

Implementación de un control neuronal directo en el controlador lógico programable S7-1200 para aplicaciones industriales

Grado en Ingeniería Mecatrónica

Trabajo Fin de Grado

Autor:

José Fernando Ramírez Rodríguez

Tutores:

Juan Martín Cáceres Tarazona

Luis Ernesto Neira Roperó

Diciembre 2020



Implementación de un control neuronal directo en el controlador lógico programable S7-1200 para aplicaciones industriales

Autor

José Fernando Ramírez Rodríguez

Tutor

Juan Martín Cáceres Tarazona
MSc. Controles Industriales

Cotutor

Luis Ernesto Neira Roperro
Ingeniero en Mecatrónica

Grado en Ingeniería Mecatrónica



PAMPLONA, Diciembre 2020

Preámbulo

El Controlador lógico programable PLC es el cerebro detrás del control de procesos industriales, tal como su nombre lo indica son los encargados de realizar el control de las variables en el proceso, estos equipos son programados y adaptados para cada proceso industrial. En la mayoría de los casos, estos se encuentran desarrollando tareas de control ON-OFF (Jaimes y Giraldo, 2015), o tareas de control regulatorio para el control estricto de las variables.

Cuando la variable a controlar no presenta un comportamiento lineal y es difícil realizarle el control por los métodos tradicionales (PD, PI o PID), el proceso industrial requiere el desarrollo de un conjunto de técnicas de Identificación y Control en línea utilizando algoritmos inteligentes (ej: Redes Neuronales y Lógica Difusa).

Teniendo en cuenta esta problemática que se presentan en los controles convencionales este documento plantea desarrollar y aplicar un algoritmo de control adaptativo directo, basado en redes neuronales tipo Adaptive Linear Element (ADALINE) y Multilayer perceptron (MLP) enfocado a un PLC para aplicaciones industriales.

Agradecimientos

Agradezco a la familia que creyó en mi y me apoyo en este proyecto, agradezco al señor Oscar Lopez amigo de la familia por ayudarme a realizar este proyecto realidad, agradezco al profesor Juan Cáceres y el profesor Luis Neira por brindarme su asesoría, conocimiento y apoyo para culminar este proyecto, agradezco a los compañeros de carrera por su ayuda, en especial a Wilkinfader y Nick grandes personas y compañeros de lucha, agradezco a los docentes del programa por compartir sus conocimientos, agradezco a todas las amistades que me han colaborado, sin ellos esto no hubiese sido posible.

Dedico este proyecto a...
A mi madre Cecilia, mi padre José, mi hermana Angie y mi sobrino Liam...
son el motivo de mi lucha...

*Si buscas resultados
distintos, no hagas
siempre lo mismo*

Albert Einstein

*Lo que sabemos es una gota
de agua, lo que
ignoramos es el océano*

Isaac Newton

Índice general

1	Introducción	5
2	Justificación	7
3	Objetivos	9
3.1	General	9
3.2	Específicos	9
4	Marco Teórico	11
4.1	Sistemas de Control	11
4.1.1	Técnicas de control Clásico	12
4.1.1.1	Control PID	12
4.1.2	Técnicas de control Avanzado	14
4.2	Redes Neuronales	14
4.2.1	El Perceptrón	17
4.2.2	Red Adaline	18
4.2.2.1	Regla delta	19
4.2.3	Perceptron Multi-capa	20
4.2.4	Redes Neuronales como controladores	22
4.3	PLC (Controlador Lógico Programable)	24
4.3.1	Lenguajes de Programación	25
4.3.1.1	Texto Estructurado (ST)	25
4.3.1.2	Lenguaje de Control Estructurado (SCL)	26
5	Metodología	29
5.1	Selección de Plantas	29
5.2	Descripción de las plantas	30
5.2.1	Planta para el control de Velocidad	30
5.2.1.1	Componentes	31
5.2.1.1.1	Motor Monofásico AC	31
5.2.1.1.2	Variador de Frecuencia	31
5.2.1.1.3	Encoder	32
5.2.2	Planta para el control de Temperatura	34
5.2.2.1	Componentes	35
5.2.2.1.1	Sensor de Temperatura RTD PT100	35
5.2.2.1.2	Resistencia Calefactora tubular AC	35
5.2.2.1.3	Relé de estado solido (SSR)	36

5.2.3	Planta para el control de Flujo	37
5.2.3.1	Componentes	37
5.2.3.1.1	Válvula proporcional	37
5.2.3.1.2	Sensor de Flujo	38
5.3	Algoritmo Adaline	39
5.3.1	Regla de aprendizaje	40
5.4	Algoritmo de control discreto	43
5.5	Comparando la red adaline con la estructura del control PID discreto	47
5.6	Algoritmo Backpropagation para redes MLP	47
5.6.1	Algoritmo backpropagation	51
5.7	PLC S7-1200	53
5.8	Lenguaje de programación SCL	55
5.8.1	Descripción del lenguaje	56
5.8.2	Ejemplos en SCL	58
5.8.3	Comparación del lenguaje con LADDER	59
6	Desarrollo	63
6.1	Acondicionamiento e identificación de las plantas	63
6.1.1	Planta de Velocidad	63
6.1.1.1	Acondicionamiento	63
6.1.1.2	Identificación	66
6.1.2	Planta de Temperatura	67
6.1.2.1	Acondicionamiento	67
6.1.2.2	Identificación	70
6.1.3	Planta de Flujo	71
6.1.3.1	Acondicionamiento	71
6.1.3.2	Identificación	73
6.2	Control clásico aplicado	74
6.2.1	Control PID en la planta de Velocidad	79
6.2.2	Control PI en la planta de Temperatura	80
6.2.3	Control PI en la planta de flujo	82
6.3	Control neuronal adaline directo	84
6.3.1	Control neuronal adaline directo en la planta velocidad	88
6.3.2	Control neuronal adaline directo en la planta de temperatura	89
6.4	Control neuronal multi-capa directo	89
6.4.1	Control neuronal multi-capa directo en la planta velocidad	92
6.4.2	Control neuronal multi-capa directo en la planta de temperatura	92
6.4.3	Control neuronal multi-capa directo en la planta de flujo	93
6.5	Interfaz generada	94
7	Resultados	97
7.1	Aplicando controladores a la planta de velocidad	97
7.1.1	Respuesta control PID	97
7.1.2	Respuesta control neuronal adaline	99
7.1.3	Respuesta controlador neuronal multi-capa	101
7.1.4	Comparación de las técnicas de control planta de velocidad	104

7.2	Aplicando controladores en la planta de temperatura	107
7.2.1	Respuesta controlador PI	107
7.2.2	Respuesta controlador neuronal adaline	108
7.2.3	Respuesta controlador neuronal multi-capa	108
7.2.4	Comparación de las técnicas de control planta de temperatura	109
7.3	Aplicando controladores en la planta de flujo	110
7.3.1	Respuesta controlador PI	110
7.3.2	Respuesta controlador neuronal multi-capa	111
7.3.3	Comparación de las técnicas de control en la planta de flujo	112
7.4	Librerías generadas	112
7.4.1	Librería para la lectura de velocidad por encoder rotativo incremental	113
7.4.2	Librería para la captura de datos por medio del servidor web de Siemens S/-1200 CPU1214c DC/CD/CD	113
7.4.3	Librería Filtro EMA	114
7.4.4	Librería de control neuronal adaline	115
7.4.5	Librería de control neuronal multi-capa	115
8	Conclusiones	117
	Bibliografía	119
	Lista de Acrónimos y Abreviaturas	123

Índice de figuras

4.1	Sistema de control en lazo abierto	11
4.2	Sistema de control en lazo Cerrado	12
4.3	Lazo de control PID	13
4.4	Estructura de una red neuronal artificial	16
4.5	Perceptrón simple y función de transferencia de su neurona	17
4.6	Estructura de una red neuronal tipo Adaline	18
4.7	Gradiente Descendente	19
4.8	Arquitectura general de un perceptron multi-capa	20
4.9	Red neuronal como controlador	23
4.10	Controlador lógico programable (PLC)	24
5.1	Diagrama del lazo de control de la planta de velocidad	30
5.2	Motor monofásico	31
5.3	Variador de Frecuencia Omron SYSDRIVE 3GMV INVERTER	32
5.4	Encoder rotatorio LPD3806-360BM-G5-24C	33
5.5	Funcionamiento Encoder	34
5.6	Diagrama del lazo de control de la planta de temperatura	34
5.7	Sensor de temperatura PT100	35
5.8	Resistencia Calefactora tubular	35
5.9	Rele de estado solido SRPH1-A220	36
5.10	Diagrama del lazo de control de la planta de control de flujo	37
5.11	Válvula Proporcional	37
5.12	Sensor de flujo FS300A	38
5.13	Característica de decisión de una red tipo Adaline	40
5.14	Actualización de pesos del algoritmo LMS	42
5.15	Estructura de la red neuronal adaline	43
5.16	Sistema de control PID DIGITAL en lazo cerrado	44
5.17	Figura a) señal analógica, Figura b) señal discreta	44
5.18	Lazo de control PID Discreto	46
5.19	Topología de Redes Neuronales Multi-capas	47
5.20	PLC S7 1200 CPU Compact 1214C DC/DC/DC 6ES7214-1AG40-0XB0	53
5.21	Signal Board 6ES7232-4HA30-0XB0 A0 0-10V o 0-20mA	53
5.22	HMI KTP700 Basic PN	54
5.23	Ejemplo 1 en LADDER	59
5.24	Ejemplo 2 en LADDER	60
5.25	Ejemplo 3 en LADDER	60
5.26	Ejemplo 4 en LADDER	61
5.27	Ejemplo 5 en LADDER	61
5.28	Ejemplo 6 en LADDER	62

6.1	Diseños del acople y soporte	63
6.2	Acople y soporte del encoder impresos en 3D	64
6.3	Bloque de función para lectura de Velocidad con el encoder	64
6.4	Comprobación de la lectura de Velocidad	65
6.5	Diagrama ampliado funcionamiento de la planta de velocidad	65
6.6	Planta de Velocidad	66
6.7	Identificación de la planta de Velocidad	67
6.8	Transmisor RTD par sensor de temperatura PT100	67
6.9	Diagrama ampliado funcionamiento de la planta de Temperatura	68
6.10	Medidor de temperatura de mercurio	69
6.11	Comparación del valor capturado de temperatura	69
6.12	Planta temperatura	70
6.13	Identificación planta de temperatura	70
6.14	Filtro EMA (Media móvil exponencial)	72
6.15	Planta de control de flujo ampliada	72
6.16	Planta de control de Flujo	73
6.17	Caudal vs porcentaje abierto de la válvula	73
6.18	Control PID en lazo cerrado	74
6.19	Creación del bloque de interrupción	75
6.20	Creación del bloque "PID_Compact"	75
6.21	Objeto tecnológico PID	76
6.22	Ajuste Controlador PID velocidad	77
6.23	Parámetros del bloque PID	77
6.24	Sintonización PID	78
6.25	Carga de constantes calculadas del PID	79
6.26	Sintonización final PID velocidad	79
6.27	Parámetros PID velocidad	80
6.28	Ajustes control PI temperatura	81
6.29	Sintonización final PI temperatura	81
6.30	Parámetros PI temperatura	82
6.31	Ajustes control PI de flujo	82
6.32	Sintonización final PI de flujo	83
6.33	Parámetros PI flujo	83
6.34	Control Neuronal adaline en lazo cerrado	84
6.35	Topología red neuronal aplicada	84
6.36	Bloque de función control neuronal adaline para velocidad	86
6.37	Variables del bloque de función control neuronal adaline	88
6.38	Bloque de función control neuronal adaline para temperatura	89
6.39	Control neuronal multi-capas en lazo cerrado	90
6.40	Topología de la red neuronal aplacada	90
6.41	Bloque control neuronal multi-capas en la planta de velocidad	91
6.42	Bloque de control neuronal multi-capas en la planta de temperatura	93
6.43	Bloque de control neuronal multi-capas en la planta de flujo	94
6.44	Interfaz de la HMI en el control de velocidad	95

7.1	Respuesta PID en la planta de Velocidad	97
7.2	Respuesta controlador PID ante perturbaciones	98
7.3	Respuesta control neuronal adaline velocidad $Fa=0.00135$	99
7.4	Respuestas ante variación del factor de aprendizaje control neuronal adaline .	100
7.5	Respuestas ante variación del factor de aprendizaje control neuronal adaline ampliada	101
7.6	Respuestas control neuronal adaline $Fa=0.00135$ ante perturbaciones	101
7.7	Respuestas control neuronal multi-capa $Fa=0.071$	102
7.8	Respuestas ante variación del factor de aprendizaje control neuronal multi-capa	103
7.9	Respuestas ante variación del factor de aprendizaje control neuronal multi-capa ampliada	103
7.10	Respuestas control neuronal multi-capa $Fa=0.071$ ante perturbaciones	104
7.11	Respuestas controladores en la planta de velocidad	105
7.12	Respuestas controladores en la planta de velocidad ampliado setpoint 1000 rpm	105
7.13	Respuestas controladores en la planta de velocidad ampliado setpoint 900rpm	106
7.14	Respuesta controlador PI en la planta de temperatura	107
7.15	Respuesta controlador neuronal adaline en la planta de temperatura $Fa=0.0015$	108
7.16	Respuesta controlador neuronal multi-capa en la planta de temperatura $Fa=0.28$	109
7.17	Respuestas controladores en la planta de temperatura	110
7.18	Respuesta controlador PI en la planta de flujo	111
7.19	Respuesta controlador neuronal multi-capa en la planta de flujo $Fa=0.01$. . .	111
7.20	Respuestas controladores en la planta de flujo	112
7.21	Librería Lectura de encoder incremental	113
7.22	Librería Captura de datos	114
7.23	Librería Captura de datos	115
7.24	Librería control neuronal adaline	116
7.25	Librería control neuronal multi-capa	116

Índice de tablas

4.1	Funciones de activación	22
4.2	Lenguajes de programación en PLC's	27
5.1	Características del Motor trifásico	31
5.2	Características generales del variador de frecuencia	32
5.3	Características del Encoder	33
5.4	Características sensor de temperatura PT100	35
5.5	Especificaciones del SSR	36
5.6	Características válvula proporcional	38
5.7	Características Sensor de flujo	39
5.8	Algoritmo backpropagation	52
5.9	Características generales PLC S7-100 CPU 1214C 6ES7 214-1AG40-0XB0	55
7.1	Tiempo de estabilización de los controladores en el setpoint de 1000 rpm (subida)	106
7.2	Tiempo de estabilización de los controladores en el setpoint de 900 rpm (bajada)	106

Índice de Códigos

5.1	Declaración de variables en SCL	56
5.2	Asignación Variable SCL	56
5.3	operaciones matemáticas en SCL	57
5.4	operaciones lógicas en SCL	57
5.5	Estructuras condicionales en SCL	57
5.6	Estructuras de estado de iteración en SCL	57
5.7	Calculo del valor promedio	58
5.8	Calculo del volumen en un tanque	58
5.9	Ejemplo 1 en SCL	59
5.10	Ejemplo 2 en SCL	60
5.11	Ejemplo 3 en SCL	60
5.12	Ejemplo 4 en SCL	61
5.13	Ejemplo 5 en SCL	61
5.14	Ejemplo 6 en SCL	62
6.1	Algoritmo de control neuronal adaline	86

Resumen

Para responder a las exigencias de la industria hoy en día se realizan nuevos avances en la implementación de técnicas de control no convencionales (ej: control adaptativo, control inteligente) buscando la optimización, calidad y competitividad en los procesos de producción. El presente trabajo propone la implementación de control neuronal directo en un controlador lógico programable (PLC) S7-1200 de la marca Siemens, un algoritmo adaptativo basado en redes neuronales artificiales para el control de procesos, el algoritmo de control se implantará directamente en el PLC y podrá ser utilizado en diferentes aplicaciones.

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, el algoritmo esta disponible en en el lenguaje de programación de de alto nivel de PLC's SCL, este lenguaje permite flexibilidad para los cálculos matemáticos.

Se desarrollaron dos algoritmos de control neuronal uno basado en una red de tipo ADALINE y otro basado en una red tipo perceptron multi-capas (MLP), los dos algoritmos de control se validaron en diferentes plantas junto a un controlador PID, donde se analizaron y se compararon las respuestas de cada uno de los controles.

Abstract

To respond to the demands of the industry today, new advances are being made in the implementation of unconventional control techniques (eg: adaptive control, intelligent control) seeking optimization, quality and competitiveness in production processes. The present work proposes the implementation of direct neural control in a Siemens S7-1200 programmable logic controller (PLC), an adaptive algorithm based on artificial neural networks for process control, the control algorithm will be implemented directly in the PLC and it can be used in different applications.

Adaptive control is very useful for solving a variety of non-linear control problems where conventional control is not efficient, this characteristic makes these techniques attractive for a large number of control applications in industry, the algorithm is Available in the high-level programming language of PLC's SCL, this language allows flexibility for mathematical calculations.

Two neuronal control algorithms were developed, one based on an ADALINE type network and the other based on a multi-layer perceptron type network (MLP), the two control algorithms were validated in different plants together with a PID controller, where they were analyzed and The responses of each of the controls were compared.

1 Introducción

La presente investigación aborda el tema de controladores inteligentes basados en redes neuronales aplicados a un controlador lógico programable S7-1200 de la marca Siemens para el control de procesos industriales, llevando el área de control inteligente a la industria. La característica principal de estos tipos de controladores es la capacidad de controlar sistemas lineales y no lineales, además los controladores basados en redes neuronales tipo ADALINE y tipo MLP tienen una ventaja adicional, son controladores adaptativos, lo que les permite realizar un control sin tener un modelado o conocimiento detallado del sistema a controlar.

El control PID es la 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), sin embargo, en la actualidad tenemos un sector industrial cada día más exigente y competitivo, en el cual surgen nuevos retos en el área de control y automatización. Aunque los controladores PID son bastante robustos al realizar una buena sintonización, con el desgaste de los instrumentos en el sistema, se requiere estar haciendo ajustes que pueden tomar tiempo, en la industria parar un proceso genera pérdidas económicas, muchas empresas no pueden permitirse fallos o paros. Los controladores inteligentes basados en redes neuronales es una área que ya lleva años emergiendo a nivel de investigación, generando múltiples estudios, aunque, en el área industrial no ha sido muy implementada, más que todo por la complejidad matemática que maneja estos tipos de algoritmos a la hora de programarlos en autómatas.

El interés de llevar estos nuevos sistemas de control a equipos industriales y realizar pruebas físicas reales, es de interés para la comunidad científica, ya que la mayoría de investigaciones se encuentran solo a nivel de simulación. Por otra parte, los neurocontroladores en sistemas embebidos como microcontroladores han demostrado tener un excelente desempeño.

La investigación se realizó en una serie de pasos, donde se planteó en un primer lugar, el uso de un autómata que tuviera un lenguaje de programación de alto nivel, para implementar el algoritmo en este lenguaje. Se seleccionó el PLC S7-1200 como el mejor dispositivo disponible para la implementación, ya que cuenta con el lenguaje de alto nivel SCL. Al momento de profundizar en el lenguaje uno de los inconvenientes principales es la poca información académica que se encuentra de estos tipos de lenguajes de alto nivel. Al programar el algoritmo de control neuronal basado en el tipo ADALINE, se realizaron pruebas con lo que se comprobó su funcionamiento para después generar otro neurocontrolador, pero este último ya basado en redes MLP (Multicapa), en donde se realizaron pruebas del desempeño de ambos algoritmos comparándolos con el control convencional más usado en la industria que es el PID.

En la estructura del libro se revisa en un primer apartado toda la teoría detrás de este tipo de algoritmos de redes neuronales, para luego pasar por una introducción de lo que es el lenguaje SCL en los autómatas de la marca Siemens, dando unos ejemplos de su sintaxis. Se realiza la selección de las plantas, acondicionamiento e identificación de estas. En un segundo

apartado se realiza el paso a paso para la sintonización del controlador PID ha usar como punto de comparación y el desarrollo de los algoritmos de control neuronal. Para finalizar con las pruebas en las plantas seleccionadas y análisis de los resultados obtenidos.

2 Justificación

Las técnicas de control más usadas hoy en día son las tradicionales, debido a su simplicidad y robustez (Yalçm y Kurtulan, 2003), este tipo de control funciona de una manera muy óptima y robusta solo cuando la variable a controlar tiene un comportamiento lineal. Sin embargo, estos controladores deben sintonizarse para cada planta y este proceso es bastante complicado por el hecho que se debe tener información de la planta, ya sea el modelo matemático o también llamado función de transferencia, estos modelados son complicados de obtener, por tal caso conseguir los mejores coeficientes para el controlador es bastante tedioso.

Las técnicas de control avanzado resuelven estos problemas, ya que una de sus ventajas es que permite tener un buen control de sistemas no lineales, además no es necesario tener el modelado de la planta a controlar, no obstante, algunos de estas técnicas pueden ser complicadas de implementar o aun no son suficiente robustas en algunos sistemas. Por un lado, los controles basados en redes neuronales han demostrado sus ventajas en el campo de control (El-Araby, 2006), ya que se adaptan a los parámetros de la planta, con la capacidad de compensar la incertidumbre del sistema en un tiempo que puede ser largo o corto. Las ventajas de las redes neuronales es que tienen la capacidad de generalización que les permite manejar datos nuevos que no existían en un conjunto de entrenamiento (El-Araby, 2006).

Por lo tanto, el control con redes neuronales o el neurocontrol es una técnica de control avanzado adaptativo no convencional (Tanomaru y Omatu, 1992), en la actualidad hay varios estudios referente a los controladores basados en redes neuronales y muchos han validado su funcionamiento mediante simulaciones, como en (Noriega y cols., 2008) donde se diseña un neurocontrolador y se valida en una simulación, obteniendo una respuesta muy óptima; en (Meneses Jurado, 2019) se implementa controles avanzados en sistemas embebidos en este caso un microcontrolador ARM y se valida su funcionamiento en una planta real en la cual demuestra un desempeño excelente. Los resultados obtenidos por los neurocontroladores son prometedores, por lo tanto, llevar su funcionamiento a más dispositivos físicos para estudiar su desempeño es un tema de interés en la comunidad científica.

Por medio de esta investigación se implementó un algoritmo de control adaptativo basado en redes neuronales directamente en un PLC y se validó su funcionamiento en una planta escala, en búsqueda de que se pueda implementar esta técnica a futuro en diferentes procesos industriales, de tal manera que el control de los procesos industriales se vuelva más sencillo de realizar. Debido a que la técnica que se propuso es adaptativa y resuelve varios de los problemas que presentan los métodos de control clásico, evitando re-configuraciones en la sintonización del controlador que en muchos casos generan pérdidas.

3 Objetivos

3.1 General

Implementar un control neuronal directo en el controlador lógico programable S7-1200 para aplicaciones industriales

3.2 Específicos

- Estudiar la arquitectura y lenguajes de programación que soporta el PLC S7-1200.
- Programar el algoritmo de control adaptativo basado en redes neuronales.
- Optimizar y guardar el algoritmo desarrollado como un bloque de función.
- Validar el desempeño del algoritmo de control en una planta a escala.

4 Marco Teórico

4.1 Sistemas de Control

Un **sistema de control** es una interconexión de componentes que forman una configuración del sistema que proporcionará una respuesta deseada. La base para el análisis de un sistema es el fundamento proporcionado por la teoría de los sistemas lineales, que supone una relación entre causa y efecto para sus componente. La relación entrada-salida representa la relación entre causa y efecto del proceso, que a su vez representa un procesamiento de la señal de entada para proporcionar una señal de salida.(Dorf, 2005)

Un sistema de control es un arreglo de componentes cuyo objetivo es comandar o regular la respuesta de una parte del proceso, conocida como planta, sin que el operador intervenga en forma directa sobre sus elementos de salida, El operador manipula únicamente magnitudes de baja potencia denominadas de consigna, mientras que el sistema de control, a través de los accionamientos conectados en sus salidas, se encarga de producir los cambios energéticos en la planta.(Daneri, 2008)

Existen dos tipos de sistemas de control, uno se denomina a **lazo abierto** ver figura4.1, por el hecho que no recibe ningún tipo de información del comportamiento de la planta. Se puede definir un sistema de lazo abierto como aquel en el cual la acción de control es independiente de la/las señales de salida. El segundo tipo de sistemas de control se denomina **lazo cerrado** ver figura4.2 ya que su estructura denota claramente una cadena directa y un retorno o retroalimentación, formando lo que se denomina lazo de control, se considera entonces que un sistema de control a lazo cerrado como aquel en el cual la acción de control es, en cierto modo, dependiente de la/las señales de salida.(Daneri, 2008)

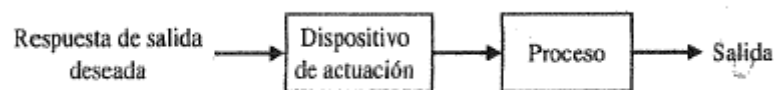


Figura 4.1: Sistema de control en lazo abierto

Fuente: Dorf y cols.

En contraste con un sistema de control en lazo abierto, un sistema de control en lazo cerrado utiliza una medida adicional de la salida real, para compararla con la respuesta de la salida deseada. La medida de la salida se denomina señal de realimentación. Un sistema de control con realimentación es aquel que tiende a mantener una relación prescrita de una variable del

sistema con otra, comparando funciones de estas variables y usando la diferencia como un medio de control.(Dorf, 2005)

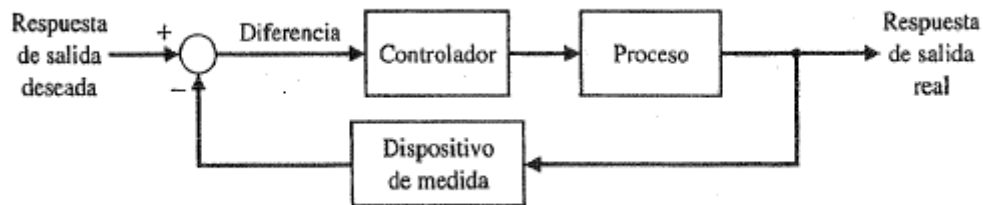


Figura 4.2: Sistema de control en lazo Cerrado

Fuente: Dorf y cols.

4.1.1 Técnicas de control Clásico

Las técnicas de control clásico más usadas en la actualidad son:

- Control de dos posiciones ON-OFF.
- Control proporcional + integral (PI).
- Control proporcional + derivativo (PD).
- Control proporcional + integral + derivativo (PID).

4.1.1.1 Control PID

El control PID es la técnica de control básico más utilizada en la industria, donde mas de un 90% de los lazos de control utilizan la acción proporcional combinada con la acción integral. Existe otro tipo de lazo que, debido a sus características dinámicas de retardo, utilizan además la acción derivativa, como ocurre en general con las temperaturas.(Acedo Sanchez, 2003)

El controlador PID tiene la función de transferencia en tiempo continuo es:

$$G_c(s) = K_1 + \frac{K_2}{s} + K_3s \quad (4.1)$$

Se dice que es 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 ningún intervalo de tiempo. La popularidad de los controladores PID se puede atribuir parcialmente a su comportamiento robusto en un rango amplio de condiciones de operación y parcialmente también a su simplicidad funcional que permite a los ingenieros operarlos de una forma simple y directa. Para implementar un controlador de este tipo, se deben determinar tres parámetros para el proceso dado: ganancia

proporcional, ganancia integral y ganancia derivativa. (Dorf, 2005) El lazo de control PID se muestra en la figura 4.3

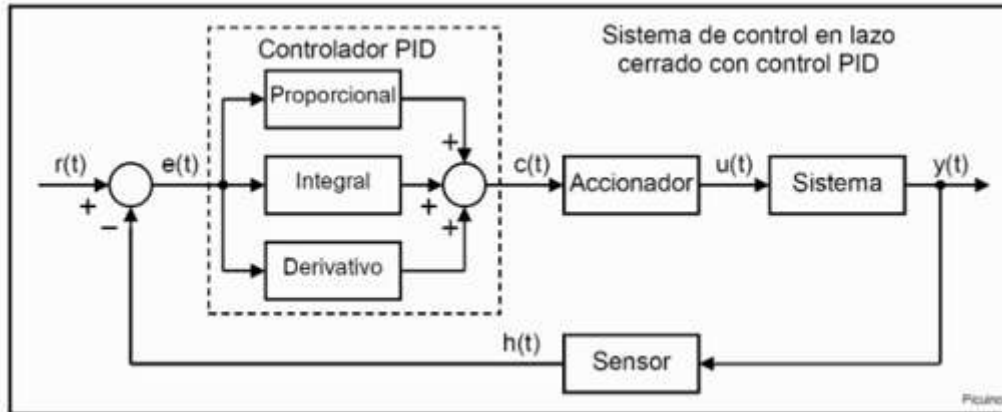


Figura 4.3: Lazo de control PID

Fuente: (Pardo, 2020)

Como se mencionaba anteriormente los controladores más usados son del tipo PI, PD y PID. Estos están conformada por tres acciones P, I y D, a continuación se expande más acerca de cada acción, tomando como referencia (Mazzone, 2002)

- P: acción de control proporcional, da una salida del control que es proporcional al error.

$$C_p(s) = K_p \quad (4.2)$$

Donde K_p es una ganancia proporcional.

- I: acción de control integral, da una salida del controlador que es proporcional al error acumulado.

$$u(t) = K_i \int_0^t e(\tau) \delta\tau \rightarrow C_i(s) = \frac{K_i}{s} \quad (4.3)$$

Donde K_i es la ganancia integral.

- D: acción derivativa, esta acción tiene carácter de previsión.

$$C_d = K_d \frac{de(t)}{dt} \quad (4.4)$$

Donde K_d es el tiempo derivativo.

- PI: acción proporcional-integral.

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(\tau) \delta\tau \rightarrow C_{pi} = K_p \left(1 + \frac{1}{T_i} s\right) \quad (4.5)$$

- PD: acción proporcional-derivativa

$$u(t) = K_p e(t) + K_p K_d \frac{de(t)}{dt} \rightarrow C_{pd}(s) = K_p + s K_p T_d \quad (4.6)$$

- PID: acción proporcional-integral-derivativa

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(\tau) d\tau + K_p T_d \frac{de(t)}{dt} \rightarrow C_{pid}(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s\right) \quad (4.7)$$

4.1.2 Técnicas de control Avanzado

El termino control avanzado no tiene una demarcación clara. Se puede definir como un conjunto de técnicas y herramientas que permiten adaptarse a las condiciones cambiantes de proceso. (Acedo Sanchez, 2003) Existen diferentes formas de control automático que responden a este tipo de denominación. En general, todas se basan en un conocimiento más o menos acabado del sistema a controlar, tanto de su fenomenología como de sus condiciones de operación, a través de un modelo matemático que lo describa en forma bastante aproximada, en todo su rango de operación y para cualquier instante de tiempo (situación relevante en procesos por lotes o “batch”). Este modelo se usa tanto como parte integrante del sistema de control como para evaluar el desempeño del sistema a través de técnicas de simulación. (Meneses Jurado, 2019)

Cada una de estas técnicas puede apoyarse en diversos cálculos matemáticos para complementar el efecto sobre el elemento final de control, siempre con el objetivo de producir estabilidad en la operación. Es necesario hacer hincapié en que el objeto principal del control avanzado es producir estabilidad en la operación. Como consecuencia de la estabilidad se puede mejorar el objetivo económico al acercarse más a los límites restrictivos de las unidades de proceso, obteniendo: (Acedo Sanchez, 2003)

- Mejora de rendimiento en productos de mayor valor añadido.
- Ahorro de energía.
- Incremento de la capacidad de producción.

Sin duda, y siguiendo la definición dada de control avanzado, existen diversidad de estrategias de control que responden a esta clasificación, principalmente provenientes del mundo académico científico. Sin embargo, la dificultad para poder ser operadas por personal con escaso nivel de formación profesional, así como para generalizarlas a sistemas de diferente naturaleza, ha provocado que pocas sean las que han trascendido del ámbito académico al industrial. (Meneses Jurado, 2019)

4.2 Redes Neuronales

La Agencia de Investigación de Proyectos Avanzados de Defensa (DARPA), define una red neuronal artificial como un sistema compuesto de muchos elementos simples de procesamiento los cuales operan en paralelo y cuya función es determinada por la estructura de la red y el peso de las conexiones, donde el procesamiento se realiza en cada uno de los nodos o

elementos de cómputo.(Caicedo B y López S, 2009)

Las redes neuronales imitan la estructura y la función del cerebro humano mediante el procesamiento de información no lineal y la computación paralela a gran escala con una capacidad de almacenamiento distribuido. Cuando se trata con varios tipos de información, las redes neuronales aprenden por sí mismas y ajustan constantemente sus sistemas para adaptarse a un entorno variable. Por lo tanto, se usan ampliamente en el reconocimiento de patrones, para predecir cambios en las cosas, optimizar decisiones, mejorar el control del proceso y otras tareas.(Ye y Kim, 2018)

Según Haykin, una red neuronal es un procesador paralelo masivamente distribuido que tiene una facilidad natural para el almacenamiento de conocimiento obtenido de la experiencia para luego hacerlo utilizable. Se parece al cerebro en dos aspectos:(Caicedo B y López S, 2009)

- El conocimiento es obtenido por la red a través de un proceso de aprendizaje.
- Las conexiones entre neuronas, conocidas como pesos sinápticos, con utilizadas para almacenar dicho conocimiento.

El algoritmo Algoritmo Levenberg-Marquardt (LM) es una red neuronal utilizada en un algoritmo de entrenamiento no lineal, donde combina el método de descenso de gradiente y el método Quasi-Newton para garantizar la velocidad de convergencia localmente rápida y mantener un mejor rendimiento general. La idea básica de este algoritmo es que cada iteración no es más larga de lo necesario a lo largo de un solo gradiente en la dirección negativa y permite buscar el error en la dirección de descenso. Además, los pesos de la red pueden optimizarse mediante el ajuste adaptativo entre el método de descenso de gradiente más pronunciado y el método de Gauss-Newton, que permite que la red converja de manera efectiva y esto mejora en gran medida la velocidad de convergencia y la generalización de la red.(Marulanda Gonzalez, 2010)

Las redes neuronales se pueden utilizar para implementar controladores altamente no lineales con pesos o parámetros internos que se auto-determinan mediante un proceso de aprendizaje, al cambiar el peso de un elemento alterará el comportamiento de la red. El objetivo es encontrar los pesos de la red para lograr una relación entrada/ salida deseada.(Nguyen y Widrow, 1990)

La ventaja de las redes neuronales artificiales frente a sistemas matemáticos o expertos es que la función gana complejidad cuanto mayor es el número y las combinaciones de estas. Al igual que las neuronas biológicas, la muerte o deterioro de una neurona afecta cuantitativamente, pero no cualitativamente. Esto les confiere características que las hacen muy adecuadas para la realización de tareas tales como identificación, reconocimiento de patrones y sobre todo control.(Marulanda Gonzalez, 2010)

La distribución de las neuronas dentro de una red neuronal artificial se realiza formando niveles de un número de neuronas determinado. Si un conjunto de neuronas artificiales recibe simultáneamente el mismo tipo de información, lo denominaremos capa. En una red podemos diferenciar tres tipos de niveles:(Caicedo B y López S, 2009)

- **Entrada:** Es el conjunto de neuronas que recibe directamente la información proveniente de las fuentes externas de la red.
- **Oculto:** Corresponde a un conjunto de neuronas internas a la red y no tiene contacto directo con el exterior.
- **Salida:** Es el conjunto de neuronas que transfieren la información que la red ha procesado hacia el exterior.

La arquitectura de una red neuronal artificial ver figura 4.4 es la forma como se organizan las neuronas en su interior y está estrechamente ligada al algoritmo de aprendizaje usado para entrenar la red. Dependiendo del número de capas, definimos las redes como monocapa y multicapa. (Caicedo B y López S, 2009)

Una red neuronal artificial es capaz de aprender por si misma para controlar sistemas dinámicos no lineales (es decir, que sus parámetros cambian conforme pasa el tiempo), esto es quizá una de las razones por las que más se utiliza este tipo de control. Una red neuronal de múltiples capas, aprende a identificar las características dinámicas del sistema. El controlador, otra red neuronal múltipara, que aprende a controlar al emulador. El controlador autodidacta se usa para controlar el sistema dinámico real. El proceso de aprendizaje continúa a medida que el emulador y el controlador mejoran y rastrean el proceso físico el cual es el calculo procedente al relacionara las entradas y salidas del sistema. (Nguyen y Widrow, 1990)

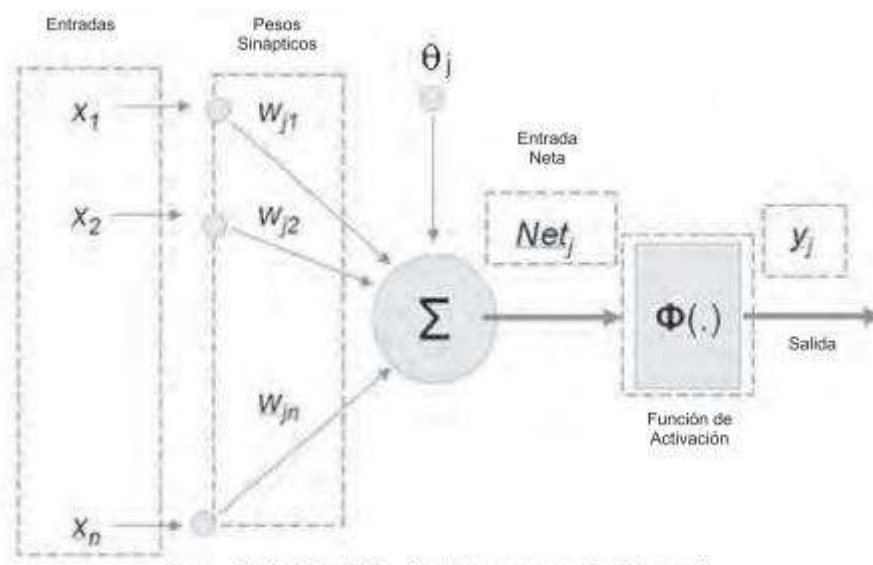


Figura 4.4: Estructura de una red neuronal artificial

Fuente: Caicedo y cols.

Las redes neuronales o sistemas neuronales artificiales constituyen en la actualidad un activo campo multidisciplinario, en el que confluyen investigadores procedentes de muy diferentes áreas, como la electrónica, física, matemáticas, ingeniería, biología o psicología. No obstante, hay que tener muy presente que en las redes neuronales artificiales no se persigue ningún tipo de ambición prometeica, como las de la IA en sus inicios, sino que se utilizan en el

control de procesos industriales, reconocimiento de vehículos en los peajes de las autopistas o la previsión del consumo eléctrico, objetivos mucho más modestos que la creación de un cerebro artificial, y extremadamente útiles desde un punto de vista tecnológico. Es en este tipo de problemas prácticos en los que los sistemas neuronales están alcanzando excelentes resultados. (Martín del Brio y Sanz, 2001)

4.2.1 El Perceptrón

El Perceptrón fue el primer modelo de RNA presentado a la comunidad científica por el psicólogo Frank Rosenblatt en 1958. Como es natural despertó un enorme interés en la década de los años sesenta, debido a su capacidad para aprender a reconocer patrones sencillos con una superficie de separación lineal, razón por la cual fue también objeto de severas críticas que terminaron por dejar en el olvido la propuesta de Rosenblatt. La estructura del Perceptrón es supremamente sencilla: en su entrada posee varias neuronas lineales que se encargan de recibir el estímulo externo de la red y a la salida presenta una única capa de neuronas, con función de activación binaria o bipolar lo cual trae como consecuencia que la salida de cada neurona esté entre dos posibles valores. (Caicedo B y López S, 2009)

La estructura del perceptrón ver figura 4.5 se inspira en las primeras etapas de procesamiento de los sistemas sensoriales de los animales (por ejemplo, el de visión), en los cuales la información va atravesando sucesivas capas de neuronas, que realizan un procesamiento progresivamente de más alto nivel. El perceptrón simple es un modelo unidireccional, compuesto por dos capas de neuronas, una sensorial o de entradas, y otra de salida. (Martín del Brio y Sanz, 2001)



Figura 4.5: Perceptrón simple y función de transferencia de su neurona

Fuente: Caicedo y cols.

Las neuronas de entrada no realizan ningún computo, únicamente envían la información a las neuronas de salida. La función de activación de las neuronas de la capa de salida es de tipo escalón o Heaviside. El perceptrón puede utilizarse tanto como clasificador, como para la representación de funciones booleanas, pues su neurona es esencialmente de tipo MacCulloch-Pitts, de salida binaria. Cada neurona del perceptrón representa una determinada clase, de modo que dado un vector de entrada, una cierta neurona responde con 0 si no pertenece a la clase que representa, y con un 1 si pertenece. Es fácil ver que una neurona tipo perceptrón solamente permite discriminar entre dos clases linealmente separables (es decir, cuyas regiones de decisión pueden ser separadas mediante una única condición lineal o hiperplano). (Martín del Brio y Sanz, 2001)

Por lo tanto, pese a su gran interés, el perceptrón presenta serias limitaciones, pues solamente puede representar funciones linealmente separables. Así, aunque pueda aprender automáti-

camente a representar complejas funciones booleanas o resolver con éxito muchos problemas de clasificación, en otras ocasiones fallará estrepitosamente. (Martín del Brio y Sanz, 2001)

4.2.2 Red Adaline

De manera casi simultánea al desarrollo del Perceptrón, Bernard Widrow y su estudiante Marcian Hoff presentaron su red neuronal ADALINE, esta red es similar al Perceptrón, pero en vez de tener una función de activación binaria o bipolar posee una función de activación lineal. La red ADALINE, a pesar de tener las mismas limitaciones de la red Perceptrón, fue un gran adelanto en el desarrollo de las redes neuronales pues el mecanismo de aprendizaje que utiliza es basado en elementos de la teoría de optimización; específicamente se usa la propuesta del gradiente descendente para la deducción de la regla de entrenamiento. El uso del gradiente descendente es la base de los algoritmos de aprendizaje que hoy se usan para entrenar redes neuronales más sofisticadas. Uno de los aspectos más interesante de la red ADALINE es su aplicación en problemas de filtrado de señales, pues su estructura se ajusta a la de un filtro adaptativo. Es importante recalcar que la red tipo ADALINE puede tener más de una neurona en su capa de procesamiento, dando origen a un sistema con N entradas y M salidas. (Caicedo B y López S, 2009)

La estructura de la red ADALINE ver figura 4.6 es similar a la del perceptrón con la única diferencia de que la red ADALINE posee una función de activación lineal. No obstante, la

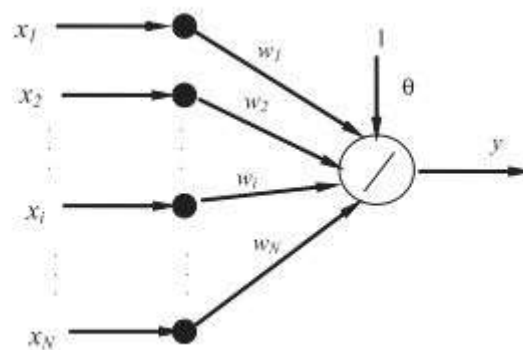


Figura 4.6: Estructura de una red neuronal tipo Adaline

Fuente: Caicedo y cols.

diferencia más importante con el perceptrón y red adaline reside en la regla de aprendizaje que implementa. En la adalina se utiliza la regla de Widrow-Hoff, también conocida como regla LMS (Least Mean Squares, mínimos cuadrados), que conduce a actualizaciones de tipo continuo, siendo la actualización de los pesos proporcional al error que la neurona comete. (Martín del Brio y Sanz, 2001)

La adalina se viene utilizando con asiduidad desde los años sesenta como filtro adaptativo, por ejemplo, para cancelar el ruido en la transmisión de señales (un ejemplo clásico es su empleo como supresor de ecos en las comunicaciones telefónicas por satélite. Su utilidad se ve

limitada por tratarse de un sistema lineal. Así, solamente podrá separar correctamente patrones linealmente independientes, fallando en ocasiones ante patrones linealmente separables, que el perceptrón siempre discrimina. No obstante, ante patrones no separables linealmente, los resultados que proporciona son en promedio mejores que los del perceptrón pues la adalina siempre opera reduciendo el error cuadrático medio al mínimo posible. (Martín del Brio y Sanz, 2001)

4.2.2.1 Regla delta

La regla de Widrow-Hoff o LMS, que en un caso particular es conocida como regla delta, constituye el algoritmo de aprendizaje asociado a la adalina. La regla LMS conducirá a asociaciones perfectas cuando sean linealmente independientes, proporcionando cuando no lo sean una matriz de pesos óptima desde el punto de vista de los mínimos cuadrados. Así, la regla delta puede considerarse en realidad como una versión iterativa aproximada de la basada en la pseudo-inversa, para el caso de vectores estocásticos. (Martín del Brio y Sanz, 2001)

Para explicarlo mejor supongamos que nos encontramos en un sistema montañoso muy complejo en un lugar a gran altura y totalmente cubierto por neblina. Estamos perdidos, pero sabemos que en el valle hay una población donde podremos obtener ayuda, ¿qué estrategia utilizamos para llegar a dicha población? Muy seguramente estamos de acuerdo en que lo mejor es empezar a bajar y siempre mantener esta tendencia, por ejemplo, podríamos seguir el cauce de un río, pues esta corriente de agua nos garantiza que siempre caminaremos paulatinamente hacia un lugar más bajo. (Caicedo B y López S, 2009)

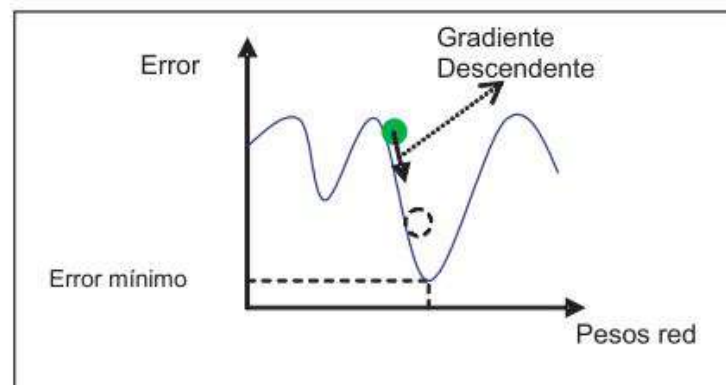


Figura 4.7: Gradiente Descendente

Fuente: Caicedo y cols.

Este es el principio del algoritmo utilizado en el proceso de aprendizaje de la red ADALINE, en donde, partiendo desde un punto aleatorio en la superficie de error, buscaremos un punto donde el error sea mínimo, siguiendo una trayectoria descendente, como se ilustra en la figura 4.7. Esta técnica de búsqueda se denomina Algoritmo de Gradiente Descendente. (Caicedo B y López S, 2009)

Debido a la linealidad de la neurona de la adalina, la función error es cuadrática en los pesos, por lo que define una superficie en forma de paraboloides. Un paraboloides, como la parábola

en el plano, usualmente posee un único mínimo, aunque en ocasiones puede presentar una forma degenerada, con uno o mis canales, pero todos de la misma profundidad. En cualquiera de los dos casos la función del error es mínima en ese punto o en cualquiera de los canales, y la regla del gradiente nos lleva directamente a él, puesto que siempre desciende por la superficie de error. Por ello, la regla LMS alcanza siempre el mínimo global, sin importar la configuración de pesos de partida, constituyendo uno de los pocos casos en redes neuronales en el que se puede realizar una afirmación de este tipo. (Martín del Brio y Sanz, 2001)

4.2.3 Perceptron Multi-capa

Debido a la imposibilidad de solucionar problemas de clasificación no lineales que presenta el Perceptron propuesto por Rosenblatt, y redes algo más evolucionadas como el Adaline, el interés inicial que las redes neuronales artificiales habían despertado, decayó fuertemente y sólo quedaron unos pocos investigadores trabajando en el desarrollo de arquitecturas y algoritmos de aprendizaje capaces de solucionar problemas de alta complejidad. Esta situación fue ampliamente divulgada por Misky y Papert en su libro *Perceptrons*; lo que significó prácticamente el olvido científico de la propuesta de Rosenblatt, sin embargo, ya intuía la manera de solucionarlo, el autor del Perceptron, percibía que si incluía una capa de neuronas entre las capas de entrada y salida, podía garantizar que la red neuronal artificial sería capaz de solucionar problemas de este tipo y mucho más complejos. A mediados de la década de los setenta, Paul Werbos en su tesis doctoral propone el Algoritmo Backpropagation, que permite entrenar al Perceptron multicapa y posibilita su aplicación en la solución de una gran variedad de problemas de alta complejidad. (Caicedo B y López S, 2009) (MLP de

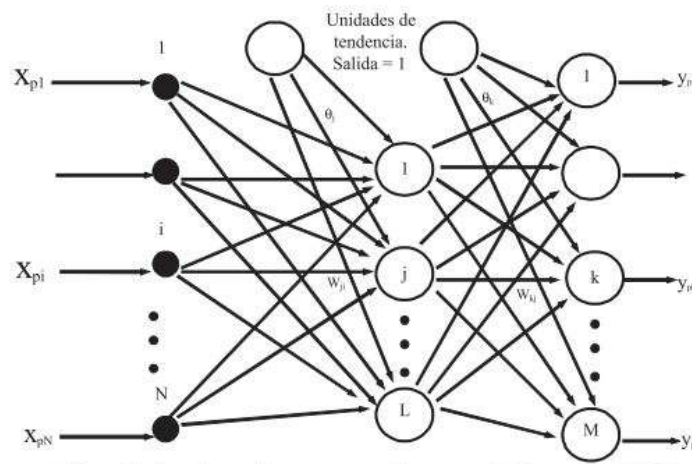


Figura 4.8: Arquitectura general de un perceptron multi-capa

Fuente: Caicedo y cols.

sus siglas en inglés Multi Layer Perceptron), que a diferencia del Perceptron y del Adaline, posee al menos tres niveles de neuronas, el primero es el de entrada, luego viene un nivel o capa oculta y, finalmente, el nivel o capa de salida. Al tener más de una capa oculta, se aumenta fuertemente la complejidad computacional del algoritmo de aprendizaje. En las redes

neuronales artificiales, el término conectividad se refiere a la forma como una neurona de una capa cualquiera está interconectada con las neuronas de la capa previa y la siguiente. Para el MLP, la conectividad es total porque si tomamos una neurona del nivel de entrada, ésta estará conectada con todas las neuronas de la capa oculta siguiente, una neurona de la capa oculta tendrá conexión con todas las neuronas de la capa anterior y de la capa siguiente. Para la capa de salida, sus neuronas estarán conectadas con todas las neuronas de la capa oculta previa, para mayor claridad observemos la figura 4.8. Por lo general, se le implementan unidades de tendencia o umbral con el objetivo de hacer que la superficie de separación no se quede anclada en el origen del espacio n-dimensional en donde se esté realizando la clasificación. La función de activación que utilizan las neuronas de una red MLP suele ser lineal o en la mayoría de los casos sigmoïdal. (Caicedo B y López S, 2009). EL funcionamiento de manera conceptual del algoritmo de aprendizaje Backpropagation es:

- Se selecciona un conjunto de patrones claramente representativos del problema a solucionar, con los cuales se entrena la red. Esta fase es fundamental pues dependiendo de la calidad de los datos utilizados para el entrenamiento, será la calidad de aprendizaje de la red.
 - Se aplica un vector de entrada a la red y se calcula la salida de las neuronas ocultas, propagando estos valores hasta calcular la salida final de la red.
 - Calcular el error entre el valor deseado y la salida de la red.
 - Propagar el error hacia atrás, es decir, estimar el error en la capa oculta con base en el error de la capa de salida.
 - Se modifica los pesos de la capa de salida y de las capas ocultas con base en una estimación del cambio de los pesos Δw en cada una de las capas que a su vez, depende del cálculo del error de la capa de salida y de la estimación del error en las capas ocultas realizado en los pasos anteriores.
 - Se verifica la condición de parada del algoritmo ya sea por que el error calculado en la salida es inferior al impuesto por el problema bajo análisis o por ejemplo, porque se ha superado un número determinado de iteraciones, en cuyo caso se considera que es necesario hacer ajustes al diseño de la red, pues la solución no converge, o simplemente que se debe ampliar el número máximo de iteraciones a realizar
 - En caso de no cumplir con ninguna de las condiciones de parada, se le vuelve a presentar a la red un patrón de entrenamiento. (Caicedo B y López S, 2009)
-

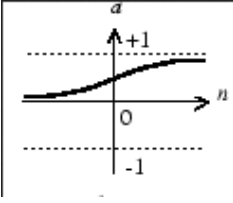
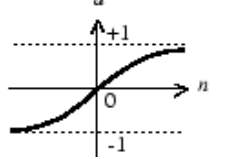
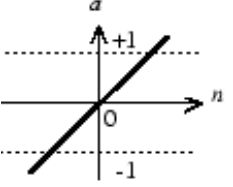
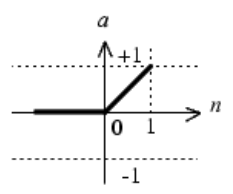
Funciones	Relación Entrada/salida Modelo matemático	Gráfica
Logsig	$a = \frac{1}{1 + e^{-n}} \quad (4.8)$	
Tansig	$a = \frac{2}{1 + e^{-2n}} - 1 \quad (4.9)$	
Purelin	$a = n \quad (4.10)$	
Poslin	$a = \begin{cases} n, & \text{if } n > 0 \\ 0, & \text{if } n \leq 0 \end{cases} \quad (4.11)$	

Tabla 4.1: Funciones de activación

4.2.4 Redes Neuronales como controladores

Las redes neuronales han sido muy utilizadas últimamente en el control de sistemas con dinámica difícil o sistemas que cambian de parámetros constantemente, como sistemas con altas no linealidades, es por ello que existen dos clases de redes neuronales que han recibido importancia en el área de inteligencia artificial en los últimos años, por una parte, se tiene las redes neuronales multi-capa y las redes recurrentes. Las redes multi-capa han demostrado ser extremadamente exitosas en problemas de reconocimiento de patrones, mientras que las redes recurrentes se han utilizado en memorias asociativas, así como para la solución de problemas de optimización. Desde el punto de vista teórico de los sistemas, las redes multi-capa representan mapas no lineales estáticos, mientras que las redes recurrentes están representadas por sistemas de retroalimentación dinámica no lineal. A pesar de las aparentes diferencias entre las dos clases de redes, existen razones convincentes para verlas de manera unificada. De hecho, los elementos dinámicos y la retroalimentación se utilizarán cada vez más en el futuro, lo que dará como resultado sistemas complejos que contengan ambos tipos de redes. Esto, a su vez, requerirá un tratamiento unificado de tales redes (Meneses Jurado, 2019). El objetivo del controlador es ajustar la señal de control para que la salida de la planta

tienda a la salida del modelo de referencia. El rendimiento de este algoritmo depende de la elección de un modelo de referencia correcto y la derivación de un buen mecanismo de aprendizaje. El error entre la salida del modelo y la respuesta de la planta es usado como señal de retroalimentación para el controlador. En el entrenamiento las ponderaciones del modelo de la planta permanecen fijas, mientras las ponderaciones del controlador se ajustan por la retropropagación del error a través del modelo neuronal de la planta.(Quiña, 2014)

El potencial de los Redes neuronales para aplicaciones de control radica en que:

- las redes neuronales podrían usarse para aproximar cualquier mapeo continuo a través del aprendizaje.
- Pueden realizar procesamiento paralelo y tolerancia a fallas.

Una de las arquitecturas de red más populares es un perceptrón multicapa con el algoritmo de retropropagación. Está demostrado que se puede utilizar un perceptrón de cuatro capas (con dos capas ocultas) para aproximar cualquier función continua con la precisión deseada. el algoritmo de retropropagación se ha utilizado con éxito para la clasificación de patrones, aunque su desarrollo original puso más énfasis en las aplicaciones de control. También se ha demostrado que, en general, los sistemas de control no lineales se pueden estabilizar utilizando redes de cuatro capas.(Cui y Shin, 1993)

La relevancia de las redes neuronales en el control de procesos radica en que se adaptan a los cambios de la planta, de esta manera tienen un comportamiento satisfactorio en plantas con dinámica variables no lineales. Al adaptarse a los cambios de la planta, no se requiere conocer el sistema dinámico de la misma, sino que estas adaptan el controlador a ellas, la desventaja está en que este entrenamiento es pesado computacionalmente.(Jaimes y Giraldo, 2015) En la figura 4.9 se muestra el esquema propuesto para el controlador neuronal auto-

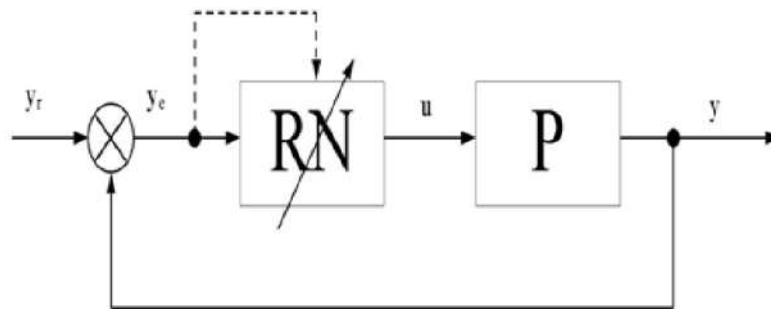


Figura 4.9: Red neuronal como controlador

Fuente:(Noriega y cols., 2008)

ajustable. La red neuronal que hace las funciones del controlador es un perceptrón de 3 capas de neuronas (una capa oculta) cuyos coeficientes de peso se ajustan al utilizar el algoritmo de retro-propagación de error.(Noriega y cols., 2008)

Las ecuaciones de actualización de pesos se pueden interpretar como las ecuaciones de adaptación del regulador neuronal en lugar de como ecuaciones de entrenamiento. No se requiere

generalmente, un entrenamiento previo de la red neuronal y una vez que se cierra el lazo de control, los coeficientes de peso se ajustan en unos pocos periodos de control, llevando el error de control a cero.(Noriega y cols., 2008)

4.3 PLC (Controlador Lógico Programable)

(Sanchis Llopis y cols., 2010)



Figura 4.10: Controlador lógico programable (PLC)

Fuente: Siemens

Un autómata programable industrial (API) es un equipo electrónico, programable en lenguaje no informático, diseñado para controlar en tiempo real y en ambiente de tipo industrial procesos secuenciales. El autómata programable también se conoce como PLC, que es la sigla de Programmable Logic Controller. Tal y como se resume en la definición, se trata de un computador especial, tanto en el software como en el hardware. En el software, porque se programa en un lenguaje especial diseñado específicamente para generar de forma sencilla el programa que implementa el algoritmo de control de procesos secuenciales (de sistemas de eventos discretos), y porque el algoritmo de control programado es ejecutado de forma periódica en un ciclo temporal que es lo bastante breve como para poder controlar los procesos en tiempo real. En el hardware, porque utiliza componentes robustos que soportan condiciones de trabajo adversas, como las que se dan en ambientes industriales (polvo, temperatura, vibraciones, etc.), y porque su constitución física incluye los circuitos de interfaz necesarios para conectarlo de forma directa a los sensores y actuadores del proceso. (Sanchis Llopis y cols., 2010)

El PLC es un instrumento electrónico que sirve de herramienta para dar solución a problemas de automatización especialmente en el ámbito industrial, dentro de los lenguajes de programación soportados están: **el lenguaje escalera, bloques funcionales y texto estructurado.**(Quiña, 2014)

Como computador que es, tiene un procesador que es el que ejecuta el programa almacenado en la memoria de programa. La memoria de programa y la de datos están físicamente separadas, constituyendo una arquitectura tipo Harvard. Además, la memoria de datos está separada en dos tipos, que en la figura se denominan memoria de datos y memoria interna. Esta última se utiliza para almacenar los valores de las señales de entrada y salida, por lo que están conectadas con los módulos de entradas y salidas, que son los elementos de interfaz donde se conectan los sensores y actuadores del proceso. También dispone de periféricos para comunicas con otros dispositivos, como pantallas táctiles, ordenadores u otros autómatas.

4.3.1 Lenguajes de Programación

En la siguiente tabla 4.2 se encuentran mencionados los lenguajes de PLC's más utilizados en la actualidad. En el siguiente apartado se verán los lenguajes de programación en PLC's que se trataran en esté trabajo.

4.3.1.1 Texto Estructurado (ST)

ST es un lenguaje de programación de alto nivel, similar a la Programación Pascal. ST es desarrollado y publicado por la Comisión Internacional Electro-técnica (IEC) en IEC 61131-3 International Standard en 1993. El estándar consiste de 5 lenguajes de programación PLC, donde la Programación LADDER es la más conocida y utilizada. La programación ST para los Controles de PLC ha sido publicada en varias ocasiones desde aproximadamente 2010; y desde 2015.(Antonsen, 2019) En los PLC de Siemens, la programación es llamada Lenguaje de Control Estructurado (SCL), el cual incluye algunas diferencias en relación al lenguaje de programación Estructurado de Texto (ST).

ST es un lenguaje de programación muy flexible y universal. El código del programa ST puede ser fácilmente replicado entre diferentes tipos de PLC y puede ser enviado vía e-mail, debido a que se encuentra basado en texto y no en gráficas como la programación LADDER lo hace. El código del programa ST es similar a las oraciones de texto y el trabajo es llevado a cabo como en un programa procesador de textos (como, por ejemplo, Microsoft Word); lo cual lo hace más fácil trabajar con él. Consecuentemente, los mismos métodos de trabajo son aplicados como en un programa procesador de textos. (Antonsen, 2019)

Debido a su muy estructurada naturaleza, ST es ideal para tareas basadas en matemáticas complejas, reutilización de códigos o toma de decisiones (por ejemplo, optimización automática de energía, algoritmos, recolección y regulación de datos en plantas de proceso). Contando con la experiencia con programación PLC, la transición hacia otros lenguajes de programación con Control PLC y automatización, deberá ser más sencilla; es decir, robótica de programación o programación Visual Basic. En los últimos años, aún más compañías han cambiado a la Programación ST, lo cual se debe al hecho de que el ST provee una serie de ventajas comparado con los otros cuatro lenguajes de programación PLC (LADDER, SFC, FBD e IL).(Antonsen, 2019) Las ventajas son las siguientes:

- El código de programación ST puede ser replicado relativamente fácil entre diferentes tipos de PLC.
 - Es el lenguaje PLC más sencillo para cálculos matemáticos, fórmulas y algoritmos 2) y una gran cantidad de datos (big-data).
-

- Las soluciones PLC son más demandantes ahora que 20 años atrás.
- Muchos lenguajes generalizados de programación de PC (C++, C, VB, PASCAL) recuerda mucho la estructura del programa ST.
- Los otros lenguajes PLC (LAD, SFC y FBD) requieren que algunas partes de ellos estén programadas en ST.
- Utiliza menos espacio cuando el código PLC debe ser documentado o descrito.
- Es el lenguaje de PLC más sencillo para el control de versiones a través de comentarios en el código de programa. (Antonsen, 2019)

4.3.1.2 Lenguaje de Control Estructurado (SCL)

SCL “Structured Control Language” es el lenguaje de texto estructurado para PLC’s de Siemens y está basado en el lenguaje de alto nivel PASCAL. Permite una fácil integración en el contexto de una solución global para un problema de automatización ya que un bloque programado en SCL puede ser llamado desde un bloque escrito en Ladder, en grafcet, en AWL o en FUP y a la inversa, un bloque escrito, por ejemplo, en KOP puede ser llamado desde un bloque escrito en SCL. Al mismo tiempo SCL trata todas las áreas de memoria del PLC como variables globales, lo que permite, como en el resto de lenguajes, intercalar una dirección absoluta de memoria (entrada, salida, marca, DB, periferia, etc.) en el área de instrucciones del bloque como si se tratara de una variable del bloque. (Barnes, 2007)

SCL está especialmente indicado para la resolución de los siguientes tipos de problemas:

1. Problemas matemáticos complejos como los siguientes ejemplos:
 - Función de normalizado de variables analógicas.
 - Cálculo de volúmenes y pesos de depósitos cónico-cilíndricos.
 - Cálculo de tiempos de activación o funcionamiento de dispositivos.
 2. Problemas de tratamiento de datos, como los siguientes ejemplos:
 - Filtrado de una variable analógica.
 - Determinación de media y valores extremos en un grupo de valores.
 - Detección de errores repetidos por desviaciones fuera de tolerancia en distintos instrumentos.
 3. Manejo de datos en matrices, como los siguientes ejemplos.
 - Determinación de la posición de carga y descarga de piezas en un almacén.
 - Mantenimiento de un histórico de eventos en un PLC.
 - Impresión de un histórico de valores de producción en un dosificador de colas.
 - Gestión de la presentación en pantalla de datos de mantenimiento y funcionamiento de válvulas y motores.
 - Gestión de contadores de producto en máquina plegadora de sábanas. (Barnes, 2007)
-

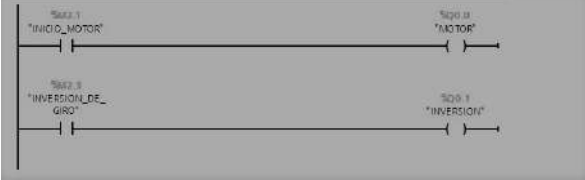
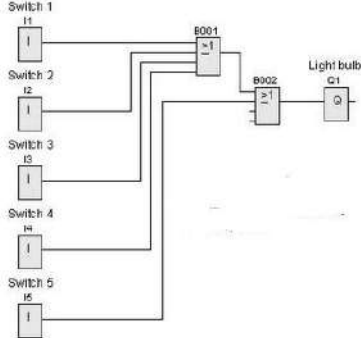
Lenguaje	Siglas	Ejemplo
Escalera (Ladder)	LD	
Bloques de Funciones	FBD	
Lista de Instrucciones	IL	<p>Instruction list language</p> <pre>LD %I1.1 R %C8 LD %I1.2 AND %M0 CU %C8 LD %C8.D ST %Q2.0</pre>
Texto Estructurado	ST	<pre>IF S1 = TRUE THEN K1:= TRUE; END_IF; IF S2 = TRUE THEN K1:= FALSE; END_IF;</pre>
Lenguaje de Control Estructurado	SCL	<pre>#Fconv := 60 * (#Frecuencia_Memoria(Hz))^4; #Count_Encoder_Real := DINT_TO_REAL(#FC_Count_Encoder); #Giros := #Count_Encoder_Real / #Resolucion; IF #M_Clock = TRUE THEN // Statement section IF #M_count:=#Count_Encoder; #rpm := #Giros * #Fconv; #M_rpm := #rpm; ELSE #M_rpm := #rpmant; END_IF;</pre>

Tabla 4.2: Lenguajes de programación en PLC's

5 Metodología

En la mayoría de los estudios anteriormente hechos sobre redes neuronales como controladores aplicados en PLC's, se encuentra a nivel de simulación como en (Quiña, 2014) en donde efectivamente se evidencia un desempeño óptimo, sin embargo, en muchos casos al momento de llevarlo a la practica se encuentra con muchos obstáculos, los cuales no se consideran en las simulaciones. En este proyecto se realizó la evaluación del algoritmo neuronal programada en un PLC en una planta industrial a escala, dicha planta debe contener instrumentos netamente industriales o semi-industriales para simular el ambiente correspondiente, cabe recalcar que el proceso industrial a controlar corresponde a un lazo de control individual por cada planta.

Se seleccionaron unos lazos de control industriales a escala, en donde se realizo las adecuaciones necesarias para poder realizar las pruebas. El objetivo fue evaluar el comportamiento de un neurocontrolador aplicado en un PLC en este caso en especifico el autómatas S7-1200 de SIEMENS, por tal razón para evaluar su funcionamiento se comparo con una referencia existente, entonces, dicha referencia que se tenía como punto de partida es un control clásico PI o PID, por consiguiente a cada proceso que se escogió se sintonizó un control clásico para después compararlo con el control neuronal, analizando las diferencias en las respuesta de cada controlador considerando las mismas condiciones iniciales.

Teniendo en cuenta lo anteriormente dicho se procedió a una selección de plantas, en donde se estableció puntos necesarios para la buen ejecución del proyecto, teniendo en cuenta la caracterización es instrumentación de las plantas escogidas.

5.1 Selección de Plantas

Como se planteo en este trabajo todo se va controlar desde el PLC S7-1200 en donde se deben tener muchas consideraciones como los niveles de señal de entrada y salida estándar, para la selección de los instrumentos que conforman el lazo de control en donde se realizó las pruebas del algoritmo, por lo tanto se tuvieron las siguientes consideraciones:

- Los instrumentos utilizados sean industriales o semi-industriales.
- La transmisión de señal este acorde al nivel estándar 0 - 10 (V) o 4 - 20 (mA).
- Los elementos finales de control sean proporcionales, es decir, se controlen mediante una señal análoga.
- Los elementos que conformen la planta deben recrear un lazo de control cerrado.
- Las variables a controlar de cada planta deben tener un comportamiento distinto.
- La dinámica en cada planta debe ser diferente.

- Se tuvo en principal consideración los elementos disponibles en el laboratorio de control de la Universidad de Pamplona.

Teniendo en cuenta las anteriores consideraciones, se escogió tres plantas disponibles en el laboratorio de control de la Universidad de Pamplona, algunos elementos se adquirieron para completar los tres lazos de control cerrado, las plantas que se seleccionaron se nombran a continuación, y en la siguiente sección se detalla cada uno de sus componentes:

- Planta para el control de Velocidad.
- Planta para el control de Temperatura.
- Planta para el control de Flujo.

5.2 Descripción de las plantas

En este apartado se detallan cada uno de los componentes que conforman las plantas seleccionadas anteriormente, se muestra las características de cada uno, y de que manera la interacción entre los componentes conforman el lazo de control cerrado.

5.2.1 Planta para el control de Velocidad

En esta planta el objetivo era controlar la velocidad en RPM (Revoluciones por minuto) de un motor trifásico, esto se hacía regulando la potencia a través de un variador de frecuencia, que recibe la señal directamente del PLC, la realimentación de la planta para cerrar el lazo de control se realiza por medio de un encoder que se encuentra directamente conectado al eje del motor. La dinámica de esta planta es rápida, ya que la variable de control tiene un intervalo de respuesta pequeño ante una señal de control proveniente del PLC.

La figura 5.1 muestra el lazo de control cerrado de la planta en bloques especificando la participación de cada componente.

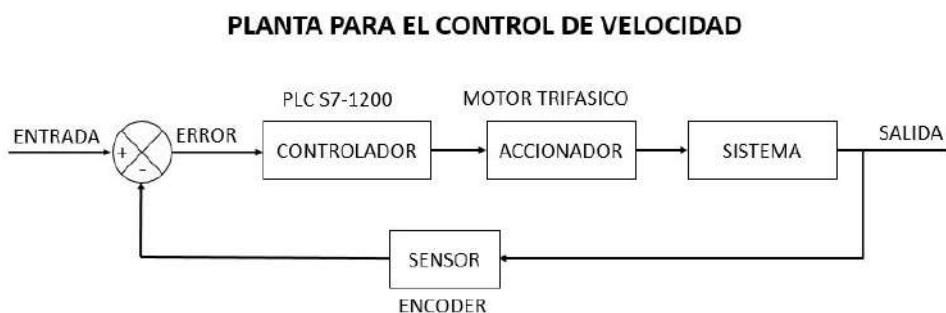


Figura 5.1: Diagrama del lazo de control de la planta de velocidad

Fuente: Autor

5.2.1.1 Componentes

5.2.1.1.1 Motor Monofásico AC Un motor trifásico ver figura 5.2 es una maquina eléctrica que transforman la energía eléctrica (corriente alterna) en energía mecánica rotacional, el suministro es trifásico (3 fases) cuyas fases están desfasadas 120° . Características del motor,



Figura 5.2: Motor monofásico

Fuente: Autor

trifásico en la siguiente tabla se muestra los datos que se encuentran en su chapa.

Artículo	Motor trifásico Dayton
Modelo	2N103R
Voltaje	220V
Potencia	1/2 HP
Corriente	2.0 A
Frecuencia	60Hz & 50Hz
RPM	1725

Tabla 5.1: Características del Motor trifásico

5.2.1.1.2 Variador de Frecuencia Un variador ver figura 5.3 es un dispositivo usado para el control de la velocidad de un motor, en este caso un motor trifásico, este aparato controla la frecuencia de alimentación que se le suministra al motor, permitiendo con esto el ajuste de una velocidad, a cierta frecuencia.



Figura 5.3: Variador de Frecuencia Omron SYSDRIVE 3GMV INVERTER

Fuente: Autor

En la siguiente tabla se pueden visualizar las principales características más generales del dispositivo.

Artículo	Variador de Frecuencia SYSDRIVE 3GMV INVERTER
Modelo	3G3MV-A2004
Voltaje de entrada	200-230V (trifásica)
Voltaje de salida	0-230V
Potencia máxima del motor aplicable	0.55 Kw o 0.74 HP
Corriente de salida máxima	3.0 A
Frecuencia	60Hz & 50Hz
Señal de Control de entrada	0-10V

Tabla 5.2: Características generales del variador de frecuencia

5.2.1.1.3 Encoder Un encoder es un dispositivo que nos convierte el movimiento rotatorio en una señal eléctrica que se pueda detectar por un dispositivo de control.



Figura 5.4: Encoder rotatorio LPD3806-360BM-G5-24C

Fuente: Autor

El encoder rotatorio usado es de la referencia LPD3806-360BM-G5-24C ver figura 5.4, a continuación se muestra la tabla 5.3 con sus principales características:

Artículo	Encoder Rotatorio incremental LPD3806-360BM-G5-24C , Diámetro 38mm, eje 6mm
Tipo de sensor	Sensor óptico
Voltaje de Funcionamiento	5 - 24 VDC
Consumo de Corriente	Max 40mA
Resolución del encoder	360 pulsos por vuelta
Frecuencia de Respuesta	Max 20 KHz
Máxima velocidad electrónica	2000 RPM
Tipo de Salida	Sensor digital
Fase de Salida	Fase A y B
Tipo de Salida	NPN Colector abierto
Máxima velocidad mecánica	5000 RPM
Temperatura ambiente de operación	(-10 70)° C
Cables	
Rojos	VCC
Negros	0V
Verdes	Fase A
Blancos	Fase B

Tabla 5.3: Características del Encoder

En la figura 5.5 se muestra la señal proveniente por las fases A y B, son altos y bajos que detecta el controlador, dependiendo de la resolución del encoder en este caso 360, si el controlador detecta 360 flancos indica que el eje del encoder ha dado una vuelta, el desfase entre A y B es de aproximadamente 90 grados y esto nos ayuda a saber la dirección en la que

esta girando el motor.

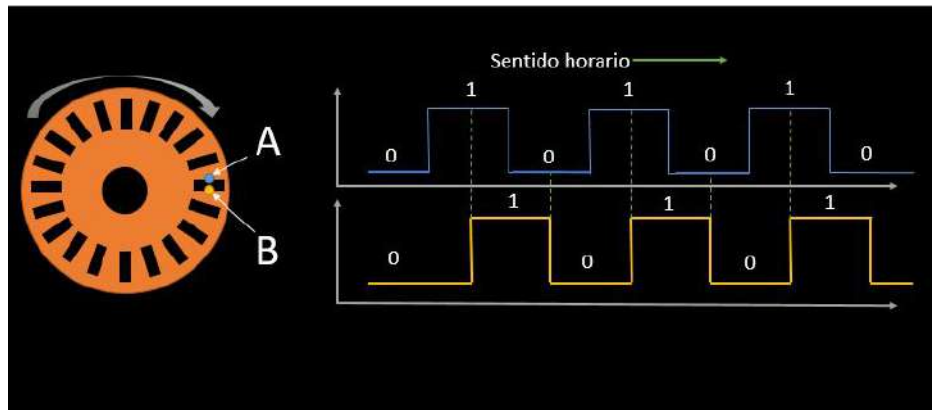


Figura 5.5: Funcionamiento Encoder

Fuente: Autor

5.2.2 Planta para el control de Temperatura

En esta planta el objetivo es controlar la temperatura de un líquido en un recipiente por medio de un calentador AC, para regular la potencia de este calentador se utiliza el relé de estado sólido (SSR) y para sensar la temperatura actual se usa un sensor RTD PT100 el cual es elemento que realimenta el lazo de control de la planta. La dinámica de esta planta es lenta, ya que la variable controlada tiene una respuesta lenta ante las señales de control. La figura 5.6 muestra el lazo de control cerrado de la planta en bloques especificando la participación de cada componente.

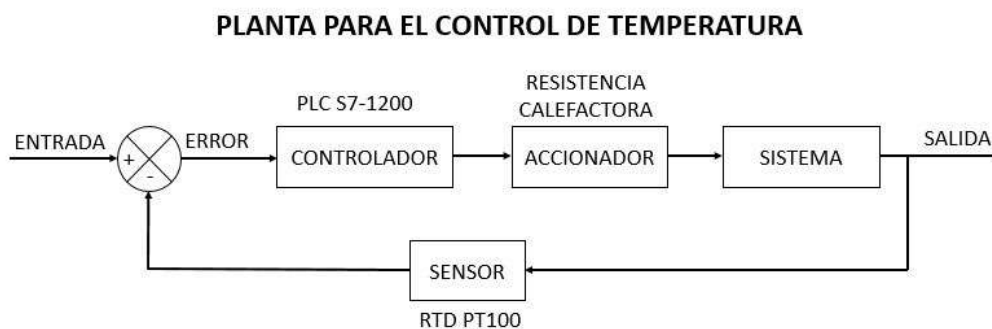


Figura 5.6: Diagrama del lazo de control de la planta de temperatura

Fuente: Autor

5.2.2.1 Componentes

5.2.2.1.1 Sensor de Temperatura RTD PT100 Una PT100 ver figura 5.7 es un sensor de temperatura RTD (detector resistivo de temperatura) ya que varía su resistencia según la temperatura ambiente que lo rodea, es un sensor muy usado en la industria.

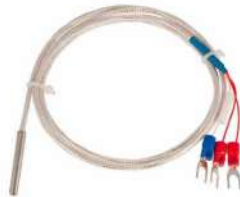


Figura 5.7: Sensor de temperatura PT100

Fuente: Ssdielect (Ssdielect, 2011)

La siguiente tabla 5.4 muestra las características más relevantes.

Artículo	Sensor de temperatura RTD PT100 (platino)
Resistencia a 0°C	100 Ohm
Precisión	+ o - 0.5° C
Rango de operación	-200 °C a +550° C
Variación de resistencia por grado	0.385 Ohm

Tabla 5.4: Características sensor de temperatura PT100

5.2.2.1.2 Resistencia Calefactora tubular AC Consiste en una resistencia que genera calor al oponerse al paso de la corriente alterna, esta resistencia es tubular mecánica de 8 pulgadas de largo ver la figura 5.8.



Figura 5.8: Resistencia Calefactora tubular

Fuente: Mercado Libre Colombia (ELECTRO HER, 2018)

5.2.2.1.3 Relé de estado solido (SSR) Un relé de estado solido ver figura 5.9 o SSR por sus siglas en ingles es un dispositivo que funciona como interruptor electrónico para permitir el paso de energía en su salida, al aplicarle una intensidad eléctrica en su entada. Es un dispositivo muy usado en la industria para realizar la conmutación de dispositivos que tienen una gran carga.



Figura 5.9: Relé de estado solido SRPH1-A220

Fuente: Autor

Las características del SSR que se va a trabajar se encuentran en la siguiente tabla 5.5:

Artículo	Relé de estado solido Autonics SRPH1-A220
Control disponible	Control por fase y control por ciclo
Voltaje de entrada nominal	Entrada analógica de 4-20 mA
Voltaje de carga nominal	100-240 VCA
Corriente de carga nominal	20A
Frecuencia de trabajo	50/60Hz

Tabla 5.5: Especificaciones del SSR

5.2.3 Planta para el control de Flujo

En esta planta se va a controlar el flujo de un líquido, la cantidad de líquido es regulado por una válvula proporcional actuando esta como el elemento final de control, la medición del flujo se realiza por medio de un sensor de flujo de paletas, que incorpora un encoder, esta sensor de flujo realiza la retroalimentación en el lazo de control cerrado. La dinámica de la planta es un poco más lento que el de velocidad. La figura 5.10 muestra el lazo de control cerrado de la planta en bloques especificando la participación de cada componente.

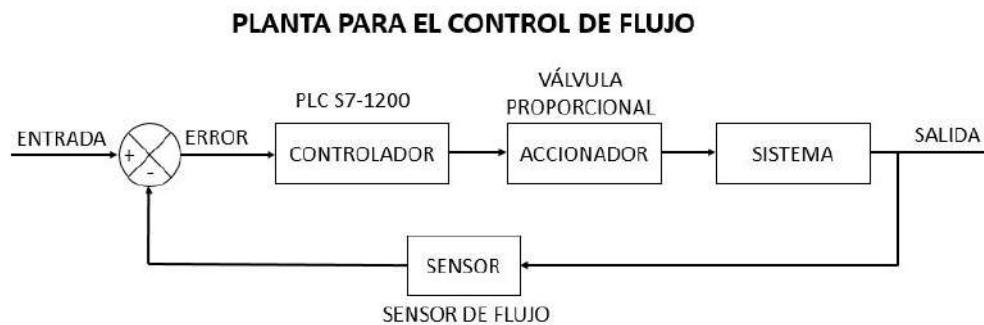


Figura 5.10: Diagrama del lazo de control de la planta de control de flujo

Fuente: Autor

5.2.3.1 Componentes

5.2.3.1.1 Válvula proporcional Una válvula proporcional es un dispositivo que cierra y abre proporcionalmente que permite el flujo de cierta cantidad de líquido desde su entrada a su salida. La siguiente tabla 5.6 contiene las características más relevantes del instrumento.



Figura 5.11: Válvula Proporcional

Fuente: Autor

Artículo	Válvula proporcional de bola DN8 1/4"
Material del cuerpo	Bronce
Voltaje de funcionamiento	9-24V DC
Señal de control	4-20mA o 0-10 V
Tiempo de giro 0°a 90°	7s
Presión máxima	150 Psi
Corriente de consumo	500mA
Temperatura de trabajo	-15°C a 60°C
Accionamiento manual	Ninguno
Temperatura del liquido	1°C a 95°C
Aplicación	Agua / Aire
Cables	
Rojo	VCC
Negro	0V
Verde	Entrada Señal de Control
Blanco	Salida PWM OC

Tabla 5.6: Características válvula proporcional

5.2.3.1.2 Sensor de Flujo Un sensor de flujo es un dispositivo que nos permite medir la cantidad de un liquido que pasa por una sección determinada, el sensor FS300A ver figura 5.12 cuenta con un sistema de paletas que hacen girar un rotor, que a su vez genera una señal de pulsos por vuelta, algo muy parecido al encoder rotativo ver figura 5.4, esto nos permite generar una frecuencia que es leída por el elemento de control, que realiza la conversión de esa frecuencia a un valor.



Figura 5.12: Sensor de flujo FS300A

Fuente: (Naylamp Mechatronics, 2018)

Artículo	Sensor de Flujo modelo FS300A
Voltaje de operación	5V-24V
Consumo de corriente	15mA a 5V DC
Rango de fluido	1-60 L/min
Salida	Onda cuadrada pulsante
Volumen promedio por pulso	3.03 mL
Pulsos por litro	330
Factor de conversión	5.5
Rosca externa	3/4" NPS
Presión de trabajo max:	1.2 MPa (12 bar)

Tabla 5.7: Características Sensor de flujo

5.3 Algoritmo Adaline

La salida de una red neuronal esta dado por la ecuación 5.1 que es la representación matemática de la figura 4.6.

$$y = \sum_{i=1}^N w_i \cdot x_i + \theta \quad (5.1)$$

para cada entrada x_i tiene asociado un peso w_i que hacen sintaxis para obtener la salida deseada y_d . El limite de las característica de decisión para la red adaline se presenta cuando $n=0$, por lo tanto:

$$w \cdot x + \theta = 0 \quad (5.2)$$

La ecuación 5.2 se asemeja a una ecuación de la recta en un plano donde una linea con una pendiente separa en dos regiones el plano, siendo esto una característica del perceptron y la red adaline, ya que pueden clasificar patrones linealmente separables.

Cuando se tiene una red con 2 entradas x_1 y x_2 a las cuales están asociadas los pesos w_1 y w_2 respectivamente, es decir:

$$y = \sum_{i=1}^2 w_i \cdot x_i + \theta \quad (5.3)$$

Se puede apreciar la linea que separa en dos regiones el espacio de entrada, como se muestra en la figura 5.13.

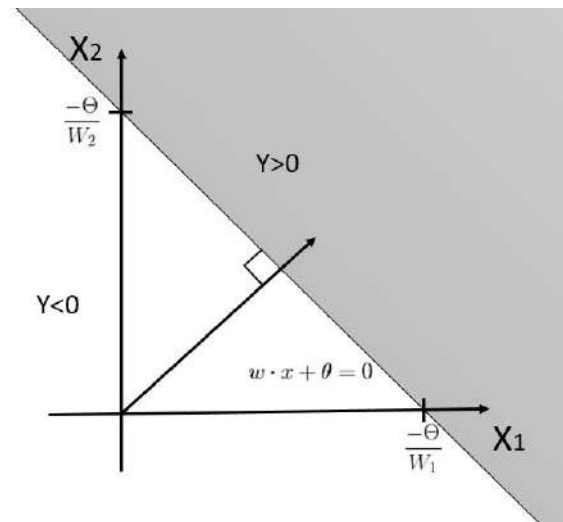


Figura 5.13: Característica de decisión de una red tipo Adaline

Fuente: Autor

Confirmando lo mencionado anteriormente, la red adaline puede clasificar correctamente patrones linealmente separables en dos categorías.

5.3.1 Regla de aprendizaje

La red adaline al igual que el perceptron es una red de aprendizaje supervisado que necesita conocer de antemano los valores asociados a cada entrada. siendo los conjuntos de entrada de la siguiente forma. (Acosta y cols., 2000)

$$\{x_1, y_1\}, \{x_2, y_2\}, \dots, \{x_n, y_n\} \quad (5.4)$$

Donde x_n son las entradas a la red y Y_n su salida asociada.

El algoritmo LMS o regla delta ver 4.2.2.1 se deriva de la regla Widrow-Hoff, de la que en términos generales para un proceso de actualización de los pesos de una red Adaline, se deduce de la siguiente manera:

$$w(k+1) = w(k) + \alpha \frac{e(k)x(k)}{|x(k)|^2} \quad (5.5)$$

En donde k representa la iteración actual del proceso de actualización, $w(k+1)$ es el siguiente valor que tomará el vector de pesos, es decir, la iteración próxima. $w(k)$ es el valor actual del array de pesos. $e(k)$ representa el error actual, que se define como la diferencia entre la respuesta deseada $Y_d(k)$ y la salida de la red $y(k) = w(k) \cdot x(k)$ antes de la actualización:

$$e(k) = Y(k) - w(k) \cdot x(k) \quad (5.6)$$

La variación del error en cada iteración es representado de la siguiente forma matemáticamente.

$$\Delta e(k) = \Delta (x(k) - w(k) \cdot x(k)) = -x(k) * w(k) \quad (5.7)$$

Teniendo en cuenta la ecuación 5.5 la actualización de pesos teniendo en consideración el error es:

$$\Delta w(k+1) = w(k) + \alpha \frac{e(k) \cdot x(k)}{|x(k)|^2} \quad (5.8)$$

Reemplazando 5.7 en 5.8, se obtiene:

$$\Delta e(k) = -\alpha \cdot \frac{e(k) \cdot x(k) \cdot x(k)}{|p(k)|^2} = -\alpha \cdot e(k) \quad (5.9)$$

De esta forma, el error es reducido por un factor α mientras los pesos van cambiando a medida que se presenta un valor de entrada. Cada vez que se presenta un nuevo patrón el ciclo de actualización inicia nuevamente; el siguiente error es reducido por un factor α , y el proceso continua. Los valores iniciales del vector de pesos usualmente escogidos como cero y se actualizan hasta que el algoritmo alcance la convergencia. (Acosta y cols., 2000)

La elección de α controla la estabilidad y velocidad de la convergencia del proceso de entrenamiento; si se escoge un valor muy pequeño de α , el algoritmo pierde velocidad y tarda mucho en alcanzar convergencia, si por el contrario se toma un valor muy grande, el algoritmo pierde estabilidad y se torna oscilante alrededor del valor de convergencia. (Acosta y cols., 2000) un rango de valores prácticos para la rata de aprendizaje es:

$$0.1 < \alpha < 1 \quad (5.10)$$

Este algoritmo es auto-normalizado en el sentido que la elección de α no depende de la magnitud de las señales de entrada; cada paso actualizado es colineal en los parámetros de entrada y su magnitud es inversamente proporcional a $|x(k)|^2$.

Una descripción geométrica del proceso de actualización de pesos en la regla Widrow-Hoff delta o algoritmo LMS, se describe en la siguiente figura 5.14. Conforme a la ecuación 5.8, $W(k+1)$ equivale a la suma $W(k)$ y $\Delta W(k)$ además $\Delta W(k)$ es paralelo con el vector de entrada $x(k)$. De la ecuación 5.7, el delta del error es igual al producto negativo de $x(k)$ y $\Delta W(k)$, como el algoritmo LMS selecciona a $\Delta W(k)$ de tal forma que sea colineal con $x(k)$, el cambio en el error deseado se calcula con la menor magnitud de $\Delta W(k)$ posible.

El algoritmo a la actualización de las ganancias es:

$$b(k+1) = b(k) + \alpha e(k) \quad (5.11)$$

El algoritmo LMS o delta corrige el error y si todos los patrones de entrada son de igual longitud, la actualización de pesos y ganancias tiende a minimizar el error medio cuadrático, esta es la principal propiedad de este algoritmo. (Acosta y cols., 2000)

Para explicar el cálculo del error medio cuadrático se considerará una red Adaline de una sola neurona y se empleará un algoritmo de pasos descendientes aproximado, como el que utilizaron Widrow y Hoff. La función de error es una función matemática definida en el espacio de pesos multidimensional para un conjunto de patrones dados, es una superficie

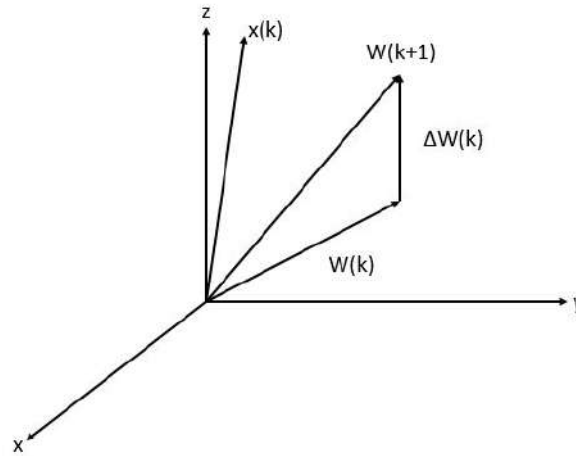


Figura 5.14: Actualización de pesos del algoritmo LMS

Fuente: Autor

que tiende a muchos mínimos (globales y locales) y la regla de aprendizaje va a buscar el punto en el espacio de pesos donde se encuentra el mínimo global de esa superficie; aunque la superficie de error es desconocida, el método de gradiente descendente consigue obtener información local de dicha superficie a través del gradiente, con esa información se decide que dirección tomar para llegar hasta el mínimo global de dicha superficie. (Acosta y cols., 2000) Para el cálculo del gradiente en cada iteración, teniendo en cuenta que la función para el error medio cuadrático es:

$$e^2(k) = (Y_d(k) - Y_s(k))^2 \quad (5.12)$$

Donde Y_d representa la salida deseada en la iteración k , Entonces, en cada iteración se tiene un gradiente del error de la siguiente forma:

$$[\nabla e^2(k)]_j = \frac{\partial e^2(k)}{\partial w_{i,j}} = 2e(k) \frac{\partial e(k)}{\partial w_{1,j}} \text{ para } j = 1, 2, \dots, \mathbb{R} \quad (5.13)$$

y

$$[\nabla e^2(k)]_{R+1} = \frac{\partial e^2(k)}{\partial b} = 2e(k) \frac{\partial e(k)}{\partial b} \quad (5.14)$$

Se evalúa la derivada parcial del error $e(k)$ con respecto a los pesos de las neuronas.

$$\frac{\partial e(k)}{\partial w_{i,j}} = \frac{\delta [Y_d(k) - (w * x(k))]}{\delta w_{i,j}} \quad (5.15)$$

$$= \frac{\partial \left[Y_d - \left[\sum_{i=1}^R w_{1,i} x_i + b \right] \right]}{\partial w_{i,j}} \quad (5.16)$$

La ecuación anterior se puede simplificar a:

$$\frac{\delta e(k)}{\delta w_{i,j}} = -x_j(k) \quad (5.17)$$

Y de forma parecida el gradiente corresponde a derivar parcialmente el error con respecto a la ganancia.

$$\frac{\delta e(k)}{\delta b} = -1 \quad (5.18)$$

Por lo cual solo se hace necesario multiplicar el error por el numero de entradas, para calcular el error medio cuadrático. Entonces, el algoritmo de actualización de pesos y ganancias del algoritmo LMS queda de la siguiente forma:

$$w(k+1) = w(k) + 2\alpha e(k)x(k) \quad (5.19)$$

$$b(k+1) = b(k) + 2\alpha e(k) \quad (5.20)$$

Para esta demostración el factor de aprendizaje α se toma como un valor constante. La figura 5.15 muestra la estructura de la red adaline donde x_i son las entradas, w_i son los pesos de sinapsis y y_{out} es la correspondiente salida de la red.

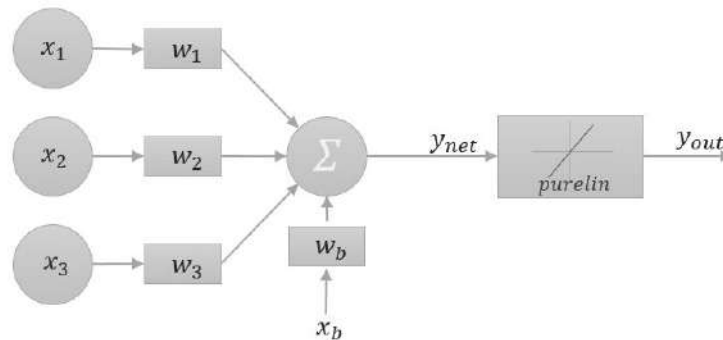


Figura 5.15: Estructura de la red neuronal adaline

Fuente: Autor

5.4 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 5.16 puede verse un esquema de un regulador controlado por un microcontrolador. (pardo Carlos, 2014)

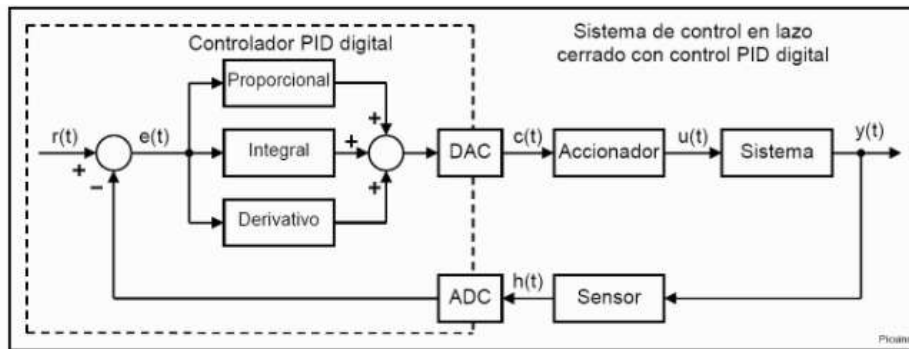


Figura 5.16: Sistema de control PID DIGITAL en lazo cerrado

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 son implementados en la mayor parte en el modelado y simulación de sistemas de control, pero la verdadera aplicación de estos algoritmos se presenta a la hora de llevarlos a un sistema embebido que trabajan con señales discretas, por tal razón, se tiende a usar estos tipos de control, aunque las señales de entrada correspondientes a la realimentación de los lazos de control en su mayoría son análogas ver figura 5.17, se pueden discretizar, valiéndonos del hecho de que resulta complicados trabajar con un tipo de señal análoga de estas en el sistema embebido, dicho de otra forma es pasar de la señal análoga a una señal discreta usando un tiempo de muestreo, que se selecciona según la dinámica del sistema.

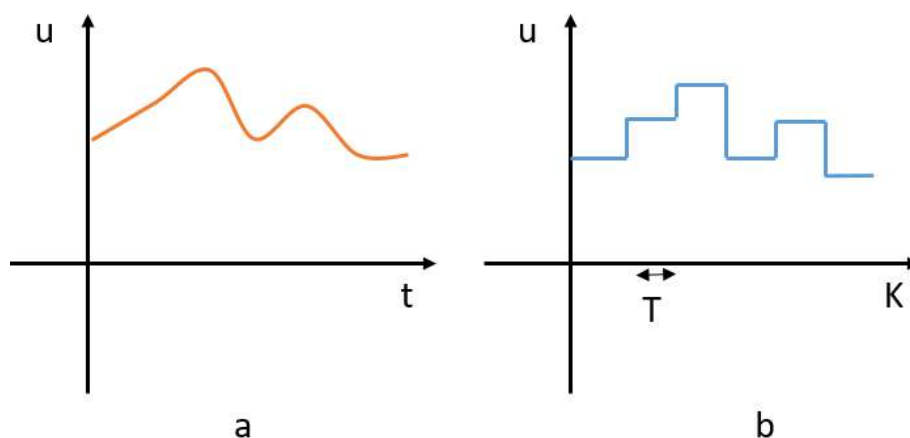


Figura 5.17: Figura a) señal analógica, Figura b) señal discreta

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.

$$\frac{u(t)}{e(t)} = \frac{k_p s + k_i + k_d s^2}{s} \quad (5.21)$$

Discretización utilizando la transformada z Los lazos de control continuo, estan formados de tal forma que los componentes del sistema, siempre tienen información 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, que sera el mismo para cualquier cantidad de muestreos en nuestro sistema, además de poseer la misma fase. Asesorado (2007), para hacer la conversión se hace uso de la siguiente ecuación: 5.22

$$s = \frac{1 - Z^{-1}}{T} \quad (5.22)$$

Ahora reemplazando s o la ecuación 5.22 en la ecuación 4.7 se obtiene:

$$\frac{U(z)}{E(z)} = \frac{K_p \left(\frac{1-z^{-1}}{T} \right) + K_i + K_d \left(\frac{1-z^{-1}}{T} \right)^2}{\left(\frac{1-z^{-1}}{T} \right)} \quad (5.23)$$

Simplificando la anterior ecuación da el siguiente resultado.

$$\frac{U(z)}{E(z)} = \frac{\left(K_p + K_i T + \frac{K_d}{T} \right) + z^{-1} \left(-2 \frac{K_d}{T} - K_p \right) + z^{-2} \left(\frac{K_d}{T} \right)}{(1 - z^{-1})} \quad (5.24)$$

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

$$z^{-1} \left\{ \frac{U(z)}{E(z)} = \frac{\left(K_p + K_i T + \frac{K_d}{T} \right) + z^{-1} \left(-2 \frac{K_d}{T} - K_p \right) + z^{-2} \left(\frac{K_d}{T} \right)}{(1 - z^{-1})} \right\} \quad (5.25)$$

Resolviendo se obtiene lo siguiente:

$$z^{-1} \left\{ U(z) (1 - z^{-1}) = E(z) \left(\left(K_p + K_i T + \frac{K_d}{T} \right) + z^{-1} \left(-2 \frac{K_d}{T} - K_p \right) + z^{-2} \left(\frac{K_d}{T} \right) \right) \right\} \quad (5.26)$$

$$z^{-1} \left\{ U(z) - U(z)z^{-1} = E(z) \left(K_p + K_i + \frac{K_d}{T} \right) + E(z)z^{-1} \left(-2 \frac{K_d}{T} - K_p \right) + E(z)z^{-2} \left(\frac{K_d}{T} \right) \right\} \quad (5.27)$$

De este modo ya se puede aplicar la transformada de la z inversa

$$u[k] - u[k-1] = e[k] \left(K_p + K_i T + \frac{K_d}{T} \right) + e[k-1] \left(-2 \frac{K_d}{T} - K_p \right) + e[k-2] \left(\frac{K_d}{T} \right) \quad (5.28)$$

Como salida final, para aplicarlo en el sistema embebido el resultado final es:

$$u[k] = e[k] \left(K_p + K_i T + \frac{K_d}{T} \right) + e[k-1] \left(-2 \frac{K_d}{T} - K_p \right) + e[k-2] \left(\frac{K_d}{T} \right) + u[k-1] \quad (5.29)$$

Se simplifica la ecuación de tal manera que se hace igualdades en los siguientes términos:

$$A_1 = \left(K_p + K_i T + \frac{K_d}{T} \right) \quad (5.30)$$

$$A_2 = \left(-2 \frac{K_d}{T} - K_p \right) \quad (5.31)$$

$$A_3 = \left(\frac{K_d}{T} \right) \quad (5.32)$$

Resultado de la salida del controlador es:

$$u[k] = e[k] A_1 + e[k-1] A_2 + e[k-2] A_3 + u[k-1] \quad (5.33)$$

La figura 5.18 representa graficamente como seria la estructura de un lazo de control PID discreto.

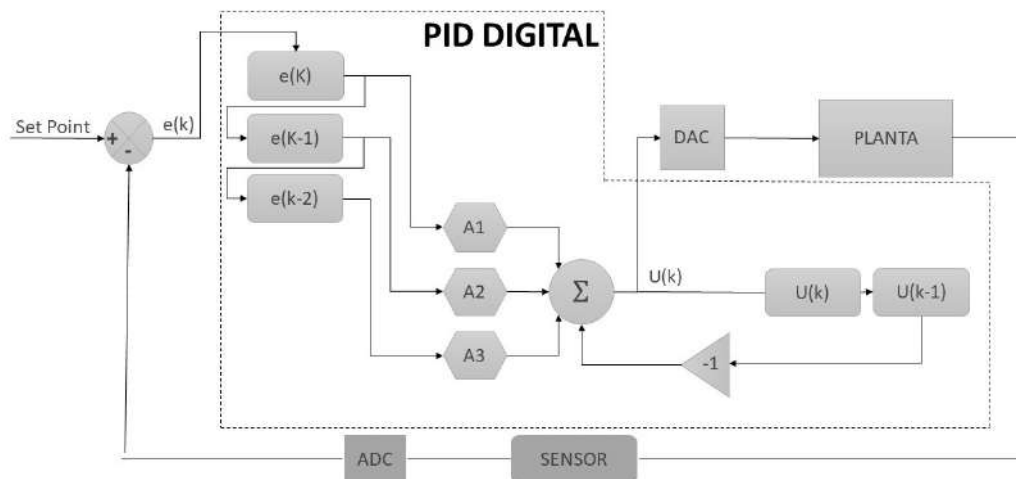


Figura 5.18: Lazo de control PID Discreto

Fuente: Autor

5.5 Comparando la red adaline con la estructura del control PID discreto

La ecuación 5.33 es muy similar a la ecuación de la salida de una red adaline 5.1, por lo tanto se podría deducir que una red adaline puede funcionar como un controlador, la figura 5.18 donde nos fijamos especialmente en la parte encerrada por la línea punteada y la figura 5.15 también confirman la similitud entre la estructura base, donde las constantes A_1 es la reciproca de w_1 en la red neuronal y el error $e(k)$ es como la entrada x_1 en la red adaline, además ambas se operan de la misma forma, la ecuación 5.34 se plantean las dos ecuaciones, en la parte izquierda la ecuación de la salida del PID discreto y en la parte derecha la salida de la neurona adaline.

$$u[k] = e[k] A_1 + e[k-1] A_2 + e[k-2] A_3 + u[k-1] \Leftrightarrow y_{out} = x_1 w_1 + x_2 w_2 + x_3 w_3 + \Theta \quad (5.34)$$

En este principio nos basamos para validar el postulado de que una red adaline puede funcionar como un controlador, partiendo de la similitud de estas dos estructuras.

5.6 Algoritmo Backpropagation para redes MLP

Este algoritmo lo publicó Rumelhart en 1986. Aquí se expone el algoritmo básico. En la figura 5.19 se observa un MLP de dos capas ocultas que nos va a servir de ejemplo para la deducción del algoritmo. (Sanchez medina, 1998)

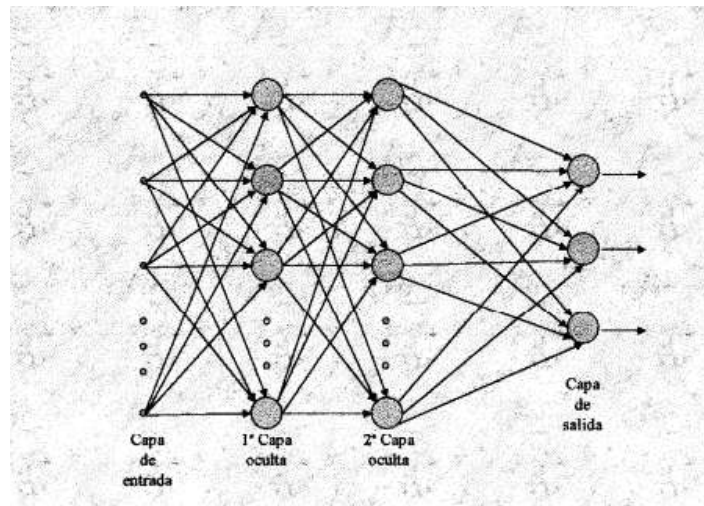


Figura 5.19: Topología de Redes Neuronales Multi-capas

Fuente: (Sanchez medina, 1998)

En cualquier tipo de NN se necesita algún tipo de ponderación o constante que asocie a cada conexión. El ajuste de estas variables que hacen la veces de sinapsis son en las que se trabaja, es decir, sobre los cuales se va a efectuar el entrenamiento. En el caso de que estos elementos 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.

Tal como pasa en una Red neuronal tipo adaline, la red multi-capas tiene un parecido similar en la salida de la red. La salida de la MLP, esta dada por la sumatoria de la multiplicación de las entradas por los pesos sinápticos, se dispone de un conjunto 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 , en la siguiente ecuación. (Sanchez medina, 1998)

$$e_j(k) = d_j(k) - y_j(k) \quad (5.35)$$

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

$$\xi(k) = \frac{1}{2} \sum_{j \in N_L} e_j^2(k) \quad (5.36)$$

donde el conjunto de N_L incluye a todas las neuronas en la capa de salida de la red. el error cuadrático medio se obtiene sumando $\varepsilon(k)$ para todos los pares de entrenamiento y normalizado, dividiendo por K tal y como se muestra en la ecuación 5.37. (Sanchez medina, 1998)

$$E = \frac{1}{K} \sum_{j \in N_L} \varepsilon(k) \quad (5.37)$$

la corrección de los pesos sinápticos se realiza mediante un incremento en estos como se indica en la ecuación 5.38. (Sanchez medina, 1998)

$$w'_{L,i,j} = w_{L,i,j} + \Delta w_{L,i,j} \quad (5.38)$$

El algoritmo de backpropagation aplica una corrección $\Delta w_{L,i,j}(k)$ a los pesos sinápticos $w_{L,i,j}(k)$, 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 5.39. (Sanchez medina, 1998)

$$\frac{\partial \xi(k)}{\partial w_{L,i,j}(k)} = \frac{\partial \xi(k)}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} \frac{\partial v_j(k)}{\partial w_{L,i,j}(k)} \quad (5.39)$$

el gradiente $\partial \xi(k)/\partial w_{L,i,j}(k)$ representa la dirección de búsqueda 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 5.36, obteniendo:

$$\frac{\partial \xi(k)}{\partial e(k)} = e_j(k) \quad (5.40)$$

derivando en ambos lados la siguiente ecuación se tiene

$$y_i = f(v_i) \quad (5.41)$$

$$\frac{\partial y_i(k)}{\partial v_i(k)} = f'_j(v_j(k)) \quad (5.42)$$

donde la prima significa derivada respecto de su argumento. Finalmente derivando 5.41 a con respecto a $w_{i,j}$, se tiene: (Sanchez medina, 1998)

$$\frac{\partial v_i(k)}{\partial w_{i,j}(k)} = y_i(k) \quad (5.43)$$

se sustituye la ecuación 5.42 a la ecuación 5.43 obteniendo como resultado:

$$\frac{\partial \xi(k)}{\partial w_{i,j}(k)} = -e_i(k) f'_j(v_j(k)) y_i(k) \quad (5.44)$$

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

$$\Delta w_{l,i,j}(k) = -\eta \frac{\partial \xi(k)}{\partial w_{l,i,j}(k)} \quad (5.45)$$

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 5.45 y 5.46

$$\Delta w_{l,i,j}(k) = -\eta \delta_j(k) y_i(k) \quad (5.46)$$

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

$$\delta_j(k) = -\frac{\partial \xi(k)}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial y_j(k)} \frac{\partial y_j(k)}{\partial v_j(k)} \quad (5.47)$$

$$\delta_j(k) = -e_j(k) f'_j(v_j(k)) \quad (5.48)$$

El gradiente indica los cambios requeridos en los pesos sinápticos. De la ecuación 5.48 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 5.46 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 5.48 (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 las 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 5.48 se puede redefinir el gradiente local $\delta_j(k)$ para la neurona oculta j como: Sanchez medina (1998)

$$\delta_j(k) = -\frac{\partial \xi(k)}{\partial y_j(k)} f'_j(v_j(k)) \quad (5.49)$$

Se toma la ecuación 5.36 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)

$$\xi(k) = \frac{1}{2} \sum_{m \in N_L} e_m^2(k) \quad (5.50)$$

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

$$\frac{\partial \xi(k)}{\partial y_i(k)} = \sum_m e_m \frac{\partial e_m(k)}{\partial y_i(k)} \quad (5.51)$$

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

$$\frac{\partial \xi(k)}{\partial y_j(k)} = \sum_m e_m \frac{\partial e_m(k)}{\partial v_m(k)} \frac{\partial v_m(k)}{\partial y_j(k)} \quad (5.52)$$

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

$$\frac{\partial e_m(k)}{\partial v_m(k)} = -f'_m(v_m(k)) \quad (5.53)$$

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

$$v_m(k) = \sum_{j=0}^q w_{m,j}(k) y_j(k) \quad (5.54)$$

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 5.54 con respecto a $y_i(k)$ conduce a (Sanchez medina, 1998):

$$\frac{\partial v_m(k)}{\partial y_j(k)} = w_{m,j}(k) \quad (5.55)$$

a partir de esto, se usa la ecuacion 5.53,5.55 y 5.52 da como resultado:

$$\frac{\partial \xi_m(k)}{\partial y_j(k)} = - \sum_m \delta_m(k) * w_{m,j}(k) \quad (5.56)$$

donde en la segunda línea, se ha usado la definición del gradiente local $\delta_m(k)$ dada en la ecuación 5.48 con el índice m sustituido por j . Finalmente, usando ja ecuación 5.56 en 5.46, se obtiene el gradiente local $\delta_m(k)$ para una neurona oculta j , tras la reorganización de términos como sigue (Sanchez medina, 1998):

$$\delta_j(k) = f'_j(v_j(k)) \sum_m \delta_m(k) w_{m,j}(k) \quad (5.57)$$

siendo la neurona j oculta. El factor $f'_j(v_j(k))$ envuelto en la computación del gradiente local δ_j ecuación 5.57 solamente de la función de activación asociada a la neurona oculta j . El otro actor, la 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.

5.6.1 Algoritmo backpropagation

Para el ejemplo se denota que se uso la función de activación sigmoide, por la derivada que aparece en los cálculos da como resultado, este algoritmo fue tomado de (Sanchez medina, 1998)

$$f'(x) = f(x)(1 - f(x)) \quad (5.58)$$

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**

$$y_{capa,nodo} = f\left(\sum_{i=1}^{N_{capa-1}} W_{capa,nodo,i} \times y_{capa-1,i}\right);$$

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 5.8:** Algoritmo backpropagation

5.7 PLC S7-1200

El dispositivo en el que se va a realizar el control de cada uno de los procesos es el Controlador Lógico Programable S7-1200 de la marca Siemens, para ser más exactos la CPU Compact 1214C DC/DC/DC 6ES7214-1AG40-0XB0 ver figura 5.20, este autómata es hoy en día uno de los más usados en la industria, cuenta con una gran cantidad de funciones lo que lo hacen bastante flexible, además cuenta con una gran de módulos y tarjetas que se le pueden acoplar o conectar en paralelo, también dispone con una gran variedad de protocolos de comunicación industrial. Uno de las tarjetas que se le acopló a la CPU utilizada en este proyecto es la Signal



Figura 5.20: PLC S7 1200 CPU Compact 1214C DC/DC/DC 6ES7214-1AG40-0XB0

Fuente: (DURTRON, 2018)

Board 6ES7232-4HA30-0XB0 ver figura 5.21 producida por la misma empresa, el autómata cuenta con un compartimiento en la parte frontal donde se incorpora este tipo de tarjetas, en especial está nos permite anexar una salida analógica al PLC que nos provee la capacidad de generar una señal de 0-10V o de 0-24mA dependiendo de la configuración que se le disponga en el software. Los criterios para la selección de este dispositivo fueron principalmente



Figura 5.21: Signal Board 6ES7232-4HA30-0XB0 A0 0-10V o 0-20mA

Fuente: (PLC-City, 2018)

su disponibilidad, además de que en su software de programación Tia Portal se cuenta con el lenguaje de alto nivel SCL que para este proyecto es el lenguaje requerido, en la próxima sección se profundiza más acerca de este lenguaje y sus ventajas. Además para realizar una asertiva interacción con el controlador, se cuenta con una HMI KTP700 Basic PN ver figura 5.22 también de la marca Siemens, que se conecta con el PLC por medio del protocolo PRO-FINET, el software Tia Portal tiene incluido la función de programación de estas pantallas, está pantalla permite interactuar de manera casi instantánea con los parámetros internos del PLC (Programación).



Figura 5.22: HMI KTP700 Basic PN

Fuente: Autor

Dispositivo	SIMATIC S7-1200, CPU 1214C, CPU COMPACTA, DC/DC/DC 6ES7 214-1AG40-0XB0
Voltaje de alimentación	24V DC
Entadas Digitales	14; integradas
Entradas usables para funciones tecnológicas	6 HSC (High Speed Counting)
Salidas Digitales	10; Transistorizadas
Salidas rápidas	4; Salida de tren de impulsos 100 KHz
Entradas analógicas	2 de 0 a 10V
Corriente nominal de consumo	500mA
Alimentación de sensores	24V DC
Memoria de trabajo	100Kbyte
Tiempos de ejecución de la CPU	
para operaciones a bits, típ.	0.085us;/instrucción
para operaciones a palabras, típ.	1.7us;/instrucción
para aritmética de coma flotante, típ.	2,3us;/instrucción
CPU-Bloques: N° de bloques(total)	DBs, FCs, FBs, contadores y temporizadores. El número máximo de bloques direccionables es de 1 a 65535. No hay ninguna restricción uso de toda la memoria de trabajo.
Interfaz	PROFINET
Norma Física	Ethernet
Programación	KOP, FUP y SCL

Tabla 5.9: Características generales PLC S7-100 CPU 1214C 6ES7 214-1AG40-0XB0

Fuente: (Siemens AG, 2015)

Las características mencionadas en la tabla 5.9, no son todas las características del dispositivo si no las más relevantes para el autor.

5.8 Lenguaje de programación SCL

El lenguaje SCL o Lenguaje de control estructurado es un lenguaje de alto nivel en la programación de PLC's, en la sección 4.3.1.2 se ha mencionado sus principales características y ventajas, en las cuales se destaca que es un lenguaje basado en PASCAL, permite cálculos matemáticos complejos, además su estructura al igual que el ST ver sección 4.3.1.1 es muy parecida a otros lenguajes de programación como C o C++, SCL es el equivalente al ST pero en autómatas Siemens, la diferencia entre cada uno de los dos es muy reducida.

La selección de este lenguaje para este proyecto se fundamenta en que el algoritmo de control neuronal directo, lleva bastantes cálculos matemáticos que al llevarlo a un lenguaje como LADDER se complica bastante, con SCL o ST se puede programar el algoritmo de manera muy sencilla en un bloque de función, cabe recalcar que SCL no es un lenguaje muy usado en

la actualidad, debido a su grado de complejidad la mayoría de técnicos o personal encargado de la programación no profundizan en lenguajes de programación como estos, SCL es un lenguaje relativamente viejo, sin embargo, debido a su capacidad para resolver problemas ha comenzado a ser más usado en la actualidad, ya que la industria se enfrenta a nuevos retos mucho más complejos que hace 20 años.

Programar en SCL ofrece la ventaja de generar bloques de programación bastante concisos, donde se puede organizar y documentar cada línea de código, incluso algunas funciones incluidas por el mismo software de programación tienen código fuente estructurado en SCL, por lo que estamos hablando de un lenguaje raíz del software.

5.8.1 Descripción del lenguaje

Como se describía anteriormente el SCL es semejante al ST que a su vez tiene un parentesco con las estructuras de otros lenguajes de programación como C++, en este apartado se va a enseñar un poco acerca de su estructura. Como bien sabemos todo lenguaje de programación tiene una sintaxis única, SCL como ST se asemejan en mucho, por lo tanto al explicar lo que es el lenguaje SCL se podría también aplicar en ST, pero teniendo en cuenta que siguen siendo diferentes en algunos puntos. El lenguaje SCL es un lenguaje muy flexible, fácilmente puede ser replicado en otros programas por su ser un lenguaje de carácter textual. En SCL se pueden encontrar todos los tipos de datos que trae por defecto la mayoría de lenguajes de programación de acuerdo a la norma IEC 61131-3 como son los tipos de datos BOOL, BYTE, WORD, INT, REAL, ARRAY entre otros. SCL cuenta con todas las funciones disponibles en los demás lenguajes de programación, además contiene estructuras de selección con IF o CASE y estructuras de bucles como FOR o WHILE.

En el siguiente apartado se muestra parte de su sintaxis, los ejemplos están enfocados para las personas con conocimientos básicos en lenguajes de programación como C, C++, o PASCAL. Para la declaración de variables en SCL y ST se realiza de la siguiente manera ver 5.1:

Código 5.1: Declaración de variables en SCL

```
1  VAR_INPUT //Declaración de variables de entrada
2      myVAR1 : Bool; // Variable myVAR1 como entrada booleana.
3  END_VAR //Cierre de declaración de variables de entrada
4  VAR //Declaración de variables estáticas
5      myVAR2 : Int; // Variable myVAR2 tipo de dato entero.
6  END_VAR //Cierre de declaración de variables estáticas.
7  VAR_OUTPUT //Declaración de variables de salida
8      myVAR3 : Real; //Variable myVAR3 tipo de dato real.
9  END_VAR //Cierre de declaración de variables de salida.
```

En el siguiente fragmento de código 5.2 se muestra como se le asigna un valor a una variable ya sea un número constante o el valor de otra variable.

Código 5.2: Asignación Variable SCL

```
1  myVAR1:=TRUE; // Asignación del valor Verdadero a la variable booleana myVAR1.
2  myVAR2:=10; // Asignación del valor 10 a la variable entera myVAR2.
3  myVAR3:=1.23; // Asignación del valor 1.23 a la variable real myVAR3.
```

```
4 myVAR5:=myVAR4; // Asignación del valor de myVAR4 a myVAR5 las dos variables ←
   ↪ deben ser del mismo tipo de datos.
```

SCL y ST ofrece una facilidad para los cálculos matemáticos, en otros lenguajes como LADDER realizar este tipo de operaciones puede ocupar bastante líneas de código, mientras que en SCL se realiza de manera sencilla, en el siguiente fragmento se muestra una operación básica ver 5.3:

Código 5.3: operaciones matemáticas en SCL

```
1 myVAR_INT:= 25 + myVAR1_INT - myVAR2_INT ; //Operación de suma y resta.
2 myVAR3_INT:= (2 * myVAR1_INT) / (myVAR2_INT); // Operacion de multiplicación y ←
   ↪ división.
3 myVAR4_INT:= myVAR4_INT**2; // Operación exponencial al cuadrado.
```

También se puede realizar operaciones lógicas en SCL y ST ver 5.4

Código 5.4: operaciones lógicas en SCL

```
1 myVAR_BOOL:=NOT(myVAR1_BOOL); // Operación lógica NOT de negación.
2 myVAR_BOOL2:= (myVAR1_BOOL AND myVAR3_BOOL) OR myVAR4_BOOL; //Operaciones ló←
   ↪ gicas AND y OR.
```

Estructuras condicionales como IF y CASE también se encuentran en SCL y su estructura se muestra en 5.5 .

Código 5.5: Estructuras condicionales en SCL

```
1 IF myVAR4_BOOL = myVAR5_BOOL THEN // Estructura condicional IF-ELSE
2   myVAR6_BOOL:=TRUE;
3 ELSE //Estructura ELSE
4   myVAR6_BOOL:=FALSE;
5 END_IF; //Fin de la estructura condicional IF-ELSE
6 CASE myVAR5_INT OF //Estructura condicional CASE
7   0:
8     myVAR6_INT:=100;
9     1..10:
10    myVAR6_INT:=50;
11    11,23,50:
12    myVAR6_INT:=10;
13 ELSE
14   myVAR6_INT:=0;
15 END_CASE; //Fin de la estructura condicional CASE
```

Los bucles o estructuras de estado de iteración como el FOR o WHILE tienen la siguiente estructura 5.6.

Código 5.6: Estructuras de estado de iteración en SCL

```
1 FOR i:=0 TO 10 BY 2 DO //Estructura FOR, iteración desde 0 a 10 en pasos de 2, ←
   ↪ cuando no se coloca el "BY 2" por defecto da pasos de 1 unidad.
2   myVAR_ARRAY_REAL[i]:=0.0;
3 END_FOR; //Cierre de la estructura FOR
```

```

4 WHILE myVAR7_BOOL=FALSE DO //Estructura WHILE, mientras myVAR7_BOOL sea falso, ←
    ↪ el bucle se ejecutara.
5     //código a ejecutar
6 END_WHILE; // Cierre de la estructura WHILE.

```

5.8.2 Ejemplos en SCL

En el siguiente ejemplo 5.7 se muestra una función para el calculo promedio.

Código 5.7: Calculo del valor promedio

```

1 VAR CONSTANT //Declaraciones de variables constantes
2     ArrayMin:INT:=0; //Valor mínimo del tamaño del Array
3     ArrayMax:INT:=9; //Valor máximo del tamaño del Array
4 END_VAR
5 VAR //Declaración variables estáticas
6     i:INT; //Variable entera para el contador del FOR
7     myArray:ARRAY[ArrayMin..ArrayMax] OF REAL; //Array donde se encuentra los ←
    ↪ valores.
8     Suma_Array:REAL; // Variable que almacena la suma de los valores del array
9     Promedio:REAL; //El resultado de la operación se guarda en esta variable
10 END_VAR
11 Suma_Array:=0.0;
12 FOR i:=ArrayMin TO ArrayMax DO //For para la suma de los valores del array
13     Suma_Array:= Suma_Array + myArray[i];
14 END_FOR;
15
16 Promedio:= (Suma_Array) / (ArrayMax - ArrayMin + 1); //Calculo del promedio.

```

En el ejemplo 5.8 se realiza el calculo del volumen del tanque correspondiente a un cilindro con la parte inferior en forma de semi-esfera. Este ejemplo se saco del libro (Antonsen, 2019)

Código 5.8: Calculo del volumen en un tanque

```

1 VAR_INPUT
2     Diametro_tanque:REAL; //Diametro del tanque
3     Altura_tanque:REAL; //Altura del tanque
4     Medida_de_nivel:REAL; // Nivel tomado desde la parte inferior.
5 END_VAR
6 VAR_OUTPUT
7     Volumen_Tanque:REAL;
8 END_VAR
9 VAR CONSTANT
10     PI:REAL:=3.1415; //Valor de PI
11 END_VAR
12 VAR
13     Nivel:REAL; //Variable para el calculo interno
14     Vol:REAL; //Variable para el calculo interno
15     Lr:REAL; // Nivel radial en el circulo
16     Radio_Tanque:REAL;
17 END_VAR;

```

```

18
19 Level := Medida_de_nivel;
20 Radio_Tanque := (Diametro_tanque / 2) ;
21 IF Nivel<0 THEN //Chequeo del nivel bajo- el nivel no puede ser negativo
22     Nivel :=0;
23 END_IF;
24 IF Nivel> (Radio_Tanque + Altura_tanque) THEN //Chequeo del nivel en alto, el ←
    ↪ tanque no puede sobrelleñarse
25     Nivel := Radio_Tanque + Altura_tanque;
26 END_IF;
27 // Calculo del nivel
28 IF Nivel <= Radio_Tanque THEN
29     Lr:=SQRT(Nivel*(Diametro_tanque-Nivel));
30     Vol:=(PI/6)*Nivel*(3*(Lr**2)+(Nivel**2));
31 ELSE
32     Vol:=2/3*PI*Radio_Tanque**3;
33 END_IF;
34
35 IF Nivel>Radio_Tanque THEN
36     Vol:=Vol + (Nivel - Radio_Tanque)*PI*(Radio_Tanque**2);
37 END_IF;
38 // Se traspasa el valor de Vol a la variable de salida Volumen_Tanque.
39 Volumen_Tanque:=Vol;

```

5.8.3 Comparación del lenguaje con LADDER

En este apartado se hace una comparación entre el lenguaje de LADDER más usadas con su equivalente en SCL, con el fin de que se entienda un poco mejor la lógica de SCL. En la figura 5.23 se muestra un fragmento de código en LADDER y en 5.9 se muestra el equivalente en SCL.

Ejemplo 1



Figura 5.23: Ejemplo 1 en LADDER

Fuente: Autor

Código 5.9: Ejemplo 1 en SCL

```
1 myVAR_BOOL2:=myVAR_BOOL;
```

Ejemplo 2

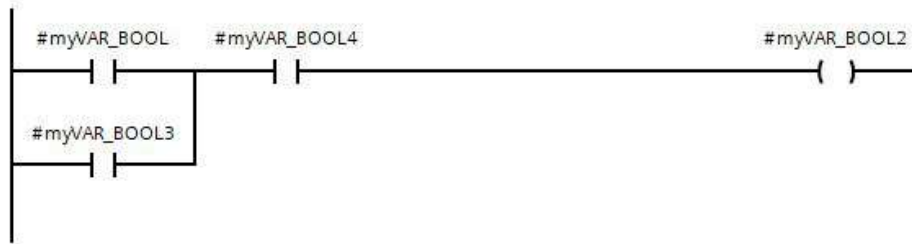


Figura 5.24: Ejemplo 2 en LADDER

Fuente: Autor

Código 5.10: Ejemplo 2 en SCL

```
1 myVAR_BOOL2 := (myVAR_BOOL OR myVAR_BOOL3) AND myVAR_BOOL4;
```

Ejemplo 3

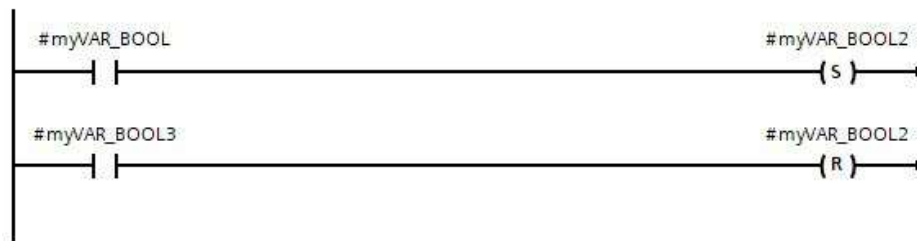


Figura 5.25: Ejemplo 3 en LADDER

Fuente: Autor

Código 5.11: Ejemplo 3 en SCL

```
1 IF myVAR_BOOL THEN
2     myVAR_BOOL2 := TRUE;
3 END_IF;
4 IF myVAR_BOOL3 THEN
5     myVAR_BOOL2 := FALSE;
6 END_IF;
```

Ejemplo4

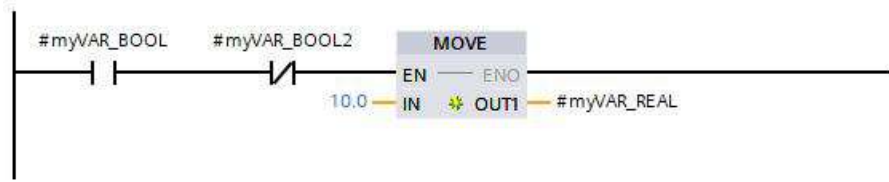


Figura 5.26: Ejemplo 4 en LADDER

Fuente: Autor

Código 5.12: Ejemplo 4 en SCL

```

1 IF myVAR_BOOL AND NOT(myVAR_BOOL2) THEN
2   myVAR_REAL:=10.0;
3 END_IF;

```

Ejemplo 5

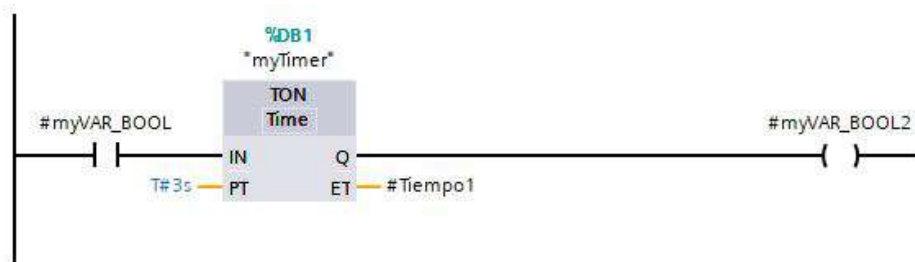


Figura 5.27: Ejemplo 5 en LADDER

Fuente: Autor

Código 5.13: Ejemplo 5 en SCL

```

1 "myTimer".TON(IN:=myVAR_BOOL,
2   PT:=T#3s,
3   Q=>myVAR_BOOL2,
4   ET=>Tiempo1);

```

Ejemplo 6

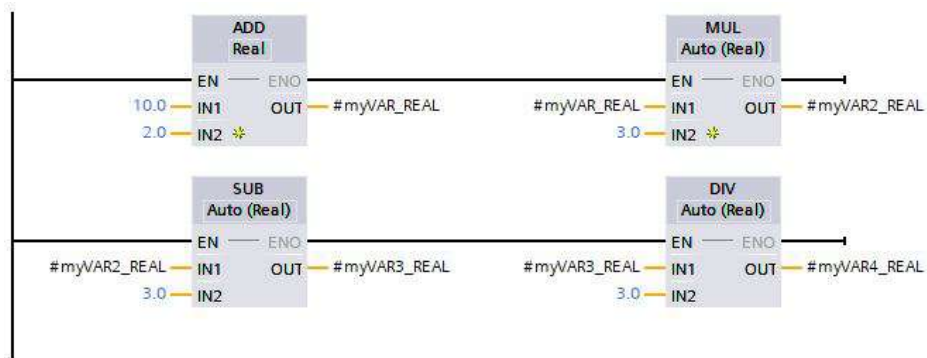


Figura 5.28: Ejemplo 6 en LADDER

Fuente: Autor

Código 5.14: Ejemplo 6 en SCL

```

1 myVAR_REAL := 10.0 + 2.0;
2 myVAR2_REAL := myVAR_REAL * 3.0;
3 myVAR3_REAL := myVAR2_REAL - 3.0;
4 myVAR4_REAL := myVAR3_REAL / 3.0;

```


6 Desarrollo

6.1 Acondicionamiento e identificación de las plantas

En el proceso de acondicionamiento de las plantas, se realizó la instalación y adecuación de cada uno de los elementos que conforman el lazo de control de cada una de las plantas escogidas.

6.1.1 Planta de Velocidad

6.1.1.1 Acondicionamiento

Para la planta de velocidad se realizó un diseño ver figura 6.1 para ajustar el eje del encoder directamente al eje del motor, además un sistema que permite soportar el encoder a una superficie sólida para que este se mantenga estático al momento de accionar el motor.



Figura 6.1: Diseños del acople y soporte

Fuente: Autor

En la siguiente figura 6.2 se puede evidenciar los diseños impresos en 3D acoplados al motor y el encoder. En la lectura del sensor de velocidad en este caso el encoder se realizó un bloque de función ver 6.3 estructurado en lenguaje SCL que se llevó a una librería en TIA PORTAL, la lectura de velocidad debe ser estable y precisa, con el bloque generado se obtiene una lectura muy exacta y estable que es un requisito para realizar una buena realimentación al controlador, permitiendo que el controlador se desempeñe de una forma óptima.

En la lectura del encoder se usan contadores rápidos que por defecto la CPU1214c cuenta, estos contadores nos permiten contar los pulsos generados por las dos fases (A y B) del encoder, en este caso se usó ese contador rápido HSC1 para poder contar el periodo en cada fase, con



Figura 6.2: Acople y soporte del encoder impresos en 3D

Fuente: Autor

estos datos se realizan operaciones matemáticas para establecer la velocidad del motor en RPM. En la figura se muestra el bloque de función generado en TIA PORTAL. Para verificar



Figura 6.3: Bloque de función para lectura de Velocidad con el encoder

Fuente: Autor

la lectura de la velocidad dada por el bloque creado era la correcta, se usó un tacómetro en este caso el UT-372 de la marca UNIT, este permite medir revoluciones por minutos en un rango de 10-99999 rpm, en la figura 6.4 se ilustra la medida real medida por el tacómetro (derecha) y la capturada por el encoder (izquierda).



Figura 6.4: Comprobación de la lectura de Velocidad

Fuente: Autor

La lectura obtenida por medio del tacómetro es de 1074,7 rpm mientras que la leída por medio del encoder es de 1076,4 lo cual nos da un error de lectura de aproximadamente 2 rpm lo cual es relativamente bajo, teniendo en cuenta que la velocidad máxima del motor es de 1850 rpm es decir, equivale a un error alrededor del 1 % lo que lo hace tolerable.

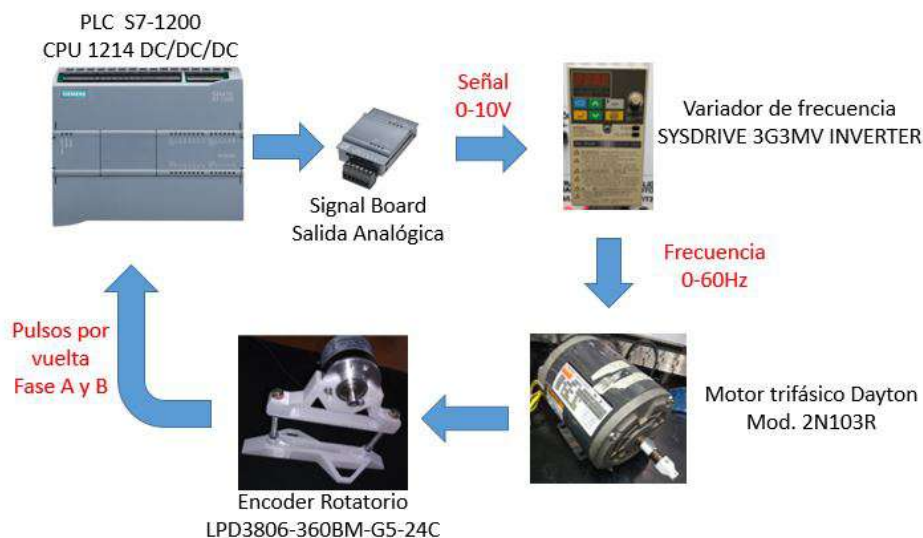


Figura 6.5: Diagrama ampliado funcionamiento de la planta de velocidad

Fuente: Autor

De acuerdo a la figura 5.1 en donde se visualiza la planta en diagramas de bloques, se expandirá en esta sección más sobre su funcionamiento, la señal entregada por el PLC que es el controlador es de 0-10V entregada a través de la signal board que a su vez se encuentra acoplada al autómatas, esta señal es recibida por el variador de frecuencia que la interpreta como una frecuencia de 0 Hz a 60 Hz, según la frecuencia suministrada por el variador el motor gira a una velocidad determinada, siendo 60 Hz la velocidad máxima del motor trifásico, en el

eje del motor se encuentra acoplado el encoder que registra 360 pulsos por vuelta, permitiendo calcular la velocidad del motor para realimentar el controlador y cerrar el lazo de control, para entender mejor el funcionamiento la figura 6.5 muestra la interacción de cada uno de los componentes.



Figura 6.6: Planta de Velocidad

Fuente: Autor

En la figura 6.6 se puede evidenciar los elementos reales interconectados de tal manera que permiten recrear el lazo de control cerrado para realizar las respectivas pruebas.

6.1.1.2 Identificación

Para identificar el comportamiento de la planta, para deducir si es lineal o no, se realizó una toma de datos en donde se captura la señal de salida del controlador y la velocidad obtenida, siendo más específicos la señal de 0-10V y la velocidad en rpm capturada por el encoder, el resultado se puede observar en la siguiente figura 6.7.

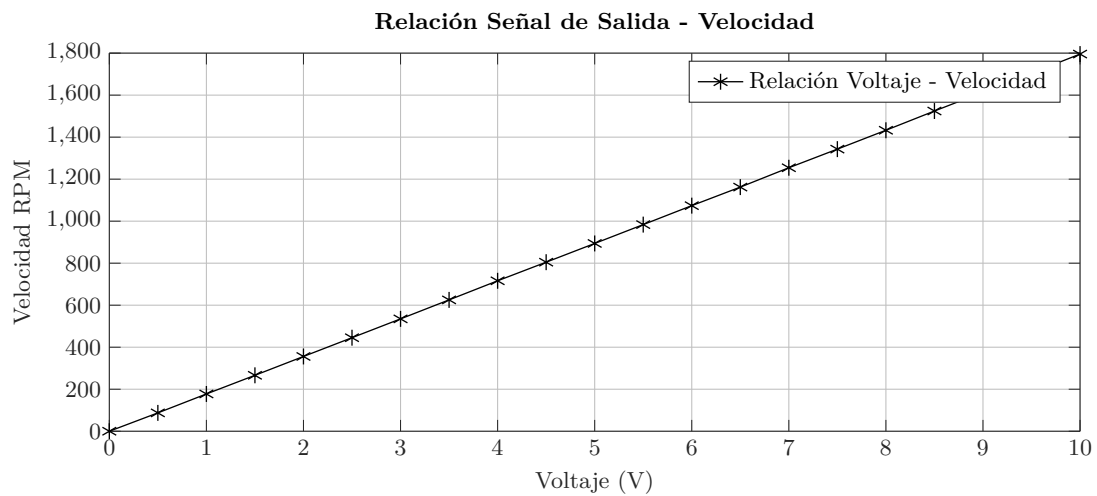


Figura 6.7: Identificación de la planta de Velocidad

Fuente: Autor

La gráfica obtenida nos indica que la planta es netamente lineal.

6.1.2 Planta de Temperatura

6.1.2.1 Acondicionamiento

En la planta de temperatura se realizó el acondicionamiento de la señal proveniente del sensor de temperatura en este caso la PT100, como es un sensor de 3 hilos se le adaptó un transmisor que cumple la función de un puente de Wheatstone para calcular la diferencia de voltaje, este transmisor ver figura 6.8 entrega una señal de 4-20mA en el rango de temperatura de 0 a 200°C.

El PLC S7-1200 CPU 1214C solo trae lectura analógica de un rango de 0-10V, por tal razón a la entrada analógica a usar se colocó en paralelo una resistencia de 500 Ohmios para convertir la señal de corriente 4-20 mA en una señal de voltaje de 2-10 V. En la figura 5.6 se denota el



Figura 6.8: Transmisor RTD par sensor de temperatura PT100

Fuente: (HUIMAIKEJI, 2018)

lazo de control cerrado de la planta de temperatura, para profundizar un poco más sobre su funcionamiento la figura 6.9 muestra la interacción de todos los componentes de la planta, la señal de control entregada por el PLC por medio de la signal board es una señal en este caso de 0-20mA pero por programación esta señal se puede configurar para que trabaje en el rango de operación estándar de 4-20 mA que además es el rango de entrada de señal del relé de estado sólido o SSR por sus siglas en ingles, este dispositivo es el encargado de realizar la conmutación de la potencia es decir la resistencia calefactora, el SSR SRPH1-A220 realiza la modulación de fase de la señal AC lo que permite regular la potencia suministrada a la resistencia, al accionar la resistencia el liquido en el tanque comienza a calentarse por lo que el sensor de temperatura que se encuentra sumergido en el tanque comienza a realizar la lectura. El sensor de temperatura RTD PT100 es un instrumento de medición en el cual dependiendo la temperatura en los alrededores del instrumento su resistencia entre las terminales va a variar, es característico de la PT100 que su resistencia en 0°C sea de 100 Ω , la PT100 es conectada al transmisor el cual entrega al autómatas una señal de 4 - 20 mA que como se mencionaba anteriormente, se convierte a voltaje usando una resistencia.

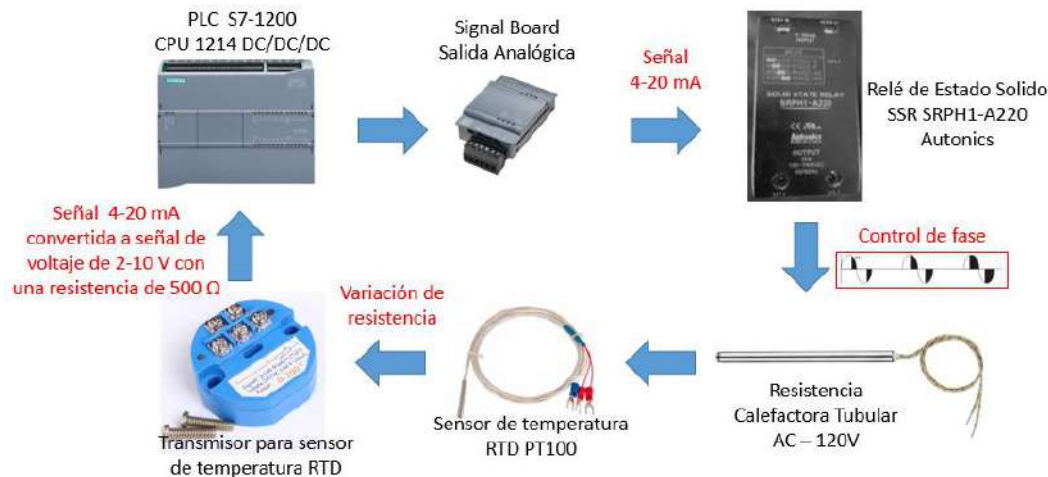


Figura 6.9: Diagrama ampliado funcionamiento de la planta de Temperatura

Fuente: Autor

Con el transmisor la conversión de la señal proveniente del sensor en la programación se realiza de manera sencilla, en el caso del autómatas utilizada solo se requiere de dos bloques, que son normalización y escalado. Para verificar si la temperatura medida es la real se usó un sensor de temperatura de mercurio ver figura 6.10 los cuales son muy exactos, este sensor de temperatura tiene un rango de 0 a 110°C en la figura 6.11 se muestra el sensor de mercurio inmerso en el tanque donde se tiene el líquido sometido a cambios de temperatura, al igual en el tanque está inmersa la PT100, se compararon resultados y el error asociado a la temperatura real es apenas de 1.3%, el cual es un error aceptable. A pesar de que la lectura era bastante precisa se encontraban momentos donde las decimales fluctuaban bastante, para evitar esto, se aplicó el filtro denominado EMA (Exponential Moving Average) o media móvil exponencial ver figura 6.14, a este filtro se le ingresa un parámetro alfa, que nos indica que

tan estricto va a realizar el filtrado.



Figura 6.10: Medidor de temperatura de mercurio

Fuente: Autor



Figura 6.11: Comparación del valor capturado de temperatura

Fuente: Autor



Figura 6.12: Planta temperatura

Fuente: Autor

6.1.2.2 Identificación

Para realizar la identificación de la planta de temperatura para determinar si es lineal o no, se tomó un valor constante, comprendido entre 4-20mA que es el rango de la señal de control, este valor escogido fue 8 mA, los cuales se enviaron de manera constante al elemento final de control en este caso el SSR para visualizar si la temperatura se establecía en algún punto, el resultado se muestra en la figura 6.13. Lo que se puede deducir de la figura 6.13 es

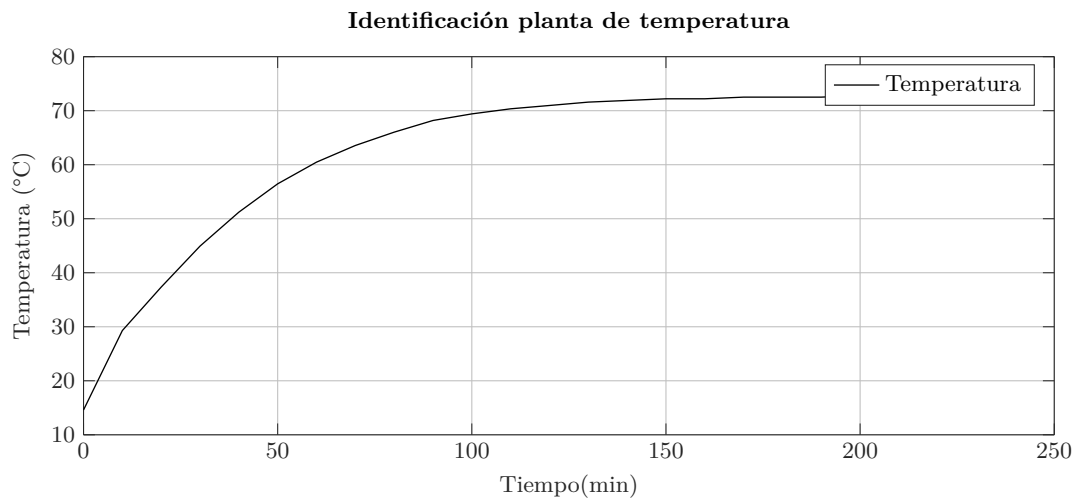


Figura 6.13: Identificación planta de temperatura

Fuente: Autor

que la temperatura va seguir aumentando mientras avance el tiempo, por lo que se puede deducir que la planta de temperatura es una planta inestable a lazo abierto, por lo tanto lo convierte en una planta no lineal.

6.1.3 Planta de Flujo

6.1.3.1 Acondicionamiento

En la planta de flujo se realizó una adaptación de la entrada del suministro, esta entrada se acopló directamente al suministro por medio de una manguera, e igualmente la salida se le adicionó una manguera para hacer llegar el agua saliente a su desembocadura.

Para la lectura del flujo en litros/minuto con el sensor de flujo se realizó un acondicionamiento de la señal, el procedimiento para realizar un circuito con un transistor como conmutador, ya que la señal que entrega el sensor es de 5V, por lo tanto el PLC no es capaz de detectarlo por eso, por medio de un transistor se realiza la etapa de amplificación de la conmutación de la señal, ya que el sensor de flujo nos entrega una señal digital de pulsos.

Para poder realizar la lectura de caudal basta con leer la frecuencia de los pulsos del encoder interno, para esto se hizo utilidad de los contadores rápidos mencionados en la planta de velocidad, se usó la función creada para medir velocidad ver figura 6.3 pero modificada para que nos entregara la frecuencia, y con la ecuación 6.1 proporcionada por el distribuidor de este tipo de sensores se calcula el caudal en litros/min donde K para el sensor usado es igual a 5.5, f es la frecuencia en Hz y Q es el caudal en l/min.

$$F = k * Q \quad (6.1)$$

Esta ecuación se ingresa en una función dentro del software para obtener la lectura en el formato necesario, sin embargo, el resultado fluctuaba bastante lo que indica que la señal trae un ruido asociado, este ruido puede afectar el control al momento de implementarlo por tal razón se implementó un filtro por programación para tratar de disminuir ese ruido, se optó por aplicar nuevamente el filtro EMA 6.14 a la señal de retroalimentación del controlador. En la figura 5.10 se ilustra el lazo de control en bloques, en la figura 6.15 se ilustra el lazo un poco más detallado donde se explica las señales que hay entre cada elemento. El controlador en este caso el autómatas genera por medio de la signal board una señal de 4 a 20 mA que lo recibe la electroválvula, la cual se abre de 0 a 90°, siendo 0° totalmente cerrada (4mA) y 90° totalmente abierta (20mA), esta permite el flujo de agua que comienza a detectar el sensor de flujo, por lo tanto internamente empieza a mover el encoder y genera una frecuencia de pulsos esta señal realiza la retroalimentación al sistema, es decir, el lazo cerrado de control.



Figura 6.16: Planta de control de Flujo

Fuente: Autor

6.1.3.2 Identificación

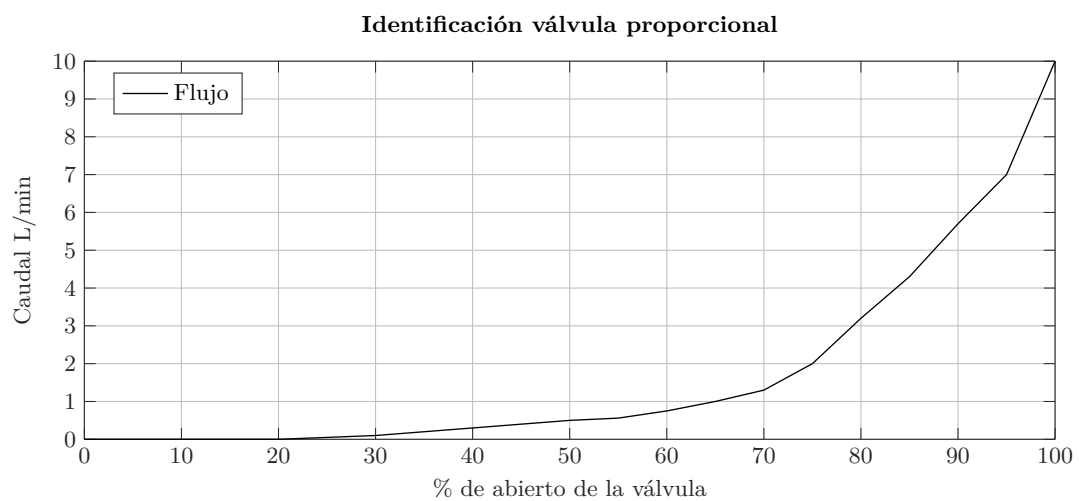


Figura 6.17: Caudal vs porcentaje abierto de la válvula

Fuente: Autor

Para identificar si el sistema es lineal o no lineal se realizó la siguiente identificación de la válvula proporcional de bola, se capturo el caudal cuando la válvula se iba a abrir proporcionalmente de 0 a 100, la respuesta se ilustra en la figura 6.17, con lo que se puede deducir que la planta por el elemento final de control no es lineal.

6.2 Control clásico aplicado

Con el fin de comparar la respuesta del neuro-controlador con otros controladores, se realiza control clásico PI, PD o PID para cada planta seleccionada, la figura 6.18 muestra la posición del controlador PID en el sistema en lazo cerrado.

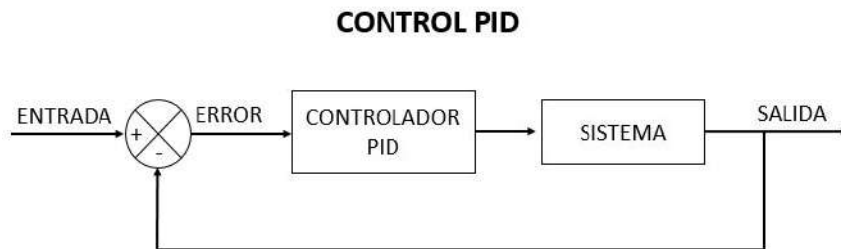


Figura 6.18: Control PID en lazo cerrado

Fuente: Autor

Para realizar este tipo de control se hizo uso de una herramienta disponible en el software de programación TIA PORTAL denominado "PID_Compact" que permite realizar un controlador PI o PID, esta función viene empaquetada como un bloque en el cual solo seleccionamos salidas y entradas, además cuenta con un menú de configuración y sintonización. A continuación se explicará de una manera más detallada el proceso de implementación y sintonización del PID, se utilizará como referencia el proceso realizado en la planta de velocidad, la metodología para el uso del controlador es la misma en cada una de las plantas, en lo único que varía es en las configuraciones de cada planta. En el llamado de la función de PID se debe realizar desde una interrupción, en el software de TIA PORTAL se pueden generar bloques de interrupción, dependiendo de la dinámica de la planta se ajusta el tiempo de interrupción, el proceso para generar este bloque se muestra en la figura 6.19, se selecciona "Agregar nuevo bloque", después se escoge la opción "bloque de organización" y del tipo "Cyclic interrupt" y por último se escoge el tiempo en el que se quiere realizar la interrupción que debe ser un valor en milisegundos que puede ir desde 1 hasta 60000 milisegundos, este valor se escoge dependiendo de la dinámica del sistema.

Creado este bloque en la parte lateral izquierda aparece con la dirección de bloque por defecto "OB30", teniendo el bloque de interrupción se procede a buscar la función "PID_Compact" el cual está ubicado en la pestaña de "Instrucciones" ubicada en un menú desplegable en la parte derecha del software y sub-pestaña "Tecnología" se selecciona la carpeta "PID Control" - "Compact PID" y luego se escoge la función a utilizar "PID_Compact" ver figura 6.20.

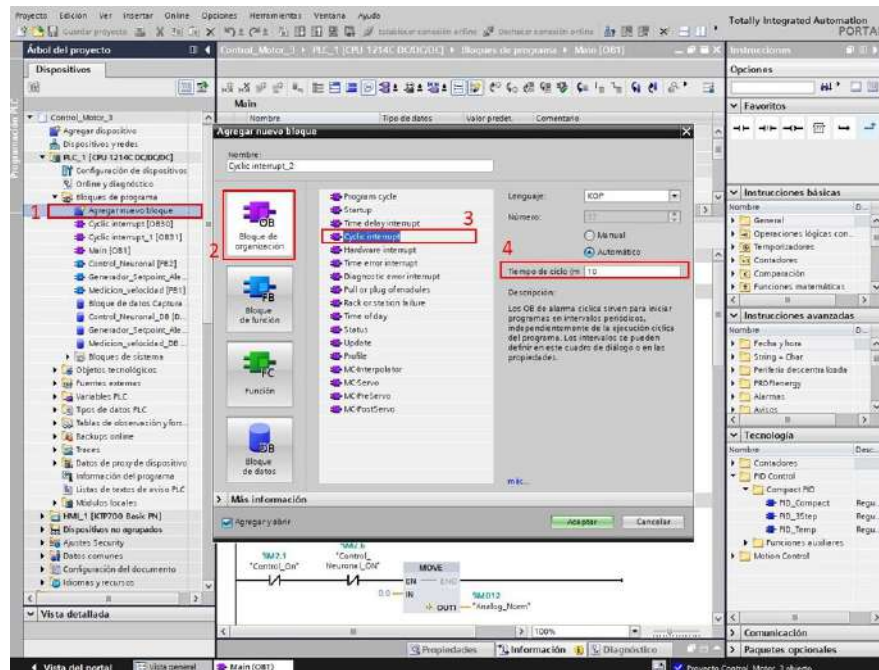


Figura 6.19: Creación del bloque de interrupción

Fuente: Autor

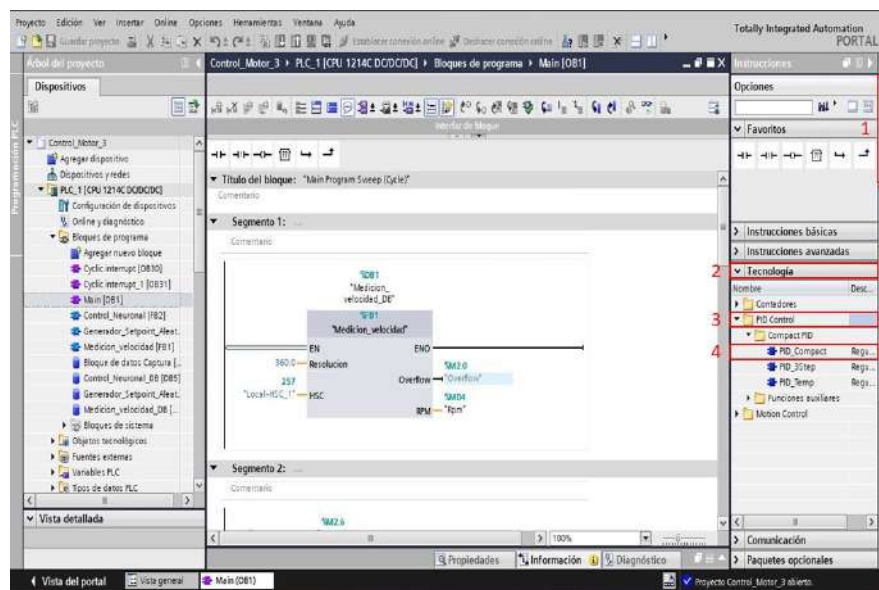


Figura 6.20: Creación del bloque "PID_Compact"

Fuente: Autor

Seleccionado y arrastrado este bloque al OB30, se genera un DB de instancia al ser llamado, además en la parte derecha del programa en la sección de "Árbol del proyecto" - "Dispositivos" - "PLC_1" - "Objetos Tecnológicos" se genera un objeto "PID_Compact_1" que al

desplegarlo tiene 2 pestañas una de configuración y otra de puesta en servicio ver figura 6.21. La pestaña de configuración abre una ventana donde se deben ajustar parámetros antes de poner a funcionar el PID, estos se dividen en 3 tipos de ajustes: "Ajustes básicos" - "Ajustes del valor real" y "Ajustes avanzados", el ajuste primordial se encuentra en ajustes básicos donde se tiene un parámetro denominado "Tipo de regulación" donde se despliega un menú que contiene los sistemas más comúnmente utilizados ver figura 6.22, en este caso este parámetro se ajustan Revoluciones y su escala en revoluciones por minuto, en la configuración de los parámetros de entrada y salida en "Input" y "Output" se debe seleccionar si es una entrada normal (Input) o una entrada proveniente directamente de una entrada analógica (Input PER), la señal de entrada es la lectura del encoder por tal razón se escoge la opción "Input" y en cuanto a la salida, se puede elegir entre una señal de 0 - 100 o una salida directa analógica (Output PER), sin embargo, para esta planta como se trabaja una CPU DC/DC/DC ofrece una tercera opción de salida, refiriéndose a la salida PWM, para efectos del programa se utilizó la señal de 0-100 ya que permite tener una mejor visión de la señal de control al momento de analizarla. Además de los 2 parámetros mencionados anteriormente,

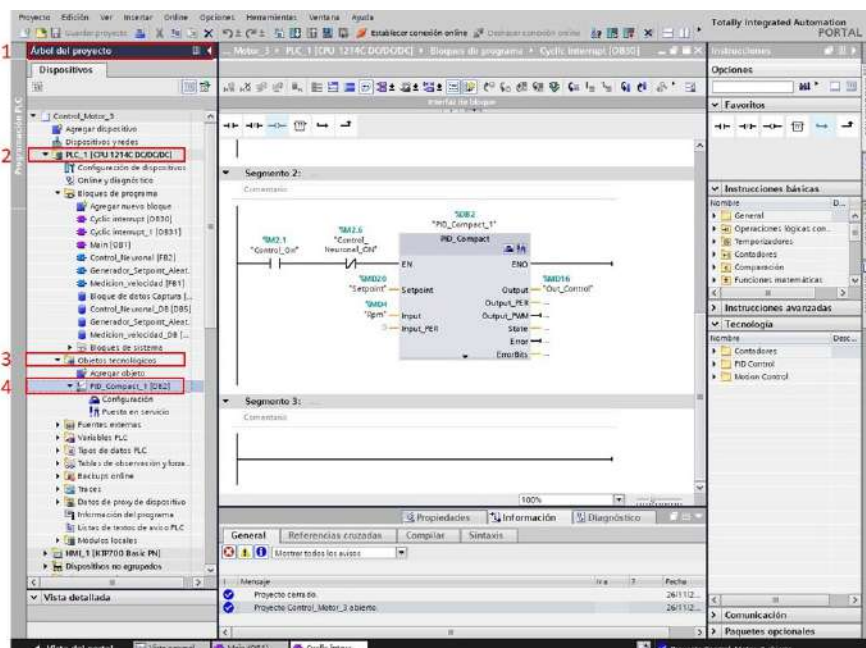


Figura 6.21: Objeto tecnológico PID

Fuente: Autor

en "Ajustes de valor real" se establece el límite superior del valor real, este parámetro es muy importante ajustarlo para un correcto desempeño del control.

Realizado los ajustes, se ingresan los datos requeridos al bloque PID, que se encuentra en el OB30 o bloque de interrupción, los parámetros que se ingresan son los mostrados en la figura 6.23, se ingresa el setpoint o valor deseado, input que es la señal de retroalimentación en este caso la velocidad en rpm, y en Output la señal de control que se envía después de un tratamiento al elemento final de control.

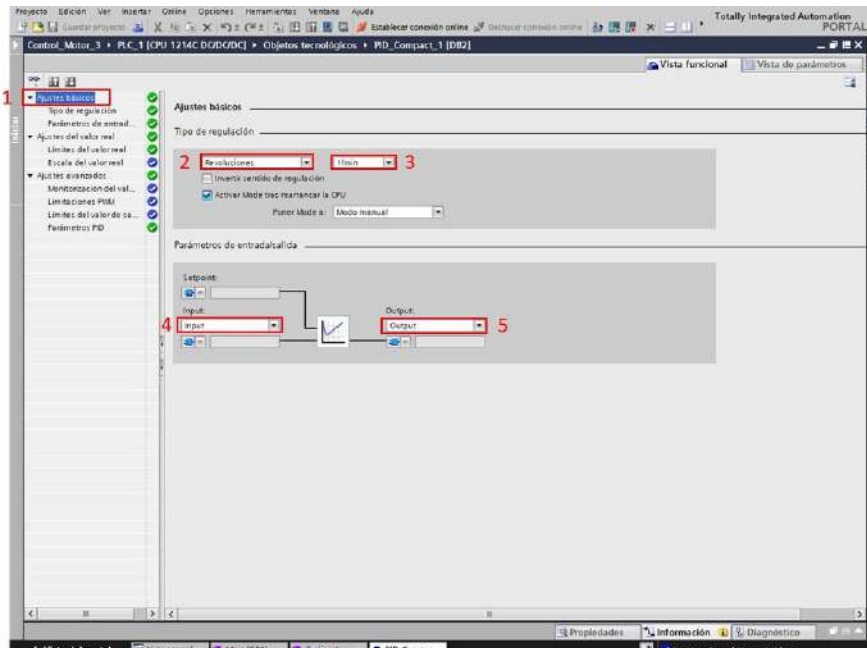


Figura 6.22: Ajuste Controlador PID velocidad

Fuente: Autor

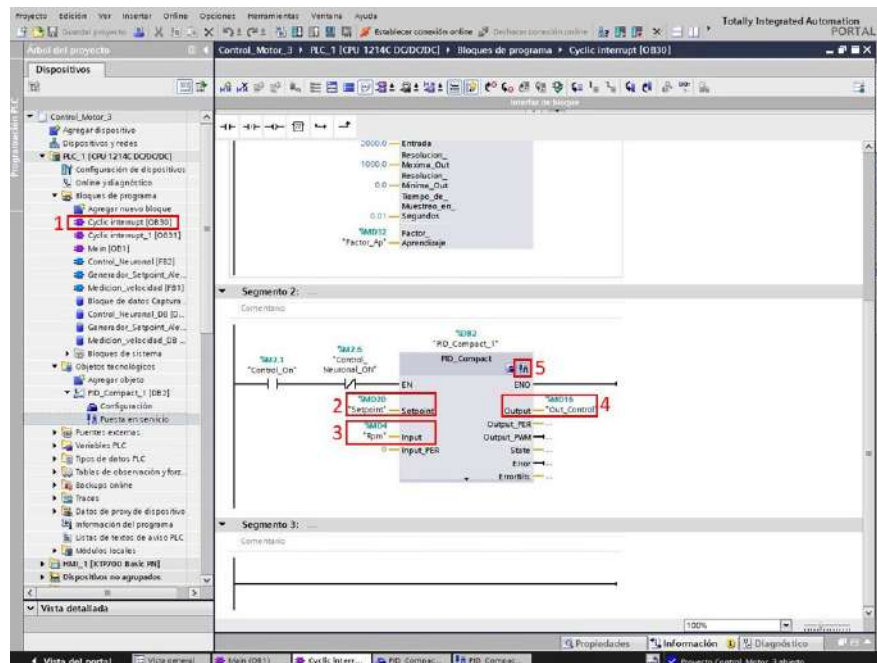


Figura 6.23: Parámetros del bloque PID

Fuente: Autor

Para poner en marcha el controlador PID, éste se debe sintonizar, es decir, calcular las constantes adecuadas para el sistema, el software TIA PORTAL cuenta con una herramienta

de sintonización para el control PID, en la pestaña "Puesta en servicio" que se encuentra justo debajo de los ajustes del PID, en la figura 6.24 se puede ver la interfaz de la herramienta, para comenzar a realizar la sintonización de la planta se activa el tiempo de muestreo, al activar este botón se comienza a medir los 3 parámetros ingresados al bloque PID: Setpoint, señal de entrada y salida de control.

Para empezar la sintonización de la planta, se debe dirigir al "Modo ajuste" en este menú desplegable se encontraran dos opciones "**Optimización inicial**" y "**Optimización fija**", las dos se ejecutan por cierto intervalo de tiempo y generan unas constantes al sistema que después de cargan. **Optimización inicial** se usa para arrancar el el sistema y su función

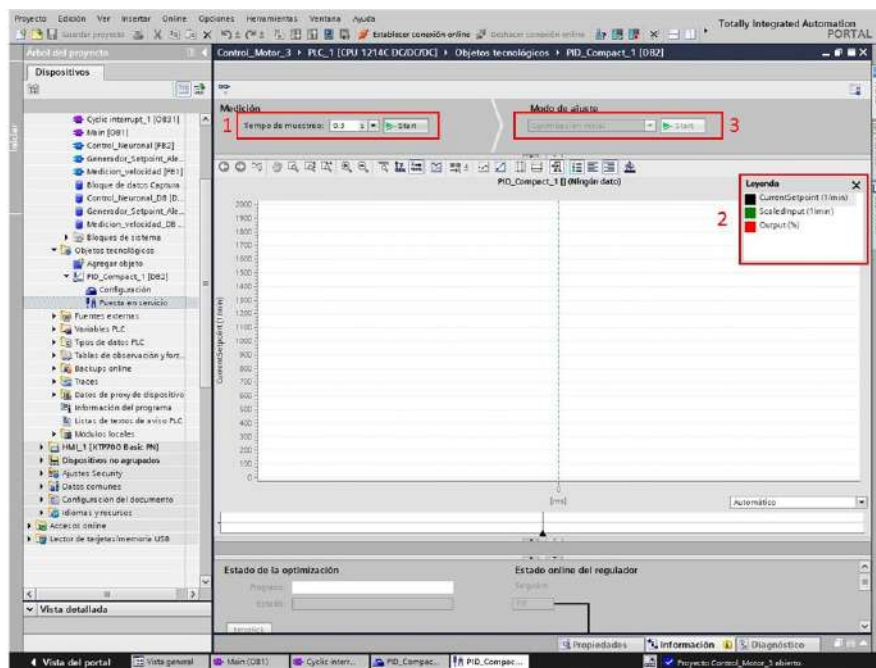


Figura 6.24: Sintonización PID

Fuente: Autor

es como la de un "step" donde se da un paso y dependiendo de la respuesta del sistema configura los parámetros adecuados, sin embargo, esta respuesta puede llegar a tener un error en estado estacionario, o simplemente no se sintoniza de la manera adecuada, para tener una mejor respuesta del controlador se ejecuta la **optimización fina**, que regularmente ocupa un intervalo más grande de tiempo en su ejecución que la inicial, está realiza una secuencia de pasos por arriba y abajo del setpoint, de esta manera buscando disminuir el error en estado estacionario, este proceso requiere de que la señal de entrada disminuya para poder realizar el cambio tanto por arriba y por abajo del setpoint, al terminar la ejecución de esta optimización la respuesta del controlador mejora bastante, suele repetirse varias veces hasta encontrar el equilibrio deseado.

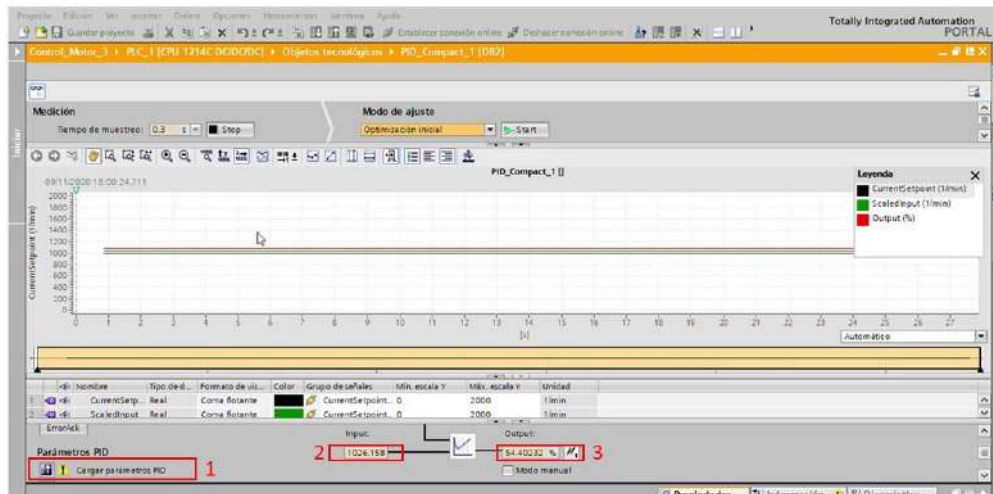


Figura 6.25: Carga de constantes calculadas del PID

Fuente: Autor

Realizado la sintonización se proceden a cargar los parámetros generados en la sintonización, en la figura 6.25 se muestra que en la parte inferior izquierda hay un apartado para realizar esta operación.

Cargado los parámetros al PLC, se puede desactivar la conexión online, ya que las constantes quedan cargadas en el autómata, para su ejecución.

6.2.1 Control PID en la planta de Velocidad

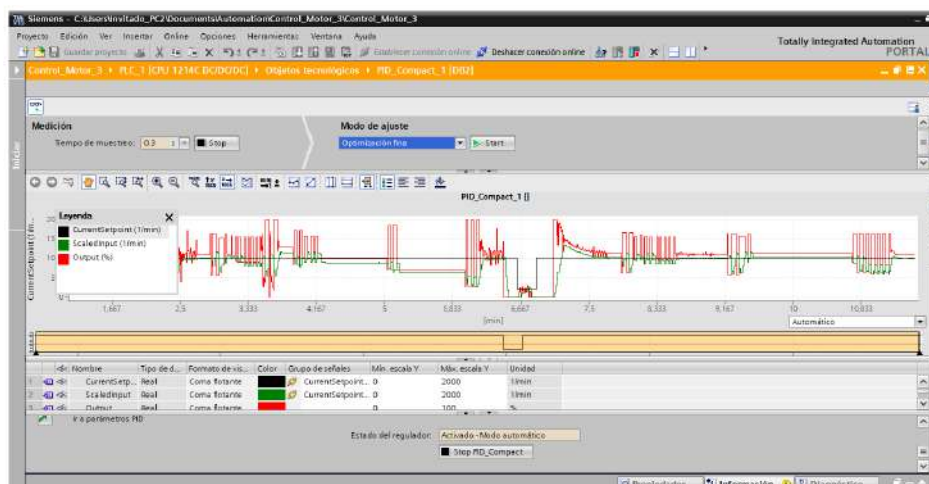


Figura 6.26: Sintonización final PID velocidad

Fuente: Autor

Para aplicar el control PID se realizó paso a paso lo anteriormente mencionado. Las configuración del bloque PID se puede ver en la figura 6.22, se configura el tipo de regulación en

revoluciones, en la magnitud de revoluciones por minuto, se escoge la entrada estándar y la salida estándar de 0 a 100, el valor máximo se establece en 1850, ingresado estos parámetros se procede a realizar la sintonización.

Parámetros PID:

Activar entrada manual

Genancia proporcional: 1.441654E-2

Tiempo de integración: 4.281239E-1 s

Tiempo derivativo: 5.255671E-2 s

Coefficiente retardo derivativo: 0.1

Ponderación de la acción P: 1.0

Ponderación de la acción D: 0.0

Tiempo muestreo algoritmo PID: 0.04999 s

Regla para la optimización

Estructura del regulador: PID

Figura 6.27: Parámetros PID velocidad

Fuente: Autor

La figura 6.26 muestra el historial del proceso de sintonización, es de importancia recalcar que el proceso de optimización fina se realizo varias veces para encontrar una respuesta aceptable. Las constantes generadas en el proceso se muestran en la siguiente figura 6.27.

6.2.2 Control PI en la planta de Temperatura

En el control PI para la planta de temperatura las configuraciones del bloque PI se muestran en la figura 6.28, el tipo de regulación escogida fue temperatura en la magnitud de grado Celsius, la entrada, en este caso la captura por la PT100 la procesamos fuera del bloque, por lo tanto se ingresa en la entrada estándar "Input" y la salida también la estándar de 0-100 que se procesa por fuera del bloque y después se direcciona a las entradas del SSR.

Con estos ajustes se realiza la sintonización de la planta, como la temperatura tiene una dinámica muy lenta el proceso de sintonización demora un tiempo considerable, la figura 6.29 muestra la interfaz después de culminar el proceso de sintonización del PID.

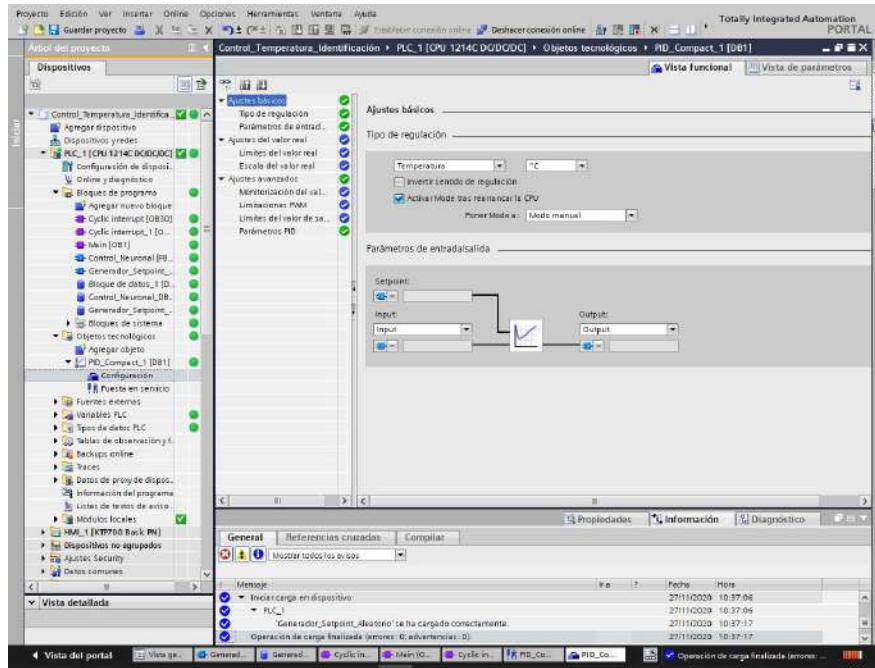


Figura 6.28: Ajustes control PI temperatura

Fuente: Autor

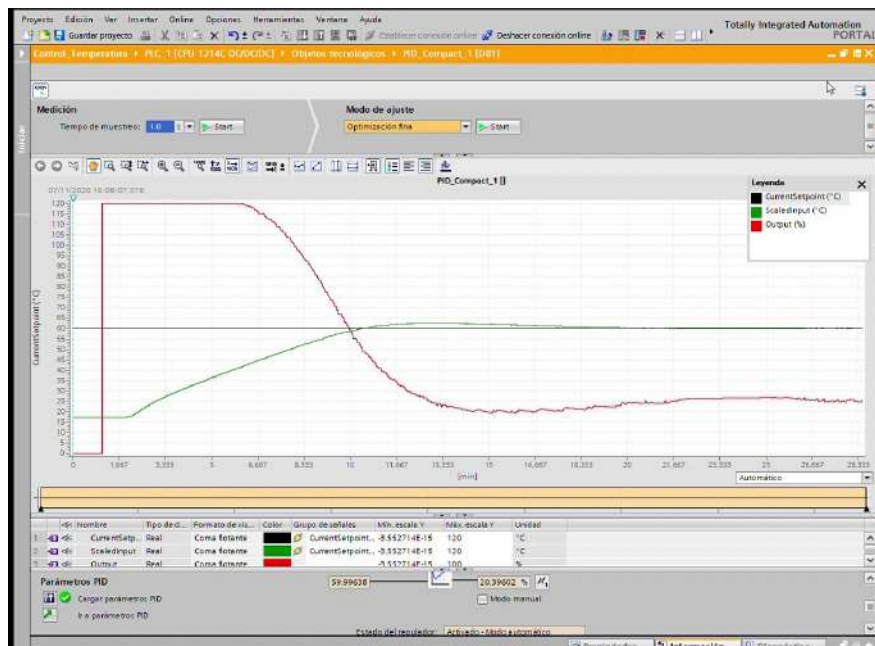


Figura 6.29: Sintonización final PI temperatura

Fuente: Autor

Las constantes del controlador PI de la planta de temperatura se muestran en la figura

6.30.



Figura 6.30: Parámetros PI temperatura

Fuente: Autor

6.2.3 Control PI en la planta de flujo

En el control PI de la planta de flujo las configuraciones del bloque se muestran en la figura 6.31, el tipo de regulación es caudal y como unidad se selecciona litros/min ya que es la unidad en la que mide el sensor, la entrada al controlador es por medio de pulsos en las entradas rápidas del PLC, que tiene como valor máximo 9L/min y en la salida se configuro la estándar de 0 a 100 que luego se normaliza y escala para enviarla a la válvula proporcional.

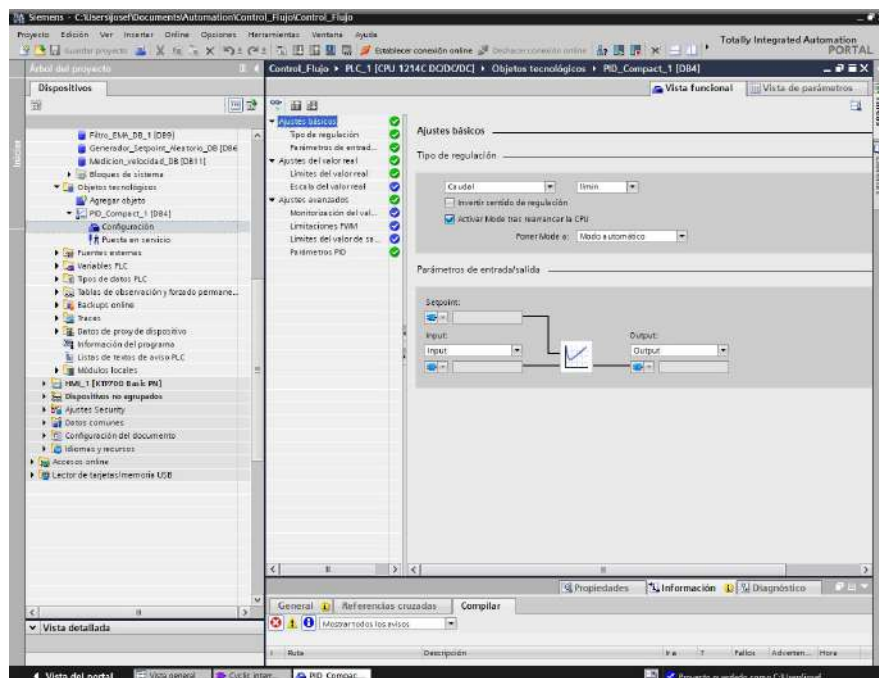


Figura 6.31: Ajustes control PI de flujo

Fuente: Autor

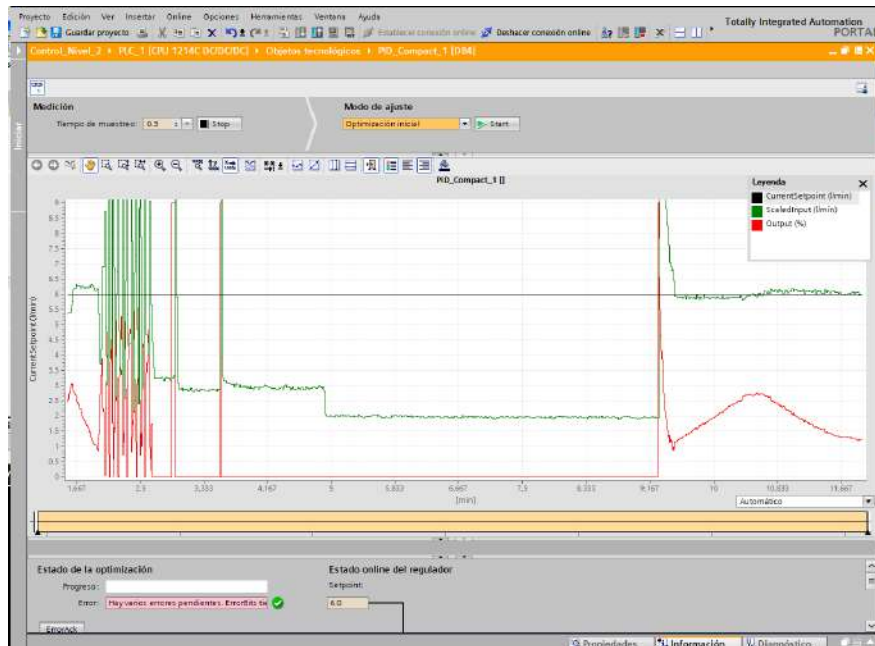


Figura 6.32: Sintonización final PI de flujo

Fuente: Autor

La captura al terminar la sintonización del controlador se evidencia en la figura 6.32, la optimización fina en la planta de flujo se realizó 4 veces para llegar a un control muy robusto. Los parámetros generados por el sintonizador se pueden ver en la figura 6.33.

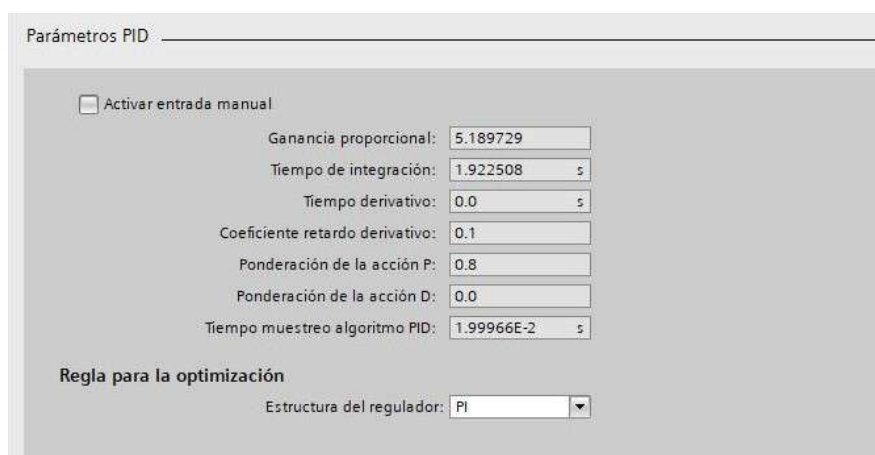


Figura 6.33: Parámetros PI flujo

Fuente: Autor

6.3 Control neuronal adaline directo

Un controlador neuronal directo adaline es una red neuronal artificial del tipo adaline que hace las veces de controlador ver figura 6.34, este tipo de red se asemeja mucho en su estructura a un controlador PID discreto ver sección 5.4, pero con las ventajas de que no hay que sintonizarlo y también de que se adapta a las condiciones cambiantes del sistema, este algoritmo es una técnica de control avanzada que no requiere conocimiento del sistema a controlar y que mediante el cambio de algunos parámetros puede realizar el control de manera rápida y estable.

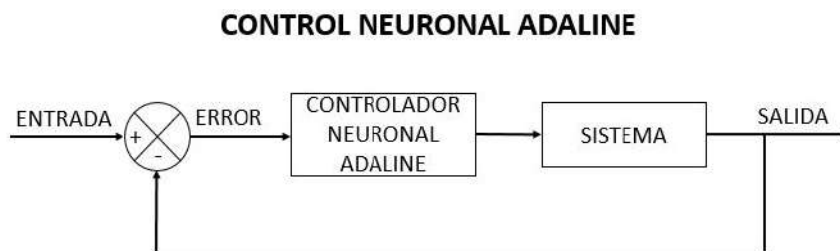


Figura 6.34: Control Neuronal adaline en lazo cerrado

Fuente: Autor

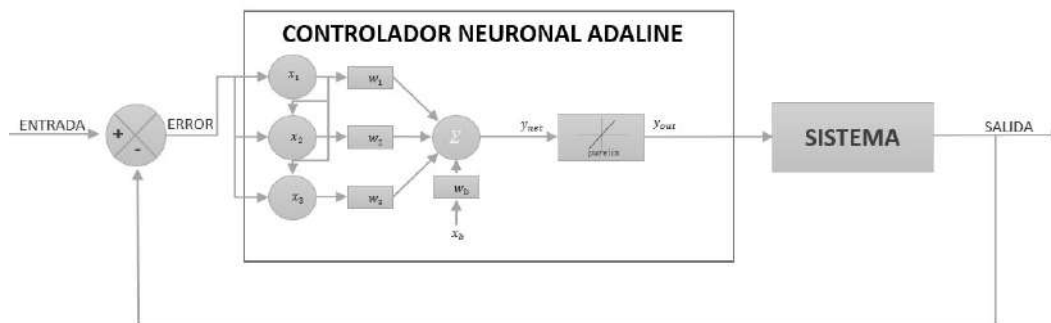


Figura 6.35: Topología red neuronal aplicada

Fuente: Autor

$$e = \text{Setpoint} - \text{Valor.actual} \quad (6.2)$$

$$x_1 = e(t) \rightarrow x_1 = e(k) \quad (6.3)$$

$$x_2 = \frac{de(t)}{dt} \rightarrow x_2 = \frac{e(k) - e(k-1)}{T_m} \quad (6.4)$$

$$x_3 = \int e(t)dt \rightarrow x_3 = \left(\frac{e(k) + e(k-1)}{2}\right) * T_m \quad (6.5)$$

$$x_b = 1.0 \rightarrow x_b = 1.0 \quad (6.6)$$

La figura 6.35 muestra la topología de la neurona a implementar que son cuatro neuronas de entrada y una de salida (4,1), de las cuatro entradas, tres son variables y una entrada fija que es llamada bias, cada una de las entradas variables se basa en el error existente en el sistema, este se define como la diferencia entre la señal deseada (Setpoint) y la señal actual (Valor actual) ecuación 6.2; la **primera** entrada de la neurona corresponde al error neto ver ecuación 6.3, la **segunda** entrada corresponde a la razón de cambio del error (derivada) ecuación 6.4, la **tercer** entrada correspondiente a la suma de los errores anteriores (integral) ecuación 6.5 y la **cuarta** entrada que es el bias constante 6.6, cada entrada a la neurona tiene un peso asociado, este es el valor se ajusta para converger a la salida deseada, la matemática asociada a este algoritmo se explica en sección 5.3, la salida y_{net} corresponde a la sumatoria de los pesos por las entradas, este valor pasa por la función de activación purelin ver ecuación 4.10, que es característica de la red neuronal adaline, el valor resultante es la salida y_{out} que es la señal de control que recibe el sistema ha controlar como se ilustra en la figura 6.34. Para recrear el algoritmo de control neuronal adaline se uso el lenguaje SCL descrito en la sección 5.8, el algoritmo se llevo a un bloque de función (FB) en el software TIA PORTAL, el bloque de función tiene asociado por defecto un bloque de datos (DB) lo que nos permite almacenar los datos de las variables de cada ciclo de ejecución ha diferencia de una función (FC). Al bloque de función se le declara entradas y salidas para que sean introducidas por el programador al momento de llamar el bloque y adaptarlos a sus necesidades, en la figura 6.36 se visualiza el bloque estructurado internamente en SCL, el bloque cuenta con siete entradas y una salida. Al igual que el bloque de PID, el control neuronal se recomienda ser llamado desde un ciclo de interrupción para un mejor funcionamiento, el tiempo varia según la dinámica del sistema que se quiera controlar, además es una entrada del bloque de control neuronal.

Parámetros de entrada al Bloque de Función "Control Neuronal Adaline":

- **Señal de Entrada:** Es la señal proveniente del instrumento de medición.
- **Setpoint:** Es el valor deseado, es decir, el valor en el que se quiere establecer la variable de entrada.
- **Valor máximo de entrada:** Es el valor máximo que alcanza la variable de "Señal de entrada".
- **Resolución máxima de salida:** Valor máximo de la salida "Señal de salida".
- **Resolución mínima de salida:** Valor mínimo de la salida "Señal de salida".
- **Muestreo en segundos:** El tiempo en segundos en el que se esta ejecutando el algoritmo.
- **Factor de Aprendizaje:** Taza o valor con el que se actualizan los pesos de la neurona, usualmente toma valores entre 0 y 1.

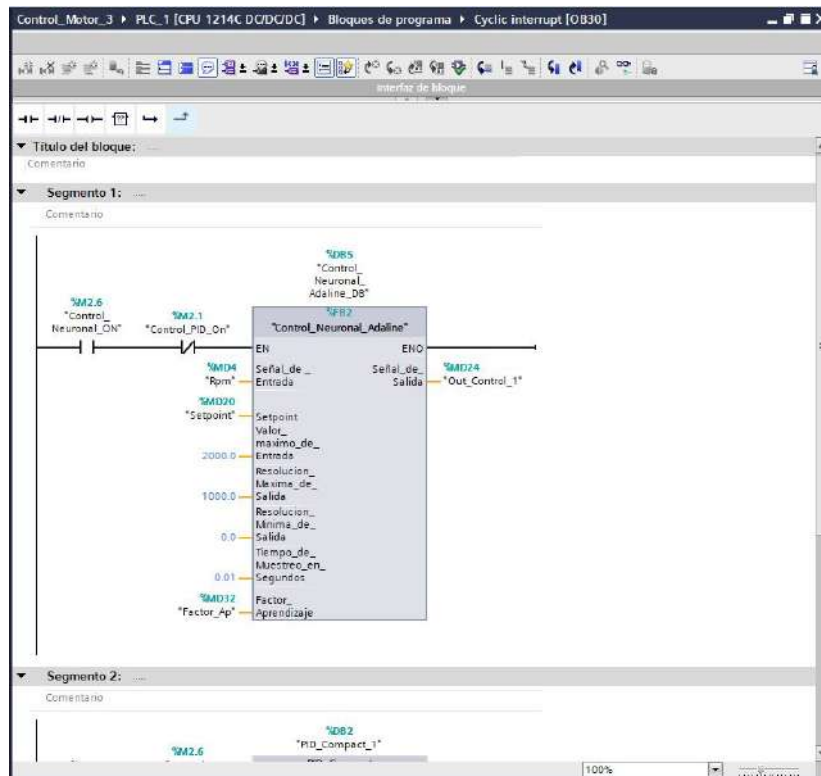


Figura 6.36: Bloque de función control neuronal adaline para velocidad

Fuente: Autor

Parámetro de salida al Bloque de Función "Control Neuronal Adaline":

- **Señal de Salida:** Es la salida de la neurona o señal de control, está es la entrada al elemento final de control del sistema, su valor máximo y mínimo se configuran con las entradas "Resolución máxima de salida" "Resolución mínima de salida" respectivamente.

El factor de aprendizaje ingresado es uno de los parámetros más importantes para el control neuronal ya que es el que le da la velocidad de actualización de pesos, encontrar el valor adecuado es fundamental para que el algoritmo converja de manera adecuada al valor deseado (Setpoint), teóricamente un factor de aprendizaje hace que la neurona converja rápidamente al valor deseado, esto ocasiona sobre-picos y oscilaciones alrededor del setpoint y uno muy pequeño puede causar que converja muy despacio al valor deseado.

A continuación se ilustra el código en SCL del control neuronal adaline 6.1, la figura 6.37 muestra la declaración de variables en la tabla del software TIA PORTAL, estas variables son propias del bloque de función y se almacenan en el DB de instancia correspondiente.

Código 6.1: Algoritmo de control neuronal adaline

```

1   //( *Algoritmo de control Neuronal Adaline
2   //Este algortimo permite realizar el control de una variable en lazo cerrado.

```



```

3 //El algoritmo esta basado en la topología de una red neuronal Adaline, es un ↵
    ↵ algoritmo de control avanzado adaptativo.
4 //Versión de prueba: 1.0
5 //Fecha de creación: 15/11/2020
6 //Autores: Juan Caceres, Fernando Ramírez // Universidad de Pamplona - Colombia
7 //El algoritmo está estructurado en SCL para ser insertado en un Bloque de ↵
    ↵ función y llamado en un ciclo de interrupción.
8 //*)
9 //Inicio de pesos
10 IF #Inicio_Pesos = FALSE THEN
11     FOR #i := 0 TO 3 DO
12         #W_train[#i] := 0.01;
13     END_FOR;
14     #Inicio_Pesos := True;
15 END_IF;
16 //Calculo del Error
17 #Error_V := (#Setpoint - #"Señal_de _Entrada");
18 //Normalización del Error
19 #Error[0] := NORM_X_REAL(MIN := 0.0, MAX := 1800.0, VALUE := #Error_V);
20 //Memoria del Error
21 #MErrorArray[0] := #Error[0];
22 FOR #i := 10 TO 1 BY -1 DO
23     // Statement section FOR
24     #MErrorArray[#i] := #MErrorArray[#i - 1];
25 END_FOR;
26 //Entradas
27 #X_Input[0] := #Error[0]; //Entrada del Error Normalizado
28 #X_Input[1] := (#Error[0] - #Error[1]) / (#Tiempo_de_Muestreo_en_Segundos); //↵
    ↵ Entrada derivativa
29 #X_Input[2] := (#X_Input[2] + ((#Error[0] + #Error[1])/2.0))* (#↵
    ↵ Tiempo_de_Muestreo_en_Segundos); //Entrada integral
30 #X_Input[3] := 1.0 ; //bias
31 //Entrenamiento
32 //Calculo de la salida de la red
33 #net := #X_Input[0] * #W_train[0] + #X_Input[1] * #W_train[1] + #X_Input[2] * #↵
    ↵ W_train[2] + #X_Input[3] * #W_train[3];
34 //Actualizacion de pesos
35 IF #Error[0] <> 0.00 THEN
36     FOR #i := 0 TO 3 DO
37         #W_train[#i] := #W_train[#i] + (#Factor_Aprendizaje* (#Error_V) * #↵
    ↵ X_Input[#i]);
38     END_FOR;
39 END_IF;
40 //Saturador
41 #Out_Red := #net;
42 #Ouput_Sat := #Out_Red;
43 IF #Out_Red >= #Resolucion_Maxima_de_Salida THEN
44     #Ouput_Sat := #Resolucion_Maxima_de_Salida;
45 END_IF;
46 IF #Out_Red <= #Resolucion_Minima_de_Salida THEN
47     #Ouput_Sat := #Resolucion_Minima_de_Salida;

```

```

48 END_IF;
49 //AntiWindoup
50 #atw := #Out_Red - #Ouput_Sat;
51 IF #atw <> 0.00 THEN
52     FOR #i := 0 TO 3 DO
53         #W_train[#i] := #W_train[#i] - (#Factor_Aprendizaje * (( #Error[0]) * ←
54             ↪ ABS_REAL(IN:=#X_Input[#i])));
55     END_FOR;
56 END_IF;
57 #Ouput_Sat_1 := #Ouput_Sat;
58 #Error[1] := #MErrorArray[10];
59 // Salida de Control
60 #Señal_de_Salida := #Ouput_Sat;

```

Nombre	Tipo de datos	Valor predet.	Remanencia	Accesible d...	Escrib...	Visible en...	Valor de s...	Comentario
Input								
Señal_de_Entrada	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Setpoint	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Valor_maximo_de_En...	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Resolucion_Minima_...	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Resolucion_Minima_d...	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Tiempo_de_Muestreo...	Real	0.01	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Factor_Aprendizaje	Real	0.001	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Output								
Señal_de_Salida	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
InOut								
<Agregan>								
Static								
X_Input	Array[0..3] of Real		No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
W_train	Array[0..3] of Real		No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Error	Array[0..2] of Real		No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Ouput_Sat_1	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Ouput_Sat	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Out_Red	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
net	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Inicio_Pesos	Bool	false	No rem...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
MErrorArray	Array[0..10] of Real		No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Error_V	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
atw	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Tiempo_M	Real	0.0	No remane...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Temp								
I	Int							
Constant								
<Agregan>								

Figura 6.37: Variables del bloque de función control neuronal adaline

Fuente: Autor

6.3.1 Control neuronal adaline directo en la planta velocidad

El control de velocidad de un motor tiene una dinámica rápida, es decir, su variable cambia muy rápido en el tiempo, en la figura 6.36 se muestra el bloque generado en SCL el cual se le ingresa los datos correspondientes al sistema de control de velocidad, en señal de control se le ingresa el valor en RPM capturado por el encoder, el valor máximo que alcanza el motor en este caso 1850 RPM, la resolución de salida para esta planta se maneja de 0 a 1000 y el tiempo de muestreo del ciclo de interrupción es de cada 10ms o 0.01s, el factor de aprendizaje se ingresa por la HMI, para lograr ajustarlo a la dinámica de la planta.

6.3.2 Control neuronal adaline directo en la planta de temperatura

El bloque de control neuronal adaline para el sistema de temperatura se muestra en la figura 6.38, el bloque se le ingresa el valor medido por el sensor de temperatura en señal de entrada, el valor máximo para este sistema es de 120°C y la resolución en la que se trabaja la salida de la señal de control es de 0 - 50, igualmente el factor de aprendizaje se deja como entrada desde la HMI ya que se requiere encontrar el valor óptimo para que la planta converja de manera adecuada al valor deseado o (Setpoint), se realiza iterando el valor de aprendizaje mientras se visualiza el comportamiento de la planta, el tiempo de muestreo en esta planta se configuro a 0.5s ya que la señal de entrada cambia de manera lenta.

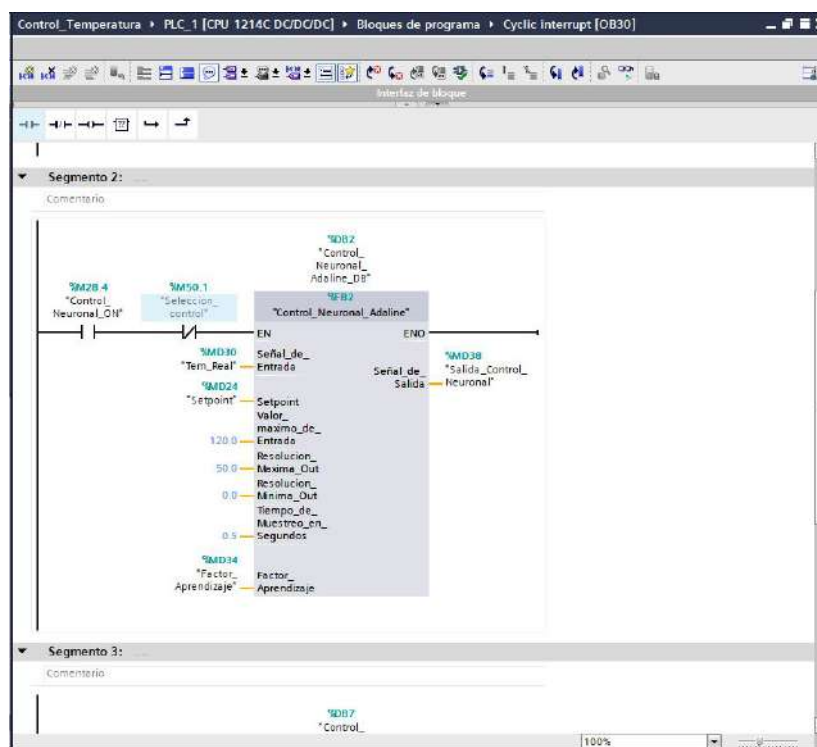


Figura 6.38: Bloque de función control neuronal adaline para temperatura

Fuente: Autor

6.4 Control neuronal multi-capa directo

Un controlador neuronal directo multi-capa es una red neuronal artificial del tipo perceptron multi-capa que hace las veces de controlador ver figura 6.39, las ventajas de estos tipos de controlados es que al igual que los controladores neuronales adalaine no hay que sintonizarlo, además, se adaptan a las condiciones cambiantes del sistema, este algoritmo es una técnica de control avanzada que no requiere conocimiento del sistema a controlar y que mediante el cambio de algunos parámetros puede realizar el control de manera estable y rápida.

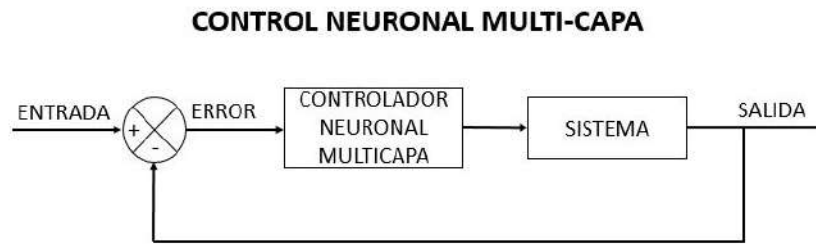


Figura 6.39: Control neuronal multi-capas en lazo cerrado

Fuente: Autor

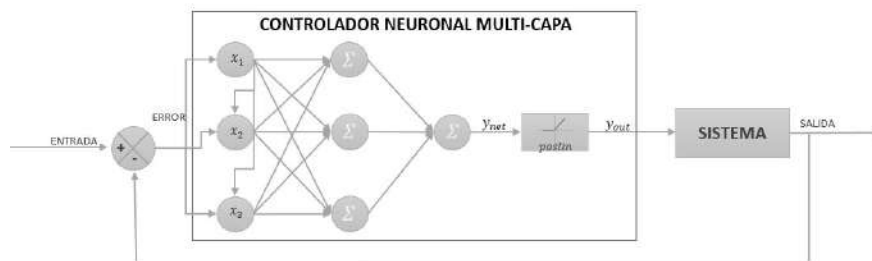


Figura 6.40: Topología de la red neuronal aplicada

Fuente: Autor

La figura 6.40 muestra la topología de la red neuronal a implementar, que es una red neuronal con tres neuronas a la entrada, tres neuronas en la capa oculta con función de activación logsig ver ecuación 4.8 y una en la capa de salida con función de activación poslim limitada ver ecuación 4.11, una estructura de (3,3,1). Las tres entradas reales son las mismas que entran al control neuronal adaline reciben el error existente en el sistema 6.2, este se define como la diferencia entre la señal deseada (Setpoint) y la señal actual (Valor actual); la **primera** entrada de la red neuronal corresponde al error neto 6.3, la **segunda** entrada corresponde a la razón de cambio del error (derivada) figura 6.4 y la **tercer** entrada correspondiente a la suma de los errores anteriores (integral) figura 6.5, cada interconexión en la red neuronal tiene un peso asociado, este es el valor que se ajusta para converger a la salida deseada, la matemática asociada a este algoritmo e explica en la sección 5.6, en donde se explica la forma de como se realiza la retro-propagación para re-calcular los pesos que hacen sintaxis en toda la red neuronal, la principal diferencia de la red neuronal multi-capas con la adaline, radica en que la multi-capas tienen muchas más conexiones lo que le permite tener más información del sistema almacenado.

Para generar el algoritmo de control neuronal multi-capas se utilizó el lenguaje SCL descrito en la sección 5.8, el algoritmo se estructuró en un bloque de función (FB) en el software TIA PORTAL, de tal manera que sea fácil de llevar a una librería, el bloque de función tiene asociado por defecto un bloque de datos (DB) lo que nos permite almacenar los datos de las variables de cada ciclo de ejecución. Al bloque de función trae variables de entrada y de salida, que son modificadas por el programador al momento de llamar el bloque para

ajustarlos a las necesidades del programa que este realizando, en la figura 6.41 se visualiza el bloque estructurado internamente en SCL.

El bloque de función cuenta con siete entradas y una salida al igual que el bloque de control neuronal adaline, además del bloque de función se generaron las funciones de activación de las red neuronal en funciones (FC):

- dposlin
- poslin_lim
- logsig
- tansig

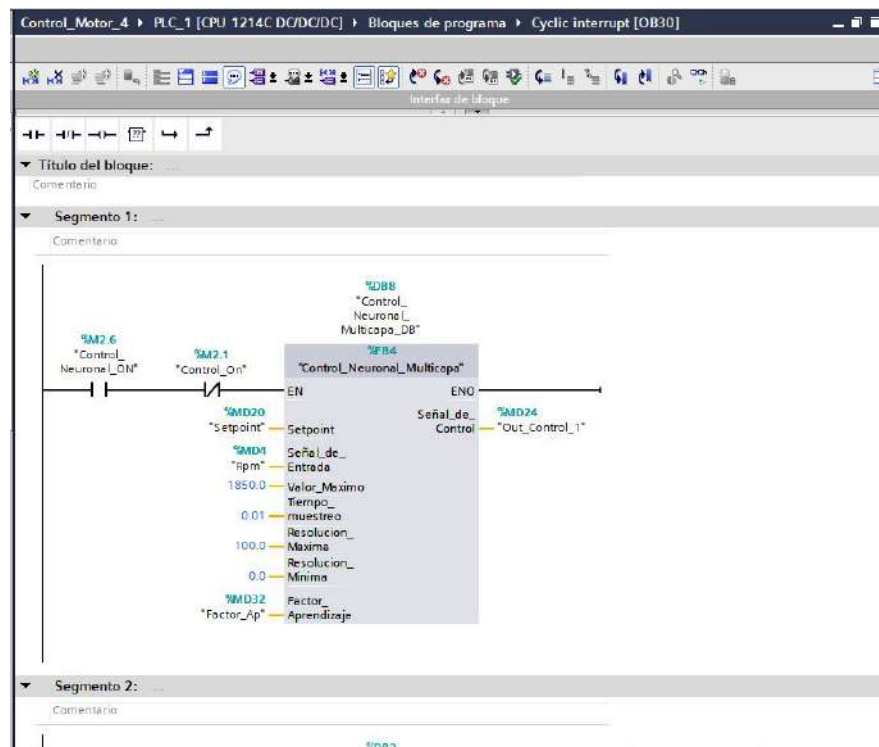


Figura 6.41: Bloque control neuronal multi-capa en la planta de velocidad

Fuente: Autor

Al igual que el bloque de control neuronal adaline, el control neuronal multi-capa se recomienda ser llamado desde una ciclo de interrupción para un funcionamiento adecuado, el tiempo varia según la dinámica del sistema que se quiera controlar, además es una entrada del bloque de control neuronal llamada tiempo de muestreo.

Parámetros de entrada al Bloque de Función "Control Neuronal Multicapa":

- **Señal de Entrada:** Es la señal proveniente del instrumento de medición.
- **Setpoint:** Es el valor deseado, es decir, el valor en el que se quiere establecer la variable de entrada.

- **Valor máximo de entrada:** Es el valor máximo que alcanza la variable de "Señal de entrada".
- **Resolución máxima de salida:** Valor máximo de la salida "Señal de salida".
- **Resolución mínima de salida:** Valor mínimo de la salida "Señal de salida".
- **Muestreo en segundos:** El tiempo en segundos en el que se esta ejecutando el algoritmo.
- **Factor de Aprendizaje:** Taza o valor con el que se actualizan los pesos de la neurona, usualmente toma valores entre 0 y 1.

Parámetro de salida al Bloque de Función " Control Neuronal Adaline":

- **Señal de Salida:** Es la salida de la neurona o señal de control, está es la entrada al elemento final de control del sistema, su valor máximo y mínimo se configuran con las entradas "Resolución máxima de salida" "Resolución mínima de salida" respectivamente.

Al igual que el control neuronal adaline el factor de aprendizaje ingresado es uno de los parámetros más importantes para el control neuronal ya que es el que le da la velocidad de actualización de pesos, encontrar el valor adecuado es fundamental para que el algoritmo converja de manera adecuada al valor deseado (Setpoint).

Importante El algoritmo desarrollado de control neuronal multi-capa, tiene la capacidad de realizar topologías más complejas, es decir, no está solamente limitado a una topología (3,3,1) si no que puede haber más salidas e igual que entradas, también se puede aumentar el numero de neuronas en la capa oculta, pero con cierto limite, sin embargo, para realizar estos cambios hay que configurar ligeramente el código fuente, por lo que se le hace complicado a un usuario normal realizar dichos cambios.

6.4.1 Control neuronal multi-capa directo en la planta velocidad

El bloque de control neuronal multi-capa de la planta de velocidad se ilustra en la figura 6.41, el bloque se llamo desde el OB30 de la planta de velocidad que tiene un tiempo de muestreo 10 ms, la señal de entrada es la lectura del encoder, el valor máximo equivale a 1800 rpm, con una resolución de salida de 0 a 100, el factor de aprendizaje se ingresa por la HMI, para ir ajustándolo de acuerdo al comportamiento de la planta.

6.4.2 Control neuronal multi-capa directo en la planta de temperatura

El bloque de control neuronal multi-capa ver figura 6.42 se le ingreso como parámetros de entrada la señal proveniente del sensor, como valor máximo 120°C, resolución de la salida de 0 a 100, además el bloque se esta ejecutando cada 100ms desde el bloque de interrupción OB30, de igual manera el factor de aprendizaje se ingresa por medio de la HMI, para ir ajustando su valor mientras está en marcha el sistema de control.

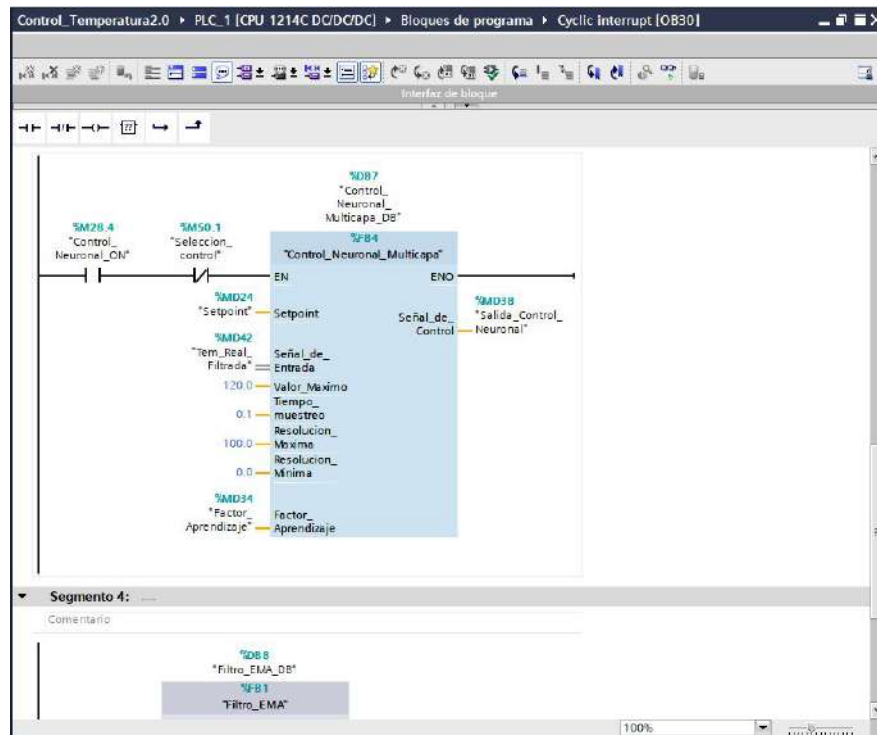


Figura 6.42: Bloque de control neuronal multi-capa en la planta de temperatura

Fuente: Autor

6.4.3 Control neuronal multi-capa directo en la planta de flujo

El bloque de control neuronal multi-capa para el sistema de control de flujo se muestra en la figura 6.43, al bloque se le ingresa el valor medido por el sensor de flujo, es decir, las pulsaciones que se traducen en una frecuencia que luego pasa por la ecuación 6.1, el valor máximo de esta planta es de 10 L/min, la resolución de salida se establece de 0 a 100, el factor de aprendizaje se deja para ingresar por al interfaz del usuario. El valor óptimo del valor de aprendizaje para lograr un buen funcionamiento del controlador se halló de manera empírica, se hace el llamado del bloque de control desde el ciclo de interrupción OB30 con un tiempo de ejecución cada 10ms.

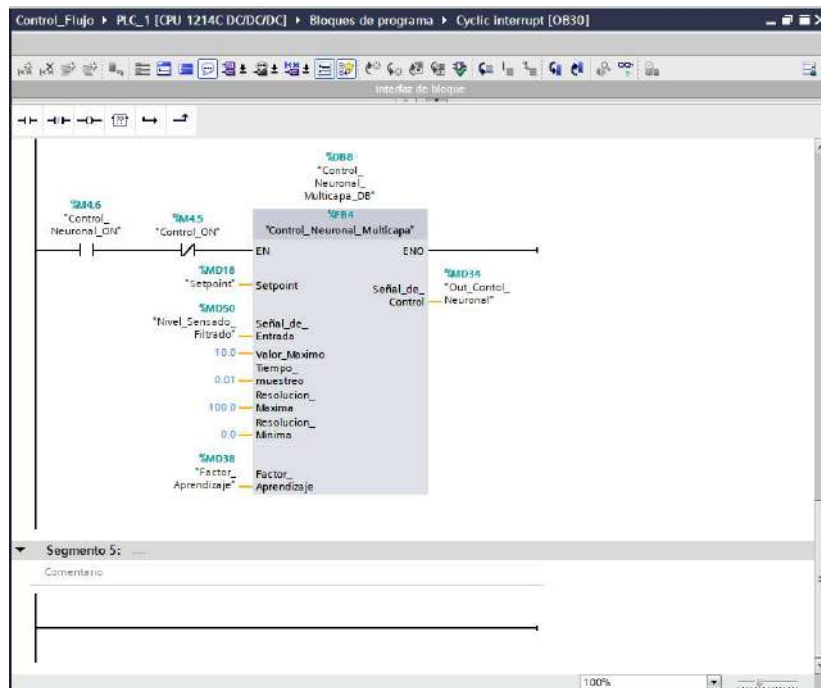


Figura 6.43: Bloque de control neuronal multi-capa en la planta de flujo

Fuente: Autor

6.5 Interfaz generada

En esta sección se presenta la interfaz generada en cada uno de los programas para cada planta, esta interfaz se realizó en el software TIA PORTAL con la pantalla KTP700 PN de la marca Siemens ver figura 5.22, la interfaz de manera general se basaba en una misma plantilla con las mismas funciones, pero cambiaban los nombres de los recuadros dependiendo de las magnitudes que se manejaban en cada planta.

De manera de ejemplo se toma la interfaz del programa de control de velocidad, en la figura 6.44 se muestra la interfaz en donde trae una serie de botones, indicadores y campos de Entradas / Salidas. En la siguiente lista se va especificar cada una de las funciones que hace cada botón y campo de entrada:

- **Lectura Velocidad RPM:** En este campo de salida se puede visualizar la velocidad actual del motor en Revoluciones por minuto.
- **Setpoint RPM:** Es este campo de entrada/salida se puede ingresar el valor deseado, o visualizarlo en caso de que el modo automático esté en modo ON.
- **Frecuencia de 0 a 100:** Este campo de Entrada se puede dar la señal de control directa desde la HMI, sin embargo, todos los controles deben estar apagados.
- **Generar Doc:** Con este botón, se genera el archivo para capturar datos, el indicador de la parte derecha nos indica si el archivo se creó con éxito cuando se pone en verde.

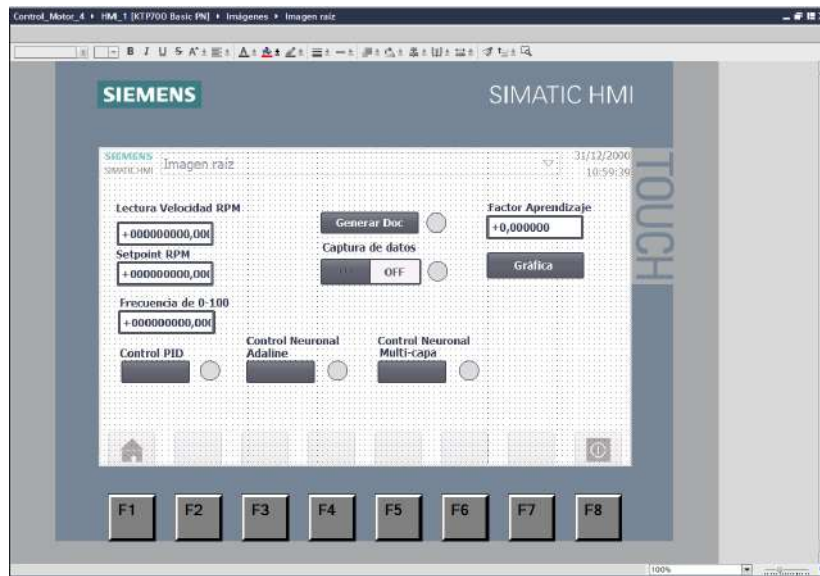


Figura 6.44: Interfaz de la HMI en el control de velocidad

Fuente: Autor

- **Captura de datos:** este interruptor nos habilita la captura de datos a una frecuencia inscrita dentro del programa, también habilita el bloque de generador automático de setpoints para tomar las datos en diferentes valores deseados.
- **Control PID:** Este pulsador habilita el bloque PID para realizar el control por este método, mientras esté activado ningún otro controlador se podrá activar, el indicador de la parte derecha nos informa si el bloque PID está activo.
- **Control Neuronal Adaline:** Este pulsador habilita el bloque de control neuronal adaline para realizar el control por este método, mientras esté activado ningún otro controlador se podrá activar, el indicador de la parte derecha nos informa si el bloque de control está activo.
- **Control Neuronal Multi-capa:** Este pulsador habilita el bloque de control neuronal multi-capa para realizar el control por este método, mientras esté activado ningún otro controlador se podrá activar, el indicador de la parte derecha nos informa si el bloque de control está activo.
- **Factor de Aprendizaje:** Este campo de entrada nos permite ingresar el factor de aprendizaje con el que trabaja tanto el bloque de control neuronal adaline o el bloque de control neuronal multi-capa, solo es de utilidad cuando alguno de los dos bloques anteriormente mencionados están activados.
- **Gráfica:** Este pulsador me redirige a otra imagen donde se puede visualizar en una gráfica el valor de lectura en este caso de velocidad en un intervalo de tiempo.

Para cada una de las plantas en las que se realizó el control, se le realizó una interfaz idéntica para poder interactuar de una manera fácil e intuitiva con el programa interno del autómeta.

7 Resultados

7.1 Aplicando controladores a la planta de velocidad

Se aplicaron los tres tipos de controladores en la planta de velocidad el PID, neuronal adaline y neuronal multi-capa, se muestra la respuesta de cada uno, y una comparativa de los tres. Los valores deseados en los que se evaluó la respuesta están en el siguiente orden 250 rpm, 1000 rpm, 1200 rpm, 1600 rpm, 900 rpm y 400 rpm.

La captura de los datos se realizó por medio del servidor web que trae por defecto el PLC S7-1200, el cual permite exportar los valores de una variable a un archivo ".csv" usando las instrucciones avanzadas de "Data logging" para crear el archivo y escribir en un determinado intervalo de tiempo, los datos se capturan como una base de datos que se abre con Excel y luego fueron exportados al software de MATLAB para su respectivo análisis.

7.1.1 Respuesta control PID

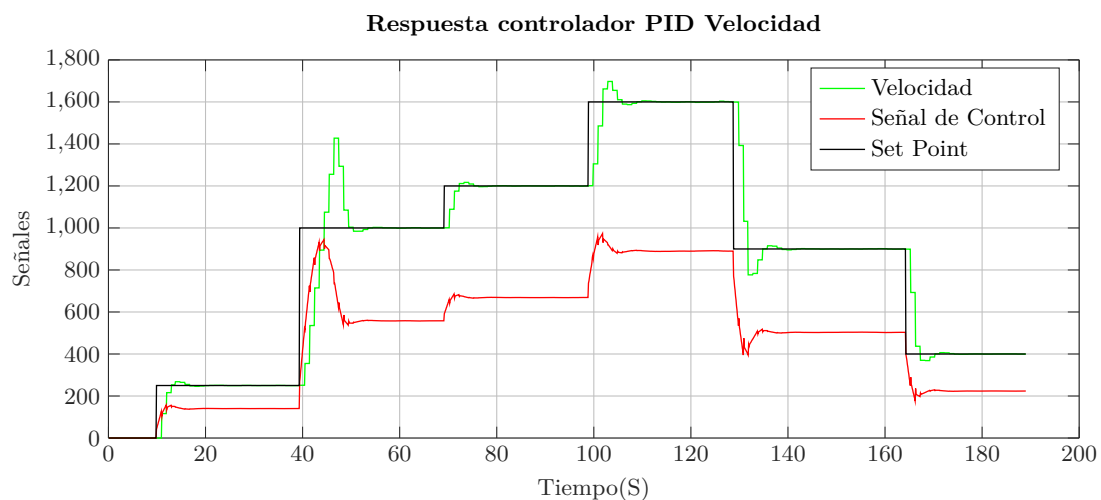


Figura 7.1: Respuesta PID en la planta de Velocidad

Fuente: Autor

Usando el sintonizador que trae por defecto el software de TIA PORTAL, se obtuvo las constantes K_p , K_i y K_d ver figura 6.27 con las que se obtuvo la siguiente respuesta 7.1, el valor deseado que se usó como referencia para sintonizar el controlador fue de 1000 rpm y el tiempo de barrido del ciclo interrupción en el que se ejecuta el bloque "PID Compact" fue de 0.01s o 10ms, ya que la dinámica de la planta es rápida. En la figura 7.1 se evidencia en verde la respuesta del motor, negro el valor deseado y en rojo la señal de control, para efectos

de una mejor visualización la señal de control se multiplico por un factor de 10, ya que está señal va de 0 a 100, en la gráfica se visualiza escalada de 0 a 1000.

La optimización fina se realizó varias veces para lograr reducir el sobre-pico, sin embargo fue el mejor resultado que se obtuvo, se debe recalcar que el motor usado es industrial por lo que para su arranque debe romper con una inercia que es considerable. La respuesta del control PID analizándola en el intervalo donde el valor deseado (Setpoint) cambia de 250rpm a 1000 rpm, donde es el cambio más grande del sistema se establece en un tiempo de 17.5s pero con un sobre-pico bastante notorio y significativo de aproximadamente 400 rpm sobre el valor deseado equivalente a un error aproximado de 22.2%, en pasos pequeños del valor deseado la velocidad aparenta solo tener un leve sobre-pico y cuando el cambio de valor deseado se establece en un valor menor, como en el intervalo de 1600 rpm a 900 rpm se detecta un pico en caída de aproximadamente 200 rpm equivalente a un 11% de error.

La figura 7.2 muestra la respuesta del controlador ante perturbaciones ocasionadas en el instrumento de medición, en este caso el encoder.

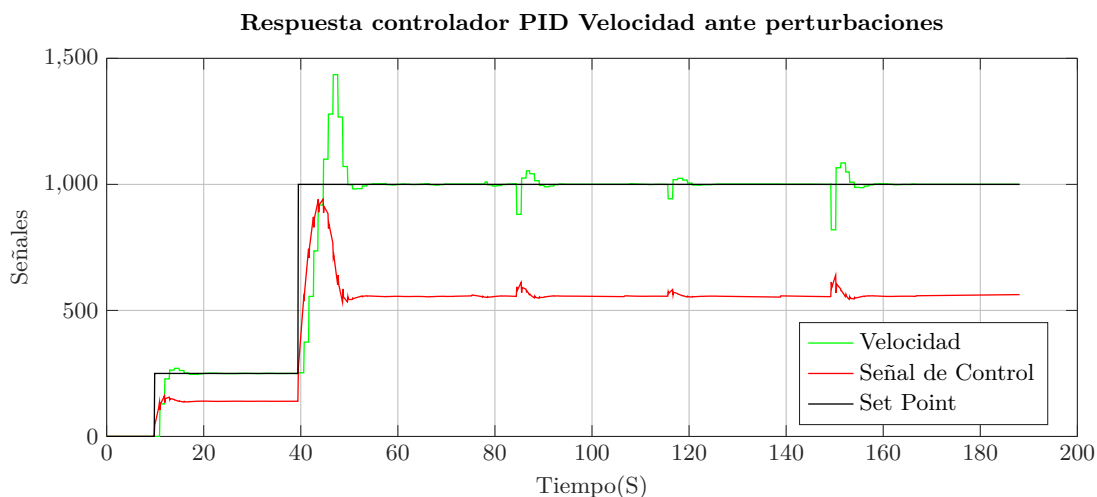


Figura 7.2: Respuesta controlador PID ante perturbaciones

Fuente: Autor

Se realiza tres perturbaciones en el valor deseado de 1000 rpm , que fue el valor en el que se sintonizo el controlador PID, una primera perturbación de 100 rpm en el intervalo de 80 s a 100 s el controlador hace que la velocidad se recupera de una manera muy rápida y con un sobre-pico no muy grande que es tolerable dependiendo de la aplicación, en la segunda perturbación de 50 rpm en el intervalo de 100 s a 120 s, el controlador responde de una manera adecuada haciendo que la velocidad de salida tenga un leve sobre-pico y para la ultima perturbación en el intervalo de 140 s a 160 s cuyo valor es aproximadamente 200 rpm el controlador genera un sobre-pico notorio en la velocidad del motor, pero se recupera en un intervalo pequeño.

este fenómeno hace que al arrancar la señal de control pueda ser más grande ocasionando el sobre-pico.

7.1.2 Respuesta control neuronal adaline

Aplicando el algoritmo de control neuronal ver fragmento de código 5.3 inmerso en el bloque de función ver figura 6.36 se obtuvo la siguiente respuesta ver figura 7.3 en donde se le ingreso un factor o tasa de aprendizaje de 0.00135. El bloque se ejecuto en el OB30 o ciclo de interrupción de 10ms configurado también para el PID.

La respuesta de velocidad del sistema se aprecia de color "verde", el valor deseado en color "negro" y la señal de control en "rojo", es importante mencionar que la señal de control en el controlador neuronal adaline tiene una resolución de salida de 0 a 1000, por lo tanto en la figura 7.3 a diferencia de la respuesta del PID no está escalada.

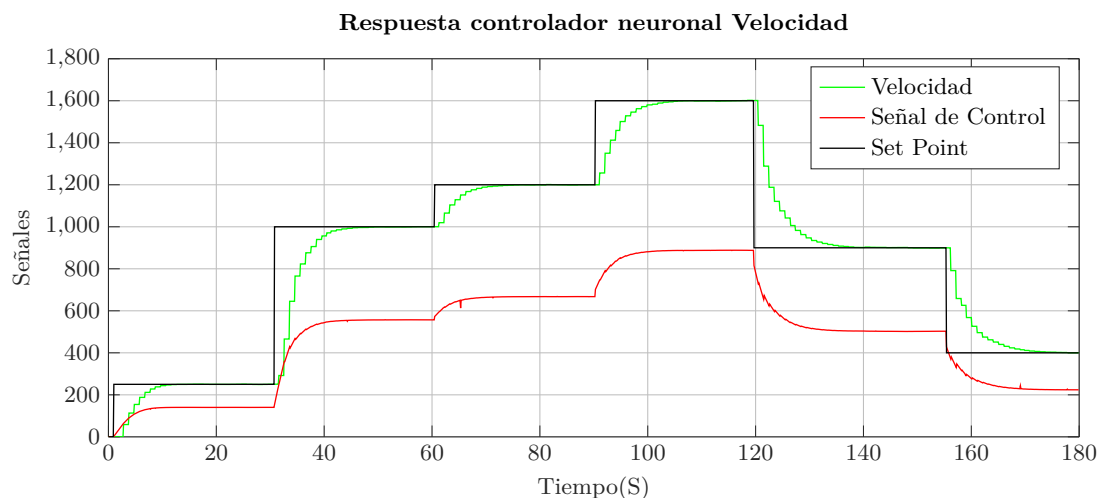


Figura 7.3: Respuesta control neuronal adaline velocidad $Fa=0.00135$

Fuente: Autor

Es una respuesta muy estable en todos los cambios de valor deseado, para efectos del análisis se toma el intervalo de cambio del valor deseado entre 250 rpm a 1000 rpm, no hay sobre-pico del valor de velocidad y se estabiliza en un tiempo de 17.7 s, de igual manera en los demás cambios del valor deseado la velocidad no genera ningún sobre-pico, en el intervalo de bajada entre el cambio de setpoint de 1600 rpm a 900 rpm tampoco se presenta un pico de bajada, lo que hace que sea una respuesta muy estable y segura para la planta.

El factor de aprendizaje de 0.00135 se halló de manera empírica, es decir, observando el comportamiento de la planta y variándolo. El factor de aprendizaje es una variable modificada por el usuario que ingresa al sistema y dicta la respuesta de la planta, dicho de otro modo, es el factor que hace que el sistema responda más rápido o más lento, entonces con ese parámetro podemos hacer que la planta se establezca en un tiempo menor, sin embargo, cuando se quiere un tiempo de respuesta muy corto, el sistema tiende a tener sobre-picos, la figura 7.4 nos muestra la influencia del factor de aprendizaje en la respuesta de velocidad del sistema, según la teoría planteada en la sección 5.3

En la figura 7.4 se observa cinco diferentes respuestas cada uno con un factor de aprendizaje diferente, el factor que se escogió como referencia es el de 0.00135 que es el de la figura 7.3, se escogieron dos valores por encima de este factor y dos por debajo.

Los factores de aprendizaje escogidos por debajo del de referencia fueron de 0.00100 y de 0.00125, que aparecen de color "rojo" y "azul turquesa" respectivamente en la figura 7.4, el comportamiento esperado del sistema con unos factores de aprendizaje más pequeños es que la respuesta de velocidad le tome más tiempo en llegar al valor deseado, es decir, que el tiempo de estabilización sea más largo, en la figura se puede evidenciar que efectivamente la respuesta del sistema con el factor de aprendizaje de 0.00100 le toma más tiempo en estabilizarse aproximadamente 27.2 s, mientras que la de factor de aprendizaje de 0.00125 por ser tan cercana al valor de referencia son casi idénticas, la diferencia entre las dos factores más pequeños y la de referencia se pueden apreciar mejor en la figura 7.5 que equivale a un zoom de la figura 7.4 en el setpoint de 1000 rpm. Los factores escogidos por arriba del

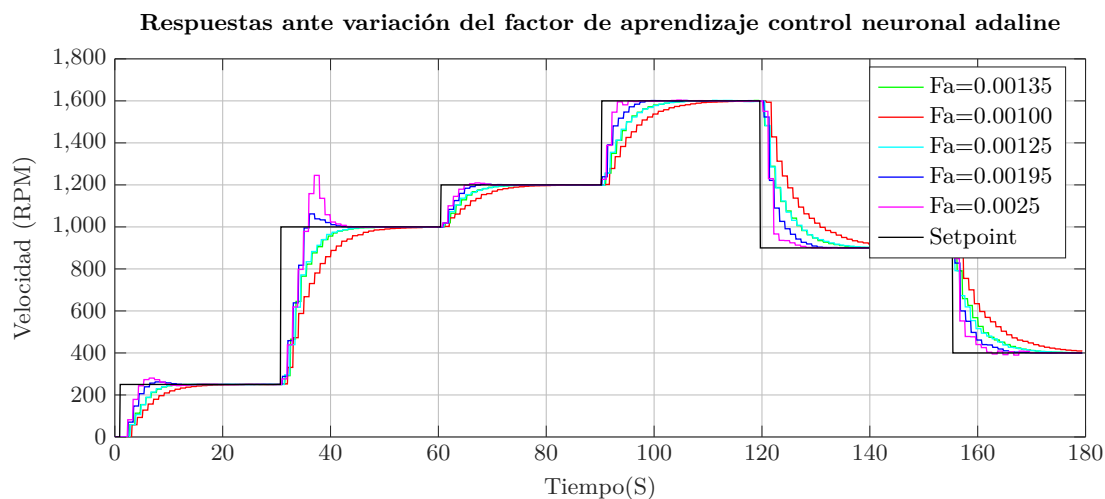


Figura 7.4: Respuestas ante variación del factor de aprendizaje control neuronal adaline

Fuente: Autor

de referencia fueron 0.00195 y 0.00125, que aparecen de color "azul oscuro" y "morado" respectivamente en la figura 7.4, el comportamiento esperado es lo contrario a los que están por debajo, por lo tanto, se espera que el sistema responda más rápido, en otras palabras, el tiempo de estabilización es más corto, en la figura se ratifica lo esperado ya que el factor de aprendizaje de 0.00195 responde más rápido, se estabiliza alrededor de 12.3 s, aunque, empieza a haber un ligero sobre-pico de aproximadamente 60 rpm equivalente a un error del 3.33% que es un error bastante pequeño, en tanto que la respuesta de velocidad con el factor de aprendizaje de 0.0025 hay un notorio sobre-pico de aproximadamente 245 rpm que equivale a un error del 13.61% que es bastante grande, a pesar de este sobre-pico el sistema se estabiliza en casi el mismo tiempo 12.3s, por lo tanto, se evidencia la influencia del factor de aprendizaje en el tiempo de respuesta de la planta y también en la estabilidad al momento de la respuesta, se puede apreciar mejor la diferencia entre los sobre-picos en la figura 7.5.

En la figura 7.6 se ilustra la respuesta del controlador neuronal adaline ante perturbaciones causadas al sensor. en el intervalo de 60s a 80s se realiza una perturbación de 120 rpm aproximadamente, el controlador responde muy rápido y provoca que la respuesta tenga un leve sobre-pico de 24 rpm que equivale a un error del 1.33%, indicando que es un sobre-pico muy pequeño, la siguiente perturbación en el intervalo de 100s a 120s tiene un valor de

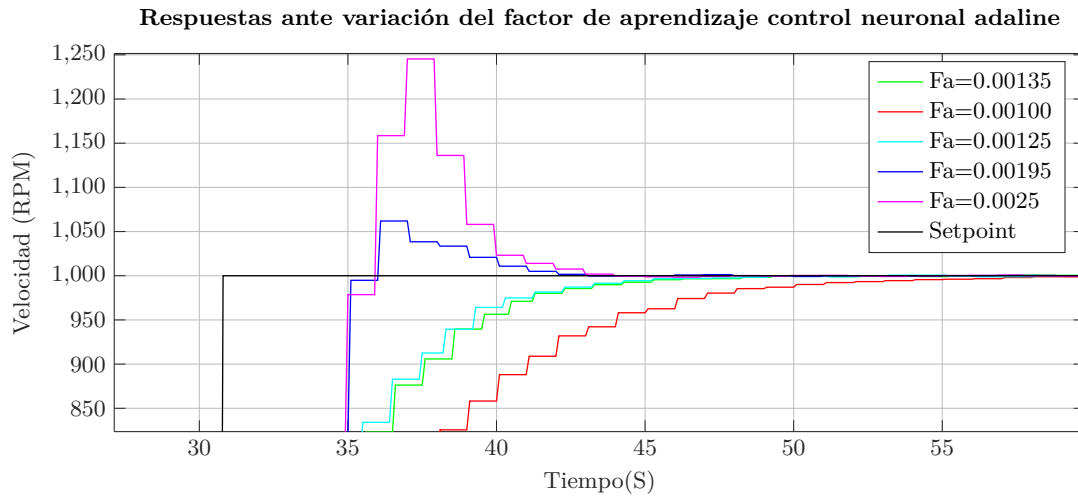


Figura 7.5: Respuestas ante variación del factor de aprendizaje control neuronal adaline ampliada

Fuente: Autor

60 rpm y no representa un problema para el controlador ya que se establece rápidamente y genera un pico de 13 rpm que representa un error de 0.722% que es tolerable, la última perturbación realizada en el sistema corresponde al valor de 200 rpm aproximadamente, donde el controlador tiene un poco pronunciado con un valor de 52 rpm, es decir, un error del 2.8%. En general la respuesta del control adaline es muy estable y sus sobre-picos son mínimos.

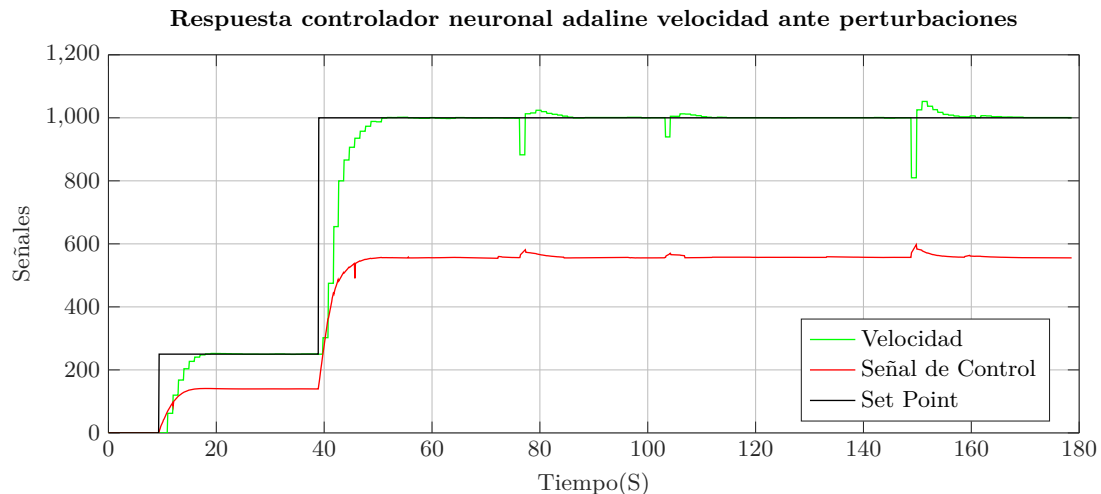


Figura 7.6: Respuestas control neuronal adaline $Fa=0.00135$ ante perturbaciones

Fuente: Autor

7.1.3 Respuesta controlador neuronal multi-capas

El algoritmo de control multi-capas que se desarrolló en un bloque de función ver figura 6.41, que cuenta con la topología de una capa de entrada de 3 neuronas, una capa oculta

de tres neuronas y una capa salida con una neurona ver figura 6.40, se aplico en la planta de velocidad la figura 7.7 muestra la respuesta del sistema con el controlador multi-capas, el factor seleccionado es de 0.071, además, el bloque de función al igual que el PID y el neuronal adaline se ejecuta en el ciclo de interrupción OB30 establecido en 10ms.

La respuesta de velocidad se puede apreciar de color "verde", el valor deseado o setpoint se ilustra de color "negro" y la señal de control de color "rojo", en la figura 7.7 la señal de salida del bloque de control multi-capas es de 0 a 100 a diferencia del adaline, por lo tanto, en la figura 7.7 se encuentra escalada por un factor de 10 para que se pueda apreciar mejor en una escala de 0 a 1000. En la figura se evidencia una respuesta muy estable que es muy parecida

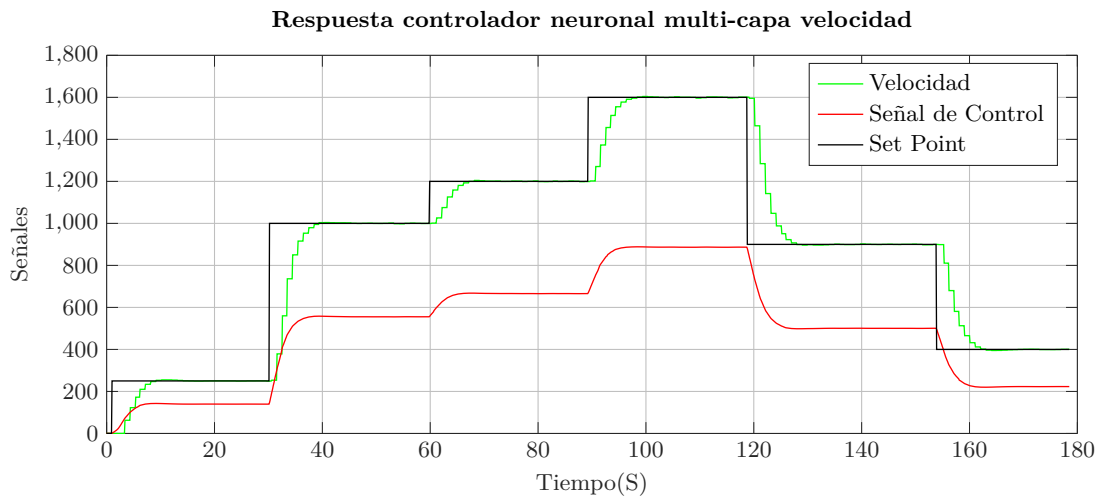


Figura 7.7: Respuestas control neuronal multi-capas $Fa=0.071$

Fuente: Autor

al de controlador neuronal adaline, sin embargo, esta se establece un poco más rápido, en el cambio de valor deseado de 250 rpm a 1000 rpm la respuesta de la velocidad se estabiliza en el valor deseado en 9.2 s, realizando un zoom se evidencia un leve sobre-pico de 4 rpm que es muy insignificante, en el intervalo donde el valor deseado cambia de 1600 rpm a 900 rpm la velocidad se establece en un tiempo de 9.2s, dicho de otra forma, el controlador tiene un tiempo de estabilización pequeño.

Para buscar el factor de aprendizaje adecuado en este caso el de 0.0710 se realizó empíricamente, observando el comportamiento del neuro-controlador en tiempo real, iterando con diferentes valores se llegó a un valor que genera una respuesta óptima, esta iteración se realizaba por medio de la interfaz de usuario conectada al autómata.

De la misma forma que el controlador neuronal adaline, el controlador neuronal multi-capas también varía el tiempo de respuesta al modificar el factor de aprendizaje, se maneja el mismo principio que el adaline, la teoría del multi-capas está contemplada en la sección 5.6. Para evaluar la teoría con la práctica se sometió el sistema de control de velocidad a cinco factores de aprendizaje diferentes, como referencia se escogió el factor de aprendizaje ilustrado en la figura 7.7 que equivale a 0.0710, por lo tanto se escogieron dos por arriba y dos por abajo. Los factores de aprendizaje por debajo del de referencia fueron 0.062 y 0.065 que aparecen de color "rojo" y "azul turquesa" respectivamente en la figura 7.8, la respuesta de velocidad con

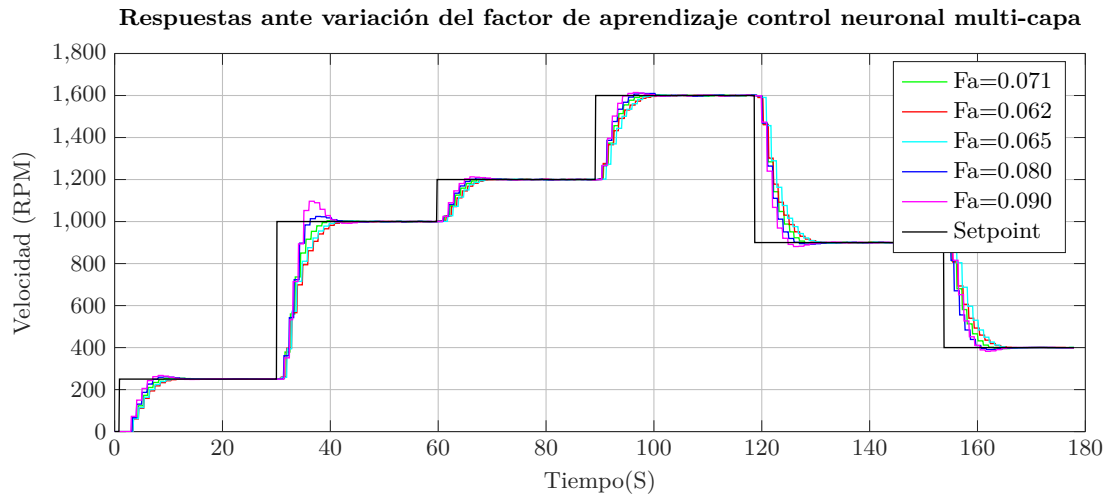


Figura 7.8: Respuestas ante variación del factor de aprendizaje control neuronal multi-cap

Fuente: Autor

estos factores de aprendizaje es un poco más demorada que la de factor de referencia, para ser exacto con el factor de 0.062 la respuesta se estabiliza en 14.1s sin sobre-picos, mientras que la de 0.065 se estabiliza en un tiempo de 12.6s sin sobre-picos, este tiempo de estabilización se tomo como referencia el cambio de valor deseado de 250 rpm a 1000rpm. En la figura 7.9 se amplía la imagen para visualizar un poco mejor la diferencia entre cada señal. Los factores de aprendizaje por encima del de referencia fueron 0.080 y 0.090 los cuales se

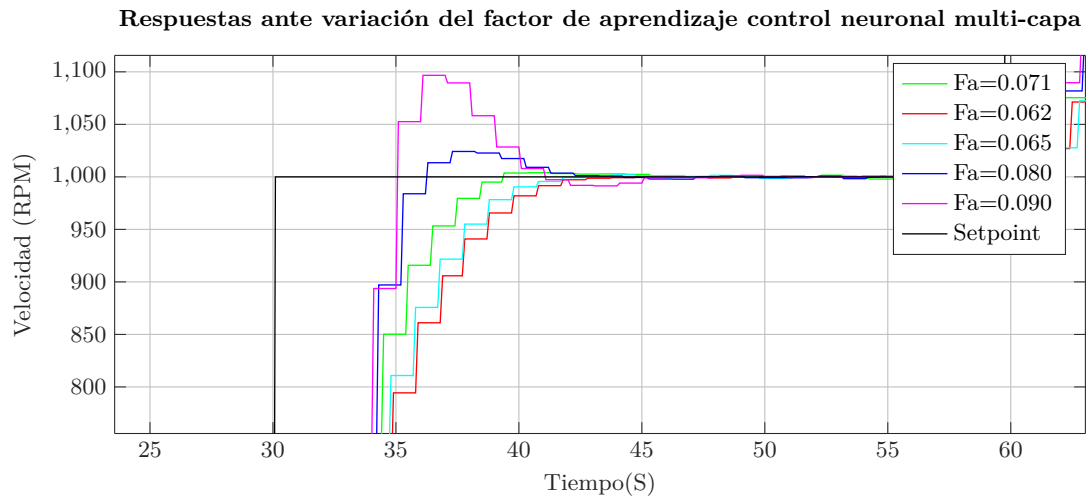


Figura 7.9: Respuestas ante variación del factor de aprendizaje control neuronal multi-cap ampliada

Fuente: Autor

aprecian de color "azul oscuro" y "morado" respectivamente, las respuesta de velocidad con este factor de aprendizaje aparenta ser mas rápida, pero si se observa bien en la figura, 7.9 se estabiliza en un tiempo más grande que el de referencia, para el factor de 0.080 el valor

se estabiliza en aproximadamente 12.1s con un sobre-pico leve equivalente a 24 rpm que en porcentaje de error es 1.33%, mientras que la respuesta con el factor de 0.090 se estabiliza en un tiempo de 10.9s, sin embargo cuenta con un sobre-pico bastante notorio que equivale a 97 rpm aproximadamente un error de 5.33%.

En la figura 7.10 se muestra la respuesta de controlador multi-capa ante tres perturbaciones ocasionadas en el sensor de medición.

La primera perturbación en el intervalo de 60s a 70s con un valor de 68 rpm no es mucho problema para el control, ya que vuelve y se estabiliza en un corto lapso de tiempo, sin embargo, genera un sobre-pico de aproximadamente 15 rpm por encima del valor deseado aproximadamente un error de 0.83%; la segunda perturbación ocasionada en el lapso de 100s a 110s con un valor de 95 rpm donde el controlador vuelve y se restablece de una manera rápida, pero, generando un sobre-pico de 22 rpm o del 1.2%; la tercera perturbación ocasionada en el intervalo de 130s a 150s con el valor más grande, aproximadamente de 800 rpm genera un sobre-pico de 74 rpm o del 4.1%, no obstante, se recupera en un tiempo bastante corto.

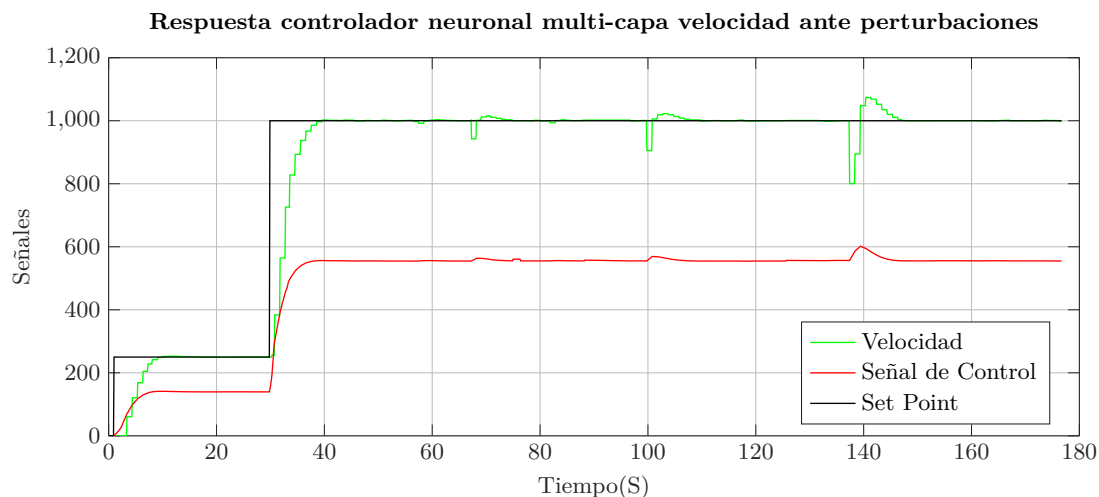


Figura 7.10: Respuestas control neuronal multi-capa $F_a=0.071$ ante perturbaciones

Fuente: Autor

7.1.4 Comparación de las técnicas de control planta de velocidad

En las anteriores sub-secciones se analizaron los controladores en la misma planta pero cada uno por separado, en este apartado se comparan las respuestas de los controladores en una sola figura para poder realizar la comparativa y análisis de la mejor respuesta obtenida. La figura 7.11 contiene la respuesta de los controladores mencionados anteriormente, el controlador PID "verde", controlador neuronal adaline "rojo", controlador neuronal multi-capa "azul" y el setpoint de "negro". Las señales están sincronizadas, es decir, el cambio de setpoint se realizó en el mismo instante tiempo para cada una de los controladores. La figura 7.12 se puede ver un poco mejor la diferencia entre cada una de las respuestas de los controladores, esta parte de la figura 7.11, corresponde al tramo en donde el valor deseado realiza el cambio de 250 rpm a 1000 rpm.

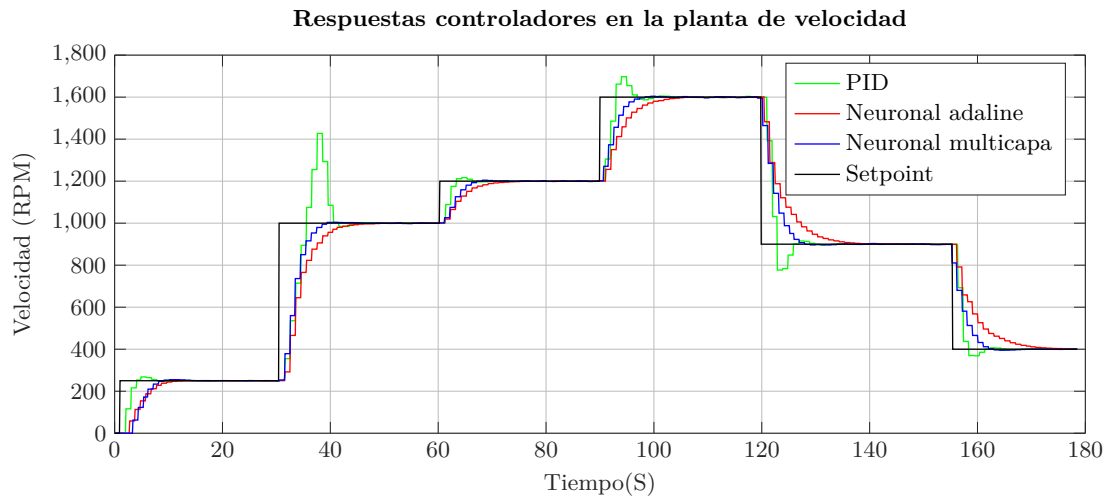


Figura 7.11: Respuestas controladores en la planta de velocidad

Fuente: Autor

El controlador PID sube bastante rápido pero genera mucho sobre-pico debido a esto le toma un tiempo considerable estabilizarse en el valor de 1000 rpm aproximadamente 15.3 s, el control neuronal adaline sube lentamente pero sin sobre-pico y aproximadamente se estabiliza en un tiempo de 15.9 s, mientras que el neuronal multi-capas sube casi igual de rápido que el PID, genera un sobre-pico diminuto, tan pequeño que se puede considerar que en ese punto ya llega al setpoint con un tiempo de 9.2 s.

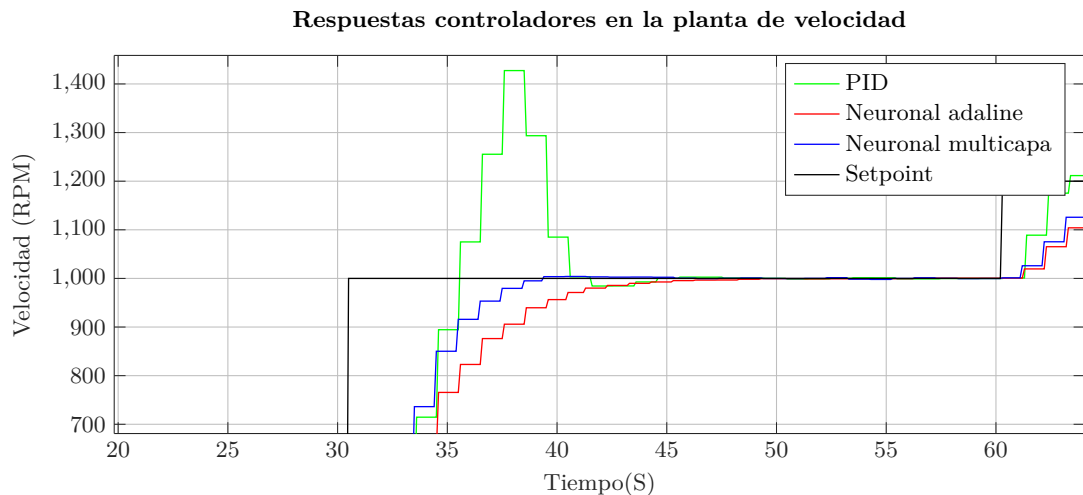


Figura 7.12: Respuestas controladores en la planta de velocidad ampliado setpoint 1000 rpm

Fuente: Autor

La tabla 7.1 muestra el tiempo de estabilización de cada uno de los controladores en el setpoint de 1000 rpm. En la figura 7.13 se muestra ampliada una parte de la figura 7.11, exactamente en el setpoint de 900 rpm para analizar el comportamiento de cada uno de los

Controlador	Tiempo de estabilización en segundos
PID	15.3
Neuronal Adaline	15.9
Neuronal Multi-capa	9.2

Tabla 7.1: Tiempo de estabilización de los controladores en el setpoint de 1000 rpm (subida)

controladores en bajada.

El controlador PID hace que la velocidad baje bastante rapido pero igual que en subida genera un sobre-pico pero en este caso hacia abajo, se estabiliza en un tiempo de 10.6 s, el controlador neuronal adaline hace que la velocidad disminuya lentamente pero se establece sin ninguna clase de sobre-pico, lo que le toma un tiempo de 18.3s, mientras que el controlador neuronal multi-capa hace que la velocidad llegue en un intervalo medio de los dos anteriormente mencionados, a diferencia del PID este no genera sobre-picos lo que hace que se estabilice en el tiempo de 9 s.

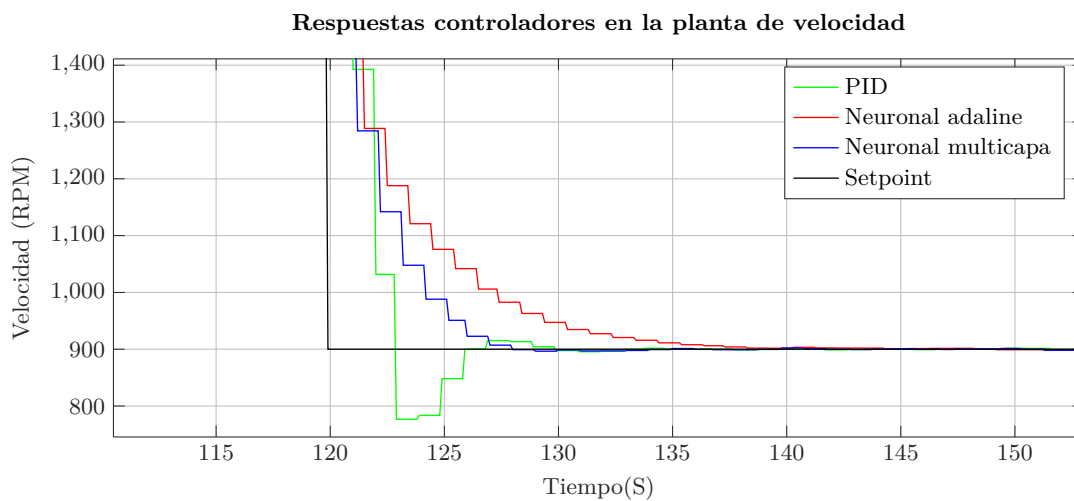


Figura 7.13: Respuestas controladores en la planta de velocidad ampliado setpoint 900rpm

Fuente: Autor

Controlador	Tiempo de estabilización en segundos
PID	10.6
Neuronal Adaline	18.3
Neuronal Multi-capa	9

Tabla 7.2: Tiempo de estabilización de los controladores en el setpoint de 900 rpm (bajada)

7.2 Aplicando controladores en la planta de temperatura

Se aplicaron los tres tipos de controladores en la planta de temperatura: PID, neuronal adaline, neuronal multi-capas; en tres setpoints para mirar el comportamiento de los controles en el sistema que son: 60°C, 90°C y 75°C. El tiempo de barrido del ciclo de interrupción OB30 usado en la planta de temperatura fue de 500ms para el PID y adaline, mientras que con la multi-capas fue 100ms. Los datos se capturaron de la misma forma que la de velocidad por medio del servidor web.

7.2.1 Respuesta controlador PI

Se utilizó el sintonizador que trae por defecto TIA PORTAL, las constantes generadas se pueden ver en la figura 6.30, la respuesta de la temperatura se evidencia en la figura 7.14, la sintonización se realizó en el setpoint de 60 °C ver figura 6.29. La señal de control que normalmente es de 0 a 100 en la figura 7.14 se multiplicó por un factor de 0.2 para escalarla de 0 a 20, para poder realizar un mejor análisis. La respuesta de temperatura se puede ver de

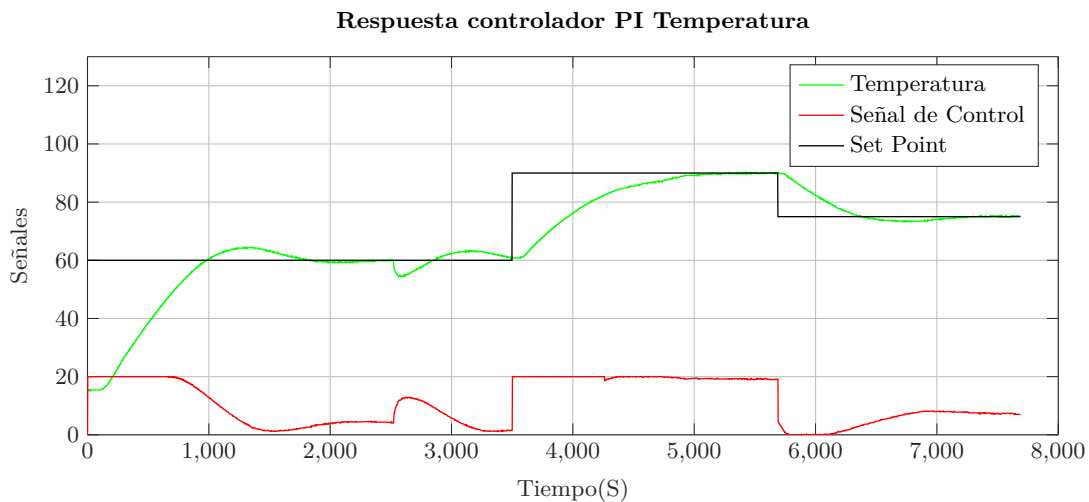


Figura 7.14: Respuesta controlador PI en la planta de temperatura

Fuente: Autor

color "verde", la señal de control de "rojo" y los diferentes setpoints de "negro". La dinámica del sistema es muy lenta ya que es una variable que sufre cambios en un intervalo grande de tiempo, en el primer setpoint de 60°C la temperatura se establece en un tiempo de 2480 s o 41.33m, alcanza un sobre-pico equivalente a 4°C o de 3.3% por encima del valor deseado, en el intervalo de 2400s a 2700 se evidencia una perturbación del sistema de aproximadamente 6°C en donde el controlador hace que la temperatura se incremente de nuevo y se establezca, para el setpoint de 90°C a la planta le cuesta llegar, debido a que el líquido en este caso agua, comienza a llegar a su estado sobre-calentada, por lo tanto el tiempo de respuesta es demorado aproximadamente de 1932s o 32.2min, sin embargo no genera sobre pico, en el último cambio de valor deseado la temperatura baja hasta 75°C y se establece en un tiempo de 1786s o 29.76 min, genera un ligero sobre-pico de 1.56°C o 1.3%.

7.2.2 Respuesta controlador neuronal adaline

Para el control adaline de la planta de temperatura se estableció una señal de salida entre 0 y 50 como se ve en la figura 6.38, además se ingreso un factor de aprendizaje de valor 0.0015, en la figura 7.15, la temperatura se muestra de color "verde", la señal de control en color "rojo" y el valor deseado en color "negro". La señal de control de salida del bloque tiene una resolución de 0 a 50, para efectos de una mejor vista la señal se multiplico por un factor de 0.4, es decir, que la señal de control en la figura 7.15 está escalada de 0 a 20.

En el intervalo de 60°C el control adaline en el sistema de temperatura no se establece en el valor deseado, se aprecia un sobre-pico pequeño en este caso de 3.9°C o 3.16%, luego de alcanzar este sobre-pico, la respuesta del sistema queda oscilando alrededor del valor deseado, lo que causa que no se establezca, en el intervalo de 2000s a 3000s se evidencia una perturbación donde el controlador responde de una manera satisfactoria, sin embargo no se establece en el valor deseado, en el intervalo de 90°C pasa lo mismo, al alcanzar la variable deseada de 90°C la temperatura empieza a oscilar, generando que la temperatura no se establezca en un punto fijo y en el último intervalo donde la temperatura desciende a 75°C no es la excepción ya que se repite el mismo comportamiento oscilatorio, el control neuronal en la planta de temperatura no se logra establecer.

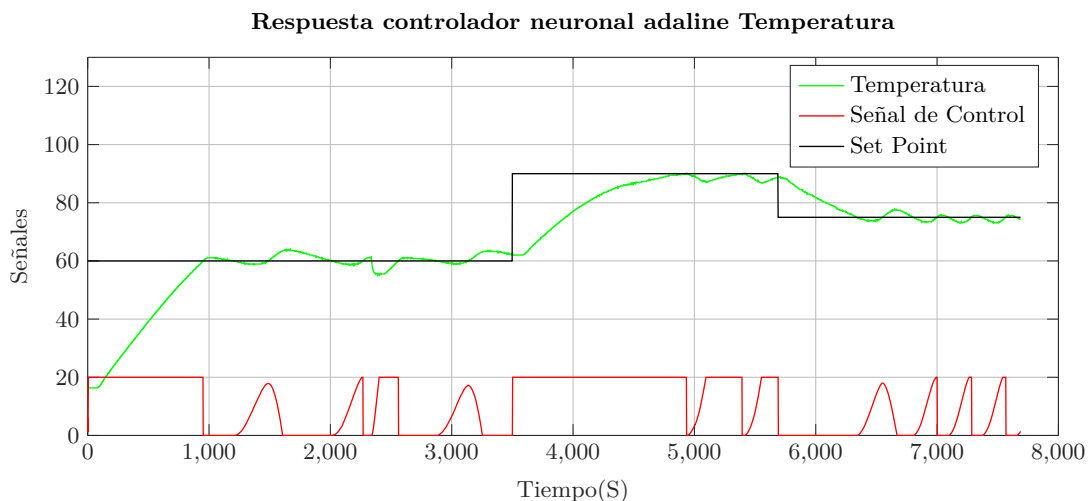


Figura 7.15: Respuesta controlador neuronal adaline en la planta de temperatura $F_a=0.0015$

Fuente: Autor

7.2.3 Respuesta controlador neuronal multi-capa

Aplicando el bloque de función controlador neuronal multi-capa ver figura 6.42, se obtuvo la siguiente respuesta ver figura 7.16, donde se ilustra la señal de temperatura color "verde" la respuesta del controlador o señal de control de color "rojo" y el valor deseado color "negro", la señal de control se estableció en un rango de 0 a 100 y el factor de aprendizaje utilizado es 0.28. Para efectos de una mejor visualización la señal de control se multiplico por un factor de 0.2, en la figura 7.16 la señal se encuentra escalada de 0 a 20.

En el primer valor deseado de 60°C el controlador hace que la temperatura se establezca en un

valor cercano al valor deseado, pero permanece oscilatorio, oscilaciones mucho más pequeñas que las generadas por el controlador adaline, esta oscilaciones van de 2°C por arriba y 2°C por abajo, pero según el comportamiento de la temperatura indica que las oscilaciones van disminuyendo poco a poco, mientras que en el intervalo de 90°C se establece aproximadamente en 1670 s o 27.83 min, sin embargo, si se realiza un zoom a la figura se puede ver un ligero error en estado estacionario equivalente a 0.72°C que es muy pequeño, en el setpoint de 75°C la temperatura queda oscilando semejante a lo que pasa en el intervalo de 60°C , pero en un rango de 0.60°C por arriba y 2°C por abajo, se puede deducir que de cierta forma el controlador asemeja un comportamiento de un controlador ON-OFF por los picos de la señal de control que van de 0 a 100, pero es importante recalcar que la planta es inestable a lazo abierto, por lo cual este comportamiento es algo que se puede esperar.

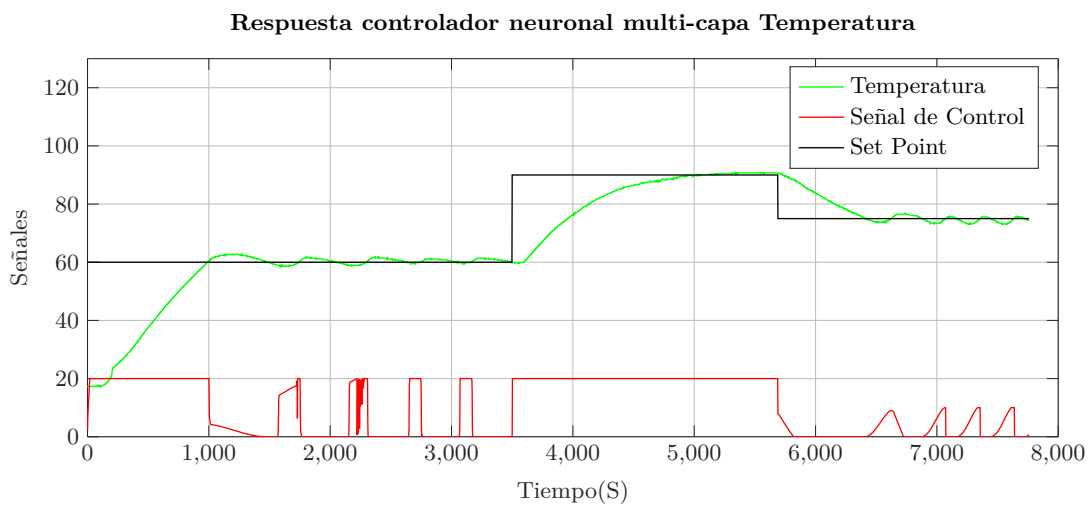


Figura 7.16: Respuesta controlador neuronal multi-capa en la planta de temperatura $F_a=0.28$

Fuente: Autor

7.2.4 Comparación de las técnicas de control planta de temperatura

En las anteriores sub-secciones se analizaron los controladores en la misma planta pero cada uno por separado, en este apartado se comparan las respuestas de los controladores en una sola figura para poder realizar la comparativa y análisis de la mejor respuesta obtenida. La figura 7.17 contiene la respuesta de los controladores mencionados anteriormente, el controlador PID "verde", controlador neuronal adaline "rojo", controlador neuronal multi-capa "azul" y el setpoint de "negro". Las señales están sincronizadas, es decir, el cambio de setpoint se realizó en el mismo instante tiempo para cada una de los controladores.

Analizar el tiempo de respuesta de esta planta no nos da mucha información para comparar los controladores ya que tiene una dinámica muy lenta y además los dos controladores neuronales tienen oscilaciones, el PID presenta un comportamiento aceptable donde tiene sobre-pico de 4.43°C o 3.69%, pero, la señal comienza a tener después de un tiempo al valor deseado, cosa que en el control neuronal adaline no sucede, la respuesta del control neuronal adaline presenta unas oscilaciones cerca al valor deseado, algunas con amplitud de 3.58°C por

arriba y 2.3°C por debajo y no muestra señales de que se establezca en un valor fijo, mientras que el control neuronal multi-capa tiene una respuesta con oscilaciones pero la magnitud es más pequeña aproximadamente de 1.5°C y parece ir disminuyendo con el tiempo.

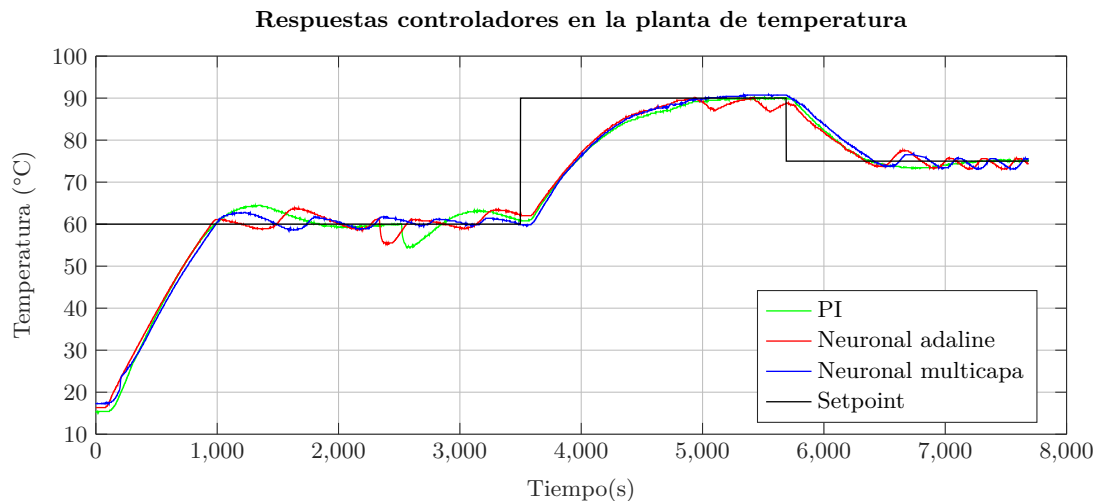


Figura 7.17: Respuestas controladores en la planta de temperatura

Fuente: Autor

En esta planta el control PID tiene una mejor respuesta, ya que se estabiliza en el valor deseado a diferencia de los otros dos controles.

7.3 Aplicando controladores en la planta de flujo

Se aplicaron dos tipos de control a la planta de flujo, el PID y el multi-capa, se muestra la respuesta de cada controlador en secciones y al final una comparativa. Los valores en los que se evaluó la respuesta fueron en el siguiente orden 4 l/min, 6 l/min y 8 l/min. La captura de los datos también se realizó por medio del servidor web, el tiempo de ejecución del ciclo de interrupción (OB30) en los que se llamaron los bloques es de 10 ms. Para efecto de una mejor visualización en los dos controladores la señal de control se dividió entre un factor de 10, ya que la señal tiene una resolución de salida 0 a 100 por lo que no se alcanzaría a apreciar bien las otras señales.

7.3.1 Respuesta controlador PI

Con los parámetros generados por el sintonizador que trae por defecto el software de TIA PORTAL ver figura 6.33, se obtuvo la respuesta de la figura 7.18, el valor en el que se sintonizó el controlador fue 6 l/min ver figura 6.32. La respuesta del flujo se puede ver de color "verde", la señal de control de "rojo" y los diferentes setpoints de "negro". El control PID hace que el flujo aumente bastante rápido al inicio con un ligero sobre-pico de 0.36 l/min o 3,6% pero rápidamente se estabiliza en la señal deseada, en un tiempo aproximado de 40 s, en el valor deseado de 6 l/min el controlador hace que el caudal incremente bastante rápido generando un sobre-pico de aproximadamente 0.90 l/min o del 9.0% pero se estabiliza en un tiempo

de 33.5 s, para el ultimo valor deseado, el flujo tiene un sobre-pico de 0.4 l/min o del 4.0%, estabilizándose en un tiempo de 58.5 s.

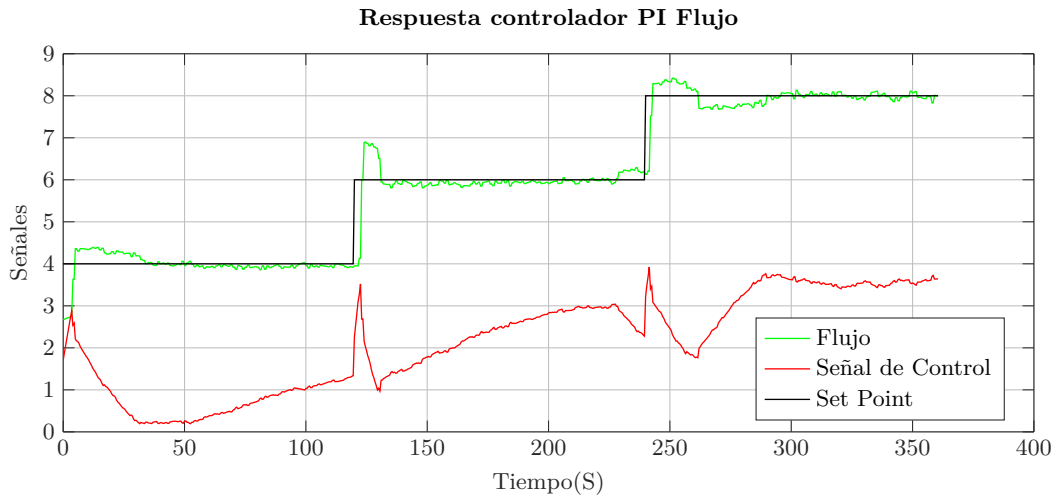


Figura 7.18: Respuesta controlador PI en la planta de flujo

Fuente: Autor

7.3.2 Respuesta controlador neuronal multi-cap

Utilizando el bloque de control neuronal multi-cap ver figura 6.43 se obtuvo la respuesta de la figura 7.19. El factor de aprendizaje utilizado es de 0.01.

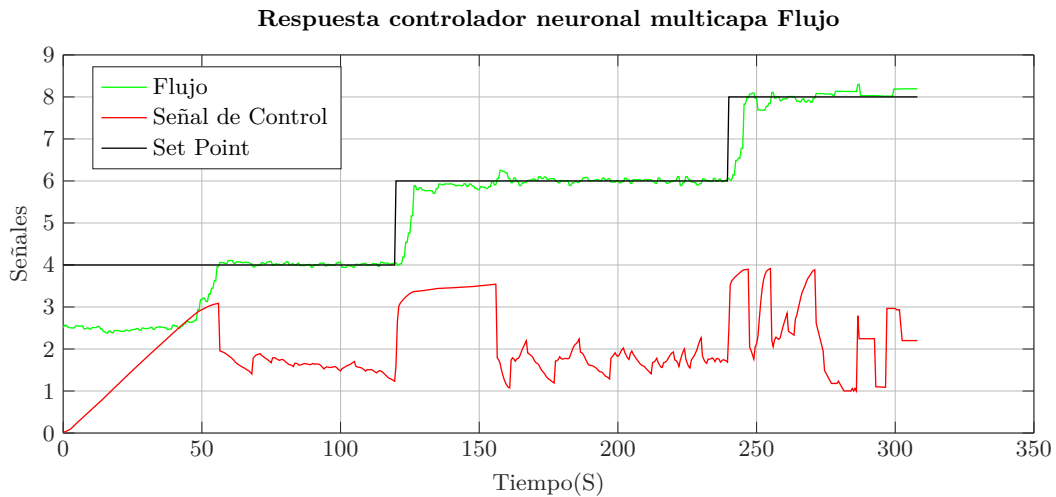


Figura 7.19: Respuesta controlador neuronal multi-cap en la planta de flujo $Fa=0.01$

Fuente: Autor

La respuesta del flujo generada por el controlador neuronal multi-cap en el primer instante es poco lenta, pero al estabilizarse no genera ningún sobre-pico, el tiempo que le lleva alcanzar

el valor deseado es de aproximadamente de 55,5 s, para el segundo valor deseado el tiempo de estabilización es de 40 s en el tercer valor deseado de 8 l/mi el flujo tiene unas oscilaciones pero se estabiliza aproximadamente en 34.5s.

7.3.3 Comparación de las técnicas de control en la planta de flujo

En las dos anteriores sub-secciones se analizaron los controladores en la misma planta pero cada uno por separado, en este apartado se comparan las respuestas de los controladores en una sola figura para poder realizar la comparativa y análisis de la mejor respuesta obtenida. La figura 7.20 contiene la respuesta de los controladores mencionados anteriormente, el controlador PID "verde", controlador neuronal multi-capa "rojo" y el setpoint de "negro". Las señales están sincronizadas, es decir, el cambio de setpoint se realizo en el mismo instante tiempo para cada una de los controladores.

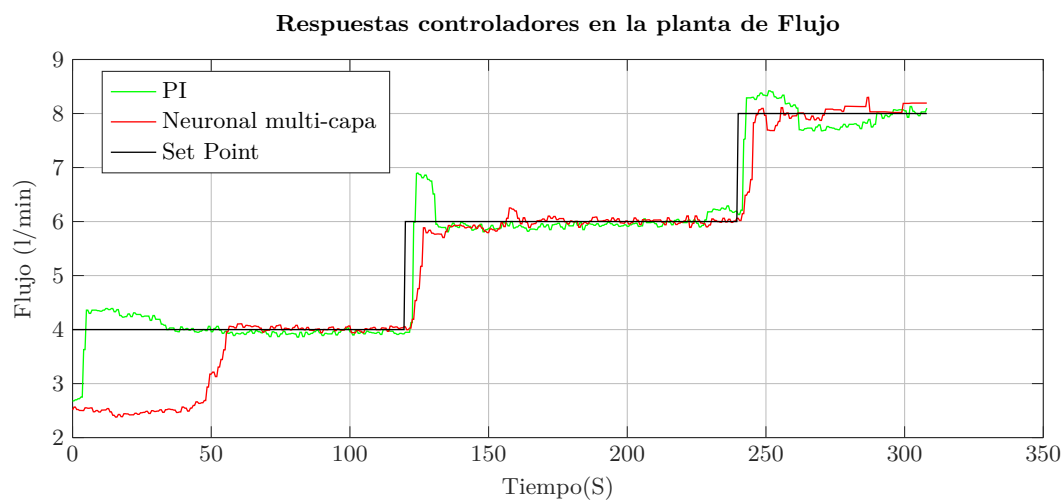


Figura 7.20: Respuestas controladores en la planta de flujo

Fuente: Autor

Al analizar la figura 7.20, se puede deducir que el controlador PID responde mas rápido en el primer setpoint de 4 l/min, sin embargo, en los siguientes setpoints el controlador neuronal multi-capa responde en el mismo intervalo de tiempo que el PID y sin sobre-picos lo que hace la respuesta más estable, entonces, el controlador neuronal multi-capa es una buena opción en este sistema, si se quiere obtener una respuesta sin sobre-picos.

7.4 Librerías generadas

Se presentan las librerías que se generaron como resultado de la ejecución del proyecto, las librerías están generadas en SCL como código fuente.

7.4.1 Librería para la lectura de velocidad por encoder rotativo incremental

Es un bloque de función que por medio de los contadores rápidos que trae el autómata lee los pulsos provenientes del encoder para calcular la velocidad con un margen de error del 0.11%, este bloque hace uso de funciones avanzadas del software TIA PORTAL pero que al momento de estar inmersas dentro de la librería el usuario solo tendrá que hacer el llamado en el main e ingresar 2 parámetros y podrá hacer la lectura.

Parámetros de entrada y salida

- **Resolución:** Es el numero de pulsos por vuelta que trae el encoder a utilizar.
- **HSC:** Es la dirección del contador rápido en el que se esta haciendo la lectura de pulsos.
- **Overflow:** Indica que el dispositivo está en reposo, es decir, la medición está en 0.0.
- **RPM:** Velocidad medida por el encoder.

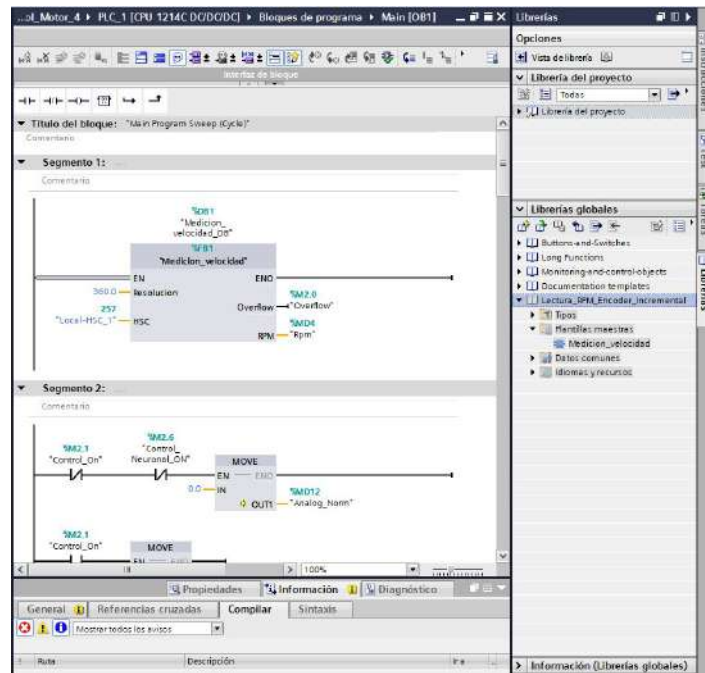


Figura 7.21: Librería Lectura de encoder incremental

Fuente: Autor

7.4.2 Librería para la captura de datos por medio del servidor web de Siemens S/-1200 CPU1214c DC/CD/CD

Esta librería permite capturar tres datos en diferentes lapsos de tiempo, por medio del servidor Web que trae por defecto el PLC S7-1200.

Parámetros de entrada y salida

- **REQ:** Entrada booleana que crea el archivo cuando detecta un flanco positivo.

- **RECORDS:** Captura de datos mientras esta entrada este en TRUE.
- **NAME:** Nombre del archivo a crear.
- **ID:** Dirección asignada al bloque.
- **HEADERS:** Encabezados de cada una de los datos a exportar.
- **M CLOCK:** Frecuencia con la que se realiza la captura de datos.
- **DATA1:** Variable 1 para guardar en el archivo creado.
- **DATA2:** Variable 2 para guardar en el archivo creado.
- **DATA3:** Variable 3 para guardar en el archivo creado.

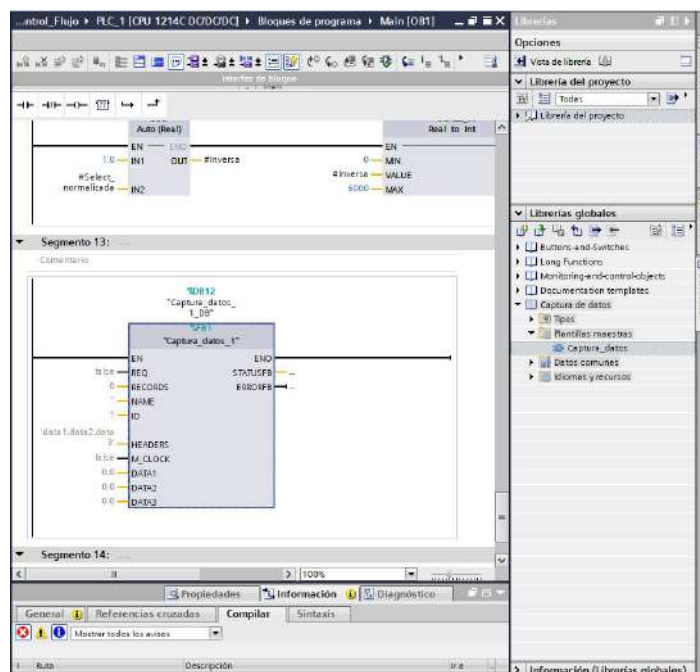


Figura 7.22: Librería Captura de datos

Fuente: Autor

7.4.3 Librería Filtro EMA

Esta librería es un filtro por programación EMA (Exponential Moving Average), según el valor de entrada que se le de en alpha, el filtro es más suave o estricto.

Parámetros de entrada y salida

- **Alpha:** Constante de filtrado, toma valores de 0 a 1, entre más cercano a cero sea el valor, es más estricto el filtrado.
- **Señal:** Señal a filtrar.

- Señal filtrada: Señal filtrada.

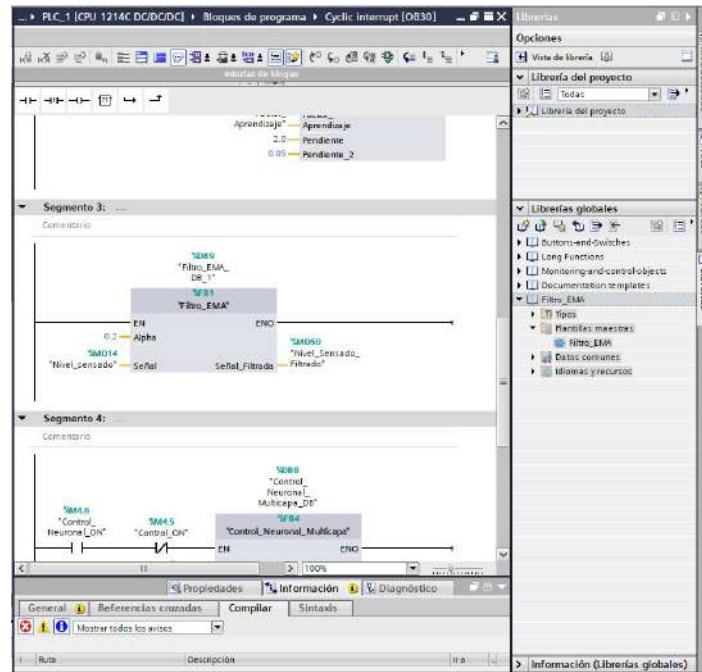


Figura 7.23: Librería Captura de datos

Fuente: Autor

7.4.4 Librería de control neuronal adaline

El bloque de control neuronal adaline mencionado en la sección 6.3 se almaceno en una librería para su uso en cualquier programa figura 7.24, los parámetros de entrada y salida se describen en la misma sección.

7.4.5 Librería de control neuronal multi-capa

El bloque de control neuronal multi-capa mencionado en la sección 6.4 se almaceno en una librería para su uso en cualquier programa figura 7.25, los parámetros de entrada y salida se describen en la misma sección.

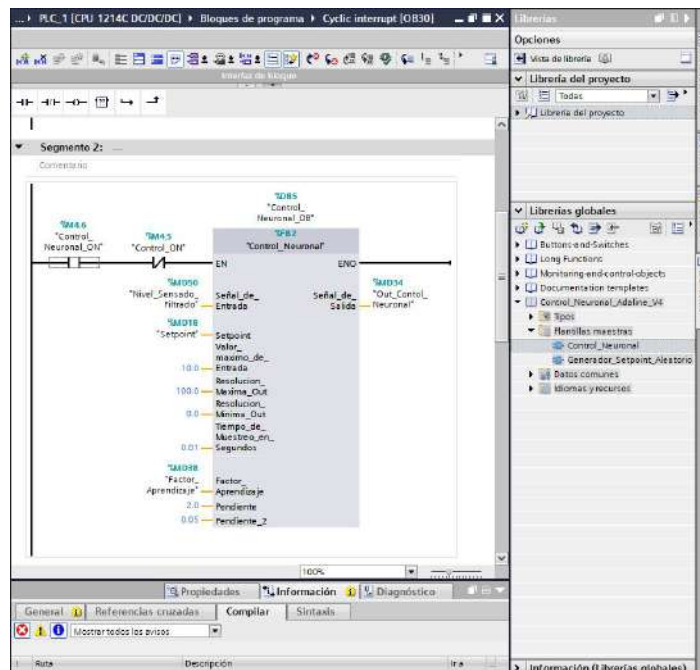


Figura 7.24: Librería control neuronal adaline

Fuente: Autor

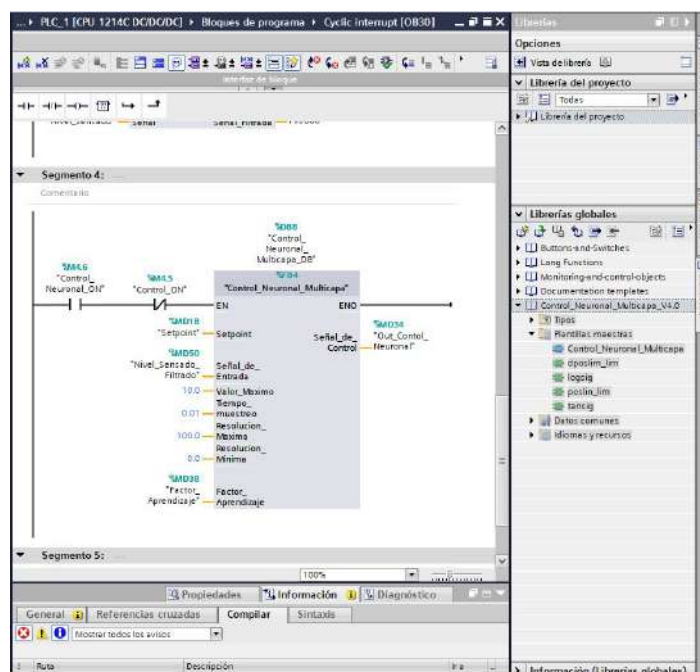


Figura 7.25: Librería control neuronal multi-capa

Fuente: Autor

8 Conclusiones

Se llevo el algoritmo de control neuronal adaline a lenguaje SCL y se comprobó su funcionamiento, demostrando buenos resultados en la planta de velocidad, llegando a tener un mejor desempeño que la respuesta generada por el control PID, esto demuestra que el algoritmo es totalmente funcional en el autómata. En sistemas con dinámicas rápidas como el de velocidad tuvo una respuesta satisfactoria, sin embargo, en el sistema de temperatura, el algoritmo presento unas oscilaciones por arriba y debajo del valor deseado, haciendo que no se estabilizara, estas oscilaciones pudieron ser causa de un factor de aprendizaje muy grande, se realizaron algunas iteraciones, pero como la planta de temperatura tiene un ciclo tan demorado, realizar varios cambios para realizar varias pruebas se vuelve tedioso, por falta de tiempo no se pudo seguir iterando con otros valores del factor de aprendizaje.

Se llevo el algoritmo de control neuronal multi-capa al lenguaje SCL con la topología de tres neuronas de entrada, tres neuronas en la capa oculta y una neurona a la salida. Se comprobó el funcionamiento del algoritmo, demostrando un buen desempeño en la plantas de velocidad y flujo, con respuestas rápidas sin errores muy grandes asociados. En general el algoritmo generado para este bloque es un algoritmo de una red MLP en la que se puede generar varias topologías lo cual permitiría realizar muchas más aplicaciones, con lo que se puede afirmar que el bloque de controlador neuronal multi-capa es solo una de varias aplicaciones que podría tener las redes neuronales de este tipo, en dispositivos PLC's.

El buen desempeño de los algoritmos de control basados en redes neuronales depende en gran medida de la calidad de los instrumentos usados, factores como la estabilidad de medida, la resolución y el tiempo de muestro son importantes, la señal que realiza la retroalimentación del sistema debe ser una señal limpia, dicho en otras palabras, el ruido de los instrumentos de medición afecta mucho la estabilidad de los controladores neuronales, por lo que se hace necesario tener una lectura muy limpia para una buena eficiencia del algoritmo de control neuronal, en la lecturas de los sensores de temperatura esté fluctuaba bastante, a pesar de que se le aplicó un filtro EMA la señal seguía variando bastante en décimas, provocando posiblemente inestabilidad en la respuestas de los controladores neuronales.

Se comprobó la teoría del factor de aprendizaje, el valor de esta constante determina la velocidad de respuesta del sistema, un factor de aprendizaje muy grande hace que el sistema responda muy rápido y existan oscilaciones; un factor de aprendizaje muy pequeño hace que el sistema tarde demasiado en estabilizarse, encontrar el factor de aprendizaje adecuado puede ser la clave para obtener la mejor respuesta con los algoritmos de control neuronal.

Los algoritmos de control neuronal no demostraron el mejor de los desempeños en la planta con dinámicas un poco mas lentas, una de las posibles razones para que no tengan un buen

desempeño es debido al saturador interno aplicado, puede que no sea la mejor manera de aplicarlo y esto haya ocasionado las oscilaciones tan pronunciadas como en el de la planta de temperatura, otra posible causa es el no encontrar el valor de aprendizaje óptimo.

El sensor usado en el sistema de control de flujo, no es un sensor muy exacto con respecto a la medida de flujo y además hace parte más de un tipo de sensor didáctico, este dispositivo se uso para reemplazar otro sensor que se tuvo en consideración desde un inicio, sin embargo tenia tanto ruido asociado que imposibilitaba el buen desempeño de los tres controladores, además es importante recalcar que el sensor se veía bastante afectado por la presión de entrada del agua. Además la válvula proporcional que se uso, daba pasos demasiados grandes lo que imposibilitaba un control muy exacto.

Se genero librerías en TIA PORTAL donde se puede compartir los bloques creados de tal manera que se pueda ingresar en otros programas las veces que sea necesario, algunos de estos bloques se dejaran disponibles para el acceso de la comunidad estudiantil y científica interesada.

Los algoritmos de control neuronal generados en SCL no usan ninguna clase de función especial propia del software de TIA PORTAL, por lo que se podría replicar en el lenguaje de ST para hacer llegar el algoritmo a otros autómatas que incorporen este tipo de lenguaje, ya que el SCL y ST son muy similares en su sintaxis.

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
- Acosta, B., Maria, I., Salazar, I., Zuluaga, y Harold;. (2000). *Tutorial de redes neuronales* (Tesis Doctoral, Universidad Tecnológica de Pereira, Colombia). Descargado de <http://ohm.utp.edu.co/neuronales/main.htm>.
- Antonsen, T. M. (2019). *Controles PLC con Texto Estructurado (ST) (Spanish Edition)* (1a ed.; D. Books on Demand GmbH, Copenhagen, Ed.). Dinamarca,Randers: Books on Demand GmbH, Norderstedt, Germany.
- 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.
- Barnes, J. C. i. (2007). *Plc S7-300 Programación estructurada*. Descargado de <http://www.infoplcn.net/files/descargas/siemens/infoPLC{ }net{ }S7{ }SCL{ }v1{ }0.pdf>
- Caicedo B, E. F., y López S, J. A. (2009). *Una aproximación práctica a las redes neuronales artificiales* (Edición di ed.). Cali, Colombia. doi: 10.25100/peu.64
- 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>
- Cui, X., y Shin, K. G. (1993). Direct Control and Coordination Using Neural Networks. *IEEE Transactions on Systems, Man and Cybernetics*, 23(3), 686–697. doi: 10.1109/21.256542
- Daneri, P. A. (2008). *PLC Automatizacion y Control Industrial*.
- Daza Salinas, S. (2018). *ESTUDIO DE LA CAIDA DE PRESIÓN A TRAVÉS DE UNA VÁLVULA DE BOLA AL GENERAR VARIACIONES EN EL ÁNGULO DE APERTURA Y EN EL CAUDAL DEL FLUJO, MEDIANTE EL USO DEL SOFTWARE DE INGENIERÍA ANSYS* (Tesis Doctoral, UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS). Descargado de <https://doi.org/10.1016/j.cell.2017.12.025><http://www.depkes.go.id/resources/download/info-terkini/hasil-risikesdas-2018.pdf><http://www.who.int/about/licensing/>
- Dorf, R. C. (2005). *Sistemas de control moderno* (10a ed.; HALL PEARSON PRENTICE, Ed.). Madrid, España: Pearson Educación.

- DURTRON. (2018). *Plc Siemens Cpu 1214c 6es7 214-1ag40-0xb0 S7-1200 – Durtron*. Descargado 2020-11-15, de <https://www.durtron.com/producto/plc-siemens-cpu-1214c-6es7-214-1ag40-0xb0-s7-1200/>
- El-Araby, E. A. (2006). Real-Time Neural Network Speed Adaptive Controller Implementation on the PLC for a DC Drive. *Menoufia Journal of Electronic Engineering Research*, 16(1), 45–59. doi: 10.21608/mjeer.2006.64776
- ELECTRO HER. (2018). *Resistencia Eléctrica Tubular Varias Medidas | Mercado Libre*. Descargado 2020-11-14, de [https://articulo.mercadolibre.com.co/MC0-463138166-resistencia-electrica-tubular-varias-medidas-\[_\]JM](https://articulo.mercadolibre.com.co/MC0-463138166-resistencia-electrica-tubular-varias-medidas-[_]JM)
- HUIMAIKEJI. (2018). *0-200 Celsius RTD PT100 SBW Sensor de Temperatura Medidor Transmisor de Temperatura Módulo Detector Aislado 0~200 Grados 4-20MA: Amazon.es: Electrónica*. Descargado 2020-11-23, de <https://www.amazon.es/Celsius-Temperatura-Medidor-Transmisor-Detector/dp/B07GD12PM9>
- Jaimés, L., y Giraldo, M. (2015). Controladores avanzados en Plc. *Revista Politécnica*, 57–66. Descargado de [https://scholar.google.es/scholar?start=10{&}q=que+es+SCADA+ARTICULOS+EN+ESPA{~{N}}OL{&}hl=es{&}as{\[_\]sdt=0,5{#}4](https://scholar.google.es/scholar?start=10{&}q=que+es+SCADA+ARTICULOS+EN+ESPA{~{N}}OL{&}hl=es{&}as{[_]sdt=0,5{#}4)
- Martín del Brio, B., y Sanz, A. (2001). *Redes neuronales y logica borrosa* (2a ed.; Alfaomega, Ed.). Zaragoza, España: Alfaomega.
- Marulanda Gonzalez, L. G. (2010). *CONTROL POR MEDIO DE REDES NEURONALES*. Santiago de Cali.
- Mazzone, V. (2002). *Controladores PID*. Bernal, Argentina. Descargado de <http://www.dia.uned.es/{~}fmorilla/MaterialDidactico/ElcontroladorPID.pdf>
- Meneses Jurado, E. E. (2019). *Implementación de técnicas de control avanzadas en un sistema embebido* (Tesis Doctoral no publicada). Universidad de Pamplona - Colombia.
- Naylamp Mechatronics. (2018). *Sensor de flujo de agua 3/4" FS300A*. Descargado 2020-12-05, de <https://www.naylampmechatronics.com/sensores-liquido/156-sensor-de-flujo-de-agua-34-fs300a.html>
- Nguyen, D. H., y Widrow, B. (1990). Neural Networks for Self-Learning Control Systems. *IEEE Control Systems Magazine*, 18–23.
- Noriega, A., Barrera, A., y Ordez, A. (2008). Diseño de un controlador neuronal y su implementación 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>
-

- PLC-City. (2018). *6ES7232-4HA30-0XB0 | Siemens Simatic S7-1200 - Signal Board*. Descargado 2020-11-15, de <https://www.plc-city.com/shop/es/siemens-simatic-s7-1200-signal-boards/6es7232-4ha30-0xb0.html>
- Quiña, P. S. (2014). DISEÑO E IMPLEMENTACIÓN DE SISTEMAS DE CONTROL NEURONAL DE TEMPERATURA UTILIZANDO EL SOFTWARE. *ESCUELA POLITÉCNICA DEL EJERCITO, ECUADOR*, 1, 9.
- 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.
- Sanchis Llopis, R., Romero Perez, J., y Ariño Latorre, C. (2010). *Automatización Industrial* (Publicacio ed., Vol. 10) (no 1). Castellon, España.
- Siemens AG. (2015). *Hoja de Datos: 6ES7214-1AG40-0XB0*. Siemens.
- Ssdielect. (2011). *PT100 SENSOR DE TEMPERATURA RTD DE PLATINO 3 HILOS*. Descargado 2020-11-14, de <https://ssdielect.com/es/magnitudes-electricas-1/2102-pt100-sensor-temperatura.html>
- Tanomaru, J., y Omatu, S. (1992). Process Control by On-Line Trained Neural Controllers. *IEEE Transactions on Industrial Electronics*, 39(6), 511–521. doi: 10.1109/41.170970
- Yalçm, Y., y Kurtulan, S. (2003). Implementation of a PID-neural network with PLC. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 36(7), 89–93. doi: 10.1016/S1474-6670(17)35812-3
- Ye, Z., y Kim, M. K. (2018). Predicting electricity consumption in a building using an optimized back-propagation and Levenberg–Marquardt back-propagation neural network: Case study of a shopping mall in China. *Sustainable Cities and Society*, 42(March 2019), 176–183. Descargado de <https://doi.org/10.1016/j.scs.2018.05.050> doi: 10.1016/j.scs.2018.05.050
-

Lista de Acrónimos y Abreviaturas

AC	Corriente alterna.
ADALINE	Adaptative Linear Element.
EMA	Exponential Moving Average.
FBD	Function Block Diagram.
IA	Inteligencia Artificial.
IEC	International Electrotechnical Commission.
IEEE	Institute of Electrical and Electronics Engineers.
IL	Instruction List.
LM	Algoritmo Levenberg-Marquardt.
LMS	Least Mean Squares.
MLP	Multilayer perceptron.
NN	Neuronal Network.
PD	Proporcional + Derivativo.
PI	Proporcional + Integral.
PID	Proporcional + Integral + Derivativo.
PLC	Programmable Logic Controller.
RNA	Redes Neuronales Artificiales.
RPM	Revoluciones por minuto.
RTD	Resistance temperature detector.
SCL	Structured Control Language.
SFC	Sequential Function Chart.
SSR	Solid State Relay.
ST	Texto Estructurado.
TFG	Trabajo Final de Grado.