Universidad Nacional de Ingeniería

Facultad de Ingeniería Mecánica

TESIS

Diseño de un Control No Lineal Adaptativo para Controlar el Nivel de Agua Aplicado a un Sistema de Tanques en Cascada usando NIDAQ USB 6211

Para obtener el Título Profesional de Ingeniero Mecatrónico.

Elaborado por

Marco Antonio Cipra Rodriguez

0009-0002-7193-2449

Asesor

Dr. Ing. Ricardo Raúl Rodriguez Bustinza

0000-0002-6411-7123

TOMO I

LIMA – PERÚ

2024

Citar/How to cite	(Cipra, 2024)
Referencia/Reference Estilo/Style: APA (7ma ed.)	Cipra, M. (2024). *Diseño de un Control No Lineal Adaptativo para controlar el nivel de agua aplicado a un Sistema de Tanques en Cascada usando NIDAQ USB 6211*. [Tesis de pregrado, Universidad Nacional de Ingeniería]. Repositorio Institucional UNI.

Dedicatoria

Dedico este trabajo a mi Madre y a mi Padre, pilares fundamentales en mi crecimiento, no solo académico, sino integral en todos los aspectos de la vida. Agradezco su amor incondicional, apoyo constante, sabios consejos, confianza inquebrantable y paciencia interminable.

Agradecimientos

Agradezco profundamente a mi familia, especialmente a mis padres, cuyo amor, apoyo y sacrificio se reflejan en cada página de esta tesis.

Extiendo mi gratitud a todas aquellas personas que fueron parte de mi camino universitario, agradezco al Dr. Ing. Ricardo Rodríguez Bustinza por su apoyo incondicional y a todos aquellos que han contribuido a la realización de esta tesis.

Agradezco a mi alma mater, la Universidad Nacional de Ingeniería, por ser mi hogar académico, y a cada docente que contribuyó con su sabiduría a mi formación durante estos significativos 5 años.

Este logro es el resultado del esfuerzo y aprendizaje conjunto, y cada expresión de gratitud en estas líneas es un sincero reconocimiento a quienes han sido parte fundamental de mi trayectoria académica.

Resumen

El presente trabajo de tesis aborda el diseño de un controlador no lineal adaptativo para el control del nivel de agua en un sistema de tanques en cascada. La hipótesis principal es que un controlador no lineal adaptativo es capaz de mejorar significativamente la respuesta del sistema en términos de tiempo de establecimiento, sobreimpulso y error en estado estacionario en comparación con controladores tradicionales. Para validar esta hipótesis, se diseñaron y compararon tres tipos de controladores: un MRAC con regla MIT, un MRAC con método Lyapunov y un controlador PID sintonizado mediante técnicas MRAC.

La metodología empleada incluyó la identificación de un modelo Hammerstein para representar el sistema de tanques, seguido por la simulación de los controladores en Simulink de MATLAB. Se realizaron pruebas con una señal de entrada cuadrada de 5 cm de amplitud para evaluar la capacidad de seguimiento y la adaptación de los parámetros de cada controlador. Los criterios de comparación fueron el tiempo de establecimiento, el tiempo de pico y el sobreimpulso.

Las conclusiones principales indican que el controlador PID sintonizado con técnicas MRAC presentó el mejor desempeño general, con un sobreimpulso nulo y un tiempo de establecimiento de 160 segundos. Los controladores MRAC, tanto con la regla MIT como con el método Lyapunov, también demostraron un desempeño adecuado. Estos hallazgos sugieren que la implementación de controladores no lineales adaptativos es una estrategia efectiva para el control preciso del nivel de agua en sistemas de tanques en cascada, con potenciales aplicaciones en diversos procesos industriales.

Palabras clave — Controlador Adaptativo No lineal, Sistema de Tanques en Cascada, Modelo Hammerstein-Wiener, Simulink MATLAB.

Abstract

This thesis work deals with the design of an adaptive nonlinear controller for water level control in a cascade tank system. The main hypothesis is that an adaptive nonlinear controller is able to significantly improve the system response in terms of settling time, overshoot and steady state error compared to traditional controllers. To validate this hypothesis, three types of controllers were designed and compared: an MRAC with MIT rule, an MRAC with Lyapunov method and a PID controller tuned using MRAC techniques. The methodology employed included the identification of a Hammerstein model to represent the tank system, followed by simulation of the controllers in MATLAB Simulink. Tests were performed with a 5 cm amplitude square input signal to evaluate the tracking capability and parameter tuning of each controller. The comparison criteria were settling time, peak time and overshoot.

The main findings indicate that the PID controller tuned with MRAC techniques had the best overall performance, with zero overshoot and a settling time of 160 seconds. The MRAC controllers with both the MIT rule and the Lyapunov method also demonstrated adequate performance. These findings suggest that the implementation of adaptive nonlinear controllers is an effective strategy for accurate water level control in cascade tank systems, with potential applications in various industrial processes.

Keywords — Adaptive Nonlinear Controller, Cascade Tank System, Hammerstein-Wiener Model, Simulink MATLAB.

Tabla de Contenido

Pág.

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Introducción

En el presente trabajo de tesis titulado “Diseño de un Control No Lineal Adaptativo para controlar el nivel de agua aplicado a un Sistema de Tanques en Cascada usando NIDAQ USB 6211”, se analiza y ejecuta el diseño de un sistema de control adaptativo no lineal en una planta piloto de tanques en cascada. La tesis está dividida en cuatro capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

En el primer capítulo se expone los antecedentes investigativos, la descripción y formulación del problema de investigación, los objetivos de estudio (general y específicos), las hipótesis (general y específicas) y las variables y operacionalización de variables, proporcionando una introducción completa al tema de investigación.

El segundo capítulo se centra en el marco teórico y conceptual. En la sección de Marco Teórico, se abordan los siguientes temas: el proceso de control del nivel en tanques en cascada, los sistemas de control, la identificación de sistemas, las redes neuronales artificiales, el control adaptativo, el software de simulación y el hardware utilizado. La sección de Marco Conceptual incluye la descripción del modelo Hammerstein-Wiener, la metodología de identificación propuesta, las redes neuronales artificiales de enlace funcional, el control adaptativo por modelo de referencia, el predictor de Smith y el protocolo Modbus.

En el tercer capítulo se detalla el desarrollo del trabajo de investigación, incluyendo el diseño del sistema de control, la identificación del modelo no lineal de la planta piloto, y la implementación de los controladores en el entorno de simulación de MATLAB/Simulink. Este capítulo también incluye la construcción del tablero eléctrico, que alberga los componentes necesarios para la alimentación, procesamiento y control del sistema.

El cuarto capítulo se dedica al análisis y discusión de los resultados obtenidos. Se presentan los gráficos de la respuesta temporal, la acción de control y el error para los diferentes controladores y parámetros de adaptación. Además, se incluyen tablas

comparativas de las características de desempeño de los controladores, evaluando el sobreimpulso, el tiempo de pico y el tiempo de establecimiento.

Esta tesis concluye con una sección de conclusiones que evalúan el cumplimiento de los objetivos del proyecto, seguida de recomendaciones para futuros trabajos en el área de control adaptativo no lineal. Las referencias bibliográficas incluyen los libros, artículos, páginas web y manuales consultados, y los anexos proporcionan información adicional relevante para el desarrollo del proyecto.

Capítulo I. Generalidades

El control de procesos es crucial en la ingeniería moderna, especialmente en la automatización industrial, y es vital para industrias como la petroquímica, tratamiento de aguas, alimentos y bebidas, y farmacéutica. El control del nivel de líquidos en sistemas de tanques en cascada, donde el fluido se transfiere de un tanque a otro, es particularmente desafiante debido a la interacción dinámica, variabilidad de condiciones y características no lineales del proceso. Mantener el nivel adecuado es esencial para evitar desbordamientos, asegurar la consistencia y optimizar recursos.

Tradicionalmente, se han usado controladores proporcionales-integrales-diferenciales (PID) por su simplicidad y eficacia. No obstante, estos tienen limitaciones significativas en sistemas con dinámicas complejas y no lineales. Esto ha impulsado el desarrollo de controladores no lineales adaptativos, que son más robustos y adaptables a las variaciones del proceso.

La implementación de estos controladores avanzados se ha simplificado con herramientas de adquisición de datos y control, como el NIDAQ USB 6211, que facilita la integración con entornos de programación y simulación, proporcionando una plataforma versátil para el desarrollo y prueba de algoritmos de control en tiempo real.

1.1 Antecedentes Investigativos

1.1.1 Investigaciones Internacionales

- Cevallos Ulloa, H. y Navarrete Díaz, D. R. (2019). Tesis de Maestría “Análisis comparativo de controladores clásicos versus controladores avanzados aplicados a una planta real de control de nivel de dos tanques no interactivos bajo la plataforma de LabVIEW”. Escuela Superior Politécnica del Litoral. Este trabajo investigó el desempeño de un controlador PID clásico frente a un controlador basado en lógica difusa en una planta educativa compuesta por dos tanques

cúbicos en cascada no interactivos. La planta se operó mediante la regulación de una bomba centrífuga trifásica y se integró con la plataforma LabVIEW para la adquisición de datos y el diseño de controladores. Los resultados mostraron que, aunque los controladores PI ofrecieron una respuesta rápida a cambios de consigna, presentaron mayores errores en estado estacionario y comportamientos bruscos que dañan a largo plazo los actuadores. En contraste, los controladores difusos lograron un mejor manejo de perturbaciones y una salida de control más suave, lo que los hace más adecuados para procesos industriales donde la suavidad y la longevidad de los actuadores son críticas.

- Vojtesek J. y Dostal P. (2015). Adaptive Control of Water Level in Real Model of Water Tank. Este artículo mostró los resultados de control de las simulaciones en el modelo matemático y las mediciones reales en el modelo real del depósito de agua. El nivel en el tanque se controló mediante el caudal volumétrico de entrada a través de la electroválvula proporcional. El enfoque adaptativo con la identificación recursiva se empleó aquí para controlar este sistema. El controlador fue sintonizado por la elección de la posición del polo en el método de colocación de polos. El beneficio de este artículo se encuentra también con la verificación de los resultados de simulación obtenidos en el modelo real del tanque de agua que mostraron la utilidad de este enfoque de control, aunque el sistema tuviera propiedades no lineales. Las mediciones de control también mostraron la histéresis de la electroválvula proporcional que se utilizó como actuador en el control.
- Higuera Arias, C. y Camacho Poveda, E. (2014). Tesis de pregrado “Evaluación de desempeño de controladores para regulación de temperatura y nivel en tanques en cascada”. Universidad Pedagógica y Tecnológica de Colombia. Esta tesis evalúa tres estrategias de control (PID, algebraico y lógica difusa) para regular nivel y temperatura en el tanque central de una planta piloto de tanques en cascada. Se obtuvo un modelo matemático del sistema utilizando modelamiento matemático e

identificación de sistemas con MATLAB. Los modelos fueron un modelo no lineal Hammerstein-Wiener para el nivel y una función de transferencia para la temperatura. Los controladores se diseñaron, simularon e implementaron utilizando un PLC conectado a la planta piloto. Se compararon curvas de respuesta ante diferentes condiciones y se analizaron para obtener parámetros de desempeño. Los resultados indicaron que el controlador por lógica difusa fue la mejor opción para la regulación propuesta, seguido por el controlador PID. El controlador algebraico mostró el menor cumplimiento de los parámetros.

1.1.2 Investigaciones Nacionales

- Cuzcano Rivas, A. (2021). Tesis de doctorado, Diseño de un Control No Lineal Por Modo Deslizante para Controlar Nivel del Proceso Tanque con Agua. En esta tesis de doctorado se resolvió el problema del control del nivel de agua basado en la técnica de control no lineal por modo deslizante. El sistema de control fue capaz de hacer que las variables controladas sigan la evolución de las consignas de referencia arbitrarias con mínimo sobre impulso, mínimo tiempo de estabilización y error en estado estable nulo, consideraciones importantes para llevar a cabo una alta performance del sistema de control propuesto.
- Rodriguez Bustinza R. et al. (2019). Grey box identification and adaptive control in a water level system. En esta investigación se estudió la aplicación del Control Adaptativo para estimar perturbaciones externas en un modelo de Sistema de Nivel de Agua, aprovechando una identificación no lineal de caja gris construida que permitió explotar su dinámica. Las perturbaciones externas estuvieron especialmente relacionadas con manipulaciones de válvulas considerando información de sensado limitada utilizando únicamente sensores de altura de agua. Se emplearon dos metodologías: (1) Se utilizó un modelo de caja gris para identificar la planta bajo un punto de operación y, (2) Se utilizó el control adaptativo

para reaccionar contra cambios en el sistema o perturbaciones en la válvula y aun así conducir a un controlador convergente y asintóticamente estable.

- Rojas Moreno, A. y Aquize Palacios, R. (2011). Tesis de Maestría “Implementación de un Sistema de Control No Lineal Backstepping Multivariable para la Planta Piloto Tanque con Agua”. Sección de Posgrado de la Facultad de Electricidad y Electrónica de la UNI. Este artículo desarrolló un procedimiento para el diseño e implementación en tiempo real de un sistema de control no lineal backstepping para controlar simultáneamente el nivel y la temperatura en el proceso tanque con agua. El sistema de control diseñado combinó el modelo dinámico no lineal de Lagrange del proceso con un controlador backstepping multivariable. Los estudios de simulación y experimentación realizados demostraron el buen rendimiento de este controlador, el cual se comparó con el rendimiento de un controlador con modos deslizantes.
- Parra Quispe, A. (2007). Tesis de Maestría “Diseño e Implementación de Controladores PID Industriales”. Sección de Posgrado de la Facultad de Electricidad y Electrónica de la UNI. Este trabajo desarrolló un procedimiento de diseño de un sistema de control de múltiples entradas y múltiples salidas (MIMO, Multiple-Input Multiple-Output) PID para controlar el nivel y la temperatura en una planta de tanques de agua. Una planta de este tipo se describe mediante ecuaciones diferenciales no lineales interconectadas, lo que complica los aspectos analíticos del modelado y el diseño del controlador. Sin embargo, se aplicó una técnica linealizada para obtener una descripción lineal MIMO, que permite configurar un sistema de control combinando un controlador PID lineal MIMO que actúa sobre la planta de tanques de agua. Los resultados experimentales demostraron que el controlador diseñado fue capaz de estabilizar el nivel y la temperatura del tanque simultáneamente. El software de control fue escrito en código LabVIEW.

1.2 Descripción del Problema de Investigación

Este trabajo de investigación se origina en el ámbito de los procesos industriales, donde es común el almacenamiento de líquidos en tanques para diversos fines, como tratamientos químicos y mezclas. En particular, el control del nivel de líquidos desempeña un papel de gran relevancia en la industria. Se destacan áreas de aplicación críticas, como las refinerías de petróleo, donde un desbordamiento de líquido tendría consecuencias peligrosas y costosas, y la falta de líquido en los tanques provocaría el funcionamiento inadecuado de bombas y sistemas de drenaje. Además, se considera el contexto de un sistema de control del suministro de agua para un reactor, donde mantener el nivel de agua en el recipiente del reactor es esencial para asegurar un equilibrio adecuado del refrigerante, mantener cubierto el núcleo del reactor y garantizar la eficiencia óptima de los separadores de vapor. El incumplimiento de estos requisitos da lugar a daños en el sistema, entre otros problemas (Cuzcano Rivas, 2021). Muchos de estos procesos requieren el control de varios tanques interconectados. Esto conlleva dificultades adicionales debidas a la no linealidad y al comportamiento impredecible causado por la interacción de los depósitos, los retardos en la respuesta de los actuadores y sensores, y las perturbaciones externas.

A pesar del uso generalizado de los controladores PID en diversos problemas de control, en los que suelen funcionar satisfactoriamente sin requerir modificaciones significativas o incluso un ajuste general, es importante señalar que en algunas aplicaciones presentan fallos y, por lo general, no garantizan un control óptimo. La principal dificultad de los controladores PID es que se trata de un sistema de control realimentado caracterizado por parámetros constantes y sin conocimiento directo del proceso. En consecuencia, el rendimiento global de estos controladores es predominantemente reactivo y está sujeto a compensaciones inevitables. Cuando se aplican individualmente, los controladores PID muestran un rendimiento subóptimo, especialmente en situaciones

en las que es necesario reducir la ganancia del bucle PID para evitar la oscilación, el rebasamiento o la desviación del valor de consigna. También se enfrentan a problemas en presencia de no linealidades, donde sus capacidades de compensación temporal, respuesta a cambios en el comportamiento del proceso y capacidad de reacción ante perturbaciones significativas se ven limitadas, mostrando cierto retardo en su respuesta.

Para abordar los problemas anteriores, se propone un controlador no lineal adaptativo para resolver eficazmente el problema del control del nivel de agua en tanques en cascada. La variable controlada es el nivel de agua. El sistema de control debe ser capaz de hacer que la variable controlada siga la evolución de puntos de consigna arbitrarios con un sobre impulso mínimo, un tiempo de establecimiento mínimo y un error de estado estacionario nulo, lo cual es una condición importante para conseguir una alta eficiencia del sistema de control propuesto.

1.3 Formulación del Problema

1.3.1 Problema General

¿De qué manera el desarrollo del algoritmo de control no lineal adaptativo aplicado a un sistema de tanques en cascada contribuirá a la mejora de la performance del control de nivel de agua?

1.3.2 Problemas Específicos

- Problema específico 1: ¿Cómo influye el modelo experimental en la actuación del control de nivel de agua?
- Problema específico 2: ¿Cómo afecta la adaptación basada en modelo de referencia en el control no lineal adaptativo para controlar el nivel de agua?
- Problema específico 3: ¿De qué manera se mide la performance del sistema de control no lineal adaptativo en el control de nivel de agua cuando se adecúa el parámetro de adaptación?

1.4 Objetivos del Estudio

1.4.1 Objetivo General

Realizar el diseño de un control no lineal adaptativo para controlar el nivel de agua aplicado a un sistema de tanques en cascada.

1.4.2 Objetivos Específicos

- Objetivo Específico 1: Simular el modelo no lineal como respuesta en lazo abierto usando el programa MATLAB.
- Objetivo Específico 2: Analizar los resultados del control no lineal adaptativo en el proceso de adaptación de una consigna tracking.
- Objetivo Específico 3: Comparar el performance del algoritmo de control no lineal adaptativo en diversas pruebas que involucren diferentes valores de adaptación para el control de nivel de agua.

1.5 Hipótesis

1.5.1 Hipótesis General

La ley de control no lineal adaptativa será capaz de darle una alta performance al control de nivel en un sistema de tanques en cascada.

1.5.2 Hipótesis Específicas

- Hipótesis especifica 1: Un adecuado modelo no lineal facilitará la simulación de las características de respuesta a lazo abierto.
- Hipótesis especifica 2: Un adecuado análisis del control no lineal adaptativo resultará en una respuesta que suprimirá el sobre impulso en el tiempo hasta mimetizarse al modelo de referencia.
- Hipótesis especifica 3: El desarrollo del algoritmo de control no lineal adaptativo y sus diferentes valores de adaptación para el control de nivel de agua permitirá validar la performance del control.

1.6 Variables y Operacionalización de variables

1.6.1 Definición conceptual de variables

- Variable independiente (VI): Es la variable de acción del actuador de la válvula de control que proporciona el caudal que se requiere en el tanque de agua, se denota por la variable u.
- Variable dependiente (VD): Esta variable está relacionada con el nivel de agua en el tanque principal y se representa por la variable h.

1.6.2 Operacionalización de variables

Para demostrar y comprobar la formulación de la hipótesis, se procede a realizar el proceso metodológico que consiste en descomponer deductivamente las variables que componen el problema de investigación, partiendo desde lo más general a lo más específico (ver Tabla 1).

Tabla 1

Características de las variables.

Variables	Dimensiones	Indicadores	Escala Medición
Variable Independiente X: voltaje	X.1. Obtención de la ley de control	Componente del sistema de control	Cuantitativa
	X.2. Medida de la acción de control adaptativa	Valor observable	Cuantitativa
	X.3. Limitación de la acción de control adaptativa	Grado de actuación apropiado	Cuantitativa
Variable dependiente Y: nivel de agua	Y.1 Medida del nivel de agua	Valor observable	Cuantitativa

1.7 Metodología de la Investigación

1.7.1 Tipo y Diseño de la Investigación

El presente trabajo de tesis, es una investigación científica, que se encuentra en el ámbito de la ingeniería aplicada, existen diversos temas relacionados con la teoría del control no lineal adaptativo como un área extensa para explorar.

1.7.2 Método de Investigación

En la aplicación de los métodos de investigación se determinarán las etapas de acuerdo con los siguientes aspectos.

1. Recopilar información.
2. Realizar el modelo experimental del proceso tanque de nivel de agua.
3. Realizar el diseño del control no lineal adaptativo.
4. Desarrollar el algoritmo de control adaptativo.
5. Pruebas usando software de simulación de MATLAB y LabVIEW.

1.7.3 Población y Muestra

Por la naturaleza del presente trabajo de investigación, no se utilizaron población y muestra.

1.7.4 Lugar de Estudio

La investigación utilizará como unidad de análisis al Laboratorio de Investigación en Inteligencia Artificial, Automatización, Robótica y Control (LIIAARC) ubicada en la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería.

1.7.5 Técnicas e Instrumentos de Recolección

Por la naturaleza del trabajo de investigación se han utilizado material bibliográfico diverso, para determinar un modelo exacto y en forma experimental desde las bases

teóricas. Esta información será procesada mediante algoritmos en el dominio del tiempo discreto.

1.7.6 Análisis y Procesamiento de Datos

Se expone y detallan las medidas de resumen de las variables y como serán presentados los resultados mediante gráficos procedentes de las simulaciones desde programas basados en texto, MATLAB y LabVIEW.

Capítulo II. Marco teórico y conceptual

2.1 Marco teórico

2.1.1 Proceso Tanque de Nivel en Cascada

Un proceso de tanque de nivel en cascada es un sistema de control que se utiliza para mantener el nivel de líquido en dos o más tanques a un valor deseado. Este proceso se utiliza comúnmente en la industria química y petroquímica para controlar el flujo de líquidos entre los tanques.

En cuanto a la teoría del control, el proceso de tanque de nivel en cascada se modela utilizando un sistema de ecuaciones diferenciales que describen la dinámica del proceso. El modelo matemático resultante se utiliza para diseñar un controlador que mantenga el nivel del líquido en los tanques a un valor deseado.

En la Figura 1 se muestra el esquema general de la planta piloto correspondiente al sistema de tanques en cascada y en la Figura 2 se muestra la planta piloto construida. La planta piloto está conformada por tres tanques conectados en cascada a través de tuberías de agua, una electroválvula, una válvula manual y una electrobomba. La Tabla 2 describe detalladamente cada una de las magnitudes involucradas en el esquema.

La electrobomba establece las condiciones iniciales en el Tanque 1. La electroválvula, regulada por un voltaje $V(t)$, controla el caudal de entrada al Tanque 2. El flujo de salida del Tanque 2 (q_3) es influenciado por una válvula manual, la cual actúa como una perturbación en el sistema. Las variables medidas incluyen los niveles de agua h_1 y h_2 en ambos tanques, así como el caudal de entrada en cada tanque. El Tanque 2 es el tanque principal, y se busca controlar su nivel de agua.

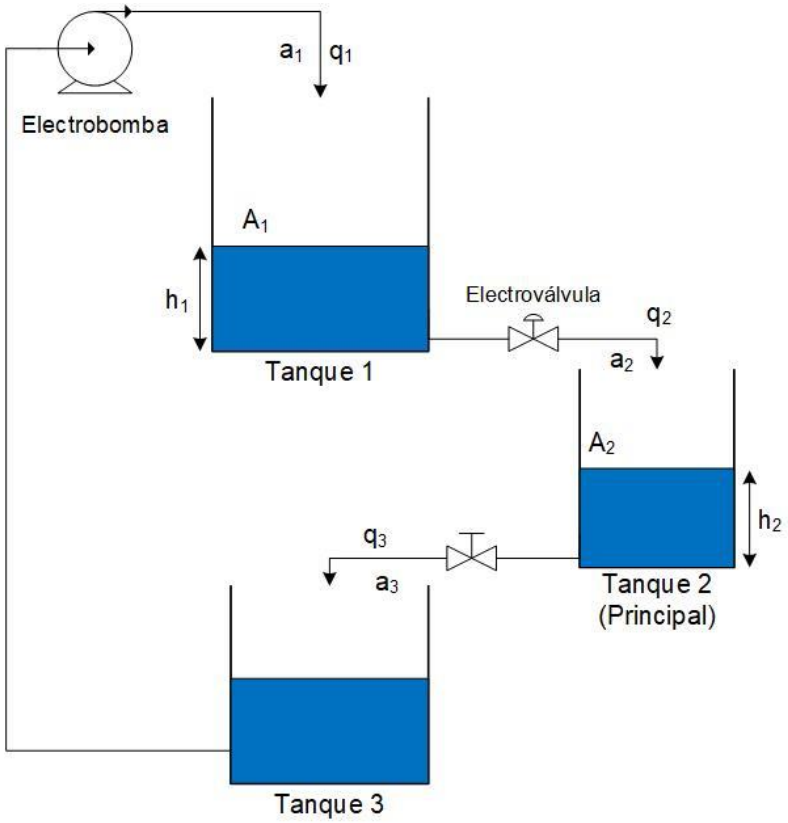
Tabla 2

Magnitudes de la planta piloto de tanques en cascada.

Magnitudes		Descripción	Unidades
Variables	$\boldsymbol{h_1, h_2}$	Nivel de agua de los tanques 1 y 2	cm
	$\boldsymbol{q_1, q_2, q_3}$	Flujo de agua volumétrico de entrada a los tanques	L/h
Parámetros	$\boldsymbol{A_1, A_2}$	Área transversal de los tanques 1 y 2	cm^2
	$\boldsymbol{a_1, a_2, a_3}$	Área transversal de las tuberías en la entrada a los tanques	cm^2

Figura 1

Diagrama del Sistema de Tanques en cascada.



Nota: La variable que se desea controlar es el nivel del agua en el Tanque 2 (h_2), la cual depende tanto del nivel de agua del Tanque 1 (h_1) como del flujo de agua q_2.

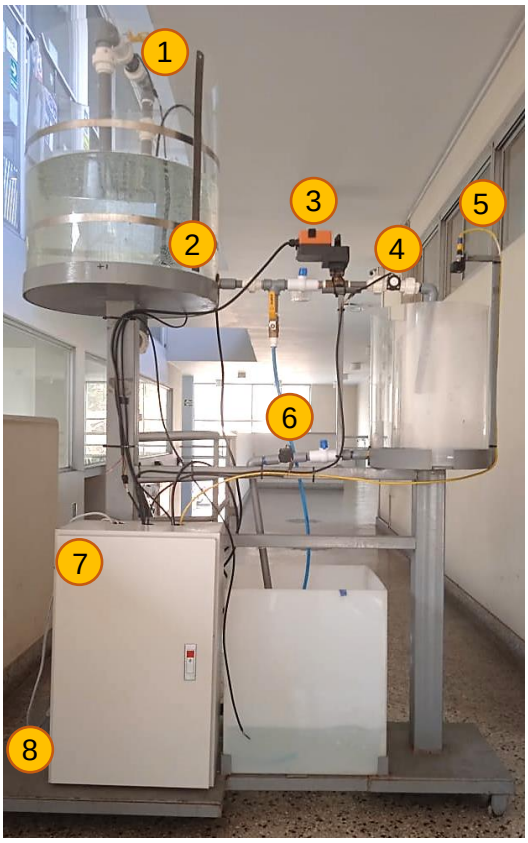
2.1.2 Sistema de Control

Desde la perspectiva de la teoría del control, un sistema o proceso está formado por elementos interconectados que producen salidas en respuesta a señales o datos de entrada. Es importante señalar que para caracterizar un sistema no es necesario

comprender la estructura interna ni las interacciones entre los distintos elementos. En cambio, basta con conocer la relación entre la entrada y la salida del proceso, según el principio de la caja negra. El aspecto más importante de un sistema es comprender su dinámica, es decir, cómo reacciona la salida a los cambios en la entrada (Gomáriz Castro et al., 2000).

Figura 2

Planta Piloto del Sistema de Tanques en Cascada.



1. Sensor de flujo de agua 1
2. Sensor de nivel de agua RS485
3. Válvula proporcional de 2 vías
4. Sensor de flujo de agua 2
5. Sensor de proximidad ultrasónico
6. Sensor de flujo de agua 3
7. Tablero Eléctrico
8. Bomba de agua

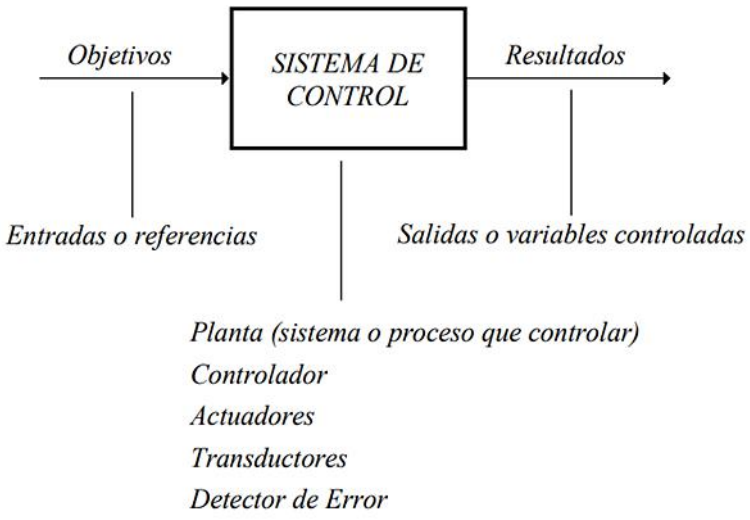
El conocimiento detallado de la relación entre la entrada y la salida permite predecir la respuesta del sistema y elegir las medidas de control adecuadas para mejorarla. Así, el diseñador, conociendo la dinámica deseada, ajusta el sistema de control para alcanzar el objetivo final. En general, un sistema de control se define como un conjunto de elementos que interactúan entre sí para conseguir el comportamiento deseado a la salida de un

proceso mediante una acción de control (Gomáriz Castro et al., 2000). En la Figura 3 se presenta un esquema general de un sistema de control.

Sistemas de Control Dinámico. Existen dos configuraciones generales de sistemas de control, que varían según cómo se maneja la señal de salida del sistema. Estas son los sistemas en lazo abierto y los sistemas en lazo cerrado.

Figura 3

Diagrama esquemático de un sistema de control típico.



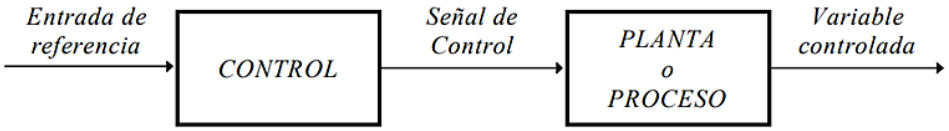
Nota: De *Teoría de control: diseño electrónico*, por Gomáriz Castro et al., 2000.

- Sistemas en Lazo Abierto

En un sistema de control de lazo abierto (ver Figura 4), la señal de salida no se compara con la consigna. En consecuencia, cada consigna se asocia a una condición de funcionamiento fija y la precisión del sistema depende de la calibración. Cuando se producen fallos, el sistema de control no cumple su función. En la práctica, un sistema de control solo se utiliza si se conoce la relación entre la entrada y la salida y no hay fallos internos y externos (Ogata, 2010).

Figura 4

Diagrama de bloques de un sistema de control en lazo abierto.



Nota: De *Teoría de control: diseño electrónico*, por Gomáriz Castro et al., 2000.

- Sistemas en Lazo Cerrado

En un sistema de control de lazo cerrado (ver Figura 5), se aplica al regulador una señal de error, que es la diferencia entre la señal de entrada y la señal de realimentación (esta última es la propia señal de salida o una función formada por la señal de salida y sus derivadas y/o integrales). El objetivo de dicho ajuste es minimizar el error y llevar la señal de salida del sistema al valor deseado. El término "control" siempre implica el uso de realimentación para minimizar el error del sistema (Ogata, 2010).

Clasificación de Sistemas.

- Sistemas Lineales

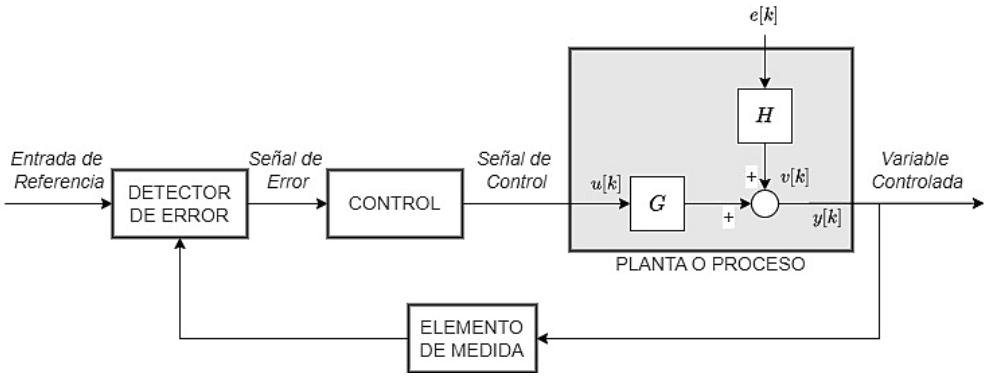
Los sistemas lineales se definen por su capacidad para ser modelados mediante ecuaciones diferenciales lineales. Su propiedad más destacada es la adherencia al principio de superposición. Este principio, fundamental en la teoría de sistemas, facilita la evaluación experimental de la linealidad de un sistema (Gomáriz Castro et al., 2000).

- Sistemas No Lineales

Los sistemas no lineales se caracterizan por poseer ecuaciones diferenciales no lineales. En realidad, la mayoría de los sistemas son inherentemente no lineales (ver Figura 6), aunque en algunos casos son aproximados como sistemas lineales en ciertos rangos (un enfoque comúnmente utilizado para simplificar el análisis). Sin embargo, en estos sistemas, el principio de superposición no es aplicable (Gomáriz Castro et al., 2000).

Figura 5

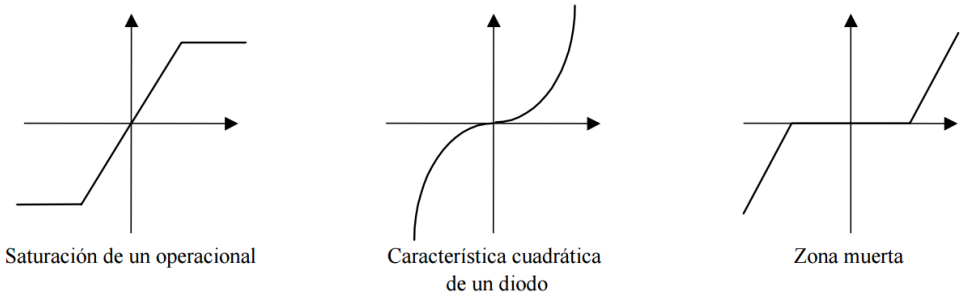
Diagrama de bloques de un sistema de control en lazo cerrado.



Nota: Adaptado de *Teoría de control: diseño electrónico*, por Gomáriz Castro et al., 2000.

Figura 6

Representación de comportamientos no lineales en sistemas electrónicos.



Nota: De *Teoría de control: diseño electrónico*, por Gomáriz Castro et al., 2000.

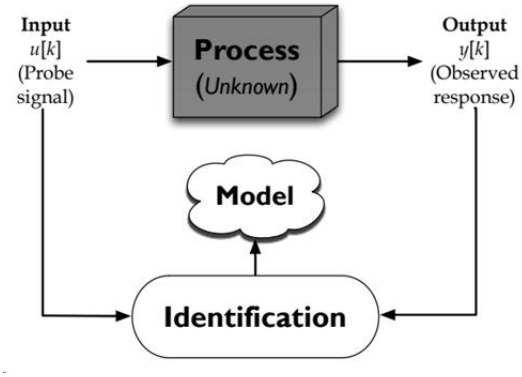
2.1.3 Identificación de Sistemas

En ingeniería de procesos, analizar características y relaciones entre variables es esencial para controlar, predecir, supervisar y diseñar sistemas. El desarrollo de modelos matemáticos, llamado identificación de sistemas, es crucial en este proceso. Existen dos enfoques principales: uno teórico, basado en principios fundamentales, y otro empírico, que utiliza datos experimentales. El enfoque empírico es comúnmente empleado en sistemas complejos, ya que captura información que sería inaccesible mediante un enfoque

teórico. La identificación de sistemas se centra en utilizar datos observados o experimentales para crear una representación matemática precisa de un sistema de procesos (Tangirala, 2015).

Figura 7

Esquema básico de Identificación de Sistemas.



Nota: De *Principles of System Identification: Theory and Practice*, por Tangirala, 2015.

En la Figura 7 se muestra una representación esquemática del proceso típico de identificación de sistemas. El objetivo principal es obtener un modelo a partir de datos de entrada y salida.

En Identificación de Sistemas, los términos entrada y salida tienen significados genéricos. Las salidas son las variables que se miden y se desean predecir, también se conocen como respuestas o variables predichas. Por otro lado, las entradas representan las variables que influyen en las salidas del sistema y se dividen en señales manipulables por el usuario (señales de control) y perturbaciones medidas.

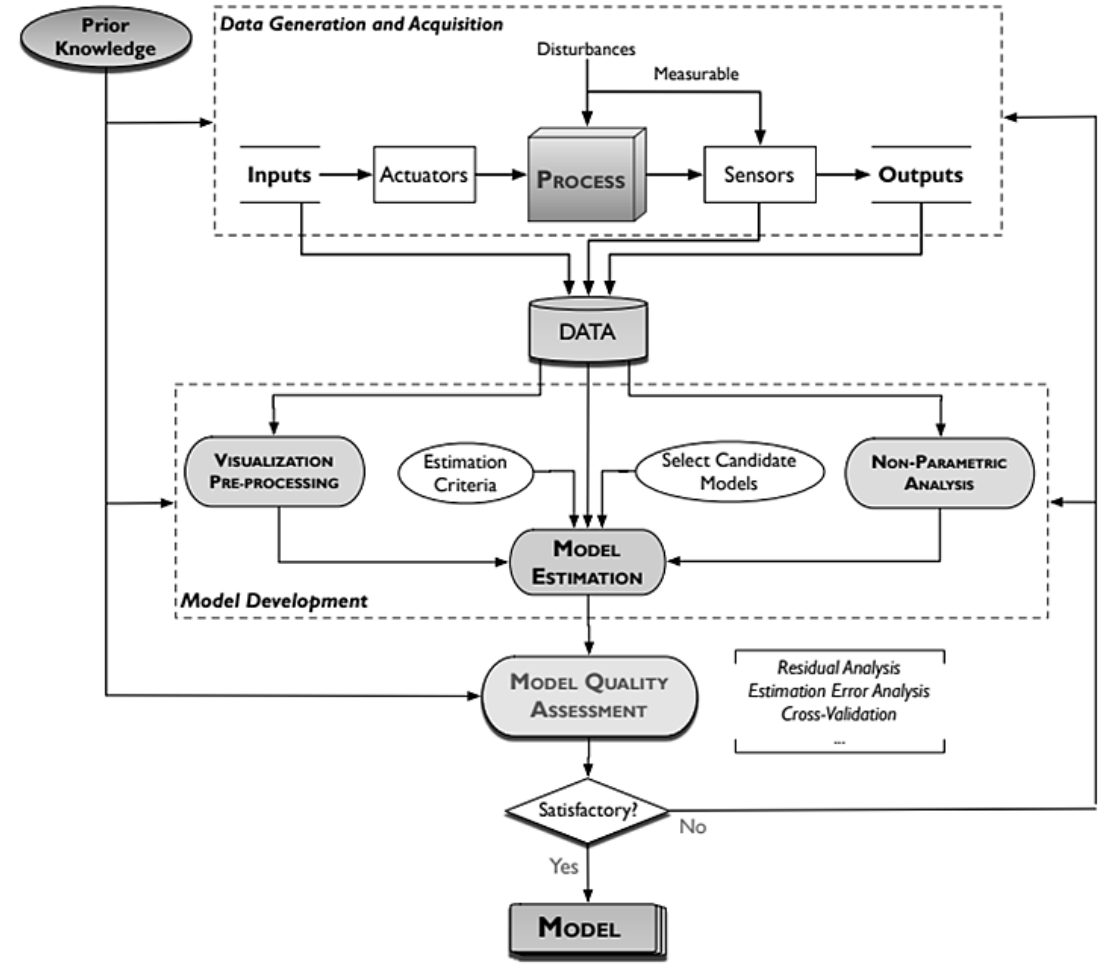
Procedimiento de Identificación de Sistemas. Tangirala (2015) divide el procedimiento de identificación en cinco pasos principales y estos se muestran a continuación:

1. Generación y Adquisición de datos: Este paso implica la obtención de datos de calidad y en cantidad suficiente. El diseño experimental es crucial, decidiendo aspectos como el tipo de entrada, el dispositivo de retención y la frecuencia de muestreo. Se busca una excitación óptima para el sistema que tenga un impacto claro en las salidas medidas, superando el ruido del sensor y las perturbaciones.
2. Preprocesamiento de datos: Los datos adquiridos generalmente necesitan ser preprocesados debido a factores como valores atípicos y datos faltantes. Los valores atípicos y los datos faltantes se abordan mediante métodos estadísticos como la maximización de expectativas y la imputación múltiples.
3. Visualización de datos: La visualización de datos es esencial para extraer información y analizar señales en todas las etapas del proceso. Ayuda a identificar desviaciones y valores atípicos, así como a evaluar la calidad de los datos. También es útil en la selección de modelos candidatos y en la evaluación de la calidad del modelo desarrollado.
4. Desarrollo de modelos: Implica especificar la estructura y el orden del modelo, así como estimar los parámetros mediante un problema de optimización. La elección de modelos candidatos se guía por factores como precisión, exactitud, horizonte de predicción y conocimientos previos. Se busca evitar el sobreajuste y construir un modelo funcional y útil.
5. Evaluación y validación de modelos: Se centra en evaluar qué tan bien el modelo explica las variaciones en los datos de entrenamiento y en verificar la precisión del modelo al predecir respuestas en datos de prueba no utilizados durante el entrenamiento. Se utilizan pruebas estadísticas y de validación cruzada para determinar si el modelo captura las características generales del proceso sin sobreajustarse.

En la Figura 8 se observa un diagrama de flujo que representa los pasos clave de la identificación. El diagrama resalta dos aspectos importantes: la identificación es un proceso iterativo y se tiene en cuenta el conocimiento previo del proceso en cada etapa. Se recomienda incorporar la mayor cantidad posible de conocimiento previo en la identificación.

Figura 8

Diagrama de flujo de un procedimiento iterativo genérico para la identificación de sistemas.



Nota: De *Principles of System Identification: Theory and Practice*, por Tangirala, 2015.

Identificabilidad. Según Tangirala (2015), la identificabilidad de un modelo para un sistema específico se ve influenciada por tres aspectos críticos:

- Características del modelo: Esta propiedad se refiere a la existencia de una correspondencia única entre el modelo y los parámetros que se están estimando, asegurando así una relación uno a uno.
- Condiciones experimentales: Estas condiciones incluyen el tipo de entrada y la frecuencia de muestreo, siendo la naturaleza de la entrada el factor más influyente. Determinar si la entrada proporciona información suficiente para distinguir entre distintos modelos candidatos es crucial y representa un elemento esencial en el diseño experimental.
- Método de estimación: Diversos métodos están disponibles para estimar los parámetros. La cuestión reside en la idoneidad del método de estimación. Es crucial evaluar si dicho método estima los parámetros "verdaderos" cuando se cuenta con una cantidad infinita de muestras, lo que se conoce como propiedad asintótica del estimador. Técnicamente, esto se denomina consistencia.

Relación Señal – Ruido. Incluso cuando se aplica una excitación adecuada, los efectos estocásticos en las mediciones son lo suficientemente significativos como para degradar la calidad del modelo. Por ejemplo, seleccionar frecuencias de muestreo mucho más altas que la velocidad a la que cambian las salidas introduce más ruido que la variación real del proceso.

Para cuantificar las contribuciones relativas de la excitación determinista y las variaciones aleatorias, se utiliza una medida denominada relación señal-ruido (SNR, Signal to Noise Ratio).

$$\mathrm{SNR} = \frac{Varianza\ de\ señal}{Varianza\ del\ ruido} \tag{1}$$

Aquí, el término "señal" hace referencia a la respuesta real del sistema. Mantener una SNR adecuada es esencial para obtener estimaciones de parámetros confiables, independientemente del método de estimación utilizado. Este concepto se comprende cualitativamente de la siguiente manera: ajustar un modelo implica, intuitivamente, explicar las variaciones en la salida. Si una parte significativa de estas variaciones en la medición se debe al ruido, entonces se debilita la contribución de la entrada y, por consiguiente, la precisión en la estimación del modelo también se ve comprometida. Desde otro punto de vista, la SNR representa la relación entre los efectos atribuibles a la variable conocida y las incertidumbres. Por lo tanto, cuanto menor sea la SNR, más ambigua será la estimación del modelo de entrada-salida (Tangirala, 2015).

Sobreajuste. El sobreajuste ocurre cuando el modelo se ajusta para capturar las características "locales" de los datos en lugar de las características "globales". En el contexto de la identificación, esto sucede cuando se interpreta incorrectamente el ruido presente en los datos como parte de los patrones deterministas (entrada-salida), es decir, cuando las variaciones aleatorias en la respuesta se atribuyen a cambios en las variables de entrada. Este error surge cuando el usuario especifica excesivamente la complejidad de la parte determinista en un intento de explicar la salida con la mayor precisión posible. En el caso más extremo, toda la variabilidad de la salida se atribuye exclusivamente a las variaciones de la entrada.

Aumentar la complejidad del modelo determinista tiene como ventaja principal una mejor adaptación a los datos de entrenamiento, lo que se traduce en un menor error de predicción. Sin embargo, la reducción del sesgo en las predicciones conlleva el riesgo de aumentar significativamente los errores estándar (varianza) en las estimaciones de los parámetros del modelo. Además, estos modelos generan predicciones deficientes en nuevos conjuntos de datos, a veces incluso inestables, sin límites claros (Tangirala, 2015).

Clasificación de Modelos. La clasificación primaria de los modelos se fundamenta en el enfoque de la modelización, distinguiendo entre modelos de primeros principios y modelos empíricos. Sin embargo, dentro de estas dos categorías, es posible realizar otras clasificaciones basadas en diversas consideraciones. Según Tangirala (2015), los modelos se clasifican en función de:

Enfoque de Modelización.

- ✓ Modelos de primeros principios: Estos modelos se basan en teorías físicas, matemáticas o químicas que describen el comportamiento del sistema a partir de las leyes fundamentales que lo rigen. Por ejemplo, un modelo de ecuaciones diferenciales que describe el comportamiento de un circuito eléctrico.
- ✓ Modelos empíricos: Estos modelos se derivan a partir de datos experimentales u observaciones del sistema, sin necesariamente conocer o considerar los principios subyacentes que gobiernan su comportamiento. Por ejemplo, un modelo de regresión que predice la temperatura ambiente en función de la hora del día y la estación del año.

Características del Sistema.

- ✓ Lineal o no lineal: Los modelos lineales asumen una relación lineal entre las variables de entrada y salida del sistema, mientras que los modelos no lineales permiten relaciones no lineales. Por ejemplo, un modelo de sistema masa-resorte-amortiguador es considerado lineal o no lineal dependiendo de cómo se formulen las ecuaciones.
- ✓ Variable en el tiempo o invariable en el tiempo: Algunos sistemas tienen propiedades que varían con el tiempo, mientras que otros permanecen constantes. Los modelos capturan estas variaciones temporales o asumen que las propiedades del sistema son constantes.

Conocimientos que Dispone el Usuario.

- ✓ Determinista o estocástico: Los modelos deterministas predicen el comportamiento del sistema sin considerar la incertidumbre, mientras que los modelos estocásticos incorporan la aleatoriedad y la incertidumbre en las predicciones. Por ejemplo, un modelo determinista predice el movimiento de un proyectil bajo la gravedad, mientras que un modelo estocástico predice el precio futuro de una acción en el mercado.
- ✓ Black-box/grey-box: Un modelo black-box (denominado caja negra) se refiere a un modelo donde no se conoce la estructura interna del sistema y solo se observan sus entradas y salidas, mientras que un modelo grey-box (denominado caja gris) es aquel donde se tiene cierto conocimiento limitado sobre la estructura interna del sistema. Por ejemplo, un modelo de red neuronal profunda se considera un modelo black-box, mientras que un modelo de espacio de estados con algunas de sus ecuaciones conocidas es considerado grey-box.

Ámbito de Modelización.

- ✓ Tiempo continuo o tiempo discreto: Los modelos son formulados en tiempo continuo, donde las variables cambian de manera continua en el tiempo, o en tiempo discreto, donde las variables solo cambian en intervalos discretos de tiempo.
- ✓ Dominio temporal o dominio de frecuencia: Algunos modelos están formulados en el dominio temporal, donde se describe el comportamiento del sistema en función del tiempo, mientras que otros están formulados en el dominio de la frecuencia, donde se analizan las características del sistema en términos de su respuesta a diferentes frecuencias de entrada.

Características de la Respuesta.

- ✓ Estática o dinámica: Los modelos estáticos describen el comportamiento del sistema en un punto particular en el tiempo, mientras que los modelos dinámicos capturan cómo cambia el sistema a lo largo del tiempo en respuesta a entradas variables. Por ejemplo, un modelo estático de una balanza predice el peso de un objeto en reposo, mientras que un modelo dinámico predice cómo se mueve un objeto en la balanza cuando se aplica una fuerza sobre él.
- ✓ Agrupada o distribuida: Algunos modelos consideran la respuesta del sistema como una entidad única (agrupada), mientras que otros modelos consideran la distribución de la respuesta en el espacio o en otras dimensiones. Por ejemplo, un modelo de distribución de temperatura en una placa metálica es distribuido, considerando la temperatura en diferentes puntos de la placa.

De lo expuesto anteriormente se deduce que existe una amplia gama de modelos disponibles para un sistema en particula, dependiendo de los supuestos y las características que se deseen describir. Sin embargo, esta investigación se enfocará en el desarrollo de un modelo de tiempo discreto, no lineal, invariante en el tiempo y dinámico.

Directrices Prácticas para el Muestreo. Tangirala (2015) menciona que una frecuencia de muestreo más alta no necesariamente conlleva una mejor situación en el análisis de sistemas discretos, como se argumenta a continuación:

- A medida que la frecuencia de muestreo aumenta, los sistemas de tiempo discreto tienden a acercarse a los límites de la estabilidad. Esto causa efectos integradores y complicar la identificación y el control de los sistemas.
- Una frecuencia de muestreo excesivamente alta aumenta drásticamente el ruido en las mediciones, lo que afecta negativamente la SNR.

- Las frecuencias de muestreo deben ser proporcionales a la escala temporal del proceso. Si la salida cambia lentamente en comparación con el intervalo de muestreo, un muestreo muy rápido hace que la mayoría de las variaciones observadas sean ruido, reduciendo la SNR y afectando la estimación de parámetros y el desarrollo de modelos.

  Por lo tanto, una regla general es seleccionar el intervalo de muestreo T_s en el rango

$$T_s \in \left[\frac{\tau}{10}, \frac{\tau}{5}\right] \ (\text{entre 20 y 40 muestras en un tiempo de estabilización}) \qquad (2)$$

  Donde τ es el constante temporal dominante o la constante temporal de una aproximación de primer orden del sistema.

Modelos Paramétricos para la Identificación. Según Tangirala (2015), en general, la salida del modelo $y[k]$ se compone de la respuesta "verdadera" del proceso (determinista) $y_{true}[k]$, y las perturbaciones o ruido (estocástico) $v[k]$.

$$y[k] = y_{true}[k] + v[k] \qquad (3)$$

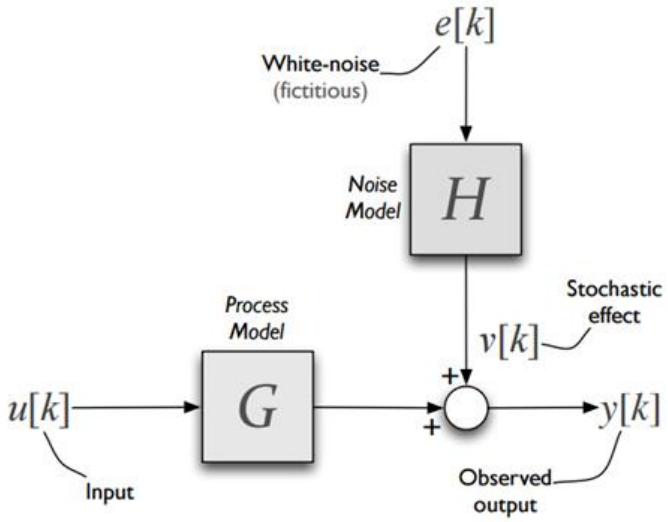
Dando un tratamiento lineal e invariante en el tiempo (LTI, Linear and Time Invariant) a los procesos determinista y estocástico, y denotando estos sistemas por G y H, respectivamente, se representa simbólicamente la situación como:

$$y[k] = Gu[k] + He[k] \qquad (4)$$

donde la notación Gu debe interpretarse como la operación de G sobre u (lo mismo para He) y no como una multiplicación. En la literatura de identificación, es común referirse a G como el modelo de planta y a H como el modelo de ruido. La representación de un sistema LTI se ilustra en la Figura 9.

Figura 9

Representación entrada-salida de un sistema LTI determinista-más-estocástico.



Nota: De *Principles of System Identification: Theory and Practice*, por Tangirala, 2015.

Para escribir la Ecuación (4) es necesario tener en cuenta las siguientes consideraciones:

- Ruido aditivo: El término estocástico se suma a la respuesta real.
- Linealidad e invariancia temporal de G: El proceso determinista es LTI. En este punto, no se requieren restricciones sobre la estabilidad de G.
- Estacionariedad de $v[k]$: La señal estocástica $v[k]$ es estacionaria. Además, cumple con el resultado de la factorización espectral, lo que significa que se expresa como la salida de un sistema LTI controlado por ruido blanco.

Las descripciones paramétricas del sistema de entrada-salida surgen al parametrizar los modelos de planta y ruido, $G(q^{-1})$ y $H(q^{-1})$. Aunque hay diversas formas de parametrizar G y H, las representaciones de la función de transferencia polinómica racional son ampliamente empleadas:

$$G(q^{-1},\theta) = \frac{B(q^{-1})}{A(q^{-1})F(q^{-1})} \qquad H(q^{-1},\theta) = \frac{C(q^{-1})}{A(q^{-1})D(q^{-1})} \tag{5}$$

donde todos los polinomios anteriores son coprimos (sin cancelaciones de polos cero), y

$$A(q^{-1}): 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}$$

$$B(q^{-1}): b_{n_k} q^{-n_k} + b_{n_k} q^{-n_k - 1} + \cdots + b_{n_k + n_b - 1} q^{-(n_b - 1 - n_k)}$$

$$n_k \, \epsilon \, \{\mathbb{Z}^+ \cup 0\}: Retardo\ de\ Entrada - Salida$$

$$C(q^{-1}): 1 + c_1 q^{-1} + \cdots + c_{n_c} q^{-n_c} \tag{6}$$

$$D(q^{-1}): 1 + d_1 q^{-1} + \cdots + d_{n_d} q^{-n_d}$$

$$\theta: \begin{bmatrix} a_1 & \cdots & a_{n_a} & b_0 & \cdots & b_{n_b - 1} & c_1 & \cdots & c_{n_c} & d_1 & \cdots & d_{n_d} & f_1 & \cdots & f_{n_f} \end{bmatrix}^T$$

La identificación de la Ecuación (5) involucra la estimación del vector de parámetros θ para valores especificados de retardo n_k y órdenes polinómicos.

En la Tabla 3 se muestran diferentes tipos de modelos paramétricos utilizados en la Identificación de sistemas junto con sus respectivas ecuaciones.

Tabla 3

Modelos Paramétricos para Identificación.

	Modelo	**Estructura**
AR	Auto-Regressive	$y[k] = \frac{1}{A(q^{-1})} e[k]$
MA	Moving Average	$y[k] = C(q^{-1}) e[k]$
ARX	Auto-Regressive eXogenous	$y[k] = \frac{B(q^{-1})}{A(q^{-1})} u[k] + \frac{1}{A(q^{-1})} e[k]$
ARMA	Auto-Regressive Moving Average	$y[k] = \frac{C(q^{-1})}{A(q^{-1})} e[k]$
ARMAX	Auto-Regressive Moving Average eXogenous	$y[k] = \frac{B(q^{-1})}{A(q^{-1})} u[k] + \frac{C(q^{-1})}{A(q^{-1})} e[k]$
OE	Output Error	$y[k] = \frac{B(q^{-1})}{A(q^{-1})} u[k] + e[k]$
BJ	Box-Jenkins	$y[k] = \frac{B(q^{-1})}{F(q^{-1})} u[k] + \frac{C(q^{-1})}{D(q^{-1})} e[k]$

2.1.4 Redes Neuronales Artificiales

En el cerebro, los estímulos son transmitidos entre las neuronas mediante conexiones sinápticas. Cuando una neurona se activa, libera pequeñas cantidades de neurotransmisores que viajan a través del axón hasta las dendritas de otras neuronas, donde el proceso se repite. Este proceso aumenta o disminuye la relación entre las neuronas involucradas, lo que resulta en la activación o inhibición de ciertos circuitos neuronales en respuesta a un estímulo específico.

Durante el proceso de aprendizaje, se establecen los niveles adecuados de activación e inhibición de las neuronas. Una vez completado, el cerebro es capaz de responder de manera adecuada a ciertos estímulos, lo que constituye el aprendizaje. El conocimiento adquirido se encuentra en los patrones de conexión entre las neuronas, y el cerebro se "entrena" mediante la repetición de estímulos.

Emulando este funcionamiento cerebral, una red neuronal artificial (RNA) es un modelo matemático inspirado en el comportamiento biológico de las neuronas y las estructuras cerebrales. Desde la primera mitad del siglo XX, se han desarrollado modelos computacionales que intentan imitar el comportamiento del cerebro humano.

La fortaleza de las RNAs radica en su habilidad para aprender funciones complejas o no lineales entre variables sin requerir la imposición de presupuestos o restricciones iniciales en los datos. Sin embargo, es evidente que la aplicación de esta tecnología computacional es relativamente novedosa en el estudio de las conductas adictivas (Morales y Moreno, 2017).

Ventajas de las RNAs. Las RNAs ofrecen una serie de ventajas significativas debido a su estructura y fundamentos, los cuales son análogos a los del cerebro humano. Estas ventajas han llevado a la aplicación de esta tecnología en diversas áreas. Según Matich (2001), las ventajas más destacadas son las siguientes:

- Aprendizaje Adaptativo: Capacidad de aprender a realizar tareas basadas en experiencias previas o entrenamiento inicial.
- Auto-organización: Las RNAs crean su propia organización o representación de la información recibida a través de un proceso de aprendizaje.
- Tolerancia a fallos: Aunque la destrucción parcial de una red resulta en la degradación de su estructura, ciertas capacidades de la red se mantienen incluso después de sufrir daños significativos.
- Operación en tiempo real: Los cálculos neuronales se llevan a cabo en paralelo, lo que permite el procesamiento rápido de datos. Para aprovechar esta capacidad, se diseñan y fabrican máquinas con hardware especializado.
- Fácil integración en la tecnología existente: Existen chips especializados para redes neuronales que mejoran su rendimiento en tareas específicas. Esta característica facilita su integración modular en los sistemas tecnológicos preexistentes.

Elementos y Características de una RNA. Las RNAs, al procurar emular el funcionamiento cerebral, simplifican el sistema neuronal humano al identificar y replicar sus elementos más esenciales, así como al imitar su comportamiento mediante algoritmos.

El procedimiento habitual para construir RNAs capaces de ejecutar una tarea específica implica la selección cuidadosa de las características de cada neurona artificial, la estructura o arquitectura de la red, y el método de operación o aprendizaje.

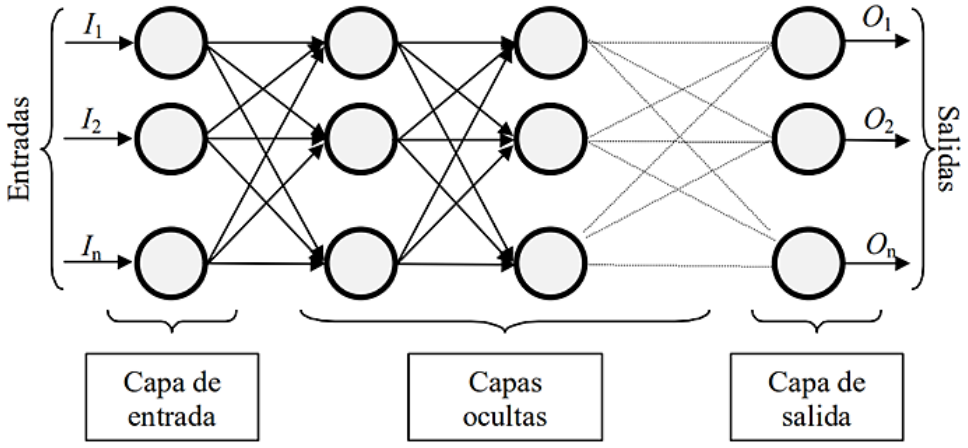
A continuación, se detallan los elementos principales de las RNAs.

Neurona Artificial. Las RNAs se componen de una serie de unidades fundamentales llamadas neuronas artificiales. Estas unidades son dispositivos de cálculo simples que, a partir de un conjunto de entradas provenientes del entorno o de otras neuronas, generan una única salida. Mihaich (2014) distingue tres tipos principales de neuronas artificiales: (ver Figura 10)

- Neuronas de entrada: Estas neuronas reciben señales del entorno, que provienen de sensores u otras partes del sistema.
- Neuronas de salida: Después de procesar la información, estas neuronas envían su señal directamente fuera del sistema, constituyendo las salidas de la red.
- Neuronas ocultas: Estas neuronas reciben estímulos y generan salidas dentro del sistema, sin interactuar con el entorno externo. Es en estas neuronas donde se lleva a cabo el procesamiento básico de la información.

Figura 10

Arquitectura de una RNA.



Nota: De *Redes Neuronales: Conceptos Básicos y Aplicaciones,* por Matich, 2001.

Mihaich (2014) define los siguientes elementos de una neurona artificial con el propósito de replicar las características más importantes de las neuronas biológicas: (ver Figura 11)

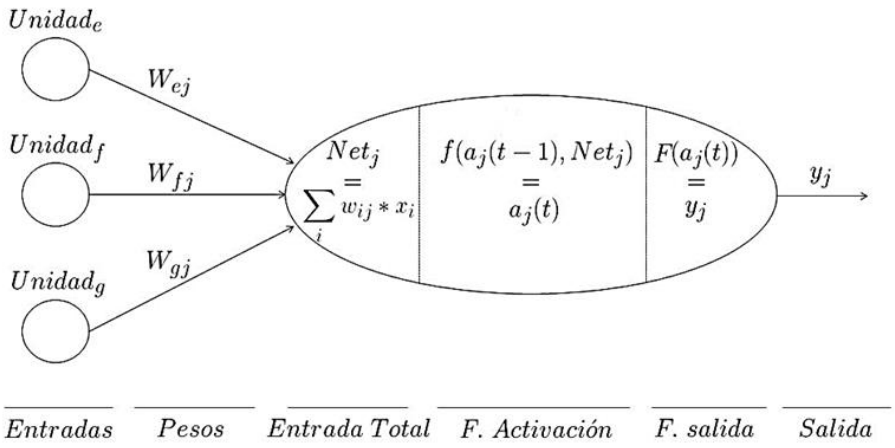
1. Entradas o Estímulos a la Neurona

   Las entradas a las neuronas (x_j) en una RNA son de naturaleza binaria o continua, dependiendo del tipo de red y la tarea en cuestión. Las neuronas de las capas superiores reciben como entradas las salidas de las unidades de capas anteriores, junto con un peso que indica su importancia relativa. Estas salidas también son

binarias o continuas. Cada neurona en la red recibe un conjunto de señales que representan el estado de activación de las neuronas conectadas a ella. Cada conexión entre dos neuronas está ponderada por un peso (w_{ij}).

Figura 11

Modelo genérico de una neurona artificial.



Nota: De *Aplicación de redes neuronales en la clasificación de imágenes*, por Mihaich, 2014.

2. Función de Propagación

La función de propagación define el método mediante el cual se combinan los valores de entrada y los pesos de las conexiones que llegan a una unidad neuronal. Esta regla permite calcular el potencial post-sináptico (Net_j) de la neurona j en un instante t, a partir de las entradas recibidas y sus respectivos pesos asociados, según una función σ_j tal que:

$$Net_j(t) = \sigma_j\left(w_{ij}, x_i(t)\right) \tag{7}$$

La función más común es de tipo lineal y consiste en una suma ponderada de las entradas con los pesos sinápticos correspondientes, como sigue:

$$Net_j(t) = \sum_i w_{ij} * x_i(t) \tag{8}$$

Desde una perspectiva formal, se interpreta como el producto escalar entre un vector de entrada y los pesos de la red, donde w_j^T representa el vector transpuesto que contiene los N pesos de entrada que llegan a la j-ésima neurona.

$$Net_j(t) = \sum_i w_{ij} * x_i(t) = w_j^T \times x \tag{9}$$

En ocasiones, el potencial post-sináptico incluye un umbral de disparo (θ_j). Esta adición se inspira en el comportamiento de las neuronas biológicas, que tienen umbrales internos de activación que modifican el impacto de los estímulos recibidos. En el contexto de las neuronas artificiales, es común incorporar este elemento en la definición de Net_j de la siguiente manera:

$$Net_j(t) = \sum_{i=1}^{N} w_{ij} * x_i(t) + \theta_j \tag{10}$$

Si representamos el umbral mediante el valor $x_0 = 1$ y le asociamos un peso w_0 que determina su efecto (positivo o negativo), la expresión resultante es:

$$Net_j(t) = \sum_{i=0}^{N} w_{ij} * x_i(t) \tag{11}$$

3. Función de Activación o Transferencia

La función de activación o transferencia combina el potencial post-sináptico de la j-ésima neurona (Net_j) con su estado inicial ($a_j(t-1)$) para generar un nuevo estado de activación en función de la información recibida ($a_j(t)$).

$$a_j(t) = f\left(a_j(t-1), Net_j(t)\right) \tag{12}$$

En varios modelos de RNAs, se asume que el estado actual de una neurona no está influenciado por su estado previo, lo que simplifica la expresión anterior a:

$$a_j(t) = f\left(Net_j(t)\right) \tag{13}$$

4. Función de Salida

   Cada neurona U_j está vinculada a una función de salida F que convierte el estado actual de activación $a_j = f\left(Net_j(t)\right)$ una señal de salida $y_j(t)$:

$$y_j = F\left(a_j(t)\right) = F\left(f\left(Net_j(t)\right)\right) \tag{14}$$

   Comúnmente, la función de salida es la función identidad $F(x) = x$, lo que implica que el estado de activación de la neurona se considera como su salida final:

$$y_j = F\left(a_j(t)\right) = a_j(t) = f\left(Net_j(t)\right) = f\left(\sum_{i=0}^{N} w_{ij} * x_i(t)\right) \tag{15}$$

Modos de Operación. En el contexto de las RNAs, el aprendizaje implica que la RNA ajuste sus conexiones en respuesta a la información de entrada, lo que le permite generalizar patrones basados en ejemplos procesados. Por lo general, las RNAs tienen dos modos de funcionamiento distintos: el modo de aprendizaje, donde se ajustan los pesos de la red, y el modo de recuerdo, donde se utilizan los pesos fijos. Sin embargo, hay modelos neuronales donde el aprendizaje y el recuerdo coinciden, lo que permite que la red ajuste sus conexiones dinámicamente durante su ciclo de operación (Mihaich, 2014).

1. Fase de Aprendizaje o Entrenamiento

   Cuando se construye una RNA, se establecen los pesos iniciales de las conexiones, a menudo utilizando valores aleatorios o nulos. Luego, la red se entrena para resolver el problema en cuestión, a través de dos procesos complementarios:

- Proceso de ajuste de los pesos sinápticos: Los pesos de la RNA se adaptan utilizando una regla de aprendizaje que busca minimizar una función de error o coste específica. La evolución del peso en el momento $t+1$ se calcula mediante la siguiente expresión:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{16}$$

  donde $\Delta w_{ij}(t)$ es la variación en el peso generada por la regla de aprendizaje en el instante t. Este proceso suele ser iterativo y concluye cuando la red alcanza el rendimiento deseado o se alcanza un número máximo de ciclos de entrenamiento.
- Proceso de modificación de la arquitectura de la red: Este proceso implica agregar nuevas neuronas a la red con pesos asociados (modelos de redes constructivas), o eliminar neuronas de la red y purgar los pesos asociados (modelos de poda).

Existen dos tipos básicos de reglas de aprendizaje para la actualización de los pesos en las redes neuronales:

- Aprendizaje supervisado: En este enfoque, un agente externo (supervisor o maestro) controla el proceso de entrenamiento proporcionando las respuestas esperadas para ciertas entradas. Los pesos de las conexiones se ajustan iterativamente para minimizar la diferencia entre la salida de la red y la respuesta esperada. El objetivo es encontrar una función desconocida a partir de submuestras de patrones de entrada-salida, mediante la minimización iterativa de una función de error. Los algoritmos de aproximación son de corrección de error, refuerzo o estocásticos.
- Aprendizaje no supervisado o autosupervisado: En este caso, la red ajusta sus pesos sin necesidad de información externa. La red estima una función

de densidad probabilística que describe la distribución de las entradas. La red reconoce correlaciones, categorías o agrupaciones en los datos de entrada sin una salida deseada explícita. Es capaz de realizar tareas como categorización, prototipos (obteniendo ejemplares representativos de las clases), o codificación (generando salidas que representan valores cifrados de las entradas). Una aplicación común es el feature mapping (denominado mapa de características), donde las neuronas de salida representan un mapa de las propiedades de los datos de entrada.

2. Fase de recuerdo, ejecución u operación

   En la fase de recuerdo, la RNA utiliza el conocimiento adquirido durante el entrenamiento para procesar nuevos datos. Este modo de operación se conoce como modo recuerdo, modo de ejecución o modo de operación.

   En redes unidireccionales, las neuronas generan directamente la salida del sistema en respuesta a cada patrón de entrada, sin problemas de estabilidad en el modelo.

   En redes con retroalimentación, se requieren ciertas condiciones para que la red converja a un estado estable, ya que representan sistemas dinámicos no lineales.

Evaluación del Aprendizaje de la red. En la construcción y desarrollo de RNAs, la capacidad de generalización es crucial para evitar la memorización simple de patrones durante el entrenamiento y garantizar respuestas correctas ante datos no vistos anteriormente. Durante el entrenamiento, además del error de aprendizaje, se debe considerar el error de generalización, evaluado en un conjunto de prueba separado del conjunto de entrenamiento.

Es más importante obtener una buena generalización que un error bajo en el conjunto de entrenamiento, ya que esto indica que la red ha capturado correctamente las relaciones entre los datos. El overfitting (denominado sobreajuste) es un fenómeno

observado cuando se entrena la red con un error de aprendizaje muy bajo, pero el error de prueba aumenta debido a un ajuste excesivo a los datos de entrenamiento.

Para evitar el sobreajuste, se utilizan procesos de cross validation (denominado validación cruzada), entrenando y validando la red simultáneamente para determinar el punto óptimo de aprendizaje.

Después del entrenamiento, es necesario evaluar la RNA para determinar su validez práctica. McNelis (2005) propone dos criterios principales para esta evaluación:

Criterios "Dentro de la Muestra". Los criterios "dentro de la muestra" se centran en evaluar la capacidad de la RNA para caracterizar adecuadamente el conjunto de datos utilizado en su entrenamiento. Se han propuesto varias medidas alternativas de rendimiento, y a continuación se analizan las más destacadas.

- Medida de la bondad del ajuste

La medida más comúnmente empleada para evaluar la bondad de ajuste de un modelo es el coeficiente de correlación múltiple, también conocido como coeficiente R^2. Este coeficiente representa la proporción de la varianza en la variable de respuesta que es explicada por el modelo en comparación con la varianza total de la variable de respuesta observada.

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{17}$$

Donde n es el número de observaciones, y_i es el valor observado, $\hat{y}_i$ es el valor estimado del modelo y $\bar{y}$ es la media de los valores observados.

El valor del coeficiente de correlación está en el intervalo de $[0, 1]$, donde un valor más cercano a 1 indica un mejor ajuste del modelo a los datos observados. Es importante destacar que este coeficiente se subestima si el modelo incluye un término constante.

- Criterio de información de Hannan-Quinn

Es posible aumentar progresivamente el valor del estadístico R^2 utilizando un modelo con un número creciente de parámetros. Una manera de ajustar el R^2 es mediante el criterio de información de Hannan-Quinn (1979), que penaliza el rendimiento de un modelo en función del número de parámetros k, que utiliza.

$$hqif = \left[\ln\left(\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{n}\right)\right] + \frac{k\{\ln[\ln(n)]\}}{n} \tag{18}$$

Este criterio se basa en la selección del modelo con el valor más bajo.

- Criterios de información de Akaike y de Schwartz

Otras medidas de validación "dentro de la muestra", que también incorporan términos de penalización, son el criterio de Akaike y el criterio de Schwartz:

$$Akaike = \left[\ln\left(\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{n}\right)\right] + \frac{2k}{n} \tag{19}$$

$$Schwartz = \left[\ln\left(\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{n}\right)\right] + \frac{k * \ln(n)}{n} \tag{20}$$

El criterio de información de Hannan-Quinn tiende a penalizar un modelo con más parámetros más que el criterio de Akaike (1974), aunque no tan rigurosamente como el criterio de Schwartz (1978).

Criterios "Fuera de la Muestra". Para evaluar la capacidad de generalización de las RNAs, es fundamental utilizar criterios "fuera de la muestra", que analizan su habilidad para responder de manera precisa ante la presentación de nuevos patrones a la red.

Se requiere definir una función de pérdida V para estimar el error de predicción generado por el modelo. Las funciones de pérdida más comunes incluyen el error absoluto o cuadrático (en problemas de aproximación de funciones) y el error total de clasificación

derivado de las tablas de contingencia (en problemas de clasificación). Algunas de las principales funciones de error de predicción se presentan en la Tabla 4.

Tabla 4

Funciones de Error de Predicción.

Función de error	Definición
Mean Absolute Error	$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
Mean Squared Error	$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
Root Mean Square Error	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
Normalised Mean Square Error	$NMSE = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sigma_y^2}$
Normalised Root Mean Square Error	$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}}{\sigma_y^2}$

Nota: σ_y^2 es la varianza de los valores reales de y.

Además, se han desarrollado varias variantes del MSE para modelos lineales, como el criterio de validación cruzada generalizada y el criterio de error cuadrático estimado. En el caso de modelos no lineales, la medida más destacada es el error de predicción generalizado (Mihaich, 2014).

Una vez definida la función de error a utilizar, se distinguen distintas reglas de validación "fuera de la muestra", entre las que se destacan:

- Error aparente o de resustitución (resubstitution error).
- División de datos o técnicas de entrenamiento y prueba (test-and-train).
- Modelos de remuestreo (resampling).

2.1.5 Control Adaptativo

El término "adaptativo" implica la capacidad de modificar el comportamiento en respuesta a condiciones cambiantes. Un controlador adaptativo se define como un controlador capaz de ajustar su comportamiento en respuesta a cambios en la dinámica y las perturbaciones del sistema. Esta idea tiene similitudes con la introducción de realimentación en el bucle de control, lo que plantea la cuestión de las diferencias entre el control por realimentación y el control adaptativo.

Se encuentran diversas definiciones de control adaptativo, una de las más comunes es la que lo describe como un tipo especial de control no lineal, en el que el estado del proceso se divide en dos escalas temporales que se desarrollan a ritmos diferentes. La escala lenta corresponde al cambio de los parámetros y, en consecuencia, a la velocidad de cambio de los parámetros del controlador. Por el contrario, la escala rápida corresponde a la dinámica normal del bucle de realimentación (Rodriguez y López, 1996).

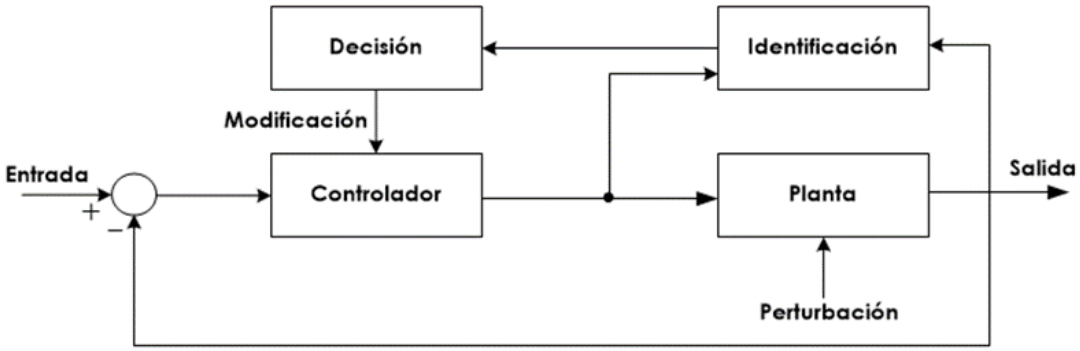
En el diseño de algoritmos de control adaptativo, especialmente en situaciones en las que la dinámica precisa de la planta no está completamente conocida, resulta evidente que los procedimientos iniciales de identificación, toma de decisiones y ajuste no son suficientes para optimizar el rendimiento. Por tanto, se torna imperativo llevar a cabo estos procesos. En el contexto de un sistema de control adaptativo representado en un diagrama de bloque (ver Figura 12), se observa un componente de adquisición de datos que permite la identificación continua de la planta y la medición constante del índice de rendimiento. Una vez obtenida esta información, el índice de rendimiento se compara con el valor óptimo y se toma una decisión con respecto a las modificaciones necesarias en la señal de control. La identificación de la planta se realiza dentro del propio sistema, la adaptación de los parámetros se lleva a cabo en un ciclo de retroalimentación.

Para identificar las características de un sistema, se requiere realizar pruebas o mediciones de entrada y salida y analizar los resultados. La identificación se basa en los

datos recopilados de la planta, y las señales de prueba utilizadas generalmente son señales senoidales de baja amplitud o señales estocásticas.

Figura 12

Diagrama de bloques básico de un control adaptativo.



En la práctica, es esencial variar las señales de entrada en comparación con la señal de escalón unitario convencional. Las entradas normales se eligen como señales de prueba ideales, ya que no generan problemas relacionados con salidas no deseadas ni confusión en las entradas. Sin embargo, la identificación con entradas normales solo es factible cuando dichas señales poseen las características adecuadas en términos de ancho de banda, amplitud y otros parámetros esenciales para su identificación precisa.

Existen dos tipos principales de controladores adaptativos:

- Sistemas con adaptación en lazo cerrado: STR (Selt Tuning Regulator) y MRAC (Model Reference Adaptive Control).
- Sistemas con adaptación en lazo abierto (Ganancia programada)

En la concepción de algoritmos para el diseño de sistemas de control adaptativo, se han formulado diversos enfoques, algunos de los cuales hacen uso de criterios de optimización, mientras que otros prescinden de ellos. En este contexto, se establece la siguiente categorización:

Criterio no óptimo:

- ✓ Asignación de polos y ceros
- ✓ Controladores de tiempo finito
- ✓ Controladores PID

Criterio óptimo:

- ✓ Controladores de mínima varianza (MVR, Minimum Variance Regulators)
- ✓ Controladores predictivos generalizados

2.1.6 Software de Simulación

NI LabVIEW. LabVIEW fue desarrollado por National Instruments, el lenguaje de programación LabVIEW, acrónimo de Laboratory Virtual Instrument Engineering Workbench, se caracteriza por su orientación gráfica. Su naturaleza gráfica lo hace ideal para aplicaciones relacionadas con test y medida (T&M), automatización, control de instrumentos, adquisición y análisis de datos, proporcionando importantes ganancias de eficiencia frente a los lenguajes de programación tradicionales (Bitter et al., 2017). En la Figura 13 se presenta el logo del software LabVIEW de la versión actual.

Figura 13

Logo del Software LabVIEW 2024 Q1.

Nota: De *LabVIEW Wiki*, por Comunidad LabVIEW, 2024 (https://labviewwiki.org/wiki/ LabVIEW_2024_Q1).

Instrumentos Virtuales. El Instrumento Virtual (VI, Virtual Instruments) de LabVIEW es el componente de programación más importante. El VI consiste en un panel frontal que muestra controles e indicadores para la interacción del usuario, un diagrama de

bloques que contiene el código básico del VI, y un ícono que representa visualmente el programa. El panel frontal y el diagrama de bloques definen la funcionalidad del VI, mientras que el ícono, con las conexiones de entrada y salida, proporciona una representación gráfica intuitiva del programa (Bitter et al., 2017).

Panel Frontal (Front Panel). El Panel Frontal de LabVIEW es la interfaz gráfica a través de la cual los usuarios interactúan con el VI. Es la cara visible del programa y muestra controles e indicadores que permiten introducir datos, establecer parámetros y observar resultados. Los elementos del panel frontal incluyen botones, interruptores, diagramas y pantallas digitales que proporcionan interactividad y observación en tiempo real.

Diagrama de Bloques (Block Diagram). El Diagrama de Bloques en LabVIEW es una representación gráfica del flujo y la lógica de un programa. Consiste en nodos, funciones y cables que conectan los elementos. Cada símbolo en el diagrama representa una operación o función específica. Las conexiones entre nodos muestran el flujo de datos o señales. Este enfoque gráfico y visual facilita la comprensión del funcionamiento del VI.

Menús y Paletas (Palettes). LabVIEW tiene dos menús que se utilizan durante la programación. El primer grupo es visible en el “Panel Frontal” y en la ventana de “Diagrama de Bloques”, similar a los menús de otras aplicaciones. El segundo grupo, los llamados menús emergentes, se activan pulsando el botón derecho del ratón.

Los menús emergentes permiten acceder a determinadas paletas, como la paleta “Controles del Panel Frontal” y la paleta “Funciones del Diagrama de Bloques”. Es posible utilizar estas paletas para seleccionar y personalizar objetos, por lo que los menús contextuales son específicos de cada objeto.

LabVIEW también tiene una paleta de Herramientas, que se abre seleccionando "Mostrar Paleta de Herramientas" desde el menú de Windows en el panel frontal o en el diagrama de bloques. Esta paleta contiene herramientas como “Operating” para editar números, “Positioning” para seleccionar y Redimensionar objetos, “Labeling” para editar texto y otras herramientas para tareas especiales. La Figura 14 muestra la Paleta de Herramientas móviles con funciones asociadas, como la herramienta de conexión, la herramienta de depuración y las opciones de ajuste del color. Estas herramientas son esenciales para un desarrollo y depuración eficientes de las aplicaciones LabVIEW (Bitter et al., 2017).

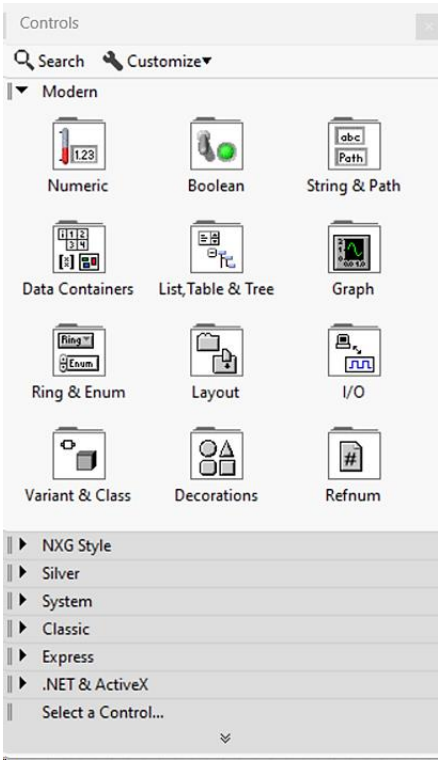
Figura 14

Paleta de Herramientas en LabVIEW.

Paleta de Controles (Controls Palette). La Paleta de Controles contiene varios elementos visuales denominados "controles". Estos elementos representan interfaces de usuario interactivas, como botones, conmutadores, controles deslizantes e indicadores. Los usuarios seleccionan estos elementos de la paleta, lo arrastran y sueltan en el panel de control, donde se utilizan para crear interfaces gráficas intuitivas. Mediante la Paleta de Controles (mostrada en la Figura 15), el aspecto y la funcionalidad de estos elementos se personaliza fácilmente para mejorar la interactividad del programa.

Figura 15

Paleta de Controles de LabVIEW.



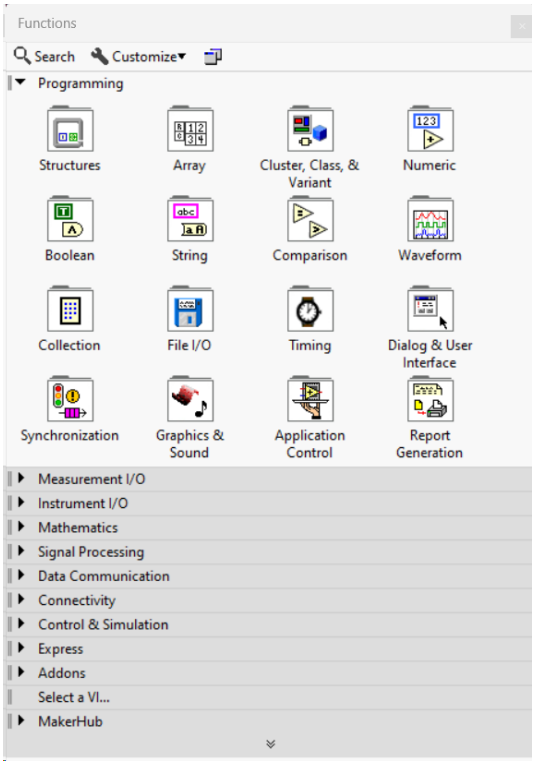
Paleta de Funciones (Functions Palette). La Paleta de Funciones, por su parte, contiene un amplio conjunto de funciones y operaciones predefinidas. Estas funciones abarcan desde operaciones matemáticas y lógicas hasta operaciones de entrada/salida y funciones específicas de LabVIEW. Los usuarios seleccionan estas funciones de la paleta y arrastrarlas y soltarlas en el diagrama de flujo, donde se integran para construir la lógica del programa. En la Figura 16 se muestra la Paleta de Funciones, esta proporciona una amplia gama de herramientas para realizar tareas específicas sin necesidad de una extensa programación manual.

MATLAB. MATLAB (abreviatura de "MATrix LABoratory") es un software ampliamente utilizado en ingeniería y ciencias aplicadas para realizar análisis numérico, cálculos matemáticos y simulaciones de sistemas dinámicos. Desarrollado por MathWorks,

MATLAB ofrece una amplia gama de herramientas y funcionalidades que lo convierten en una opción poderosa para investigadores, ingenieros y científicos en diversas disciplinas.

Figura 16

Paleta de Funciones de LabVIEW.



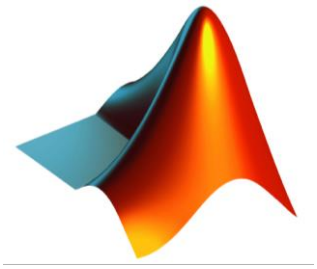
Una de las características distintivas de MATLAB (ver logo en Figura 17) es su lenguaje de programación, que está diseñado para ser intuitivo y expresivo, facilitando la implementación de algoritmos complejos y la manipulación de datos. El lenguaje de programación de MATLAB se basa en matrices, lo que permite realizar operaciones matriciales de manera eficiente y natural.

MATLAB proporciona una variedad de funciones y herramientas específicamente diseñadas para la simulación de sistemas dinámicos. Esto incluye herramientas para la creación y manipulación de modelos matemáticos, así como algoritmos de simulación numérica avanzados para resolver ecuaciones diferenciales ordinarias y parciales.

Además de su capacidad para realizar simulaciones numéricas, MATLAB también ofrece una amplia gama de herramientas para el análisis y visualización de datos. Esto incluye herramientas para la generación de gráficos 2D y 3D, así como funciones para el análisis estadístico y la representación de resultados (*MATLAB - El lenguaje del cálculo técnico*, s.f.).

Figura 17

Logo del Software MATLAB.



Nota: De *The MathWorks Logo is an Eigenfunction of the Wave Equation*, por MathWorks, s.f., (https://www.mathworks.com/company/technical-articles/the-mathworks-logo-is-an-eigenfunction-of-the-wave-equation.html).

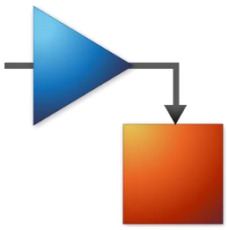
Simulink. Simulink, integrado en MATLAB, es una herramienta de simulación basada en bloques que permite modelar sistemas dinámicos mediante diagramas de bloques. Permite diseñar modelos de sistemas físicos, controladores y algoritmos de manera intuitiva (*Simulación y diseño basado en modelos con Simulink*, s.f.). Algunas características destacadas de Simulink son: (ver logo en Figura 18)

- ✓ Interfaz gráfica intuitiva que permite la construcción de modelos mediante la conexión de bloques funcionales, lo que facilita la representación visual de sistemas complejos.
- ✓ Bibliotecas de bloques predefinidos para una amplia gama de aplicaciones, incluyendo sistemas eléctricos, mecánicos, hidráulicos, de control y de procesamiento de señales.

- ✓ Capacidades avanzadas de simulación que permiten simular el comportamiento dinámico de los sistemas modelados y analizar su respuesta en el dominio del tiempo y la frecuencia.

Figura 18

Logo de herramienta de MATLAB, Simulink).



Nota: De *File: Simulink Logo (non-wordmark).png*, por Wikipedia contributors, s.f., (https://en.m.wikipedia.org/wiki/ File:Simulink_Logo_(non-wordmark).png).

2.1.7 Hardware

NI DAQ USB 6211. La tarjeta de adquisición de datos NIDAQ USB 6211 (ver Figura 19) de National Instruments desempeña un papel fundamental en la instrumentación y el control de sistemas de ingeniería. Esta tarjeta se destaca por sus capacidades avanzadas de adquisición de datos, conversión analógico-digital de alta precisión y generación de señales de control (*USB-621x User Manual*, 2009).

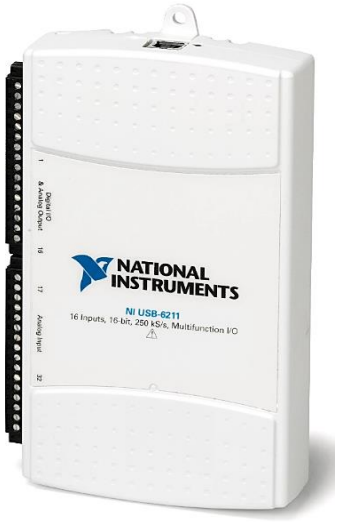
Como parte de la familia de productos NI-DAQmx, el NIDAQ USB 6211 ofrece una amplia gama de funciones de adquisición de datos, incluyendo entradas y salidas analógicas y digitales, así como módulos de temporizador y contador. Esta versatilidad lo convierte en una herramienta indispensable para aplicaciones de supervisión y control en tiempo real en ingeniería.

En el ámbito de la ingeniería de control, la NIDAQ USB 6211 se utiliza para recibir señales de sensores y transmisores y para generar señales de control para actuadores y dispositivos de respuesta. Su capacidad de integración con software como LabVIEW y su

compatibilidad con varios lenguajes de programación lo hacen ideal para aplicaciones de control adaptativo no lineal como el sistema de tanques en cascada propuesto en este estudio.

Figura 19

Tarjeta de adquisición de datos NI USB DAQ 6211.



Nota: De *Tarjeta de adquisición de datos NI USB-6211, multifunción, Serie M, DAQ*, por AliExpress, s.f. https://es.aliexpress.com/item/1005004556161762.html

NI-DAQmax. NI-DAQmx es un conjunto de drivers y librerías de software desarrollado por National Instruments específicamente para la gestión de dispositivos y la adquisición de datos. Esta plataforma de software proporciona numerosas funciones e interfaces de programación de aplicaciones (API, application programming interface) para conectarse a dispositivos de adquisición de datos, lo que la convierte en una herramienta indispensable para la configuración de sistemas, la adquisición y la gestión de datos en aplicaciones científicas y de ingeniería.

En el contexto de este estudio, la combinación de NI-DAQmx con la placa de adquisición de datos NIDAQ USB 6211 proporciona una interfaz versátil y potente para

adquirir señales analógicas y digitales y generar señales de control (National Instruments Corporation, 2009).

Además, NI-DAQmx proporciona amplia documentación, bibliotecas de funciones y ejemplos de código que simplifican el desarrollo de aplicaciones de control y adquisición de datos.

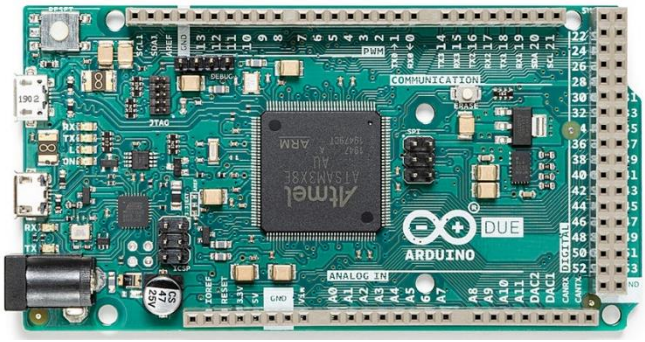
Arduino DUE. El Arduino Due (ver placa de desarrollo en la Figura 20) representa un hito significativo en la evolución de las plataformas de desarrollo de sistemas embebidos. Basado en la CPU (Central Processing Unit) Atmel SAM3X8E ARM Cortex-M3, el Due es el primer Arduino que integra un microcontrolador de núcleo ARM (Advanced RISC Machine) de 32 bits. Con 54 pines de entrada/salida digital, 12 entradas analógicas, cuatro UARTs (Universal Asynchronous Receiver / Transmitter), dos conversores digital-analógico (DAC, Digital-to-Analog Converter), y una amplia gama de otras interfaces, proporciona una plataforma robusta y adaptable para una variedad de aplicaciones.

Es importante destacar que el Arduino Due opera a 3.3V, lo que lo diferencia de otras placas Arduino que operan a 5V. Esto implica consideraciones específicas en cuanto a la compatibilidad con otros dispositivos y la protección contra sobrevoltajes. Además, la selección adecuada de la fuente de alimentación es fundamental para un funcionamiento estable y seguro del sistema (*Arduino Due*, s.f.).

El Due se programa utilizando el software Arduino IDE (Integrated Development Environment), lo que simplifica el proceso de desarrollo y depuración del código. Además, su arquitectura basada en ARM ofrece un rendimiento y una eficiencia superiores en comparación con las plataformas tradicionales de 8 bits. Las características principales de la placa de desarrollo se observan en la Tabla 5.

Figura 20

Placa de desarrollo Arduino Due.



Nota: De *Arduino Online Shop*, por Arduino due, s.f., (https://store-usa.arduino.cc/products/arduino-due?selectedStore=us).

Tabla 5

Especificaciones Técnicas de placa Arduino Due.

Especificaciones Técnicas	
Microcontrolador	AT91SAM3X8E
Voltaje de operación	3.3V
Voltaje de alimentación	7-12V
Pines digitales E/S	54 (de los cuales 12 proporcionan salida PWM)
Pines de entrada analógica	12
Pines de salida analógica	2(DAC)
Memoria Flash	512 KB disponibles para las aplicaciones de usuario
SRAM	96 KB (dos bancos: 64 KB y 32 KB)
Velocidad del reloj	84 MHz

Nota: Elaborado a partir de *Arduino Online Shop*, por Arduino due, s.f., (https://store-usa.arduino.cc/products/arduino-due?selectedStore=us).

Sensor de Nivel de Agua con Salida RS485. El sensor de nivel de agua de 10-30 VCC con transmisor de nivel integrado (ver Figura 21), tipo Modbus, se caracteriza por su innovador enfoque en la medición precisa de niveles de agua. Su diseño compacto y resistente emplea un chip de silicona de difusión para encapsular el núcleo lleno de aceite de silicona en una carcasa de acero inoxidable. Este enfoque garantiza la protección del sensor y permite el contacto suave del líquido con el diafragma sensorial.

Figura 21

Sensor de Nivel de Agua con salida RS485.

Nota: De *Sensor de nivel de agua con transmisor de nivel de líquido RS485 Modbus RTU 24VDC con cable de 5 m*, por AliExpress, s.f., (https://es.aliexpress.com/item/1005004775126836.html).

La amplia gama de aplicaciones del sensor lo convierte en un componente esencial en entornos como plantas de agua, plantas de tratamiento de aguas residuales, suministro de agua urbana, piscinas, pozos, minas, tanques de agua, entre otros. Su circuito completamente cerrado con funciones de resistencia a la humedad, condensación y fugas garantiza un rendimiento confiable en condiciones variadas. Las especificaciones técnicas se presentan en la Tabla 6.

Tabla 6

Especificaciones técnicas del Sensor de Nivel de Agua con salida RS485.

Especificaciones técnicas	
Voltaje de operación	10 – 30 VDC
Rango de medición	0 – 200 metros
Precisión	0.5% FS
Temperatura de trabajo	20 – 60 °C
Compensación de temperatura	10 – 60 °C
Clase de protección	IP68

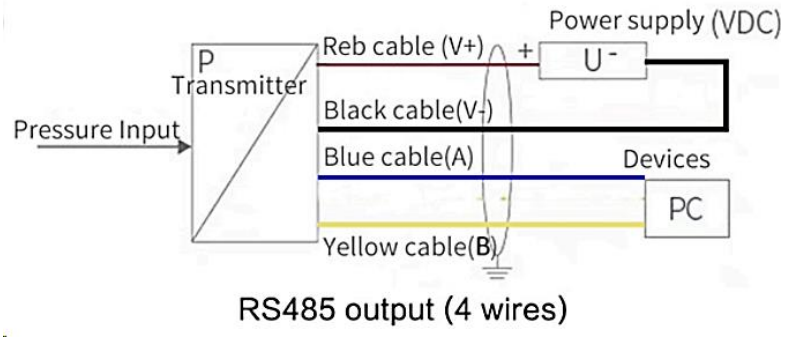
Nota: Elaborado a partir de Datasheet público, s.f.

Este sensor está compuesto por cuatro cables en total (ver Figura 22). Dos de estos cables se emplean para la alimentación eléctrica, mientras que los otros dos se destinan a la conexión de la Norma RS485. Esta norma se utiliza para establecer la comunicación del sensor mediante el Protocolo Modbus RTU.

- Cable rojo: Alimentación para el sensor (24 VDC).
- Cable negro: Referencia de tierra (GND).
- Cable azul: Línea de comunicación A.
- Cable amarillo/blanco: Línea de comunicación B.

Figura 22

Diagrama de conexión del Sensor de nivel de agua.



Nota: De Datasheet público, s.f.

Sensor de Proximidad Ultrasónico de Allen Bradley. El sensor de proximidad ultrasónico Allen Bradley modelo 873M-D18AV800-D4 (ver Figura 23) es un componente crucial en el diseño de sistemas de control automatizado, especialmente en entornos industriales donde la precisión y la confiabilidad son primordiales. Este sensor, fabricado por Allen Bradley, líder en tecnologías de automatización industrial, es altamente reconocido por su robustez y eficiencia en la detección de distintos tipos de materiales en aplicaciones diversas. Las características del sensor se observan en la Tabla 7.

Figura 23

Sensor de proximidad ultrasónico 873M-D18AV800-D4.

Nota: De *Cut Sheet 873M-D18AV800-D4*, s.f., (https://configurator.rockwellautomation.com/api/Product/873M-D18AV800-D4/cutsheet).

Tabla 7

Especificaciones técnicas de Sensor de proximidad ultrasónico de Allen Bradley 873M-D18AV800-D4.

Especificaciones técnicas	
Estilo de familia	Boletín 873M Uso general Cilíndrico 18mm
Dirección de detección	Recta
Diámetro del barril	18mm
Rango de detección	De 70 mm a 800 mm
Voltaje de alimentación	15 – 30 VDC
Configuración de salida	Voltaje analógico (0 - 10VDC)

Nota: Adaptada de *Cut Sheet 873M-D18AV800-D4*, s.f., (https://configurator.rockwellautomation.com/api/Product/873M-D18AV800-D4/cutsheet).

Este sensor de proximidad consta de un total de cuatro cables. Dos de estos cables se utilizan para la alimentación eléctrica, mientras que, de los otros dos cables, uno corresponde a la señal analógica y el otro no se utiliza y permanece desconectado:

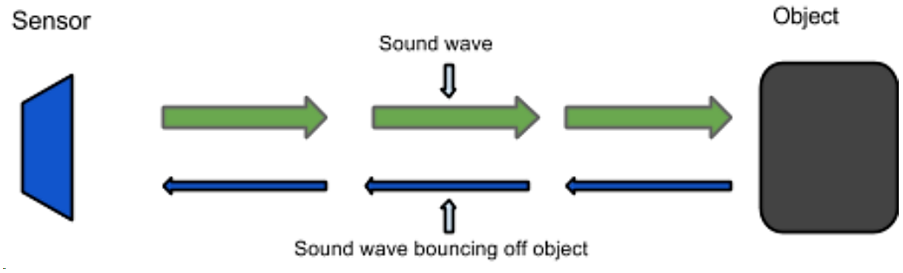
- Cable rojo: Alimentación para el sensor (24 VDC).
- Cable azul: Referencia de tierra (GND).
- Cable negro: Señal analógica (0-10V).
- Cable blanco: No se conecta.

Principio de Funcionamiento. Los sensores ultrasónicos utilizan ondas sonoras para medir la distancia entre el sensor y el objeto más cercano en su trayectoria. Su principio de funcionamiento es similar al de los sensores de sonido, pero con la diferencia de que funcionan a frecuencias más altas que las audibles para el ser humano.

El proceso comienza cuando el sensor emite una onda sonora de una frecuencia determinada y detecta el eco que se produce cuando la onda se refleja en un objeto (ver la Figura 24). Se registra el tiempo transcurrido desde la emisión hasta el retorno de la onda sonora. La distancia se calcula a partir de la ecuación $d = v * t$, basada en la velocidad del sonido y el tiempo transcurrido.

Figura 24

Gráfico de onda del sonido emitida y reflejada.

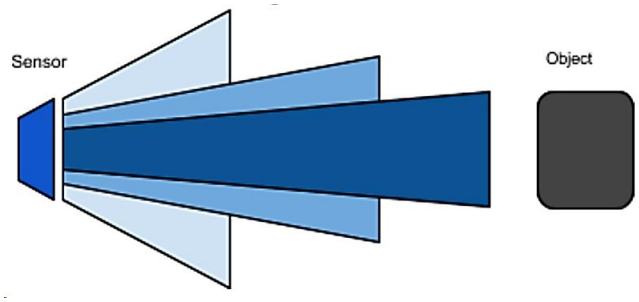


Nota: De Datasheet público, s.f.

La velocidad del sonido se determina teniendo en cuenta diversas condiciones atmosféricas, como la temperatura, la humedad y la presión. Es importante tener en cuenta que estos sensores tienen un cono de detección cuyo ángulo varía con la distancia, como se muestra en la Figura 25. La orientación del objeto con respecto al sensor también afecta al rendimiento de la detección. En los casos en que el objeto no tenga una superficie plana con respecto al sensor, es posible que la onda sonora se refleje de tal forma que no regrese al sensor (Datasheet público, s.f.).

Figura 25

Gráfico de la variación del cono de detección.



Nota: De Datasheet público, s.f.

Sensor de Flujo YF-S201. El sensor de flujo de agua YF-S201 es un componente esencial en sistemas de monitoreo y control de flujo de líquidos, especialmente diseñado para aplicaciones donde se requiere precisión y fiabilidad en la medición del caudal de agua. Este sensor, conocido por su diseño compacto y su capacidad para funcionar en una amplia gama de condiciones, es una opción popular en proyectos de automatización y sistemas de control de procesos. Las especificaciones técnicas del sensor se presentan en la Tabla 8.

Tabla 8

Especificaciones técnicas del Sensor de flujo YF-S201.

Especificaciones técnicas	
Voltaje de operación	5V-18V DC
Consumo de corriente	15mA (5V)
Salida	Onda cuadrada pulsante
Rango de flujo	1-30L/min
Factor de conversión	7.5
Presión de trabajo máximo	1.75MPa (17bar)
Temperatura de funcionamiento	-25°C a 80°C

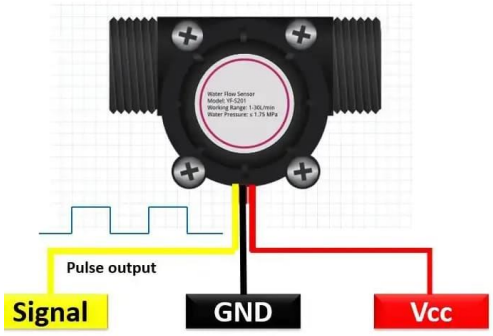
Nota: Elaborado a partir de *Sensor de flujo de agua 1/2" YF-S201*, por Naylamp Mechatronics, s.f., (https://naylampmechatronics.com/sensores-liquido/108-sensor-de-flujo-de-agua-12-yf-s201.html).

Este sensor de flujo de agua consta de tres cables, en la Figura 26 se indica la descripción de cada cable.

- Cable rojo: Alimentación para el sensor (5 VDC).
- Cable negro: Referencia de tierra (GND).
- Cable amarillo: Señal de tren de pulsos (0-5V).

Figura 26

Diagrama de conexión del Sensor de flujo de agua YF-S201.



Nota: De *Sensor de Flujo YF-S201: Medición de Caudal con Arduino* por S.A. Giraldo, 2023 (https://controlautomaticoeducacion.com/arduino/sensor-de-flujo-yf-s201-medicion-de-caudal-con-arduino/#Interfaz_del_YF-S201_con_Arduino).

Principio de Funcionamiento. Este sensor de flujo de agua incorpora dos componentes esenciales para su funcionamiento: un rotor, también conocido como rueda de turbina, y un sensor de efecto Hall.

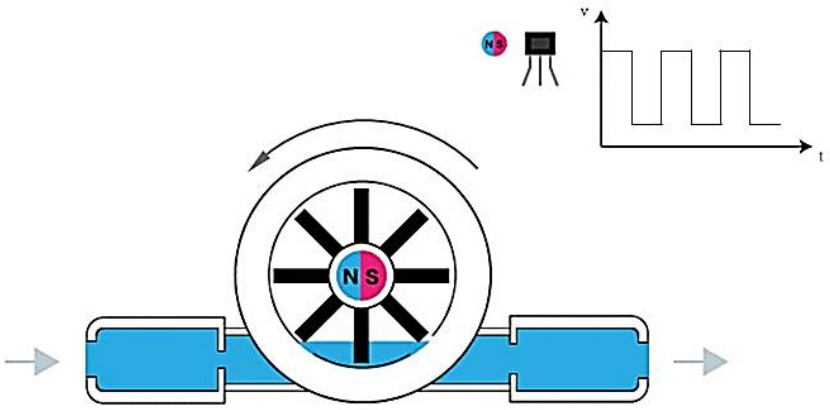
En su estructura, el YF-S201 alberga una pequeña turbina que se ve afectada por el flujo de agua que entra y circula a través del sensor. Cuando el agua impacta la rueda de turbina, esta comienza a girar. La velocidad de rotación de la rueda de turbina está directamente vinculada a la velocidad del flujo de agua que atraviesa el sensor.

El sensor YF-S201 opera mediante el principio del Efecto Hall, que se basa en la generación de un voltaje cuando un conductor eléctrico atraviesa un campo magnético. En este caso, la turbina incorpora un pequeño imán que, al girar, modifica el campo magnético.

El sensor de efecto Hall identifica estas alteraciones y, en cada revolución completa de la turbina, genera un pulso en su pin de salida, este comportamiento se muestra en la Figura 27. En otras palabras, la cantidad de pulsos que se registra en el pin de salida es directamente proporcional a la velocidad de rotación de la turbina y, por ende, al flujo de agua (Castaño Giraldo, s.f.).

Figura 27

Generación de pulsos por Efecto Hall.



Nota: De *Sensor de Flujo YF-S201: Medición de Caudal con Arduino* por S.A. Giraldo, 2023 (https://controlautomaticoeducacion.com/arduino/sensor-de-flujo-yf-s201-medicion-de-caudal-con-arduino/#Interfaz_del_YF-S201_con_Arduino).

Para utilizar el sensor, se implementan interrupciones para la lectura de los pulsos generados por el flujo del agua. En Arduino, se emplea la siguiente fórmula para calcular el caudal (Q) en las unidades correspondientes:

$$Q_{l/s} = \frac{n(pulsos)}{K}\frac{1000}{millis()} \tag{21}$$

También

$$Q_{l/min} = Q_{l/s} \times 60 \tag{22}$$

Donde K es el factor de conversión entre la Cantidad de Pulsos y Litros que han pasado por el sensor.

Válvula Proporcional de 2 Vías TFV4-304. Para controlar el caudal de entrada al Tanque 2, se empleó la válvula proporcional de 2 vías TFV4-304 (ver Figura 28). Esta elección se fundamentó en su destacada precisión, repetitividad, frecuencia y mínima histéresis.

Figura 28

Válvula proporcional de 2 vías TFV4-304.

Nota: De *proportional valve 0-10V/4-20mA modulating valve with 6Nm actuator*, s.f., (http://www.china-electricvalves.com/product/255-en.html).

Estas válvulas, caracterizadas por su alta calidad, se integran con sistemas electrónicos avanzados y operan en circuitos de lazo cerrado. Aunque su costo es superior, la utilización de válvulas proporcionales sustituye eficazmente a muchos sistemas que requerirían servo válvulas. En la Tabla 9 se presentan las características técnicas de la válvula proporcional.

La válvula proporcional está equipada con seis cables (ver Figura 29), de los cuales dos están destinados a la alimentación eléctrica, otros dos se utilizan para el control mediante voltaje o corriente, y finalmente, se cuenta con un cable de realimentación. Este

último proporciona una señal de voltaje que es proporcional al porcentaje de apertura de la válvula, permitiendo así monitorizar su posición de manera precisa.

- Cable marrón: Alimentación para la válvula (24 VDC).
- Cable negro: Referencia de tierra (GND).
- Cable anaranjado: Control por corriente (4-20 mA).
- Cable amarillo: Control por voltaje (0-10V).
- Cable verde: Señal de realimentación.
- Cable rojo: No se conecta.

Tabla 9

Especificaciones técnicas de la válvula proporcional TFV4-304.

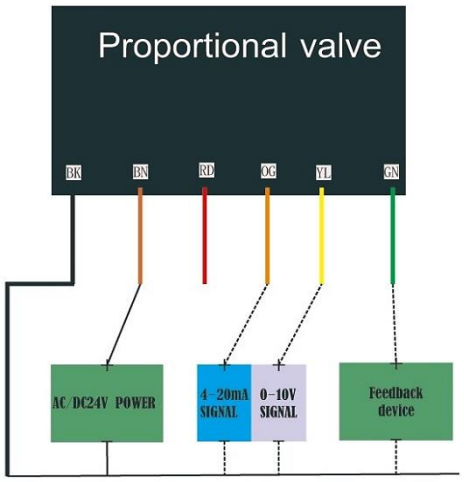
Especificaciones técnicas	
Alimentación	AC/DC24V ±15%
Señal de control	0-10V (4-20mA)
Sentido de marcha	CW o CCW (ajustable)
Potencia de trabajo	6VA
Tiempo de apertura/cierre	50-120 segundos
Tiempo de vida	70000 veces
Rotación del actuador	90°
Máxima torque	6Nm
Clase de protección	IP54
Tipo de válvula	2 vías
Temperatura del líquido	2-90°C

Nota: Elaborado a partir de *proportional valve 0-10V/4-20mA modulating valve with 6Nm actuator*, s.f., (http://www.china-electricvalves.com/product/255-en.html).

Bomba de Agua Pedrollo PKM60. Para transferir el contenido de agua desde el Tanque 3 hasta el Tanque 1, se utilizó una bomba de agua PKm60 de la marca Pedrollo, como se ilustra en la Figura 30. Estas bombas están específicamente diseñadas para el bombeo de agua, gracias a su diseño hidráulico que les permite generar presiones elevadas con una baja potencia de accionamiento del motor.

Figura 29

Diagrama de conexión de la válvula proporcional.



Nota: De *proportional valve 0-10V/4-20mA modulating valve with 6Nm actuator*, s.f., (http://www.china-electricvalves.com/product/255-en.html).

Figura 30

Electrobomba de agua marca Pedrollo PKM60-1.

Nota: De *Electrobomba Superficie Horizontal Pedrollo 0.5HP-PKm 60*, s.f., (https://autosolar.pe/bombas-perifericas/electrobomba-superficie-horizontal-pedrollo-05hp-pkm-60?srsltid=AfmBOopzCt_PoWOnVcK9FyZsS-gnKqocyeQJ16-Zn5zduMFwM7ljYrLB).

Para la utilización de una bomba de agua, es esencial tener en cuenta diversos aspectos clave, tales como el tipo de agua a ser bombeada, su origen y la potencia necesaria. Este modelo específico de bomba de agua se recomienda para el bombeo de agua limpia, exenta de partículas abrasivas y líquidos que no sean químicamente agresivos

para los materiales que conforman la bomba. La elección de esta bomba se basó en su confiabilidad, facilidad de uso y su ventaja económica. En la Tabla 10 se detallan las especificaciones técnicas más relevantes.

Tabla 10

Especificaciones Técnicas de la Bomba de agua PKm60 Pedrollo.

Especificaciones técnicas	
Caudal	5 – 40 L/min
Altura manométrica	5 – 40 m
Temperatura del líquido	-10°C hasta 60°C
Temperatura ambiente	Hasta +60°C
Presión máxima en el cuerpo de la bomba	6 bar
Potencia	0.5 HP (0.37KW)
Motor eléctrico	Monofásico 220V – 60Hz con protección térmica incorporada en el bobinado
Corriente nominal	2.6 A
Protección	IP X4

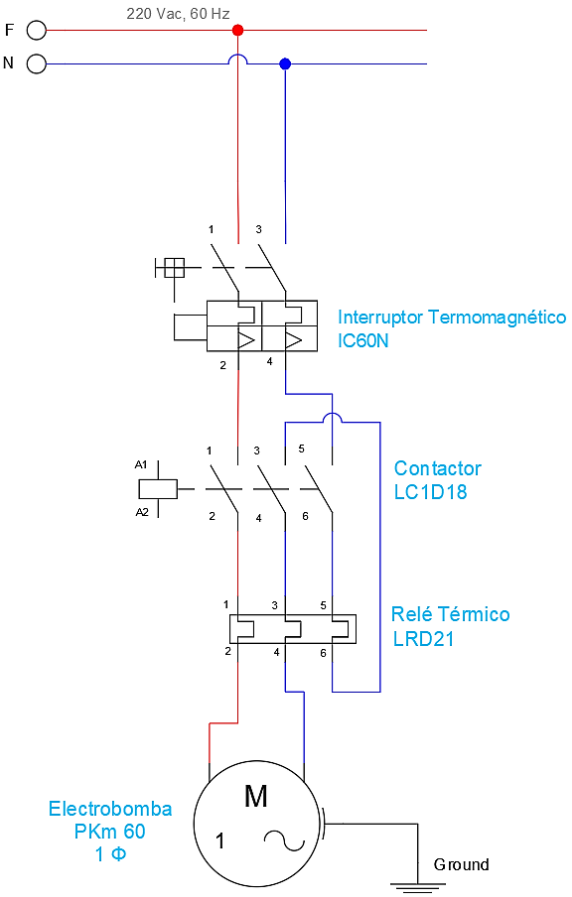
Nota: Elaborado a partir de *Electrobomba Superficie Horizontal Pedrollo 0.5HP-PKm 60*, s.f., (https://autosolar.pe/bombas-perifericas/electrobomba-superficie-horizontal-pedrollo-05hp-pkm-60?srsltid=AfmBOopzCt_PoWOnVcK9F yZsS-gnKqocyeQJ16-Zn5zduMFwM7ljYrLB).

La bomba de agua Pedrollo PKM60 es una bomba centrífuga eléctrica que requiere una conexión adecuada para funcionar de manera eficiente. A continuación, se describe el esquema de conexión eléctrica típico: (ver Figura 31)

- Alimentación Eléctrica: La bomba debe conectarse a una fuente de alimentación de 220V AC (monofásica).
- Contactor: Utilizado para controlar el encendido y apagado de la bomba de agua. Existen contactores operados con una bobina de 24V o 220V AC.
- Relé Térmico: Protege la bomba contra sobrecargas y evita el sobrecalentamiento del motor.
- Conexión a Tierra: Es esencial conectar el motor de la bomba a tierra para evitar riesgos eléctricos.

Figura 31

Diagrama de conexión para el control de una electrobomba monofásica.



2.2 Marco Conceptual

2.2.1 Modelo Hammerstein-Wiener

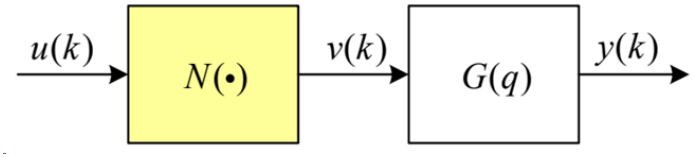
La representación óptima de la mayoría de los sistemas dinámicos se consigue utilizando modelos no lineales. La categoría ampliamente estudiada de modelos no lineales basados en bloques incluye modelos consistentes en una combinación de sistemas LTI y no linealidades estáticas sin memoria (Gómez y Baeyens, 2004). En esta clasificación, se distinguen dos estructuras de modelo más comunes:

- Modelo Hammerstein: Se caracteriza por una no linealidad estática precedida de un sistema lineal representado por la ecuación $\mathrm{y} = \mathrm{G(s)f(u)}$. Describe un modelo lineal controlado por un actuador no lineal, como un actuador saturante.
- Modelo de Wiener: Consiste en un modelo lineal seguido de una no linealidad estática representada por la ecuación $\mathrm{y} = \mathrm{f(G(s)u)}$. Este modelo describe un sistema lineal con un sensor de salida no lineal (Schoukens y Ljung, 2019).

Modelo Hammerstein. Los modelos Hammerstein consisten en conectar en cascada un bloque no lineal $N(.)$ seguido de un bloque lineal $G(q)$, como se muestra en la Figura 32.

Figura 32

Estructura del Modelo Hammerstein.



En el modelo Hammerstein, la señal intermedia $v[k]$ se obtiene mapeando $u[k]$ a través de la función $N(.)$, es decir:

$$v(k) = N(u[k]) \tag{23}$$

Luego,

$$v[k-i] = N(u[k-i]), i = 1, \dots, n_u \tag{24}$$

Si un bloque dinámico lineal se representa mediante un modelo ARX derivado de las medidas de entrada $v[k-i]$ y salida $y[k-j]$, se escribe:

$$y[k] = \sum_{i=1}^{n_a} a_i y[k-i] + \sum_{j=1}^{n_b} b_j v[k-j] \tag{25}$$

Donde:

n_a: Retardo de la salida del modelo ARX.

n_b: Retardo de la entrada del modelo ARX.

a_i: Coeficientes del término autoregresivo del modelo ARX.

b_j: Coeficientes del término de entrada exógeno del modelo ARX.

Realizando las sustituciones respectivas se obtiene:

$$y[k] = \sum_{i=1}^{n_a} a_i y[k-i] + \sum_{j=1}^{n_b} b_j N(u[k-j]) \tag{26}$$

En realidad, la señal intermedia $v[k]$ no está disponible. Como se ha mencionado anteriormente, normalmente se representa el bloque no lineal del modelo de Hammerstein en forma polinómica como una función de la entrada y la salida del bloque, $u[k]$ y $v[k]$, respectivamente (Alvarado Tabacchi, 2018).

En esta situación, la salida del bloque no lineal adopta la siguiente configuración:

$$v[k] = p_1 u[k] + p_2 u^2[k] + \cdots + p_m u^m[k] \tag{27}$$

Donde k es el instante de tiempo, $v[k]$ es la pseudo-salida del bloque no lineal, $u[k]$ es la variable de entrada y p_i $(i = 1, \dots, m)$ son los coeficientes del polinomio, donde m es el grado de no linealidad del modelo Hammerstein.

La formulación paramétrica del modelo Hammerstein se expresa como sigue:

$$A(q^{-1})y[k] = B(q^{-1}) \sum_{i=1}^{m} p_i u^i[k-d] + \varepsilon[k] \tag{28}$$

Donde d es el tiempo muerto.

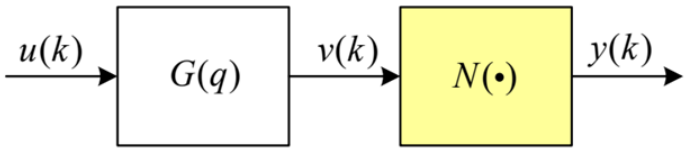
Sustituyendo la ecuación (27) en (28), se obtiene:

$$A(q^{-1})y[k] = B(q^{-1})v[k-d] + \varepsilon[k] \tag{29}$$

Modelo Wiener. La disposición de los bloques en los modelos Wiener es opuesta a la disposición de los bloques en los modelos Hammerstein, como se muestra en la Figura 33.

Figura 33

Estructura del Modelo Wiener.



Cabe señalar que en estos modelos algunos algoritmos de identificación requieren que la no linealidad estática sea invertible. Esta limitación se debe a que la señal que conecta los dos subsistemas no es medible. La salida del modelo Wiener se obtiene mapeando la señal intermedia $v[k]$ a través de la función $N(.)$ de la siguiente manera:

$$y[k] = N(v[k]) \tag{30}$$

La señal $v[k]$ no está disponible directamente, se estima invirtiendo la función $N(.)$. Por lo tanto, es muy importante para la estimación del modelo que esta función sea invertible.

Un bloque dinámico lineal es representado por un modelo ARX definido por el par de entrada $u[k]$ y salida $v[k]$ (Alvarado Tabacchi, 2018), como se muestra en la siguiente ecuación:

$$v[k] = \sum_{i=1}^{n_a} a_i v[k-i] + \sum_{j=1}^{n_b} b_j u[k-j] \tag{31}$$

Reemplazando la ecuación (31) en (30) se obtiene:

$$y[k] = N\left(\sum_{i=1}^{n_a} a_i v[k-i] + \sum_{j=1}^{n_b} b_j u[k-j]\right) \tag{32}$$

Si la función $N(.)$ es invertible, condición necesaria del modelo Wiener, se cumple:

$$N(.)^{-1} = N^{-1}(.) \tag{33}$$

Si se considera un retraso i, tal que $i = 1, \dots, n_a$ se escribe la siguiente relación:

$$v[k-i] = N^{-1}(y[k-i]) \tag{34}$$

Reemplazando la ecuación (34) en (32) se obtiene:

$$y[k] = N\left(\sum_{i=1}^{n_a} a_i N^{-1}(y[k-i]) + \sum_{j=1}^{n_b} b_j u[k-j]\right) \tag{35}$$

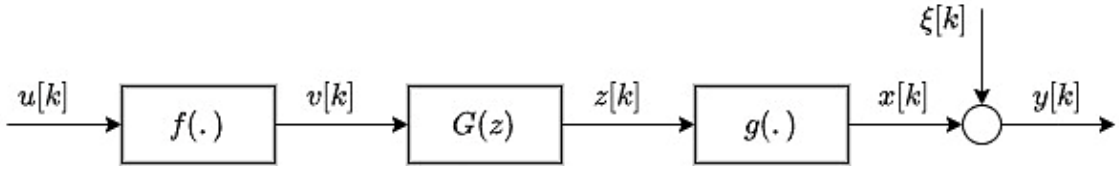
Modelo Hammerstein-Wiener. Para algunas aplicaciones, resulta más adecuada una combinación de los modelos descritos. La Figura 34 muestra un esquema en el que un bloque lineal se coloca entre dos bloques no lineales, uno a la entrada y otro a la salida, el llamado modelo Hammerstein-Wiener. En muchos casos se supone que las no linealidades a la entrada $f(.)$ y a la salida $g(.)$ son funciones polinómicas similares a las descritas en la Ecuación (27), aunque esta condición no es necesaria (Alvarado Tabacchi, 2018).

En general, la principal dificultad del problema de identificación radica, por un lado, en que las señales internas $v[k]$ y $z[k]$ no son directamente medibles y, por otro lado, en la forma de las no linealidades de entrada y salida, que suelen ser parcial o totalmente desconocidas. Además, la presencia de ruido $\xi(k)$ plantea un problema adicional en el

proceso de identificación. Por simplicidad, se suele suponer que el ruido es ergódico[1], ya que $\gamma_k = 0$ para $k \neq 0$ y tiene un valor estacionario igual a cero (Brouri et al., 2014).

Figura 34

Estructura del modelo Hammerstein-Wiener.



Nota: Adaptado de *Identification of Nonlinear Systems Structured by Hammerstein-Wiener Model*, por A. Brouri et al., 2014.

2.2.2 Metodología de Identificación Propuesta

Proceso no Lineal de Tipo Hammerstein-Wiener. En esta investigación, se considera un proceso no lineal Hammerstein-Wiener de una sola entrada y una sola salida, como se muestra en la Figura 35(a). Este proceso se representa mediante las siguientes ecuaciones:

$$v(t) = f\big(u(t)\big) = p_1 u(t) + p_2 u^2(t) + \cdots + p_m u^m(t) \tag{36}$$

$$z(t) = \sum_{i=1}^{n_a} a_i z(t-i) + \sum_{j=1}^{n_b} b_j v(t-j-T_d) \tag{37}$$

$$z(t) = g^{-1}\big(y(t)\big) = q_1 y(t) + q_2 y^2(t) + \cdots + q_r y^r(t) \tag{38}$$

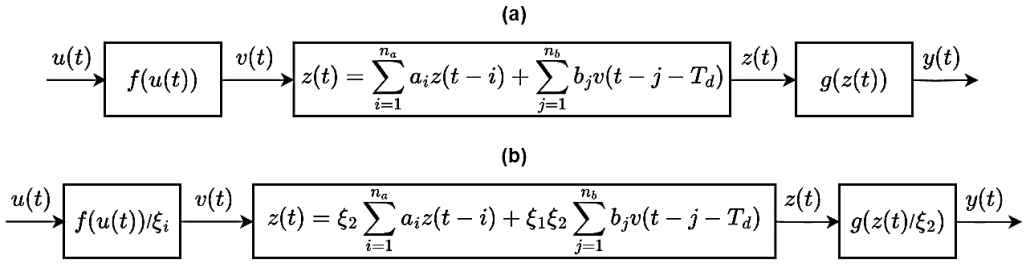
Aquí, $u(t)$ e $y(t)$ denotan la entrada y la salida del proceso, respectivamente. Las ecuaciones (36), (37) y (38) representan la función estática no lineal de entrada, el subsistema dinámico lineal y la función estática no lineal de salida, respectivamente. Las variables intermedias $v(t)$ y $z(t)$ son la salida de la función estática no lineal de entrada y

[1] Propiedad de un proceso estocástico en el que sus propiedades estadísticas, como la media y la varianza, se estiman con precisión a partir de una única realización observada durante un tiempo suficientemente largo.

la salida del subsistema dinámico lineal, respectivamente. T_d es el retardo de entrada (Park et al., 2006).

Figura 35

Dos procesos no lineales Hammerstein-Wiener equivalentes.



Nota: Adaptado de Modeling of Hammerstein − Wiener processes with special input test signals, por H.C. Park et al., 2006.

La función estática no lineal de salida se representa mediante la forma polinómica inversa de $g^{-1}(.)$ para simplificar el procedimiento de identificación, se supone que la función estática no lineal de salida es estrictamente monótona e invertible. Sin embargo, la función no lineal de entrada tiene multiplicidades de entrada, lo que significa que diferentes valores de entrada producen la misma salida (Park et al., 2006). La cuestión principal de la identificación del proceso es estimar los parámetros a_i , b_j , T_d , p_i $(i = 1, 2, \dots, m)$, y q_i $(i = 1, 2, \dots, r)$ a partir de los datos de salida muestreados $y(t)$ y los datos de entrada definidos por $u(t)$.

Activación del Proceso. Park et al. (2006) proponen tres señales de prueba específicas para activar el proceso no lineal de Hammerstein-Wiener, permitiendo separar completamente los tres problemas de identificación: la función estática no lineal de entrada, el subsistema dinámico lineal y la función estática no lineal de salida.

1. La primera señal de prueba, u_b, es una señal binaria (de dos pasos) con dos valores: cero y una constante diferente de cero.

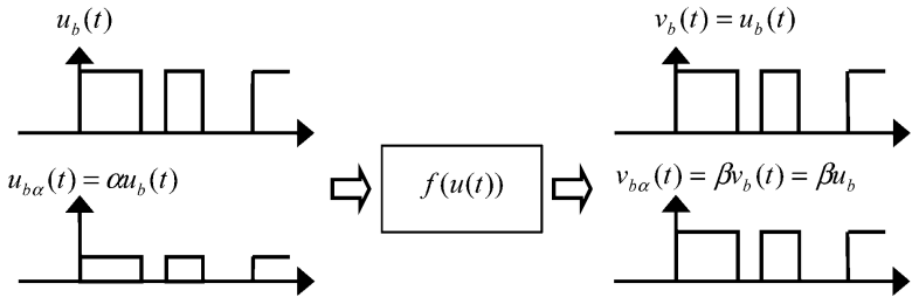
2. La segunda señal de prueba, $u_{b\alpha}$, es una señal binaria que es un múltiplo α de u_b (donde $1 > \alpha > 0$).
3. La tercera señal de prueba, u_{ms}, es una señal multipaso.

En la Figura 37(a) se muestra un ejemplo de estas señales. La entrada $u_b(t)$ es una señal de secuencia binaria pseudoaleatoria (PRBS, Pseudo-Random Binary Sequence) de valores 0 y 1 desde $t = 0$ hasta $t = 50$, y $u_{b\alpha}(t)$ es una señal PRBS que es un múltiplo 0.4 de $u_b(t)$. La entrada $u_{ms}(t)$ está compuesta por $u_b(t)$ desde $t = 0$ hasta $t = 50$, seguida de una señal aleatoria uniformemente distribuida entre 0 y 2 desde $t = 50$ hasta $t = 100$.

Identificación de la Función Estática no Lineal de Salida. Para la identificación de la función no lineal de salida, se utilizan únicamente las dos primeras señales de prueba. Estas señales son la señal binaria $u_b(t)$ y su múltiplo escalado $u_{b\alpha}(t)$, junto con las salidas de proceso correspondientes, como se muestra en la Figura 37(b). La función no lineal de entrada es una función estática no lineal, $v_b(t)$ y $v_{b\alpha}(t)$ también son señales binarias, tal como se ilustra en la Figura 36.

Figura 36

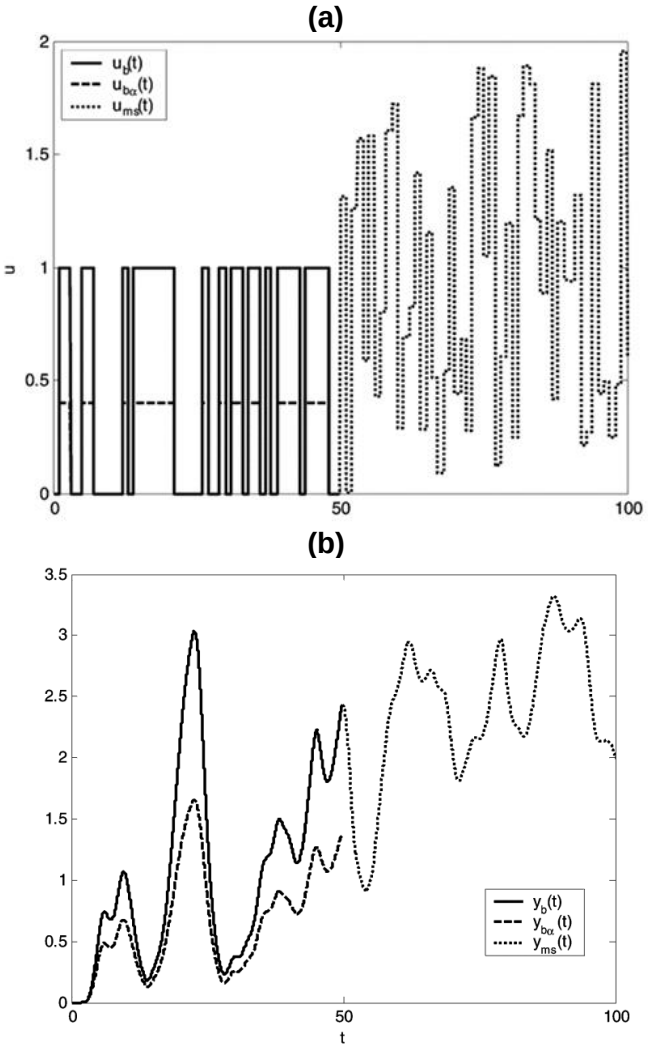
Suposición de la función estática no lineal de entrada.



Nota: De *Modeling of Hammerstein − Wiener processes with special input test signals*, por H.C. Park et al., 2006.

Figura 37

Activación del proceso mediante las señales de prueba propuestas: (a) señales de prueba de entrada; (b) salidas del proceso activadas.



Nota: De *Modeling of Hammerstein − Wiener processes with special input test signals*, por H.C. Park et al., 2006.

Se tiene $v_{b\alpha}(t) = \beta v_b(t) = \beta u_b(t)$, donde β es desconocido. Considerando que las variables intermedias $v(t)$ y $z(t)$ representan la entrada y la salida del subsistema dinámico lineal, según la regla de superposición, se tiene que:

$$z_{b\alpha}(t) = \beta z_b(t) \tag{39}$$

cuando ambas entradas de prueba se aplican en el estado estacionario (es decir, cuando $x(t) = 0$). Utilizando la relación de la Ecuación (39) y asumiendo un valor de β distinto de cero, es posible estimar los parámetros del modelo de la función estática no lineal de salida minimizando la siguiente función de coste.

$$V\left(\hat{\beta}, \hat{q}_1, \hat{q}_2, \dots, \hat{q}_r\right) = \frac{0.5}{N_b} \sum_{i=1}^{N_b} \left(\hat{z}_{b\alpha}(t_i) - \hat{\beta}\hat{z}_b(t_i)\right)^2 \tag{40}$$

Donde:

$$\hat{z}_{b\alpha}(t) = \hat{q}_1 y_{b\alpha}(t) + \hat{q}_2 y_{b\alpha}^2(t) + \cdots + \hat{q}_r y_{b\alpha}^r(t) \tag{41}$$

$$\hat{z}_b(t) = \hat{q}_1 y_b(t) + \hat{q}_2 y_b^2(t) + \cdots + \hat{q}_r y_b^r(t) \tag{42}$$

Aquí, $\hat{z}_{b\alpha}(t)$ y $\hat{z}_b(t)$ representan las salidas previstas del subsistema dinámico lineal. Los términos $y_{b\alpha}(t)$ y $y_b(t)$ corresponden a las salidas medidas del proceso. N_b es el número de puntos de datos correspondientes a las señales binarias. Es importante destacar que no se requiere información específica sobre el subsistema dinámico lineal ni sobre la función estática no lineal de entrada para realizar esta estimación (Park et al., 2006).

Identificación del Subsistema Dinámico Lineal. Una vez identificada la función estática no lineal de salida ($\hat{q}_i, i = 1, \dots, r$), se calcula la salida del subsistema dinámico lineal mediante $\hat{z}_b(t_i) = \hat{q}_1 y_b(t_i) + \hat{q}_2 y_b^2(t_i) + \cdots + \hat{q}_r y_b^r(t_i)$. Se asume que la entrada binaria del subsistema dinámico lineal es $v_b(t_i) = u_b(t_i)$ sin pérdida de generalidad, ya que tenemos un grado de libertad para ajustar b_j, como se ilustra en la Figura 35(b). De este modo, se identifica directamente el subsistema dinámico lineal a partir de $u_b(t_i)$ y la salida estimada $\left(\hat{z}_b(t_i)\right)$ del subsistema dinámico lineal aplicando métodos existentes de identificación de sistemas lineales (Park et al., 2006).

Identificación de la Función Estática no Lineal de Entrada. Una vez identificadas la función estática no lineal de salida y el subsistema dinámico lineal, se determina la función estática no lineal de entrada minimizando la siguiente función de coste utilizando los conjuntos de datos de $u_{ms}(t)$ de la Figura 37(a) y la correspondiente salida estimada del subsistema dinámico lineal.

$$V(\hat{p}_1, \hat{p}_2, \dots, \hat{p}_m) = \frac{0.5}{N_{ms}} \sum_{i=1}^{N_{ms}} (z^*_{ms}(t_i) - \hat{z}_{ms}(t_i))^2 \tag{43}$$

Donde:

$$z^*_{ms}(t) = \hat{q}_1 y_{ms}(t) + \hat{q}_2 y^2_{ms}(t) + \cdots + \hat{q}_r y^r_{ms}(t) \tag{44}$$

$$\hat{v}_{ms}(t) = \hat{p}_1 u_{ms}(t) + \hat{p}_2 u^2_{ms}(t) + \cdots + \hat{p}_m u^m_{ms}(t) \tag{45}$$

$$\hat{z}_{ms}(t) = \sum_{i=1}^{n_a} \hat{a}_i \hat{z}_{ms}(t-i) + \sum_{j=1}^{n_b} \hat{b}_j \hat{v}_{ms}\left(t - j - \hat{T}_d\right) \tag{46}$$

Aquí, $z^*_{ms}(t)$, $\hat{z}_{ms}(t)$ y $y_{ms}(t)$ denotan la salida estimada del subsistema dinámico lineal, la salida estimada del modelo y la salida medida del proceso correspondiente a $u_{ms}(t)$, respectivamente. N_{ms} es el número de puntos de datos correspondientes a la señal multipaso. $\hat{a}$, $\hat{b}$ y $\hat{T}_d$ representan las estimaciones del subsistema dinámico lineal obtenidas en la sección anterior (Park et al., 2006).

2.2.3 Red Neuronal Artificial de Enlace Funcional

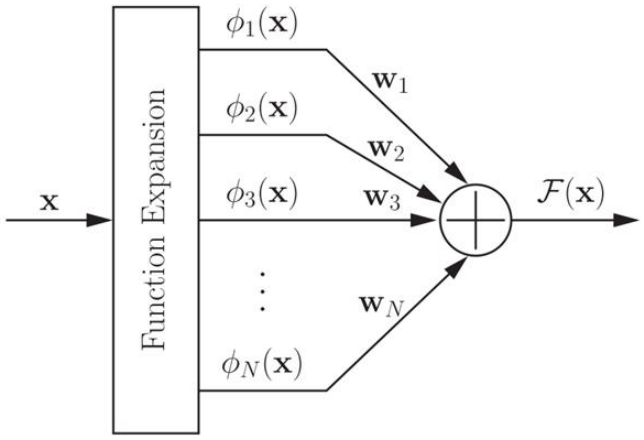
En la literatura se han propuesto numerosos métodos para identificar el modelo Hammerstein. Una dificultad inherente a esta identificación radica en la falta de disponibilidad de la señal intermedia entre la no linealidad estática y el subsistema lineal. Para mitigar esta dificultad, se han desarrollado métodos tanto iterativos como no iterativos, generalmente utilizando una no linealidad estática polinomial.

Aunque los algoritmos de identificación iterativa son sencillos y eficaces, presentan el problema de que, teóricamente, no se garantiza que los parámetros estimados converjan a sus valores verdaderos. Por otro lado, los métodos no iterativos evitan la dificultad de convergencia inherente a los métodos iterativos. Sin embargo, estos métodos no iterativos a menudo resultan en una sobreparametrización del modelo Hammerstein (Cui et al., 2014).

Las RNAs ofrecen una herramienta poderosa para modelar sistemas no lineales debido a su excelente capacidad de aproximación. Entre las diversas RNAs, la red neuronal artificial de enlace funcional (FLANN, Functional Link Artificial Neural Network) destaca por su estructura de una sola capa (ver Figura 38), que proporciona una tasa de convergencia más rápida y una carga computacional menor en comparación con la estructura de un perceptrón multicapa (MLP, Multilayer Perceptron). FLANN emplea funciones para la expansión funcional en la identificación de sistemas dinámicos no lineales. Con una adecuada selección de la expansión funcional, FLANN se desempeña tan bien como, o incluso mejor que, un MLP en la tarea de identificación del sistema (Patra y Kot, 2002).

Figura 38

Estructura de una FLANN.



Nota: De *Identification of Hammerstein model using functional link artificial neural network*, por M. Cui et al., 2014.

FLANN es una RNA de una sola capa, sin capa oculta, que captura las relaciones no lineales entre la entrada y la salida mediante la expansión de funciones. El enlace de funciones actúa sobre un elemento de un vector de entrada o sobre todos los vectores de entrada, generando un conjunto de funciones linealmente independientes. De este modo, se incrementa la dimensionalidad del vector de entrada, mejorando significativamente su representación. Consideremos un conjunto de funciones de base $B = \{\phi_i \in \mathcal{L}\}_{i \in I}$ con las siguientes propiedades:

- ✓ $\phi_1 = 1$
- ✓ El subconjunto $B_j = \{\phi_i \in \mathcal{L}\}_{i=1}^{J}$ es linealmente independiente, es decir, si $\sum_{i=1}^{N} w_i \phi_i = 0$, entonces $w_i = 0$ para todo $i = 1, 2, \dots, J$
- ✓ $\sup\left[\sum_{i=1}^{N} \|\phi_i\|_A^2\right]^{\frac{1}{2}} < \infty$

Sea $B_N = \{\phi_i \in B\}_{i=1}^{N}$ un conjunto de funciones base a considerar para el FLANN, como se observa en la Figura 38. El vector de entrada $x = [x_1, x_2, \dots, x_n]^T$ es expande primero a un vector como

$$x_{FE} = [\phi_1(x), \phi_2(x), \dots, \phi_N(x)]^T \tag{47}$$

con longitud m, donde $m = n \times N$. La salida de FLANN es la suma ponderada de cada elemento del vector expandido x_{FE}, que viene dada por

$$F(x) = \sum_{l=1}^{N} w_l \phi_l(x) \tag{48}$$

o en forma de matriz

$$F(x) = w^T x_{FE} \tag{49}$$

Donde $w = [w_1, w_2, \dots, w_N]^T$ es el vector de pesos.

Lee y Jeng (1998) presentan una RNA de modelo unificado basada en polinomios de Chebyshev para la aproximación de funciones estáticas. Este modelo se basa en una FLANN con expansión polinómica de Chebyshev y utiliza el algoritmo de aprendizaje recursivo de mínimos cuadrados. Se destaca que esta red tiene una capacidad de aproximación universal y una convergencia más rápida que una red MLP. En este trabajo, se seleccionan los polinomios de Chebyshev como función base para la expansión de funciones y el algoritmo de aprendizaje de retropropagación (BP, Back-Propagation). Los polinomios de Chebyshev se definen recursivamente de la siguiente manera:

$$T_{m+1}(x) = 2xT_m(x) - T_{m-1}(x) \tag{50}$$

con $T_0(x) = 1$ y $T_1(x) = x$

Para que los polinomios de Chebyshev estén bien definidos y tengan las propiedades que les caracterizan (particularmente ortogonalidad y la propiedad de minimizar el error en la interpolación de funciones), x debe estar en el intervalo $[-1,1]$.

2.2.4 Control Adaptativo por Modelo de Referencia

El método MRAC fue propuesto por Whitaker en 1958 y se muestra en su forma más común en la Figura 39. En este método, se utiliza un controlador primario para obtener el control en lazo cerrado, como en cualquier otro método de control convencional. Sin embargo, debido a la posible falta de conocimiento y a la variabilidad de los parámetros del proceso, es difícil encontrar un controlador fijo que responda eficazmente en todas las situaciones.

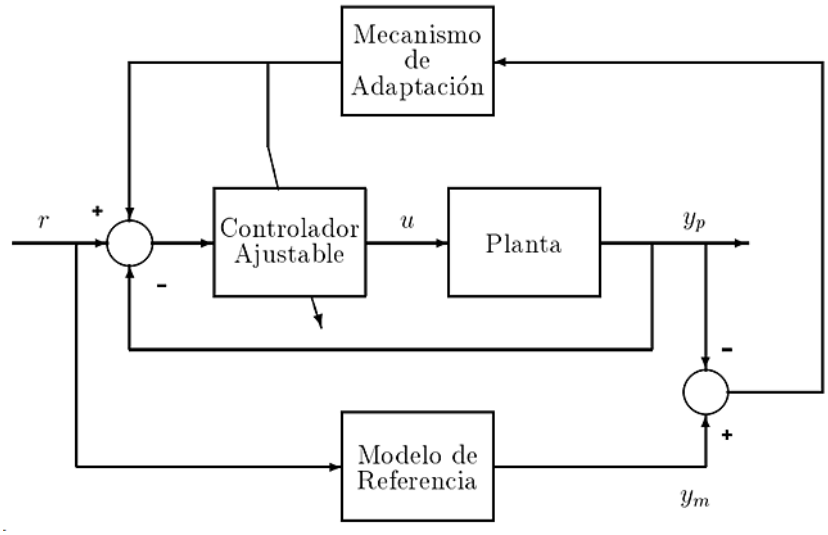
En el método MRAC, la respuesta deseada a la entrada del proceso se establece como modelo de referencia. El mecanismo de adaptación analiza la señal de salida (y_p) del proceso y la señal de salida del modelo de referencia (y_m) y calcula los parámetros adecuados para minimizar la diferencia entre ambas. Si es posible, el mecanismo de

adaptación se utiliza no solo las señales de salida del proceso y el modelo de referencia, sino también las señales de entrada, las señales de referencia y las variables de estado del proceso.

En el diagrama de la Figura 39, el modelo de referencia se dispone en paralelo al proceso. Aunque se trata de un esquema habitual, existen diversas variantes, como la disposición en serie o el uso de una combinación serie-paralelo (Rodriguez y López, 1996).

Figura 39

Diagrama de bloques básico del MRAC.



Nota: De *Control Adaptativo y Robusto*, por F. Rodríguez y M. J. López, 1996.

Diseño de Controladores Adaptativos. Un sistema de control adaptativo consta esencialmente de tres elementos: un controlador primario, un modelo de referencia y una ley adaptativa. Por lo tanto, a la hora de diseñar un sistema de control adaptativo, es necesario definir estos tres elementos.

- Controlador primario: El controlador primario asume inicialmente cualquiera de las configuraciones conocidas para el diseño de controladores lineales. Sin embargo, debe cumplir la condición de que el proceso y la disposición del controlador emulen el modelo de referencia. Esta condición restringe la secuencia y la estructura del

controlador. Además, la señal de control debe ser una función lineal de los parámetros para permitir la adaptación directa.

- Modelo de referencia: Un modelo de referencia que describe el comportamiento deseado del lazo cerrado de control suele representarse en forma paramétrica. La condición antes mencionada para el seguimiento del modelo de referencia también impone ciertas restricciones sobre el posible modelo de referencia en términos del orden relativo del proceso (polos redundantes). Por otra parte, el modelo elegido debe ser sensible a la dinámica del proceso. Por ejemplo, si se elige un modelo con una dinámica muy rápida, la señal de control será mucho mayor, lo que provocará una saturación y dificultará la respuesta del sistema a esta dinámica. Por este motivo, la elección del modelo de referencia no es sencilla y suele optarse por un modelo conservador (Rodriguez y López, 1996).

Regla de MIT. La regla del MIT es un enfoque original del control adaptativo basado en modelos. Su nombre se debe a que se desarrolló en el Laboratorio de Instrumentación del MIT (Massachusetts Institute of Technology).

Astrom y Wittenmark (1994) consideran un sistema de bucle cerrado en el que el controlador tiene un parámetro ajustable θ. La respuesta deseada en lazo cerrado viene dada por un modelo cuya salida es y_m. Sea e el error entre la salida y de lazo cerrado y la salida del modelo y_m. Una posibilidad es ajustar los parámetros de modo que la función de pérdida

$$J(\theta) = \frac{1}{2}e^2 \tag{51}$$

se minimiza. Para reducir J, es lógico ajustar los parámetros en la dirección del gradiente negativo de J, es decir,

$$\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta} = -\gamma e \frac{\partial e}{\partial \theta} \tag{52}$$

Esto se refiere a la conocida regla MIT. La derivada parcial $\partial e/\partial\theta$, la llamada derivada de sensibilidad del sistema, muestra cómo el parámetro ajustable afecta al error. Si se supone que el parámetro cambia más lentamente que otras variables del sistema, la derivada $\partial e/\partial\theta$ se estima suponiendo que θ es constante (Astrom y Wittenmark, 1994).

Método de Lyapunov. Dada la naturaleza no lineal y variable en el tiempo de los sistemas MRAC, los criterios de estabilidad de los sistemas lineales no son aplicables. Un método muy conocido es el método directo de Lyapunov (Rodriguez y López, 1996). Según este método, un sistema tiene un equilibrio asintóticamente estable en $x = 0$ si existe una función, la llamada función de Lyapunov, $V(x)$, que satisface las siguientes condiciones:

$$\begin{gathered} V(x) > 0 \; para \; x \neq 0 \; definida \; positiva \\ \dot{V}(x) < 0 \; para \; x \neq 0 \; definida \; negativa \\ V(x) \to \infty \; para \; \|x\| \to \infty \\ V(0) = 0 \end{gathered} \tag{53}$$

La función de Lyapunov se asemeja a una función de energía, esta debe disminuir con el tiempo. Al emplear este método en el diseño de sistemas adaptativos, las especificaciones de estabilidad se trasladan directamente a la ley de adaptación, siguiendo los siguientes pasos (Rodriguez y López, 1996):

1. Encontrar la ecuación de error, sea en la salida $(y_p - y_m)$ o en las variables de estado $(x_p - x_m)$.
2. Encontrar una función de Lyapunov que sea una función del error entre las señales y del error en los parámetros $(\phi = \hat{\theta} - \theta)$. En su forma más simple, esta función adopta la forma:

$$V = e^T P e + \phi^T \Gamma^{-1} \phi \tag{54}$$

Donde las matrices P y Γ^{-1} deben ser definidas positivas.

3. Calcular la derivada de la función de Lyapunov. La derivada debe ser definida positiva. Generalmente toma la forma:

$$\dot{V} = -e^T Q e + f(\phi) \tag{55}$$

El primer término asegura que la derivada es definida negativa, por lo que, al igualar el resto a cero, se obtiene una posible solución. La matriz Q es definida positiva. Las matrices P y Q, para un sistema gobernado por una matriz A, están vinculadas por la ecuación de Lyapunov:

$$-Q = A^T P + PA \tag{56}$$

4. Al igualar el término adicional a cero, se obtiene la ley de adaptación. Por lo general, tiene la forma:

$$\dot{\theta} = -\Gamma \varepsilon \xi \tag{57}$$

Donde ε está directamente relacionado con el error e y ξ es una versión modificada del vector de señales (referencia, salida, etc.).

2.2.5 Predictor de Smith

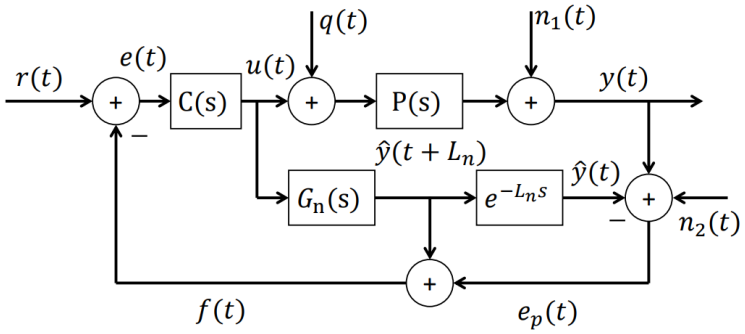
El retardo de transporte en un sistema de control en lazo cerrado presenta un desafío significativo para alcanzar un comportamiento rápido y robusto, ya que reduce el margen de fase del sistema, lo que degrada el rendimiento del controlador. Para abordar este problema, Smith propuso un compensador que elimina eficazmente el efecto del retardo de la ecuación característica del sistema de control para un proceso estable en el caso nominal, es decir, sin errores de modelización. Este compensador, conocido como Predictor de Smith, ha sido la base de numerosos estudios actuales que tratan el problema del retardo del transporte.

En 1957, Otto Smith introdujo la estructura de control que se muestra en la Figura 40. En esta figura, $P_{n(s)} = G_n(s)e^{-L_{ns}}$ representa el modelo nominal del proceso, $G_n(s)$ es el modelo sin retardo de transporte, conocido como el modelo rápido, L_n es el retardo de transporte del modelo nominal, $C(s)$ es el controlador primario, $P(s)$ es la planta real, $u(t)$ es la señal de control, $y(t)$ es la variable de proceso, $q(t)$ es una perturbación de carga, $n_1(t)$ es una perturbación en la salida, $n_2(t)$ es ruido de medida, $r(t)$ es la señal de referencia, $e_p(t)$ es el error de predicción y $f(t)$ es la salida predicha.

Esta estructura predice la salida del proceso real, $y(t)$, basándose en el modelo sin retardo, $G_n(s)$, anticipando el comportamiento del sistema como si no existiera el retardo de transporte, $-L_{ns}$. De este modo, el control prevé el comportamiento probable que el sistema, $P(s)$, presentará en un tiempo igual al retardo de transporte. A partir de esta estructura, se diseña el controlador primario, $C(s)$, considerando el proceso sin retardo (Castaño Giraldo, 2016).

Figura 40

Diagrama de bloques general del Predictor de Smith.



Nota: De *Estudo de técnicas de sintonia do preditor de Smith filtrado para sistemas multivariáveis com atraso*, por S.A. Castaño, 2016.

2.2.6 Protocolo Modbus

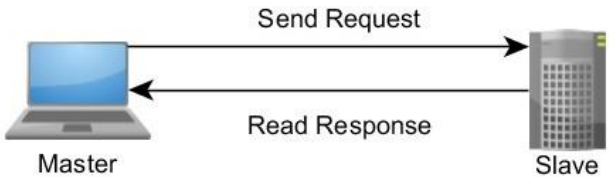
Modbus es un protocolo industrial desarrollado en 1979 para facilitar la comunicación entre dispositivos de automatización. Inicialmente diseñado para transferir

datos a través de una capa serie, Modbus se ha expandido para incluir implementaciones sobre serial, TCP/IP (Transmission Control Protocol/Internet Protocol) y el Protocolo de Datagramas de Usuario (UDP, User Datagram Protocol).

El protocolo funciona en un modelo de solicitud-respuesta bajo una relación maestro-esclavo (ver Figura 41). El maestro, típicamente un sistema de Interfaz de Hombre-Máquina (HMI, Human-Machine Interface) o un Control de Supervisión y Adquisición de Datos (SCADA, Supervisory Control and Data Acquisition), es responsable de iniciar cada interacción y los esclavos, como sensores o Controladores Lógicos Programables (PLC, Programmable Logic Controllers), responden. La interacción siempre ocurre en pares: el maestro inicia una solicitud y espera una respuesta (*What is the Modbus Protocol & How Does It Work?*, 2014).

Figura 41

Esquema de comunicación Maestro-Esclavo.



Nota: De *What is the Modbus Protocol & How Does It Work?*, por National Instruments, 2014 (https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html).

En su implementación inicial, Modbus era un protocolo único basado en comunicación serial, lo que impedía su división en múltiples capas. Con el tiempo, se introdujeron diferentes unidades de datos de aplicación para modificar el formato del paquete utilizado en serial y para permitir el uso de redes TCP/IP y UDP. Esta evolución llevó a una separación del protocolo central, que define la Unidad de Datos de Protocolo (PDU, Protocol Data Unit), y la capa de red, que define la Unidad de Datos de Aplicación (ADU, Application Data Unit).

PDU. La PDU y su código forman el núcleo de la Especificación del Protocolo de Aplicación Modbus (Modbus Organization, 2012). Esta especificación define el formato de la PDU, los conceptos de datos utilizados, el uso de códigos de función para acceder a esos datos y las implementaciones específicas y restricciones de cada código de función. La PDU de Modbus consta de un código de función seguido de un conjunto de datos cuyo tamaño y contenido dependen del código de función. La PDU completa no supera los 253 bytes. Cada código de función tiene un comportamiento específico que los esclavos implementan de manera flexible según sus necesidades de aplicación.

En Modbus, los datos se organizan en cuatro bancos o rangos de direcciones: bobinas, entradas discretas, registros de retención y registros de entrada. Estos bancos especifican el tipo y los derechos de acceso a los datos. Los dispositivos esclavos acceden directamente a estos datos, que están alojados localmente, mientras que los maestros deben solicitar acceso a través de varios códigos de función. Los datos accesibles en Modbus generalmente constituyen un subconjunto de la memoria principal del dispositivo, (*What is the Modbus Protocol & How Does It Work?*, 2014). El comportamiento de cada bloque se describe en la Tabla 11.

Tabla 11

Bloques del Modelo de Datos Modbus

Bloque de Memoria	Tipos de Datos	Acceso de Maestro	Acceso de Esclavo
Bobinas	Booleano	Lectura/Escritura	Lectura/Escritura
Entradas discretas	Booleano	Solo Lectura	Lectura/Escritura
Registros de retención	Palabra sin signo	Lectura/Escritura	Lectura/Escritura
Registros de entrada	Palabra sin signo	Solo Lectura	Lectura/Escritura

Nota: De *What is the Modbus Protocol & How Does It Work?*, por National Instruments, 2014 (https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html).

Modelo de Direccionamiento de Datos. La especificación Modbus define que cada bloque de datos contiene un espacio de direcciones de hasta 65536 elementos, numerados de 0 a 65535 en la PDU, pero los elementos están numerados de 1 a 65536. Por ejemplo, la bobina 1 está en la dirección 0, y el registro de retención 54 está en la dirección 53. Los dispositivos no están obligados a implementar todos los rangos; por lo que se elige implementar solo ciertos registros. Los intentos de acceso a registros no válidos se manejan mediante excepciones.

Para facilitar la comprensión y documentación, se utiliza un esquema de numeración con prefijos. Por ejemplo, un registro de retención en la dirección 13 se refiere como 4014, 40014 o 400014, donde "4" indica registros de retención. La notación varía según el espacio de direcciones utilizado por el dispositivo: 4XXXXX para 65536 registros y 4XXX para un rango menor (*What is the Modbus Protocol & How Does It Work?*, 2014).

En este esquema de direccionamiento, a cada tipo de dato se le asigna un prefijo, como se muestra en la Tabla 12.

Tabla 12

Prefijos de los Bloques de datos.

Bloque de datos	Prefijo
Bobinas	0
Entradas discretas	1
Registros de entrada	3
Registro de retención	4

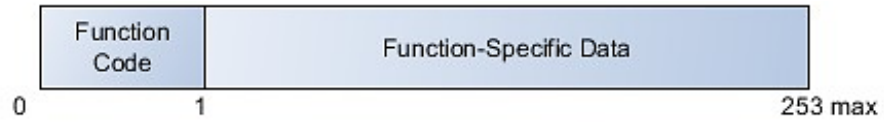
Nota: De *What is the Modbus Protocol & How Does It Work?*, por National Instruments, 2014 (https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html).

Códigos de Función Modbus. Los códigos de función en Modbus están definidos por la norma y siguen un patrón establecido. El esclavo valida la solicitud verificando el código de función, la dirección y el rango de datos. Si todo es correcto, ejecuta la acción

solicitada y envía una respuesta. La PDU contiene el código de función seguido de hasta 252 bytes de datos específicos, como se muestra en la Figura 42.

Figura 42

Estructura de la PDU Modbus.



Nota: De *What is the Modbus Protocol & How Does It Work?*, por National Instruments, 2014 (https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html).

Si el código de función es aceptado, el esclavo descompone los datos según la definición de la función. Debido al límite de tamaño del paquete de 253 bytes, los dispositivos tienen restricciones en la cantidad de datos que transfieren, que varía según el código de función utilizado.

Los códigos de función más comunes llevan el nombre del rango de datos conceptual que modifican o al que tienen acceso (*What is the Modbus Protocol & How Does It Work?*, 2014). Por ejemplo, "Lectura de registro de retención" realiza la acción de extraer datos de la memoria definida como registros de retención y devolverlos al maestro. La Tabla 13 identifica los códigos de función más comunes.

ADU. Modbus utiliza varios protocolos de red, como serie, TCP/IP y UDP. Para adaptarse a cada protocolo, Modbus incluye variantes de la ADU. Estas variantes garantizan la comunicación fiable al incluir la dirección, la PDU completa con el código de función y los datos asociados, así como información de comprobación de errores. Los tres formatos estándar de ADU son TCP, RTU (Remote Terminal Unit) y ASCII (American

Standard Code for Information Interchange), cada uno utilizado en diferentes entornos de red.

Tabla 13

Códigos de Función Modbus más comunes.

			Modbus functions	Code	(hex)
Acceso a los datos	Acceso a bits	Entradas físicas discretas	Lectura de entradas discretas	02	02
		Bobinas físicas	Lectura de bobinas	01	01
			Escritura de una bobina	05	05
			Escritura de varias bobinas	15	0F
	Acceso de 16 bits	Registros físicos de entrada	Lectura de registros de entrada	04	04
		Registros físicos de salida	Lectura de registros de retención	03	03
			Escritura de un registro	06	06
			Escritura de varios registros	16	10
			Lectura/Escritura de Registros Múltiples	23	17
			Enmascarar registro de escritura	22	16
	Acceso al expediente		Leer cola FIFO	24	18
			Leer registro de archivo	20	14
			Leer registro de archivo	21	15
	Diagnóstico		Leer Estado de excepción	07	07
			Diagnóstico	08	08
			Obtener contador de eventos Com	11	0B
			Obtener registro de eventos Com	12	0C
			Informar ID de servidor	17	11
			Leer identificación del dispositivo	43	2B

Nota: De *What is the Modbus Protocol & How Does It Work?*, por National Instruments, 2014 (https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html).

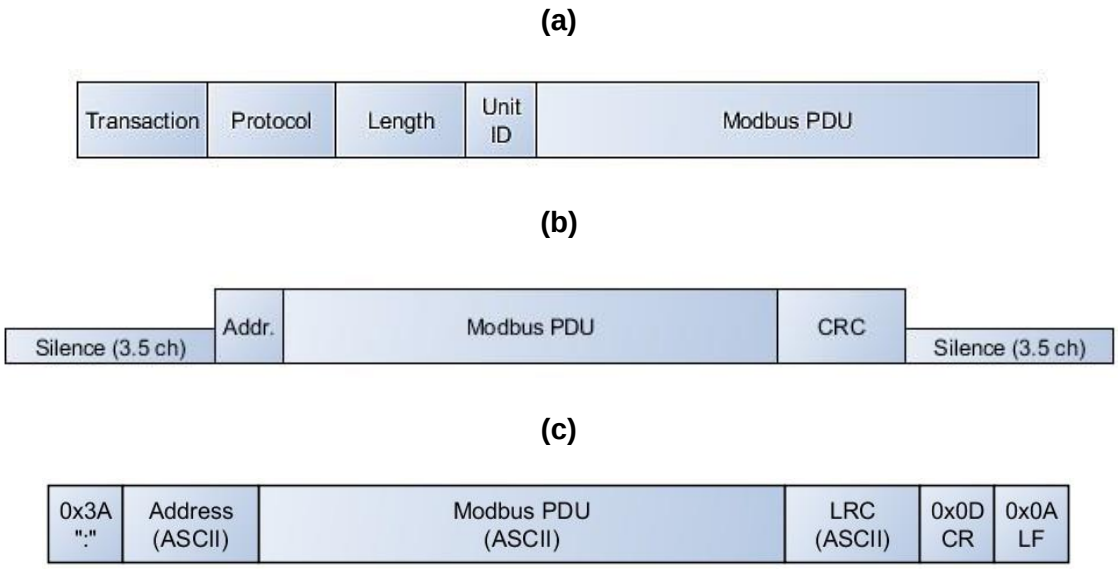
TCP/IP. En el caso de las ADUs TCP en el protocolo Modbus, estas constan de la Cabecera del Protocolo de Aplicación Modbus (MBAP, Modbus Application Protocol Header) seguida de la PDU Modbus. La MBAP es una cabecera que incluye un identificador de transacción, un identificador de protocolo, una longitud y un Identificador (ID, identifier) de unidad, como se muestra en la Figura 43(a).

El ID de transacción permite el seguimiento preciso de las solicitudes y respuestas en redes Ethernet. El ID de protocolo permite ampliar el comportamiento del protocolo, y la longitud del paquete delinea el resto del mensaje. El ID de unidad se utiliza para determinar la dirección del dispositivo esclavo. La PDU sigue estando limitada a 253 bytes para el protocolo estándar.

RTU. La ADU Modbus más simple, utilizada en redes serie, consta de la PDU principal junto con una dirección que define el destino del mensaje y una verificación de redundancia cíclica (CRC, Cyclic Redundancy Check) para verificar la integridad de los datos. La estructura de la ADU RTU se presenta en la Figura 43(b).

Figura 43

Estructura de la ADU: (a) TCP/IP; (b) RTU; (c) ASCII.



Nota: De *What is the Modbus Protocol & How Does It Work?*, por National Instruments, 2014 (https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html).

Sin embargo, en implementaciones modernas, surgen problemas debido a intervalos de silencio entre paquetes, introducidos por tecnologías de transferencia serie

más rápidas y la abstracción de la comunicación serie en controladores modernos. Esto resulta en la interpretación errónea de los mensajes y en una mala detección de los intervalos de silencio. Para resolver estos problemas, se recomienda desglosar la capa de abstracción entre la PDU Modbus y la capa de red, permitiendo un mejor manejo de los intervalos de silencio y una detección más precisa de los mensajes completos.

ASCII. La ADU ASCII es más compleja que la RTU, como se muestra en la Figura 43(c), pero también evita muchos de los problemas asociados con el paquete RTU. Sin embargo, presenta sus propias desventajas.

La ADU ASCII resuelve el problema de determinar el tamaño de los paquetes mediante un inicio y un final claramente definidos para cada paquete. Sin embargo, todos los datos se transfieren como caracteres hexadecimales codificados en ASCII, lo que significa que se envían el doble de datos y tanto las aplicaciones emisoras como receptoras deben ser capaces de analizar los valores ASCII. A pesar de esto, las APIs serie modernas, como NI-VISA y la clase SerialPort de .NET Framework, procesan fácilmente el flujo de datos en la línea serie de manera eficiente.

Modbus en LabVIEW. El API de Modbus de bajo nivel en LabVIEW es ideal para aplicaciones que requieren un alto control sobre las secuencias y la temporización de las solicitudes Modbus, aunque esto conlleva una mayor complejidad en el código de la aplicación. LabVIEW proporciona ejemplos para ayudar a entender esta complejidad. En la Figura 44 se presenta un ejemplo del uso de la librería Modbus de LabVIEW.

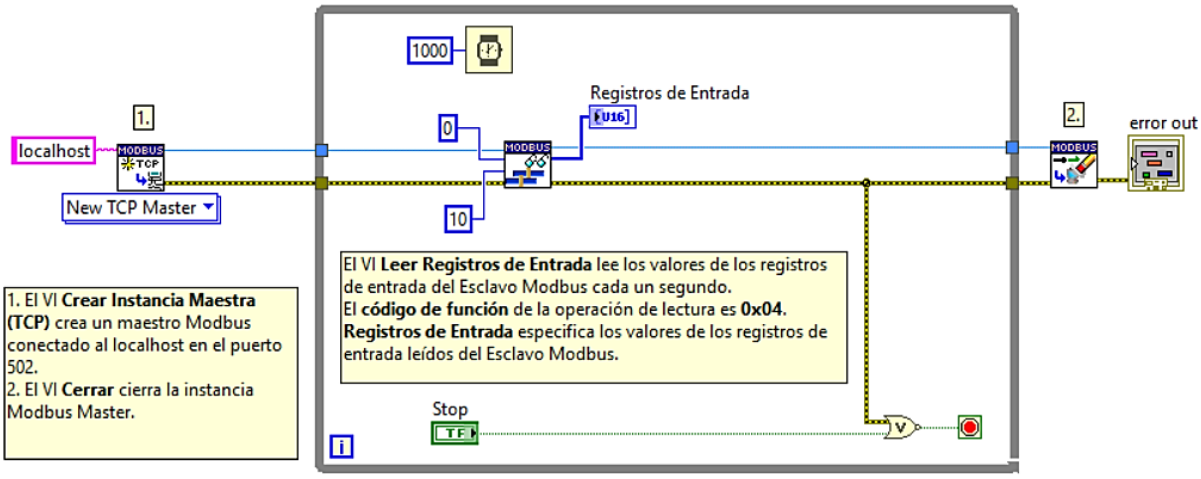
Todos los ejemplos siguen el patrón de abrir, leer/escribir, y cerrar.

- ✓ Instancia de Modbus: Se crea una instancia de Modbus. El API permite elegir entre Modbus Serial o TCP, y definir si actuará como Maestro o Esclavo (ver Figura 45).

- ✓ Consulta de Datos: Se utiliza el código de función `Read Input Registers`. El API soporta varios códigos de función, los cuales se ilustran en la Figura 46.
- ✓ Cierre de Instancia: Se cierra la instancia de Modbus, liberando la memoria y cerrando la conexión TCP o Serial.

Figura 44

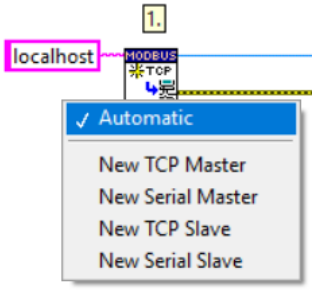
Ejemplo del uso de la librería Modbus de LabVIEW.



Nota: Adaptado de *Introducción a Modbus*, por National Instruments, 2014 (https://www.ni.com/es/shop/labview/introduction-to-modbus-using-labview.html).

Figura 45

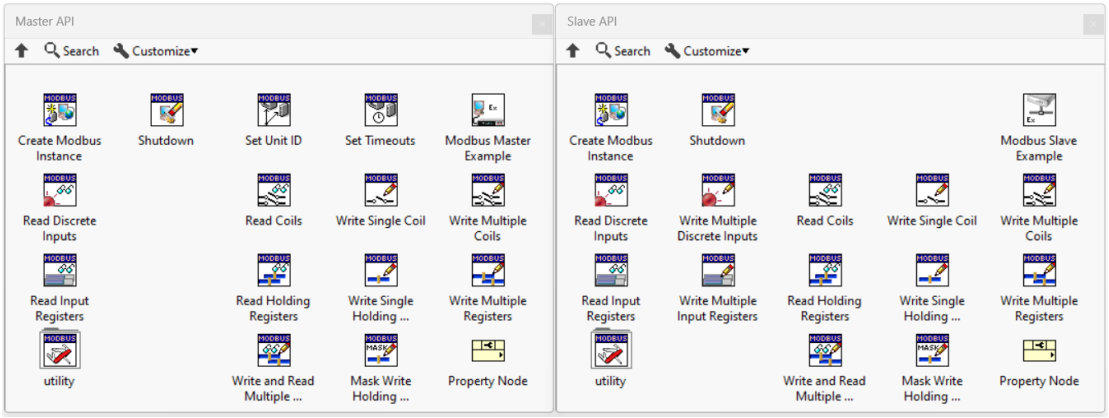
Selección del tipo de Maestro Modbus.



Nota: Adaptado de *Introducción a Modbus*, por National Instruments, 2014 (https://www.ni.com/es/shop/labview/introduction-to-modbus-using-labview.html).

Figura 46

Las Paletas de Maestro y Esclavo de Modbus muestran los Códigos de Función.



Nota: Adaptado de *Introducción a Modbus*, por National Instruments, 2014 (https://www.ni.com/es/shop/labview/introduction-to-modbus-using-labview.html).

Capítulo III. Desarrollo del trabajo de investigación

3.1 Descripción del Diseño de la Planta Piloto

Se construyó un tablero eléctrico para suministrar energía y gestionar el control de todos los dispositivos conectados, garantizando una operación ordenada, segura y eficiente. Este diseño integral asegura una adecuada lectura y control de las variables del sistema, permitiendo una gestión precisa y fiable del nivel de agua en los tanques.

Dentro del tablero eléctrico se encuentran diversos componentes esenciales para el funcionamiento del sistema: (Ver Figura 47)

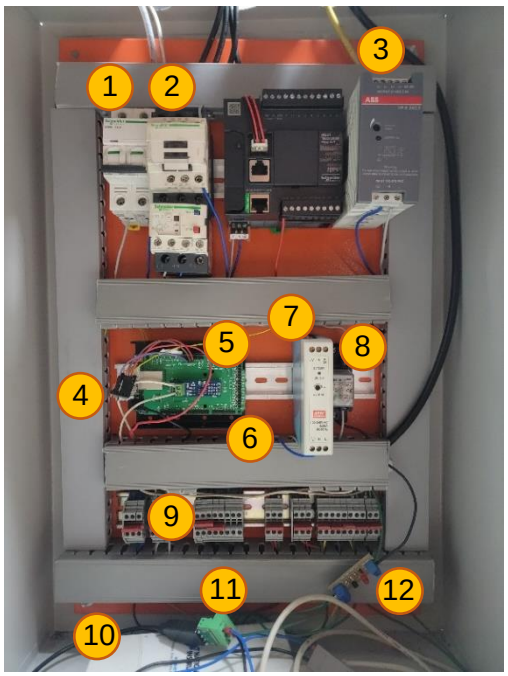
- ✓ Alimentación: Fuentes de alimentación de 5V y 24V, que proporcionan la energía necesaria para los diferentes dispositivos electrónicos y actuadores.
- ✓ Potencia y protección: Incluye un interruptor eléctrico manual para el encendido y apagado del sistema, un contactor de 220V AC, un relé térmico y un relé de 24V DC para la activación y desactivación de la bomba de agua.
- ✓ Procesamiento: Utiliza un Arduino Due, equipado con un shield de comunicación Modbus de elaboración propia, y una tarjeta NI DAQ USB 6211 para la adquisición de datos y la interacción con los sensores y actuadores.
- ✓ Conexionado: Compuesto por borneras, puentes y cables de distintos colores, que facilitan una conexión clara y ordenada entre todos los componentes del sistema.

El tablero eléctrico está diseñado para integrar de manera eficiente todos los elementos del sistema de control. En la

Figura 48 se presenta la estructura de funciones del sistema, que ilustra la organización y conexión de los diferentes componentes.

Figura 47

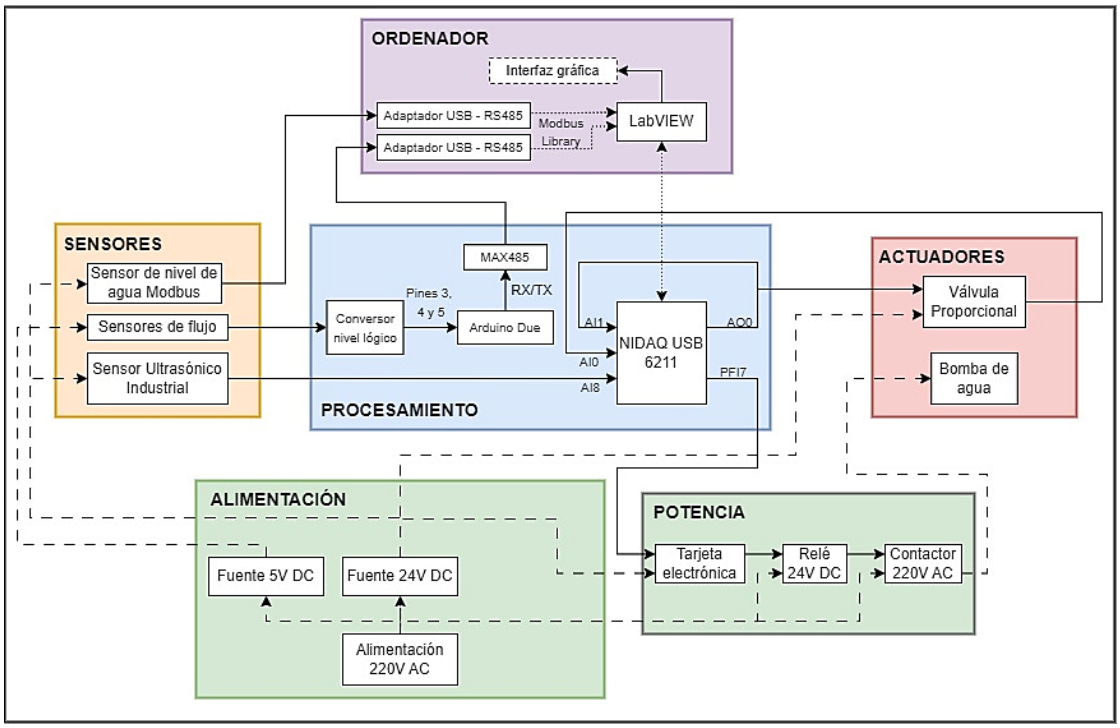
Tablero eléctrico de la planta piloto.



1. Interruptor Manual
2. Contactor y Relé Térmico
3. Fuente de Alimentación (+24Vdc)
4. Convertidor lógico 5-3.3V
5. Placa Shield Arduino DUE
6. Arduino DUE
7. Fuente de alimentación (+5Vdc)
8. Relé Electromagnético
9. Borneras
10. USB NIDAQ 6211
11. Convertidor USB a RS485
12. Placa electrónica interfaz

Figura 48

Estructura de Funciones del Sistema de tanques en cascada.



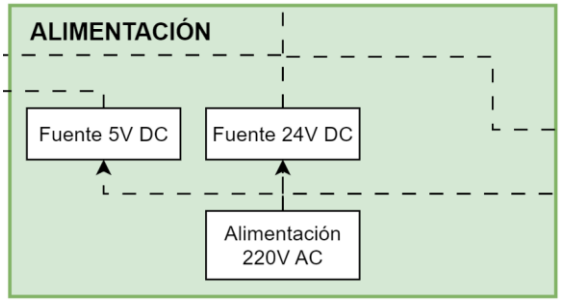
Este diagrama de bloques proporciona una visión general del sistema, destacando los bloques de Alimentación, Sensores, Procesamiento, Ordenador, Potencia y Actuadores. A continuación, se detallará cada bloque, explicando su función y conexión dentro del sistema.

3.1.1 Alimentación

- Alimentación 220V AC: El tablero eléctrico requiere de una alimentación de 220V AC obtenida de un tomacorriente mediante un cable de doble conductor.
- Fuente 5V DC: Esta fuente convierte el voltaje alterno a un voltaje continuo de 5V DC. Su función es alimentar tanto al Arduino Due como a los tres sensores de flujo.
- Fuente 24V DC: Esta fuente convierte el voltaje alterno a un voltaje continuo de 24V DC. Se emplea para alimentar el sensor ultrasónico industrial, la válvula proporcional y la tarjeta electrónica.

Figura 49

Bloque Alimentación.



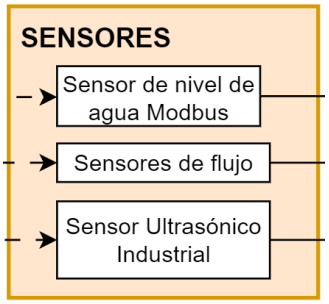
3.1.2 Sensores

- Sensor de Nivel de Agua Modbus: El sensor de nivel de agua con salida RS485 tiene la responsabilidad de medir el nivel de agua en el Tanque 1. Este sensor funciona como un esclavo en la red Modbus.

- Sensores de Flujo: Los sensores de flujo YF-S201 están destinados a medir el caudal de entrada a cada tanque. Se encuentran conectados a los pines 3, 4 y 5 del Arduino Due, configurados como pines de interrupción.
- Sensor Ultrasónico Industrial: El sensor Allen Bradley 873M-D18AV800-D4 se utiliza para medir el nivel de agua en el Tanque 2, el cual se desea controlar. Este sensor está conectado a la entrada analógica AI8 del NIDAQ USB 6211.

Figura 50

Bloque Sensores.



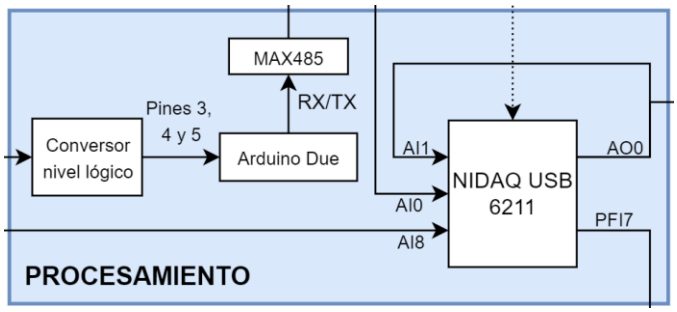
3.1.3 Procesamiento

- Arduino Due: Su función es procesar las señales provenientes de los sensores de flujo, ofreciendo lecturas precisas de caudal en las unidades específicas requeridas. El Arduino Due funciona como un esclavo en la red Modbus.
- Conversor Nivel Lógico: Su función es convertir los pulsos de 5V proveniente de los sensores de flujo a 3.3V para evitar dañar las entradas digitales del Arduino Due.
- MAX485: Permite al Arduino Due funcionar como un esclavo en la red Modbus. Este dispositivo facilita la comunicación bidireccional entre el Arduino y otros dispositivos en una red Modbus, permitiendo al Arduino recibir y responder a las solicitudes de lectura o escritura enviadas por el maestro de la red.
- NIDAQ USB 6211: La tarjeta de adquisición de datos se encarga de la lectura analógica del sensor ultrasónico industrial, ejecuta la escritura analógica para

controlar la válvula proporcional y activa la bomba de agua según las necesidades del sistema.

Figura 51

Bloque Procesamiento.

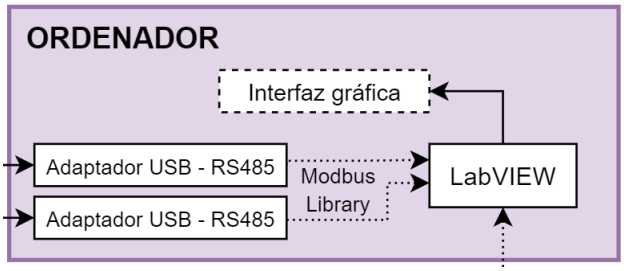


3.1.4 Ordenador

- Adaptador USB – RS485: Este dispositivo convierte la señal RS485 a USB, lo que permite conectar un ordenador a una red Modbus y operar como maestro en dicha red. Este adaptador facilita la comunicación entre el ordenador y otros dispositivos esclavos que utilizan la norma RS485 para la comunicación en la red Modbus.
- Arduino IDE: Este entorno sirve para desarrollar el programa que posteriormente se carga en el Arduino Due. En el código se programó el procesamiento de las señales provenientes de los sensores de flujo y el envío de los datos al maestro de la red Modbus mediante la librería “SimpleModbusSlave_DUE.h” (ver Anexo N°12).
- LabVIEW: En este entorno de programación gráfica se configura las entradas y salidas analógicas y digitales del NIDAQ USB 6211. El ordenador actúa como maestro de la red Modbus mediante el bloque “Modbus Library” de LabVIEW.
- Interfaz gráfica: Se crea en el Front Panel de LabVIEW y contiene las lecturas de todos los sensores, el porcentaje de apertura de la válvula y la activación de la bomba de agua. Además, presenta gráficos de las señales, incluyendo referencia, acción de control y variable controlada.

Figura 52

Bloque Ordenador.

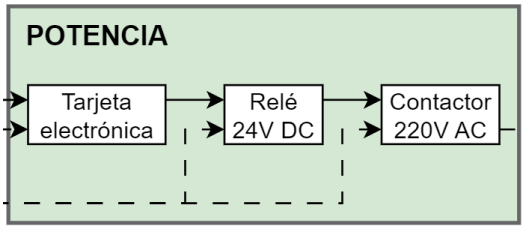


3.1.5 Potencia

- Tarjeta electrónica: Esta placa cumple la función de activar el relé de 24V DC utilizando la salida digital PFI7 del NIDAQ USB 6211. El esquema PCB se detalla en el Anexo N°14.
- Relé de 24V DC: Su tarea principal es activar el contactor Schneider de 220V AC.
- Contactor de 220V AC: Responsable de encender o apagar la bomba de agua monofásica.

Figura 53

Bloque Potencia.



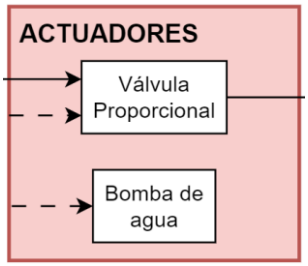
3.1.6 Actuadores

- Válvula Proporcional: Este actuador se utiliza para controlar el porcentaje de apertura, lo que regula el caudal de entrada al Tanque 2. La acción de control está dada por la salida analógica (AO0) del NIDAQ USB 6211.

- Bomba de Agua: Su función principal es transportar el agua desde el Tanque 3 hasta el Tanque 1.

Figura 54

Bloque Actuadores.



3.2 Calibración de Dispositivos

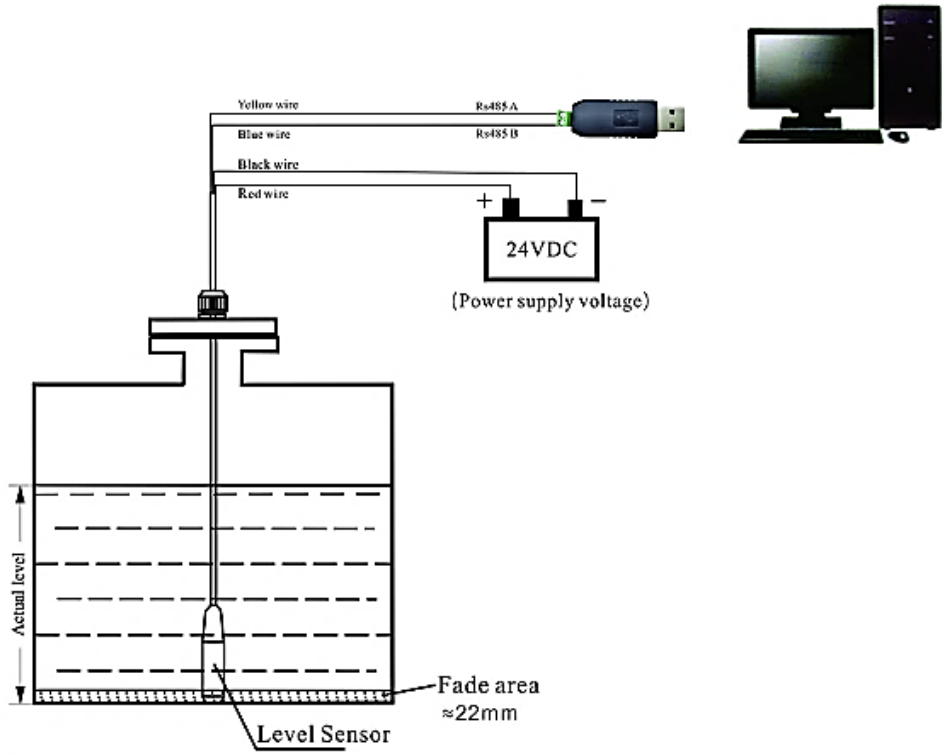
3.2.1 Sensor de Nivel de Agua con salida RS485

Se realizaron las conexiones de acuerdo con el esquema mostrado en la Figura 55 para establecer la comunicación entre el sensor de nivel de agua y un ordenador mediante el Protocolo Modbus RTU. En esta configuración de red, el sensor actúa como esclavo, mientras que el ordenador actúa como maestro. Se utilizó el programa Modbus Poll, un simulador de maestro Modbus, para facilitar la comunicación y obtención de registros.

Para habilitar la comunicación Modbus RTU, es necesario realizar algunas configuraciones en el programa Modbus Poll. En primer lugar, se debe seleccionar el puerto COM correspondiente, así como establecer la velocidad en baudios y otros parámetros de la comunicación serial, tal como se muestra en la Figura 56. Además, para que el dispositivo reconozca el adaptador, es crucial instalar el controlador CH340. Este controlador permite que el sistema operativo identifique y funcione correctamente con el adaptador serial, asegurando así una comunicación fluida entre el dispositivo y el programa Modbus Poll.

Figura 55

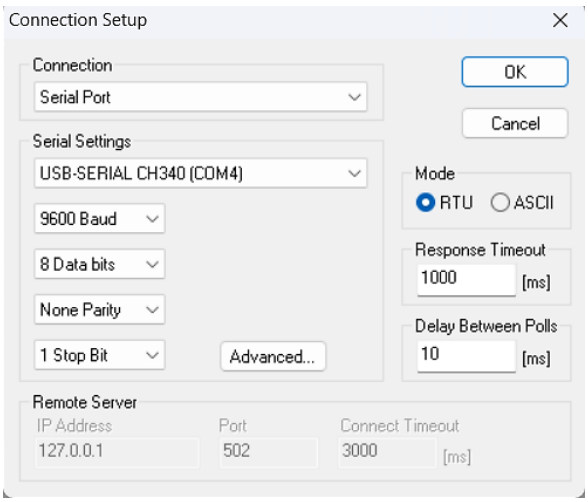
Circuito de prueba del sensor de nivel de agua con salida RS485.



Nota: De *Sensor de nivel de agua con transmisor de nivel de líquido RS485 Modbus RTU 24VDC con cable de 5 m*, por AliExpress, s.f., (https://es.aliexpress.com/item/1005004775126836.html).

Figura 56

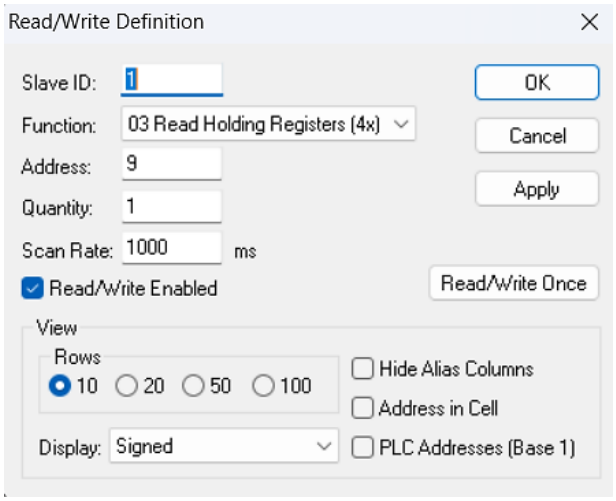
Configuración de la conexión en Modbus Poll.



Después de realizar las configuraciones para la conexión, se procede a definir los parámetros específicos para la comunicación Modbus RTU. En este caso, se debe establecer el ID del sensor de nivel de agua, que actúa como esclavo en la red Modbus. Según la hoja técnica, el ID de dicho sensor es 1. Luego, se elige la función Modbus adecuada, que en este caso será la lectura de registros de retención. Por último, se especifica la dirección del registro, que en este caso es 9 (ver Figura 57). Estos parámetros permitirán al programa Modbus Poll comunicarse correctamente con el sensor de nivel de agua y leer los datos necesarios.

Figura 57

Definición de Lectura/Escritura en Modbus Poll.

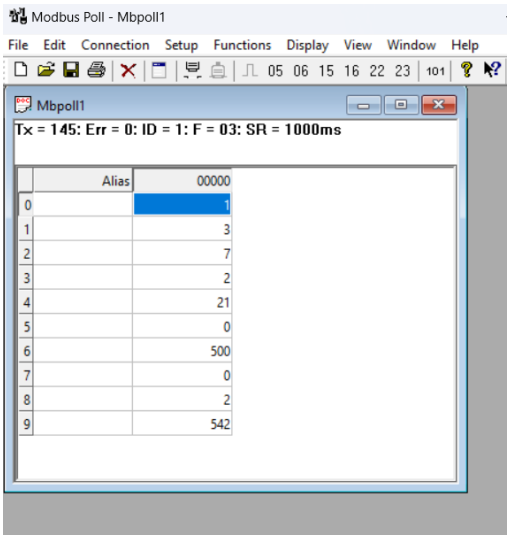


Se aplicaron varios niveles de agua conocidos y se midieron los valores correspondientes del registro 9 (ver Figura 58). Los datos obtenidos se guardaron en un archivo de texto (.txt), el cual fue importado por un script en Matlab. En dicho script, se emplea la función polyfit para obtener los coeficientes del polinomio de primer grado que mejor se ajusta a los puntos, utilizando el Método de Mínimos Cuadrados. A continuación, se presenta el código de Matlab:

```
clc, clearvars, close all;
% ============================================================
% CALIBRACIÓN DE SENSOR DE NIVEL MODBUS
% ============================================================
% Obtención de Coeficientes de Polinomio ajustado por Mínimos Cuadrados
data = load('DataCalibracionSensorNivelModbus.txt') % Carga de datos
registerValue = data(:,1); % Valor del Registro 9
distance = data(:,2); % Distancia en cm
coef = polyfit(registerValue,distance,1); % Polinomio ajustado
a1 = coef(1) % Pendiente
a0 = coef(2) % Intercepto
```

Figura 58

Lectura de los registros del sensor de nivel mediante Modbus Poll.

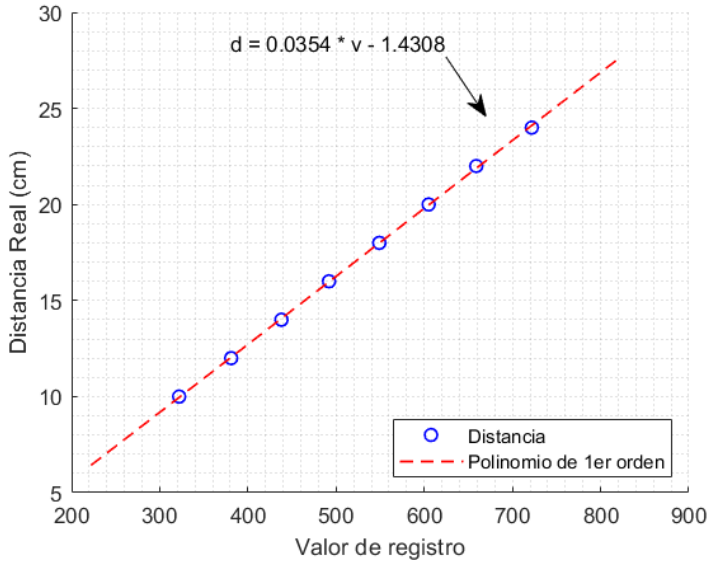


Se obtuvo la relación entre el valor del registro 9 y la distancia real. La curva de calibración se presenta en la Figura 59.

$$distancia = 0.0354 * valor - 1.4308 \quad (58)$$

Figura 59

Curva de Calibración de Sensor de Nivel de Agua con salida RS485.



3.2.2 Sensor Ultrasónico Allen Bradley 873M-D18AV800-D4

Con el objetivo de recopilar los datos necesarios para el proceso de calibración, se desarrolló un programa en LabVIEW. En este programa, se empleó el bloque “DAQ Assistant”, configurado de manera que el NI DAQ USB 6211 lea el voltaje proveniente del sensor ultrasónico industrial a través de una de sus entradas analógicas. El programa correspondiente se presenta en la Figura 60.

En el procedimiento de calibración, se recopilaron datos a diferentes distancias de un objeto. Las lecturas se llevaron a cabo a una frecuencia de 10 Hz durante un período de 5 segundos. La Tabla 14 exhibe la información obtenida durante el primer segundo de medición de cada prueba.

Los datos se guardaron en un archivo de texto, el cual es posteriormente importado por un script de Matlab. En dicho script, se emplea la función *polyfit* para obtener los coeficientes del polinomio de primer grado que mejor se ajuste a los puntos por Método de Mínimos Cuadrados. El código de Matlab se presenta a continuación:

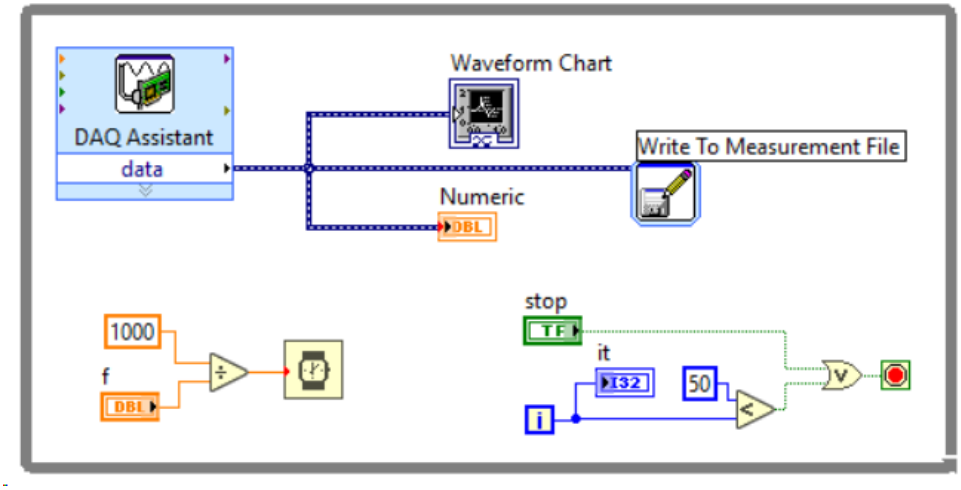
Tabla 14

Registro de datos para calibración de Sensor Ultrasónico Industrial.

N° Muestra	Medida de voltaje a diferentes distancias				
	50 cm	40 cm	30 cm	20 cm	10 cm
1	7.8880	8.3544	8.8211	9.2869	9.7555
2	7.8900	8.3541	8.8205	9.2892	9.7542
3	7.8900	8.3541	8.8178	9.2832	9.7552
4	7.8900	8.3544	8.8221	9.2869	9.7546
5	7.8907	8.3537	8.8238	9.2878	9.7516
6	7.8900	8.3544	8.8251	9.2878	9.7542
7	7.8887	8.3557	8.8205	9.2869	9.7539
8	7.8900	8.3547	8.8198	9.2862	9.7542
9	7.8893	8.3570	8.8198	9.2846	9.7559
10	7.8887	8.3537	8.8201	9.2842	9.7529

Figura 60

Programa en LabVIEW para calibración de Sensor Ultrasónico Industrial.




```
clc, clearvars, close all;

% ============================================================

% CALIBRACIÓN DE SENSOR DE PROXIMIDAD ULTRASÓNICO

% ============================================================

% Obtención de Coeficientes de Polinomio ajustado por Mínimos Cuadrados

data = load("DataUltrasonicoIndustrial.txt")
```

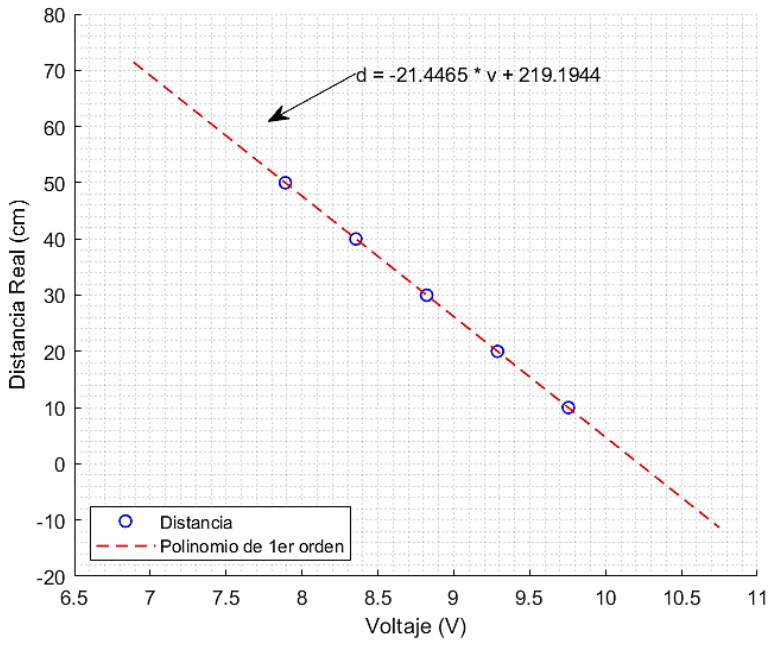
```
voltage = mean(data); % Voltaje promedio de cada prueba
distance = [50 40 30 20 10]; % Distancia real
coef = polyfit(voltage,distance,1); % Polinomio ajustado
a1 = coef(1) % Pendiente
a0 = coef(2) % Intercepto
```

Se obtuvo una relación entre el voltaje proveniente del sensor ultrasónico y la distancia real. La curva de calibración se presenta en la Figura 61.

$$distancia = -21.4465 * voltaje + 219.1944 \quad (59)$$

Figura 61

Curva de Calibración de Sensor Ultrasónico Industrial.



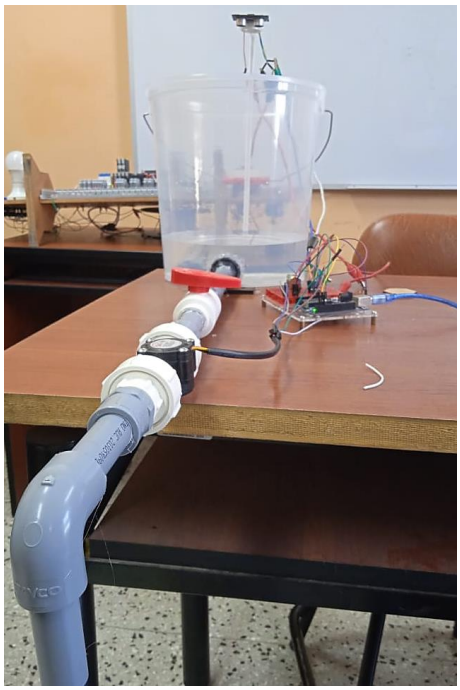
3.2.3 Sensor de Flujo YF-S201

Para calibrar el sensor de flujo, se requiere calcular el factor K, el cual convierte la cantidad de pulsos por segundo en la magnitud física de caudal (L/min). Para lograr esto, se llevaron a cabo diversas pruebas para contar el número de pulsos generados por el

paso de un volumen específico de agua a través del sensor de flujo. Con este propósito, se construyó un prototipo pequeño para realizar las pruebas y obtener los datos necesarios. En la Figura 62 se ilustra dicho prototipo.

Figura 62

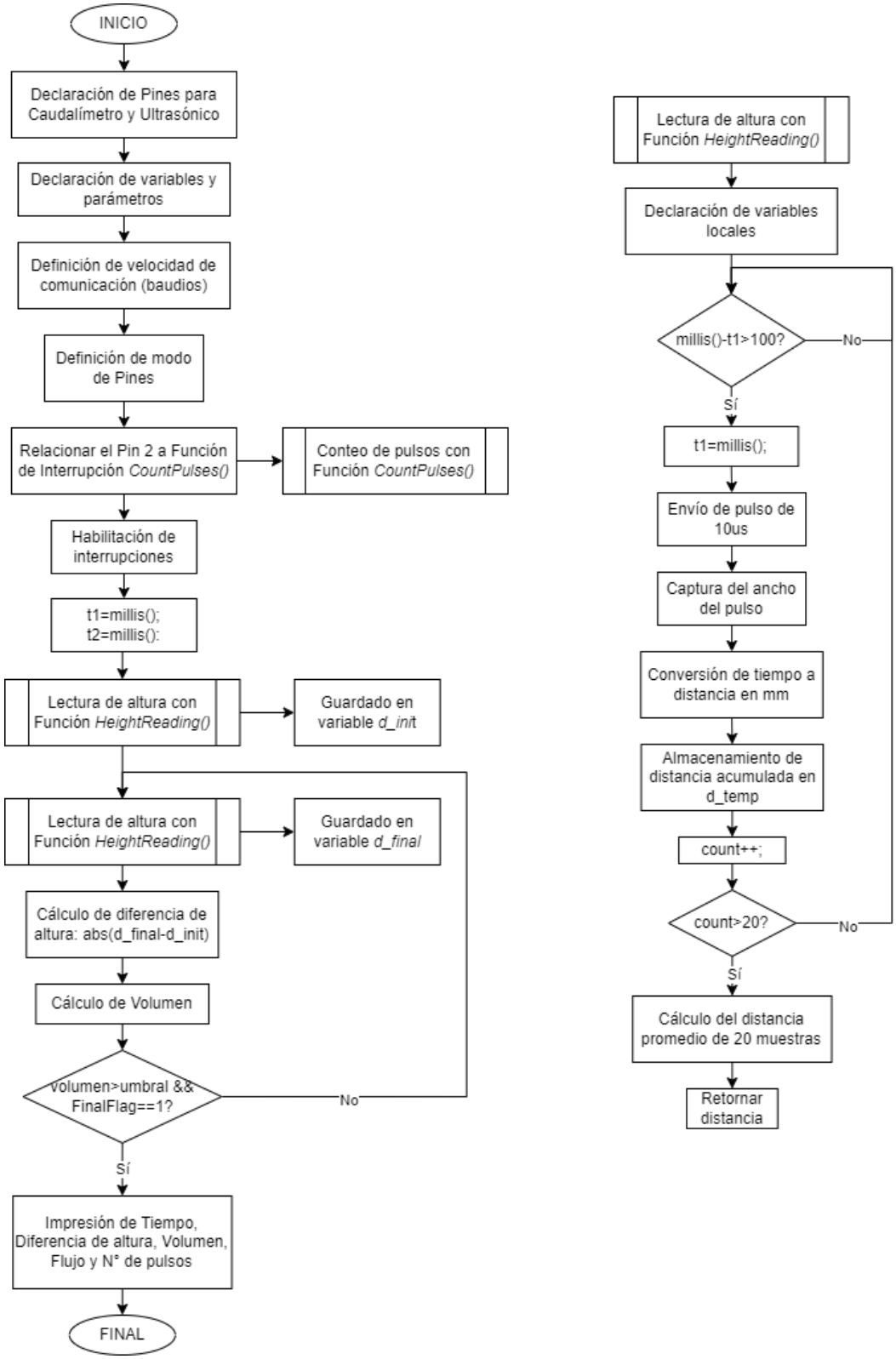
Prototipo de pruebas para obtención de factor de conversión K.



Se implementó un código en Arduino IDE. Este código cuenta los pulsos generados por el sensor de flujo mediante el pin de interrupción 2 de Arduino. Además, se empleó un sensor ultrasónico para medir el volumen de agua desplazado, calculando la diferencia de nivel al inicio y al final, junto con el área conocida del recipiente. El diagrama de flujo del programa para calcular el factor de conversión K del sensor de flujo se muestra en la Figura 63.

Figura 63

Diagrama de flujo del código para calcular el factor de conversión K de sensor de flujo YF-S201.



Se realizó diez pruebas y se registraron los datos en la Tabla 15. El factor de conversión K se obtiene de la división entre el número de pulsos y el volumen de agua desplazado.

Tabla 15

Registro de pruebas para calibración de Sensor de flujo YF-S201.

N° Pulsos	Volumen (L)	Factor de conversión K
484	1	484
483	1.01	478.2178
619	1.32	468.9394
619	1.36	455.1471
754	1.61	468.3230
745	1.54	483.7662
835	1.82	458.7912
821	1.79	458.6592
958	2.13	449.7653
965	2.15	448.8372

El factor de conversión promedio de las pruebas fue

$$K_H = 465.4446 \tag{60}$$

Este factor posibilita el cálculo del caudal en litros por hora (L/H) a partir del número de pulsos. Para obtener el caudal en litros por minuto (L/min), se dividió el factor entre 60, resultando en el siguiente valor:

$$K_M = 7.7574 \tag{61}$$

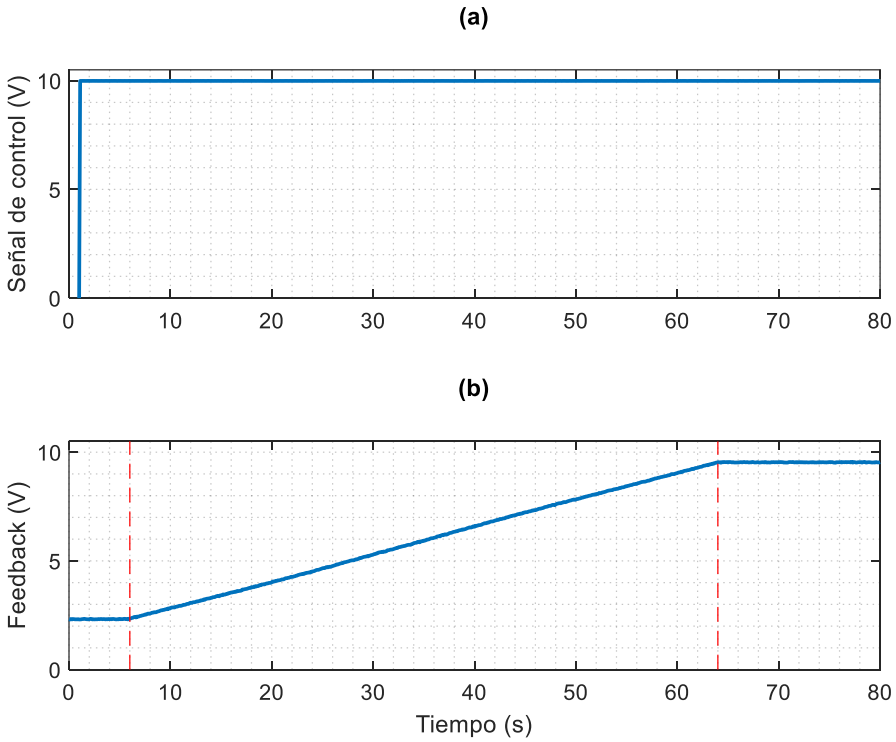
3.2.4 Válvula Proporcional de 2 vías TFV4-304

Para comprender el comportamiento de la válvula proporcional, se le aplicó una señal escalón de 10V, destinada a abrir completamente la válvula. La Figura 64 muestra tanto la señal aplicada como la señal capturada. Se observa que la respuesta de la válvula

presenta un retraso de 5 segundos y requiere 58 segundos para abrirse por completo. Además, la señal de retroalimentación de la válvula tiene un rango de 2.3V a 9.6V, donde 2.3V indica que la válvula está cerrada y 9.6V indica que la válvula está completamente abierta.

Figura 64

Gráfico de comportamiento de la apertura de válvula proporcional: (a) Señal de control; (b) Señal de realimentación.



3.2.5 Bomba de agua Pedrollo PKM60

El correcto conexionado de tuberías es crucial para el funcionamiento eficiente de la bomba. A continuación, se detallan los aspectos importantes:

- Tubería de Succión: Conecta la entrada de la bomba al Tanque 3. Se aseguro que esta tubería sea hermética para evitar la entrada de aire.

- Tubería de Descarga: Conecta la salida de la bomba al Tanque 1. Se aseguró que esta tubería soporte la presión de operación de la bomba.
- Válvula Antirretorno: Se instaló una válvula antirretorno en la tubería de succión para evitar el flujo inverso cuando la bomba se apaga. Esto ayuda a mantener la bomba cebada y reduce el riesgo de daños.

En este trabajo no se abarcó el control del flujo de la bomba, por lo que el flujo de entrada al Tanque 1, cuando la bomba está encendida, es aproximadamente constante. Por otro lado, el flujo que ingresa al Tanque 2 está determinado tanto por el nivel de agua del Tanque 1 como por el porcentaje de apertura de la válvula proporcional. Por lo tanto, para que el control del nivel del Tanque 2 sea posible, es necesario que la bomba se encienda bajo ciertas condiciones que eviten afectar de manera abrupta el flujo de entrada al mismo.

En la Figura 65, se ilustra la dinámica del nivel de agua del Tanque 2 para distintos niveles iniciales de agua en el Tanque 1, manteniendo la válvula de control completamente abierta.

Se eligió un rango de operación para el nivel del Tanque 1, que va desde 15 cm hasta 17.5 cm. Esta elección se basó en la observación de que la diferencia entre los niveles de equilibrio es de aproximadamente 1 cm. Esta medida asegura que el nivel del Tanque 1 no afecte significativamente el flujo de entrada al Tanque 2, cumpliendo así con el propósito establecido.

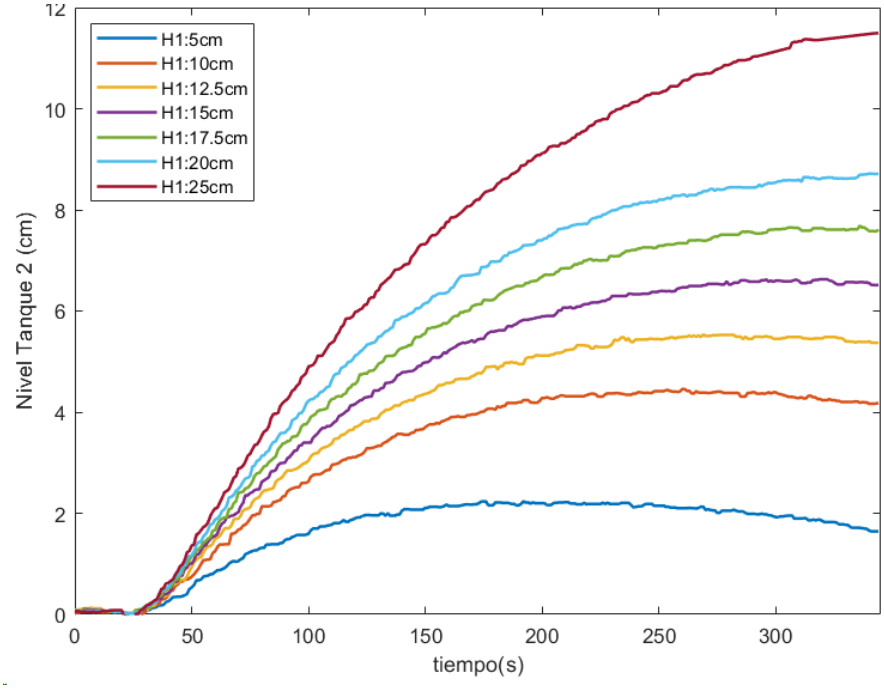
3.3 Identificación mediante Modelos Wiener y Hammerstein

En esta sección, se implementan los métodos de identificación discutidos en el Capítulo II. Para obtener el modelo mediante estas técnicas, es esencial analizar algunas características fundamentales del proceso a identificar, lo cual implica realizar ensayos preliminares. Una vez completados estos análisis iniciales, se diseñan las señales de

entrada más apropiadas para aplicar al proceso. A continuación, se planifica y lleva a cabo el experimento, y finalmente, se aplican los algoritmos de identificación. La comparación de los resultados demuestra que el modelo Hammerstein identificado con FLANN resulta ser el más adecuado.

Figura 65

Comparativa de la dinámica del nivel del Tanque 2 ante diferentes niveles iniciales del Tanque 1.

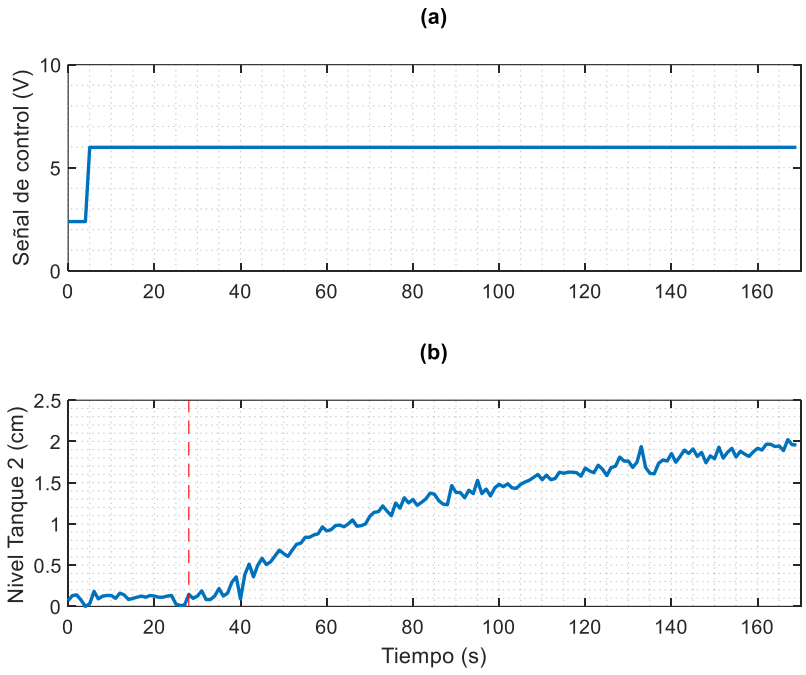


3.3.1 Ensayos Preliminares para la Identificación

Para la identificación del sistema, es crucial establecer un periodo de muestreo adecuado para la adquisición de datos. Para ello, se realizó la siguiente prueba: se aplicó una señal escalón de 6V a la válvula proporcional para analizar el comportamiento del sistema. La Figura 66 muestra tanto la señal escalón como la respuesta del sistema. En esta prueba, se utilizó un periodo de muestreo de 1 segundo. Cabe destacar que se observó un retraso de 24 segundos en la respuesta del sistema.

Figura 66

Prueba para determinar el periodo de muestreo: (a) Señal de control escalón de 6V; (b) Respuesta del sistema.



Según el criterio establecido en la ecuación (2) para determinar el tiempo de muestreo, es necesario obtener la constante de tiempo (τ) de una aproximación de primer orden del sistema. Se utilizó el modelo ARX para identificar el sistema, y mediante un código desarrollado en Matlab, se obtuvo la función de transferencia de primer orden en tiempo continuo (Ecuación (62)), alcanzando una precisión del 82.92%, como se muestra en la Figura 67. La precisión del modelo es aceptable en esta etapa preliminar, ya que el único objetivo es estimar la constante de tiempo para seleccionar el periodo de muestreo óptimo. El ajuste moderado se explica por las limitaciones inherentes del modelo ARX para sistemas no lineales y el ruido presente en los datos.

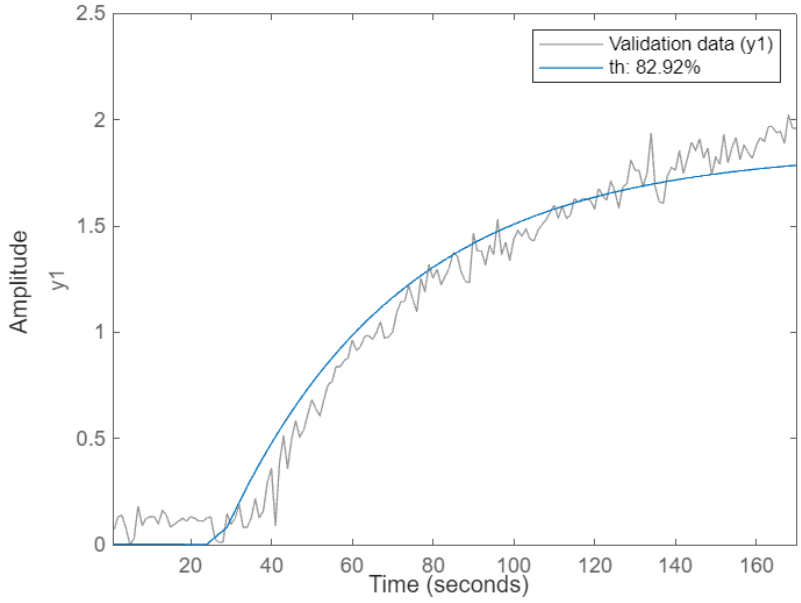
$$G_p(s) = \frac{0.3097}{43.55s + 1} e^{-24s} \tag{62}$$

La constante de tiempo es de 43.55 segundos. Según el criterio mencionado, se recomienda que el periodo de muestreo esté en el rango de [4.355, 8.71] segundos. Para la adquisición de datos, se seleccionó un periodo de muestreo de 4 segundos.

$$T_s = 4s \tag{63}$$

Figura 67

Aproximación de la respuesta del proceso a un sistema de primer orden.



3.3.2 Diseño de las Señales de Prueba para la Identificación

Una vez determinado el periodo de muestreo, se procede a aplicar a la planta las señales de prueba específicas propuestas en (Park et al., 2006). Como se mencionó en la sección anterior, la válvula proporcional requiere un voltaje de control superior a 2.3V para comenzar a abrirse (ver Figura 66(a)). Por este motivo, para mayor comodidad, se realizará una conversión previa de la señal de control de la válvula proporcional, de modo que se trabaje con el porcentaje de apertura en lugar de directamente con el voltaje. La relación será la siguiente:

$$voltaje = \left(\frac{v_{máx} - v_{mín}}{100\%}\right)\%_{apertura} + v_{mín} \tag{64}$$

Donde: $v_{máx} = 9.6V$ es el voltaje medido cuando la válvula está completamente abierta y $v_{mín} = 2.3V$ es el voltaje medido cuando la válvula está cerrada.

1. La primera señal de prueba, u_b, es una señal binaria pseudo aleatoria con dos valores: cero y una constante de 80% de apertura de la válvula, dicha señal se muestra en la Figura 68.
2. La segunda señal de prueba, $u_{b\alpha}$, es una señal múltiplo α de la primera señal de prueba. Se eligió que α sea 0.6, dicha señal se ilustra en la Figura 69.
3. La tercera señal de prueba, u_{ms}, está compuesta por la primera señal de prueba (u_b) y una señal aleatoria uniformemente distribuida entre 0 y 100% de apertura de la válvula, la mencionada señal se presenta en la Figura 70.

Figura 68

Señal de prueba binaria u_b.

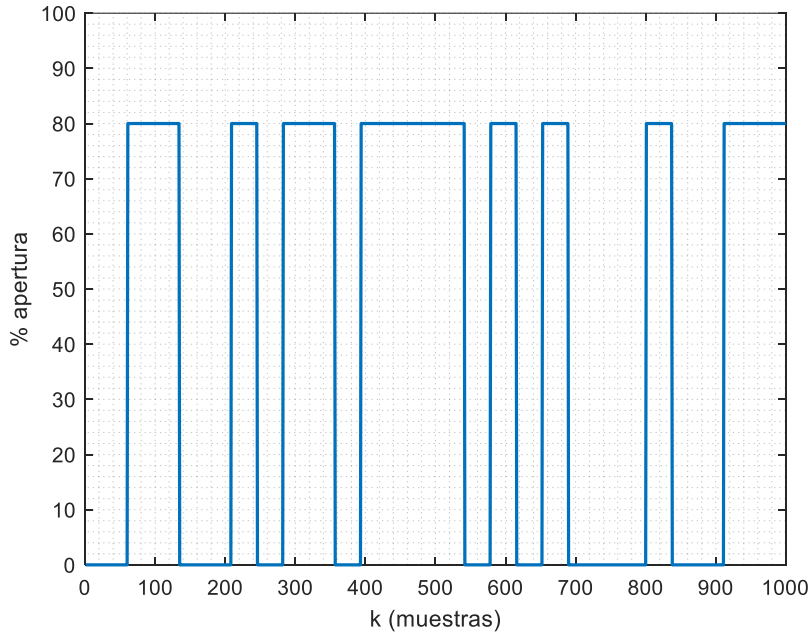


Figura 69

Señal de prueba binaria $u_{b\alpha}$.

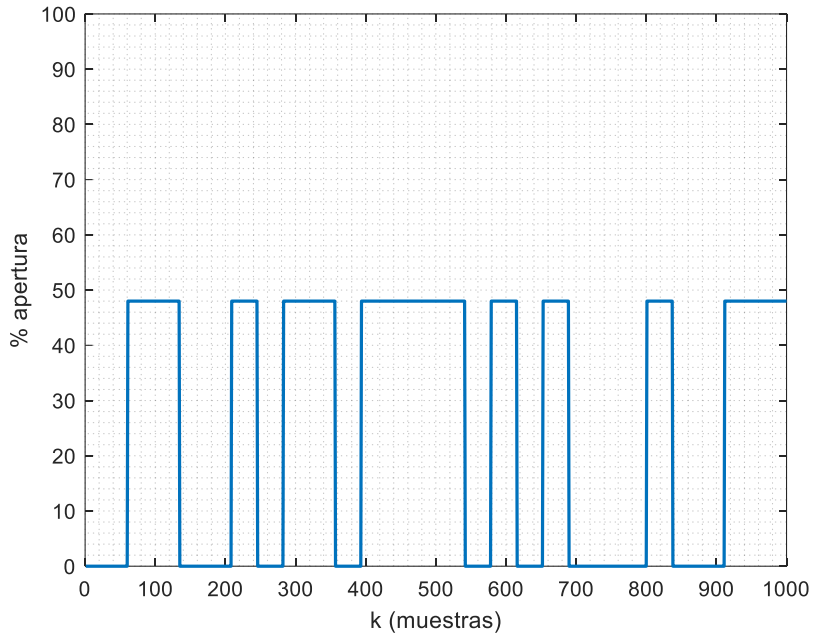
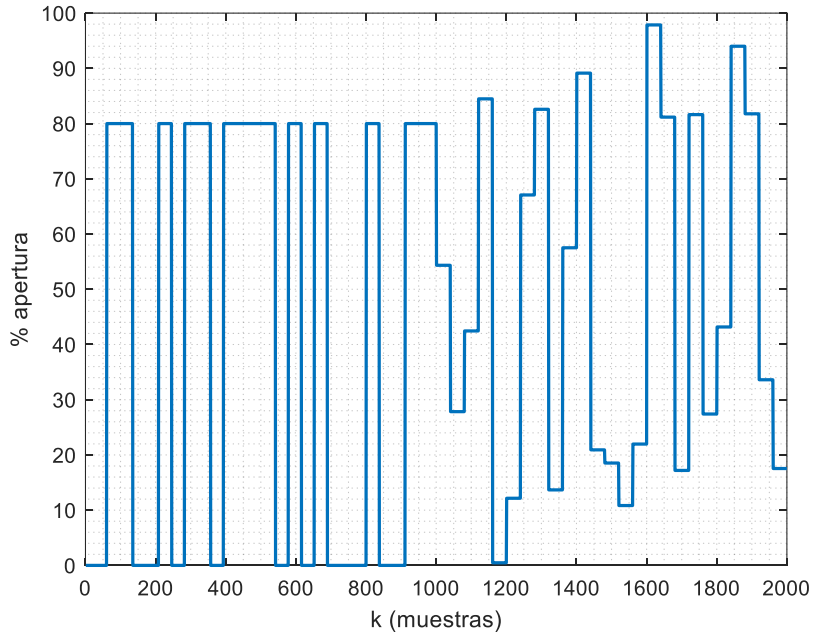


Figura 70

Señal de prueba multipaso u_{ms}.



Por otro lado, antes de proceder con la identificación no lineal del sistema de tanques en cascada utilizando el Modelo Hammerstein-Wiener, se describen las condiciones iniciales establecidas para la realización de las pruebas de identificación:

- Tanque 1 con un nivel de agua de 17.5cm.
- Tanque 2 con un nivel de agua de 0 cm.
- Válvula proporcional completamente cerrada.

3.3.3 Identificación del Modelo Wiener

Las ecuaciones que describen el proceso no lineal siguiendo el Modelo Wiener se presentan a continuación:

$$z[k] = \sum_{i=1}^{n_a} a_i z[k-i] + \sum_{j=1}^{n_b} b_j u[k-j-K_d] \tag{65}$$

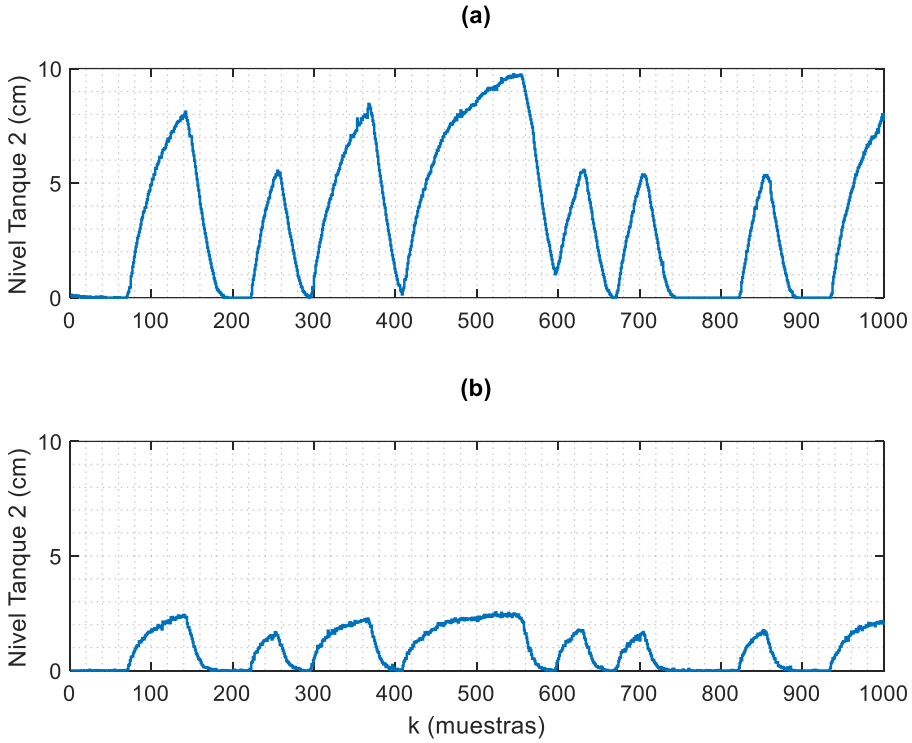
$$z[k] = g^{-1}(y[k]) = q_1 y[k] + q_2 y^2[k] + \cdots + q_r y^r[k] \tag{66}$$

Donde, $u[k]$ e $y[k]$ denotan el vector de entrada y salida del proceso en el instante k, respectivamente. Las ecuaciones (65) y (66) representan el subsistema dinámico lineal y la función estática no lineal de salida, respectivamente. La variable intermedia $z[k]$ es la salida del subsistema dinámico lineal en el instante k. K_d es el número de pasos de tiempo de retraso.

Función Estática no Lineal de Salida. Según lo explicado en la Sección 2.2.2, para identificar la función estática no lineal de salida se utiliza las dos primeras señales de prueba. La Figura 71 ilustra las respuestas del proceso ante las señales de excitación de entrada u_b y $u_{b\alpha}$. La experiencia de la adquisición de datos tuvo una duración de 4000 segundos con un periodo de muestreo de 4 segundos, por tanto, la cantidad de muestras (N_b) es de 1000 muestras.

Figura 71

Respuestas del proceso ante señales de prueba: (a) Señal de salida y_b; (b) Señal de salida $y_{b\alpha}$.



La ecuación (66) es representada en forma vectorial de la siguiente manera.

$$z[k] = g^{-1}(y[k]) = Q^T \times Y[k] \tag{67}$$

Donde

$$Q = [q_1, q_2, \dots, q_r]^T \tag{68}$$

$$Y[k] = \left[y[k], y^2[k], \dots, y^r[k]\right]^T \tag{69}$$

Aquí, Q es el vector de coeficientes de la función estática no lineal de salida y $Y[k]$ es el vector que contiene la salida del proceso en el instante k elevada a potencias consecutivas, empezando por 1 hasta r.

La variable intermedia $z[k]$ es la salida del subsistema dinámico lineal, sigue la regla de superposición, por tanto, se cumple:

$$z_{b\alpha}[k] = \beta z_b[k] \tag{70}$$

Según Park (2006), la estimación de los parámetros de la función no lineal estática de salida se logra a partir de minimizar la función de costo, que en este caso es el MSE.

$$V(\hat{\beta}, \hat{Q}) = \frac{1}{2}\left(\hat{z}_{b\alpha}[k] - \hat{\beta}\hat{z}_b[k]\right)^2 \tag{71}$$

Donde:

$$e(k) = \hat{z}_{b\alpha}(k) - \hat{\beta}\hat{z}_b(k) \tag{72}$$

$$\hat{z}_{b\alpha}(k) = \hat{q}_1 y_{b\alpha}(k) + \hat{q}_2 y_{b\alpha}^2(k) + \cdots + \hat{q}_r y_{b\alpha}^r(k) = \hat{Q}^T \times Y_{b\alpha}(k) \tag{73}$$

$$\hat{z}_b(k) = \hat{q}_1 y_b(k) + \hat{q}_2 y_b^2(k) + \cdots + \hat{q}_r y_b^r(k) = \hat{Q}^T \times Y_b(k) \tag{74}$$

Aquí, $\hat{Q}$ es el vector de coeficientes estimado de la función estática no lineal de salida, $Y_{b\alpha}[k]$ e $Y_b[k]$ son el vector $Y[k]$ obtenido de las señales de excitación $u_{b\alpha}[k]$ y $u_b[k]$, respectivamente.

Se estimaron los coeficientes de la función no lineal estática de salida mediante una RNA de una sola neurona, la estructura se detalla en la Figura 72.

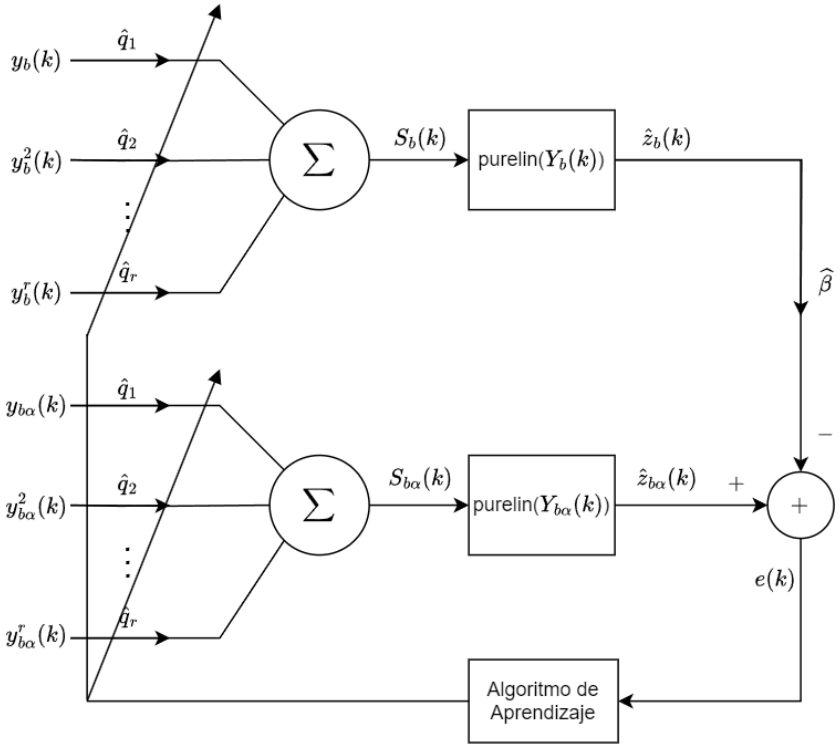
La función de activación utilizada es *purelin*. Para minimizar la función de costo, se empleó el algoritmo de Descenso del gradiente. Esto implica calcular la derivada parcial de la función de costo con respecto a sus variables, es decir, respecto a $\hat{\beta}$ y al vector de coeficientes de la función no lineal estática ($\hat{Q}$), utilizando la regla de la cadena.

$$\frac{\partial V}{\partial \hat{\beta}} = \frac{\partial V}{\partial e}\frac{\partial e}{\partial \hat{\beta}} = e[k](-\hat{z}_b[k]) \tag{75}$$

$$\frac{\partial V}{\partial \hat{Q}} = \frac{\partial V}{\partial e}\frac{\partial e}{\partial \hat{Q}} = e[k]\left(Y_{b\alpha}[k] - \hat{\beta}Y_b[k]\right) \tag{76}$$

Figura 72

Esquema de la RNA para Identificar el Modelo Wiener.



La actualización de los parámetros de la RNA se realiza en dirección opuesta al gradiente de la función de costo, empleando una tasa de aprendizaje (η) para controlar el tamaño del paso de actualización. Este proceso se lleva a cabo para cada patrón de entrenamiento (Modo patrón). Las ecuaciones utilizadas para la actualización de los parámetros son las siguientes:

$$\hat{\beta}[k+1] = \hat{\beta}[k] - \eta \frac{\partial V}{\partial \hat{\beta}} \tag{77}$$

$$\hat{Q}[k+1] = \hat{Q}[k] - \eta \frac{\partial V}{\partial \hat{Q}} \tag{78}$$

Esta actualización se aplica iterativamente hasta que la función de costo converja a un valor mínimo aceptable.

A partir de las ecuaciones mostradas, se desarrolló un script en MATLAB (ver Anexo N°3) para entrenar la RNA y estimar los coeficientes de la función estática no lineal de salida. Para evitar el sobreajuste se dividió los datos en datos de entrenamiento y validación. La bondad del ajuste se evaluó utilizando dos estadísticos: NRMSE y MSE. En la Tabla 16 se presentan las pruebas realizadas para distintas tasas de aprendizaje y la precisión de cada una. Se consideró que la función de costo converja a un valor menor que 0.02.

Tabla 16

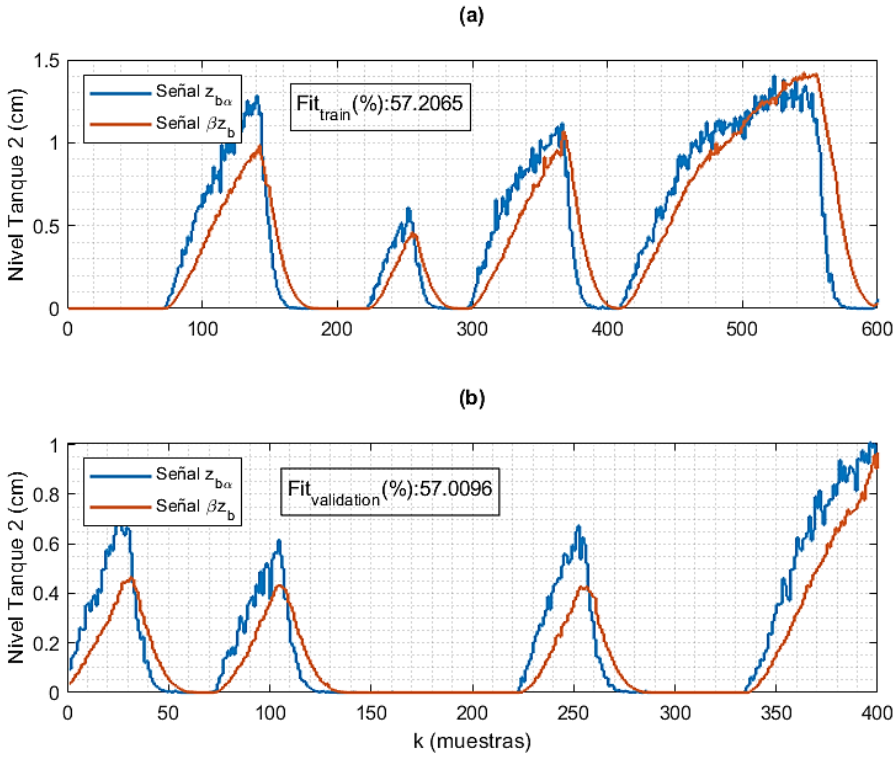
Evaluación de la bondad del ajuste para diferentes tasas de aprendizaje en la estimación de la función no lineal de salida.

N°	Forma de la función	η	Épocas	Datos de Entrenamiento		Datos de Validación	
				NRMSE (%)	MSE	NRMSE (%)	MSE
1		$1e^{-4}$	21	50.6207	0.0347	45.0843	0.0158
2	$z = q_2 y^2$	$1e^{-5}$	122	57.0459	0.0399	55.5816	0.0157
3		$1e^{-6}$	1057	57.2065	0.04	57.0096	0.0148
4		$1e^{-4}$	9	21.5975	0.022	18.1126	0.0058
5	$z = q_3 y^3$	$1e^{-5}$	58	45.1427	0.0386	37.1328	0.0123
6		$1e^{-6}$	331	54.7474	0.0399	50.5647	0.0115

La elección de las funciones utilizadas para las pruebas se basó en la necesidad de que la función estática no lineal del Modelo Wiener fuera invertible, como se señaló en la Sección 2.2.2. Se realizaron pruebas con polinomios de segundo y tercer orden, ya que los órdenes superiores no proporcionaban una buena aproximación. Como se observa en la Tabla 16, la prueba 3 es la que presenta la precisión más alta tanto en los datos de entrenamiento como en los de validación. En la Figura 73 se muestra la prueba 3, en la que se obtuvo un ajuste de la señal $\hat{\beta} z_b$ y $z_{b\alpha}$ con una precisión del 57.0096% en los datos de validación.

Figura 73

Gráfico de la bondad del ajuste entre $\hat{\beta} z_b$ y $z_{b\alpha}$: (a) Datos de entrenamiento; (b) Datos de Validación.



La RNA se entrenó para minimizar la función de costo, con el objetivo de estimar el vector de coeficientes de la función estática no lineal de salida ($\hat{Q}$) y el parámetro $\hat{\beta}$. Los valores obtenidos fueron:

$$\hat{\beta} = 0.0686 \tag{79}$$

$$\hat{Q} = [0, 0.2177]^T \tag{80}$$

Una vez estimado el vector $\hat{Q}$, se determinó la función estática no lineal de salida, la cual se muestra a continuación:

$$z[k] = g^{-1}(y[k]) = 0.2177\, y^2[k] \tag{81}$$

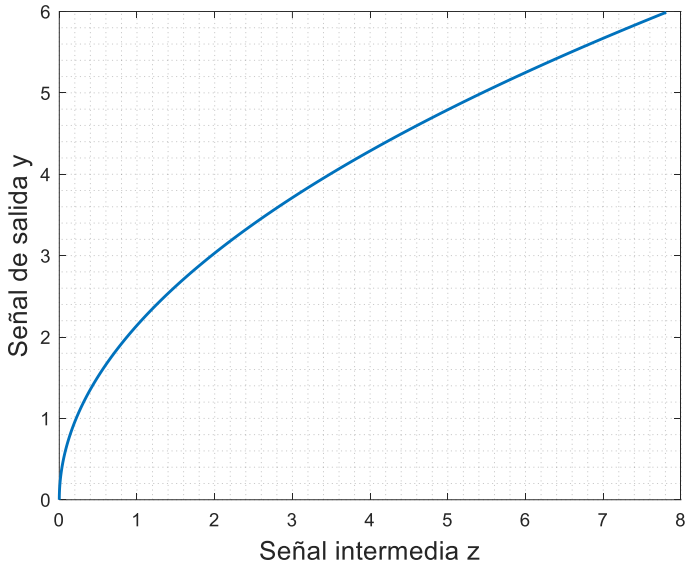
Como se explicó anteriormente, es posible invertir esta función para calcular $y[k]$ a partir de $z[k]$:

$$y[k] = g(z[k]) = 2.1432\sqrt{z[k]} \tag{82}$$

En la Figura 74 se muestra la no linealidad identificada del Modelo Wiener.

Figura 74

No linealidad del Modelo Wiener.



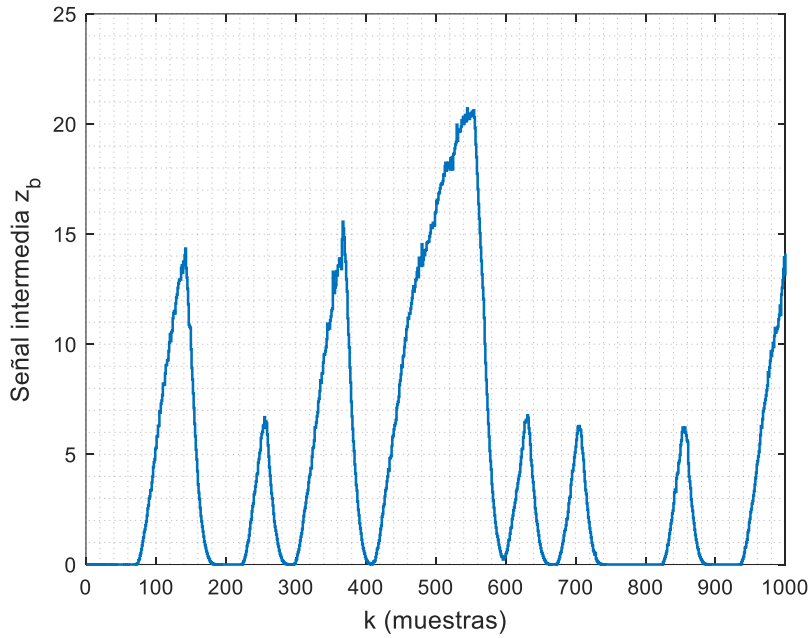
Subsistema Dinámico Lineal. Para identificar el subsistema dinámico lineal, se utiliza la señal de excitación u_b (ver Figura 68) y la señal intermedia estimada $\hat{z}_b$, la cual se calcula mediante la función estática de salida (Ecuación (81)) encontrada anteriormente, dicha señal se ilustra en Figura 75.

Para determinar el modelo matemático que describa el comportamiento del subsistema dinámico lineal, se empleó el Modelo ARX. Se seleccionó una aproximación de segundo orden para el proceso ($n_a = 2$ y $n_b = 1$). En la Sección III-C-1, se determinó que el retraso de la respuesta del sistema es de 24 segundos. El tiempo de muestreo seleccionado es de 4 segundos, el número de muestras de retraso sería 6 ($n_k = 6$). La

Figura 76 muestra la comparación entre la señal intermedia estimada $\hat{z}_b[k]$ y la salida del Modelo ARX identificado (ver código MATLAB en el Anexo N°3). En conclusión, para esta identificación se utilizó el Modelo ARX con los parámetros $n_a = 2$, $n_b = 1$, y $n_k = 6$. La precisión del modelo es del 31.5909%.

Figura 75

Señal intermedia estimada $\hat{z}_b[k]$.



La función de transferencia en tiempo discreto del Modelo ARX identificado se muestra a continuación

$$G_{arx}(z) = \left(\frac{0.002849z^{-1}}{1 - 1.274z^{-1} + 0.2897z^{-2}}\right)z^{-5} \tag{83}$$

El modelo ARX se expresa en ecuaciones en diferencias a partir de la Ecuación (83) de la siguiente manera:

$$\hat{y}[k] = \sum_{i=1}^{2} \hat{a}_i \hat{y}[k-i] + \hat{b}\hat{v}[k-6] \tag{84}$$

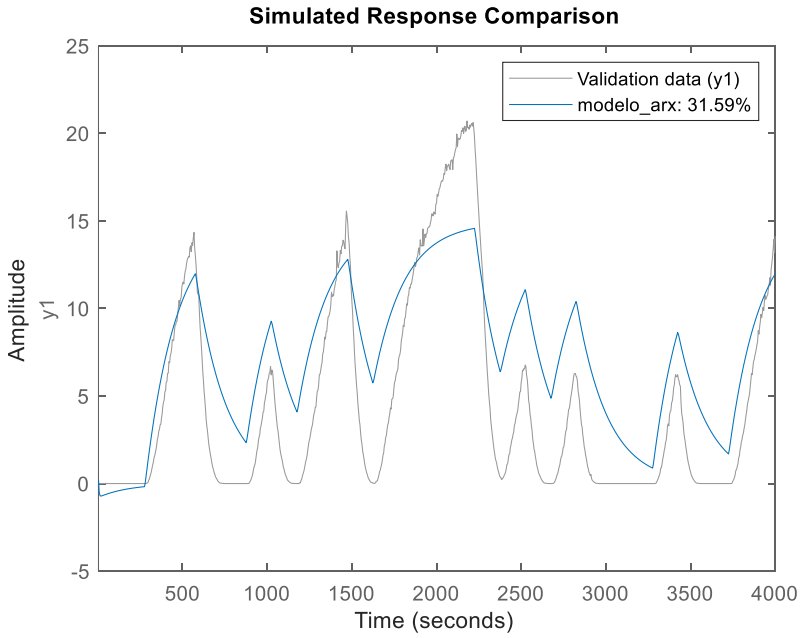
Donde:

$$\hat{a} = [1.274, -0.2897]^T \tag{85}$$

$$\hat{b} = 0.002849 \tag{86}$$

Figura 76

Gráfico comparativo entre $\hat{z}_b$ y Modelo ARX.



El modelo Wiener que representa el sistema de tanques en cascada se obtiene conectando en serie el subsistema dinámico lineal identificado con la función estática no lineal de salida. La Figura 77 muestra la comparación entre la salida del proceso real y la salida del Modelo Wiener identificado (ver código MATLAB en el Anexo N°3). La precisión del modelo Wiener es del 17.1324%.

3.3.4 Identificación del Modelo Hammerstein

Las ecuaciones que describen el proceso no lineal siguiendo el Modelo Hammerstein se presentan a continuación:

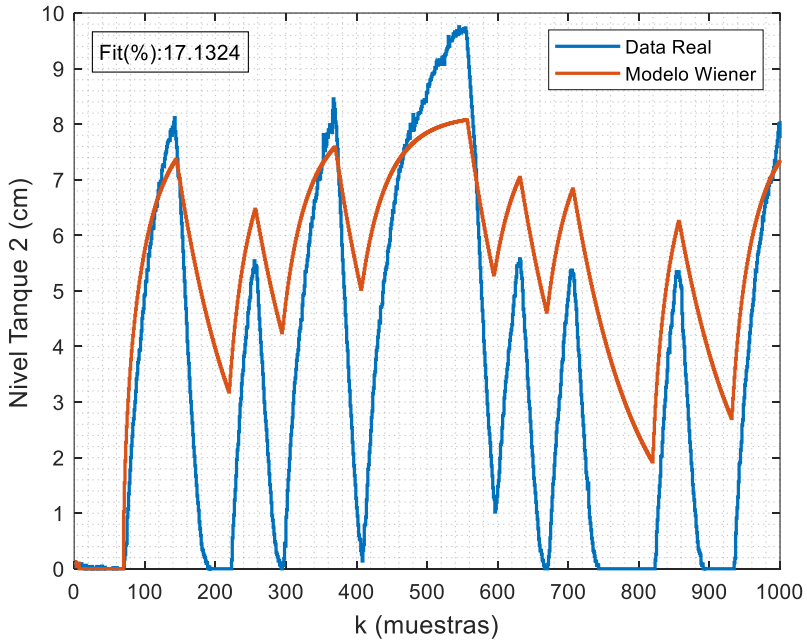
$$v[k] = f(u[k]) = p_1 u[k] + p_2 u^2[k] + \cdots + p_m u^m[k] \tag{87}$$

$$y[k] = \sum_{i=1}^{n_a} a_i y[k-i] + \sum_{j=1}^{n_b} b_j v[k-j-K_d] \tag{88}$$

Donde, $u[k]$ e $y[k]$ denotan el vector de entrada y salida del proceso en el instante k, respectivamente. Las ecuaciones (87) y (88) representan la función estática no lineal de entrada y el subsistema dinámico lineal, respectivamente. La variable intermedia $v[k]$ es la salida de la función estática no lineal de entrada en el instante k. K_d es el número de pasos de tiempo de retraso.

Figura 77

Gráfico comparativo entre Proceso real y Modelo Wiener.



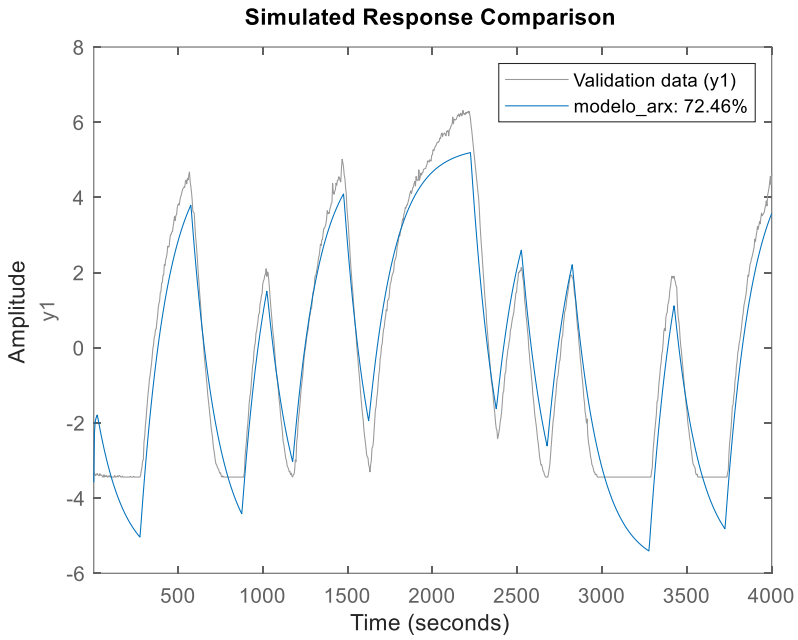
Subsistema dinámico lineal

Como se explicó en la Sección II-B-2-d es posible asumir que la entrada binaria del subsistema dinámico lineal es $v_b[k] = u_b[k]$ sin pérdida de generalidad. De este modo, se identificó directamente el subsistema dinámico lineal a partir de la señal de entrada u_b y la señal de salida y_b. Para determinar el modelo matemático que describa el comportamiento

del subsistema dinámico lineal, se empleó el Modelo ARX. La Figura 78 muestra la comparación entre la salida del proceso real y la salida del Modelo ARX identificado (ver código MATLAB en el Anexo N°4). Para la identificación del Modelo ARX se utilizó los parámetros $n_a = 2$, $n_b = 1$, y $n_k = 6$. La precisión del modelo ARX es del 72.4595%.

Figura 78

Gráfico comparativo entre y_b y Modelo ARX.



Es importante señalar que, en el código de MATLAB, antes de la identificación del Modelo ARX, se utilizó la función '*detrend'* para sustraer los desplazamientos o tendencias de los datos de entrada y salida. Esto es fundamental debido a que dichas tendencias introducen sesgos en el análisis de los modelos y afectan negativamente la precisión de los resultados.

La función de transferencia en tiempo discreto del Modelo ARX identificado se muestra a continuación

$$G_{arx}(z) = \frac{Y(z)}{V(z)} = \left(\frac{0.002851z^{-1}}{1 - 1.163z^{-1} + 0.1834z^{-2}}\right) z^{-5} \tag{89}$$

El Modelo ARX se expresa en ecuaciones en diferencias a partir de la Ecuación (89) de la siguiente manera:

$$\hat{y}[k] = \sum_{i=1}^{2} \hat{a}_i \hat{y}[k-i] + \hat{b}\hat{v}[k-6] \tag{90}$$

Donde:

$$\hat{a} = [1.163, -0.1834]^T \tag{91}$$

$$\hat{b} = 0.002851 \tag{92}$$

Función estática no lineal de entrada

Para la identificación de la función no lineal de entrada, se excitó la planta con la tercera señal de prueba u_{ms}, la cual tiene una duración de 8000 segundos. El periodo de muestreo es de 4 segundos, se obtuvieron 2000 muestras (N_{ms}). La Figura 79 muestra la señal de salida del proceso (y_{ms}) ante la señal de excitación (u_{ms}).

La función estática no lineal de entrada se determinó mediante una FLANN. Esta red se compone de una única neurona y una expansión de funciones que actúa en cada elemento del vector de entrada normalizado entre los valores de -1 y 1. Se seleccionó los polinomios de Chebyshev como función base para la expansión de funciones de acuerdo con lo explicado en la Sección 2.2.3. La función de activación de la FLANN es el subsistema dinámico lineal encontrado en el paso anterior. Finalmente, se añadió una saturación para que la salida del proceso esté entre 0 y 30 cm. La Figura 80 ilustra la estructura detallada de la FLANN.

Figura 79

Señal de salida y_{ms}.

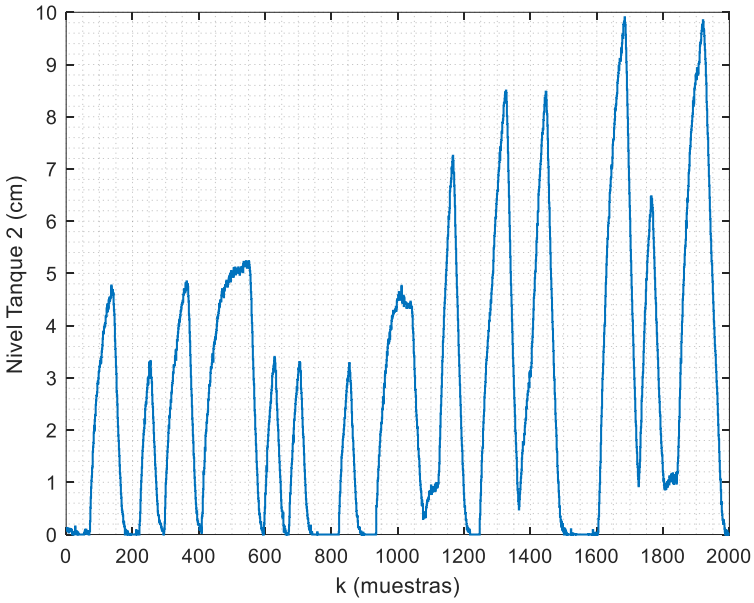
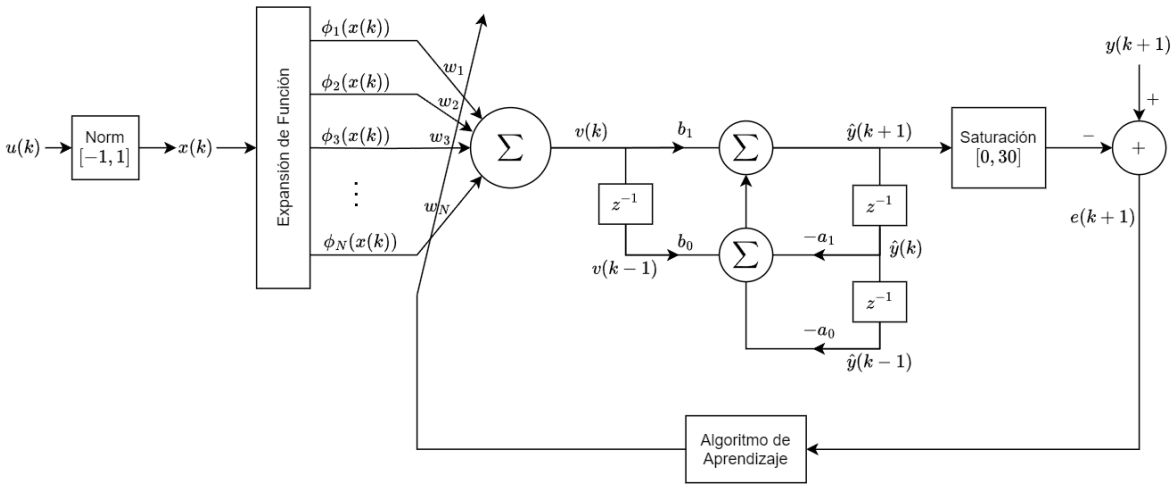


Figura 80

Estructura de la FLANN para Identificación de Modelo Hammerstein.



La identificación de la función no lineal está relacionada con la determinación de los pesos óptimos de la FLANN. Esto se logra minimizando la siguiente función de costo:

$$V(W) = \frac{1}{2}e^2[k] = \frac{1}{2}(y[k] - \hat{y}[k])^2 \tag{93}$$

Donde:

$$\hat{y}[k] = \sum_{i=1}^{2} \hat{a}_i \hat{y}[k-6] + \hat{b}\hat{v}[k-6] \tag{94}$$

$$\hat{v}[k] = \sum_{l=1}^{N_{ms}} w_l \phi_l(u[k]) \tag{95}$$

y $W = [w_1, w_2, \dots, w_N]^T$ es el vector de pesos de la FLANN. El vector expandido de la FLANN para la entrada $u(k-6)$.

$$u_{FE}[k] = [\phi_1(u[k-6]), \phi_2(u[k-6]), \dots, \phi_N(u[k-6])]^T \tag{96}$$

La Ecuación (94) se reescribe como sigue

$$\hat{y}[k] = \sum_{i=1}^{2} \hat{a}_i \hat{y}[k-i] + W^T u_{FE}[k]\hat{b} \tag{97}$$

Para minimizar la función de costo, se empleó el algoritmo de Descenso del gradiente. Esto implica calcular la derivada parcial de la función de costo con respecto al vector de pesos, utilizando la regla de la cadena.

$$\frac{\partial V}{\partial W} = \frac{\partial V}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial W} = e[k](-1)\left(u_{FE}[k]\hat{b}\right) = -e[k]\,\hat{b}u_{FE}[k] \tag{98}$$

La actualización de los pesos de la FLANN se realiza en dirección opuesta al gradiente de la función de costo, empleando una tasa de aprendizaje (η) para controlar el tamaño del paso de actualización. Este proceso se lleva a cabo para cada patrón de entrenamiento (Modo patrón). La ecuación utilizada para la actualización de los pesos es la siguiente:

$$W[k+1] = W[k] - \eta \frac{\partial V}{\partial W} \tag{99}$$

Esta actualización se aplica iterativamente hasta que la función de costo converja a un valor mínimo aceptable.

A partir de estas ecuaciones mostradas, se desarrolló un script en MATLAB (ver Anexo N°4) para entrenar la FLANN y estimar los pesos óptimos de la red. Para evitar el sobreajuste se evaluó la precisión del modelo ante datos de validación (ver Figura 81). La bondad del ajuste se evaluó utilizando dos estadísticos: NRMSE y MSE. En la Tabla 17 se presentan las pruebas realizadas para distintas tasas de aprendizaje y la precisión de cada una. Se consideró que la función de costo converja a un valor menor que 0.031.

Tabla 17

Evaluación de la bondad del ajuste para diferentes tasas de aprendizaje en la estimación de la función no lineal de entrada.

N°	Orden del polinomio	η	Épocas	Datos de Entrenamiento		Datos de Validación	
				NRMSE (%)	MSE	NRMSE (%)	MSE
1	3	1	86	90.3615	0.0619	85.2186	0.2196
2	3	5	23	90.3568	0.0619	85.2006	0.2201
3	4	1	100	89.7504	0.07	84.5553	0.2397
4	4	5	27	90.3742	0.617	85.0888	0.2234
5	5	1	100	89.8893	0.0681	85.2181	0.2196
6	5	5	26	90.3725	0.0617	85.3286	0.2163
7	6	5	53	90.3537	0.062	85.144	0.2218

Como se observa en la Tabla 17, la precisión del ajuste de los modelos en el entrenamiento es muy similar, todos tienen una precisión alrededor del 90%. Se seleccionó la prueba 1 por presentar una precisión alta en los datos de entrenamiento y validación. En la Figura 82 se muestra el modelo de la prueba 1, en la que se obtuvo un ajuste con una precisión del 85.2186% en los datos de validación.

Figura 81

Datos de validación para la FLANN: (a) Señal de excitación; (b) Señal de salida del sistema.

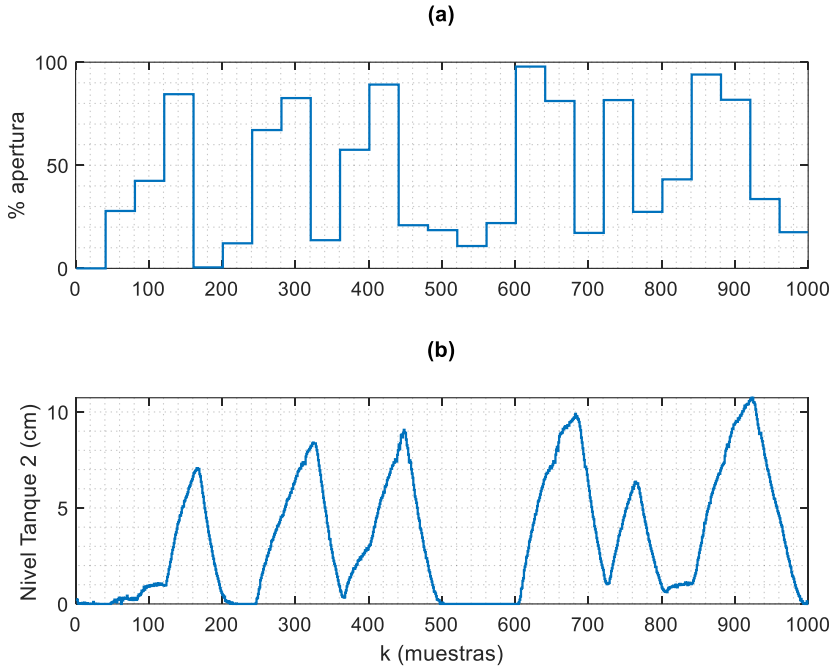
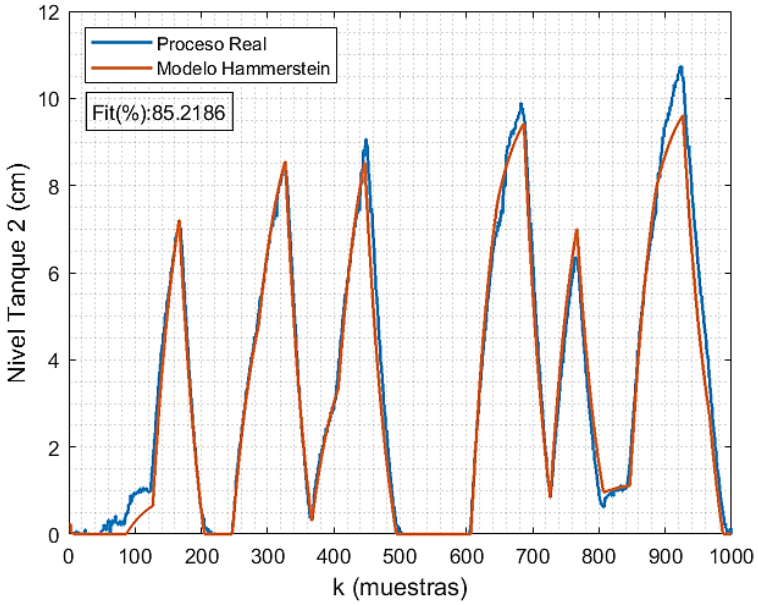


Figura 82

Bondad del ajuste del Modelo Hammerstein identificado ante los datos de validación.



A partir del vector de pesos óptimo W se determinó la función estática no lineal de entrada, la cual se muestra a continuación:

$$v[k] = -38.8975x^3[k] + 4.8377x^2[k] + 97.3844x[k] + 21.7029 \qquad (100)$$

Siendo $x[k]$ la normalización de la entrada $u[k]$ entre los valores de -1 y 1, la relación entre ellas es la siguiente:

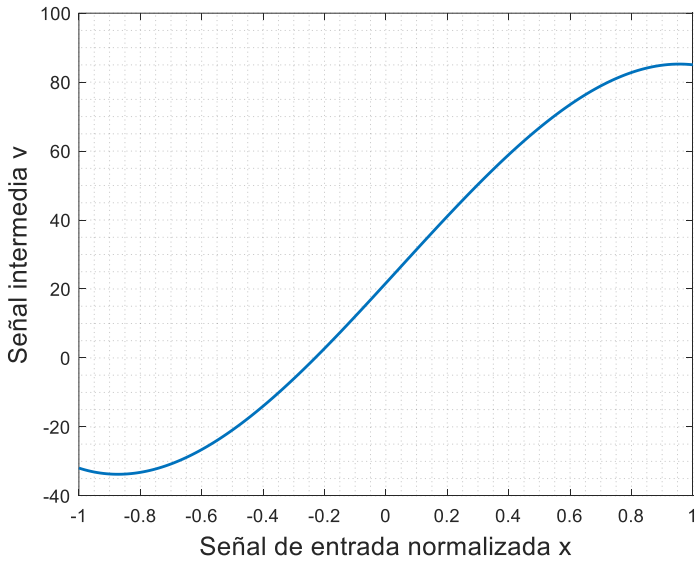
$$x[k] = \frac{2u[k] - u_{mín} - u_{máx}}{u_{máx} - u_{mín}} \qquad (101)$$

Donde $u_{mín} = 0$ y $u_{máx} = 100$ son el valor mínimo y máximo de la señal $u[k]$, respectivamente.

En la Figura 83 se ilustra la no linealidad identificada del Modelo Hammerstein.

Figura 83

No linealidad de entrada del Modelo Hammerstein.

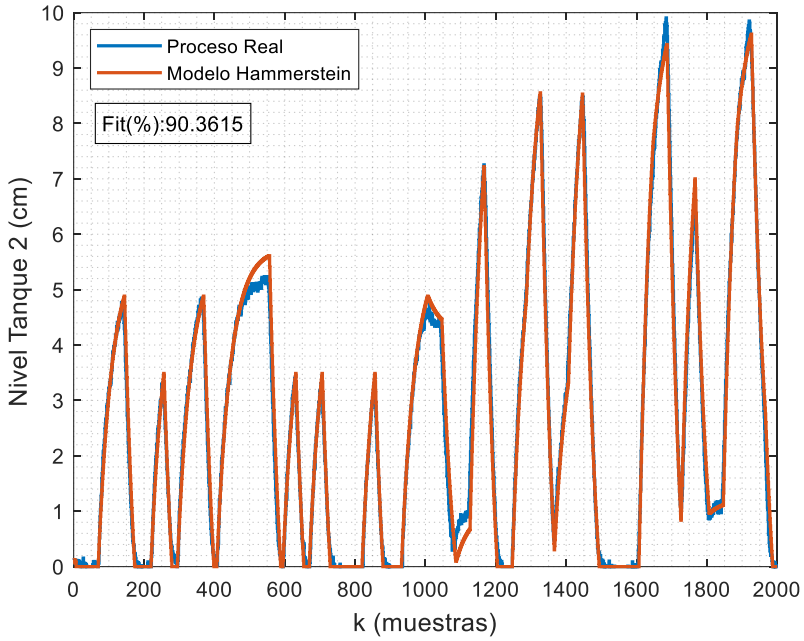


El modelo Hammerstein que representa el sistema de tanques en cascada se obtiene conectando en serie la función estática no lineal de entrada y el subsistema

dinámico lineal identificado. La Figura 84 muestra la comparación entre la salida del proceso real y la salida del Modelo Hammerstein identificado. La precisión del modelo es del 90.3615%.

Figura 84

Gráfico comparativo entre Proceso real y Modelo Hammerstein.



3.3.5 Identificación del Modelo Hammerstein-Wiener

Las ecuaciones que describen el proceso no lineal siguiendo el Modelo Hammerstein-Wiener se presentan a continuación:

$$v[k] = f(u[k]) = p_1 u[k] + p_2 u^2[k] + \cdots + p_m u^m[k] \tag{102}$$

$$z[k] = \sum_{i=1}^{n_a} a_i z[k-i] + \sum_{j=1}^{n_b} b_j v[k-j-K_d] \tag{103}$$

$$z[k] = g^{-1}(y[k]) = q_1 y[k] + q_2 y^2[k] + \cdots + q_r y^r[k] \tag{104}$$

Función estática no lineal de salida

La ecuación (81), que representa la función estática no lineal de salida obtenida en la Identificación del Modelo Wiener, se utilizará para este modelo.

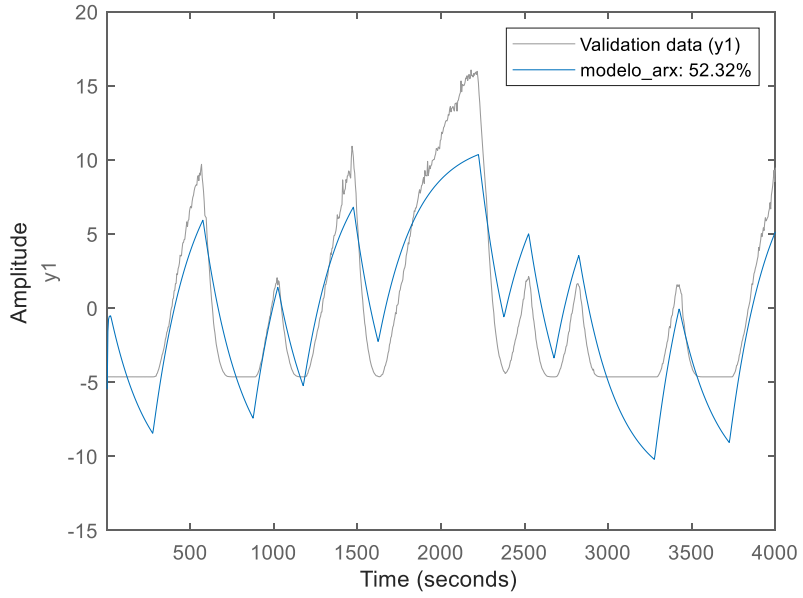
$$z[k] = g^{-1}(y[k]) = 0.2177y^2[k] \tag{81}$$

Subsistema dinámico lineal

El subsistema dinámico lineal se identificará a partir de la señal binaria $u_b(k)$ y la señal intermedia estimada $\hat{z}_b(\mathrm{k})$ utilizando un Modelo ARX con los parámetros $n_a = 2$, $n_b = 1$ y $n_k = 6$. En la Figura 85 se observa la comparación entre la salida del proceso real y la salida del Modelo ARX identificado (ver código MATLAB en el Anexo N°5). La precisión del Modelo ARX es del 52.3228%.

Figura 85

Gráfico comparativo entre $\hat{z}_b(k)$ y Modelo ARX.



Al igual que en el Modelo Hammerstein, antes de la identificación del modelo ARX, se utilizó la función '*detrend*' para sustraer los desplazamientos o tendencias de los datos de entrada y salida.

La función de transferencia en tiempo discreto del Modelo ARX identificado se muestra a continuación

$$G_{arx}(z) = \frac{Z(z)}{V(z)} = \left(\frac{0.00406z^{-1}}{1 - 1.189z^{-1} + 0.2027z^{-2}}\right)z^{-5} \tag{105}$$

El modelo ARX se expresa en ecuaciones en diferencias a partir de la Ecuación (105) de la siguiente manera:

$$\hat{z}[k] = \sum_{i=1}^{2} \hat{a}_i \hat{z}[k-i] + \hat{b}\hat{v}[k-6] \tag{106}$$

Donde:

$$\hat{a} = [1.189, -0.2027]^T \tag{107}$$

$$\hat{b} = 0.00406 \tag{108}$$

Función estática no lineal de entrada

La función estática no lineal de entrada se determinó utilizando una FLANN con una estructura similar a la empleada en la identificación del Modelo Hammerstein (ver Figura 80). La red se entrenó con la señal multipaso u_{ms} de la Figura 70 y la correspondiente salida estimada del subsistema dinámico lineal, mostrada en la Figura 86.

Como se explicó anteriormente, para determinar la función no lineal es necesario obtener los pesos óptimos de la FLANN, lo cual se logra minimizando la siguiente función de costo:

$$V(W) = \frac{1}{2}e^2[k] = \frac{1}{2}(z[k] - \hat{z}[k])^2 \tag{109}$$

Donde:

$$e[k] = z[k] - \hat{z}[k] \tag{110}$$

$$\hat{z}[k] = \sum_{i=1}^{2} \hat{a}_i \hat{z}[k-i] + \hat{b}\hat{v}[k-6] \tag{111}$$

$$\hat{v}[k] = \sum_{l=1}^{N_{ms}} w_l \phi_l(u[k]) \tag{112}$$

y $W = [w_1, w_2, \dots, w_N]^T$ es el vector de pesos de la FLANN. El vector expandido de la FLANN para la entrada $u(k-6)$

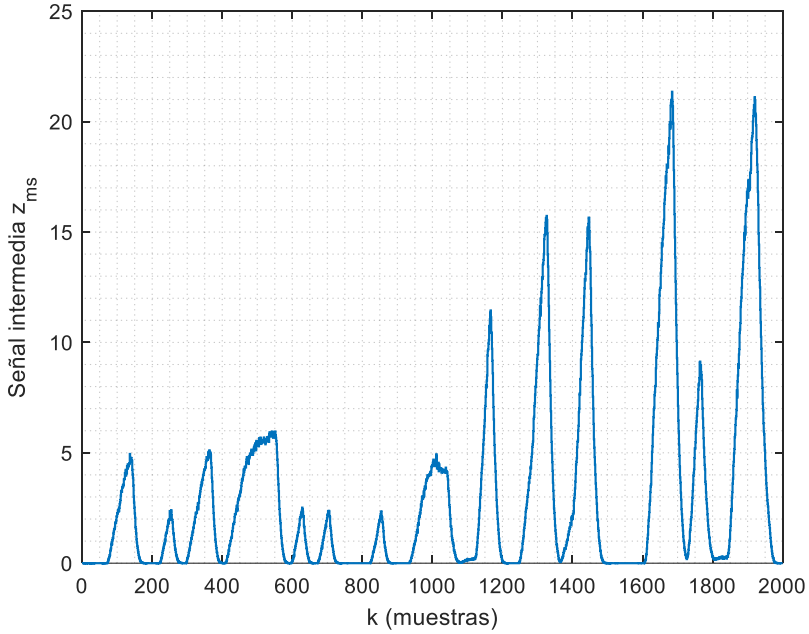
$$u_{FE}[k] = [\phi_1(u[k-6]), \phi_2(u[k-6]), \dots, \phi_N(u[k-6])]^T \tag{113}$$

La Ecuación (111) se reescribe como sigue

$$\hat{z}[k] = \sum_{i=1}^{2} \hat{a}_i \hat{z}[k-i] + W^T u_{FE}[k]\hat{b} \tag{114}$$

Figura 86

Señal de salida estimada z_{ms}.



Para minimizar la función de costo, se empleó el algoritmo de Descenso del gradiente. Esto implica calcular la derivada parcial de la función de costo con respecto al vector de pesos, utilizando la regla de la cadena.

$$\frac{\partial V}{\partial W} = \frac{\partial V}{\partial e}\frac{\partial e}{\partial \hat{z}}\frac{\partial \hat{z}}{\partial W} = e[k](-1)\left(u_{FE}[k]\hat{b}\right) = -e[k]\hat{b}u_{FE}[k] \tag{115}$$

La actualización de los pesos del FLANN se realiza mediante la fórmula especificada en la Ecuación (99), esta actualización se aplica iterativamente hasta que la función de costo converja a un valor mínimo aceptable.

Se desarrolló un script en MATLAB (ver Anexo N°5) a partir de estas ecuaciones para entrenar la FLANN y estimar los pesos óptimos de la red. Para evitar el sobreajuste se evaluó la precisión del modelo ante datos de validación (ver Figura 81). La bondad del ajuste se evaluó utilizando dos estadísticos: NRMSE y MSE. En la Tabla 18 se presentan las pruebas realizadas para distintas tasas de aprendizaje y la precisión de cada una.

Tabla 18

Evaluación de la bondad del ajuste para diferentes tasas de aprendizaje en la estimación de la función no lineal de entrada.

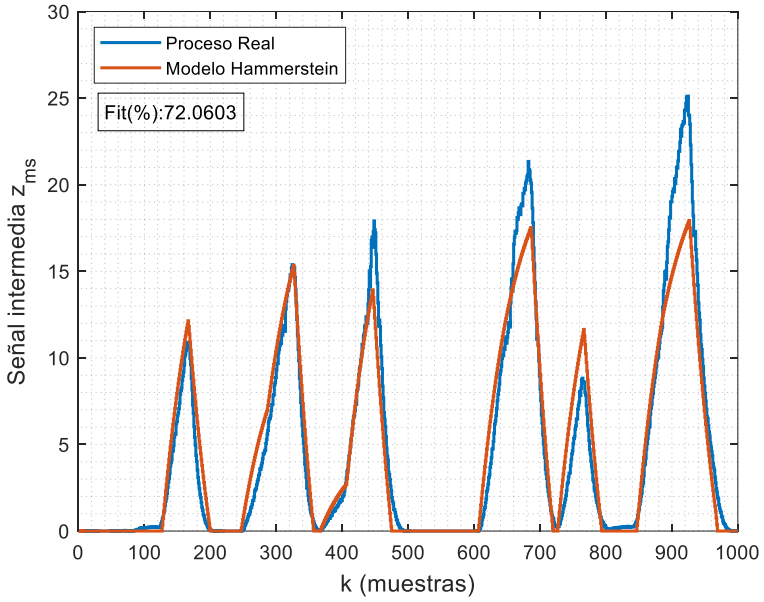
N°	Orden del polinomio	η	Épocas	Datos de Entrenamiento		Datos de Validación	
				NRMSE (%)	MSE	NRMSE (%)	MSE
1	3	1	26	78.001	0.98	69.3517	3.6108
2	3	5	10	77.8059	0.9974	72.0603	3.0008
3	4	1	76	77.7916	0.9987	70.2986	3.3911
4	4	5	23	77.7936	0.9985	71.25	3.1774
5	5	1	86	77.7773	1	71.0286	3.2265
6	5	5	25	77.7777	1	71.6258	3.0948

Como se observa en la Tabla 18, la precisión del ajuste de los modelos en el entrenamiento es muy similar, todos tienen una precisión alrededor del 78%. Se seleccionó la prueba 2 por presentar la precisión más alta en los datos de entrenamiento y validación.

En la Figura 87 se muestra el modelo de la prueba 2, en la que se obtuvo un ajuste con una precisión del 72.0603% en los datos de validación.

Figura 87

Bondad del ajuste de la FLANN ante los datos de validación.



A partir del vector de pesos óptimo W se determinó la función estática no lineal de entrada, la cual se muestra a continuación:

$$v[k] = -96.4611x^3[k] + 18.8609x^2[k] + 160.3935x[k] - 5.8998 \qquad (116)$$

Siendo $x(k)$ la normalización de la entrada $u(k)$ entre los valores de -1 y 1, la relación entre ellas es la siguiente:

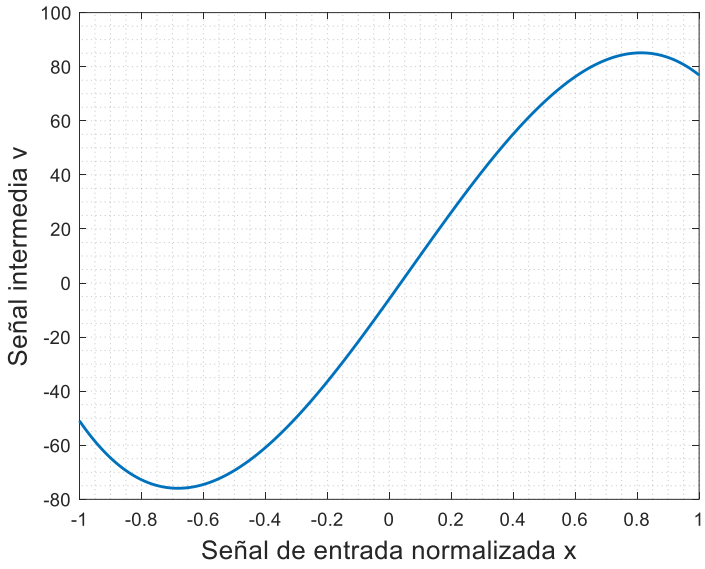
$$x[k] = \frac{2u[k] - u_{mín} - u_{máx}}{u_{máx} - u_{mín}} \qquad (117)$$

Donde $u_{mín} = 0$ y $u_{máx} = 100$ son el valor mínimo y máximo de la señal $u(k)$, respectivamente.

En la Figura 88 se ilustra la no linealidad identificada del Modelo Hammerstein-Wiener.

Figura 88

No linealidad de entrada del Modelo Hammerstein-Wiener.



El modelo Hammerstein-Wiener que representa el sistema de tanques en cascada se obtiene conectando en serie la función estática no lineal de entrada, el subsistema dinámico lineal identificado y la función estática no lineal de salida. En la Figura 89 se observa la comparación entre la salida del proceso real y la salida del Modelo Hammerstein-Wiener identificado. La precisión del modelo es del 74.4701%.

Finalmente, después de identificar tres posibles modelos no lineales del sistema de tanques en cascada (ver Tabla 19), se seleccionó el Modelo Hammerstein. Este modelo fue el que mejor capturó la dinámica del sistema y obtuvo el porcentaje de precisión más alto en comparación con los datos reales. En base a este modelo, se diseñará el controlador adaptativo no lineal, cuyo diseño se detallará en la siguiente sección.

Figura 89

Gráfico comparativo entre Proceso real y Modelo Hammerstein-Wiener.

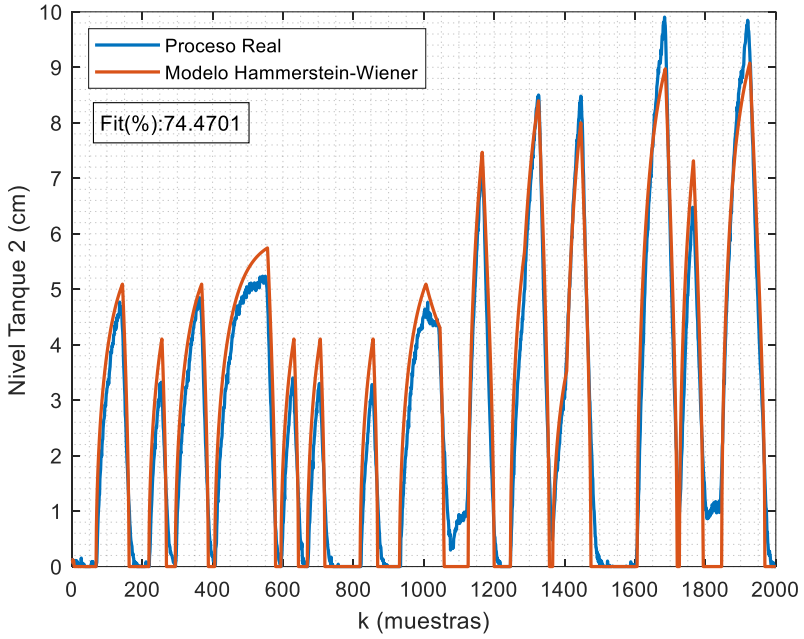


Tabla 19

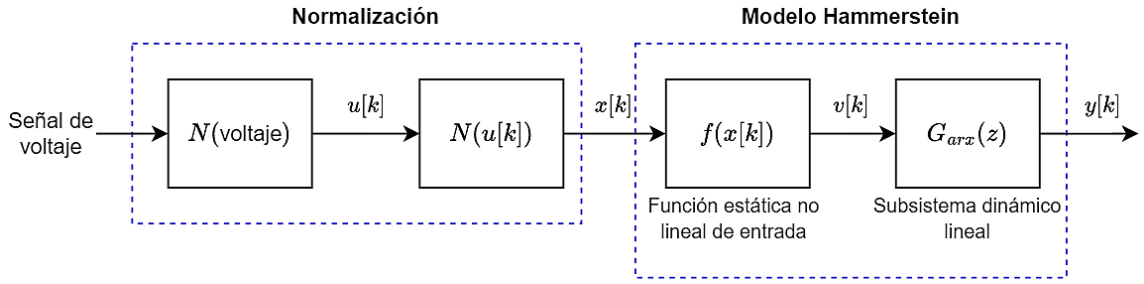
Comparativa de la bondad del ajuste entre los Modelos No Lineales.

Modelo No Lineal	Precisión
Modelo Wiener	17.1324%
Modelo Hammerstein	90.3615%
Modelo Hammerstein-Wiener	74.4701%

En la Figura 90 se muestra el diagrama de bloques completo del modelo que representa el sistema de tanques en cascada. Primero, la señal de control de voltaje es convertida a porcentaje de apertura de la válvula proporcional mediante la relación expresada en la Ecuación (64). Luego, el porcentaje de apertura es normalizado al rango de $[-1; 1]$ (ver Ecuación (117)). Finalmente, ese valor ingresa al Modelo Hammerstein para obtener la señal de salida estimada.

Figura 90

Diagrama de bloques completo del modelo de la planta.



3.4 Diseño del Controlador Adaptativo No Lineal

En esta sección se presenta el diseño del controlador adaptativo no lineal para el sistema de tanques en cascada. Utilizando un modelo Hammerstein, que incluye una función no lineal de entrada y un subsistema lineal de segundo orden con retardo, se desarrollaron controladores MRAC mediante la regla MIT y el método de Lyapunov. Además, se implementó un controlador PID sintonizado mediante técnicas MRAC y un predictor de Smith para optimizar el desempeño del sistema, garantizando un mínimo sobreimpulso, un tiempo de establecimiento reducido y un error de estado estacionario nulo.

3.4.1 Diseño de Controlador MRAC

Para el diseño del controlador MRAC se considera un sistema de segundo orden representado por la siguiente ecuación en diferencias

$$y_p[k] = -a_1 y_p[k-1] - a_2 y_p[k-2] + b_1 u[k-1] \tag{118}$$

Donde u y y_p son la entrada y salida del sistema que se quiere controlar, y a_1, a_2 y b_1 son parámetros que determinan la dinámica del sistema. Se supuso que estos parámetros son desconocidas durante el procedimiento de diseño del controlador.

El modelo de referencia se define mediante la siguiente ecuación

$$y_m[k] = -a_{m1} y_m[k-1] - a_{m2} y_m[k-2] + b_{m1} u[k-1] \tag{119}$$

Donde y_m es la salida del modelo de referencia, y a_{m1}, a_{m2} y b_{m1} son parámetros del modelo de referencia y se asume que son conocidos.

La estructura de la acción de control viene dada por la siguiente ecuación

$$u[k-1] = \theta_1 u_c[k-1] - \theta_2 y_p[k-1] \tag{120}$$

donde $\theta_1(k)$ y $\theta_2(k)$ son los parámetros de control adaptativo y $u_c(k)$ es la entrada de referencia.

Sustituyendo la Ecuación (120) en (118), se obtiene

$$y_p[k] = -(a_1 + b_1\theta_2)y_p[k-1] - a_2 y_p[k-2] + b_1\theta_1 u_c[k-1] \tag{121}$$

Este modelo es un sistema de control en bucle cerrado. Para que los sistemas dinámicos de las Ecuaciones (119) y (121) sean idénticos, los parámetros $\theta_1[k]$ y $\theta_2[k]$ deben tomar los siguientes valores constantes

$$\theta_1 = \theta_1^* = \frac{b_{m1}}{b_1} \tag{122}$$

$$\theta_2 = \theta_2^* = \frac{a_{m1} - a_1}{b_1} \tag{123}$$

Se aplicaron dos técnicas de ajuste de los parámetros del MRAC, estas se detallan a continuación.

Regla MIT

En términos generales, la regla de control adaptativo del MIT pretende minimizar la siguiente función de coste

$$J = \frac{1}{2}e^2 \tag{124}$$

Donde e es el error de control definido del siguiente modo

$$e[k] = y_p[k] - y_m[k] \tag{125}$$

Para disminuir el valor de J es razonable ajustar los parámetros $\theta_1(k)$ y $\theta_2(k)$ en la dirección del gradiente negativo de J. Usando esta idea, se obtiene

$$\Delta\theta_1[k] = -\gamma \frac{\partial J}{\partial \theta_1} = -\gamma e \frac{\partial e}{\partial \theta_1} \tag{126}$$

$$\Delta\theta_2[k] = -\gamma \frac{\partial J}{\partial \theta_2} = -\gamma e \frac{\partial e}{\partial \theta_2} \tag{127}$$

Para calcular estas derivadas parciales, se necesita obtener $Y_p(z)$. Aplicando la transformada z a la Ecuación (121) se obtiene

$$Y_p(z) = -(a_1 + b_1\theta_2)z^{-1}Y_p(z) - a_2 z^{-2}Y_p(z) + b_1\theta_1 z^{-1}U_c(z)$$

$$Y_p(z) = \frac{b_1\theta_1 z^{-1}}{1 + (a_1 + b_1\theta_2)z^{-1} + a_2 z^{-2}} U_c(z) \tag{128}$$

Aplicando la transformada z en la Ecuación (125) y reemplazando la Ecuación (128), se obtiene

$$E(z) = \frac{b_1\theta_1 z^{-1}}{1 + (a_1 + b_1\theta_2)z^{-1} + a_2 z^{-2}} U_c(z) - Y_m(z) \tag{129}$$

Se calculan las derivadas parciales de las Ecuaciones (126) y (127)

$$\frac{\partial e}{\partial \theta_1} = \frac{\partial E(z)}{\partial \theta_1} = \frac{b_1 z^{-1}}{1 + (a_1 + b_1\theta_2)z^{-1} + a_2 z^{-2}} U_c(z) = \frac{b_1 z^{-1}}{1 + a_{m1} z^{-1} + a_{m2} z^{-2}} U_c(z) \tag{130}$$

$$\frac{\partial e}{\partial \theta_2} = \frac{\partial E(z)}{\partial \theta_2} = -\frac{b_1 z^{-1}}{1 + (a_1 + b_1\theta_2)z^{-1} + a_2 z^{-2}} Y_p(z) = -\frac{b_1 z^{-1}}{1 + a_{m1} z^{-1} + a_{m2} z^{-2}} Y_p(z) \tag{131}$$

Finalmente, sustituyendo estas derivadas parciales en las Ecuaciones (126) y (127) se obtienen las expresiones finales que describen la actualización de los parámetros

$$\Delta\theta_1[k] = -\gamma e \frac{b_1 z^{-1}}{1 + a_{m1} z^{-1} + a_{m2} z^{-2}} U_c(z) \tag{132}$$

$$\Delta\theta_2[k] = +\gamma e \frac{b_1 z^{-1}}{1 + a_{m1} z^{-1} + a_{m2} z^{-2}} Y_p(z) \tag{133}$$

Método de Lyapunov

La función de Lyapunov propuesta para analizar la estabilidad del sistema es

$$V[k] = \frac{1}{2} e[k]^2 + \frac{b_1}{2\gamma} (\theta_1[k] - \theta_1^*)^2 + \frac{b_1}{2\gamma} (\theta_2[k] - \theta_2^*)^2 \tag{134}$$

Esta función es semidefinida positiva ($V[k] > 0$) y es cero cuando el error es cero ($V[0] = 0$). Para la estabilidad según las condiciones especificadas en la Ecuación (53), la variación de $V[k]$ ($\Delta V[k]$), debe ser semidefinida negativa.

La variación de la función de Lyapunov es como sigue

$$\Delta V[k] = e[k]\Delta e[k] + \frac{b_1}{\gamma} (\theta_1[k] - \theta_1^*)\Delta\theta_1[k] + \frac{b_1}{\gamma} (\theta_2[k] - \theta_2^*)\Delta\theta_2[k] \tag{135}$$

Reemplazando las Ecuaciones (122) y (123) en (135), se obtiene

$$\Delta V[k] = e[k]\Delta e[k] + \frac{1}{\gamma} (b_1\theta_1[k] - b_{m1})\Delta\theta_1[k] + \frac{1}{\gamma} (b_1\theta_2[k] - a_{m1} + a_1)\Delta\theta_2[k] \tag{136}$$

Para determinar si $\Delta V[k]$ es semidefinida negativa, se calcula la variación del error $\Delta e[k]$, la cual es como sigue

$$\Delta e[k] = e[k+1] - e[k] \tag{137}$$

Reemplazando la Ecuación (125) en (137), se obtiene

$$\Delta e[k] = \Delta y_p[k] - \Delta y_m[k] \tag{138}$$

A partir de las Ecuaciones (118), (119) y (120), se obtiene, respectivamente

$$\Delta y_p[k] = y_p[k+1] - y_p[k] = -a_1 \Delta y_p[k-1] - a_2 \Delta y_p[k-2] + b_1 \Delta u[k-1] \quad (139)$$

$$\Delta y_m[k] = y_m[k+1] - y_m[k] = -a_{m1} \Delta y_m[k-1] - a_{m2} \Delta y_m[k-2] + b_{m1} \Delta u_c[k-1] \quad (140)$$

$$\Delta u[k-1] = u[k] - u[k-1] = \theta_1 \Delta u_c[k-1] - \theta_2 \Delta y_p[k-1] \quad (141)$$

Reemplazando la Ecuación (141) en (139), se obtiene

$$\Delta y_p[k] = -(b_1\theta_2 + a_1)\Delta y_p[k-1] - a_2 \Delta y_p[k-2] + b_1\theta_1 \Delta u_c[k-1] \quad (142)$$

Se calcula $\Delta e[k]$ reemplazando las Ecuaciones (142) y (140) en (138), además, se añaden y sustraen los términos $a_{m1}\Delta y_p[k-1]$ y $a_{m2}\Delta y_p[k-2]$, luego de agrupar de manera conveniente, se obtiene

$$\begin{aligned}\Delta e[k] = -a_{m1}\Delta e[k-1] - (b_1\theta_2 - a_{m1} + a_1)\Delta y_p[k-1] + (b_1\theta_1 - b_{m1})\Delta u_c[k-1] \\ -(a_2 - a_{2m})\Delta y_p[k-2] - a_{m2}\Delta e[k-2]\end{aligned} \quad (143)$$

Esta expresión se reemplaza en la Ecuación (136), al agrupar convenientemente se obtiene lo siguiente

$$\begin{aligned}\Delta V[k] = -\big(a_{m1}\Delta e[k-1] + (a_2 - a_{2m})\Delta y_p[k-2] + a_{m2}\Delta e[k-2]\big)e[k] \\ +\frac{1}{\gamma}(b_1\theta_1 - b_{m1})(\Delta\theta_1[k] + \gamma e[k]\Delta u_c[k-1]) \\ +\frac{1}{\gamma}(b_1\theta_2 - a_{m1} + a_1)\big(\Delta\theta_2[k] - \gamma e[k]\Delta y_p[k-1]\big)\end{aligned} \quad (144)$$

Para que $\Delta V[k]$ sea semidefinida negativa, se debe cumplir lo siguiente

$$\Delta\theta_1[k] = -\gamma e[k]\Delta u_c[k-1] \quad (145)$$

$$\Delta\theta_2[k] = \gamma e[k]\Delta y_p[k-1] \quad (146)$$

Estas expresiones finales describen la actualización de los parámetros $\theta_1[k]$ y $\theta_2[k]$.

3.4.2 Diseño de Controlador PID mediante técnicas MRAC

Para el diseño del controlador PID mediante técnicas MRAC se considera un sistema de segundo orden representado por la siguiente función de transferencia discreta

$$G_p = \frac{Y_p(z)}{U(z)} = \frac{b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{147}$$

Por otro lado, el controlador PID discreto se muestra a continuación

$$G_c = K_p + K_i \frac{T_s}{1 - z^{-1}} + K_d \frac{1 - z^{-1}}{T_s} \tag{148}$$

A partir del controlador, se obtiene la ley de control PID discreta, la cual se muestra a continuación

$$U(z) = K_p \varepsilon(z) + K_i \frac{T_s}{1 - z^{-1}} \varepsilon(z) + K_d \frac{1 - z^{-1}}{T_s} \varepsilon(z) \tag{149}$$

Donde $\varepsilon(z)$ es el error de realimentación del lazo de control. La función de transferencia en lazo cerrado se obtiene como sigue

$$\frac{Y_p(z)}{U_c(z)} = \frac{G_p G_c}{1 + G_p G_c}$$

$$\frac{Y_p(z)}{U_c(z)} = \frac{\sigma_1 z^{-1} + \sigma_2 z^{-2} + \sigma_3 z^{-3}}{1 + (\sigma_1 + a_1 - 1) z^{-1} + (\sigma_2 + a_2 - a_1) z^{-2} + (\sigma_3 - a_2) z^{-3}} \tag{150}$$

$$\sigma_1 = b_1 K_i T_s^2 + b_1 K_p T_s + b_1 K_d$$

$$\sigma_2 = -b_1 K_p T_s - 2 b_1 K_d$$

$$\sigma_3 = b_1 K_d$$

A partir de la Ecuación (150) y del rendimiento requerido del sistema, se obtiene el siguiente modelo de referencia

$$G_m = \frac{b_{m1} z^{-1} + b_{m2} z^{-2} + b_{m3} z^{-3}}{1 + a_{m1} z^{-1} + a_{m2} z^{-2} + a_{m3} z^{-3}} \tag{151}$$

El objetivo de este diseño es que el sistema en lazo cerrado (Ecuación (150)) y el modelo de referencia (Ecuación (151)) sean idénticos. Para lograr esto, se aplicaron dos técnicas de ajuste de los parámetros del MRAC, estas se detallan a continuación.

Regla MIT

Se aplica las reglas del gradiente MIT para determinar el valor de los parámetros del controlador PID de la siguiente manera

$$\Delta K_p[k] = -\gamma_p \frac{\partial J}{\partial K_p} = -\gamma_p E(z) \frac{\partial E(z)}{\partial K_p} = -\gamma_p E(z) \frac{\partial Y_p(z)}{\partial K_p} \tag{152}$$

$$\Delta K_i[k] = -\gamma_i \frac{\partial J}{\partial K_i} = -\gamma_i E(z) \frac{\partial E(z)}{\partial K_i} = -\gamma_i E(z) \frac{\partial Y_p(z)}{\partial K_i} \tag{153}$$

$$\Delta K_d[k] = -\gamma_d \frac{\partial J}{\partial K_d} = -\gamma_d E(z) \frac{\partial E(z)}{\partial K_d} = -\gamma_d E(z) \frac{\partial Y_p(z)}{\partial K_d} \tag{154}$$

Las derivadas parciales de la salida de la planta respecto a los parámetros del controlador PID se calculan a continuación

$$\frac{\partial Y_p(z)}{\partial K_p} = \frac{b_1 T_s z^{-1}(1 - z^{-1})}{1 + (\sigma_1 + a_1 - 1)z^{-1} + (\sigma_2 + a_2 - a_1)z^{-2} + (\sigma_3 - a_2)z^{-3}} \Big(U_c(z) - Y_p(z)\Big) \tag{155}$$

$$\frac{\partial Y_p(z)}{\partial K_i} = \frac{b_1 T_s^2 z^{-1}}{1 + (\sigma_1 + a_1 - 1)z^{-1} + (\sigma_2 + a_2 - a_1)z^{-2} + (\sigma_3 - a_2)z^{-3}} \Big(U_c(z) - Y_p(z)\Big) \tag{156}$$

$$\frac{\partial Y_p(z)}{\partial K_d} = \frac{b_1 z^{-1}(1 - z^{-1})^2}{1 + (\sigma_1 + a_1 - 1)z^{-1} + (\sigma_2 + a_2 - a_1)z^{-2} + (\sigma_3 - a_2)z^{-3}} \Big(U_c(z) - Y_p(z)\Big) \tag{157}$$

Debido a que se busca que la Ecuación (150) y (151) sean idénticas, el denominador de las derivadas parciales calculadas debe ser igual al del modelo de referencia. Finalmente, se obtiene la actualización de los parámetros $K_p[k]$, $K_i[k]$ y $K_d[k]$

$$\Delta K_p[k] = -\gamma_p E(z) \frac{b_1 T_s z^{-1}(1 - z^{-1})}{1 + a_{m1} z^{-1} + a_{m2} z^{-2} + a_{m3} z^{-3}} \Big(U_c(z) - Y_p(z) \Big) \tag{158}$$

$$\Delta K_i[k] = -\gamma_i E(z) \frac{b_1 T_s^2 z^{-1}}{1 + a_{m1} z^{-1} + a_{m2} z^{-2} + a_{m3} z^{-3}} \Big(U_c(z) - Y_p(z) \Big) \tag{159}$$

$$\Delta K_d[k] = -\gamma_d E(z) \frac{b_1 z^{-1}(1 - z^{-1})^2}{1 + a_{m1} z^{-1} + a_{m2} z^{-2} + a_{m3} z^{-3}} \Big(U_c(z) - Y_p(z) \Big) \tag{160}$$

3.4.3 Modelo de Referencia

La elección del modelo de referencia debe satisfacer dos requisitos clave. Primero, debe reflejar las características deseadas del sistema, tales como tiempo de subida, tiempo de asentamiento, sobreimpulso máximo, entre otros. Segundo, el comportamiento ideal del modelo debe ser alcanzable por el sistema de control. Esto implica que no se debe seleccionar un modelo de referencia sin considerar si el controlador o la planta son capaces lograr dicho comportamiento.

Estas restricciones imponen ciertas limitaciones en la estructura del modelo de referencia, como su orden y grado. La selección del modelo de referencia debe cumplir con los siguientes criterios:

- Tener el mismo orden y grado que el polinomio de la planta a controlar.
- Ser estable.
- Ser de fase mínima.

Para este trabajo de investigación se plantea la implementación de un controlador MRAC para un sistema de segundo orden. Por lo tanto, se selecciona una función de transferencia de segundo orden estándar como modelo de referencia, tal como se muestra en la Ecuación (161).

$$G_m(s) = \frac{K w_n^2}{s^2 + 2\zeta w_n s + w_n^2} \tag{161}$$

Donde K es la ganancia, ζ es el factor de amortiguamiento y w_n es la frecuencia natural. El modelo elegido cumple con las siguientes características:

- Amplitud máxima $K = 1$.
- Sistema sub-amortiguado con $\zeta = 0.8$.
- Tiempo de asentamiento $t_s = 150$ segundos.

Con estos parámetros se calcula la frecuencia natural w_n del sistema, utilizando la siguiente expresión

$$w_n = \frac{3.912}{\zeta t_s} = 0.0307 \tag{162}$$

Sustituyendo el valor de los parámetros ζ, w_n y K en la Ecuación (161), se obtiene el modelo de referencia en tiempo continuo

$$G_m(s) = \frac{0.0009414}{s^2 + 0.05216s + 0.0009414} \tag{163}$$

Se procede a convertir el modelo de referencia de tiempo continuo a discreto mediante el método de discretización ZOH (Zero-Order Hold) y se obtiene la siguiente función de transferencia

$$G_m(z) = \frac{0.007025z^{-1} + 0.006553z^{-2}}{1 - 1.798z^{-1} + 0.8117z^{-2}} \tag{164}$$

En la Figura 91 se presenta la respuesta al escalón del modelo de referencia discreto (Ecuación (164)) y en la Tabla 20 se muestra las características de la respuesta escalón.

Figura 91

Respuesta escalón del Modelo de referencia discreto $G_m(z)$.

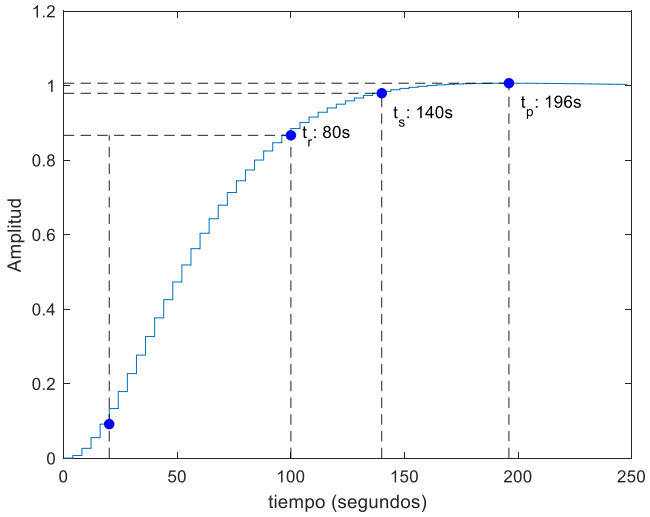


Tabla 20

Características de la respuesta escalón del Modelo de referencia.

Parámetro	Valor
Sobreimpulso máx. M_p	0.627%
Tiempo de pico t_p	196 s
Tiempo de asentamiento t_s	140 s
Tiempo de subida t_r	80 s

3.4.4 Diseño de Predictor de Smith

El empleo del predictor de Smith en este diseño es fundamental debido al retraso presente en la planta, que dificulta el desempeño óptimo del controlador MRAC. Los controladores MRAC, aunque efectivos en sistemas sin retardo, tienen limitaciones significativas cuando se enfrentan a retardos temporales en el sistema a controlar. Para abordar esta limitación, se implementa un predictor de Smith que compensa el retardo del sistema, mejorando así la precisión y la estabilidad del control.

El modelo del sistema sin retardo, $G_n(z)$, se deriva del modelo Hammerstein identificado, eliminando el componente de retardo del subsistema dinámico lineal

$$G_n(z) = \frac{Y(z)}{V(z)} = \frac{0.002851z^{-1}}{1 - 1.163z^{-1} + 0.1834z^{-2}} \quad (165)$$

El predictor de Smith emplea este modelo sin retardo $G_n(z)$ para anticipar la respuesta del sistema, compensando así el retardo real del sistema y permitiendo al controlador adaptativo no lineal actuar de manera más efectiva.

Capítulo IV. Análisis y Discusión de los Resultados

En este capítulo se presentan los resultados obtenidos de las simulaciones realizadas en Simulink de MATLAB para los controladores diseñados. Se muestran gráficos de la respuesta temporal, la acción de control y el error para cada controlador. Posteriormente, se comparan los controladores basados en los criterios de sobreimpulso, tiempo de pico y tiempo de establecimiento, concluyendo con una tabla comparativa que permite seleccionar el controlador óptimo para el sistema.

4.1 Controlador MRAC con Regla MIT

Se presenta los resultados del Controlador MRAC diseñado utilizando la Regla MIT. La Figura 92, Figura 93, Figura 94 y Figura 95 presentan la respuesta temporal, la acción de control, el error y la evolución de los parámetros del controlador , respectivamente, para diferentes coeficientes de adaptación del controlador MRAC con Regla MIT. En la Tabla 21 se muestra una comparativa de las características de las respuestas temporales para los tres coeficientes de adaptación.

Tabla 21

Características de la Respuesta temporal de Controladores MRAC con Regla MIT.

Coeficientes de adaptación	Sobreimpulso (%)	Tiempo de pico (s)	Tiempo de establecimiento (s)
$\boldsymbol{\gamma_1 = 0.001}$	9.8382	264	396
$\boldsymbol{\gamma_2 = 0.002}$	10.7933	240	344
$\boldsymbol{\gamma_3 = 0.003}$	11.7871	232	332

Figura 92

Respuesta temporal para diferentes coeficientes de adaptación del controlador MRAC con regla MIT.

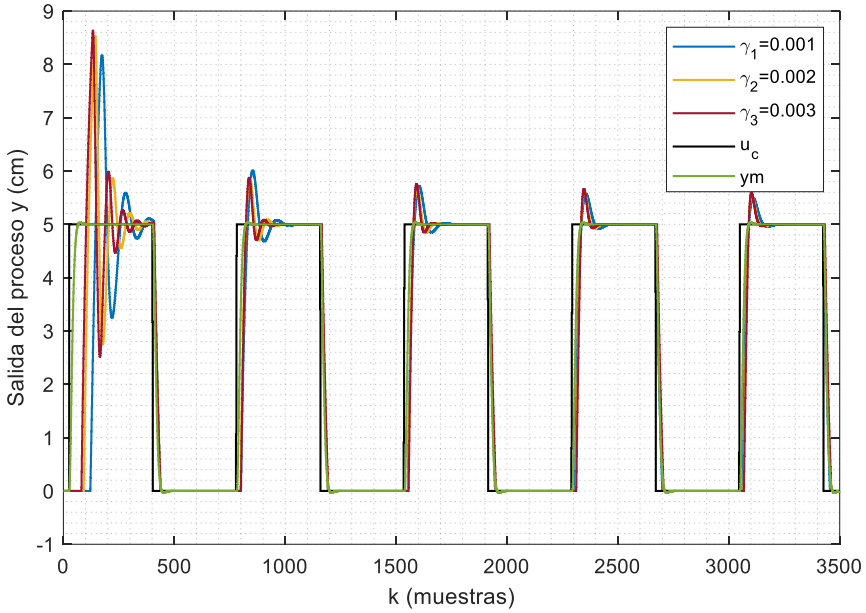


Figura 93

Acción de control para diferentes coeficientes de adaptación del controlador MRAC con regla MIT.

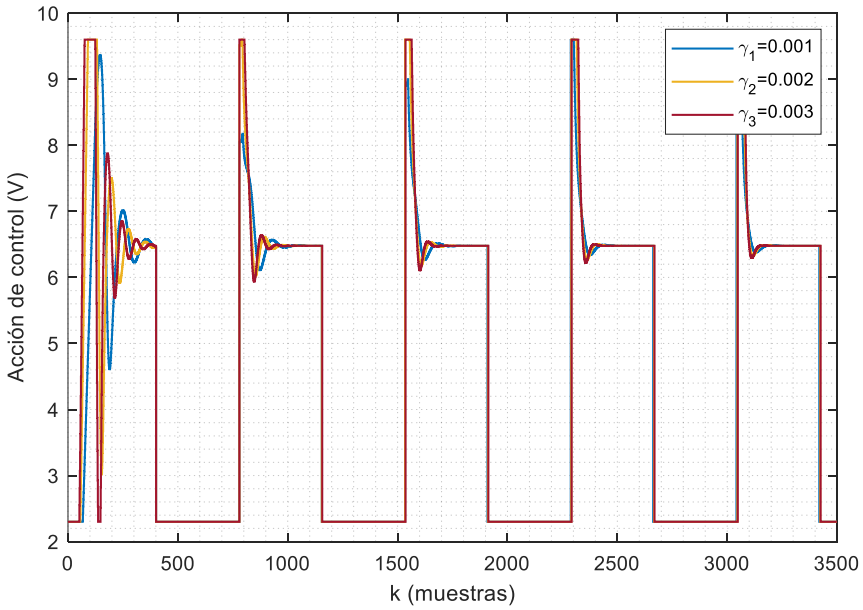


Figura 94

Error para diferentes coeficientes de adaptación del controlador MRAC con regla MIT.

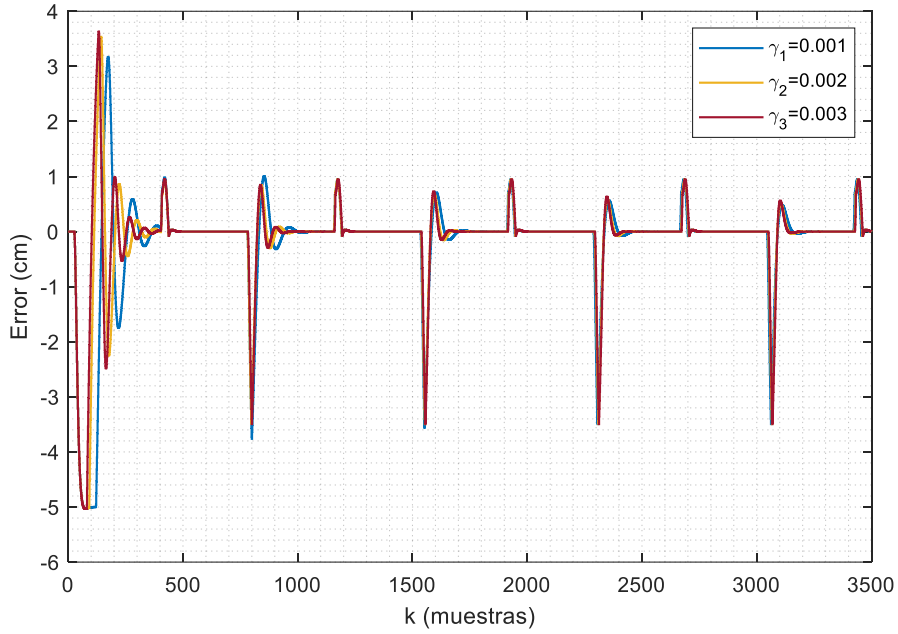
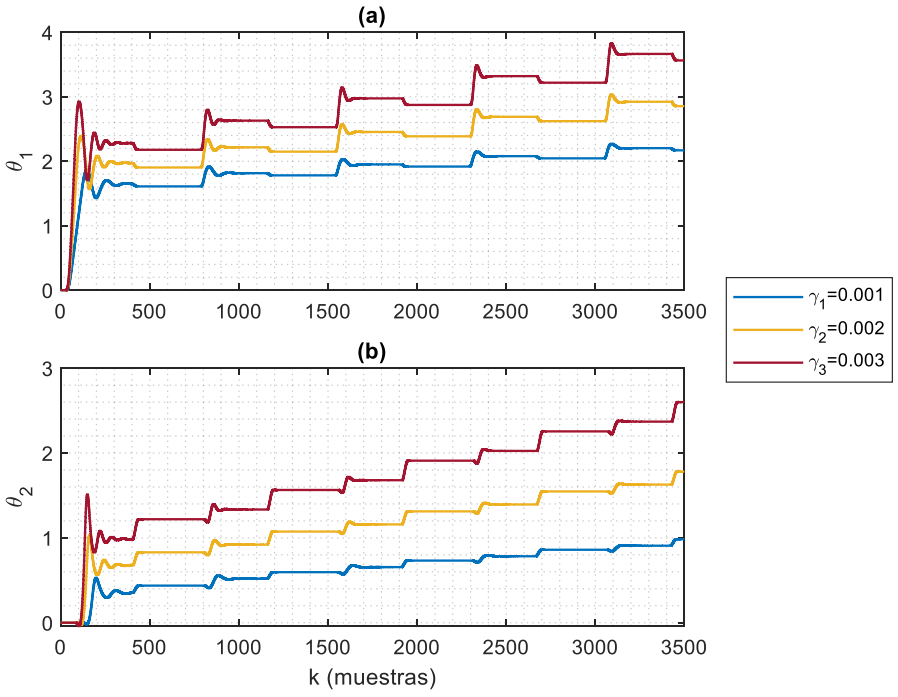


Figura 95.

Ajuste en el tiempo de los parámetros para diferentes coeficientes de adaptación del controlador MRAC con regla MIT: (a) Parámetro θ_1; (b) Parámetro θ_2.



4.2 Controlador MRAC con Método de Lyapunov

Se presenta los resultados del Controlador MRAC diseñado utilizando el Método de Lyapunov. La Figura 96(a), Figura 96(b), Figura 97 y Figura 98 muestran la respuesta temporal, el error, la acción de control y la evolución de los parámetros del controlador, respectivamente, para diferentes coeficientes de adaptación de controlador MRAC con Método de Lyapunov. En la Tabla 22 se muestra una comparativa de las características de las respuestas temporales para los tres coeficientes de adaptación.

Figura 96

Comportamiento del controlador MRAC con método de Lyapunov para diferentes coeficientes de adaptación: (a) Respuesta temporal; (b) Error.

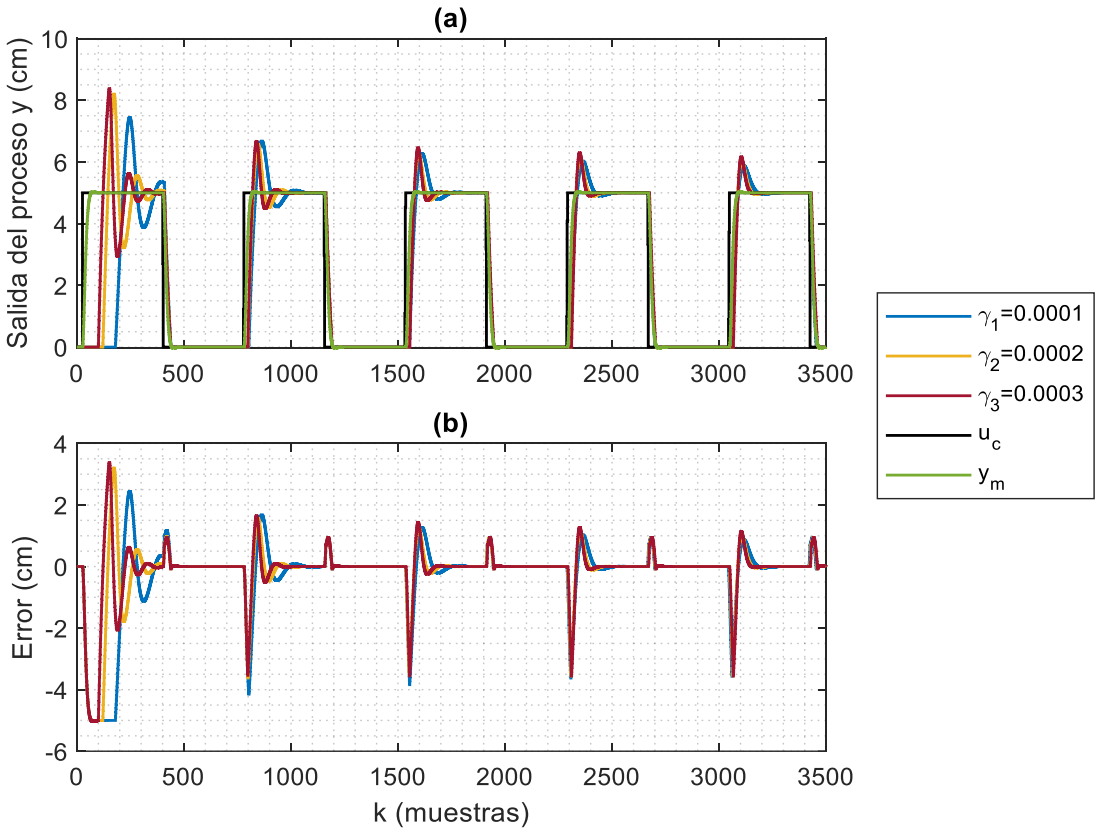


Figura 97

Acción de control para diferentes coeficientes de adaptación del controlador MRAC con método de Lyapunov.

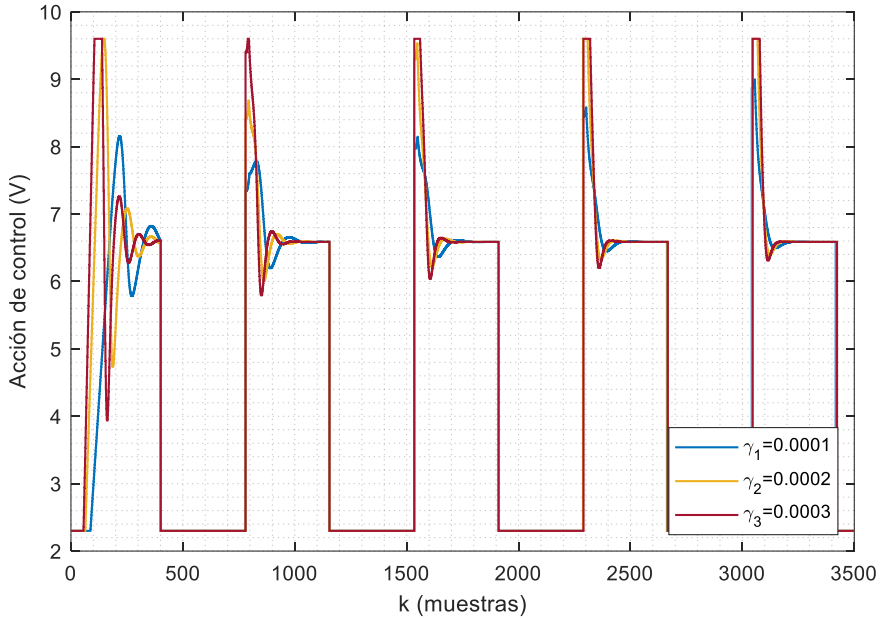


Figura 98

Ajuste en el tiempo de los parámetros para diferentes coeficientes de adaptación del controlador MRAC con método de Lyapunov: (a) Parámetro θ_1; (b) Parámetro θ_2.

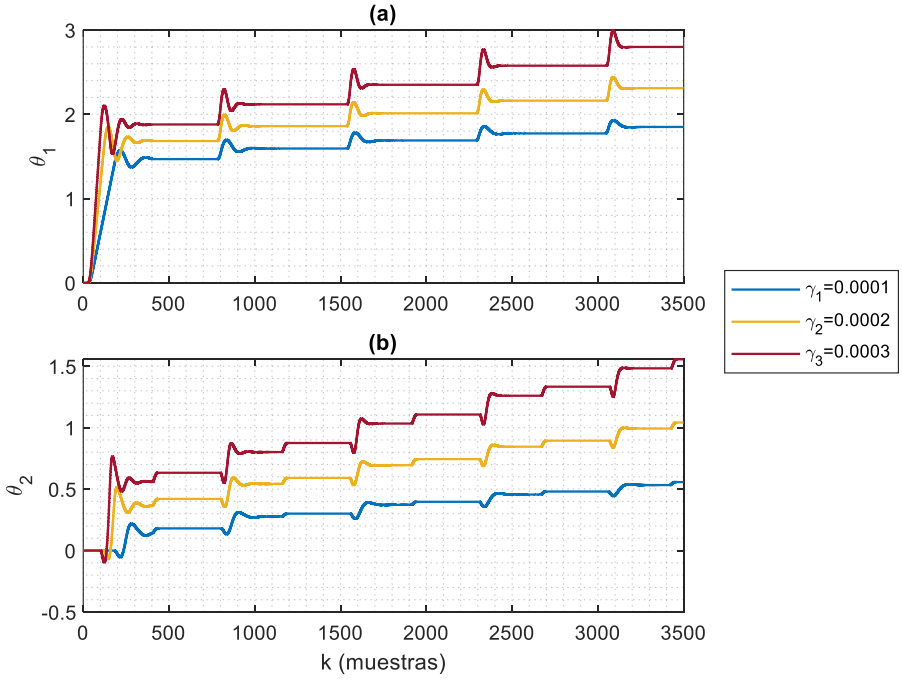


Tabla 22

Características de la Respuesta temporal de Controladores MRAC con Método de Lyapunov.

Coeficientes de adaptación	Sobreimpulso (%)	Tiempo de pico (s)	Tiempo de establecimiento (s)
$\gamma_1 = 0.0001$	17.5089	292	500
$\gamma_2 = 0.0002$	20.4714	252	408
$\gamma_3 = 0.0003$	23.2937	244	384

4.3 Controlador PID sintonizado con técnicas MRAC

Se presenta los resultados del Controlador PID sintonizado con técnicas MRAC. La Figura 99, Figura 100, Figura 101 y Figura 102 ilustran la respuesta temporal, la acción de control, el error y la evolución de los parámetros del controlador, respectivamente, para diferentes coeficientes de adaptación del controlador PID sintonizado con técnicas MRAC.

Figura 99

Respuesta temporal para diferentes coeficientes de adaptación del controlador PID sintonizado con técnicas MRAC.

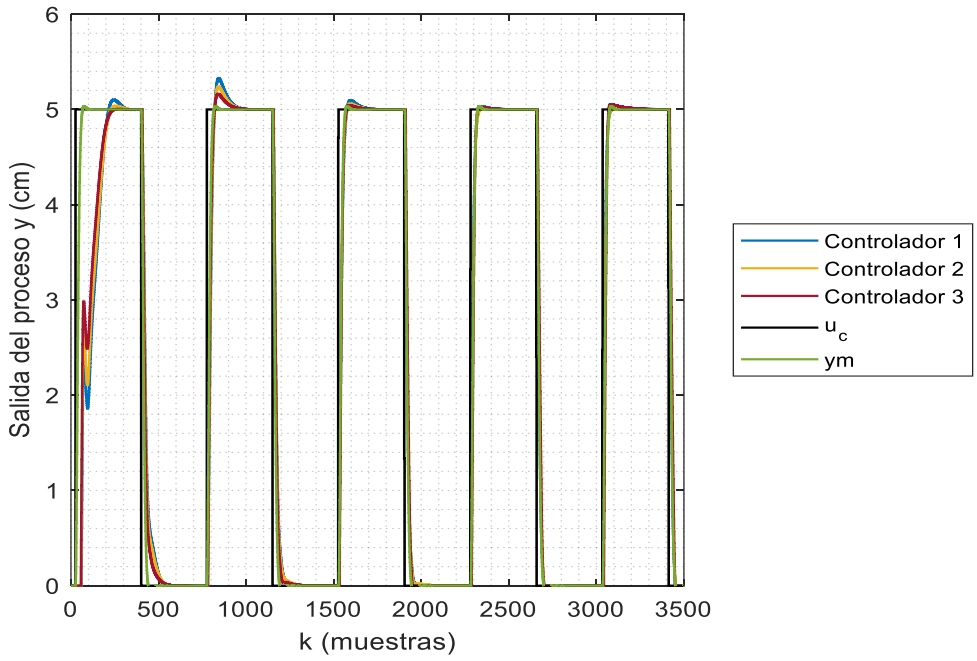


Figura 100

Acción de control para diferentes coeficientes de adaptación del controlador PID sintonizado con técnicas MRAC.

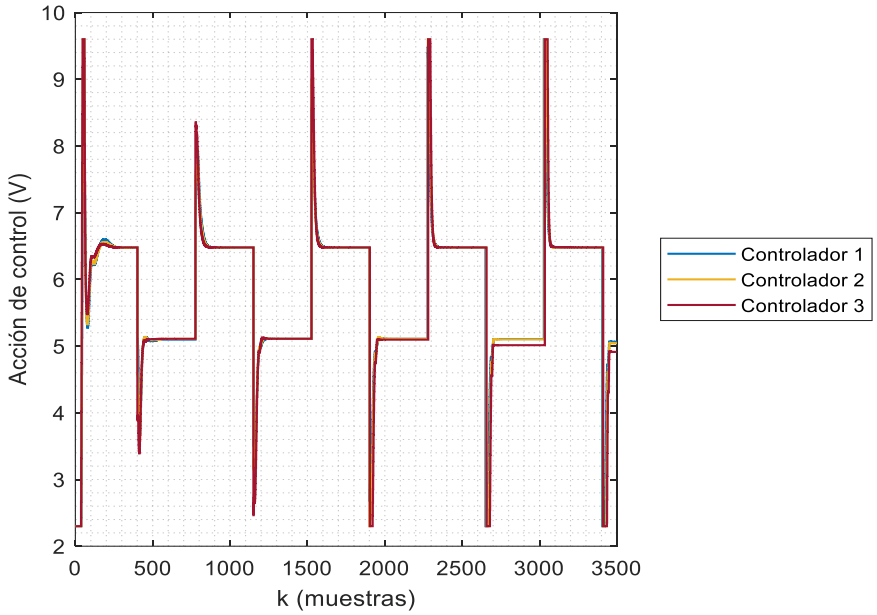


Figura 101

Error para diferentes coeficientes de adaptación del controlador PID sintonizado con técnicas MRAC.

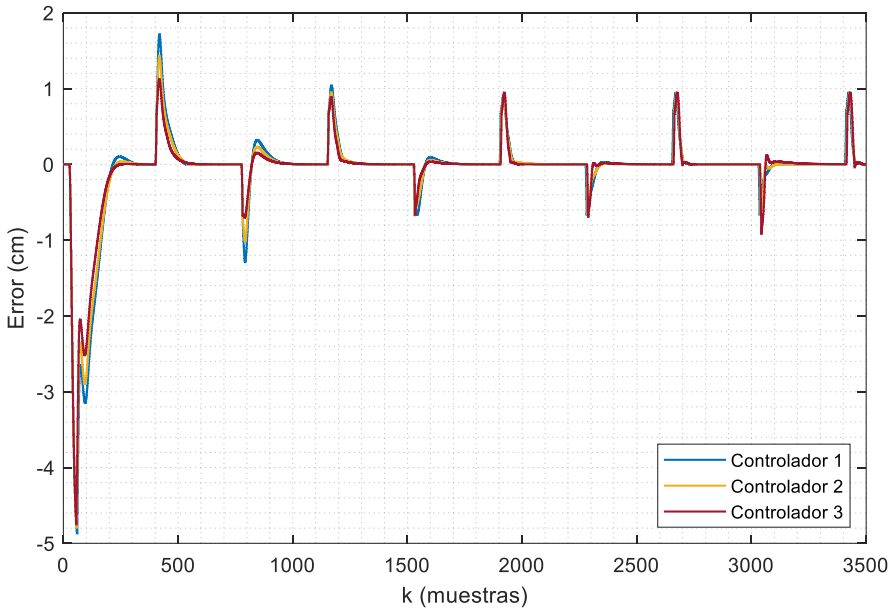
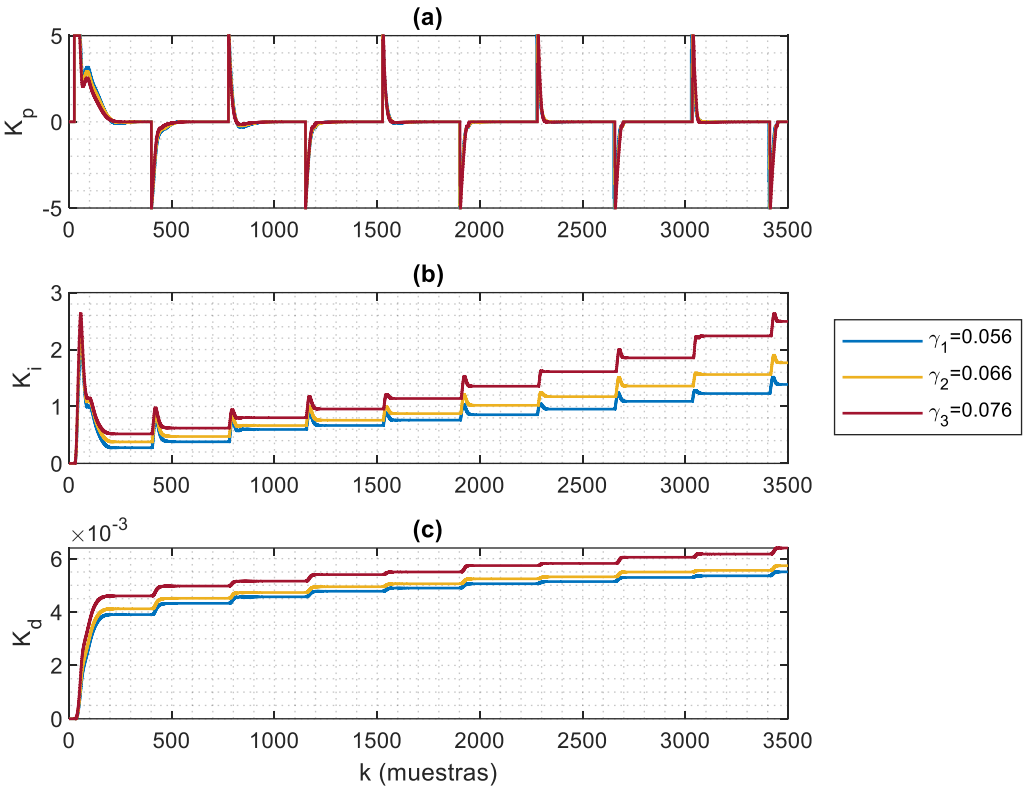


Figura 102

Ajuste en el tiempo de los parámetros para diferentes coeficientes de adaptación del controlador PID sintonizado con técnicas MRAC: (a) K_p; (b) K_i; (c) K_d.



En la Tabla 23 se muestra una comparativa de las características de las respuestas temporales para los diferentes coeficientes de adaptación de controladores PID sintonizados con técnicas MRAC.

Finalmente, se presenta la Tabla 24 con los mejores controladores, permitiendo seleccionar el mejor controlador en función de los criterios de desempeño establecidos.

Se observa que el controlador PID sintonizado con técnicas MRAC muestra el mejor desempeño entre los controladores evaluados, presentando un sobreimpulso prácticamente nulo y un tiempo de establecimiento de 160 segundos.

Tabla 23

Características de la Respuesta temporal de Controladores PID sintonizado con técnicas MRAC.

	Coeficientes de adaptación	Sobreimpulso (%)	Tiempo de pico (s)	Tiempo de establecimiento (s)
Controlador 1	$\gamma_p = 0.056$ $\gamma_i = 2.5e^{-7}$ $\gamma_d = 5$	0.0778	296	164
Controlador 2	$\gamma_p = 0.066$ $\gamma_i = 3e^{-7}$ $\gamma_d = 7$	0.000598	288	160
Controlador 3	$\gamma_p = 0.076$ $\gamma_i = 4e^{-7}$ $\gamma_d = 9$	1.0728	216	148

Tabla 24

Características de la Respuesta temporal de los diferentes Controladores MRAC.

Controlador	Coeficientes de adaptación	Sobreimpulso (%)	Tiempo de pico (s)	Tiempo de establecimiento (s)
Controlador MRAC con Regla de MIT	$\gamma_1 = 0.001$	9.8382	264	396
Controlador MRAC con Método de Lyapunov	$\gamma_1 = 0.0001$	17.5089	292	500
Controlador PID sintonizado con técnicas MRAC	$\gamma_p = 0.066$ $\gamma_i = 3e^{-7}$ $\gamma_d = 7$	0.000598	288	160

CONCLUSIONES

- Se logró diseñar un controlador no lineal adaptativo eficiente para el control del nivel de agua en un sistema de tanques en cascada, cumpliendo con el objetivo general del estudio. El controlador demostró ser capaz de mantener el nivel de agua dentro de los parámetros deseados.
- La simulación del modelo no lineal en lazo abierto usando MATLAB permitió identificar las características dinámicas del sistema. Los resultados mostraron que el modelo no lineal Hammerstein es adecuado para representar el comportamiento real del sistema de tanques en cascada, proporcionando una base sólida para el diseño y ajuste del controlador adaptativo.
- El análisis de los resultados del control no lineal adaptativo en el proceso de adaptación de una consigna tracking reveló una capacidad destacada del sistema para seguir la señal de referencia con un mínimo error. El controlador adaptativo ajustó sus parámetros de manera efectiva para minimizar el sobreimpulso y el tiempo de establecimiento, mejorando la estabilidad y precisión del sistema.
- Al comparar el rendimiento del algoritmo de control no lineal adaptativo bajo diversas pruebas y valores de adaptación, se determinó que el controlador PID sintonizado con técnicas MRAC presenta un mejor desempeño en términos de tiempo de establecimiento y sobreimpulso en comparación con los controladores MRAC con el método Lyapunov y la regla MIT. El controlador PID sintonizado con técnicas MRAC mostró el mejor desempeño presentando un sobreimpulso prácticamente nulo y un tiempo de establecimiento de 160 segundos.

RECOMENDACIONES

- Se recomienda implementar mejoras adicionales en el algoritmo del controlador no lineal adaptativo para optimizar aún más su rendimiento. Esto incluye la exploración de técnicas avanzadas de sintonización de parámetros y la integración de métodos de control predictivo para anticipar y compensar perturbaciones antes de que afecten significativamente el sistema.
- Realizar pruebas experimentales adicionales con diferentes configuraciones de los tanques y variaciones en los niveles iniciales de agua proporcionan una comprensión más completa del rendimiento del controlador bajo diversas condiciones. Estas pruebas ayudarán a identificar posibles limitaciones del sistema y áreas para futuras mejoras.
- El éxito del controlador no lineal adaptativo en este estudio, se recomienda investigar su aplicación en otros sistemas no lineales en la industria. Particularmente, su uso en procesos industriales complejos que requieren un control preciso y adaptativo genera beneficios significativos en términos de eficiencia y reducción de costos operativos.
- Explorar la integración del sistema de control con tecnologías de Internet Industrial de las Cosas (IIoT, Industrial Internet of Things) permitirá una supervisión y control remoto más eficiente. La implementación de sens ores inteligentes y comunicación en tiempo real mejora la capacidad del sistema para responder a cambios dinámicos y optimizar su desempeño en tiempo real.

REFERENCIAS BIBLIOGRÁFICAS

Alvarado-Tabacchi, I. (2018). *Identificación de modelos Wiener y Hammerstein aplicados al diseño de un control PID predictivo de pH.* [Tesis de Máster en Ingeniería Mecánico-Eléctrica con Mención en Automática y Optimización, Universidad de Piura]. Repositorio Institucional Pirhua. https://pirhua.udep.edu.pe/item/bda04cbb-c1ce-4ab0-95c6-444dd949972c

Arduino Due. (s.f.). Arduino Store. https://store-usa.arduino.cc/products/arduino-due?selectedStore=us

Astrom, K. J., & Wittenmark, B. (2008). *Adaptive Control* (2.ª ed.). Dover Publications. (Publicado originalmente en 1994).

Bitter, R. Mohiuddin, T., & Nawrocki, M. (2017). *LabVIEW: Advanced Programming Techiniques*. CRC press. https://www.taylorfrancis.com/books/mono/10.1201/9780849333255/labview-rick-bitter-matt-nawrocki-taqi-mohiuddin

Brouri, A., Giri, F., Mkhida, A., Chaoui, F. Z., Elkarkri, A., & Chhibat, M. L. (2014). Identification of Nonlinear Systems Structured by Hammerstein-Wiener Model. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, 8*(5), 738-741.

Castaño Giraldo, S. A. (2016). *Estudo de técnicas de sintonia do preditor de Smith filtrado para sistemas multivariáveis com atraso*. [Tesis de maestría, Universidad Federal de Santa Catarina]. Repositorio Institucional UFSC. https://repositorio.ufsc.br/xmlui/handle/123456789/167672

Castaño Giraldo, S. A. (s.f.). *Control Automático Educación*. https://controlautomaticoeducacion.com/arduino/sensor-de-flujo-yf-s201-medicion-de-caudal-con-arduino/#Interfaz_del_YF-S201_con_Arduino

Cevallos Ulloa, H., & Navarrete Díaz, D. R. (2019). *Análisis comparativo de controladores clásicos versus controladores avanzados aplicados a una planta real de control de*

nivel de dos tanques no interactivos bajo la plataforma de LABVIEW. [Tesis de Maestría en Automatización y Control Industrial, Escuela Superior Politécnica del Litoral]. Repositorio de ESPOL. https://www.dspace.espol.edu.ec/handle/123456789/52559

Cui, M., Liu, H., Li, Z., Tang, Y., & Guan, X. (2014). Identification of Hammerstein model using functional link artificial neural network. *Neurocomputing*, 142, 419-428. https://doi.org/10.1016/j.neucom.2014.03.051

Cuzcano Rivas, A. B. (2021). *Diseño de un Control No Lineal Por Modo Deslizante para Controlar Nivel del Proceso Tanque con Agua*. [Tesis de doctorado, Universidad Nacional del Callao]. Repositorio Institucional UNAC. https://repositorio.unac.edu.pe/handle/20.500.12952/6321

Gomáriz, S., Biel, D., Matas, J., & Reyes, M. (2000). *Teoría de control: Diseño electrónico* (2.ª ed.). Edicions UPC. https://upcommons.upc.edu/handle/2099.3/36214

Gómez, J. C., & Baeyens, E. (2004). Identification of block-oriented nonlinear systems using orthonormal bases. *Journal of Process Control*, 14(6), 685-697. https://doi.org/10.1016/j.jprocont.2003.09.010

Higuera Arias C., & Camacho Poveda, E. C. (2014). *Evaluación de desempeño de controladores para regulación de temperatura y nivel en tanques en cascada*. [Tesis de pregrado, Universidad Pedagógica y Tecnológica de Colombia]. Repositorio Institucional UPTC. https://repositorio.uptc.edu.co/items/b8afdfcf-474d-424e-9be5-e04aca07237a

Introducción a Modbus. (2014). National Instruments. https://www.ni.com/es/shop/labview/introduction-to-modbus-using-labview.html

Lee, T. T., & Jeng, J. T. (1998). The Chebyshev-Polynomials-Based Unified Model Neural Networks for Function Approximation. *IEEE Transactions on Systems, Man and Cybernetics*, 28(6), 925-935. https://ieeexplore.ieee.org/abstract/document/735405

Matich, D. J. (2001). Redes Neuronales: Conceptos Básicos y Aplicaciones [cátedra]. *Informática Aplicada a la Ingeniería de Procesos – Orientación I*, Universidad Tecnológica Nacional, México.

MATLAB - El lenguaje del cálculo técnico. (s.f.). MathWorks. https://la.mathworks.com/products/ matlab.html

McNelis, P. D. (2005). Neuronal Networks in Finance: Gaining predictive edge in the market (1.ª ed.). Academic Press.

Mihaich, F. (2014). *Aplicación de redes neuronales en la clasificación de imágenes*. [Tesis de pregrado, Universidad Nacional de Córdoba]. Repositorio Digital UNC. http://hdl.handle.net/11086/29841

Modbus Organization. (2012). *MODBUS Application Protocol Specification V1.1b3*. https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

Morales Espitia, C. A., & Moreno Bermudez, J. L. (2017). *Identificación de un Sistema de control de nivel y flujo utilizando Redes Neuronales Artificiales*. [Tesis de pregrado, Universidad Distrital Francisco José de Caldas]. Repositorio Institucional UDFJC. http://hdl.handle.net/11349/7712

Ogata, K. (2010). *Ingeniería de control moderna* (5.ª ed.). Pearson Educación.

Park, H. C., Sung, S. W., & Lee, J. (2006). Modeling of Hammerstein − Wiener processes with special input test signals. *Industrial & engineering chemistry research*, 45(3), 1029-1038. https://pubs.acs.org/doi/abs/10.1021/ie050540a

Parra Quispe, A. A. (2007). *Diseño e Implementación de Controladores PID Industriales.* [Tesis de maestría, Universidad Nacional de Ingeniería]. Repositorio Institucional UNI. http://hdl.handle.net/20.500.14076/149

Patra, J. C., & Kot, A. C. (2002). Nonlinear Dynamic System Identification Using Chebyshev Functional Link Artificial Neural Networks. *IEEE Transactions on Systems, Man. and Cybernetics*, 32(4), 505-511. https://ieeexplore.ieee.org/abstract/document/1018769

Rodriguez Rubio, F., & López Sánchez, M. (1996). *Control Adaptativo y Robusto* (1.ª ed.). Universidad de Sevilla. https://editorial.us.es/es/detalle-libro/90009/control-adaptativo-y-robusto

Rodriguez, R., Paredes, V., Noa, G., & Trucios, L. (2019). Grey box identification and adaptive control in a water level system. *IOP Conference Series: Materials Science and Engineering*, 707(1), 012003. https://iopscience.iop.org/article/10.1088/1757-899X/707/1/012003/meta

Rojas Moreno, A., & Aquize Palacios, R. D. (2011). *Implementación de un sistema de control no lineal Backstepping multivariable para la planta piloto tanque con agua*. [Tesis de Maestría, Universidad Nacional de Ingeniería]. Repositorio Institucional UNI. http://hdl.handle.net/20.500.14076/157

Schoukens, J., & Ljung, L. (2019). Nonlinear System Identification: A User-Oriented Road Map. *IEEE Control Systems Magazine*, 39(6), 28 - 99. https://ieeexplore.ieee.org/abstract/document/8897147

Simulación y diseño basado en modelos con Simulink. (s.f.). MathWorks. https://la.mathworks.com/ products/simulink.html

Tangirala, A. K. (2015). *Principles of System Identification: Theory and Practice* (1.ª ed.). CRC Press. https://doi.org/10.1201/9781315222509

USB-621x User Manual. (2009). National Instruments. https://www.ni.com/docs/en-US/bundle/usb-621x-manual/resource/usb-621x-manual.pdf

Vojtesek, J., & Dostal, P. (2015). Adaptive Control of Water Level in Real Model of Water Tank. *International Conference on Process Control (PC)*, 308 – 313. https://ieeexplore.ieee.org/abstract/document/7169981

What is the Modbus Protocol & How Does It Work?. (2014). National Instruments. https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html

ANEXOS

Anexo N°1 Matriz de Consistencia

Título: Diseño de un Control No Lineal Adaptativo para controlar el nivel de agua aplicado a un Sistema de Tanques en Cascada usando NIDAQ USB 6211

PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLES	METODOLOGÍA
Problema General ¿De qué manera el desarrollo del algoritmo de control no lineal adaptativo aplicado a un sistema de tanques en cascada contribuirá a la mejora de la performance del control de nivel de agua? **Problema Específico 1:** ¿Cómo influye el modelo experimental en la actuación del control de nivel de agua? **Problema Específico 2:** ¿Cómo afecta la adaptación basada en modelo de referencia en el control no lineal adaptativo para controlar el nivel de agua? **Problema Específico 3:** ¿De qué manera se mide la performance del sistema de control no lineal adaptativo en el control de nivel de agua cuando se adecúa el parámetro de adaptación?	**Objetivo General** Realizar el diseño de un control no lineal adaptativo para controlar el nivel de agua aplicado a un sistema de tanques en cascada. **Objetivo Específico 1:** Simular el modelo no lineal como respuesta en lazo abierto usando el programa MATLAB. **Objetivo Específico 2:** Analizar los resultados en el control no lineal adaptativo en el proceso de adaptación de una consigna tracking. **Objetivo Específico 3**: Comparar el performance del algoritmo de control no lineal adaptativo en diversas pruebas que involucren diferentes valores de adaptación para el control de nivel de agua.	**Hipótesis general** La ley de control no lineal adaptativa será capaz de darle una alta performance al control de nivel en un sistema de tanques en cascada. **Hipótesis Específica 1:** Un adecuado modelo no lineal facilitara la simulación de la respuesta a lazo abierto. **Hipótesis Específica 2:** Un adecuado análisis del control no lineal adaptativo resultará en una respuesta que suprimirá el sobre impulso en el tiempo hasta mimetizarse al modelo de referencia. **Hipótesis Específica 3:** El desarrollo del algoritmo de control no lineal adaptativo y sus diferentes valores de adaptación para el control de nivel de agua permitirá validar la performance del control.	**Variable Independiente:** Es la variable de acción del actuador de la válvula de control que proporciona el caudal que se requiere en el tanque de agua, se denota por la variable u. **Variable dependiente:** Esta variable está relacionada con el nivel de agua en el tanque principal y se representa por la variable h.	**Tipo de investigación:** Es una investigación científica, en el ámbito de la ingeniería aplicada. **Población y muestra:** No se utilizaron población y muestra. **Técnicas e instrumentos de recolección:** Se han utilizado diversas fuentes bibliográficas para definir un modelo teórico y experimental preciso. La información se procesará utilizando algoritmos en el dominio del tiempo discreto. **Técnicas e instrumentos de análisis y procesamiento de datos:** Se describen las formas en que se resumen las variables y cómo se presentarán los resultados. Estos se mostrarán en gráficos obtenidos de simulaciones realizadas en MATLAB y LabVIEW.

Anexo N°2 Código en MATLAB del Diseño de las señales de prueba para Identificación

```
% ============================================================
% Diseño de las Señales de Prueba para la Identificación
% ============================================================
clc, clearvars, close all;
tsim = 4000;
Ts = 4;
f = 1/Ts;
N = tsim/Ts;
% ============================================================
% 1. Primera Señal de Prueba: u_b
% ============================================================
Band = [0 Ts/150]; % Gama de frecuencias de la señal generada
Range = [0 80]; % Rango de la señal de entrada generada
t0 = 500;
M = tsim + t0;
signal_ub = idinput(M*f,'prbs',Band,Range); % Generación de Señal PRBS
ub = signal_ub(t0*f+1:end);
tb = (1:N)';
figure
plot(tb,ub,'LineWidth',1.5)
title('Señal de prueba u_b')
xlabel('k (muestras)')
ylabel('% apertura')
axis([0 N 0 100])
grid minor
%save('Signal_ub.lvm', 'ub',"-ascii");
% ============================================================
% 2. Segunda Señal de Prueba: u_ba
% ============================================================
uba = ub*0.6; % Señal múltiplo α de la primera señal de prueba
tba = (1:N)';
figure
plot(tba,uba,'LineWidth',1.5)
title('Señal de prueba u_b_\alpha')
xlabel('k (muestras)')
ylabel('% apertura')
axis([0 N 0 100])
grid minor
%save('Signal_uba_06.lvm', 'uba',"-ascii");
% ============================================================
% 3. Tercera Señal de Prueba: u_ms
% ============================================================
numSamples = 25;
holding = (tsim/numSamples)*f;
rng(100);
ruido = 100*rand(1,numSamples);
```

```
ums = zeros(numSamples*holding,1);
for i=1:numSamples
    for j=1:holding
        ums(j+holding*(i-1)) = ruido(i);
    end
end
ums = [ub; ums];
tms = (1:2*N)';
figure
plot(tms,ums,'LineWidth',1.5);
title('Señal de prueba u_m_s')
xlabel('k (muestras)')
ylabel('% apertura')
axis([0 2*N 0 100])
grid minor
save('Signal_PRBS_Noise.lvm', 'signal',"-ascii");
```

Anexo N°3 Código en MATLAB de la Identificación del Modelo Wiener

Función estática no lineal de salida:

```
% ============================================================
% IDENTIFICACIÓN MODELO WIENER - FUNCIÓN DE SALIDA
% ============================================================
clc, clearvars, close all;
% ============================================================
% 1. Carga de Datos
% ============================================================
Ts = 4; % Tiempo de muestreo
tsim = 4000; % Tiempo de simulación
N = tsim/Ts;
% Data ub: 80% de apertura de válvula
ub = load('Signal_ub.lvm');
ub = ub(1:Ts:tsim);
data_ub = load('data_ub_80%.lvm');
yb = data_ub(1:Ts:tsim,6);
yb(yb < 0) = 0;
figure
subplot(2,1,1)
stairs(1:N,yb,LineWidth=1)
title('(a)',' ')
%xlabel('k (muestras)')
ylabel('Nivel Tanque 2 (cm)')
axis([0 N 0 10])
grid minor
% Data uba: 48% de apertura de válvula
uba = load('Signal_uba_06.lvm');
uba = uba(1:Ts:tsim);
data_uba = load('data_uba_48%.lvm');
yba = data_uba(1:Ts:tsim,6);
yba(yba < 0) = 0;
%figure
subplot(2,1,2)
stairs(1:N,yba,LineWidth=1)
title('(b)',' ')
xlabel('k (muestras)')
ylabel('Nivel Tanque 2 (cm)')
axis([0 N 0 10])
grid minor
% División en Datos de Entrenamiento y Validación
div = 600;
yb_t = yb(1:div); yb_v = yb(div+1:end);
yba_t = yba(1:div); yba_v = yba(div+1:end);
% ============================================================
% 2. Declaración de parámetros y variables
% ============================================================
% Construyendo vectores Y[k] para entrenamiento
```

```
n = 2; % Grado del polinomio
Nb_t = length(yb_t); % Número de muestras de Datos de Entrenamiento
Nb_v = length(yb) - Nb_t; % Número de muestras de Datos de Validación
Yb_t = zeros(n,Nb_t);
Yba_t = zeros(n,Nb_t);
for i=1:n
    Yb_t(i,:) = yb_t'.^(i);
    Yba_t(i,:) = yba_t'.^(i);
end
% Inicializando los parámetros
Q = ones(n,1); % Vector de coeficientes
Q(1) = 0; %W(2) = 0;
beta = 1;
eta = 1e-6; % Tasa de aprendizaje
num_epocas = 1500; % Número de épocas para entrenamiento
% Inicializando vectores
e = zeros(Nb_t,num_epocas); % Matriz de errores
V = zeros(num_epocas,1); % Vector de Función de Costo
% ============================================================
% 3. Entrenamiento de la Red Neuronal Artificial
% ============================================================
for epoca=1:num_epocas
    for k = 1:Nb_t
        % Calculando la variable intermedia z[k]
        zb = Q'*Yb_t(:,k);
        zba = Q'*Yba_t(:,k);
        % Calculando el error
        e(k,epoca) = zba - beta*zb;
        % Algoritmo de optimización: Descenso del Gradiente
        % Calculando las derivadas parciales de la Función de costo
        dV_dbeta = e(k,epoca)*(-zb);
        dV_dQ = e(k,epoca)*(Yba_t(:,k)-beta*Yb_t(:,k));

        % Actualizando los pesos
        % Modo Pattern
        beta = beta - eta * dV_dbeta;
        Q = Q - eta * dV_dQ;
        Q(1) = 0; %W(2) = 0;    % Haciendo q1 = 0
    end
    % Calculando el error del ajuste
    Zb_t = Q'*Yb_t;
    Zba_t = Q'*Yba_t;
    er = Zba_t - beta*Zb_t;
    % Función de costo: Error cuadrático promedio
    V(epoca) = sum(er.^2)/(2*(Nb_t));
    % Condición para salir del bucle
    if V(epoca)<2e-2 || epoca>1500
        break;
    end
end
```

```
% Mostrando los parámetros luego del entrenamiento
disp('PARÁMETROS CALCULADOS')
disp('Beta:')
disp(beta)
disp('Coeficientes:')
disp(Q)
% Gráfico de la Función de Costo en función de la época
figure
plot(1:epoca,V(1:epoca),'m')
title('Función de costo vs Épocas')
xlabel('Épocas')
ylabel('Función de Costo')
grid minor
% ================================================================
% 4. Cálculo de la Bondad del Ajuste
% ================================================================
% Calculando bondad del ajuste en Datos de Entrenamiento
Zb_t = Q'*Yb_t;
Zba_t = Q'*Yba_t;
fit_t =(1-goodnessOfFit(beta*Zb_t',Zba_t','NRMSE'))*100;
mse_t = norm(Zba_t'-beta*Zb_t')^2/Nb_t;
% Calculando bondad del ajuste en Datos de Validación
Yb_v = zeros(n,Nb_v);
Yba_v = zeros(n,Nb_v);
for i=1:n
    Yb_v(i,:) = yb_v'.^(i);
    Yba_v(i,:) = yba_v'.^(i);
end
Zb_v = Q'*Yb_v;
Zba_v = Q'*Yba_v;
fit_v = (1-goodnessOfFit(beta*Zb_v',Zba_v','NRMSE'))*100;
mse_v = norm(Zba_v'-beta*Zb_v')^2/Nb_v;
% ================================================================
% 5. Gráfico de los Resultados del Ajuste
% ================================================================
% Gráfico del ajuste en Datos de Entrenamiento
figure
subplot(2,1,1)
stairs(1:Nb_t,[Zba_t' Zb_t'*beta],'LineWidth',1.5)
title('(a)',' ')
%xlabel('k (muestras)')
ylabel('Nivel Tanque 2 (cm)')
legend('Señal z_b_\alpha','Señal \betaz_b',Location='northwest')
text(190,1.2,['Fit_{train}(%):',num2str(fit_t)],'EdgeColor','black','LineSty
le','-')
grid minor
disp('Gráfico del ajuste en Datos de Entrenamiento')
fprintf('Fit: %.4f \nmse: %.4f\n',fit_t,mse_t);
% Gráfico del ajuste en Datos de Validación
subplot(2,1,2)
```

```
stairs(1:Nb_v,[Zba_v' Zb_v'*beta],'LineWidth',1.5)
title('(b)',' ')
xlabel('k (muestras)')
ylabel('Nivel Tanque 2 (cm)')
legend('Señal z_b_\alpha','Señal \betaz_b',Location='northwest')
text(130,0.8,['Fit_{validation}(%):',num2str(fit_v)],'EdgeColor','black','Li
neStyle','-')
grid minor
disp('Gráfico del ajuste en Datos de Validación')
fprintf('Fit: %.4f \nmse: %.4f\n',fit_v,mse_v);
% ============================================================
% 6. No linealidad de salida
% ============================================================
yy = 0:0.01:Nb_t/100-0.01;
yy_ = zeros(n,Nb_t);
for i=1:n
    yy_(i,:) = yy'.^(i);
end
zz = Q'*yy_;
figure
plot(zz,yy,'LineWidth',1.5)
% title('No linealidad de salida','FontSize',16)
xlabel('Señal intermedia z','FontSize',14)
ylabel('Señal de salida y','FontSize',14)
grid minor
qi = Q; % vector de coeficientes de polinomio z = f(y)
save('qi.txt','qi',"-ascii")
```

Subsistema Dinámico Lineal:

```
% ============================================================
% IDENTIFICACIÓN MODELO WIENER - SUBSISTEMA LINEAL
% ============================================================
clc, clearvars, close all;
% ============================================================
% 1. Carga de Datos
% ============================================================
Ts = 4; % Tiempo de muestreo
fs = 1/Ts;
tsim = 4000; % Número de muestras
% Data ub: 80% de apertura de válvula
data_ub = load('data_ub_80%.lvm');
ub = load('Signal_ub.lvm');
ub = ub(1:Ts:tsim);
yb = data_ub(1:Ts:tsim,6);
yb(yb < 0) = 0;
% Coeficientes de polinomio z = f(y)
qi = load('qi.txt');
disp('Coeficientes del polinomio z = f(y)')
disp(qi)
Nb = length(ub);
N = length(qi);
zb = zeros(Nb,1);
for i=1:N
    zb = zb + qi(i)*yb.^i;
end
stairs(1:Nb,zb,'LineWidth',1.2)
xlabel('k (muestras)',FontSize=12)
ylabel('Señal intermedia z_b',FontSize=12)
grid minor
 ============================================================
% 2. Identificación Lineal con Modelo ARX
% ============================================================
data = iddata(zb,ub,Ts,'Name','data_ident_Lineal');
delay = 24;
na = 2;
nb = 1;
nk = delay*fs;
modelo_arx = arx(data, [na nb nk]);
[yy, fit] = compare(data,modelo_arx);
figure
compare(data,modelo_arx)
fprintf('Fit: %.4f',fit);
% Función de Tranferencia
Gz_arx = tf(modelo_arx)
```

Bondad del Ajuste del Modelo Wiener:

```
% ============================================================
% IDENTIFICACIÓN MODELO WIENER - COMPLETO
% ============================================================
clc, clearvars, close all;
% ============================================================
% 1. Carga de Datos
% ============================================================
Ts = 4; % Tiempo de muestreo
tsim = 4000; % Tiempo de simulación
ret = 5; % Número de muestras de retraso
% Data ub: 80% de apertura de válvula
ub = load('Signal_ub.lvm');
ub = ub(1:Ts:tsim);
data_ub = load('data_ub_80%.lvm');
yb = data_ub(1:Ts:tsim,6); yb(yb < 0) = 0;
Nb = length(ub); % Número de muestras
% Coeficientes de polinomio z = f(y)
qi = load('qi.txt');
% Parámetros de la Función de activación
a_hat = [1.274, -0.2897]';
b_hat = 0.002849;
na = length(a_hat);
nb = length(b_hat);
% ============================================================
% 2. Cálculo de la Bondad del Ajuste del Modelo Wiener
% ============================================================
z = zeros(Nb,1); z(1:nb+ret) = qi(2)*yb(1:nb+ret).^2;
for k = 3+ret:Nb
    z(k) = a_hat(1)*z(k-1) + a_hat(2)*z(k-2) + b_hat*ub(k-1-ret);
    if z(k)<0
        z(k)=0;
    end
end
y_wiener = sqrt(z/qi(2));
fit=(1-goodnessOfFit(y_wiener,yb,'NRMSE'))*100;
mserror = norm(yb-y_wiener).^2/Nb;
% ============================================================
% 3. Gráfico de los Resultados del Ajuste
% ============================================================
figure
stairs(1:Nb,[yb y_wiener],'LineWidth',1.5)
xlabel('k (muestras)',FontSize=12)
ylabel('Nivel Tanque 2 (cm)',FontSize=12)
legend('Data Real','Modelo Wiener')
text(35,9.3,['Fit(%):',num2str(fit)],'EdgeColor','black','LineStyle','-')
grid minor
disp('Modelo Wiener'); fprintf('Fit: %.4f \nmse: %.4f\n',fit,mserror);
```

Anexo N°4 Código en MATLAB de la Identificación del Modelo Hammerstein

Subsistema Dinámico Lineal:

```
% ============================================================
% IDENTIFICACIÓN MODELO HAMMERSTEIN - SUBSISTEMA DINÁMICO LINEAL
% ============================================================
clc, clearvars, close all;
% ============================================================
% 1. Carga de Datos
% ============================================================
Ts = 4; % Tiempo de muestreo
fs = 1/Ts;
tsim = 4000; % Número de muestras
% Data ub: 80% de apertura de válvula
data_ub = load('data_ub_80%.lvm');
ub = load('Signal_ub.lvm');
ub = ub(1:Ts:tsim);
yb = data_ub(1:Ts:tsim,6);
yb(yb < 0) = 0;
Nb = tsim/Ts;
stairs(1:Nb,yb,'LineWidth',1.2)
xlabel('k (muestras)',FontSize=12)
ylabel('Nivel Tanque 2 (cm)',FontSize=12)
grid minor
% ============================================================
% 2. Identificación Lineal con Modelo ARX
% ============================================================
datos = iddata(yb,ub,Ts,'Name','data_ident_Lineal');
[datosd,Tr]=detrend(datos);
plot(datosd)
delay = 24;
na = 2;
nb = 1;
nk = delay*fs;
modelo_arx = arx(datosd, [na nb nk]);
[yy, fit] = compare(datosd,modelo_arx);
figure
compare(datosd,modelo_arx)
fprintf('Fit: %.4f',fit);
% Función de Tranferencia
Gz_arx = tf(modelo_arx)
```

Función estática no líneal de entrada:

```
% ============================================================
% IDENTIFICACIÓN MODELO HAMMERSTEIN - FUNCIÓN DE ENTRADA
% ============================================================
clc, clearvars, close all;
% ============================================================
% 1. Carga de Datos
% ============================================================
Ts = 4; % Tiempo de muestreo
tsim = 8000; % Tiempo de simulación
M = tsim/Ts;
% Data de Entrenamiento
ums = load('Signal_ub_ums.lvm');
ums = ums(1:Ts:tsim);
a = 0; b = 100;
x = (2*ums - a - b)/(b - a); % Normalización de la entrada [-1,1]
datos_t = load('data_ub_ums.lvm');
yms = datos_t(1:Ts:tsim,6);
yms(yms < 0) = 0;
datos = iddata(yms,x,Ts);
figure
stairs(1:M,yms,LineWidth=1)
xlabel('k (muestras)')
ylabel('Nivel Tanque 2 (cm)')
axis([0 M 0 10])
grid minor
% Data de Validación
u_v = load('Signal_ums.lvm');
u_v = u_v(1:Ts:tsim/2);
datos_v = load('data_ums.lvm');
y_v = datos_v(1:Ts:tsim/2,6);
y_v(y_v < 0) = 0;
figure
subplot(2,1,1)
stairs(1:M/2,u_v,LineWidth=1)
title('(a)',' ')
ylabel('% apertura')
grid minor
subplot(2,1,2)
stairs(1:M/2,y_v,LineWidth=1)
title('(b)',' ')
xlabel('k (muestras)')
ylabel('Nivel Tanque 2 (cm)')
grid minor
% ============================================================
% 2. Declaración de parámetros para la FLANN
% ============================================================
% Parámetros de la Función de activación
a_hat = [1.163, -0.1834]';
```

```
b_hat = 0.002851;
na = length(a_hat);
nb = 1;
delay = 5; % Retraso
% Parámetros de la FLANN
N = 3; % Grado de la función no lineal
Nms = length(x); % Cantidad de muestras
W = ones(N+1,1); % Vector de pesos
eta = 1; % Tasa de aprendizaje
num_epocas = 100; % Número máximo de épocas
epoca = 0;
% Inicializando vectores
e = zeros(Nms,num_epocas); % Matriz de errores
V = zeros(num_epocas,1); % Vector de Función de Costo
y_hat = zeros(Nms,1); % Vector de salida estimada
% Vector de Expansión de función
u_FE= importdata('uFE_N3_Ham.mat');
% u_FE = zeros(N+1,Nms);
% for k = 1+nb+ret:Nms
%     u_FE(:,k) = chebyshevT((0:N)',x(k-1-ret));
% end
% ================================================================
% 3. Entrenamiento de la FLANN
% ================================================================
y_hat(1:delay+nb) = yms(1:delay+nb);
while 1
    epoca = epoca+1;
    for k = 1+nb+delay:Nms
        % Calculando la salida y_hat(k)
        y_hat(k) = a_hat(1)*y_hat(k-1)+ a_hat(2)*y_hat(k-2)+
W'*u_FE(:,k)*b_hat;
        if y_hat(k) < 0
            y_hat(k) = 0;
        end
        % Calculando el error
        e(k,epoca) = yms(k) - y_hat(k);

        % Algoritmo de optimización: Descenso del Gradiente
        % Actualizando los pesos mediante retropropagación
        dE_dW = - e(k,epoca) * u_FE(:,k) * b_hat;

        % Actualizando los pesos: Modo Pattern
        W = W - eta * dE_dW;
    end
    % Calculando el error del ajuste
    y_hat = zeros(Nms,1);
    y_hat(1:delay+nb) = yms(1:delay+nb);
    for k = 1+nb+delay:Nms
        y_hat(k) = a_hat(1)*y_hat(k-1)+ a_hat(2)*y_hat(k-2) +
W'*u_FE(:,k)*b_hat;
```

```
        if y_hat(k)<0
            y_hat(k)=0;
        end
    end
    er = yms - y_hat;
    % Función de costo: Error cuadrático promedio
    V(epoca) = sum(er.^2)/(2*Nms);
    % Condición para salir del bucle
    if V(epoca)<=3.1e-2 || epoca>=num_epocas
        break;
    end
end
% Mostrar los pesos luego del entrenamiento
disp('Pesos finales:');
disp(W);
% Gráfico de la Función de Costo en función de la época
figure
plot(1:epoca,V(1:epoca),'m')
title('Función de costo vs Épocas')
xlabel('Épocas')
ylabel('Función de Costo')
grid minor
% ================================================================
% 4. Cálculo de la Bondad del Ajuste
% ================================================================
% Calculando bondad del ajuste en Datos de Entrenamiento
syms u
u_s = (2*u-a-b)/(b-a);
u_FE_sym = vpa(chebyshevT((0:N)',u),8);
f= vpa(W'*u_FE_sym,8)
v = double(subs(f,u,x)); % Variable intermedia
y_ham = zeros(Nms,1); y_ham(1:delay+nb) = yms(1:delay+nb);
for k=1+nb+delay:Nms
    y_ham(k) = a_hat(1)*y_ham(k-1) + a_hat(2)*y_ham(k-2) + b_hat(1)*v(k-1-
delay);
    if y_ham(k)<0
        y_ham(k)=0;
    end
end
fit = (1-goodnessOfFit(y_ham,yms,'NRMSE'))*100; % NRMSE
mserror = norm(yms-y_ham)^2/Nms; % MSE
% Calculando bondad del ajuste en Datos de Validación
x_v = (2*u_v - a - b)/(b - a);
data_v = iddata(y_v,x_v,Ts);
yms_v = data_v.y; x_v = data_v.u;
Nms_v = length(yms_v);
v_v = double(subs(f,u,x_v)); % Variable intermedia
y_hat_v = zeros(Nms_v,1); y_hat_v(1:delay+nb) = yms_v(1:delay+nb);
for k=1+nb+delay:Nms_v
```

```
    y_hat_v(k) = a_hat(1)*y_hat_v(k-1) + a_hat(2)*y_hat_v(k-2) +
b_hat(1)*v_v(k-1-delay);
    if y_hat_v(k)<0
        y_hat_v(k)=0;
    end
end
fit_v = (1-goodnessOfFit(y_hat_v,yms_v,'NRMSE'))*100;
mse_v = norm(yms_v-y_hat_v)^2/Nms_v;
% ============================================================
% 5. Gráfico de los Resultados del Ajuste
% ============================================================
% Gráfico del ajuste en Datos de Entrenamiento
figure
stairs(1:Nms,[yms y_ham],'LineWidth',1.5)
xlabel('k (muestras)',FontSize=12)
ylabel('Nivel Tanque 2 (cm)',FontSize=12)
legend('Proceso Real','Modelo Hammerstein',Location='northwest')
text(80,8,['Fit(%):',num2str(fit)],'EdgeColor','black','LineStyle','-')
grid minor
disp('Modelo Hammerstein');
fprintf('Fit: %.4f \nmse: %.4f\n',fit,mserror);
% Gráfico del ajuste en Datos de Validación
figure
stairs(1:Nms_v,[yms_v y_hat_v],'LineWidth',1.5)
xlabel('k (muestras)',FontSize=12)
ylabel('Nivel Tanque 2 (cm)',FontSize=12)
legend('Proceso Real','Modelo Hammerstein',Location='northwest')
text(40,9.7,['Fit(%):',num2str(fit_v)],'EdgeColor','black','LineStyle','-')
grid minor
disp('Modelo Hammerstein');
fprintf('Fit: %.4f \nmse: %.4f\n',fit_v,mse_v);
% ============================================================
% 6. No linealidad de entrada
% ============================================================
uu = -1:0.01:1;
vv = double(subs(f,u,uu));
figure
plot(uu,vv,'LineWidth',1.5)
xlabel('Señal de entrada normalizada x','FontSize',12)
ylabel('Señal intermedia v','FontSize',12)
grid minor
```

Anexo N°5 Código en MATLAB de la Identificación del Modelo Hammerstein-Wiener

Subsistema dinámico lineal:

```
% ================================================================
% IDENTIFICACIÓN MODELO HAMMERSTEIN-WIENER - SUBSISTEMA LINEAL
% ================================================================
clc, clearvars, close all;
% ================================================================
% 1. Carga de Datos
% ================================================================
Ts = 4; % Tiempo de muestreo
fs = 1/Ts;
tsim = 4000; % Número de muestras
% Data ub: 80% de apertura de válvula
data_ub = load('data_ub_80%.lvm');
ub = load('Signal_ub.lvm');
ub = ub(1:Ts:tsim);
yb = data_ub(1:Ts:tsim,6);
yb(yb < 0) = 0;
% Coeficientes de polinomio z = f(y)
qi = load('qi.txt');
disp('Coeficientes del polinomio z = f(y)')
disp(qi)
Nb = length(ub);
N = length(qi);
zb = zeros(Nb,1);
for i=1:N
    zb = zb + qi(i)*yb.^i;
end
stairs(1:Nb,zb,'LineWidth',1.2)
xlabel('k',FontSize=12)
ylabel('Señal intermedia z_b',FontSize=12)
grid minor
% ================================================================
% 2. Identificación Lineal con Modelo ARX
% ================================================================
datos = iddata(zb,ub,Ts,'Name','data_ident_Lineal');
[datosd,Tr]=detrend(datos);
delay = 24;
na = 2;
nb = 1;
nk = delay*fs;
modelo_arx = arx(datosd, [na nb nk]);
[yy, fit] = compare(datosd,modelo_arx);
figure
compare(datosd,modelo_arx)
fprintf('Fit: %.6f',fit);
% Función de Tranferencia
Gz_arx = tf(modelo_arx)
```

Función estática no lineal de entrada:

```
% ============================================================
% IDENTIFICACIÓN MODELO HAMMERSTEIN-WIENER - FUNCIÓN DE ENTRADA
% ============================================================
clc, clearvars, close all;
% ============================================================
% 1. Carga de Datos
% ============================================================
Ts = 4; % Tiemp ode muestreo
tsim = 8000; % Tiempo de simulación
M = tsim/Ts;
% Data de Entrenamiento
ums = load('Signal_ub_ums.lvm');
ums = ums(1:Ts:tsim);
a = 0; b = 100;
x = (2*ums - a - b)/(b - a); % Normalización de la entrada [-1,1]
data = load('data_ub_ums.lvm');
yms = data(1:Ts:tsim,6);
yms(yms < 0) = 0;
% Coeficientes de polinomio z = f(y)
qi = load('qi.txt');
disp('Coeficientes del polinomio z = f(y)')
disp(qi)
Mz = length(ums);
Nz = length(qi);
zms = zeros(Mz,1);
for i=1:Nz
    zms = zms + qi(i)*yms.^i;
end
figure
stairs(1:M,zms,LineWidth=1)
xlabel('k (muestras)')
ylabel('Señal intermedia z_{ms}')
grid minor
% ============================================================
% 2. Declaración de parámetros para la FLANN
% ============================================================
% Parámetros de la Función de activación
a_hat = [1.189, -0.2027]';
b_hat = 0.00406;
na = length(a_hat);
nb = 1;
delay = 5; % Retraso
% Parámetros de la FLANN
N = 3; % Grado de la función no lineal
Nms = length(x); % Cantidad de muestras
W = ones(N+1,1); % Vector de pesos
eta = 5; % Tasa de aprendizaje
num_epocas = 100; % Número máximo de épocas
```

```
epoca = 0;
% Inicializando vectores
e = zeros(Nms,num_epocas); % Matriz de errores
V = zeros(num_epocas,1); % Vector de Función de Costo
z_hat = zeros(Nms,1); % Vector de salida estimada
% Vector de Expansión de función
u_FE= importdata('uFE_N3_Ham.mat');
% u_FE = zeros(N+1,Nms);
% for k = 1+nb+ret:Nms
%     u_FE(:,k) = chebyshevT((0:N)',x(k-1-ret));
% end
% ================================================================
% Entrenamiento de la Red Neuronal Artificial de Enlace Funcional
% ================================================================
z_hat(1:delay+nb) = zms(1:delay+nb);
while 1
    epoca = epoca+1;
    for k = 1+nb+delay:Nms
        % Calculando la salida z_hat(k)
        z_hat(k) = a_hat(1)*z_hat(k-1)+ a_hat(2)*z_hat(k-2)+
W'*u_FE(:,k)*b_hat;
        if z_hat(k) < 0
            z_hat(k) = 0;
        end
        % Calculando el error
        e(k,epoca) = zms(k) - z_hat(k);

        % Algoritmo de optimización: Descenso del Gradiente
        % Actualizando los pesos mediante retropropagación
        dE_dW = - e(k,epoca) * u_FE(:,k) * b_hat;
        % Actualizando los pesos: Modo Pattern
        W = W - eta * dE_dW;
    end
    % Calculando el error del ajuste
    z_hat = zeros(Nms,1);
    z_hat(1:delay+nb) = zms(1:delay+nb);
    for k = 1+nb+delay:Nms
        z_hat(k) = a_hat(1)*z_hat(k-1)+ a_hat(2)*z_hat(k-2) +
W'*u_FE(:,k)*b_hat;
        if z_hat(k)<0
            z_hat(k)=0;
        end
    end
    er = zms - z_hat;
    % Función de costo: Error cuadrático promedio
    V(epoca) = sum(er.^2)/(2*Nms);
    % Condición para salir del bucle
    if V(epoca)<=5e-1 || epoca>=num_epocas
        break;
    end
```

```
end
% Mostrar los pesos luego del entrenamiento
disp('Pesos finales:');
disp(W);
% Gráfico de la Función de Costo en función de la época
figure
plot(1:epoca,V(1:epoca),'m')
title('Función de costo vs Épocas')
xlabel('Épocas')
ylabel('Función de Costo')
grid minor
% ============================================================
% 4. Cálculo de la Bondad del Ajuste
% ============================================================
% Calculando bondad del ajuste en Datos de Entrenamiento
syms u
u_s = (2*u-a-b)/(b-a);
u_FE_sym = vpa(chebyshevT((0:N)',u),8);
f= vpa(W'*u_FE_sym,8)
v = double(subs(f,u,x)); % Variable intermedia
z_ham = zeros(Nms,1); z_ham(1:delay+nb) = zms(1:delay+nb);
for k=1+nb+delay:Nms
    z_ham(k) = a_hat(1)*z_ham(k-1) + a_hat(2)*z_ham(k-2) + b_hat*v(k-1-
delay);
    if z_ham(k)<0
        z_ham(k)=0;
    end
end
fit = (1-goodnessOfFit(z_ham,zms,'NRMSE'))*100; % NRMSE
mserror = norm(zms-z_ham)^2/Nms; % MSE
% Calculando bondad del ajuste en Datos de Validación
u_v = load('Signal_ums.lvm');
u_v = u_v(1:Ts:tsim/2);
x_v = (2*u_v - a - b)/(b - a);
data = load('data_ums.lvm');
y_v = data(1:Ts:tsim/2,6);
y_v(y_v < 0) = 0;
y_v = qi(2)*y_v.^2;
data_v = iddata(y_v,x_v,Ts);
yms_v = data_v.y; x_v = data_v.u;
Nms_v = length(yms_v);
v_v = double(subs(f,u,x_v)); % Variable intermedia
y_hat_v = zeros(Nms_v,1); y_hat_v(1:delay+nb) = yms_v(1:delay+nb);
for k=1+nb+delay:Nms_v
    y_hat_v(k) = a_hat(1)*y_hat_v(k-1) + a_hat(2)*y_hat_v(k-2) +
b_hat(1)*v_v(k-1-delay);
    if y_hat_v(k)<0
        y_hat_v(k)=0;
    end
end
```

```
fit_v = (1-goodnessOfFit(y_hat_v,yms_v,'NRMSE'))*100;
mse_v = norm(yms_v-y_hat_v)^2/Nms_v;
% ============================================================
% 5. Gráfico de los Resultados del Ajuste
% ============================================================
% Gráfico del ajuste en Datos de Entrenamiento
figure
stairs(1:Nms,[zms z_ham],'LineWidth',1.5)
xlabel('k (muestras)',FontSize=12)
ylabel('Señal intermedia z_{ms}',FontSize=12)
legend('Proceso Real','Modelo Hammerstein',Location='northwest')
text(100,15,['Fit(%):',num2str(fit)],'EdgeColor','black','LineStyle','-')
grid minor
disp('Modelo Hammerstein');
fprintf('Fit: %.4f \nmse: %.4f\n',fit,mserror);
% Gráfico del ajuste en Datos de Validación
figure
stairs(1:Nms_v,[yms_v y_hat_v],'LineWidth',1.5)
xlabel('k (muestras)',FontSize=12)
ylabel('Señal intermedia z_{ms}',FontSize=12)
legend('Proceso Real','Modelo Hammerstein',Location='northwest')
text(39,24.2,['Fit(%):',num2str(fit_v)],'EdgeColor','black','LineStyle','-')
grid minor
disp('Modelo Hammerstein');
fprintf('Fit: %.4f \nmse: %.4f\n',fit_v,mse_v);
% ============================================================
% 6. No linealidad de entrada
% ============================================================
uu = -1:0.01:1;
vv = double(subs(f,u,uu));
plot(uu,vv,'LineWidth',1.5)
xlabel('Señal de entrada normalizada x','FontSize',12)
ylabel('Señal intermedia v','FontSize',12)
grid minor
```

Bondad del Ajuste del Modelo Hammerstein-Wiener:

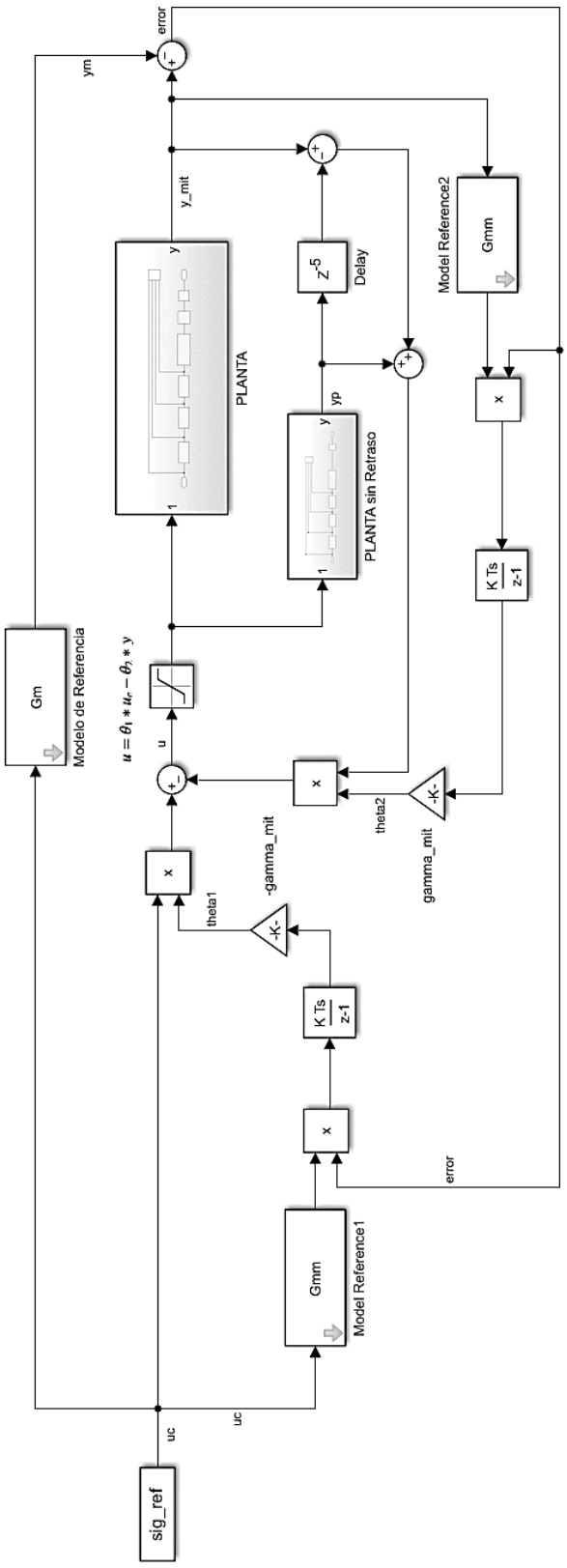
```
% ============================================================
% IDENTIFICACIÓN MODELO HAMMERSTEIN-WIENER - COMPLETO
% ============================================================
clc, clearvars, close all;
% ============================================================
% 1. Carga de Datos
% ============================================================
Ts = 4; % Tiemp ode muestreo
tsim = 8000; % Tiempo de simulación
% Data ums: Señal multinivel
ums = load('Signal_ub_ums.lvm');
ums = ums(1:Ts:tsim);
a = 0; b = 100;
x = (2*ums - a - b)/(b - a); % Normalización de la entrada [-1,1]
data = load('data_ub_ums.lvm');
yms = data(1:Ts:tsim,6);
yms(yms < 0) = 0;
% Coeficientes de polinomio z = f(y)
qi = load('qi.txt');
pi = [-5.8998214 160.39353 18.860866 -96.461057];
% Parámetros de la Función de activación
a_hat = [1.189, -0.2027]';
b_hat = 0.00406;
na = length(a_hat);
nb = 1;
Nms = length(ums); % Número de muestras
delay = 5;
v = zeros(Nms,1);
% ============================================================
% 2. Cálculo de la Bondad del Ajuste del Modelo Wiener
% ============================================================
for k = 1:Nms
    v(k) = pi(1);
    for i=1:3
        v(k) = v(k) + pi(i+1)*x(k)^i;
    end
end
z = zeros(Nms,1); z(1:nb+delay) = qi(2)*yms(1:nb+delay).^2;
for k = 1+nb+delay:Nms
    z(k) = a_hat(1)*z(k-1) + a_hat(2)*z(k-2) + b_hat*v(k-1-delay);
    if z(k)<0
        z(k)=0;
    end
end
y_ham_wie = sqrt(z/qi(2));
fit=(1-goodnessOfFit(y_ham_wie,yms,'NRMSE'))*100;
mserror = norm(yms-y_ham_wie).^2/Nms; % MSE
```

```
% ============================================================
% 3. Gráfico compartivo de los Modelos
% ============================================================
figure
plot(1:Nms,[yms y_ham_wie],'LineWidth',1.5)
xlabel('k (muestras)')
ylabel('Nivel Tanque 2 (cm)')
legend('Proceso Real','Modelo Hammerstein-Wiener',Location='northwest')
text(80,8,['Fit(%):',num2str(fit)],'EdgeColor','black','LineStyle','-')
grid minor
disp('Modelo Hammerstein');
fprintf('Fit: %.6f \nmse: %.6f',fit,mserror);
```

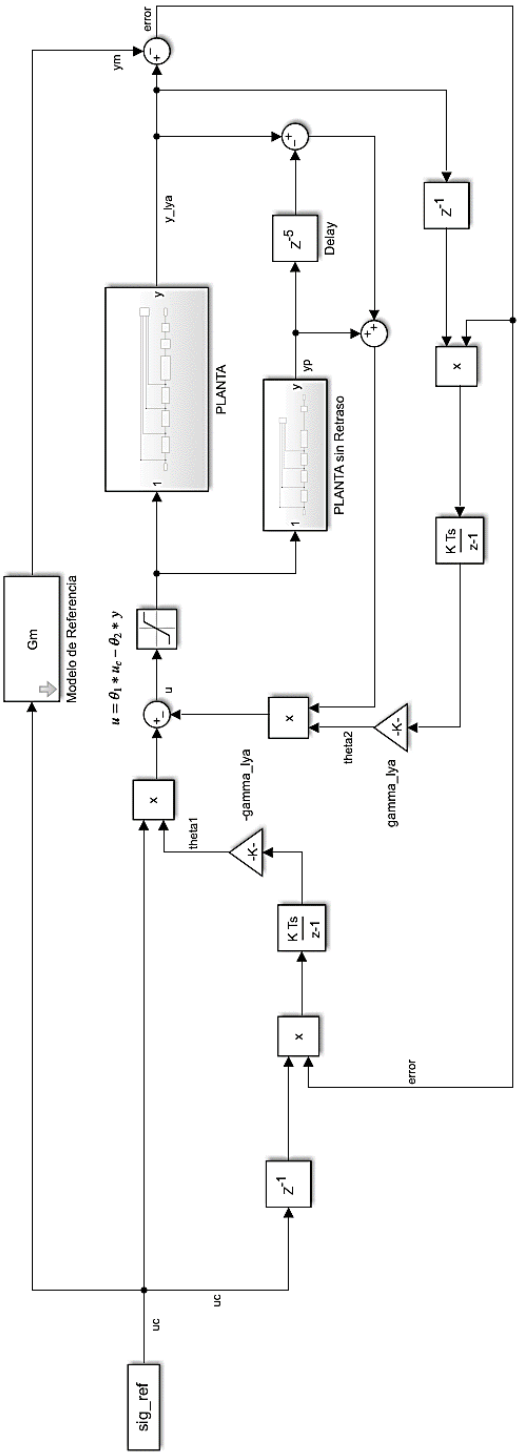
Anexo N°6 Código en MATLAB del Diseño de los Controladores MRAC

```
% ============================================================
% Diseño de Controlador MRAC: Regla MIT y Método Lyapunov
% ============================================================
clc, clearvars
% ============================================================
% 1. Modelo de la Planta
% ============================================================
Ts = 4;
a = [1.163, -0.1834]';
b = 0.002851;
disp('Modelo de la Planta')
Gz = tf([b 0],[1 -a(1) -a(2)],Ts)
% ============================================================
% 2. Modelo de Referencia
% ============================================================
ts = 150; % Tiempo de establecimiento
C = 0.85; % Coeficiente de amortiguamiento
wn = 3.912/(C*ts); % Frecuencia natural
Gms = tf(wn^2,[1 2*C*wn wn^2])
disp('Modelo de Referencia')
Gm = c2d(Gms,Ts)
[num, den] = tfdata(Gm,'v');
step(Gm)
Gmm = tf(b*[1 0],den,Ts)
% ============================================================
% 3. Definición de variables para Simulink
% ============================================================
% Parámetro de adaptación
gamma_mit = 1e-3 % Regla MIT
gamma_lya = 1e-4 % Método de Lyapunov
% Señal de referencia
h_init = 5;
periodo_ventana = 1500;
N = 5;
referencia = zeros(100,1);
for i=1:N
    level =  h_init;
    referencia = [referencia; level*ones(periodo_ventana,1);
zeros(periodo_ventana,1)];
end
tsim = length(referencia);
t = 1:tsim;
sig_ref = [t' referencia];
plot(t',referencia)
grid minor
```

Anexo N°7 Diagrama de bloques del Controlador MRAC con Regla MIT en Simulink



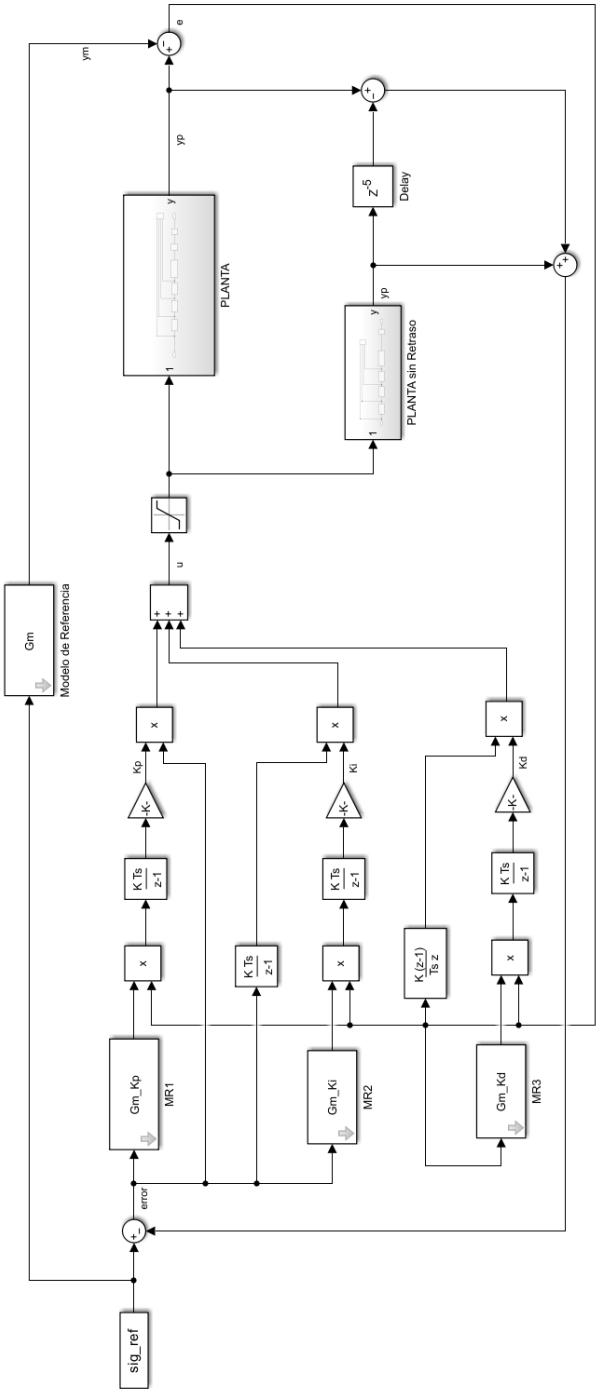
Anexo N°8 Diagrama de bloques del Controlador MRAC con Método Lyapunov en Simulink



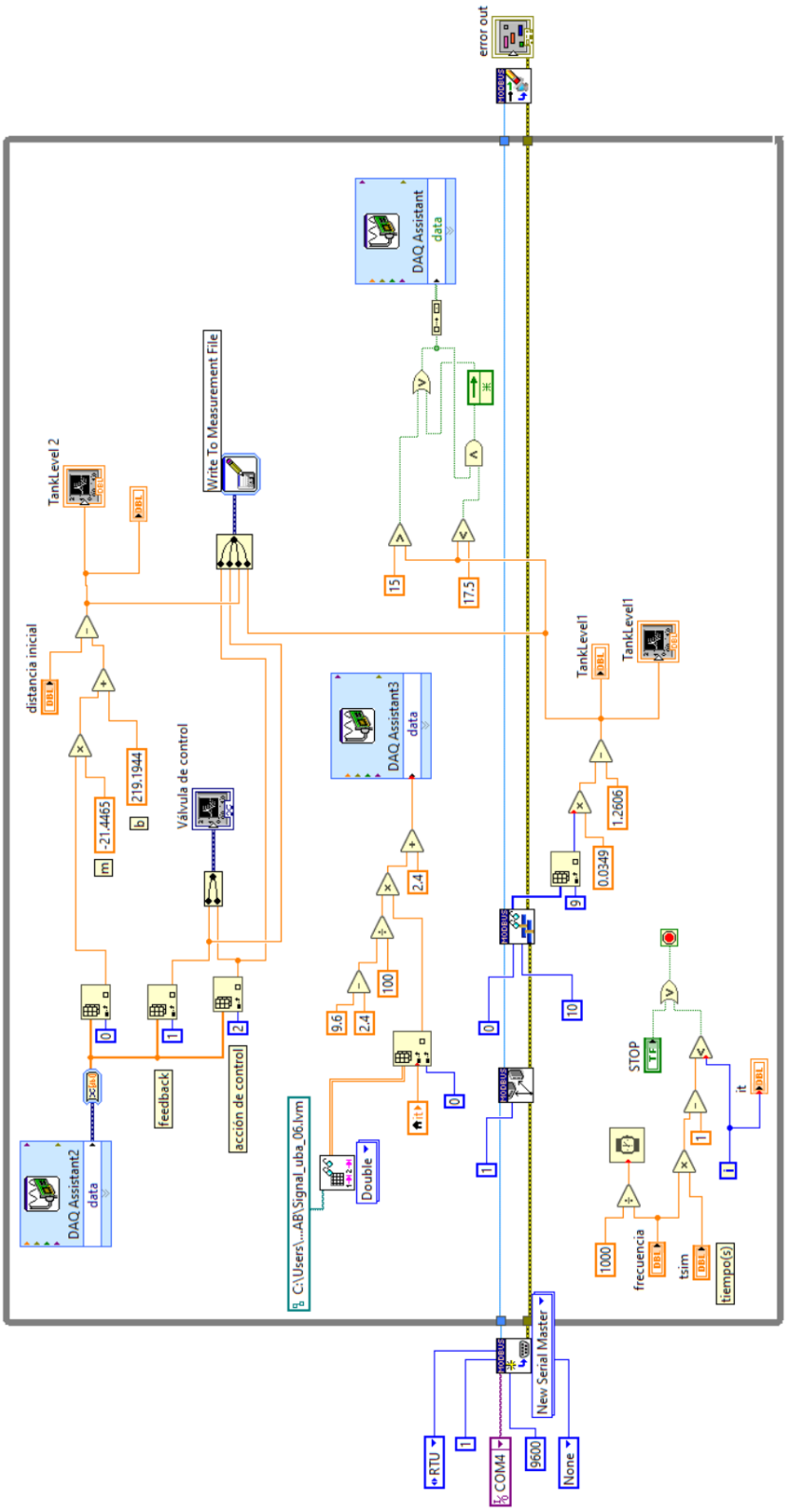
Anexo N°9 Código en MATLAB del Diseño del Controlador PID sintonizado con técnicas MRAC

```
%============================================================
% Diseño de Controlador PID sintonizado con técnicas MRAC
% ============================================================
clc, clearvars
% ============================================================
% 1. Modelo de la Planta
% ============================================================
Ts = 4;
a = [1.163, -0.1834]'; b = 0.002851;
disp('Modelo de la Planta')
Gz = tf([b 0],[1 -a(1) -a(2)],Ts)
% ============================================================
% 2. Modelo de Referencia
% ============================================================
ts = 150; % Tiempo de establecimiento
C = 0.85; % Coeficiente de amortiguamiento
wn = 3.912/(C*ts); % Frecuencia natural
Gms = tf(wn^2,[1 2*C*wn wn^2])
Gm = c2d(Gms,Ts)
[num, den] = tfdata(Gm,'v');
disp('Modelo de Referencia')
step(Gm)
Gm_Kp = tf(b*Ts*[1 -1 0],den,Ts)
Gm_Ki = tf(b*Ts^2*[1 0 0],den,Ts)
Gm_Kd = tf(b*[1 -2 1],den,Ts)
% ============================================================
% 3. Definición de variables para Simulink
% ============================================================
% Parámetro de adaptación
gamma_p = 6.6e-2;
gamma_i = 3e-7;
gamma_d = 1e-2;
% Señal de referencia
h_init = 5;
periodo_ventana = 1500;
N = 5;
referencia = zeros(100,1);
for i=1:N
    level =  h_init;
    referencia = [referencia; level*ones(periodo_ventana,1);
zeros(periodo_ventana,1)];
end
tsim = length(referencia);
t = 1:tsim;
sig_ref = [t' referencia];
plot(t',referencia), grid minor
```

Anexo N°10 Diagrama de bloques del Controlador PID sintonizado con técnicas MRAC



Anexo N°11 Diagrama de bloques del programa en LabVIEW para Identificación



Anexo N°12 Código en Arduino para Envío de Lectura de flujómetros por Protocolo Modbus RTU

```
#include <SimpleModbusSlave_DUE.h>
// ****** Definiendo pines de lectura ******
#define pinSignal_FM1 3
#define pinSignal_FM2 4
#define pinSignal_FM3 5
#define ID_SLAVE 2
#define RS485_TX_ENABLE_PIN 2
// ****** Declaración de variables ******
const float ConvFactor1=7.7475, ConvFactor2=7.7475, ConvFactor3=7.7475;
const int scalingFactor = 100;
unsigned long t1;
bool flag_modbus = false;
// ****** Registros del Esclavo ******
enum
{
  FLOW_RATE_1 = 1,
  FLOW_RATE_2 = 2,
  FLOW_RATE_3 = 3,
  HOLDING_REGS_SIZE // leave this one
};
unsigned int holdingRegs[HOLDING_REGS_SIZE]; // function 3 and 16 register
array
// ****** Definición de clase flujómetro ******
class Flowmeter {
  private:
    uint8_t pinSignal;
    volatile int numPulses;
    float convertionFactor;
    int frequency;
  public:
    // Constructor
    Flowmeter(uint8_t _pinSignal, float _convertionFactor):
pinSignal(_pinSignal), convertionFactor(_convertionFactor) {
      numPulses = 0;
      pinMode(pinSignal, INPUT);
    }
    void CounterPulses() {
      numPulses++;
    }
    int GetFrequency() {
      frequency = numPulses;
      return frequency;
    }
    void ResetNumPulses() {
```

```
      numPulses = 0;
    }
    float CalculateFlow() {
      return frequency / convertionFactor;
    }
};
// ****** Instanciando los flujómetros ******
Flowmeter FM1(pinSignal_FM1,ConvFactor1);
Flowmeter FM2(pinSignal_FM2,ConvFactor2);
Flowmeter FM3(pinSignal_FM3,ConvFactor3);
// ****** Funciones de interrupciones ******
void ISR1(){
  FM1.CounterPulses();
}
void ISR2(){
  FM2.CounterPulses();
}
void ISR3(){
  FM3.CounterPulses();
}
void setup(){
  Serial.begin(9600);
  // ****** Declaración de pines de interrupción ******
  attachInterrupt(digitalPinToInterrupt(pinSignal_FM1), ISR1, RISING);
  attachInterrupt(digitalPinToInterrupt(pinSignal_FM2), ISR2, RISING);
  attachInterrupt(digitalPinToInterrupt(pinSignal_FM3), ISR3, RISING);

  modbus_configure(&Serial1, 9600, ID_SLAVE, RS485_TX_ENABLE_PIN,
HOLDING_REGS_SIZE, holdingRegs);
  modbus_update_comms(9600, ID_SLAVE);
  t1 = millis();
}
void loop(){
  // ****** Lectura de Flujómetros YF-S201 ******
  if(millis()-t1>1000){
      noInterrupts();
      float frequency1 = FM1.GetFrequency();
      float frequency2 = FM2.GetFrequency();
      float frequency3 = FM3.GetFrequency();
      // Cálculo del flujo
      unsigned int flowRate1 = FM1.CalculateFlow()*scalingFactor;
      unsigned int flowRate2= FM2.CalculateFlow()*scalingFactor;
      unsigned int flowRate3= FM3.CalculateFlow()*scalingFactor;
      holdingRegs[FLOW_RATE_1] = flowRate1;
      holdingRegs[FLOW_RATE_2] = flowRate2;
      holdingRegs[FLOW_RATE_3] = flowRate3;
      FM1.ResetNumPulses();
      FM2.ResetNumPulses();
```

```
        FM3.ResetNumPulses();
        t1 = millis();
        interrupts();
    }
    if(!flag_modbus){
        flag_modbus = true;
        Serial.flush();
    }
    modbus_update();
}
```

USB-6211/6215

USB-6211/6215 Pinout

Figure A-2 shows the pinout of the USB-6211 and USB-6215.

For a detailed description of each signal, refer to the *I/O Connector Signal Descriptions* section of Chapter 3, *Connector and LED Information*.

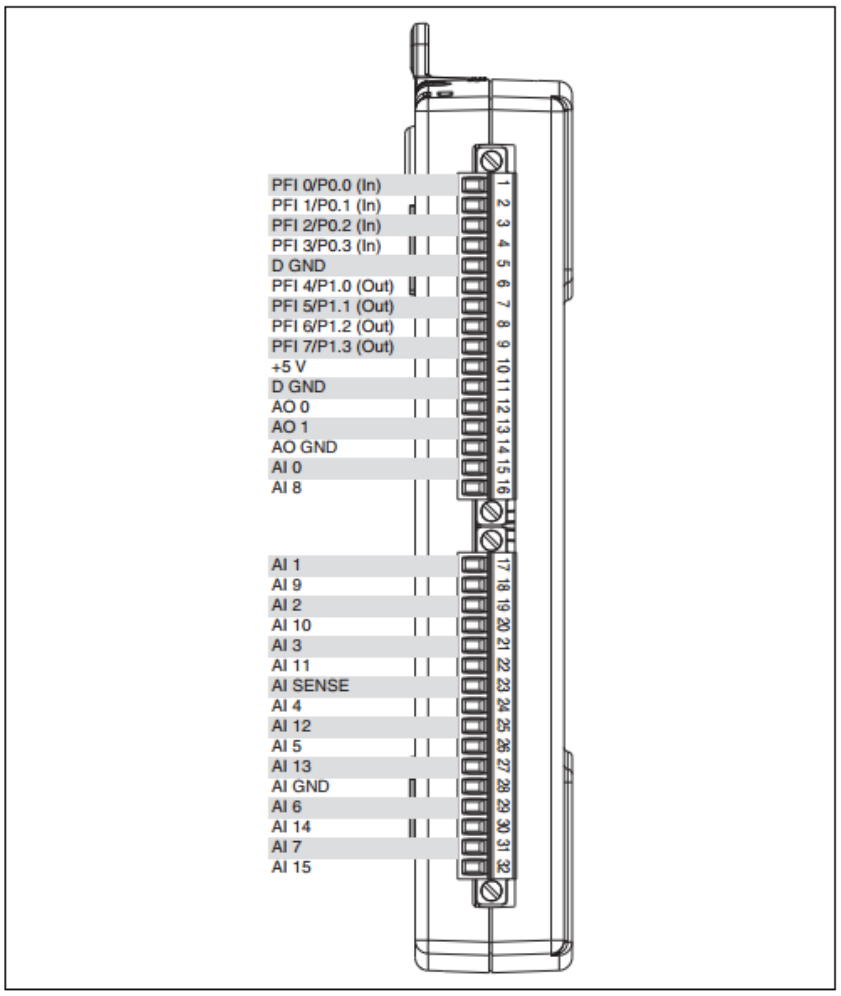


Figure A-2. USB-6211/6215 Pinout

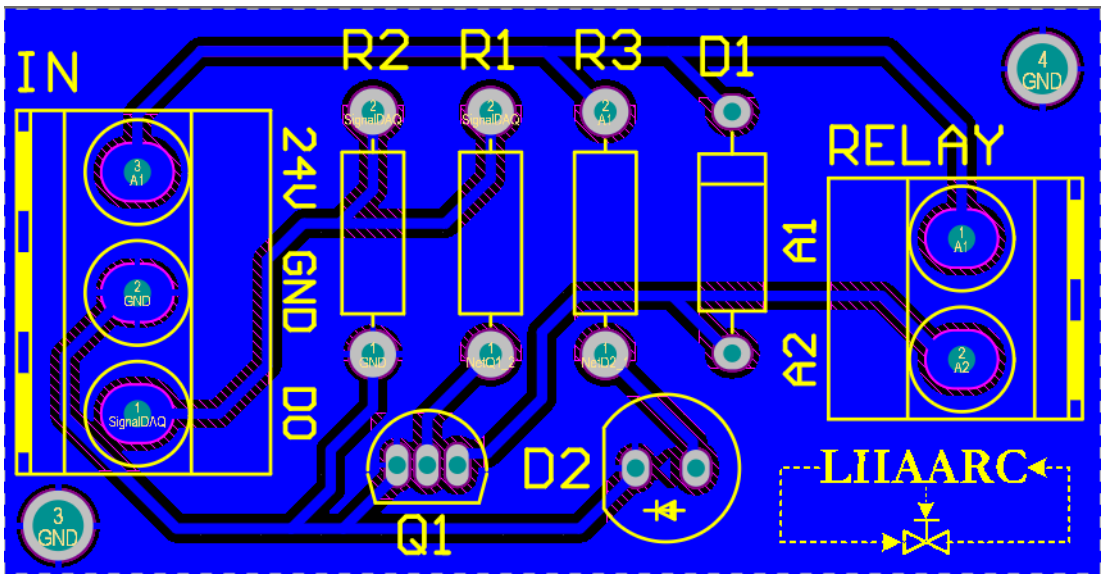
Table A-2. Default NI-DAQmx Counter/Timer Pins

Counter/Timer Signal	Default Terminal Number (Name)
CTR 0 SRC	1 (PFI 0)
CTR 0 GATE	2 (PFI 1)
CTR 0 AUX	1 (PFI 0)
CTR 0 OUT	6 (PFI 4)
CTR 0 A	1 (PFI 0)
CTR 0 Z	3 (PFI 2)
CTR 0 B	2 (PFI 1)
CTR 1 SRC	4 (PFI 3)
CTR 1 GATE	3 (PFI 2)
CTR 1 AUX	4 (PFI 3)
CTR 1 OUT	7 (PFI 5)
CTR 1 A	4 (PFI 3)
CTR 1 Z	2 (PFI 1)
CTR 1 B	3 (PFI 2)
FREQ OUT	8 (PFI 6)

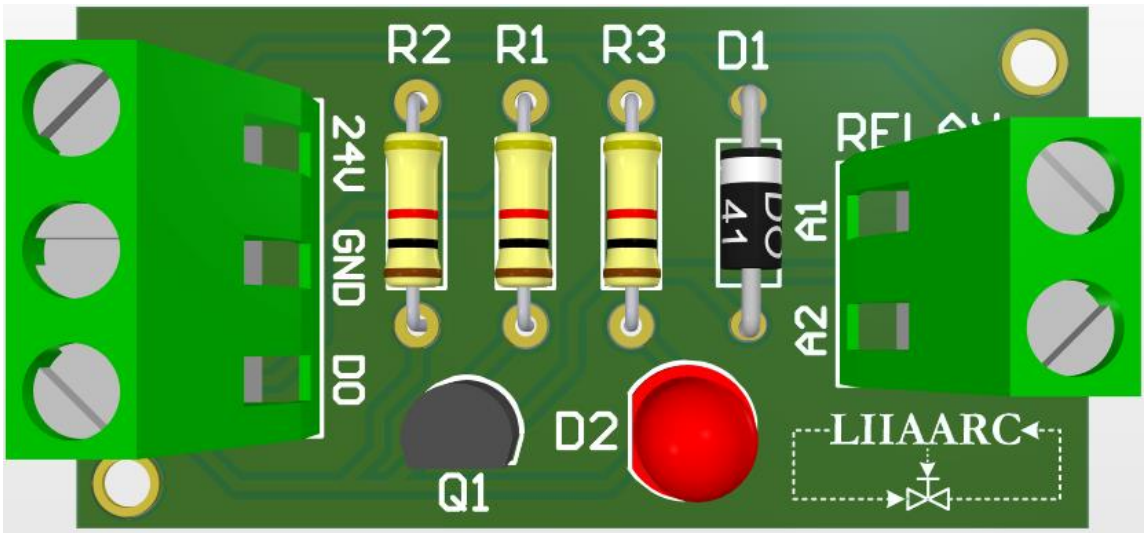
Note For more information about default NI-DAQmx counter inputs, refer to *Connecting Counter Signals* in the *NI-DAQmx Help* or the *LabVIEW Help* in version 8.0 or later.

Anexo N°14 Placas electrónicas elaboradas en Altium Designer

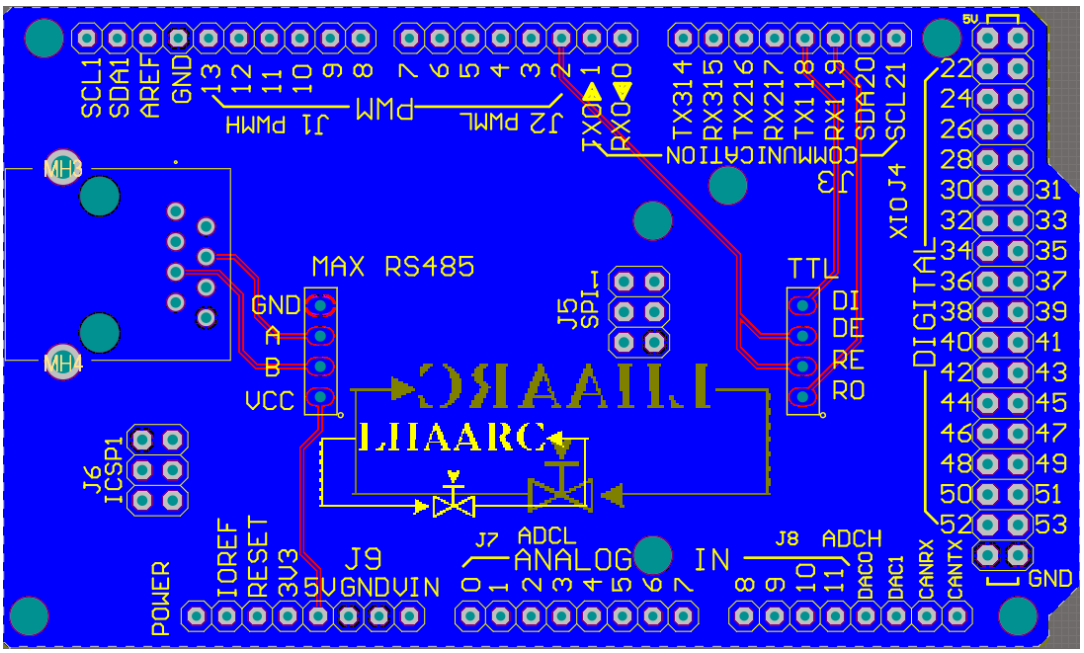
PCB de Tarjeta electrónica para activación de bomba de agua:



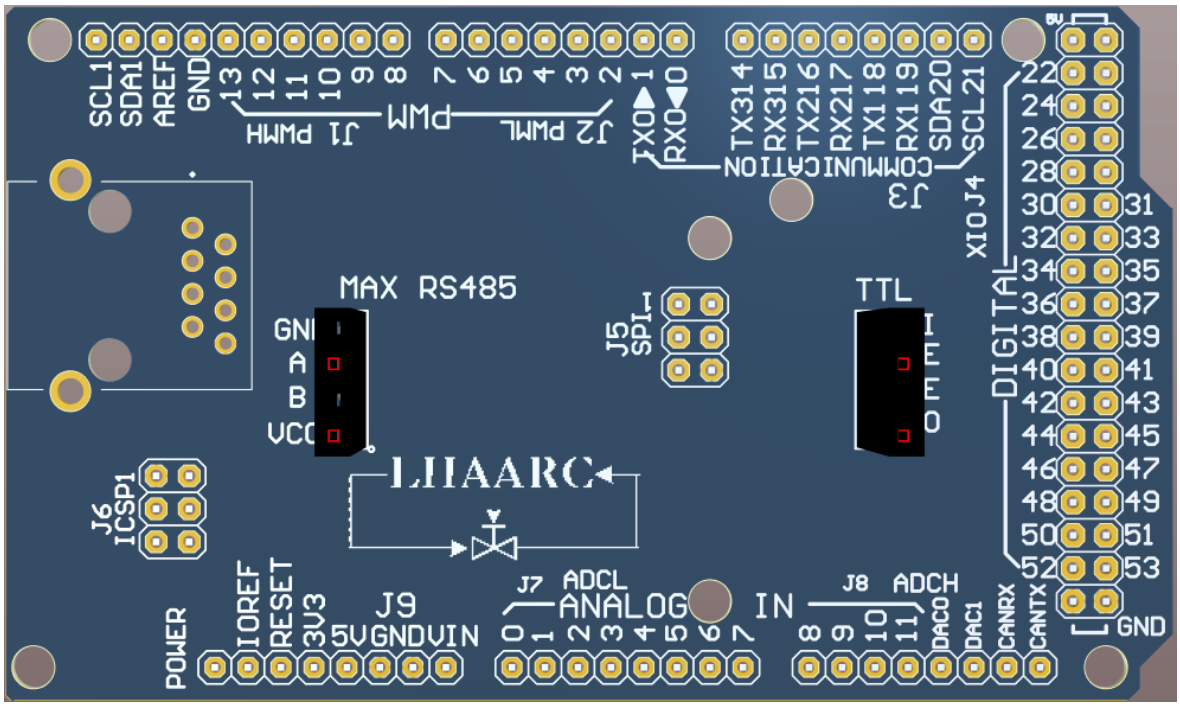
Diseño 3D de Tarjeta electrónica para activación de bomba de agua:



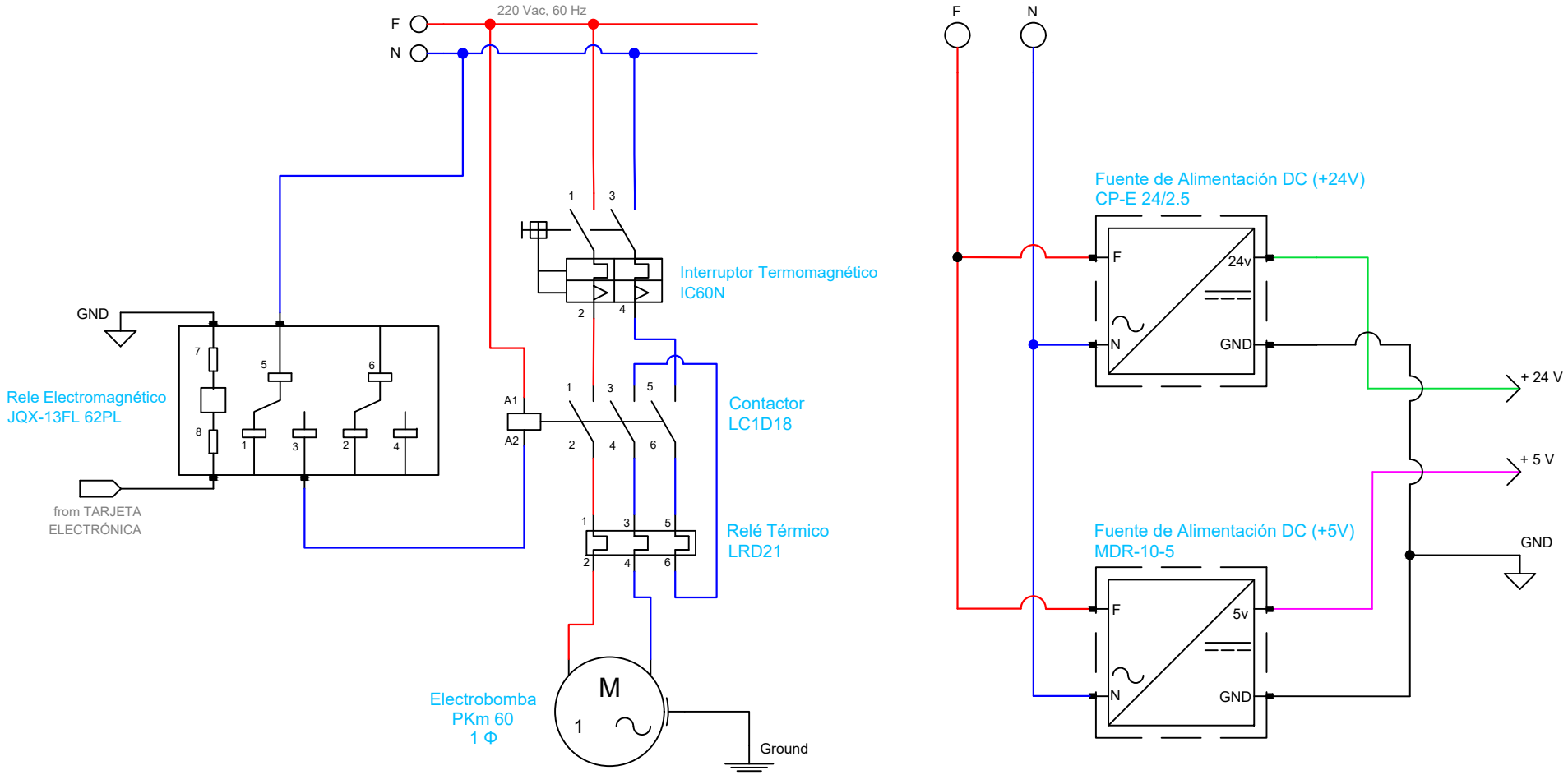
PCB de Shield de Arduino Due para comunicación Modbus:



Diseño 3D de Shield de Arduino Due para comunicación Modbus:



Anexo N°15 Diagrama eléctrico del conexionado del sistema completo



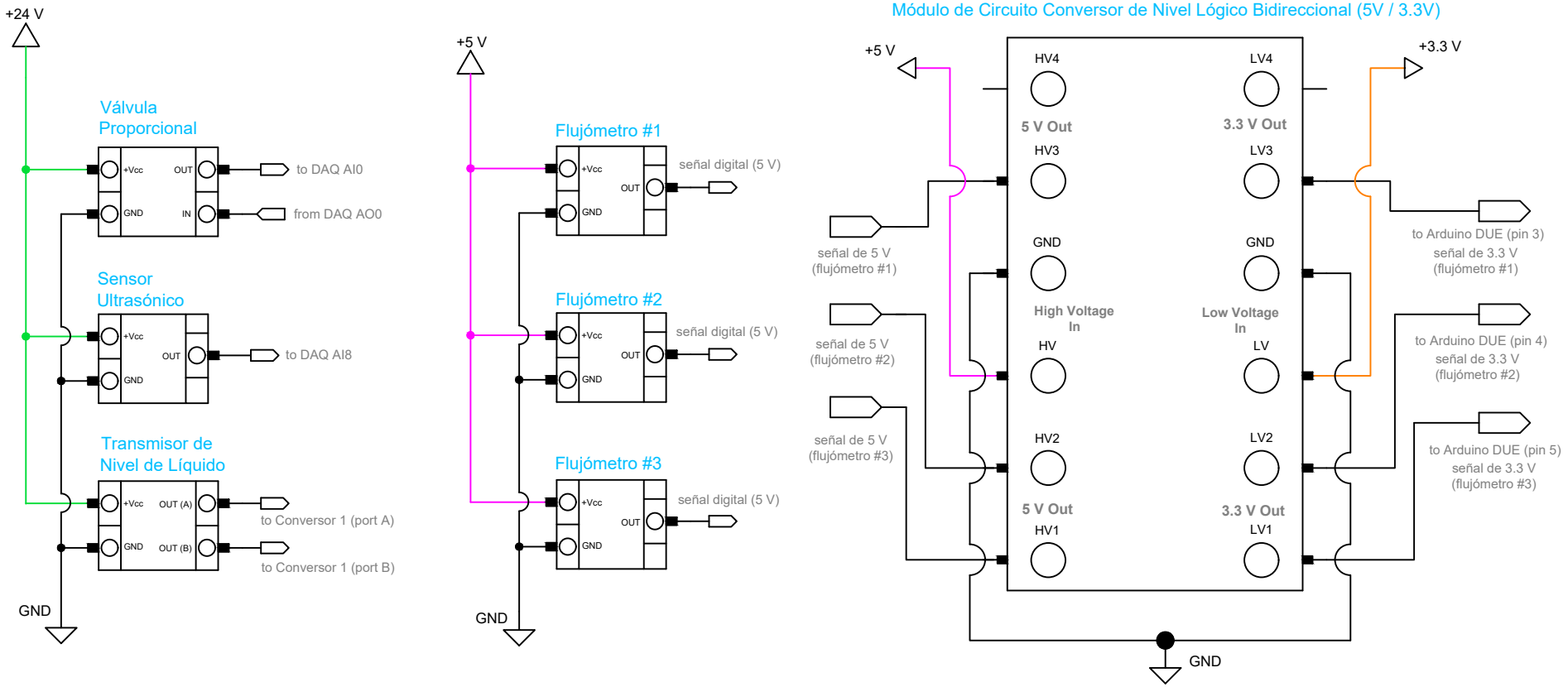
DIBUJO	Cipra Rodriguez	Marco Antonio	ESCALA 1:1	DIAGRAMA ELÉCTRICO DE PLANTA PILOTO DE TANQUES EN CASCADA
DISEÑO	Cipra Rodriguez	Marco Antonio		
APROBO	Rodriguez B.	Ricardo Raúl	FECHA 15/09/2024	
NORMA	IEC 60617			

UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

DPTO. ACAD. DE ING. MECATRÓNICA

PLANO N°	N°
SUSTITUYE A	1/4
SUSTITUIDO POR	



DIBUJO	Cipra Rodriguez	Marco Antonio	ESCALA 1:1
DISEÑO	Cipra Rodriguez	Marco Antonio	
APROBO	Rodriguez B.	Ricardo Raúl	FECHA 15/09/2024
NORMA	IEC 60617		

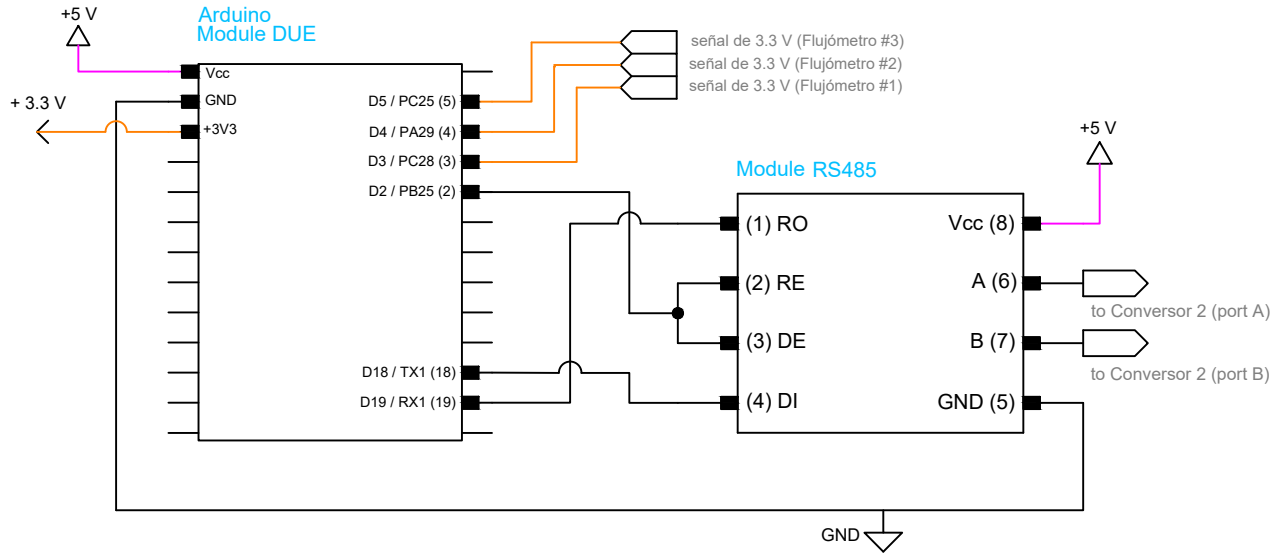
DIAGRAMA ELÉCTRICO DE PLANTA PILOTO DE TANQUES EN CASCADA

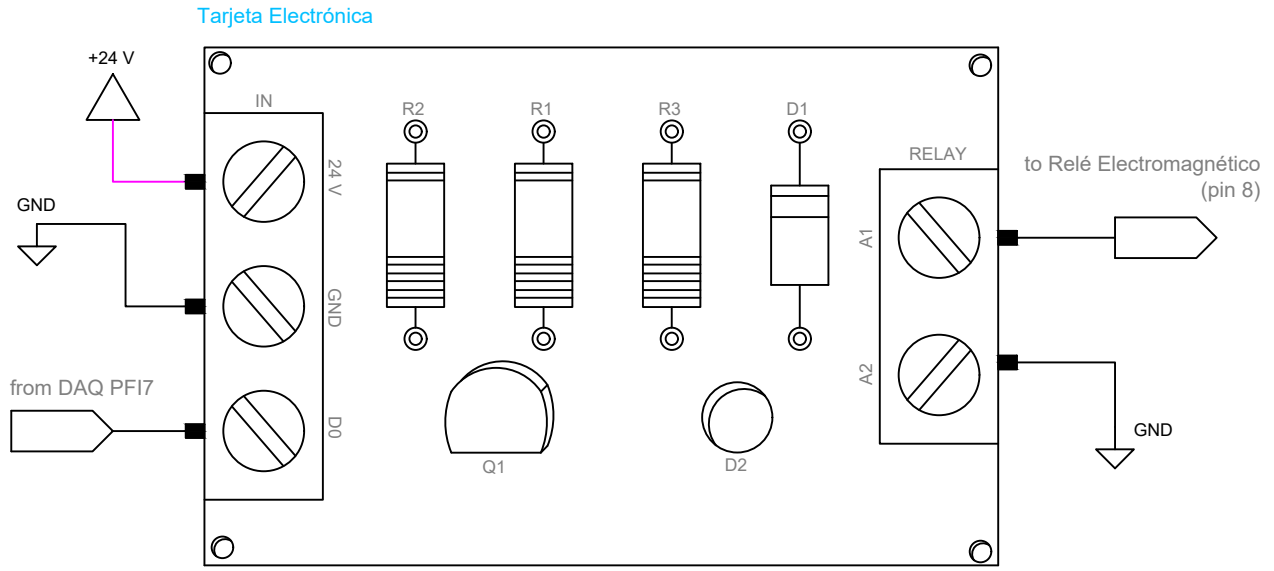
UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

DPTO. ACAD. DE ING. MECATRÓNICA

PLANO N°	N°
SUSTITUYE A	2/4
SUSTITUIDO POR	





DIBUJO	Cipra Rodriguez	Marco Antonio	ESCALA 1:1
DISEÑO	Cipra Rodriguez	Marco Antonio	
APROBO	Rodriguez B.	Ricardo Raúl	FECHA 15/09/2024
NORMA	IEC 60617		

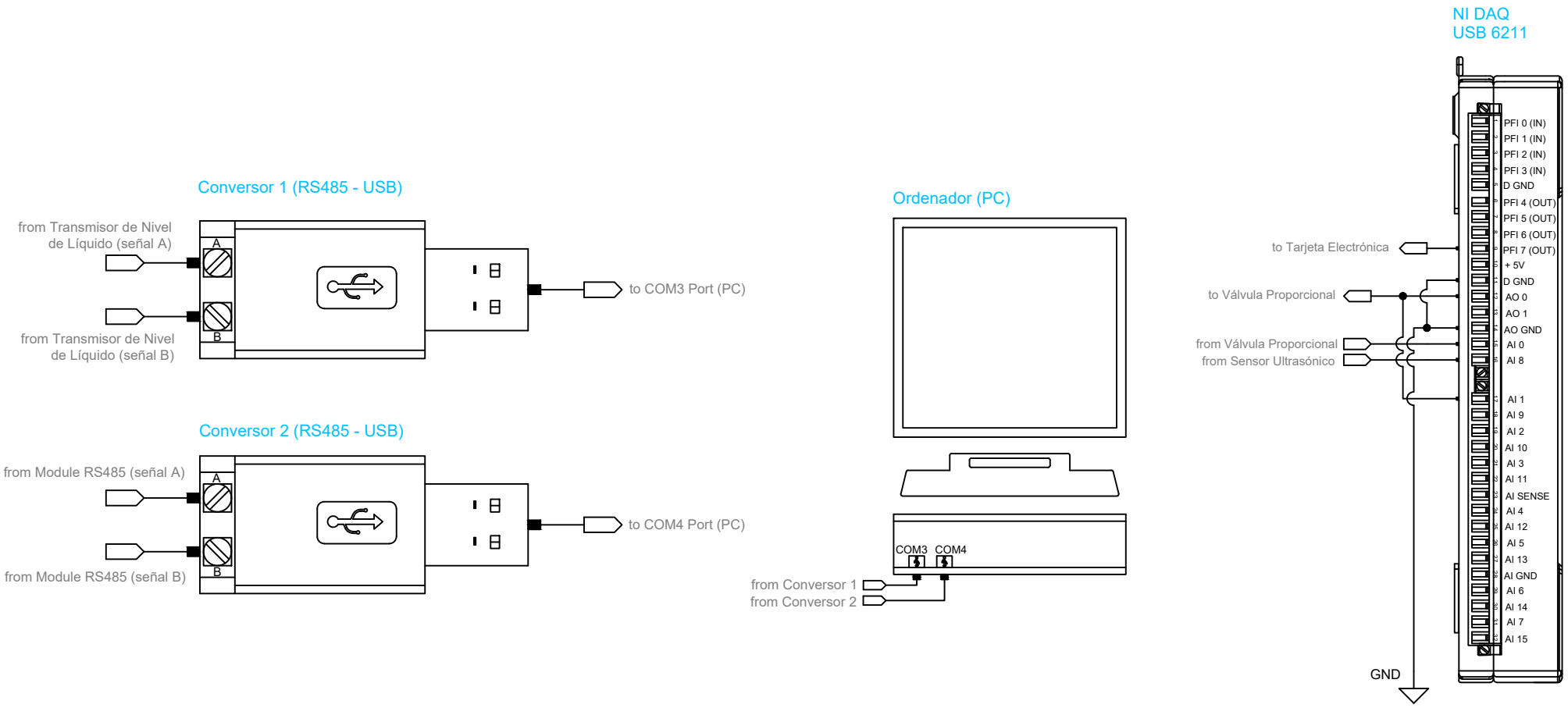
DIAGRAMA ELÉCTRICO DE PLANTA PILOTO DE TANQUES EN CASCADA

UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

DPTO. ACAD. DE ING. MECATRÓNICA

PLANO N°	N°
SUSTITUYE A	3/4
SUSTITUIDO POR	



DIBUJO	Cipra Rodriguez	Marco Antonio	ESCALA 1:1
DISEÑO	Cipra Rodriguez	Marco Antonio	
APROBO	Rodriguez B.	Ricardo Raúl	FECHA 15/09/2024
NORMA	IEC 60617		

DIAGRAMA ELÉCTRICO DE PLANTA PILOTO DE TANQUES EN CASCADA

UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

DPTO. ACAD. DE ING. MECATRÓNICA

PLANO N°	N°
SUSTITUYE A	4/4
SUSTITUIDO POR	