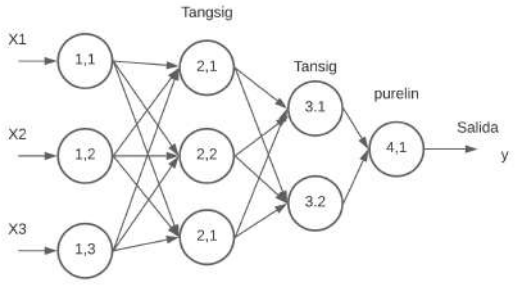


Figura 5.47: Topología de ejemplo para uso de la función
Fuente: AUTOR

en la figura 5.47, se tiene una red neuronal con la siguiente características de la topología; contando con las neuronas de entrada, son 4 capas en totales, contando la primera capa, se tiene que esta compuesta de 3 neuronas de entrada, la segunda capa o primer capa oculta cuenta con 3 neuronas, la segunda capa oculta cuenta 2 neuronas, y por ultima la 4 capa,

con una neurona de salida, cabe aclarar que los pesos asociados se ubicarían en cada flecha que sale de una neurona y entra a la siguiente.
La función FEED_FORWARD_NET va asociada la librería Neural_Networks_FF, como se programa en el entorno de desarrollo de arduino, se explicara el uso y llamado de variables para implementar del modo correcto la librería.
en el algoritmo 5.2,se mencionaran las primeras lineas de código que deben ir antes del void setup y el void loop, esta librería debe estar previamente guardada en el gestor de liberáis de IDE de arduino en conjunto con la librería Dynamic_Array

```
1  // se incluye la libreria.
2 #include<Neural_Networks_FF.h>
3 //Contructor de la red neuronal denominada net
4 Neural_Networks_FF net=Neural_Networks_FF();
5 // se hace uso de las funciones de arrys dinamicos para crear el
6 //Array dinamico para ingresar las entradas a la red
7 Dynamic_Array XI=Dynamic_Array();
8 //Array dinamico para ingresar las salidas deseadas
9 Dynamic_Array D=Dynamic_Array();
10 //Array dinamico tipo string para ingresar las funciones
11 //de activacion por capa
12 Dynamic_Array FAC_STR=Dynamic_Array();
13 //Array dinamico para ingresar la estructura de la red
14 //como vector
15 Dynamic_Array EST=Dynamic_Array();
16
17 void setup(){}
18 void loop() {
19 }
```

Código 5.2: Librería y variables globales para uso de la librería Neural_Networks_FF

e ejd una vez se inicializaron los parámetros principales.primero de deben ingresar en el formato de arreglos dinámicos, esto se realiza en el void setup ejecutándose una sola vez y de este modo solo reservar la memoria dinámica para las variables una sola vez, en el código se aprecia el como se hace el llamado de los parámetros de entrada y salida, a modo de ejemplo de dira que se trabaja con solo patron de entrada donde x1=1, x2=0, X3=1 y la salida y =1.

```
1 void setup(){
2 // XI representara las e primeras neuronas en la primera capa, como neuronas de entrada.
3 // primero se crea un arreglo dinamico de 1 fila, 3 columnas
4 int n=1;   // n representa el numero de patrones a ingresar a la red
5 XI.NewArray_2D(n,3);
6 // se inicializa todo el arreglo de entrada
7 // en valor de 0.0 en todas las posiciones
8 XI.ToinitializeArray_2D(0.0);
9 //Crear e inicializar el array de salidas deseadas, como en la topología
10 //se aprecia solo una neurona, queda un arreglo 1x1
11 D.NewArray_2D(1,1);
12 // se inicializa en 0
13 D.ToinitializeArray_2D(0.0);
14 //Por uso de la funcion get.columns
15 // se consulta el valor de columnas del
16 // arreglo de entrada, esto de modo que si se cambia la toplogia de
17 // entrada, no se tenga que estar cambiando parámetros.
```

```
18 byte XC=XI.GetColumns();
19 //Se consulta el numero de filas del array
20 // de salida.
21 byte DF=D.GetRows();
22 // se le tiene que dar un pratron o valor a las entradas
23 //en este ejemplo se dira que
24 //entrada consta para X1=1,X2=0,X3=1 y salida D=1; con un solo patron de entrada
25 // por lo que se procede a llenar los arreglos dinámicos de entrada y salida.
26  XI.WriteArray_2D(0,0,1); // para X1
27  XI.WriteArray_2D(0,1,0); // para X2
28  XI.WriteArray_2D(0,2,1); // para X3
29  D.WriteArray_2D(0,0,1)  // para el arreglo de salida o patrón de salida.
30 .
31 .
32 .
```

Código 5.3: Configuración de Parámetros de entrada y salida para crear la red neuronal con la libreria Neural_Networks_FF

En el codigo 5.3, se tiene el patrón de entradas, pasados al formato de arreglo dinámico. en el código 5.4 se enuncia como se crea la estructura de la red, tomando el valor de entradas salida, numero de neuronas por cada.

```
1 .
2 .
3 .
4
5 // se hace uso se la comando para encontrar el numero de entradas de forma dinámica y guardar en ↩
      ↪ una variable XC
6 byte XC=XI.GetColumns();
7 // se hace uso se la comando para encontrar el numero de salidas de forma dinámica y guardar en ↩
      ↪ una variable DF
8 byte DF=D.GetRows();
9 // esta función realiza el centrado de los datos y lo reescribe en la arreglo dinámico de entrada
10 net.DATA_CENTERING(XI);
11 // la funcion Normalizado de datos encontrando el valor máximo en el arreglo dinámico
12 net.NORMALIZE_DATA(XI,XI.MaxArray_2D());
13
14 //L representara el numero de capas, la topología consta de 4 capas.
15 byte L=4;
16 // Creamos e inicializamos el array donde se guardara la estructura de la red
17 EST.NewArray_2D(1,L);
18 EST.ToinitializeArray_2D(0.0);
19 // Se guarda el valor de capas en una variable llamada NC, de tal forma que todo se pueda realizar ↩
      ↪ de modo dinámico solo cambiando el parámetro L
20 float NC=EST.GetColumns();
21 // se crea un vector estatico con los parametros XC, siendo el numero de entradas, en este caso 3, se ↩
      ↪ coloca de modo manual el numero de neuronas de la primera capa 3 y de la segunda capa 2 ↩
      ↪ y DF siendo el numero de salidas en este ejemplo 1. todo guardado en un vector estático.
22 float V[ ]={XC,4,2,DF};
23 // con el siguiente for, reescribimos estos parámetros pero en el arreglo dinámico de la estructura
24 for (byte i = 0; i < NC; i++)
25 {
26   EST.WriteArray_2D(0,i,V[i]);
27 }
28 .
```

```
29 .
30 .
```

Código 5.4: Configuración de Parámetros para crear la estructura de la red con la libreria Neural_Networks_FF

creando la estructura, se puede proceder a asignar las funciones de activacion respectiva por cada capa, del modo como se ve en el código 5.5.

```
1 .
2 .
3 .
4 //Se crea el array dinamico tipo string para pasar as funciones de activacion, con el numero de capas
5 FAC_STR.CharNewArray_1D(L);
6 //se crea un vector tipo char estático que que contiene las funciones de activación, como la primera ↩
      ↪ capa consta de el patrón de entradas se  puede dejar sin funcion de activacion pero llenado ↩
      ↪ con un vacio entre comillas.
7 char *myStrings[] = {" ","tansig", "tansig","purelin"};
8 //Copiamos el vector al array al arreglo dinamico de FAC_STR
9  for (byte i = 0; i < FAC_STR.GetRows(); i++)
10    {
11 FAC_STR.CharWriteArray_1D(i,myStrings[i]);
12     }
13 }
14 void loop() {
15 }
```

Código 5.5: Configuración de Parámetros de funciones de activación de la red con la libreria Neural_Networks_FF

De este modo se realiza el procedimiento para establecer la estructura de la red , y asignar las funciones de activación que pide la función net.FEED_FORWARD_NET (Topología,funciones_activación), colocando como siguiente linea en el void setup de la siguiente forma **net.FEED_FORWARD_NET (EST,FAC_STR)**. se tiene creada toda la tipología de la red de la imagine 5.47

5.6.2.1.1. Ejemplo Red neuronal Como aproximador de funciones

El siguiente ejemplo es implementar la red neuronal como aproximador de la función coseno, tomando el patrón de entradas como los punto a evaluar y salida la repuesta de la función coseno trasladada

```
1 #include<Neural_Networks_FF.h>
2 //Contructor de la red neuronal
3 Neural_Networks_FF net=Neural_Networks_FF();
4 //Array dinamico para ingresar las entradas a la red
5 Dynamic_Array XI=Dynamic_Array();
6 //Array dinamico para ingresar las salidas deseadas
7 Dynamic_Array D=Dynamic_Array();
8 //Array dinamico tipo string para ingresar las funciones
9 //de activacion por capa
10 Dynamic_Array FAC_STR=Dynamic_Array();
11 //Array dinamico para ingresar la estructura de la red
12 //como vector
13 Dynamic_Array EST=Dynamic_Array();
14 void setup(){
15   delay(3000);
16   Serial.begin(9600);
17   //Pin 13 para visialisar el entrenamiento
```

```
18  pinMode(13, OUTPUT);
19  //randomSeed para los pesos aleatorios iniciales
20  randomSeed(analogRead(0));
21  //Crear e inicializar el array de los datos de entrada
22  XI.NewArray_2D(68,4);
23  XI.ToinitializeArray_2D(0.0);
24  //Crear e inicializar el array de salidas deseadas
25  D.NewArray_2D(1,68);
26  D.ToinitializeArray_2D(0.0);
27  ///Creamos los datos para entrenar la RED//
28    for (int i = 0; i<68 ; i++)
29    {
30    XI.WriteArray_2D(i,2,i*0.1);
31    XI.WriteArray_2D(i,3,3.0*cos(i*0.1)+5.0);
32    }
33
34    for (int i = 1; i<69 ; i++)
35   {
36     XI.WriteArray_2D(i-1,0,i*0.1);
37     XI.WriteArray_2D(i-1,1,3.0*cos(i*0.1)+5.0);
38     D.WriteArray_2D(0,i-1,3.0*cos(i*0.1)+5.0);
39      //D.WriteArray_2D(1,i-1,3.0*cos(i*0.1)+5.0);
40   }
41      /// FIN DATOS///
42
43    //Columnas de XI
44   byte XC=XI.GetColumns();
45   //Filas de D
46   byte DF=D.GetRows();
47   //Filas de XI
48   byte XF=XI.GetRows();
49
50   //Centrado de datos
51   net.DATA_CENTERING(XI);
52   // Normalizado de datos
53   net.NORMALIZE_DATA(XI,XI.MaxArray_2D());
54   // Definomos numero de capas
55   byte L=3;
56   // Creamos e inicializamos el array donde se guardara
57   // la estructura de la red
58   EST.NewArray_2D(1,L);
59   EST.ToinitializeArray_2D(0.0);
60   float NC=EST.GetColumns();
61   //Vector que contiene la estructura de la red
62   float V[]={XC,8,DF};
63   //Copiamos el vector al array
64   for (byte i = 0; i < NC; i++)
65   {
66    EST.WriteArray_2D(0,i,V[i]);
67    }
68   //creamos array tipo string para pasar
69   //las funciones de activacion
70   FAC_STR.CharNewArray_1D(L);
71   //Vector que contine las funciones de activacion
72   char *myStrings[] = {"    ", "tansig","purelin"};
73   //Copiamos el vector al array
```

```
74    for (byte i = 0; i < FAC_STR.GetRows(); i++)
75    {
76      FAC_STR.CharWriteArray_1D(i,myStrings[i]);
77     }
78   //Creamos la red
79  net.FEED_FORWARD_NET(EST,FAC_STR);
80  //Entrenamos la red
81  net.TRAIN_NET(XI,D,1e-5,5000);
82  //Simulamos la red
83  net.SIM_NET(XI);
84   //Imprimimos por serial la salida de la red y la salida deseada
85    for (byte i = 0; i<XF ; i++)
86 {
87   Serial.print(D.ReadArray_2D(0,i),4);
88   Serial.print(" ");
89   Serial.println(net.Ysim.ReadArray_2D(0,i),4);
90 }
91 // imprimimos epochs y error cudratico medio //Serial.println(net.EPOCHS);
92 //Serial.println(net.ERROR_C_M);
93 }
94 void loop() {
95 }
```

Código 5.6: Ejemplo Aproximador de función coseno

5.6.2.1.2. Ejemplo Red neuronal Como Controlador Este código, toma la señal de un sensor analógico real, asumiendo como entrada los errores existentes en n intervalos de tiempo, y la salida como el setpoint o valor deseado.

```
1  #include<Neural_Networks_FF.h>
2  //Contructor de la red neuronal
3  Neural_Networks_FF net=Neural_Networks_FF();
4  //Array dinamico para ingresar las entradas a la red
5  Dynamic_Array XI=Dynamic_Array();
6  //Array dinamico para ingresar las salidas deseadas
7  Dynamic_Array D=Dynamic_Array();
8  //Array dinamico tipo string para ingresar las funciones
9  //de activacion por capa
10 Dynamic_Array FAC_STR=Dynamic_Array();
11 //Array dinamico para ingresar la estructura de la red
12 //como vector
13 Dynamic_Array EST=Dynamic_Array();
14
15 Dynamic_Array temperatura=Dynamic_Array();
16 float tem=0.00;
17 float error=0.00;
18 float error_1 = 0.00;
19 float error_2 = 0.00;
20 float setPoint = 0.00;
21 float Output=0.00;
22 float valor_maximo_alcanzado=500;
23 void setup() {
24 pinMode(13, OUTPUT);
25 pinMode(9, OUTPUT);
26 pinMode(A1, INPUT);
```

```
pinMode(A2, INPUT);
randomSeed(analogRead(0));
Serial.begin(9600);

  //Crear e inicializar el array de los datos de entrada
  XI.NewArray_2D(1,3);
  XI.ToinitializeArray_2D(0.0);
  //Crear e inicializar el array de salidas deseadas
  D.NewArray_2D(1,1);
  D.ToinitializeArray_2D(0.0);
  //Columnas de XI
  byte XC=XI.GetColumns();
  //Filas de D
  byte DF=D.GetRows();
  //Filas de XI
  byte XF=XI.GetRows();
  // Definomos numero de capas
  byte L=3;
  // Creamos e inicializamos el array donde se guardara
  // la estructura de la red
  EST.NewArray_2D(1,L);
  EST.ToinitializeArray_2D(0.0);
  float NC=EST.GetColumns();
  //Vector que contiene la estructura de la red
  float V[]={XC,4,DF};
  //Copiamos el vector al array
  for (byte i = 0; i < NC; i++)
  {
   EST.WriteArray_2D(0,i,V[i]);
   }
   //creamos array tipo string para pasar
   //las funciones de activacion
   FAC_STR.CharNewArray_1D(L);
   //Vector que contine las funciones de activacion
   char *myStrings[] = {"logsig","tansig","poslin_lim"};
   //Copiamos el vector al array
   for (byte i = 0; i < FAC_STR.GetRows(); i++)
   {
     FAC_STR.CharWriteArray_1D(i,myStrings[i]);
    }
    net.dposlin_limits(1,0);
    //Creamos la red
    net.FEED_FORWARD_NET(EST,FAC_STR);
    //se crea arreglo dinamico para guardar valor de temperatura
    temperatura.NewArray_2D(1,1);
    temperatura.ToinitializeArray_2D(0.0);
    // se asgina valor de aprendizaje del proceso
    net.LearningRate=0.017;

}


void loop() {
  // lectura del sensor
  // valor_maximo_alcanzado=500 grados centigrados
  tem =((analogRead(A1)*5.00)/1024.00)*100.00;
```

```
83   //asignacion del setpoint con entrada de un potenciometro
84   setPoint=(analogRead(A2)*valor_maximo_alcanzado)/1024.00;
85   // Se escribe el valor leido de la temperatura real aun arreglo
86   // dinamico denominado temperatura, ya que la red solo recibe arreglos dinamicos
87   // y normalizado.
88   temperatura.WriteArray_2D(0,0,tem/valor_maximo_alcanzado);
89   // se calcula el error de la variable real.
90   // para luego ser guardado en el arreglo dinamico de entradas
91   // guardando el retardo del error de la iteracion anterior
92   // el retador del error de la iteracion anterior anterior.
93   error = setPoint-tem;
94   XI.WriteArray_2D(0,0,error);
95   XI.WriteArray_2D(0,1,error-error_1);
96   XI.WriteArray_2D(0,2,error_1-error_2);
97
98   //se realiza la normalizacion de los datos
99   net.NORMALIZE_DATA(XI,valor_maximo_alcanzado);
100  // se escribe el valor del setpoint normalizado en el arreglo dinamico de salida
101  D.WriteArray_2D(0,0,setPoint/valor_maximo_alcanzado);
102  // se realiza el entreamiento en linea.
103  net.TRAIN_NET_ONLINE(XI,D,temperatura);
104  // se optiene el valor de salida de la red
105  // en una variable en la escala de 0-1, tomado de la ultima neurona.
106  Output=net.y.ReadArray_2D(net.y.GetRows()-1,0);
107  // el valor de salida optinido se multiplica por un factor
108  // en resolucion de salida del microcontrolado en este caso
109  // opera de 0-225, per se utiliza una resolucion maxima de 110
110  analogWrite(9,Output*110.00);
111  // imprime el comportamiento de la variable
112  // Setpoint, valor de temperatura, error y selak de control
113
114    Serial.print(setPoint);
115    Serial.print(" ");
116    Serial.print(tem);
117    Serial.print(" ");
118    Serial.print(error);
119    Serial.print(" ");
120    Serial.println(Output*110.00,5);
121    // se realizan los dos retardos del error al final de cada
122    // ciclo de ejecucion del codigo
123    error_2= error_1;
124    error_1 = error;
125    delay(1000);
126 }
```

Código 5.7: Ejemplo red neuronal como control de temperatura

6. Resultados

6.1. Fabricación Del Sistema Embebido

La fabricación del sistema embebido parte de de los diseño realizados en el software de diseño eagle visto en el desarrollo, y diseño 3D fusión 360. ya teniendo el diseño desarrollado Se buscaron varios fabricantes de PCB's en la región, pero ninguno cumplía con las exigencias para la elaboración del circuito impreso o la calidad de fabricación era baja, por lo que se octava mandar a fabricar la PCB a una empresa extrajera, aunque podían realizar el circuito impreso en buena calidad, los tiempo de fabricación y envió eran extensos, podían exceder el tiempo de espera de 1 mes.
debido a ello se siguió buscando una empresa a nivel nacional. Se logro contactar con un empresa llamada col-circuitos ubicada en Medillin, Siendo experta en fabricación de PCB's, pero para poder fabricar un placa se debia realizar una cotización y corroboración del diseño por parte de ellos, si el circuito no cumplía con la exigencias mínimas de fabricación impuesta por la empresa, se debían acatar los termino y volver a rediseñar el circuito, como el circuito se realizo con las exigencias de esta empresa, no se tuvo ningún cambio de diseño de ultima hora, y el resultado de la placa de circuito impreso ya elaborada se aprecia en la figura 6.1, la capa top del circuito impreso

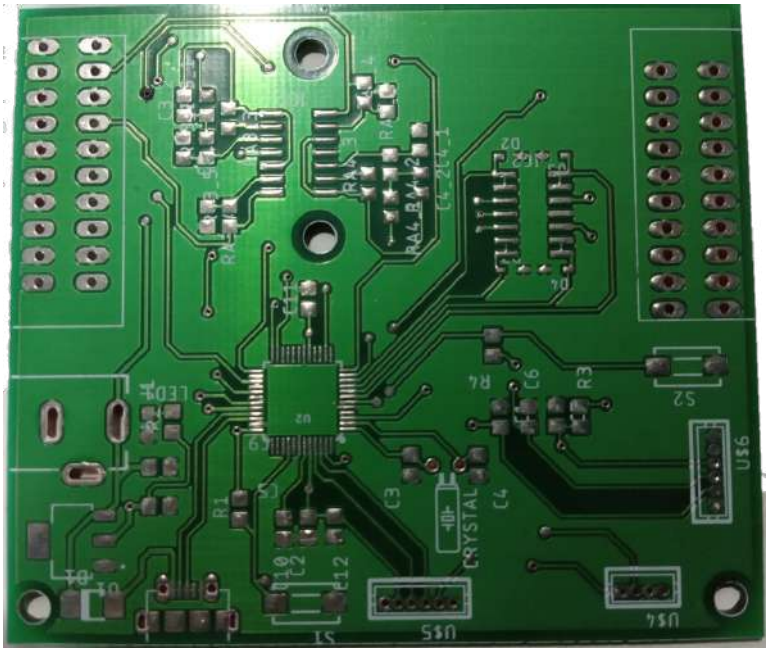


Figura 6.1: PCB capa Top
Fuente: AUTOR

los beneficios que aporte la fabricación de una PCB en una empresa de esta categoría, es al momento de soldar los componentes, se aprecia los pads, con estaño de color plateado y los puentes que comunican una cara con la otra, ya vienen unidos, por lo que ahorra trabajo y garantiza su funcionamiento adecuadamente, también vienen con los nombres de cada componente y sus dimensiones para una mayor comprensión del circuito, en la figura tal se tiene una vista de la cara button de la PCB 6.2.

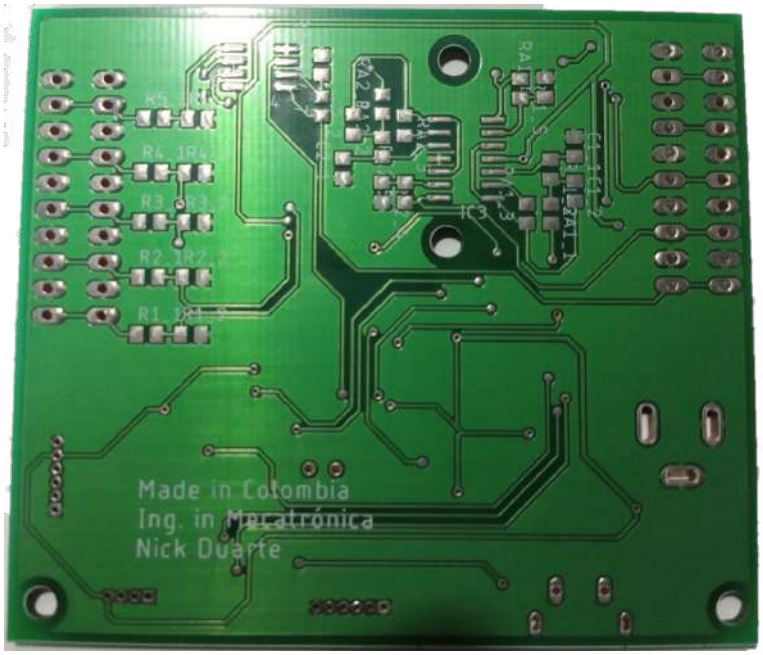


Figura 6.2: PCB capa Button
Fuente: AUTOR

el siguiente paso de la fabricación del sistema embebido consistía en soldar los componentes seleccionados en la placa de circuito impreso. la soldadura y todo los procesos respectivos fueron realizados por el autor del libro, enmarcando la amplia gama de capacidades que pueden tener un ingeniero en mecatrónica de la universidad de Pamplona.

debido a que el sistema embebido operara con entras y salidas analógicas en el rango de 0-10V se comprueban el funcionamiento de todos los puertos de entrada y salida, en la figura 6.3 se tiene una fuente de voltaje regulable con un display ilustrando el valor de tensión, esta fuente va conectada a un puerto de entrada del sistema embebido, este valor de tension es reflejada, en un puerto de salida siendo medido por un multimetro, este procesos se realizo con todos los puertos de entrada y salida. en la imagen se ve una notoria diferencia de 0.8V, aunque tiene una diferencia entre 0.1V y 0.2V, esta desvió es debido a la lectura del voltaje del display que no suele ser exacto, y la diferencia de voltaje, tiende a ser por los componentes o valores de resistencias que no suelen ser tan exactos.

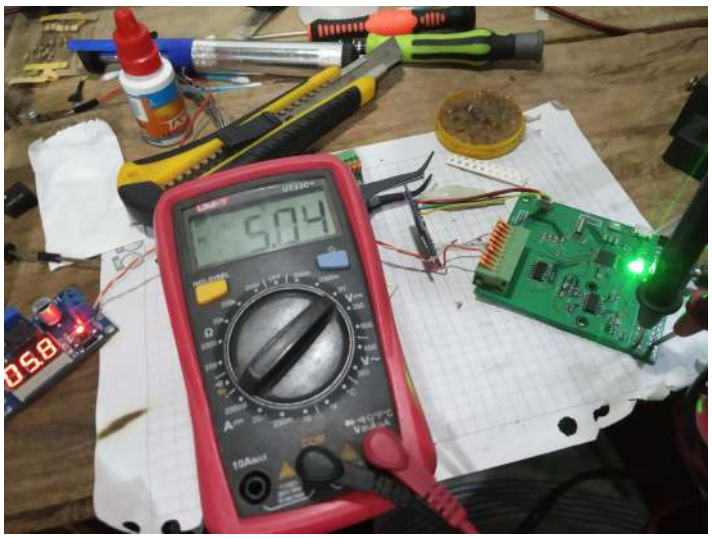


Figura 6.3: Prueba de entradas y salidas del sistema embebido
Fuente: AUTOR

una vez teniendo confirmado que la placa de circuito impreso esta funcional, se sigue con el proceso de fabricación del sistema, tomando los diseños realizados por el software de fusión 360 para la elaboración del case, esta fabricación fue realizada con una maquina de impresión 3D por proceso de manufactura aditiva, de estos diseños extrajeron los archivos G.code que lee la maquina y puestos en marcha, la impresión de total de las piezas tomaron un tiempo alrededor de 15 horas, sin contar el tiempo entre pieza y pieza, los tiempos del arreglo de la piezas por el material de soporte, y así obtenido como resultado las piezas impresas vistas en la figura 6.4, siendo las piezas que se diseñaron en el desarrollo

Figura 6.4: Piezas del case con impresión 3D
Fuente: AUTOR

Para finalizar, se realizo todo el proceso de ensamblaje, y obtener como resultado el modulo del sistema embebido. ver figura 6.5

Figura 6.5: Sistema embebido
Fuente: AUTOR

las características físicas del sistema embebido son las siguientes:

- Voltaje de operación: 5 V - 12 V.
- Consumo en corriente: 27mA en vació.
- Voltaje de entrada analógico: 0V-10V.
- Voltaje de salida Analógico: 0V-10V.
- Numero de puertos de entrada ADC: 5.
- Numero de Puertos de salida DAC: 5
- puertos: Puerto Scom USB, puerto de programación para programador externo.

6.2. Interfaz

Acorde a la estructura de la interfaz establecida en el desarrollo, vista en la figura 5.32, se realiza la programación en el entorno de desarrollo arduino, optimizando el uso de memoria flahs del microcontrolador para no ocupar tanto espacio de memoria y genere problemas con los algoritmos de control y de este modo tener una interfaz interactiva. como resultado se desarrollo el menu del sistema embebido en los siguientes submenus se evidenciaran la forma de uso directamente desde la interfaz del sistema.

Figura 6.6: Logo Inicio de la interfaz del menu
Fuente: AUTOR

6.2.1. Modo de uso del menú

el sistema embebido es un dispositivo con algoritmos de control, por lo que el poder seleccionar variables que se desea controlar, como la alteración de algún parámetro, es importante, la comunicación con el dispositivo y la persona que lo manipule. Debido a ello cuenta con una pantalla oled, mostrando menú principal de 3 tipos de variable que cada variable abarca un tipo diferente de parámetros a introducir, los menús que se visualizarán al momento de usar el sistema, serán explicados en el siguiente ejemplo. Solo se explicará el modo de uso de una sola variable, debido a que los submenús en todas las variables serán los mismo, lo que cambiaría, serian lo parámetros a tener en cuenta de la planta física a controlar.

6.2.1.1. Configurar menú para planta de velocidad

cuando se energiza el modulo del sistema embebido se debe considerar que variable queremos realizar control clásico o neuronal. Por lo que es debido tener en cuenta, el uso de los sensores para un correcto funcionamiento. Para el control de velocidad se asignaran dos puertos de entrada predeternimados para la captura de la señal, estos puertos pueden verse en la ficha técnica del dispositivo, para el control de velocidad, no se necesita configurar ningún parámetro de entrada, solo tener en cuenta las consideraciones de conexión del sensor al sistema, en las siguientes tablas se verán los pasos seguidos para configurar el sistema para realizar un control de velocidad. La arquitectura dispuesta para este menú se puede ver en el siguiente apartado.

6.2.1.1.1. Configuración parámetros iniciales Antes de iniciar aplicar el controlador se debe llenar unos datos de parámetros iniciales, que son los principales para usar en los dos controladores que dispone este módulo, el control PID y el control Neuronal de topología configurable. En la siguiente tabla se verán los primeros datos que se deben ingresar al menu.

Pasos	Imagen	Descripción
1		Al energizar el modulo, se inicializa el menú indicando que tipo de variable se va controlar, en este ejemplo selecciona la variable velocidad
2		En la primera línea indica que esta ubicado en el menú de velocidad. las casillas que se indican con un puntero, son asignación de setpoint, valor Maximo, valor mínimo puerto de entrada o salida, y menú siguiente.
3		Cuando se selecciona el setpoint, valor máximo, o valor minimo se desplegara este menú, indicando cual será la escala del incremento de la variable, si de 100, 10 o 1 pasos en el cambio de la variable.
4		Al seleccionarse la casilla del setpoint e indicar el incremento, este será la interfaz que muestra el estado de la variable, si se gira la perilla a la derecha aumenta la variable, si se gira a la izquierda decrece. Cuando se asigne el valor, se regresa al menú de velocidad con el botón back. mostrando el menú del paso 2.
5		De igual manera para el valor de valor máximo, se selecciona la casilla de valor máximo, se indica el incremento. esta será la interfaz visual del cambio de la variable de valor máximo. Asignado el valor se regresa al menú velocidad con el botón back.

Tabla 6.1: Pasos de configuración de parámetros iniciales
Fuente: AUTOR

Pasos	Imagen	Descripción
6		Los mismos pasos se realizan para asignar un valor de valor mínimo, se selecciona la casilla de valor mínimo, se indica el incremento. esta será la interfaz visual del cambio de la variable de valor Mínimo. Asignado el valor se regresa al menú velocidad con el botón back.
7		Al elegir la casilla de puertos I/O, sale un menú indicando, si desea alterar un puerto de entrada o de salida.
8		Interfaz visual de selección del puerto de entrada e ilustra la ubicación de puerto seleccionado. una vez seleccionado el puerto se devuelve al menú de velocidad.
9		Interfaz visual de selección del puerto de salida, e indica la posicion ubicada del puerto en el terminal de salida. Una vez se seleccione el puerto se dirige al menú velocidad..

Tabla 6.2: Pasos de configuración de parámetros iniciales continuación
Fuente: AUTOR

6.2.1.1.2. Parámetros del controlador PID Cuando ya se tiene los valores de parámetros iniciales de la planta de velocidad, con puerto de salida configurado. En la tabla anterior en el paso 2, la última casilla se llama Siguiente, al entrar en ella, nos da a elegir entre dos controladores Control PID, control Neuronal visto en la figura 6.7. Este apartado se indicará como se llenan los valores de constantes del PID.

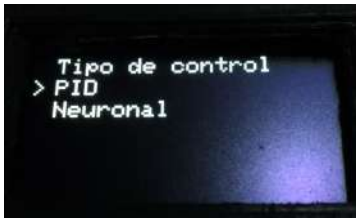


Figura 6.7: Selección tipo de control
Fuente: AUTOR

Al seleccionarse el controlador PID, la interfaz visual permite realizar cambios directamente sobre las variables, en la siguiente figura 6.8 se visualiza las posiciones de cada constante en la interfaz, posición 1, constante proporcionar Kp, posición 2 constante integral Ki, posición 3 constante integral Ki

Figura 6.8: Selección tipo de control
Fuente: AUTOR

Para cambiar el valor de la variable, solo se debe ubicar en la posición indicada por el tag, ingresar a ella, con el botón de select, de igual manera como se configuraron los valores de setpoint, valor máximo y mínimo, girar la perilla a la derecha para incrementar la variable o a la izquierda para hacerla más pequeña. Además, permite visualizar y realizar cambio de setpoint en esta interfaz. Una vez configurado esto parámetros, solo se ubica en la posición iniciar, e inicializa el proceso de control de velocidad del motor en rpm, la interfaz permite ver el error existente entre el valor real y el valor del setpoint, y por motivos de espacio no se pudo ubicar el tag que indica el valor real de entrada, sin embargo, este valor se aprecia debajo del error.

6.2.1.1.3. Parámetros del controlador neuronal Cuando se elige realizar control neuronal es importante importantes tener un conocimiento básico para configurar estos parámetros que la casilla de control neuronal comenzara a pedir al momento de seleccionarse.
El número de capas totales comienza a contar desde las neuronas de entrada, el número de capas ocultas y la capa de salida, queriendo decir que cuando se elige una topología de 3 capas, la primera capa corresponde a las entradas ya definidas con 3 neuronas, la segunda capa será la capa oculta, la tercera capa será la capa de salida con una neurona.
En la imagen 6.9 se ilustra una topología de 4 capas, con sus respectivas 3 neuronas de entrada ya pre-asignadas, 2 neuronas en su primera capa oculta y 2 neuronas en su segunda capa oculta, y la capa de salida con 1 neurona. se visualiza que la primera capa oculta, segunda capa oculta y la capa de salida, cuenta un nombre encima, indicando la función de activación correspondiente a cada capa.

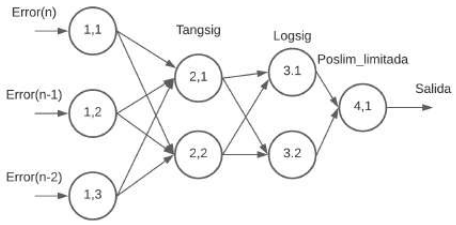


Figura 6.9: Topología de ejemplo para uso de la interfaz
Fuente: AUTOR

Este ejemplo de topología de red neuronal lo aplicaremos como ejemplo para la configuración de parámetros, que el sistema embebido va a requerir en el proceso. Cuando se elige el controlador neuronal en el menu, de tipo de control, enseguida de seleccionarlo, cambia de interfaz visual indicando el número de capas que se van a utilizar tal como se aprecia en la imagen 6.10, asignando las 4 capas vistas en la topología de la imagen anterior.

Figura 6.10: Interfaz selección numero de capas
Fuente: AUTOR

Moviendo la perilla se pueden cambiar el valor de capas desde un valor minio de capas de 3 hasta un valor máximo de 6 capas, cuando se asigna el valor de capas que se van a usar, se pulsa la perilla para seleccionar ese valor, iniciando la siguiente interfaz muestra el número de la primera capa oculta que se le asignara el número total de neuronas.

Figura 6.11: Interfaz selección de numero de neuronas capa oculta 1
Fuente: AUTOR

Esta interfaz pide el número total de neuronas por capa, en la imagen de la topología está compuesta por 2 neuronas en su primera capa oculta, mediante la perilla asignan 2 neuronas

a esta capa y se presiona la perilla de select.
Como se solicitó un valor de 4 capas totales, con 2 capas ocultas. cuando ya se selecciona el valor de neuronas de la capa anterior, la interfaz arroja nuevamente la asignación de neuronas, solo que esta vez se le asigna el número de neuronas la segunda capa oculta, tal como se visualiza en la siguiente imagen, en el ejemplo esta capa cuenta con 2 neuronas, se selecciona este valor y se presiona la perilla para seguir con el siguiente parámetro

Figura 6.12: Interfaz selección de numero de neuronas capa oculta 2
Fuente: AUTOR

En las siguientes interfaces, después de haber seleccionado el número de neuronas por capa. se le debe asignar la función de activación a cada capa, siguiendo con la nomenclatura, ya que la primera capa de la red, solo consiste de las entradas, no se le asigna función de activación, por ello comienza desde la primera capa oculta hasta la capa de salida, asignando a la primera capa oculta el valor de función de activación tansig.

Figura 6.13: Interfaz selección de función de activación de neuronas capa oculta 1
Fuente: AUTOR

Se selecciona la función de activación con el pulsador de select de la perilla, de este modo pase a la siguiente capa para asignar función de activación. Así como se muestra en la siguiente imagen, asignado la función logsig a la segunda capa oculta.

Figura 6.14: Interfaz selección de función de activación de neuronas capa oculta 2
Fuente: AUTOR

Por último, se muestra la última capa, siendo la capa de salida vista en la siguiente imagen. como se utilizará una red neuronal como controlador, se utiliza la función poslim_limitda, la cual actúa como la función identidad, solo que limita en un rango de 0 a 1 en valores de control, en salida del sistema embebido es reflejado como salida análoga de 0V-10V.

Figura 6.15: Interfaz selección de función de activación de neuronas capa salida
Fuente: AUTOR

Cuando se termina de seleccionar la función de activación de la última capa de salida, la siguiente interfaz, pedirá el valor el valor iniciar del factor de aprendizaje, iniciando en un valor que se requiera, cuando se tenga el factor de aprendizaje seleccionado, hay un indicativo que iniciar, al darle select con la perilla se inicializa el proceso de control de la variable.

Figura 6.16: Interfaz selección factor de aprendizaje
Fuente: AUTOR

Cuando se inicia el proceso de control neuronal, arroja por ultimo una interfaz visual, similar al del PID, solo que este indica el nombre del control aplicado, muestra la topología usada en el controlador neuronal, el valor del setpoint, el valor de entrada real registrado, el valor de error, y por último el factor de aprendizaje, que permite ser modificable en el proceso.

Figura 6.17: Interfaz de control de red neuronal
Fuente: AUTOR

6.3. Implementación de Controladores en la Planta de Temperatura

En la planta de Temperatura, se aplicaron dos tipos de controles, un controlador clásico proporcional integral PI y un control neuronal de topología 3 neuronas en la capa de entrada, 4 neuronas en la capa oculta y 1 neurona de capa de salida, con función de activación logsig y poslim_limitada en la capa de salida, se realizaron pruebas de tiempos de estabilización, perturbación y cambio de setpoints, y de este modo se capturaron los datos por el puerto serial del sistema embebido siendo guardados en un archivo de excel, acorde con estos datos observar y comparar las respuestas entre el controlador PI con la Red neuronal, los datos se tomaron tratando de tener las mismas condiciones posibles de las plantas para una validación óptima del controlador neuronal.

6.3.1. Respuesta del control PI

En el control Proporcional integral de la planta de Temperatura su constantes del Kp y Ki fueron tomadas de la sintonización del controlador por el programa tuner con el modelo hallado a partir de los datos de identificación. Aunque se tuvieron que utilizar otro valores de las constantes, ya que lo encontrados en el modelado de la planta no tuvieron un buen desempeño en el sistema físico real y los tiempos de estabilización eran muy extensos, pero con esos datos de la sintonización dieron base para variar el valor de las constantes usándose los valores de Kp= 295.665 y Ki=25.69678.

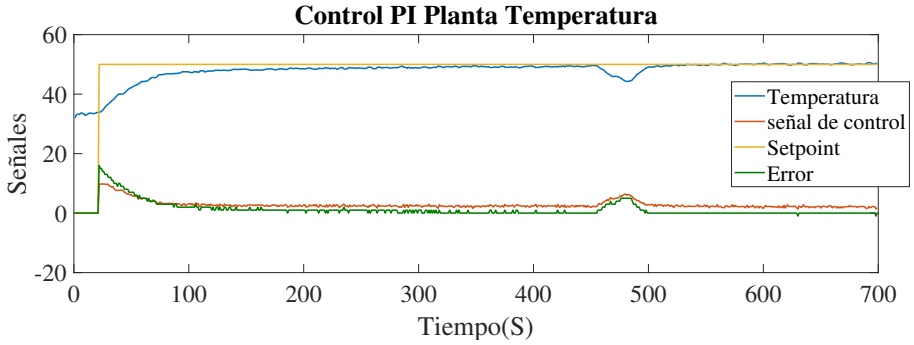


Figura 6.18: Repuesta de control PI ante una perturbación

Fuente: AUTOR

como bien se sabe la planta de temperatura por su naturaleza tiene una dinámica lenta, y una buena sintonización de un controlador puede ayudar al tiempo de respuesta de la planta. por lo que en la figura 6.18, se aprecia la respuesta del comportamiento de la planta con las constantes mencionadas anteriormente, el valor de inicio de la variable de temperatura comenzó en un rango de 30° antes de aplicarle el setpoint, una vez se le aplica el setpoint de 50°, hay un cambio abrupto de la señal de control llevando la variable de temperatura a un ritmo rápido al setpoint, sin embargo en el intervalo de tiempo de aproximadamente 100 segundo la variable de temperatura deja crecer de manera exponencial, y la señal de control se trata de estabilizar en un valor de 2.5V a la salida del sistema embebido, y llegándose acer-

car a la respuesta en un intervalo de 470 segundos, en un tiempo de 7.48 minutos, en donde segundos después se le aplica una perturbación de -6° del valor del setpoint, de este modo se puede corrobarar el funcionamiento del controlador, restableciéndose al valor deseado en unos aproximados 10 segundos.
cabe aclarar que las señales vista en la gráfica 6.18, están con mucho ruido, esto es debido a la instrumentación usada, y que debe pasar por un circuito de reducción hasta el micro-controlador, sin embargo el controlador PI cumple con su tarea de mantener la temperatura en el setpoint establecido, teniendo un error medio de la temperatura de 1.5° centígrados en estado estacionario.

otro método de comprobar la eficiencia del controlador, se realiza enviando cambios de set-point en el transcurso del proceso, para esta prueba se comienza con un valor de setpoitn de temperatura de 30°, seguido de un valor de 40°, seguido de un valor de 50° y por ultimo un valor te temperatura de 35°, estos cambios de setpoint se realizan cada 400 segundos, el figura 6.19, se ilustra la respuesta del comportamiento de la planta, inicialmente, no llega al valor de 30°, debido a que el sistema mantuvo una inercia o un temperatura en su interior, luego de ello se aplicaron los cambios de setpoints, de este modo validando el funcionamiento del controlador clásico en el sistema embebido.
en cambios de temperatura cercanos la como visto en el segundo 400, que se envía un valor deseado de 40°, la respuesta del controlador tiende a ser rápida y eficiente llegando a un estado estacionario en tiempo de aproximadamente 5 minutos, y en cambios de setpoints mas distantes, como vistos en el segundo 800 a un valor de de temperatura deseada de 60°, la señal de control necesita mas tiempo para estabilizarse. siendo en cierto caso una mala sintonización del controlador.
en el segundo 1200, se realiza el ultimo cambio de setpoint a un valor de 35°, en este caso la señal de control cae a 0, hasta que por la naturaleza del sistema reduzca su temperatura al valor de 35 y el controlador envie nuevamente un valor de la señal de control.

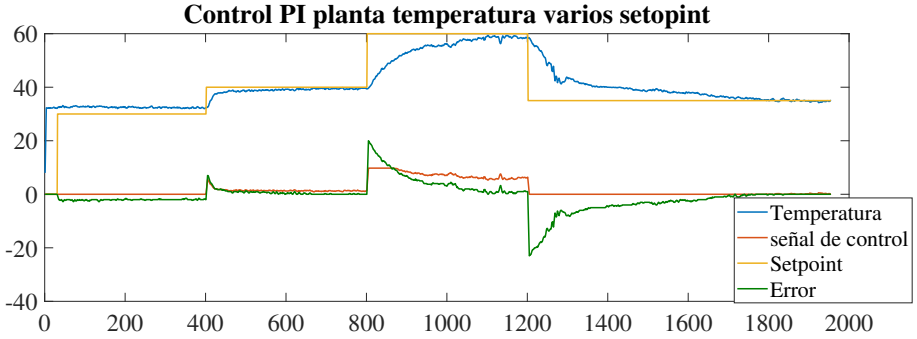


Figura 6.19: Repuesta de control PI ante varios cambios de setpoint
Fuente: AUTOR

6.3.2. Respuesta del controlador neuronal

la implementación del controlador neuronal en la planta, esta definido por una topología, 3 neuronas en la capa de entrada, 4 neuronas en la capa oculta y una neurona en la capa

de salida, con función de activación logsig en la capa oculta y poslim_limitada en la capa de salida, esta topología puede apreciarse en la figura 6.20, teniendo como entradas de las 3 primeras neuronas, el error, y en las siguientes el retardo del error anterior y retardo de dos errores.

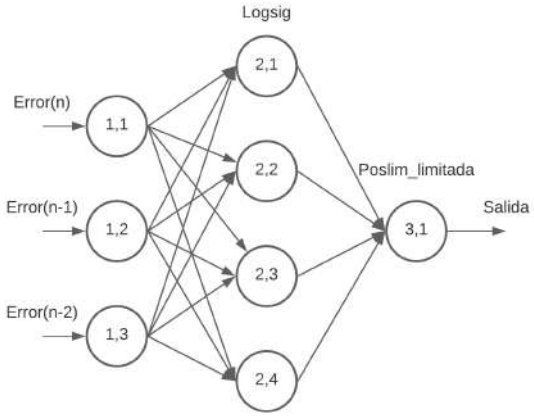


Figura 6.20: Topología de Red Neuronal como controlador
Fuente: AUTOR

Para utilizar la red neuronal en la planta de temperatura, se usaron tiempos de muestreo de 1 s a consecuencia de la naturaleza de la planta ya que la variable no cambia tan rápido en el tiempo, seria redundante hacer tiempo de muestre inferiores a este valor, tanto como lectura de la señal, como para la ejecución del algoritmo neuronal, se uso un factor de aprendizaje de la red de un valor de 0.015, siendo este valor el que presento mejor comportamiento al momento de controlar, teniendo tiempos de respuesta relativamente rápidos y con poco sobrepico.

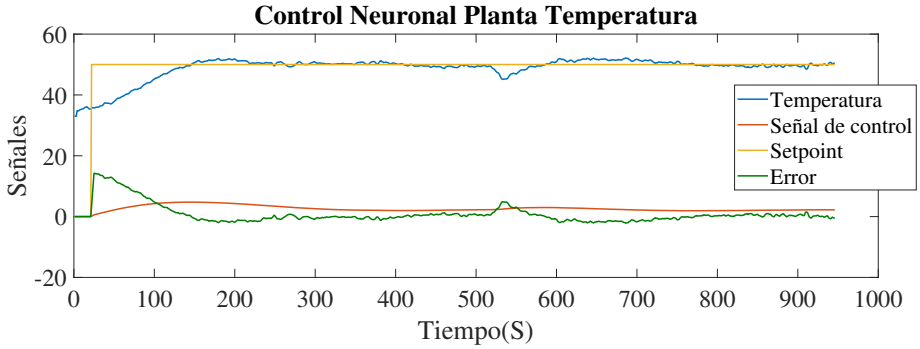


Figura 6.21: Repuesta de la planta temperatura con controlador neuronal
Fuente: AUTOR

se hizo una evaluación de la respuesta del controlador ante un setpoint establecido de 50° y una perturbación, similar a la vista en la figura 6.18. La respuesta del comportamiento de

la planta con el control neuronal se puede ver en la figura 6.21.
analizando la respuesta obtenida, se tiene que la señal de color azul sera el valor leído por el sensor lm35 registrando la temperatura real, la señal de color amarillo el valor de Setpoint establecido, la señal de color naranja la señal de control en un rango de voltaje de 0-10V y por ultimo la señal de error. Se evidencia que el valor de temperatura a simple vista llega al valor deseado en un tiempo corto y la señal del error deciende hasta acercarse al 0, a comparación del controlador clásico PI. acercándose al valor de estado estacionario en el tiempo de 250 segundo, en aproximadamente 3.8 minutos de aplicarse el cambio de setpoint. acorde con este tiempo presentado en la respuesta del controlador con una topología de 3 capas demuestra ser mas rápido que el control PI, con las constates utilizadas. Aunque la respuesta que presenta un sobre-pico de un valor aproximado entre 1-2 grados o del 4%, este sobre-pico es debido a que el controlador debe detectar el cambio de signo del error, para que de este modo pueda corregir con la señal de control de manera incremental, una vez el valor de temperatura se encuentra en el setpoint, la señal de control tiende a mantenerse en un voltaje de 2.68V oscilando por encima de este valor de igual modo por debajo,de este modo mantener la temperatura en el Setpoint de 50°.
como se puede observar la lectura de temperatura, oscila levente sobre y por debajo del valor del setpoint. esto es a causa de una mala lectura del sensor debido al ruido interno que tenga la lectura, lo cual generara lecturas falsas de la temperatura real, haciendo creer al controlador que la temperatura deseada sobre paso el valor o esta muy por debajo. además la salida del sistema embebido, opera con una resolución de 12 bits, se esta codificando el valor de voltaje que varia de 0-10V en pasos de 0 a 4095, en una tarjeta de desarrollo arduino, que actúa como la parte de potencia, adquiriendo la señal en una resolución de 10 bits en pasos de 0-1023, y teniendo una salida al bombillo de 0-100% de control de fase de la intensidad maxima. lo que conllevaría a oscilar un poco mas la respuesta en estado estacionario de la temperatura.

En la figura 6.22, se visualiza el comportamiento de la planta ante-varias entradas, siendo evaluadas en los mismos intervalos de tiempo y en los mismo valores de setpoint evaluados en la planta de temperatura, con el controlador clásico proporcional integral PI.

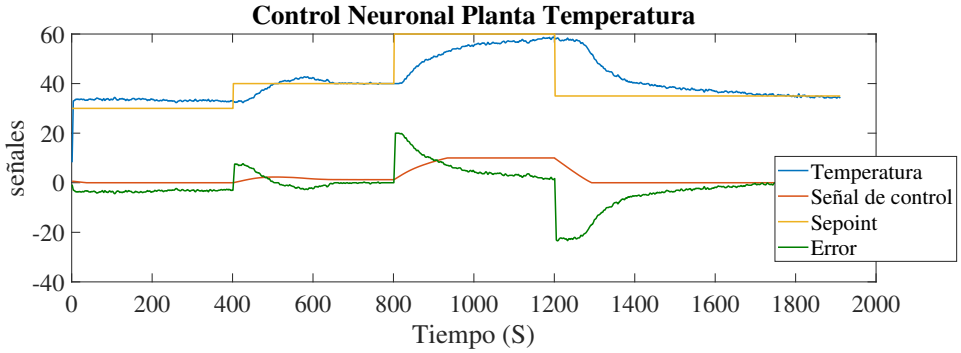


Figura 6.22: Repuesta de control Neuronal Multi-capa en varios setpoints
Fuente: AUTOR

De la misma manera de lo sucedido con el control clásico, al inicio del proceso. la planta mantiene una inercia de temperatura, por motivos que se esta trabajando en condiciones

semi-cerradas, y la temperatura ambiente tiende a incidir en este comportamiento, pero se mantiene en un valor constate de 32 - 33° centígrados.los colores de las señales, temperatura, setpoint y error se mantienen la misma relación para la representación del comportamiento de la planta.
Al aplicarse el primer cambio de setpoint a un valor de 40°,la lectura tomada por el sensor, tiende a mantenerse constante durante un corto periodo de tiempo de 20 segundos, este comportamiento es apreciado debido a la inercia que tiene la temperatura en ese momento,y que la señal de control aun no ha alcanza el umbral para romper esa inercia para poder aumentar la temperatura del sistema, pasado ese tiempo, que para el controlador neuronal significan números de iteraciones para la corrección del error, la señal de control, ya alcanza un valor que permite aumentar la temperatura en el sistema de este modo llevarlo al setpoint establecido, en un tiempo de 4.1 minutos, con un sobrepico del 5% , siendo corregido en un tiempo de 2 minuto, siendo un periodo de tiempo algo extenso, sin embargo sigue siendo un buen tiempo, ya que no se tiene control de la velocidad del decremento de la temperatura, en ese caso actuaria solo la inercia de la planta y llegar al punto de estabilización. que se mantiene un tiempo de estabilización cercano al tiempo de la figura 6.21, del control neuronal ante un setpoint, los tiempos pueden variar un poco, debido a los pesos iniciales de la red neuronal que siempre van a ser aleatorios, en cada inicio del proceso, lo que significaría que podrían llevar la respuesta de la planta al valor establecido en un intervalo de tiempo mayor o menor, usando el mismo factor de aprendizaje.
el comportamiento de la temperatura en estado estacionario va a tener a oscilar por encima y por debajo de los valores del setpoint como se logra apreciar en la figura 6.23, por las perdidas de resolución de salida del controlador y por el ruido de la lectura del sensor, logrando adecuación con elementos de mayor robustez se lograría tener un mayor control.

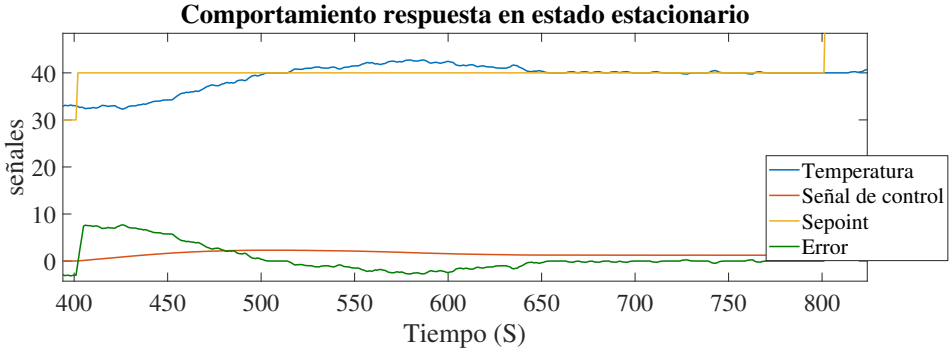


Figura 6.23: Respuesta en estado estacionario setpoint de 40
Fuente: AUTOR

continuando con el análisis de la respuesta presentada en la figura 6.22,se tiene el siguiente cambio de setpoint al valor de 60,evidenciando igualmente la inercia que tiene la temperatura,como el valor de setpoint esta cerca del rango máximo que ofrece la planta, en este punto la temperatura asciende a un ritmo mas lento, por lo que el numero de iteraciones que el controlador realiza, son las suficientes para que esta señal llegue al limite, en un valor de 10V saturándose en ese valor, hasta aproximarse al valor del setpoint.
en el ultimo cambio de setpoint, la señal de control actúa de modo inverso, al setpoint an-

terior, llegando a su valor mínimo de 0 V a su salida, quedando a la espera que la misma dinámica de la planta baje el valor que se desea, para que en ese punto el controlador retome su función de mantener la temperatura en ese valor.

6.3.2.1. Comparativa del controlador clásico vs el controlador neuronal

En el siguiente aparto, se aparecieran las señales de respuesta de la planta de temperatura, con un controlador diferentes, la señal de color azul, sera la referencia del controlador PID, la señal de color naranja la referencia del controlador neuronal multicapa.

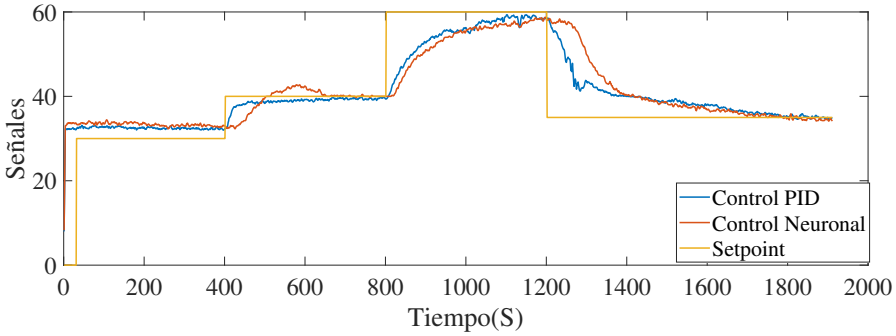


Figura 6.24: Repuesta de controladores PID y Neuronal
Fuente: AUTOR

en los primeros 400 segundos se mantuvo una tempera ruta de aproximadamente 32 grados en el sistema, en las dos pruebas, siendo esto una inercia propia del sistema y de las condiciones externas, por lo que se analiza la señal una vez realice el cambio de setpoint, a un valor ed 40.
Cuando el cambio de setpoint es aplicado, el valor de señal registrado por el PID aumenta de forma exponencial, hasta pasado unos 40 segundos, donde el valor comienza a mantenerse constante, mientras que el valor registrado por el controlador neuronal, incrementa su valor gradualmente, en base al valor de factor de aprendizaje impuesto.
mientras que el valor de temperatura, sube gradualmente a un valor bajo, la temperatura del controlador neuronal genera un sobre-pico de casi 2 grados, pero teniendo un tiempo de estabilización desde la aplicación del setpoint, de un valor de 4.1 minutos, mientras el valor de PID llega al valor de estado estacionario en 6.4 minutos
en el segundo cambio de setpoint a un valor de 60°, se aprecia la similitud de las dos respuestas, esto debido a que en ese punto las señales de control están al máximo, por lo que el comportamiento debería ser similar en los dos respuestas, y acercándose a la respuesta establecida.
en el tercer cambio de sepoint al valor de 35, al principio se ve una caída abrupta por parte del controlador PID, debido a que la señal de control se vuelve cero, ese seria el comportamiento natural de al dinámica de la planta en descenso de temperatura, mientras que l neuronal, decrece leven mente mientras el computo de los pesos se vuelven cero por influexia del factor de aprendizaje.

6.4. Implementación de Controladores en la Planta de Velocidad

El control de velocidad de la planta, se implemento un controlador clásico PID, y control neuronal multicapa de 4 capas, 3 neuronas en la capa de entrada, 2 en la primera capa oculta, 2 en la segunda capa oculta, y una neurona de salida. por la dinámica rápida en la que se comporta esta la planta, se tomaron tiempos de muestreo de 100ms, para la aplicación del controlador clásico y el controlador PID.por consiguiente para una óptima validación de estos dos tipos de controladores, las condiciones de la planta se mantuvieron similares en la implementación los controladores. los datos fueron capturados por puerto serial del sistema embebido, para así poder graficarlos y realizar el respectivo análisis.

6.4.1. Respuesta del controlador PID

Para Las constantes del controlador PID se utilizaron las constantes halladas por el sintonizador tuner de matlab, KP=0.04966, Ki=0.19933, Kd= 0.0000458, que presentaron un buen desempeño al momento de realizar las pruebas, controlando la velocidad del motor en rangos de valores diferentes del setpoint.en la figura 6.25, se visualiza la respuesta de la planta sometida a cambios de setpoint, de valor de 0 a 1000 rpm, 1500 rpm, 2000 rpm, 2500 rpm, 3000 rpm decreciendo a 2500 rpm, 2000 rpm a 1500 rpm , permitiendo ver en la figura las señales del estado de la variable controlada de color azul, el estado de la señal de control de color naranja este valor se tomo de la resolucion maxima permitida del sistema embeido en valores de 0-4095, lo que seria en la salida del sistema embebido una salida reflejada de 0V-10V, pudiendo tener 4095 pasos de este voltaje, el setpoint de color amarillo y error existente en el proceso en color verde.

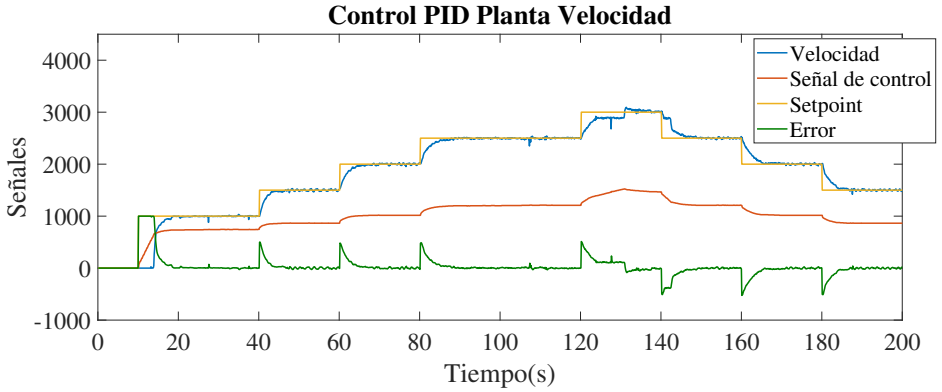


Figura 6.25: Repuesta de control PID ante cambios de Setpoint

Fuente: AUTOR

al aplicarse el primer valor de valor deseado en un valor de 1000 rpm, la respuesta de la velocidad incrementa de forma gradual hasta llegar al valor del setpoint, sin generar algun tipo de sobre-paso, llega al valor de estado estacionario en un intervalo de tiempo de de 12.1 segundo de aplicarse el setpoint, siendo un tiempo de respuesta relativamente bueno al valor de rpm asignado, una vez llega al valor de estado estacionario el valor de la velocidad se mantiene en ese punto, sin embargo no se logra mantener un valor de 1000 rpm, si no que se

mantiene oscilando ese valor, por encima y debajo del valor deseado en un rango máximo de 10 rpms generando un error del aproximadamente del aproximadamente 1% siendo un valor despreciable, esto debido a la perdida de resolución que envía el sistema embebido de 12 bits, que es captado por la tarjeta arduino, para replicar su señal en PWM en resolución de 8 bits, por ello la señal no tiende a mantenerse estable, por ello para realizar control a una planta como velocidad, es importante tener buena instrumentación de sensores, como lo usados en la etapa de potencia.
cuando se realizan cambios de setpoints de 1000 rpm hasta el valor de 1500, el tiempo de respuesta para llegar la variable de velocidad a estado estacionario reduce unos segundos, a un valor de 10.3 segundos, este valor cambiar del primero por un tiempo corto, debido a la influencia de la inercia de debe romper en el primer setpoint, cuando el motor se encuentra en una velocidad 0, de igual forma, el control PID, responde correctamente a la solicitud del setpoint sin llegar a generar algún sopre-pico en la señal, este mismo comportamiento y tiempos de estabilización se mantienen constantes, solo por frac iones de segundos, en los valor de setpoint de 2000,2500 rpms
en el valor deseado de 3000 rpm, se visualiza un comportamiento inusual en el comportamiento de la gráfica, porque la variable de velocidad aumenta gradualmente, hasta un intervalo de tiempo y se queda en un valor de 2880 rpm tratando de subir lentamente, pasado un tiempo este es interpretado este valor aumenta generando un sobre pico,por el controlador como una perturbación en el sistema, trato de corregidlo rápidamente, sobre pasando el valor deseado en 2% del valor, pero corrigiendo llegando al valor antes de los cambios de setpoint, el mismo comportamiento se visualiza en la gráfica de control en ese punto, aumentando su señal de control para corregir ese error. este comportamiento de que la variable aumente rápidamente y se quede en un valor bajo del valor deseado.este comportamiento es debido a problemas físicos de la planta o el valor de pwm en ese punto solo hace efecto hasta sobre pasar cierto valor, pero esto es propio de la planta en ese punto de operación.
el comportamiento de la respuesta de la variable de velocidad de ahora decrementando el valor de setpoint, se visualiza el error un error negativo, porque se esta sobre el valor deseado, la señal de control se hace mas pequeña para que la velocidad sea reducida, y estabilizandose en un tiempo similar al visto en los setpoint de modo incremental, un valor de 11.7 segundos bastante cercado a los demas valores, sin generar sobrepico, por debajo de la señal.
este buen comportamiento del controlador en la planta de velocidad, es debido, a una buena sintonización de las constantes del pid, ya que se modelo el comportamiento con los datos de la planta real, sin ello fuese sido difícil llegar a sintonizar de modo tan óptimo.

6.4.2. Respuesta del Controlador Neuronal multi-capa

La topología usada para aplicarse en esta planta, fue de 4 capas, 3 neuronas en la capa de entrada, 3 neuronas en la primera capa oculta con función de activación tansig, 2 neuronas en la segunda capa oculta con función de activación logsig y 1 neurona de salida con función de activación poslim_limitada, con un factor de aprendizaje encontrado iterativamente de 0.008, esta topología descrita puede observarse en la figura 6.26

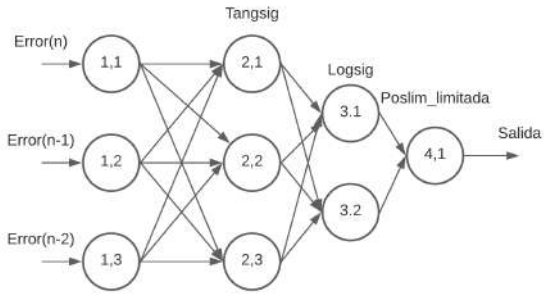


Figura 6.26: Topología de Red Neuronal como controlador planta velocidad
Fuente: AUTOR **Fuente:** AUTOR

este controlador neuronal se evaluó en los mismos valores de setpoints e intervalos de tiempo que fue evaluado el controlador clásico PID, envidenciando la respuesta de la planta con este tipo de controlador inmerso en el sistema embebido en la figura 6.27

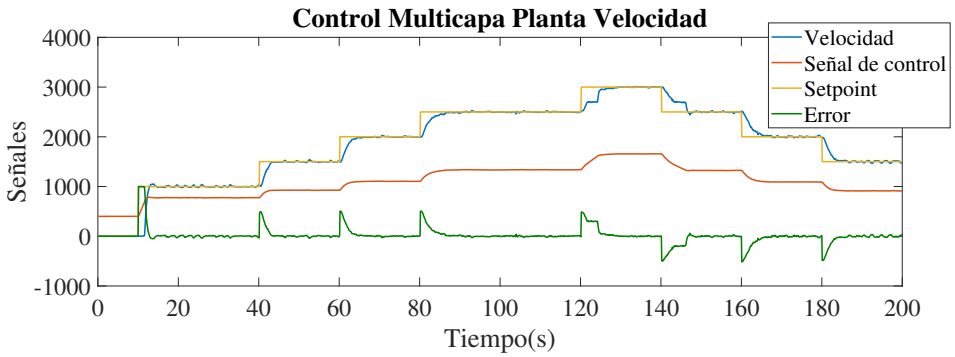


Figura 6.27: Repuesta de control del controlador Neuronal ante varios setpoint

la señal de control opera en un rango de salida de 0-10V, lo que seria imposible de apreciar si su comportamiento cambia,por lo que se uso el valor de la resolución que necesita el controlador para generar este tipo de salida, la resolución es de 12bits.
El comportamiento de la respuesta del control neuronal multi-capa, ante el primer cambios de setpoint de 1000, responde rápidamente para corregir el error por medio de la señal de control que varia rápidamente, logrando tener un tiempo de estabilización estimado de 4.5 segundos, para llevar la variable de velocidad a un estado estacionario,se genera un sobrepico de 44 rpm en un rango del 4.4% siendo un valor que puede entrar entre valores despreciables porque se recupera rápido entrando a una respuesta en estado estacionario, es normal que la señal tienda a oscilar un poco, en un rango de 11 rpms por encima del valor deseado y del aproximadamente el mismo valor por debajo de la señal de referencia. pero manteniéndose entre los valores del setpoint,estos factores son dados por la etapa de potencia. debido a la resolución de salida que permiten tener estos elementos que se usaron, de este modo no se logra sacar el mayor eficiencia de la resolución de 12 bits que entrega el sistema embebido, la etapa de potencia opera con una resolución de 8 bits, por lo que seria natural que se presente

este tipo de comportamiento en estado estacionario, teniendo un error del 1.1% que se podría decir que es un valor despreciable para el rango de valor que se esta pidiendo que se lleve la variable de velocidad.
para los demás cambios de setpoint, se mantuvo una relación cercana en el tiempo de la variable de velocidad al valor establecido,realizando control en los siguientes valores de setpoint, puntos a evaluar que efectivamente llega al valor deseado con un error del aproximadamente del 1.1%, sin presentar sobre-picos en las respuestas.
el factor de aprendizaje utilizado en esta planta es un valor relativamente bajo, pero en compensación a ello se realizan tiempos de muestreo de 100mS, por lo que la señal de control con respecto al tiempo cambiara rápidamente, se generarían mayor numero de iteraciones, con la relación de la dinámica de la planta.
de la misma manera que se presenta un error del comportamiento de la planta, en el cambio de setpoint de 3000 rpm, se visualiza que en un punto la señal trada de quedarse en un punto,pero a lo que la señal de control aumenta es corregido de manera mas eficiente que lo sucedido en el control PID, llegando al valor deseado sin generar algun tipo de sobrepico, manteniendo la señal en el rango del valor deseado.

6.4.3. Comparativa de controladores en la planta de velocidad

se tomaron los datos de la respuesta del comportamiento de la planta con el controlador PID, visto en la figura 6.25, con los datos de la respuesta del comportamiento con el controlador neuronal multicapa visto en la figura 6.27, anexándolo en una sola gráfica, de este modo evidenciar cual de los dos controladores entregan una mejor respuesta en menor tiempo, sin presentar sobre oscilaciones.
en la figura 6.28, se aprecia la respuesta del control PID de color azul, la respuesta del la red neuronal de color naranja y el valor del setpoint en color amarillo.

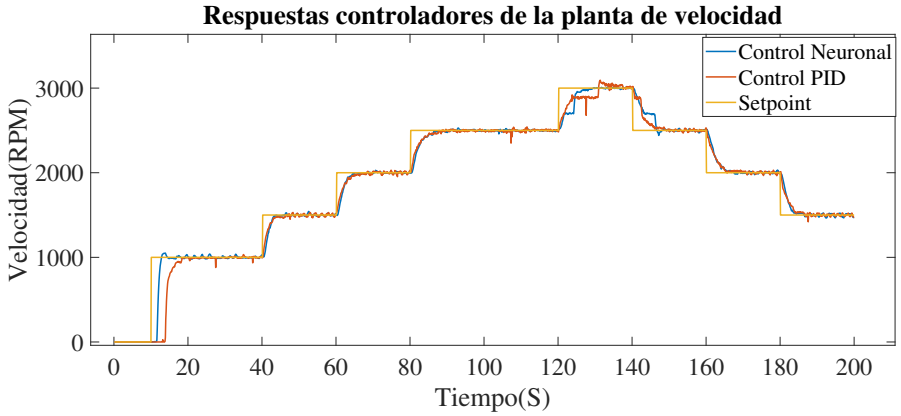


Figura 6.28: Repuesta del controlador PID, y control Neuronal
Fuente: AUTOR **Fuente:** AUTOR

Al aplicarse el setpoint, se aprecia notoriamente que el valor de respuesta de la velocidad entregado por el controlador neuronal, tiene mejores tiempos de estabilización. en tiempos de 4.5 segundos, mientras que el PID cuenta con tiempos de 12 segundos. aunque puede que el

PID le hace falta sintonizares de manera correcta, esos fueron los valores que mejor respuesta dieron al modelo de identificación que se le realizo, sin entregar sobre-picos que desestabilizan la respuesta,aunque se usaron los mismos tiempos de muestreo en el PID y la red multicapa, se desempeña de manera óptima el control neuronal
aunque en relación del trabajo que se realizo para hallar los valores de constantes PID, el controlador neuronal para una planta con dinámica rápida como la de velocidad, solo se deben tener en cuenta parámetros como el numero de capas y neuronas usar, función de activación y la importancia del tiempo de muestreo, evitando un trabajo de más, del cual se debe volver a sintonizar la planta donde las constantes no operen debidamente.
por lo que en esta comparativa el controlador que presento mayor desempeño fue el control neuronal multicapa, con factor de aprendizaje de 0.008 y tiempos de muestreo de 1 segundo.

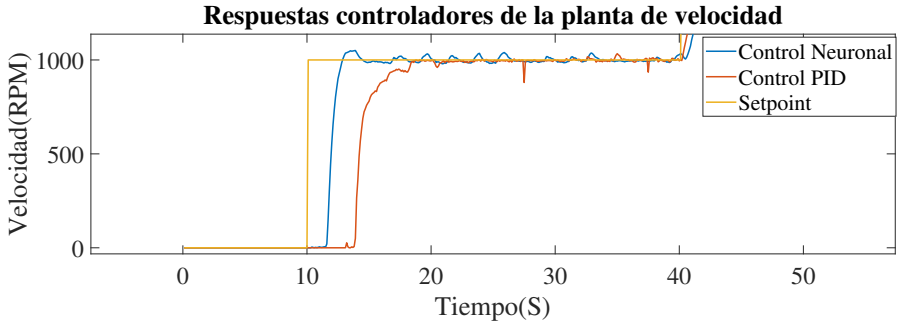


Figura 6.29: Repuesta del controlador PID, y control Neuronal setpoint 1000

se le hizo súper zoom a a la figura 6.28, de este modo visualizar mejor el comportamiento de la respuesta de los dos controladores, la respuesta en el setpoint de 1000 visto en la figura 6.29, evidencia que el controlador neuronal, mantiene la variable velocidad en 0 por un corto periodo de tiempo, mientras la señal de control es lo suficiente para romper la inercia existente en el motor, de igual manera se visualiza un comportamiento similar en el control PID, sin embargo le tarda mas vencer la inercia al control PID, lo que le genera tiempo de retraso en su respuesta, y aunque el control PID llegara a romper esa inercia en el mismo punto que el control neuronal, existe un error antes de llegar la respuesta a estado estacionario, por ello tiene tiempos tan largos de estabilización.
se visualiza unas oscilaciones sobre y por debajo del setpoint en el neuronal, debido a la perdida de resolución de la señal de control, y tambien a la resolucion de lectura del sensor encoder, este comportamiento es similar en el PID, una vez se situa la variable de velocidad sobre el setpoint, antes del cambio de setpoint.

6.4.3.1. Comportamiento de la respuesta del Control Neuronal ante valores del factor de aprendizaje distintos

para comprobar la influencia del comportamiento de la respuesta de la planta de velocidad, con una misma topología tomada de la imagine 6.26 pero con diferentes factores de aprendizaje, se escogió esta planta para realizar las respectivas pruebas, ya que su dinámica de la variable a controlar es rápida, permitiendo tomar varios datos en menos tiempo, de este modo evidenciar la afectación que tiene este factor en la respuesta de la planta, se utilizaron

valores por encima y por debajo del valor factor de aprendizaje de 0.008 utilizado para el control en diferentes valores de setpoint.
en la figura 6.30, se visualizara el comportamiento de la planta de velocidad ante los factores de aprendizaje de Fa: 0.006 representado de color Magenta, Fa:0.0007 siendo representado por el color azul, Fa:0.008 representado por el color verde, Fa:0.009 representado por el color cyan, Fa:0.015 representado por el color rojo, Fa:0.035 representado por el color gris y el valor del setpoint o valor deseado de color amarillo

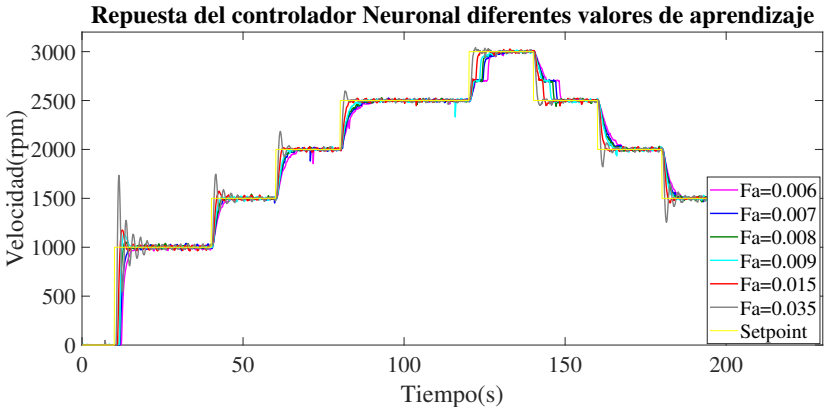


Figura 6.30: Respuesta control Neuronal bajo diferente valor de factor de aprendizaje

Fuente: AUTOR

en la figura 6.30, se visualizan 6 señales con comportamiento distinto, tanto al iniciar el proceso con el primer valor del setpoint, como en los distinto puntos del valor deseado, por lo que para poder apreciar mejor la respuesta, se le hará el análisis, del primer valor del setpoint, el segundo punto a evaluar sera el valor 2500 rpm en el inetvalo de tiempo de 80 s, el tercero el valor de 1500 rpm en decenos en el intervalo de tiempo de 150 .

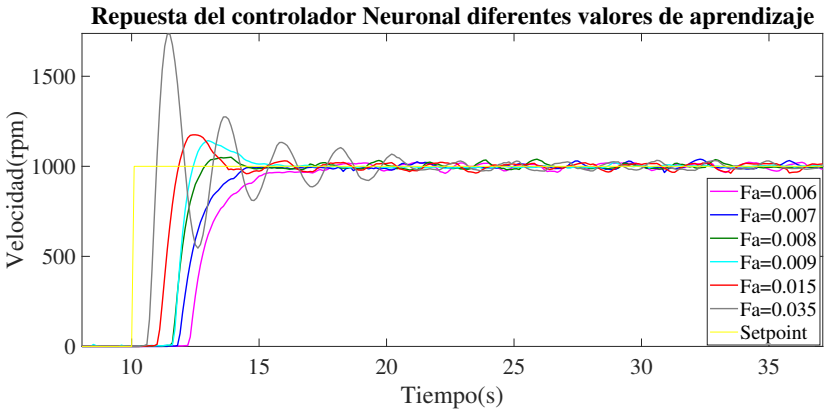


Figura 6.31: Respuesta control Neuronal bajo diferente valor de factor de aprendizaje setpoint de 1000 rpm

Fuente: AUTOR

para realizar el análisis del primer punto del valor de setpoint, se tomo un fragmento de tiempo en los intervalos de 0 hasta el tiempo 35, tal como se ve en la siguiente 6.31
El comportamiento esperado de los factores de aprendizaje cercanos a cero, es representando en la respuesta con tiempos de estabilización mayores, sin generar sobre-picos, pero le toma un poco mas de tiempo llegar al valor deseado, mientras se va incrementando el valor de aprendizaje, se evidencia que la variable de velocidad poco a poco va llegando de manera mas rápida al valor deseado, hasta llegar a un factor donde la respuesta genera sobre picos de gran magnitud y sobre oscilaciones en el sistema hasta estabilizarse.
el factor de aprendizaje de 0.006 induce una respuesta en el sistemas estable, llegando al valor de estado estacionario en un valor de 8 segundos, siendo un tiempo de respuesta rapido y limpio para el sistema, sin sobrepico. También se observa que en estado estacionario la respuesta oscila sobre valores por encima y por debajo del setpoint, influencia de los dispositivos de acondicionamiento y su baja resolución, que afecta a todas las repuestas de factor de aprendizaje.

Factor aprendizaje	**Tiempo estabilizan (s)**	**porcentaje de sobre pico (%)**
Fa:0.006	8	0
Fa:0.007	5.4	0
Fa:0.008	4.5	1.1
Fa:0.009	6.7	13.8
Fa:0.015	15.1	17.41
Fa:0.035	25.2	80

Tabla 6.3: Tiempos de estabilizan control neuronal con factor de aprendizaje distinto setpoint 1000
Fuente: AUTOR

el factor de aprendizaje de 0.007 genera una respuesta sin llegarse a pasar del valor deseado en tiempo de estabilización de 5.4 segundos, a medida que se aumenta el factor el tiempo de estabilización se va haciendo menor.
El factor de aprendizaje de 0.008 el que fue utilizado para la prueba de control neuronal vs el PID fue el que mejor dio desempeño en tiempo para llevar la respuesta a estado estacionario con un tiempo de 4.5 segundos con un leve sobre pico del 1.1%. a partir de este valor de aprendizaje la respuesta comienza a tener mayor sobre-pico y aunque la variable de velocidad llegue mas rápido al valor deseado, se genera ese sobre pico y los tiempos de estabilización se tendrían hacer mas grandes. en la tabla 6.3 se visualiza todos los tiempo de estabilización y sobre pico máximo en la respuesta, el factor de aprendizaje 0.035 no llega a estabilizar y queda oscilando la respuesta en un porcentaje del error del 13%.
para el segundo análisis del comportamiento de los factores de aprendizaje en el punto de operación de 2500 rpm en el intervalo de tiempo de 80 segundos se toma los datos de ese punto y visto en la figura 6.32

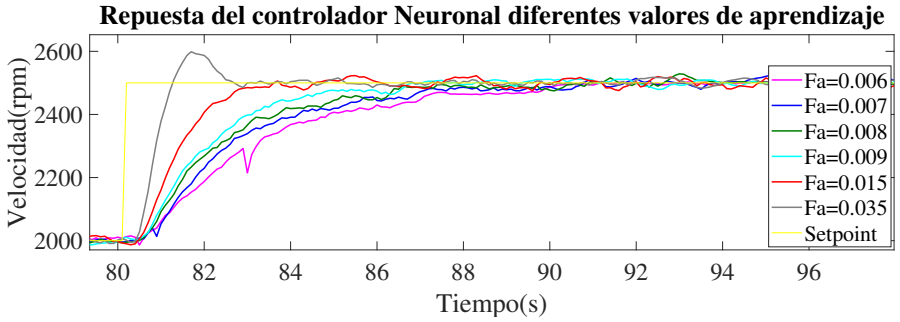


Figura 6.32: Respuesta control Neuronal bajo diferente valor de factor de aprendizaje setpoint de 2500 rpm

Fuente: AUTOR

en este punto de operación la variable a controlar o setpoint se encuentra en un valor mas alto, el cambio de setpoint se realiza con el motor en marcha, por lo que la respuesta ya no partería de cero, de este no tiene que vencer inercia existente, por lo que en marcha el comportamiento de la respuesta varia un poco, anteriormente el valor de factor de aprendizaje de 0.035 no se logro estabilizar o le hizo falta tiempo, sin embargo en este punto se genera un sobre pico, pero estabilizándose en el valor deseado, ya que la variable de control no debe elevarse tanto para corregir el error, en comparación cuando se inicio con el motor en velocidad 0. los tiempos de estabilización de este punto de operación son los vistos en la tabla 6.4.
lo valores de estabilización en este punto cambiaron un par de segundos, debido a que el sistema se encuentra en un punto de operación diferentes, debido a la dinámica de la planta, como opera en un punto de revoluciones altas, no opera de igual forma que en un punto de revoluciones baja.
solo se visualizo sobrepaso de la señal de la variable de velocidad de velocidad con el factor de aprendizaje de 0.0035, con un error del 6.4%

Factor aprendizaje	**Tiempo estabilizan (s)**	**porcentaje de sobre pico (%)**
Fa:0.006	11.2	0
Fa:0.007	10.8	0
Fa:0.008	7.3	0
Fa:0.009	6.8	0
Fa:0.015	3.7	0
Fa:0.035	8.8	6.4

Tabla 6.4: Tiempos de estabilizan control neuronal con factor de aprendizaje distinto setpoint 2500

Fuente: AUTOR

el tercer punto a verificar el comportamiento de la respuesta bajo diferente factor de aprendizaje, es observar la respuesta de un setpoint mayor a un setpoint menor, tal como se visualiza en la figura que parte de un valor de setpoint de 2000 rpm a un valor de 1500 rpm

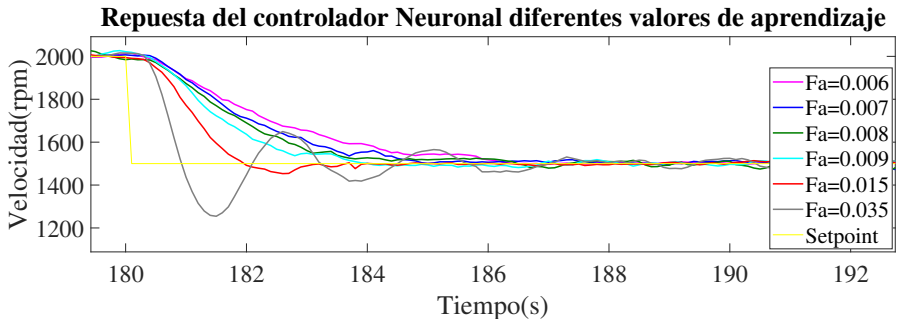


Figura 6.33: Respuesta control Neuronal bajo diferente valor de factor de aprendizaje setpoint de 2500 rpm a 1500 rpm

Fuente: AUTOR

la respuesta del controlador en ese punto de operación, la señal con factor de 0.035 y 0.015 presentan oscilaciones en su respuesta, influencia de la dinámica de la planta, y la inercia de caída que lleva la velocidad, los tiempos de estabilización y porcentaje se sobre pico se puede ver en la tabla 6.5

Factor aprendizaje	Tiempo estabilizan (s)	porcentaje de sobre pico (%)
Fa:0.006	6.6	0
Fa:0.007	5.6	0
Fa:0.008	4.8	0
Fa:0.009	4.5	0
Fa:0.015	4.3	3
Fa:0.035	8.8	16.4

Tabla 6.5: Tiempos de estabilizan control neuronal con factor de aprendizaje distinto setpoint 2500 rpm a 1500 rpm

Fuente: AUTOR

6.5. Implementación de Controladores en la Planta de Nivel

los controladores a usar en la planta de nivel, por parte de controladores clásico, un controlador PI.para el controlador neuronal una red neuronal de 3 capas.
se implementara un control PI, con las constantes respectivamente halladas en tuner de matlab anteriormente, la data se tomo por puerto serial del sistema embebido para luego así poder hacerles el respectivo análisis.

6.5.1. Respuesta del Controlador PI

como en las demás planta anteriores se implemento un controlador clásico, para su validación y comprobar el funcionamiento del controlador neuronal. Esta planta también es conveniente aplicarle este tipo de control. debido por su dinámica lenta, como en el comportamiento de la planta de temperatura, se opto usar un controlador PI, proporcional integra.

como ya contamos con estas constantes a partir de la identificación de la planta fueron tomadas las constantes y de ese modo evidenciar su comportamiento en pruebas reales. las constantes del PI son las siguientes Kp=990.928 y para Ki=400.56, los de setpoint utilizados para la evaluación comienza en un nivel de 2cm pasando a un nivel 4 cm a un nivel de 6 cm, hasta llegar al nivel de 8 cm.

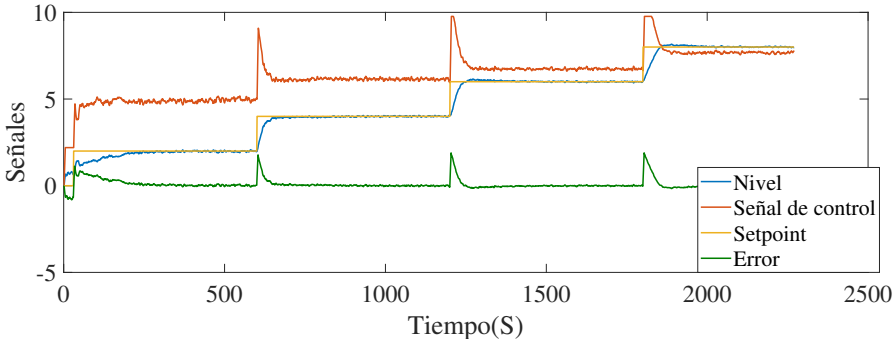


Figura 6.34: Repuesta de control PI Planta Nivel ante cambios de Setpoint

Fuente: AUTOR

La respuesta del control PI en la planta de nivel se puede apreciar el comportamiento dado por este controlador en la figura 6.34, su condiciones físicas de un valor minino de 0 cm y un 11 cm siendo el nivel máximo de agua permitido, con una señal en alto del sistema embebido. la aplicación de este controlador de realizo en tiempos de muestreo de 1s, en la figura se aprecia el nivel dado en cm de color azul, la señal de control de color naranja, el valor deseado o setpoint de color amarillo, y el error de color verde.
se evidencia a simple vista que el controlador PI cumple su funcion de llevar el nivel de agua al valor el setpoint establecido.
analizando el primer setpoint de un nivel de agua de 2cm, el tiempo que tarda y llegar a un valor de estado estacionario es unos 250 segundo o de 4.1 minutos, con una respuesta limpia sin sobre picos sobre el valor deseado. también se puede evidenciar un ruido oscilatorio debido a la resolución de baja calidad que tiene el sensor sharp, y lo susceptible que puede ser al ruido externo haciendo variar la señal de salida en un rango de 0.08, aunque el ruido es casi inapreciable esta en un porcertaje de 4%, sin embargo el control PI cumple su función de llevar la variable de nivel al valor impuesto.
respectivamente se aprecia la señal de control el como detectar un cambio de setpoint envía un aumento su valor a la salida del sistema embebido debido a que cuenta con una constante proporcional de un valor elevado ayudando a que la respuesta sea mas rápida, esta señal es recibida por el convertidor de señal analógica a señal PWM y enviada a la bomba el elemento final de control que regula su caudal de agua en relacion al valor del dutty del PWM, esta respuesta tan limpia es debido a una buena identificación y sintonización realizada anteriormente, denotando que el valor de error llega a cero.

6.5.2. Respuesta del Controlador neuronal multicapa

la topología dispuesta para el control neuronal multi capa en la planta de nivel, esta dada por 3 neuronas en la capa de entrada, 2 neuronas en la primera capa oculta con función activación tangisg, 2 neuronas en la segunda capa oculta con función logisig y 1 neurona de salida con función de activación poslim_limitada, el factor de aprendizaje utilizado y que mejor dio respuesta para la dinámica de esta planta fue el valor de 0.01582, con un tiempo de muestreo de 1 segundo en la toma y procesamiento de los datos. la topología anteriormente descrita puede visualizarse en la figura 6.35

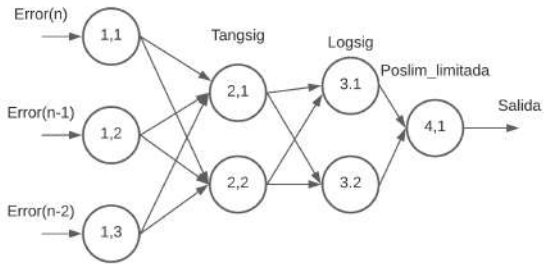


Figura 6.35: Topología de Red Neuronal como controlador planta Nivel
Fuente: AUTOR

se implemento esta topología de red neuronal en el sistema embebido para el control de nivel en cm de agua de la planta, la variables se representaran de la siguiente manera, de color azul el nivel en cm registrado por el sensor sharp, de color naranja la señal de control respectivamente con la señal de voltaje que brinda el sistema embebido, de color amarillo el valor de referencia o setpoint, y el color ver el error registrado en el proceso. en la figura 6.36 El control de nivel, fue la planta que demostró poco desempeño ante el controlador neuronal,esto es resultado de la dinámica de la planta, siendo una dinámica mas rápida que la vista en el control de temperatura, y mucho mas lenta que la de control de velocidad.

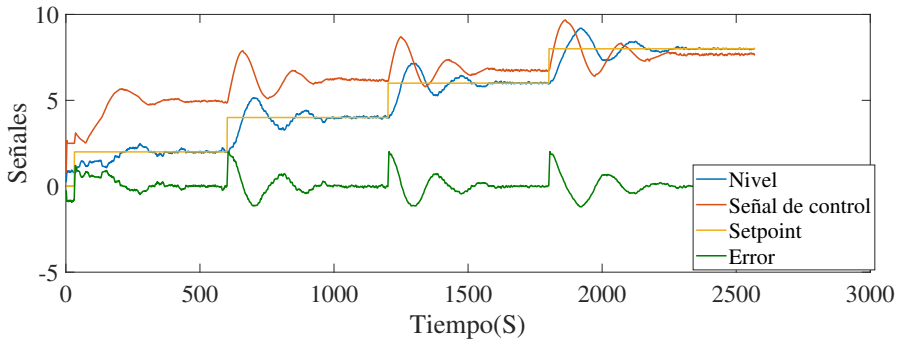


Figura 6.36: Repuesta de control Neuronal Planta Nivel ante cambios de Setpoint
Fuente: AUTOR

al aplicarse el valor de referencia, en un valor de 2cm, la variable de nivel llega a la respuesta

de estado estacionario en un tiempo de 6 minutos a partir del cambio del setpoint, aunque se genera un sobre-pico en la repuesta por encima del setpoint del 21%, y nuevamente otro sobrepico al bajar la señal de 13%, siendo esta las dos afectaciones, pero a partir de esos dos sobrepicos el controlador mantiene la la variable de nivel sobre el valor deseado. sin embargo se aprecia un poco de ruido esto generado por el sensor de distancia.
para el segundo cambio de setpoint a un valor de 4 cm en el intervalo de tiempo de 600 segundo, nuevamente la variable de velocidad presenta otro sobrepico, sobre el valor deseado, el porcentaje del sobrepico esta en el 22% ,pero esta vez presenta otro dos sobrepico mas al tratar de recuperarse, siendo del 17% de valor deseado, la relación de porcentaje es alta, pues se esta trabajando en una escala pequeña, y por influencia de la sensorica aun menta mas el error en la señal, antes de la variable de nivel entrar a valor de estado estacionario, se genera un nuevo sobrpico, esta vez es menos pronunciado este valor en el rango del 9% , desde del tiempo que se aplico el cambio de setpoint al tiempo que llego a estabilizarse la señal le tomo 7 minutos. esta misma relación se mantiene con los demás valores del setpoint.

6.5.2.1. Comparativa de controladores en la planta de Nivel

la comparativa del control clásico PI y control neuronal multicapa de la planta de nivel sera reflejado en este apartado, tomando los datos de las gráficas anteriores y uniéndolas en una sola, para de este modo realizar una correcta comparativa, los datos tomados con el control PI y el control Neuronal, mantienen los mismos setpoint, y mismo tiempos de cambio de valor deseado, tal como se aprecia en la figura 6.37, de color azul se representa la respuesta entregada por el control neuronal, en color naranja la respuesta entregada por el control PI, la escala de las señales están dada en cm.

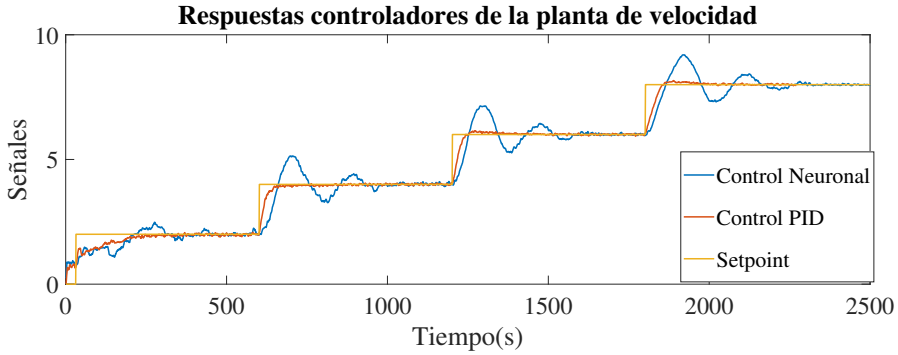


Figura 6.37: Repuesta de control Neuronal Planta Nivel ante cambios de Setpoint
Fuente: AUTOR

a simple vista el controlador PI, demuestra tener gran desempeño en este planta con tiempo de estabilización en su primer setpoint de 4.1 minutos, mientras el controlador neuronal un tiempo de 6 minutos, presentado sobre oscilaciones , por lo que el control que mejor se comporto para esta planta es el control PI llevando la variable de nivel al nivel deseado, por tener la capacidad de predecir su repuesta en cierto punto, mientras el controlador neuronal por la dinamica de la planta que es un poco mas rapida que la plante de temperatura, le cuesta bajar la señal de control rápidamente con el factor de aprendizaje impuesto, por ello la señal

se generan unos sobrepicos, que ademas la influencia de una mala lectura en el sensor se le ve afecta al control neuronal, generando señales de salida falsas, creyendo que la variable de nivel esta sobre el nivel deseado cuando, en realidad son defectos provocados por el ruido del sensor.

aunque el controlador no se desempeño de manera efectiva en esta planta, como en las demás, es debido a las instrumentación, pero demostrando que pasado un tiempo con unas sobre-picos máximo de 22% el control neuronal llega a su valor deseado, mantiene su respuesta en ese punto.

6.6. Ficha Técnica

Cumpliendo a los objetivos establecidos, se redacto una ficha técnica de las características físicas, puertos de entrada y salida, como del manejo del dispositivos y uso de la interfaz realizada. En los archivos anexados al libro de trabajo de grado va un documento llamado **libro_ficha_tecnica_sistema_embebido**, describe los elementos asociados al uso y el modo de conexión que se debe realizar para el acondicionamiento del sensor.

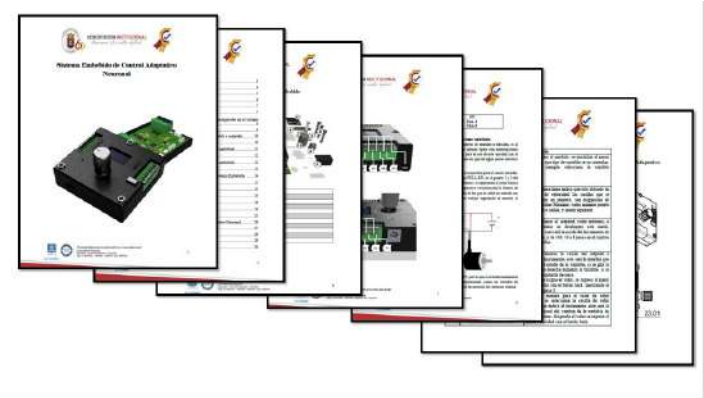


Figura 6.38: Ficha técnica
Fuente: AUTOR

6.7. Resultados Extras

En esta Sección de resultados extras se enunciaran los resultados obtenidos por la librería desarrollada Neural_Networks_FF, para el entrenamiento de una red neuronal en tarjeta de desarrollo arduino. usando los 4 ejemplos que contiene esta librería, como aproximador de funciones y como clasificador, los datos fueron tomados por puerto serial guardados en un archivo de excel, debido la resolución tan baja que ofrece el serial.ploter del IDE de arduino para visualización de las figuras.

6.7.1. Resultados de la librería como aproximador de función seno

El siguiente código 6.1 consiste, en replicar el comportamiento de la función sin(x)+5, en una topología de red neuronal con 3 capas, 4 neuronas en su capa de entrada, 8 neuronas en

su capa oculta y 1 neurona en su capa de salida, este entrenamiento se realiza con 68 patrones, y dos condiciones de salida, si el error global es inferior a 0.0001 se termina el entrenamiento de la red o con un total de 5000 iteraciones si la condición del error no llega ser cumplida. el resultado de la salida de la neuronal se visualiza en la figura 6.39, en conjunto con la función seno, para una respectiva validación.

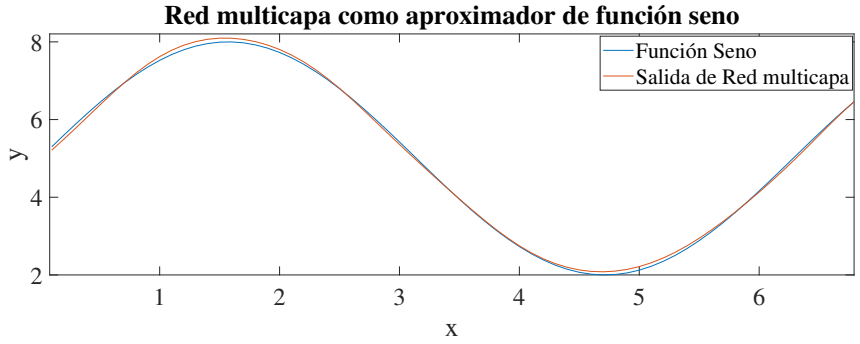


Figura 6.39: Salida de la red neuronal multicapa como aproximador de función seno
Fuente: AUTOR

La respuesta generada por la red neuronal. Efectivamente replica el comportamiento de la función seno en un total de iteraciones de 291 que tardo un tiempo promedio de 5 minutos usando la tarjeta de desarrollo genuino zero y con un error global de 0.00009792, comprobando de este modo que la topología usada y la librería desarrollada operan del modo correcto, aunque se ve un desfase en la respuesta de salida de la red, debido que se uso una condición de salida un poco baja, y si se quiere una aproximación mas exacta, simplemente es hacer mas estricto la condición de salida del error colocando el error de salida aun mas cercano a cero, sin embargo esto ocasionaría mayor numero de iteraciones y demora en en el entrenamiento. aunque si se requiere precisión en la respuesta, el tiempo que le tome a red entrenar solo se realiza una única vez. porque se guardan los pesos asociados a la red, y simplemente seria ser evaluada.
el uso de replicar el comportamiento de una señal para el ejemplo el comportamiento de la función seno, también permite ser usado en identificación de modelos de plantas, en predicción de datos o clonación de controladores entre otras aplicaciones que permiten ser utilizadas las redes neuronales.

```
#include<Neural_Networks_FF.h>
Neural_Networks_FF net = Neural_Networks_FF();
Dynamic_Array XI = Dynamic_Array();
Dynamic_Array D = Dynamic_Array();
Dynamic_Array FAC_STR = Dynamic_Array();
Dynamic_Array EST = Dynamic_Array();
void setup() {
  delay(3000);
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  randomSeed(analogRead(0));
  XI.NewArray_2D(68, 4);
```

```
13  XI.ToinitializeArray_2D(0.0);
14  D.NewArray_2D(1, 68);
15  D.ToinitializeArray_2D(0.0);
16  for (int i = 0; i < 68 ; i++){
17    XI.WriteArray_2D(i, 2, i * 0.1);
18    XI.WriteArray_2D(i, 3, 3.0 * sin(i * 0.1) + 5.0); }
19  for (int i = 1; i < 69 ; i++) {
20    XI.WriteArray_2D(i - 1, 0, i * 0.1);
21    XI.WriteArray_2D(i - 1, 1, 3.0 * sin(i * 0.1) + 5.0);
22    D.WriteArray_2D(0, i - 1, 3.0 * sin(i * 0.1) + 5.0); }
23  byte XC = XI.GetColumns();
24  byte DF = D.GetRows();
25  byte XF = XI.GetRows();
26  net.DATA_CENTERING(XI);
27  net.NORMALIZE_DATA(XI, XI.MaxArray_2D());
28  byte L = 3;
29  EST.NewArray_2D(1, L);
30  EST.ToinitializeArray_2D(0.0);
31  float NC = EST.GetColumns();
32  float V[] = {XC, 8, DF};
33  for (byte i = 0; i < NC; i++)  {
34    EST.WriteArray_2D(0, i, V[i]); }
35  FAC_STR.CharNewArray_1D(L);
36  char *myStrings[] = {"tansig", "tansig", "purelin"};
37  for (byte i = 0; i < FAC_STR.GetRows(); i++){
38    FAC_STR.CharWriteArray_1D(i, myStrings[i]); }
39  net.FEED_FORWARD_NET(EST, FAC_STR);
40  net.TRAIN_NET(XI, D, 1e-4, 5000);
41   net.SIM_NET(XI);
42  float j = 0;
43  for (byte i = 0; i < XF ; i++) {
44    j = j + 0.1;
45    Serial.println(D.ReadArray_2D(0, i), 4);
46    Serial.print(" ");
47    Serial.println(net.Ysim.ReadArray_2D(0, i), 4); }
48     //Serial.println(net.EPOCHS);
49  //Serial.println(net.ERROR_C_M);
50  //  Serial.println("Error_n_ecpoas");
51 }
52 void loop() {
53 }
```

Código 6.1: Aproximador de función seno de la librería Neural_Networks_FF

6.7.2. Resultados de la librería como aproximador de función Coseno

El codigo usado para la validacion de salida de la red neuronal como aproximador de la funcion coseno, es el código 5.6 explicado en el desarrollo. El contiene la misma topología establecida en la aproximación de función seno, con el mismo numero del error asociado a la condición de salida, este código código fue cargado y ejecuta generando la salida en la red neuronal vista en la figura 6.40

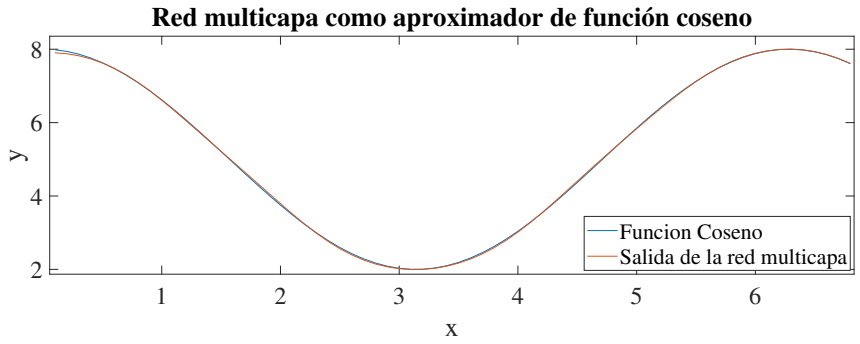


Figura 6.40: Salida de la red neuronal multicapa como aproximador de función coseno
Fuente: AUTOR

Se evidencia una correcta aproximación a la función de la red neuronal multicapa a la función coseno, apreciando que una gráfica se sobre-pone sobre la otra, el error global existente en la red neuronal fue de 0.00009845 con un total de 385 iteraciones en un tiempo de 6.3 minutos, manteniendo un tiempo cercado al evaluado anteriormente, aunque cable aclarar que si se realiza un nuevo entrenamiento, la red neuronal tomaría otro camino de entrenamiento debido a que los pesos son aleatorios en cada ejecución del programa.

6.7.3. Resultados de la librería como aproximador de función Coseno y seno

La validación de esta aproximación de las funciones seno y coseno, se hace mediante una sola topología de red, tomando como entradas, los puntos a evaluar de la función seno, con sus respectivos retardos en la entrada, la topología cuenta con 3 capas, 4 neuronas en la capa de entrada, 8 neuronas en la capa oculta y 2 neuronas en la capa de salida. Donde una salida representa la aproximación de la función seno, y la segunda salida la aproximación de la función coseno, la programación usada que fue cargada en la tarjeta de desarrollo se puede ver en el código 6.2, generando como resulta, la respuesta vista en la figura 6.41, donde se aprecia 4 gráficas, la de color azul representa la función seno y la gráfica de color rojo representa la salida de la neurona 1 de la capa de salia como aproximador de función se, la gráfica de color verde representa la función coseno, y la gráfica de color magenta representa la salida de la neurona numero 2 de la red con la aproximación de la función coseno.

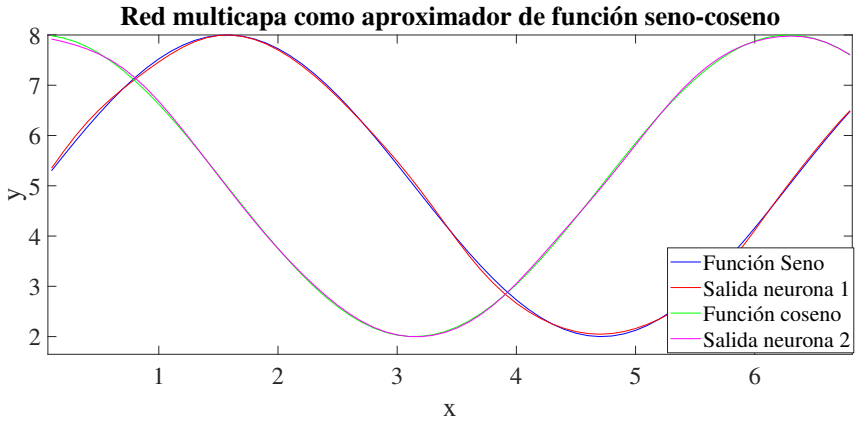


Figura 6.41: Salida de la red neuronal multicapa como aproximador de función seno-coseno con dos salida en su ultima capa

Fuente: AUTOR

en la gráfica anterior se visualizan 4 gráficas, siendo la función seno y coseno, a modo de validación que efectivamente se esta realizando la aproximación, las gráficas salida neurona 1, representa la aproximación de la función seno, la gráfica salina neurona 2 representa la aproximación de la función coseno.
la aproximación realizada por la red neuronal muestra resultados satisfactorios, sin embargo para entrenar la red tarda un tiempo aproximado de 30 minutos, con un error de condición de salida del 0.0001, debido a que se tienen dos salidas de la misma red neuronal la corrección del error global se vuelve mas prolongado, a comparación de la aproximación la función seno o coseno que tiene una única salida, al realizar la pruebas se evidencio que la red neuronal termina el entrenamiento por medio de la condición de salida, si no por las 5000 iteraciones asignadas generando un error global de 0.001879.esto demuestra que las dos condiciones funcionan en la red y no se queda en un ciclo infinito tratando de encontrar la solución.

```
#include<Neural_Networks_FF.h>
Neural_Networks_FF net = Neural_Networks_FF();
Dynamic_Array XI = Dynamic_Array();
Dynamic_Array D = Dynamic_Array();
Dynamic_Array FAC_STR = Dynamic_Array();
Dynamic_Array EST = Dynamic_Array();

void setup() {
  delay(3000);
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  randomSeed(analogRead(0));
  XI.NewArray_2D(68, 4);
  XI.ToinitializeArray_2D(0.0);
  D.NewArray_2D(2, 68);
  D.ToinitializeArray_2D(0.0);
  for (int i = 0; i < 68 ; i++)  {
    XI.WriteArray_2D(i, 2, i * 0.1);
    XI.WriteArray_2D(i, 3, 3.0 * sin(i * 0.1) + 5.0);
  }
```

```
21  for (int i = 1; i < 69 ; i++)  {
22    XI.WriteArray_2D(i - 1, 0, i * 0.1);
23    XI.WriteArray_2D(i - 1, 1, 3.0 * sin(i * 0.1) + 5.0);
24    D.WriteArray_2D(0, i - 1, 3.0 * sin(i * 0.1) + 5.0);
25    D.WriteArray_2D(1, i - 1, 3.0 * cos(i * 0.1) + 5.0);
26  }
27  byte XC = XI.GetColumns();
28  byte DF = D.GetRows();
29  byte XF = XI.GetRows();
30  net.DATA_CENTERING(XI);
31  net.NORMALIZE_DATA(XI, XI.MaxArray_2D());
32  byte L = 3;
33  EST.NewArray_2D(1, L);
34  EST.ToinitializeArray_2D(0.0);
35  float NC = EST.GetColumns();
36  float V[] = {XC, 8, DF};
37  for (byte i = 0; i < NC; i++)  {
38    EST.WriteArray_2D(0, i, V[i]);
39  }
40  FAC_STR.CharNewArray_1D(L);
41  char *myStrings[] = {"tansig", "tansig", "purelin"};
42  for (byte i = 0; i < FAC_STR.GetRows(); i++)  {
43    FAC_STR.CharWriteArray_1D(i, myStrings[i]);
44  }
45  net.FEED_FORWARD_NET(EST, FAC_STR);
46  net.TRAIN_NET(XI, D, 1e-4, 5000);
47  net.SIM_NET(XI);
48  float j = 0;
49  //Imprimimos por serial la salida de la red y la salida deseada
50  for (byte i = 0; i < XF ; i++) {
51    j = j + 0.1;
52    Serial.println(D.ReadArray_2D(0, i), 4);
53    Serial.println(net.Ysim.ReadArray_2D(0, i), 4);
54    Serial.println(D.ReadArray_2D(1, i), 4);
55    Serial.println(net.Ysim.ReadArray_2D(1, i), 4);
56    delay(100);
57  }
58 }
59 void loop() {
60 }
```

Código 6.2: Aproximador de función seno y coseno de la librería Neural_Networks_FF

6.7.4. Resultados de la librería como Clasificador

Las redes neuronales a parte de ser utilizadas como aproximador de funciones, predicción o clonación de controladores, permiten entrenar una red neuronal para que de tal modo funcione como clasificador, esto se hace mediante una data de patrones, con N numero de entradas, y N numero de salidas.
El ejemplo usado consta de clasificar una clase A de setosas de la clase B, se tiene un patrón de datos de entrada y salida. el patrón de entrada cuenta con dos parámetros, entrada x1 y x2, en el algoritmo 6.2, los parámetros de entradas están definidos por el vector I[120][2], con 120 filas y 2 columnas, y un vector T[120] con 120 parámetros de salida entre 1 y 0, la clase

A, se representara con el 1 en el código y en el gráfico 6.42 con el color rojo, la clase B se presentara en el código con el valor de 0 y en el gráfico de color azul, con este parámetro de salida se entrenada la red, para que aprendan los patrones y pueda realizar una clasificación. el patrón de entradas simple vista se aprecia que el comportamiento entre una clase y la otra es linealmente separable, por lo que este tipo de aplicaciones se le es sencillo aprender al momento del entrenamiento.

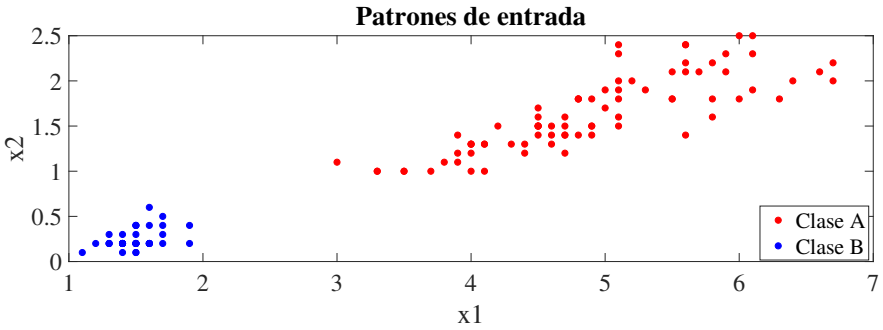


Figura 6.42: Patrones de Entrenamiento para La Red neuronal
Fuente: AUTOR

con este patrón de entrada y clasificación de salida , se entrena la red, con la topología de 3 capas, con dos neuronas en su capa de entrada, 12 neuronas en su capa oculta y función de activación tansig y 1 neurona en su capa de salida con función de activación purelin, el entrenamiento de la red por el gran numero de patrones, en el microcontrolador le tomo un tiempo de 22 minutos las características anteriormente mencionadas, con acierto del 100% en el entrenamiento y validación, con un numero de 542 iteraciones y un error 0.00099.
con el entrenamiento de esta red, se puede realizar clasificación. ingresando algún patrón de entrada en x1 y x2, generando una salida, la cual indicaría a que clase pertenece el patrón ingresado.

```
#include<Neural_Networks_FF.h>
Neural_Networks_FF net = Neural_Networks_FF();
Dynamic_Array XI = Dynamic_Array();
Dynamic_Array D = Dynamic_Array();
Dynamic_Array FAC_STR = Dynamic_Array();
Dynamic_Array EST = Dynamic_Array();
void setup() {
  delay(3000);
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  randomSeed(analogRead(0));
  XI.NewArray_2D(120, 2);
  XI.ToinitializeArray_2D(0.0);
  D.NewArray_2D(1, 120);
  D.ToinitializeArray_2D(0.0);
  ///Creamos los datos para entrenar la RED//
  float I[120][2] = {{3.9000 , 1.4000},
    { 3.7000 , 1.0000},
    { 4.5000 , 1.5000},
```

```
20    { 1.5000 , 0.3000},
21    { 4.5000 , 1.5000},
22    { 1.5000 , 0.2000},
23    { 6.0000 , 2.5000},
24    { 3.3000 , 1.0000},
25    { 6.7000 , 2.0000},
26    { 4.8000 , 1.8000},
27    { 3.0000 , 1.1000},
28    { 4.4000 , 1.3000},
29    { 1.5000 , 0.2000},
30    { 1.9000 , 0.2000},
31    { 1.3000 , 0.3000},
32    { 1.5000 , 0.2000},
33    { 5.0000 , 1.9000},
34    { 4.0000 , 1.3000},
35    { 1.4000 , 0.3000},
36    { 5.8000 , 2.2000},
37    { 4.3000 , 1.3000},
38    { 5.1000 , 2.4000},
39    { 5.1000 , 1.6000},
40    { 4.8000 , 1.8000},
41    { 6.6000 , 2.1000},
42    { 5.1000 , 1.9000},
43    { 5.6000 , 1.4000},
44    { 5.0000 , 1.7000},
45    { 1.7000 , 0.5000},
46    { 1.5000 , 0.2000},
47    { 1.3000 , 0.2000},
48    { 1.6000 , 0.2000},
49    { 4.5000 , 1.5000},
50    { 1.4000 , 0.2000},
51    { 5.6000 , 2.4000},
52    { 1.6000 , 0.6000},
53    { 3.9000 , 1.2000},
54    { 1.3000 , 0.2000},
55    { 5.5000 , 1.8000},
56    { 3.3000 , 1.0000},
57    { 6.1000 , 2.5000},
58    { 1.5000 , 0.4000},
59    { 5.1000 , 1.8000},
60    { 5.9000 , 2.1000},
61    { 6.0000 , 1.8000},
62    { 4.5000 , 1.4000},
63    { 1.5000 , 0.1000},
64    { 4.8000 , 1.8000},
65    { 5.9000 , 2.3000},
66    { 4.0000 , 1.3000},
67    { 3.5000 , 1.0000},
68    { 5.6000 , 2.1000},
69    { 4.9000 , 1.4000},
70    { 5.1000 , 2.0000},
71    { 1.5000 , 0.4000},
72    { 6.1000 , 2.3000},
73    { 1.6000 , 0.2000},
74    { 5.2000 , 2.0000},
75    { 4.1000 , 1.3000},
```

```
76    { 1.5000 , 0.1000},
77    { 6.4000 , 2.0000},
78    { 4.5000 , 1.5000},
79    { 1.4000 , 0.2000},
80    { 1.4000 , 0.2000},
81    { 4.6000 , 1.4000},
82    { 5.5000 , 1.8000},
83    { 1.4000 , 0.2000},
84    { 4.7000 , 1.4000},
85    { 1.7000 , 0.2000},
86    { 4.7000 , 1.6000},
87    { 1.5000 , 0.2000},
88    { 4.8000 , 1.4000},
89    { 3.5000 , 1.0000},
90    { 4.5000 , 1.7000},
91    { 1.4000 , 0.1000},
92    { 6.1000 , 1.9000},
93    { 3.9000 , 1.1000},
94    { 5.5000 , 2.1000},
95    { 1.3000 , 0.2000},
96    { 3.8000 , 1.1000},
97    { 1.5000 , 0.4000},
98    { 1.7000 , 0.4000},
99    { 5.8000 , 1.8000},
100   { 4.7000 , 1.2000},
101   { 1.6000 , 0.2000},
102   { 4.6000 , 1.5000},
103   { 5.1000 , 1.5000},
104   { 1.1000 , 0.1000},
105   { 1.4000 , 0.2000},
106   { 1.7000 , 0.3000},
107   { 1.6000 , 0.4000},
108   { 4.0000 , 1.0000},
109   { 4.0000 , 1.3000},
110   { 5.7000 , 2.1000},
111   { 6.7000 , 2.2000},
112   { 4.4000 , 1.2000},
113   { 4.5000 , 1.6000},
114   { 6.3000 , 1.8000},
115   {5.3000 , 1.9000},
116   { 1.5000 , 0.2000},
117   { 4.1000 , 1.0000},
118   { 4.7000 , 1.4000},
119   { 4.0000 , 1.2000},
120   { 1.4000 , 0.2000},
121   { 1.6000 , 0.2000},
122   { 4.9000 , 1.5000},
123   { 1.9000 , 0.4000},
124   { 5.6000 , 2.4000},
125   { 1.4000 , 0.2000},
126   { 1.2000 , 0.2000},
127   { 5.6000 , 2.2000},
128   { 4.9000 , 1.8000},
129   { 4.7000 , 1.5000},
130   { 4.1000 , 1.3000},
131   { 1.6000 , 0.2000},
```

```
  { 4.9000 ,  1.5000},
  { 4.2000 ,  1.5000},
  { 5.8000 ,  1.6000},
  {5.1000 ,  2.3000},
  { 4.6000 ,  1.3000}
};

float T[] = {1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1
    , 0, 1, 0, 1, 1,
          1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1
    , 1, 1, 0, 1, 0,
          0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1
    , 1, 1, 1
         };
for (byte j = 0; j < 2 ; j++) {
  for (byte i = 0; i < 120 ; i++) {
    XI.WriteArray_2D(i, j, I[i][j]);
  }
}

for (int i = 0; i < 120 ; i++) {
  D.WriteArray_2D(0, i, T[i]);
}
byte XC = XI.GetColumns();
byte DF = D.GetRows();
byte XF = XI.GetRows();
net.DATA_CENTERING(XI);
net.NORMALIZE_DATA(XI, XI.MaxArray_2D());
byte L = 3;
EST.NewArray_2D(1, L);
EST.ToinitializeArray_2D(0.0);
float NC = EST.GetColumns();
float V[] = {XC, 12, DF};
for (byte i = 0; i < NC; i++) {
  EST.WriteArray_2D(0, i, V[i]);
}
FAC_STR.CharNewArray_1D(L);
char *myStrings[] = {"tansig", "tansig", "hardlim"};
for (byte i = 0; i < FAC_STR.GetRows(); i++)  {
  FAC_STR.CharWriteArray_1D(i, myStrings[i]);
}

net.FEED_FORWARD_NET(EST, FAC_STR);
net.TRAIN_NET(XI, D, 1e-3, 1200);
net.SIM_NET(XI);
byte correct = 0;
for (byte i = 0; i < XF ; i++)  {
  if (D.ReadArray_2D(0, i) == round(net.Ysim.ReadArray_2D(0, i)))
  {
    correct = correct + 1;
  }
  Serial.print(D.ReadArray_2D(0, i), 4);
  Serial.print(" ");
  Serial.println(round(net.Ysim.ReadArray_2D(0, i)), 4);
}
float resultado = correct * 100.00 / XF;
```

```
185   Serial.print("porcentaje de clasificacion:" );
186   Serial.println(resultado);
187   // //imprimimos epochs y error cudratico medio
188   Serial.println(net.EPOCHS);
189   Serial.println(net.ERROR_C_M, 6);
190 }
191 void loop() {
192 }
```

Código 6.3: Clasificador de Setosas de la librería Neural_Networks_FF

7. Conclusiones

- Una correcta selección de componentes electrónicos en dispositivos como los sistemas embebidos, permiten tener un gran grado de libertar en uso de implementación de dispositivos externos facilitando su uso, generando confiabilidad del sistema. al escogerse el microcontrolador ATSAMD21G18-AU, brindo grandes ventajas en comparación de otros microcontroladores, ya que el sistema requería el uso mínimo de 4 pines de interrupción externas, este microcontrolador contaba con 12 pines de interrupción configurables y 6 puertos de entradas analógicas. además para la compilación de algoritmos neuronales, se necesita una gran velocidad de computo, contando este microcontrolador con una velocidad de ciclo de reloj 48Mhz y un ancho de bus de 32 bits siendo La cantidad de datos transferidos entre la CPU del microcontrolador y el sistema de memoria, un poco mas de doble de velocidad con la comparativa que se realizo en la elección de dicho dispositivo.

- Cuando se desea fabricar una placa de circuito impreso, se debe tener en cuenta aparte de las prestaciones que brinda el elemento electrónico, realizar la elección de los dispositivos en términos de optimización de espacio en referencia al tipo de encapsulado,debido a estos parámetro permitió colocar gran cantidad y diversidad de componentes electrónicos, conectores entre otros en un reducido espacio.

- la implementación de una interfaz visual interactiva, en el sistema embebido, permitió una comunicación entre el dispositivo la persona que lo manipule,permitiéndole tener un mayor control sobre las variables, que necesiten ser alteradas, que además, la integración de la interfaz, permite realizar un uso rápido y directo del sistema, sin necesitar de utilizar algún programa o sistema externo para cargar las variables que necesiten ser configuradas.

- El sistema embebido Para aplicación de Control Adaptativo Basado en redes Neuronales permitió controlar 3 tipos de variable con dinámica diferentes con el tipo de control neuronal, las cuales son: velocidad en rpm, temperatura en grados celsius y nivel de un liquido en cm, que por medio de la interfaz le permite seleccionar que tipo de variable se desea controlar y llenar los parámetros de la planta.

- se pudieron desarrollar dos librerías compatibles con toda la gama de tarjetas de desarrollo arduino, para facilitar el uso de los algoritmos de redes neuronales. La librería Dynamic_Array, que permite generar arreglos dinámicos que se crean en la memoria SRAM y estos arreglos son compatibles con la libreria Neural_Networks_FF en donde esta ubicado todo lo referente a la creación de la red neuronal, con topologías variables, con N numero de capas, N numero de neuronas por capa, y selección de función de activación por capa y ajuste de factor de aprendizaje. además en la librería cuenta con

ejemplos de entrenamiento de la red por entrada de patrones, y el uso de la red neuronal como controlador.

- los algoritmos de redes neuronales, presentaron un desempeño favorable al momento de ser aplicados en plantas con dinámica rápida y con sistemas lineales, debido a la relación que existe entre el cambio en el tiempo de la variable controlada, con la señal de control generada por los algoritmos neuronales caso contrario con lo que ocurre con plantas con dinámicas demasiado lentas no lineales,debido a que la variable a controlar no aumenta rápidamente como lo hace la señal de control, haciendo que la señal de control se sature, cuando la variable a controlar pasa por el valor deseado, la señal de control desciende gradualmente, sin embargo la inercia presente es tan fuerte que se genere un sobre paso de la señal, visto este comportamiento en la planta de temperatura, donde su comportamiento de descenso de la temperatura no existe un medio en estos montajes para ayudarle a bajar el nivel de temperatura queda esperar que la temperatura descienda por la naturaleza de la planta. En el caso de la planta de nivel es una dinámica no rápida como la planta de velocidad, pero no tan lenta como la planta de temperatura, el controlador neuronal no ofreció el rendimiento visto en las otras plantas, generando sobre paso en la señal sobre y por debajo del setpoint, esto influencia de la naturaleza de la planta. el valor de factor de aprendizaje usado fue el que mejor prestaciones dio en términos de control, factores mas altos, hacían oscilar la planta y factores mas bajos, le tomaba periodos de tiempos muy largos para llegar al valor deseado.

- Se comprobó la influencia que existe en el factor de aprendizaje en el controlador, determinando la agresividad de la respuesta del sistema,Cuando se aplican factores de aprendizaje muy altos,la velocidad respuesta del sistema aumenta lo cual podría generar sobre pasos de la señal y sobre oscilaciones en la respuesta en estado estacionario, con factores de aprendizaje bajos, la velocidad de la respuesta del sistema le tomaría mucho tiempo llegar al valor deseado, la correcta elección del factor de aprendizaje, va con ligado tiempo de muestreo del controlador y este tiempo de muestreo va asociado la velocidad de respuesta de la dinámica de la planta a controlar.

- Siendo el sistema embebido un dispositivo de control, se realizo una ficha técnica del dispositivo, de modo que permita tener una visión mas clara del uso del dispositivo, enmarcando, cuales son los puertos de entrada salida, tierra o GND, elementos con los que cuenta, pulsador perilla, la tensión de alimentación que soportar el dispositivo, ejemplo de diagramas conexión de los sensores ligados para el correcto uso del sistema embebido, modo de uso de la pantalla.

- como trabajo agregado se comprobó el correcto funcionamiento de las librerías desarrolladas para las tarjetas de desarrollo arduino en código abierto, validándolas como funciones de aproximación y clasificación, presentando respuestas muy cercanas, y para generar repuestas aun mas exactas, se debe volver muy estricta la condición de salida del error, que lo que ocasionaría un mayor numero de iteraciones para el entrenamiento de la red.

Bibliografía

Acedo Sanchez, J. (2003). *CONTROL AVANZADO DE PROCESOS (Teoría y práctica)* (1a ed.; E. D. d. S. S.A., Ed.). Madrid, España. doi: M.43.580-2002

Alonso, R. (2018). *ARM en procesadores: qué es y cómo funciona esta arquitectura.* Descargado 2020-09-02, de `https://hardzone.es/tutoriales/componentes/procesador-arm/`

Asesorado, S. E. T. C. (2007). *DISEÑO DE CONTROLADORES PID EN TIEMPO DISCRETO, Y ANÁLISIS DE RESPUESTA UTILIZANDO HERRAMIENTAS COMPUTACIONALES* (Tesis Doctoral no publicada). UNIVERSIDAD DE SAN CARLOS DE GUATEMALA FACULTAD.

BASURTO, S. D. M. (1991). *sistemas-de-control-automatico-benjamin.*

Caicedo, B., y Lopez, J. (2010). Redes Neuronales Artificiales. *Charlas de fisica*, 276. doi: 10.1016/S0210-5691(05)74198-X

Carelli, I. R. (2014). *Controladores Discreto de Bajo Orden* (Vol. 7) (no 2). Descargado de `http://www3.fi.mdp.edu.ar/control4c7/APUNTES/Clase7-PID.pdf`

Carlos, J., y Calzadilla, F. (2019). *La enseñanza de los arrays estáticos , dinámicos y listas enlazadas ¿ cuál usar ? Análisis de códigos The teaching of static arrays , dynamics and linked lists . What to use ? Code analy ... The teaching of static arrays , dynamics and linked lists . What to* (no April).

Carolina Roman Bueno, J., y Johanna Gonzalez Mantilla, K. (2017). *Sistemas embebidos y Hardware libre.* Descargado de `http://wiki.sc3.uis.edu.co/images/e/e1/GR7.pdf`

Clara Bolonia. (2017). *¿Cuáles son las partes de una neurona? | La-Reserva.com.* Descargado 2020-11-06, de `https://www.lareserva.com/Cuales{_}son{_}las{_}partes{_}de{_}una{_}neurona`

COPA-DATA. (2019). *¿Qué significa HMI? Interfaz humano-máquina | COPA-DATA.* Descargado 2021-02-12, de `https://www.copadata.com/es/productos/zenon-software-platform/visualizacion-control/que-significa-hmi-interfaz-humano-maquina-copa-data/`

Cruz Avilés, A., Ortiz Domínguez, M., y Muños-Sánchez, Y. (2018). *Ingeniería de control moderna* (Vol. 5) (no 10). doi: 10.29057/ess.v5i10.3323

daniel, P. (2011). *Encapsulados: DIP (Dual in-line package).* Descargado 2020-11-18, de `http://capsulacpu.blogspot.com/2011/04/dip-dual-in-line-package.html`

david, M. (2017). *5 reglas y recomendaciones de diseño de PCB que debe conocer | Altium.* Descargado 2021-02-13, de https://resources.altium.com/es/p/top-5-pcb-design-guidelines-every-pcb-designer-needs-know

David Johnson-Davies. (2019). *Technoblogy - Minimal ATSAMD21 Computer 2.* Descargado 2021-01-23, de http://www.technoblogy.com/show?2HQ2{#}cite{_}note1

Doctoral, T., Francisco, D., y Zamorano, O. (2015). *Algoritmos de aprendizaje neurocomputacionales para su implementación hardware.*

Embention. (2018). *Control adaptativo y sus aplicaciones en la industria | Embention.* Descargado 2019-11-29, de https://www.embention.com/es/news/control-adaptativo-y-sus-aplicaciones/

Esparza, C. H., y Núñez, R. A. (2014). Controlador adaptativo PD por modelo de referencia para una mesa vibratoria biaxial basada en el mecanismo biela-manivela. *Informacion Tecnologica*, *25*(2), 189–202. Descargado de https://scielo.conicyt.cl/scielo.php?script=sci{_}arttext{&}pid=S0718-07642014000200021{&}lng=es{&}nrm=iso{&}tlng=eshttps://scielo.conicyt.cl/scielo.php?script=sci{_}abstract{&}pid=S0718-07642014000200021{&}lng=es{&}nrm=iso{&}tlng=es doi: 10.4067/S0718-07642014000200021

Facultad de Tecnología y Ciencias Aplicadas - UNCA. (2016). *Laboratorio de Sistemas Embebidos - LaSE: Qué es un Sistema Embebido?* Descargado 2020-11-17, de http://lasetecno.blogspot.com/p/que-es-un-sistema-embebido.html

Fernando Valdes, R. P. A. (2007). *Microcontroladores Fundamentos y Aplicaciones con PIC.* Descargado 2020-08-31, de https://books.google.es/books?hl=es{&}lr={&}id=ODenKGOHMRkC{&}oi=fnd{&}pg=PA9{&}dq=que+es+un+microcontrolador{&}ots=umxT4l876A{&}sig=EZpp0D344UX5{_}MYrasIjk-Q55o8{#}v=onepage{&}q=queesunmicrocontrolador{&}f=false

Gómez, H. (2017). *Programación de microcontroladores usos y aplicaciones.* Descargado 2020-11-17, de https://docplayer.es/51802554-Programacion-de-microcontroladores-usos-y-aplicaciones-desde-ensamblador-a-c.html

González García, D. (2012). *Encapsulado SOT-233 | ———————————————— Diego González García ————————————————*. Descargado 2020-11-18, de https://mecatronic2.wordpress.com/2012/10/06/encapsulado-sot-233/

Humberto, M., y Contreras, B. (2017). *Un sistema de recomendaci on basado en factorización de matrices y descenso en gradiente estocástico* (Tesis Doctoral no publicada). Instituto Tecnol´ogico Aut´onomo de M´exico U.

Izaurieta, F., y Saavedra, C. (1999). Redes Neuronales Artificiales. *Charlas de fisica*, 1–15. doi: 10.1016/S0210-5691(05)74198-X

Jmnelectronics.com. (2018). *Microcontroladores: Condensadores y Alimentación. | JMN Electronics.* Descargado 2021-01-23, de http://jmnelectronics.com/archives/726

Jonathan López. (2015). *Osciladores para microprocesadores o microcontroladores - Electrónica Unicrom.* Descargado 2020-11-18, de https://unicrom.com/osciladores-para-microprocesadores-o-microcon/

López, M. (2017). *Identificación de sistemas no lineales con redes neuronales convolucionales.*

López, P. (2006). *Memoria Dinamica.*

Martín del Brio, B., y Sanz, A. (2001). *Redes neuronales y logica borrosa.*

Martínez, P., Siavicha, B., y Teneseca, J. (2000). *Diseño De Un Controlador Avanzado Basado En Redes Neuronales Para La Gestión De La Mezcla Aire-Gasolina En Un Motor Alternativo* (Tesis Doctoral, UNIVERSIDAD POLITÉCNICA DE CATALUÑA). Descargado de https://dspace.ups.edu.ec/bitstream/123456789/1774/13/UPS-CT002050.pdf{%}0Ahttps://upcommons.upc.edu/handle/2117/93248

MATLAB. (2020). *Log-sigmoid transfer function.* Descargado 2020-11-07, de https://www.mathworks.com/help/deeplearning/ref/logsig.html

May. (2015). *UNIDAD 4 REDES NEURONALES - INTELIGENCIA ARTIFICIAL.* Descargado 2020-08-17, de https://sites.google.com/site/mayinteligenciartificial/unidad-4-redes-neuronales

Mazzone, V. (2002). *Controladores PID Industriais.*

Noriega, A., Barrera, A., y Ordez, A. (2008). Diseño de un controlador neuronal y su implantación en un microcontrolador. *XIII Congreso Latinoamericano de Control Automático / VI Congreso Venezolano de Automatización y Control*(1).

Pardo, C. (2020). *Controlador PID - Control Automático - Picuino.* Descargado 2020-09-03, de https://www.picuino.com/es/arduprog/controlAsesorado2007-pid.html

pardo Carlos. (2014). *Controlador PID digital - Control Automático - Picuino.* Descargado 2020-12-03, de https://www.picuino.com/es/arduprog/control-pid-digital.html

PAÚL, C. (2013). *ARREGLOS DINAMICOS.*

Peñafiel, C., y Ing. Ávila, R. (2020). Inteligencia Artificial. *Inteligencia Artificial*, *2*(6), 1–33. Descargado de http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Inteligencia+Artificial:+un+enfoque+moderno{#}0 doi: 10.4114/ia.v2i6.614

Perez, M., Perez, A., y Perez, E. (2007). Introduccion a los sistemas de control y modelo matemático para sistemas lineales invariantes en el tiempo. *Universidad Nacional de San Juan*, *1*, 1–69.

Queralt, R. (2004). Instrumentación y control. *Tecnologia del Agua*, *24*(253), 5–8.

Reyes, H., y Montaña, M. (2010). *Modelamiento y control digital de temperatura para horno eléctrico* (Tesis Doctoral). Descargado de http://repository.javeriana.edu.co/bitstream/10554/7044/1/tesis489.pdf

Rosas Oscar. (2017). *Arquitecturas Von-Newman VS Harvard | Compilando Conocimiento.* Descargado 2020-12-17, de https://compilandoconocimiento.com/2017/01/29/arquitecturasvon-newmanvsharvard/

Sánchez., S. (2013). *Introducción y Arquitectura de microcontroladores | Microcontroladores.* Descargado 2020-09-03, de https://microcontroladoressesv.wordpress.com/arquitectura-de-los-microcontroladores/

Sanchez medina, J. (1998). *Linealización del algoritmo de backpropagation para el entrenamiento de redes neuronales* (Tesis Doctoral no publicada). UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA.

Santos, M. (2011). Un enfoque aplicado del control inteligente. *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, *8*(4), 283–296. doi: 10.1016/j.riai.2011.09.016

Seabrookewindows. (2020). *Rendimiento microcontrolador / Seabrookewindows.com.* Descargado 2020-11-17, de https://www.seabrookewindows.com/xWzOX55Pd/

Serrano, Á. R. (2012). *Arquitectura y Organización de un microcontrolador genérico.* Descargado de https://docplayer.es/5865375-Arquitectura-y-organizacion-de-un-microcontrolador-generico.html

Soriano, J. (2012). *Diseño de un extensor de entradas y salidas analógicas por MODBUS RTU sobre RS – 485* (Vol. 13).

Surtel. (2019). *Componentes SMD ¿Qué son?* Descargado 2020-11-18, de https://www.surtel.es/blog/que-son-los-componentes-smd/

Tacconi, E., Mantz, R., y Solsona, J. (2005). *Versión electrónica editada por : Tania Salazar y Ana Roquez. Año 2005.*

Tang, N. (2018). *Introducción a las Redes Neuronales.*

Távara Idrogo, J. G. (2019). *Modelo de reconocimiento automático de señales de trámsito vehicular mediante aprendizaje profundo de redes neuronales convolucionales* (no 061).

Tony R. (2006). *LAZO DE CONTROL.* Descargado 2021-02-12, de http://cursoinstrumentacionycontrol.blogspot.com/2016/10/lazo-de-control.html

Trecedb. (2009). *Partes del microcontrolador | trece dBs.* Descargado 2020-11-17, de https://trecedb.wordpress.com/2009/02/13/partes-del-microcontrolador/

Perez Molina, C., Gomez, M. J. A., Gil, R., Orueta, G. D., Sancristobal, E., Martin, S., … Loro, F. G. (2013). Performance-centered adaptive curriculum for employment needs. *ASEE Annual Conference and Exposition, Conference Proceedings*, 1–21. doi: 10.1109/taee.2012.6235460

Lista de Acrónimos y Abreviaturas

ADALINE	Adaptative Linear Element.
ADC	Convertidos Digital analógico.
DAC	Convertidos Analógico Digital.
PCB	Placa de circuito impreso.
PD	Proporcional + Derivativo.
PI	Proporcional + Integral.
PID	Proporcional + Integral.
PWM	Modulación por ancho de pulso.

A. Páginas horizontales

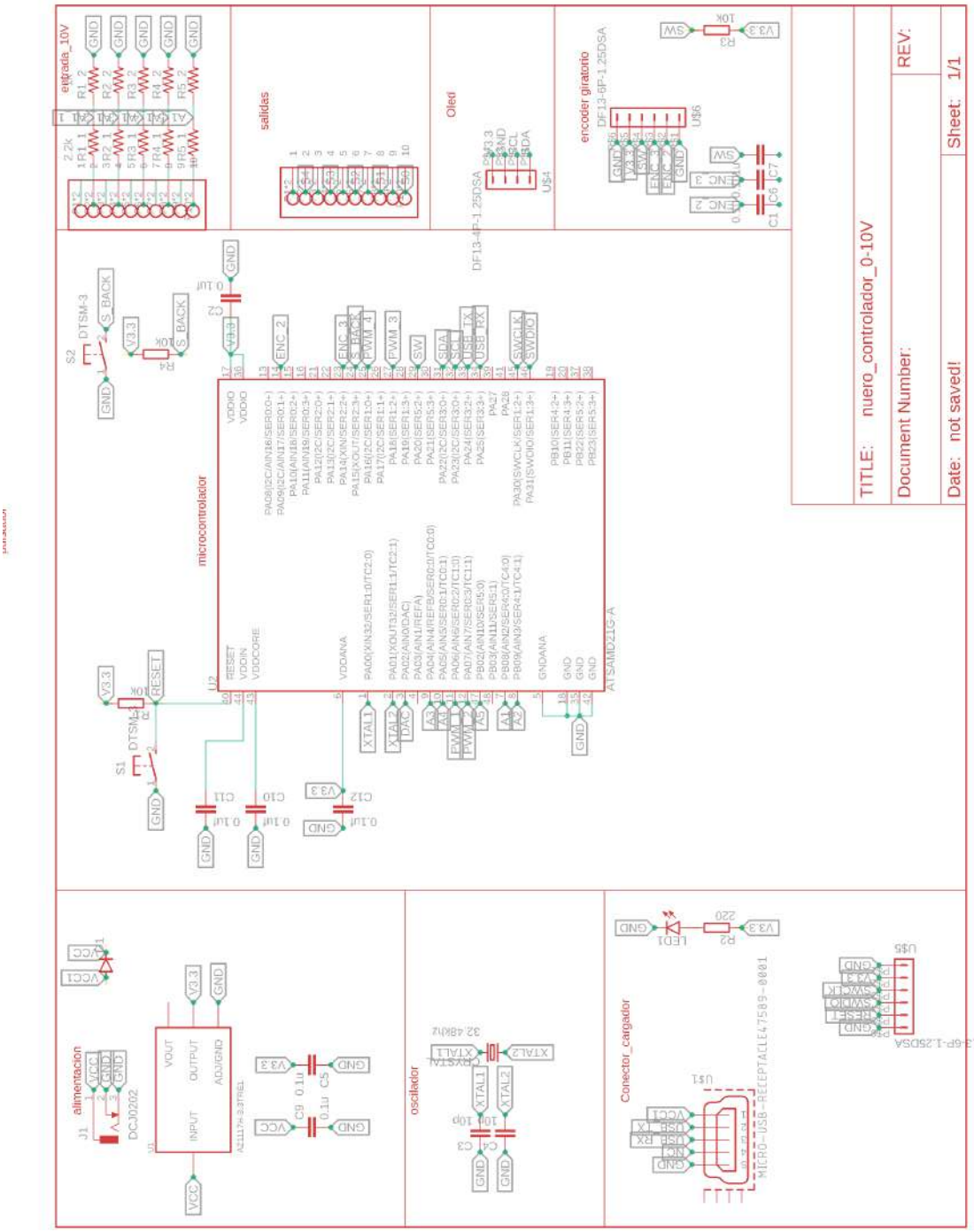


Figura A.1: Esquemático Sistema Embebido 1

Fuente: AUTOR

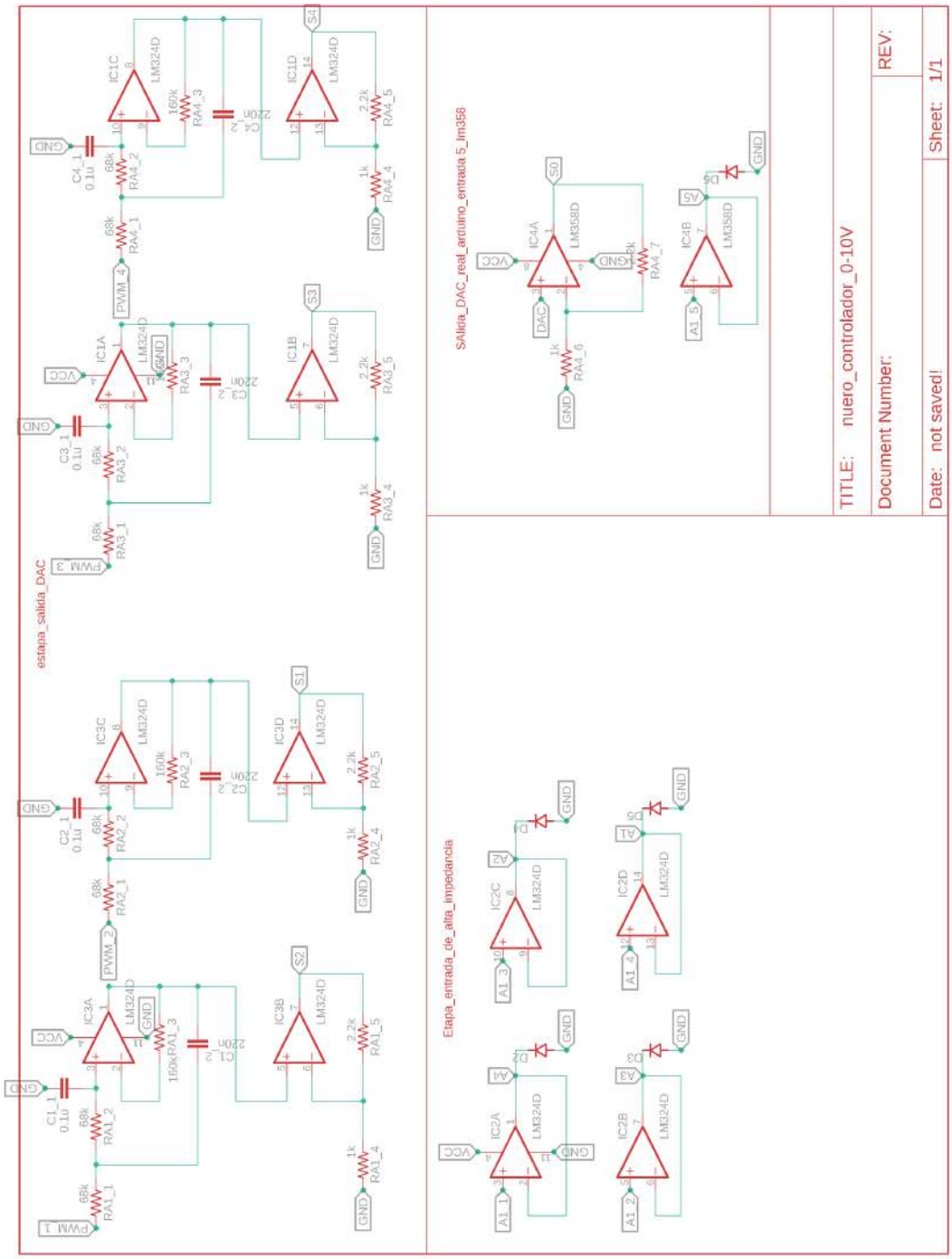


Figura A.2: Esquemático Sistema Embebido 2

Fuente: AUTOR