UNIVERSIDAD NACIONAL DEL CALLAO

ESCUELA DE POSGRADO

UNIDAD DE POSGRADO DE LA FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

"DESARROLLO DE UN SISTEMA PORTÁTIL PARA LA IDENTIFICACIÓN Y EMULACIÓN DE SISTEMAS SISO EN PLANTAS INDUSTRIALES"

SUSTENTACIÓN DE TESIS

PARA OPTAR EL GRADO

DE MAESTRO EN CIENCIAS DE LA ELECTRÓNICA

CON MENCIÓN EN CONTROL Y AUTOMATIZACIÓN

LEONARDO NIKOLAI VINCES RAMOS

2021

HOJA DE REFERENCIA DEL JURADO

MIEMBROS DEL JURADO

Dr.	**: CÉSAR AUGUSTO SANTOS MEJÍA**	**PRESIDENTE**
Dr.	**: ADAN ALMIRCAR TEJADA CABANILLAS**	**SECRETARIO**
Mg.	**: JORGE ELÍAS MOSCOSO SÁNCHEZ**	**MIEMBRO**
Msc.	**: RUSSELL CÓRDOVA RUIZ**	**MIEMBRO**
Dr.	**: JACOB ASTOCONDOR VILLAR**	**ASESOR**

Nº DE LIBRO	**: 01**
FOLIO	**: 104**
FECHA DE APROBACIÓN	**: 31 mayo 2021**
RESOLUCIÓN DIRECTORAL	**: 026-2021-DUPFIEE**

DEDICATORIA

A mis hijos, padres y especialmente a mi esposa.

AGRADECIMIENTO

Agradecer a Dios, a mi familia, a mis compañeros de trabajo de la Universidad Peruana de Ciencias Aplicadas, a mis profesores de la Universidad Nacional Callao y especialmente a mi Asesor el Dr. Jacob Astocondor, quienes en todo momento me animaron en la culminación de esta investigación.

INDICE

Tabla de Contenido

Tabla de figuras

Tabla de tablas

Tabla de Gráficos

RESUMEN

El presente trabajo de investigación comprende el desarrollo de un sistema portátil para la identificación y emulación de sistemas SISO, el proceso de identificación se realizó mediante la estrategia de lazo abierto haciendo uso del Método de Strejc, mediante este método se logró la identificación calculando la pendiente máxima y sus respectivos cruces con los ejes. Se identificó sistemas de tipo SISO hasta de orden 5. Para el proceso de emulación se hizo uso del método de aproximación de Tustin, para obtener un sistema de ecuaciones en diferencia, luego se envían los valores a un conversor Digital-Analógico y se emula la función de transferencia hasta de orden 5. Bajo esta configuración Hardware in-the-Loop (HIL), donde se interactúa con un sistema de control, como puede ser desde controladores stand alone hasta controladores de gama alta. Donde el controlador del proceso será un Controlador Industrial y como componente emulado serán los modelos matemáticos en sistemas continuos o discretos del modelo de la planta.

Palabras claves: Tustin, Control Digital, Hardware-in-the-Loop, PLC, identificación de sistemas, Strejc, emulador, SISO

RESUMO

O presente trabalho de pesquisa inclui o desenvolvimento de um sistema portátil para identificação e emulação de sistemas SISO, o processo de identificação foi realizado através da estratégia de malha aberta utilizando o Método Strejc, através deste método a identificação foi realizada pelo cálculo da inclinação máxima e sua respectivas intersecções com os eixos. Sistemas do tipo SISO foram identificados até a ordem 5. Para o processo de emulação, foi utilizado o método de aproximação de Tustin para obter um sistema de equações às diferenças, então os valores são enviados para um conversor Digital-Analógico e a função de transferência até a ordem 5 Sob essa configuração de Hardware in-the-Loop (HIL), onde ele interage com um sistema de controle, como de controladores independentes a controladores de última geração. Onde o controlador de processo será um Controlador Industrial e como componente emulado estarão os modelos matemáticos em sistemas contínuos ou discretos do modelo da planta.

Palavras-chave: Tustin, Controle Digital, Hardware-in-the-Loop, PLC, Identificação do Sistema, Strejc, Emulador, SISO

INTRODUCCIÓN

La presente investigación tiene como principal objetivo el desarrollo de un sistema embebido, el cual pueda realizar la identificación y posterior emulación de un sistema SISO hasta de orden 5, esta herramienta nos permitirá reducir considerablemente los tiempos de puesta en marcha de una planta, así como la reducción de tiempos para la identificación de los sistemas y los tiempos en la ingeniería para el diseño de los controladores, en la actualidad se deben realizar diferentes tipos de pruebas, las cuales pueden generar, daños, pérdidas y retrasos. Es en este punto donde no se cuenta con una herramienta portátil, que cumpla con los estándares industriales de las señales de entrada y salida, que no requiera de un software o aplicativo el cual se deba incurrir en un gasto por licenciamiento, es por eso la importancia de esta investigación.

Se pretende desarrollar un sistema el cual pueda realizar la identificación de un sistema de tipo SISO y posteriormente emular la función de transferencia de una planta, se trata de desarrollar un equipo embebido para el uso en distintas ramas de la ingeniería. Este tipo de herramienta es de gran utilidad para la rama de automatización industrial, por ejemplo: es una gran ventaja verificar el algoritmo de control, de esta forma se detectan fallas antes de implementar el control sobre la planta real evitando posibles daños en la misma.

I PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1 Identificación del problema

En el Perú, en la actualidad, el crecimiento económico ha ido de la mano con el crecimiento de la industria. Según los Boletines de la Demografía Empresarial en el Perú (INEI, 2019), realizados por el Instituto Nacional de Estadística e Informática, desde el primer trimestre del 2016 al tercer trimestre del año 2019 el número de empresas en el país ha ido desde 2 085 miles a 2 699 miles de empresas. Es decir, en casi 3 años se ha tenido un crecimiento de 614 miles de empresas. De estas empresas, un 26% representan la actividad económica de Explotación de minas y canteras, así como la de Industrias Manufactureras, Bebidas y Alimentos, entre otras.

En los procesos industriales, se tienen diferentes etapas al momento del diseño e implementación de un sistema de control, entre estas etapas podemos tener: diseño de la estrategia de control, los planos de instrumentación y de tuberías, instalación de los sensores, configuración de los mismos, montaje de los actuadores, implementación del algoritmo de control en los diferentes tipos de contralores industriales, luego de esto se tiene la puesta en marcha del proceso.

En este punto hay varias cosas que se deben tomar en cuenta, por ejemplo, revisión de las señales de campo, cableado de todos los instrumentos, verificación de correcto montaje de los instrumento y actuadores, revisión del programa de control, como se ve, se cuenta con tareas que son de trabajo de escritorio como otras que no. Otras requieren estar físicamente en el proceso, y es acá donde se tiene un problema, ¿se cuenta con el modelo matemático de la planta o del sistema a controlar?, y luego ¿cómo se prueba el algoritmo de control del proceso?

Al margen de contar con la disponibilidad del proceso para hacer las pruebas, se tiene otro problema, ¿cuánto tiempo? y ¿el costo de estas pruebas? para la verificación del algoritmo de control. Ahora, después de esta pregunta se tiene las siguientes situaciones, si se cuenta con el tiempo y los recursos para hacer las pruebas, ¿cómo sabemos que no se va a realizar una mala operación?, ¿se contará con los equipos suficientes para hacer recambio en el caso de una mala operación?, ¿al momento de realizar las pruebas como sabemos que una mala operación pueda causar daños o lesiones a los trabajadores?

Para la verificación de los algoritmos de control y del correcto funcionamiento del proceso industrial o de la planta, se requiere contar con todos los elementos instalados, contar con todos los recursos para realizar las pruebas, entonces, se tienen los siguientes escenarios, ejemplo, en la industria de bebidas y alimentos es muy sensible, una mala operación puede echar a perder todo el lote, también se cuenta con procesos que por su magnitud, sería muy costoso la verificación del algoritmo, en otros procesos no se cuenta con el tiempo suficiente para realizar las pruebas (en algunas ocasiones se cuenta con intervalos de 8 horas a 24 horas).

En tal sentido se requiere de un sistema embebido portátil, el cual permita identificar y emular el comportamiento del proceso, sin necesidad de realizar varias pruebas con la planta o en el proceso, con este sistema se pueden realizar todas las pruebas necesaria para el correcto funcionamiento del algoritmo de control, si el algoritmo de control no funciona de forma correcta, no estaría en riesgo ni los componentes de la planta o proceso, ni tampoco habría daños a nivel de operarios y/o trabajadores del sistema.

Entonces, de aquí nace la idea del desarrollo de un sistema portátil para la identificación y emulación que sea capaz de recrear el comportamiento un sistema SISO. Este dispositivo tendría como finalidad la realización de ensayos industriales basados en el desarrollo de un sistema embebido, sin la

necesidad de tener costos asociados como el uso de una laptop, software o licencias de programas, y sin la necesidad de disponer de las plantas reales. Se puede inferir en este punto que es crucial el desarrollo de un sistema que permita obtener la identificación de la planta y emular su comportamiento para los fines requeridos.

1.2 Formulación del problema

1.2.1 Problema general

¿Es posible desarrollar un sistema portátil para la identificación y emulación de procesos industriales de tipo SISO hasta Orden 5, mediante el uso de herramienta de tipo open source (Software y Hardware) para plantas industriales?

1.2.2 Problemas específicos

¿Qué consideraciones se deberían tener el diseño de las interfaces electrónicas de entrada y salida para la identificación de un sistema de tipo SISO hasta orden 5 para lograr un sistema embebido?

¿Cuáles serían las condiciones tecnológicas de hardware y de software de tipo open source para el desarrollo de los algoritmos de emulación en un sistema embebido de tipo SISO hasta orden 5 para lograr un sistema embebido?.

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar un sistema portátil con herramientas de tipo open software y hardware, que permita la identificación y emulación de un sistema tipo SISO hasta de orden 5 para plantas industriales.

1.3.2 Objetivos específicos

a) Desarrollar el algoritmo para la identificación de sistemas de tipo SISO hasta orden 5 haciendo uso de herramientas de tipo open source para lograr un sistema embebido.

b) Desarrollar el algoritmo para la emulación de sistemas de tipo SISO hasta 5 haciendo uso de herramientas de tipo open source para lograr un sistema embebido.Desarrollar el algoritmo para la identificación de sistemas de tipo SISO hasta orden 5 haciendo uso de herramientas de tipo open source para lograr un sistema embebido.

1.4 Justificación

1.4.1 Teórico

En la presente investigación aborda los campos de estudio como: teoría de control clásico, control digital, identificación de sistemas, sistemas embebidos, programación de controladores industriales, diseño de circuitos electrónicos y programación de interfaces graficas en open source.

1.4.2 Tecnológica

Al momento de realizar la investigación, se tomará en cuenta las restricciones propias de la tecnología actual, como, por ejemplo, el controlador (sistema embebido), las interfaces gráficas y de visualización de resultados, componentes electrónicos a emplearse, las últimas versiones de las herramientas de diseño y programación.

1.4.3 Económica

Se trata del desarrollo de un sistema el cual cuenta con la integración de diversas tecnologías, adicionalmente se trata del uso de sistemas los cuales se encuentran muy alcance de los investigadores, no hace uso de equipos computacionales de alto rendimiento, como computadoras industriales y/o laptops, no hace falta de uso de software licenciandos, los cuales siempre es tos han sido un limitante para el desarrollo de este tipo de equipo.

1.4.4 Social

Con el desarrollo de esta investigación, dará como resultado un equipo el cual nos permite realizar la identificación de un sistema de tipo SISO y su posterior emulación, sin la necesidad de exponer a los trabajadores de la planta a riesgos innecesarios.

II MARCO TEÓRICO

2.1 Antecedentes

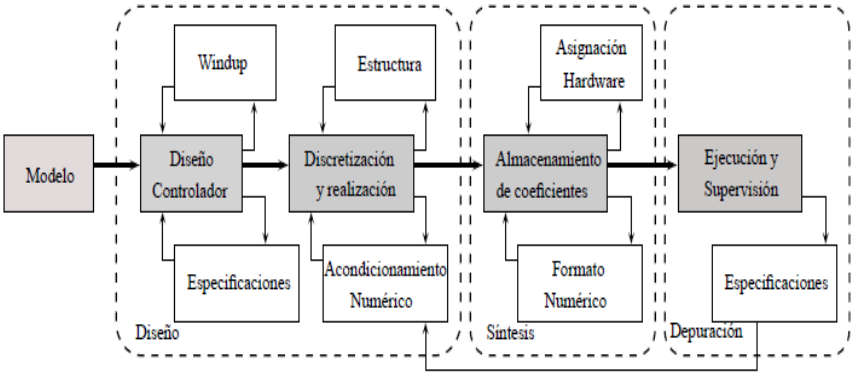
En esta sección de la investigación se realizó una búsqueda en diferentes bases científicas tales como: bases científicas internacionales donde se menciona 04 tesis de maestrías en temas relacionados, luego también se menciona 04 artículos científicos. Los cuáles serán mencionados y en las siguientes secciones de la investigación.

2.1.1 Bases Científicas Internacionales

Se realizó una búsqueda de diferentes tesis, artículos científicos indizados y trabajos de final de carrera relacionados con la investigación, a continuación, el detalle:

En la Tesis de Maestría de Fajardo (Juan Pablo, 2010) se basa en el desarrollo de controladores lineales, haciendo uso de herramientas computacionales como Matlab ® y Python, así mismo, el hardware utilizado es un ARMv7 y realiza un acondicionamiento numérico para evitar un posible desborde de la pila en el caso que se encuentre en una no linealidad. Sin embargo, en esta investigación solo aborda los temas de implementación en hardware y el desarrollo de controladores multivariables, pero no realiza la identificación y posterior emulación.

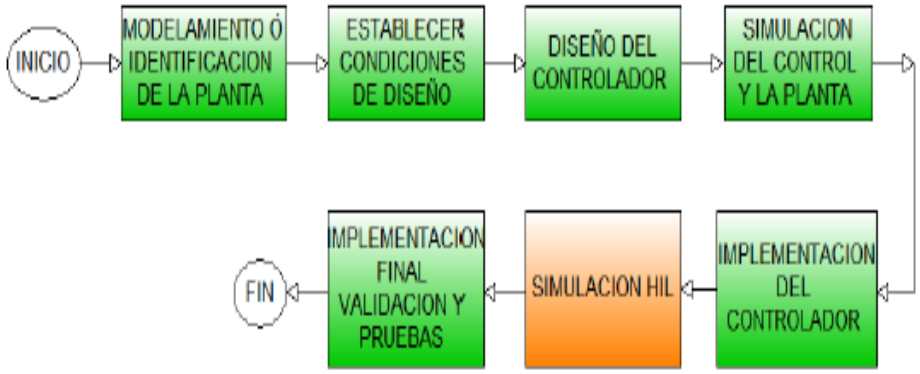
Figura II-1 Diagrama de Bloques de la solución 1



Fuente: (Juan Pablo, 2010) Tesis Desarrollo de un sistema para la implementación de controladores lineales multivariables

En la tesis de Maestría de Martínez et al (Juan Carlos, Martínez Quintero; Jaime Eduardo, Andrade Ramírez, 2013) desarrollan una investigación en base a un modelo analógico implementado con OPAMPS de un sistema de segundo orden, y en base a este modelo realiza el diseño del controlador, haciendo uso de herramientas que son licenciadas como IDENT, Matlab® y LabVIEW, para luego implementar el controlador en un sistema FPGA. No obstante, carece de un algoritmo o método para la identificación, y para la simulación solo hace uso de herramientas de software licenciado, no tiene una herramienta para la emulación del sistema con open source y solo trata de sistemas de segundo de orden.

Figura II-2 Diagrama de Bloques de la solución 2

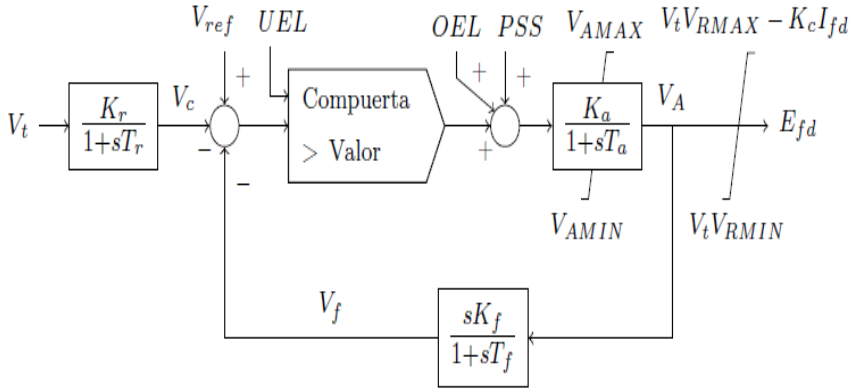


Fuente (Juan Carlos, Martínez Quintero; Jaime Eduardo, Andrade Ramírez, 2013) : Tesis Implementación de controladores en sistemas retroalimentados usando electrónica embebida y simulación hardware in the loop

En la tesis de Maestría de Biera (Andrés Ricardo, 2010) implementa una metodología del PEM (Método del error de predicción) para la identificación del sistema, la cual tiene como principal característica que al final todas las perturbaciones del

sistema son ruido blanco filtrado, entonces para una correcto método de identificación dependemos de muchas configuraciones como ARX, ARMAX, BJ, OE, etc; lo que hace complejo el método de identificación y requiere de varias pruebas y no trata el tema de la emulación. En esta tesis hace uso de herramientas de ingeniería licenciadas como Matlab ® - Simulink.

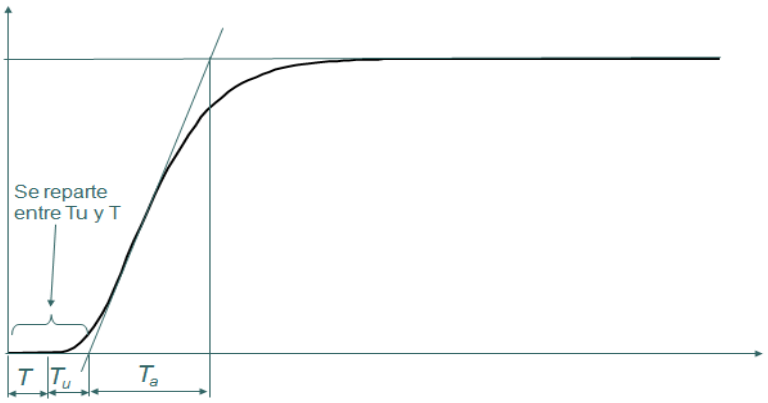
Figura II-3 Diagrama de Bloques de la solución 3



Fuente (Andrés Ricardo, 2010) : Metodología para la identificación del sistema de excitación de un generador eléctrico de potencia para propósitos de control.

En la publicación de Martínez et al (Angel, Martínez Bueno; Jorge, Pomares, 2011) realizan una investigación acerca de métodos de identificación experimental de sistemas, en el documento se hace referencia a métodos de identificación de diversos sistemas (Orden 1, Orden 2 y de polos múltiples) por el método de Strejc, se cuenta con los fundamentos matemáticos, no obstante la publicación es de mucho interés, está fue validada con Matlab ®. Por otro lado, no se trata de un sistema embebido, no hace uso de herramientas open source ni open hardware. Tampoco realizar la emulación del sistema.

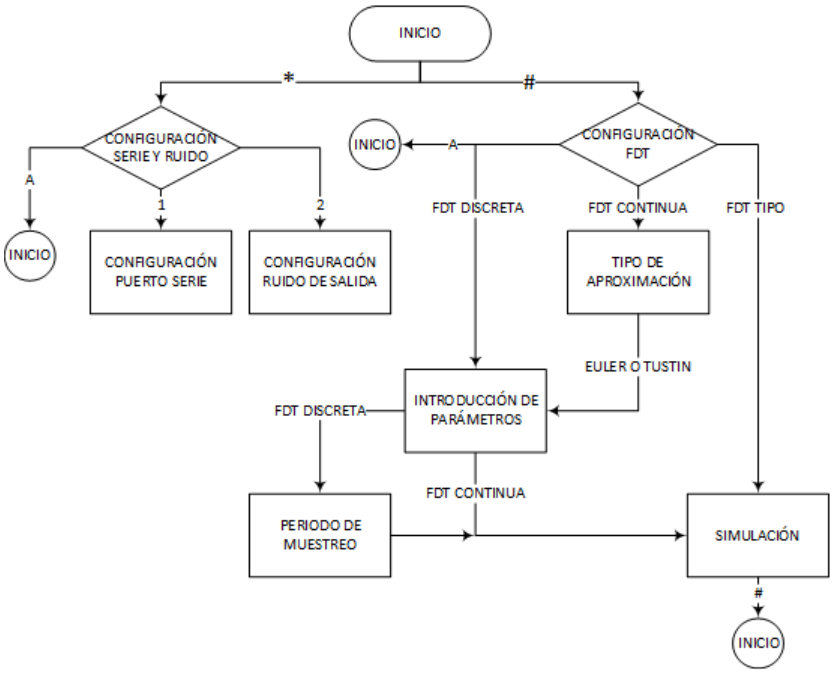
Figura II-4 Método Gráfico de Strejc Solución 4



Fuente (Angel, Martínez Bueno; Jorge, Pomares, 2011): Método Gráfico

En el trabajo de Fin de Grado de Velo Sánchez (Esteban, 2014) realiza un trabajo de investigación donde propone el desarrollo de un emulador de orden 4 para un sistema de tipo SISO, si bien es cierto que hace uso de hardware y software que se consideran de tipo open source y es implementado en un sistema embebido, para parte gráfica solo cuenta en un menú para el ingreso de los coeficientes y configuración de algunos parámetros. Sin embargo, solo realiza la emulación del sistema, no permite la visualización de la función de transferencia y no implementa un sistema identificación.

Figura II-5 Diagrama de Bloques de la solución 5



Fuente (Esteban, 2014) : Desarrollo y testeo de un emulador de plantas industriales basado en arduino.

2.1.2 Bases Científicas Nacionales

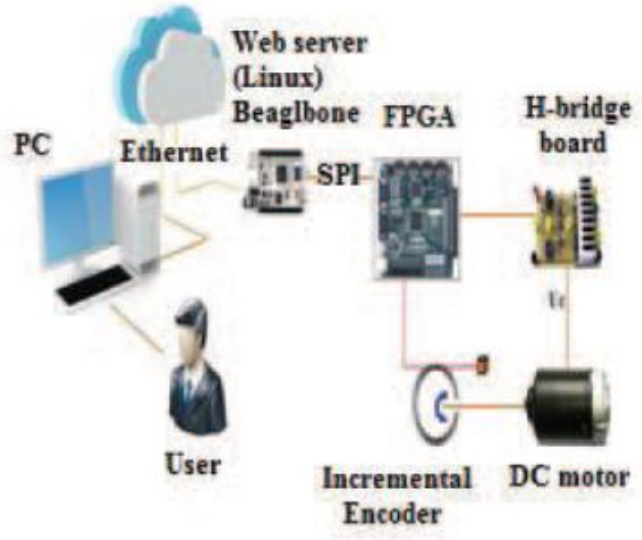
Se realizo una búsqueda de publicaciones, tesis y/o documentos referidos a la investigación, donde se determinó que hasta a fecha de la publicación de este trabajo de investigación no se cuenta con referencia nacionales de trabajos similares.

2.1.3 Publicaciones Científicas.

Después de una búsqueda y teniendo en cuenta el estado del arte se tomó como referencia las publicaciones con 5 años de antigüedad de temas relacionados con el objetivo de esta investigación, mencionamos algunas de ellas:

En el artículo científico de I. JAZIRI et al (I.JAZIRI, L.CHAARABI, K.JELASSI, 2015), se trata de una investigación de un sistema de un control remoto de un motor de CC haciendo uso de Linux y FPGA. Se ha desarrollado una aplicación en el sistema operativo Linux en una Single Board Computer (Beaglebone), donde se logra la comunicación a través del protocolo SPI con el FPGA, en una arquitectura maestro-esclavo. En la investigación se ha desarrollado una interfaz para la visualización y control. Lo más resaltante es el uso de nuevas tecnologías como los SBC's y cual es un sistema embebido, tiene un sistema operativo, hace uso de diferentes interfaces de comunicación, sin embargo, no realiza la identificación del sistema, ni realiza la emulación del sistema.

Figura II-6 Diagrama de Bloques de I. Jazari

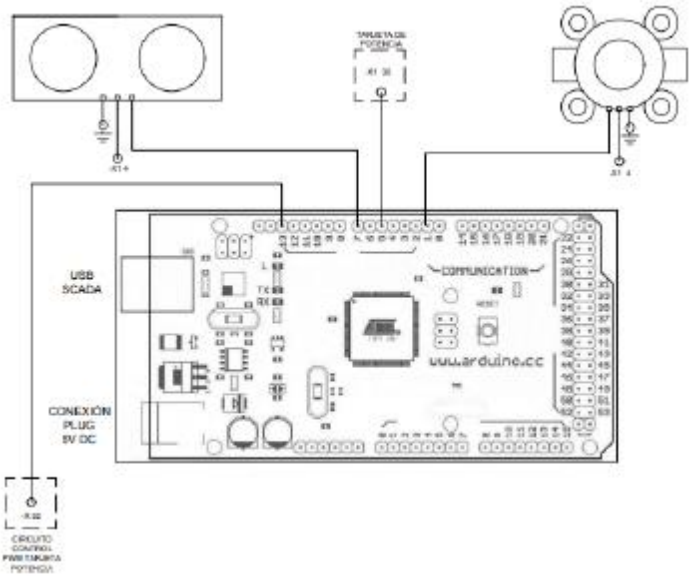


Fuente (I.JAZIRI, L.CHAARABI, K.JELASSI, 2015): A remote DC motor control using Embedded
Linux and FPGA

En el artículo científico de Vega et al (Nino Vega; Pablo Parra; Luis Córdova; Joselyne Andramuño; Johnny Álvarez, 2018), se trata de una investigación un algoritmo de control en cascada para el control de las variables caudal y nivel, el cual ha sido implementado con elementos convencionales como los PLC y

variadores de velocidad, en su reemplazo se hizo uso de una electrónica de bajo costo como los sistemas embebidos, para luego evaluar su desempeño con pruebas experimentales, con la intención de generar una alternativa. En la solución desarrollada se inicia con un método para controlar una planta, realiza un método para obtención del modelo dinámico, y realiza el desarrollo de los controladores. Sin embargo, el modelamiento del sistema se ha realizado haciendo uso de la herramienta Matlab ®, la cual necesita de una licencia y no se trata de una solución de tipo open source, donde calcula los parámetros del controlador, pero no logra hacer la emulación del sistema.

Figura II-7 Diagrama de Bloques Vega et al

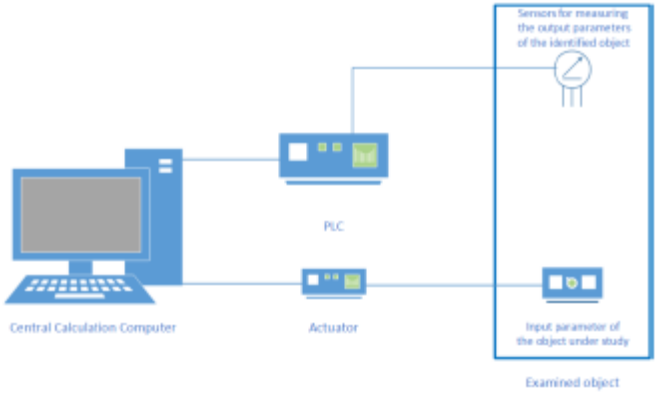


Fuente (Nino Vega; Pablo Parra; Luis Córdova; Joselyne Andramuño; Johnny Álvarez, 2018): Cascade Control Algorithm developed with
Embedded Systems

En el artículo científico de Fatuev et al (Fatuev, Victor A.; Mishin Anton A., 2019), el objetivo de esta investigación es obtener un modelo óptimo de regresión dinámica, el criterio utilizado fue el de

D-óptima, que es una de los más utilizados para modelos de regresión dinámica parametrizados no lineales, el procedimiento utilizado le permite explorar una amplia clase de sistemas dinámicos lineales y no lineales descritos por entrada-salida modelos y espacio estado. Sin embargo, hace uso de un computador para realizar los cálculos, por lo tanto, no se trata de un sistema embebido, por otro lado, identifica el sistema, pero no logra hacer la emulación.

Figura II-8 Diagrama de Bloques de Fatuev



Fuente (Fatuev, Victor A.; Mishin Anton A., 2019): Realization of Optimal Identification Tasks for Dynamic Systems in Real Time Scale

En el artículo de Kumar Verma et al (Santosh Kumar Verma and Shyam Krishna Nagar, 2016) se presenta una aproximación de los sistemas de orden fraccional en modelos de espacio estado de segundo orden. La operación se realiza en dos pasos; en el primer paso, la aproximación de orden entero del sistema de orden fraccional se realiza usando el algoritmo de aproximación y luego, el sistema aproximado se realiza en forma de espacio estado usando Matlab ® y la dimensión del modelo de orden entero resultante es ser reducido usando eficiente realización equilibrada basada. No obstante que el sistema tiene un buen rendimiento y logra un método de identificación, lo hace mediante

herramientas licenciadas y no implementa la emulación de la identificación.

2.2 **Motivación**

Después de realizar una búsqueda en diferentes medios como tesis, artículos científicos, publicaciones o trabajos de fin de carrera, vemos que no hay trabajos de investigación similares, es por eso la importancia de esta investigación.

2.3 Conceptual

Para el desarrollo de esta investigación se hará uso de diferentes técnicas, metodologías, recursos tecnológicos, entre ellos tenemos:

2.3.1 Adquisición de datos

Según National Instruments (Instruments, 2020) La adquisición de datos es un proceso de medir una variable de tipo eléctrica o física como tensión, corriente, humedad, velocidad, temperatura, presión u otro tipo de variables que pueda ser sensada y luego ser enviadas a una PC para su posterior análisis. Al tener este atributo de usar la potencia de las PC’s, el manejo de estas variables es mucho más fácil el procesamiento, análisis y visualización.

Según el fabricante MC Measurement Computing (Computing M. , 2020) define a este dispositivo como el proceso de digitalización de datos del mundo que nos rodea para que pueda visualizarse, analizarse y almacenarse en una computadora. Los sistemas modernos de adquisición de datos pueden incluir la adición de análisis de datos y software de informes, conectividad de red y opciones de control remoto y monitoreo. Para el caso de esta investigación se tomó en consideración la siguiente tabla comparativa entre diferentes tipos de adquisición de datos.

Tabla II-1 Tabla Comparativa de sistemas de adquisición de datos

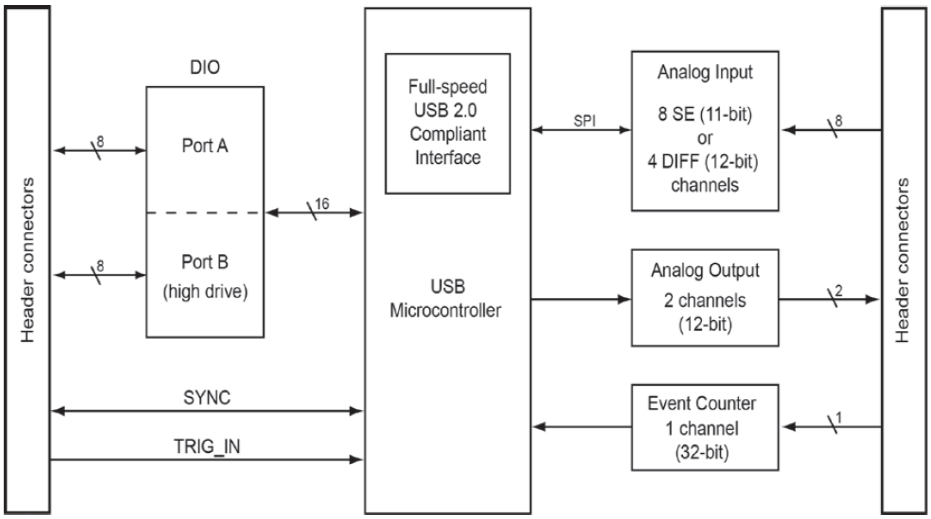
Descripción	NI DAQ USB 6009	USB DAQ 1208FS OEM	Pine A 64	DI-2108
Fabricante	National Instruments	Measurement Computing	LabJack	Dataq
Entradas Analógicas	08 canales en modo común. 04 canales en modo diferencial	08 canales en modo común. 04 canales en modo diferencial	04 canales en modo diferencial	08 canales en modo común. 04 canales en modo diferencial
Resolución	14 bits en modo común 13 bits en modo diferencial	14 bits en modo común 14 bits en modo diferencial	12 bits en modo diferencial	16 bits en modo común 16 bits en modo diferencial
Velocidad de muestreo	48Ks/S	50Ks/S	50Ks/S	200Ks/S
Salidas Analógicas	02 canales	02 canales	02 canales	01 canal
Resolución	12 bits	12 bits	12 bits	12 bits
Velocidad de muestreo	150s/S	10Ks/S	10Ks/S	10Ks/S
Interface	USB 2.0	USB 2.0	USB 2.0	USB 2.0
Sistema Operativo	Windows/Linux	Windows/Linux/ Android/	Windows/Linux/	Windows/Linux/
Precio US$D	245.00	290.00	209.00	378.00

Fuente: Elaboración Propia

Después de haber realizado una búsqueda de diferentes fabricantes de tarjetas de adquisición de datos y según la Tabla II-1 se visualiza que la mejor opción para el desarrollo de la investigación es USB DAQ 1208FS OEM de Measurement Computing.

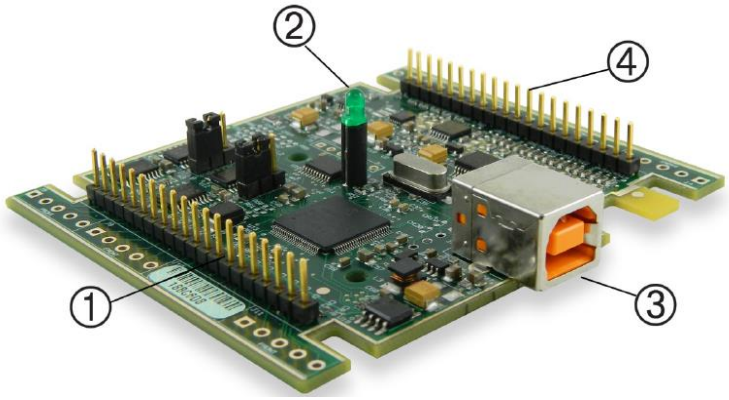
El dispositivo puede trabajar con varios rangos de muestreo, desde los 6.25Ks/S hasta 50Ks/S.

Figura II-9 Diagrama de Bloques de USB-1208FS-Plus-OEM



Fuente: https://www.mccdaq.com/pdfs/manuals/USB-1208FS-Plus-OEM.pdf

Figura II-10 Tarjeta de Adquisición de Datos



1. Conector para los pines del 21 al 40. 2. Indicador Led de encendido. 3. Conector USB. 4. Conector para los pines del 1 al 20

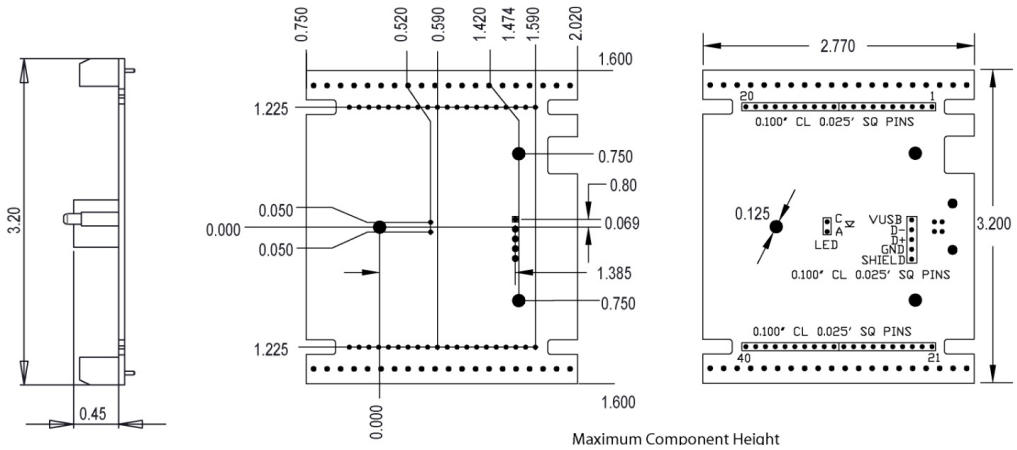
Fuente: https://www.mccdaq.com/usb-data-acquisition/USB-1208FS.aspx

El dispositivo en mención tiene las siguientes características (COMPUTING, 2020):

- 08 canales de entradas analógicas con rangos ±20 V, ±10 V, ±5 V, ±4 V, ±2.5 V, ±2.0 V, ±1.25 V, and ±1.0 V
- 02 canales de salidas analógicas con un rango de 0V a 5V
- 16 entradas/salidas digitales con un nivel de 5 V.

- Dimensiones:

Figura II-11 Dimensiones DAQ USB 1208FS OEM



Fuente: https://www.mccdaq.com/usb-data-acquisition/USB-1208FS.aspx

2.3.2 Sistema embebido

Según Salas (Arriaran, 2015) define como un sistema embebido como a todo circuito electrónico digital capaz de realizar operaciones computacionales, generalmente en tiempo real, que sirven para cumplir una tarea específica. Para el caso de esta investigación se tomó en consideración la siguiente tabla comparativa entre diferentes tipos de sistemas embebidos.

Tabla II-2 Tabla Comparativa de Sistemas Embebidos

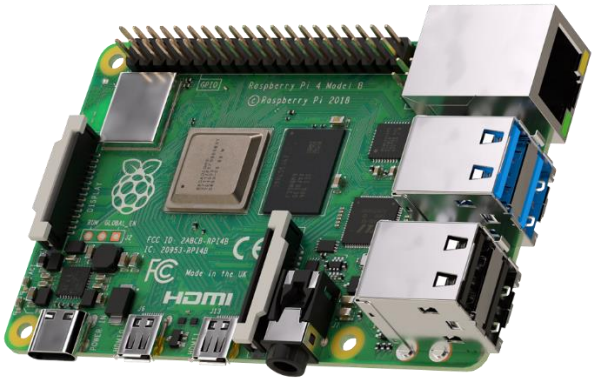
Descripción	Raspberry Pi 4	Ordroid C2	Juagar One	Pine A 64	Latte Panda
Procesador	1.5Ghz Quad-Core ARM Cortex A72 Broadcom BCM2711B0	1.5Ghz Quad-Core ARM Cortex A53	1.3 Ghz Quad-Core INTEL ATOM Z3735G/F	1.2 Ghz 64 Bits Quad-Core ARM Cortex A53	1.8 Ghz Quad-Core INTEL Z8300
Memoria de expansión	Tarjeta Micro SD	Tarjeta SD	16 Gb	Tarjeta Micro SD	64 Gb
RAM	4 Gb	2 Gb	1 Gb	2 Gb	4 Gb
Conectividad	2 USB 2.0 2 USB 3.0 ETHERNET BLUETOOTH WIFI 2 HDMI 1 PORT CAMERA 40 PINOUT	4 USB 2.0 1 USB OTG ETHERNET 1 HDMI	3 USB 2.0 ETHERNET 1 HDMI	2 USB 2.0 ETHERNET 1 HDMI	2 USB 2.0 1 USB 3.0 ETHERNET BLUETOOTH WIFI
Sistema Operativo	LINUX	LINUX/ ANDROID	LINUX/ ANDROID/ WINDOWS 10	ANDROID/ LINUX/ WINDOWS IOT	WINDOWS 10
Precio US$D	60.00	46.00	79.00	29.00	139.00

Fuente: Elaboración Propia

Después de haber realizado una búsqueda de diferentes sistemas embebidos y según la Tabla II-2 se visualiza que la mejor opción para el desarrollo de la investigación es Raspberry Pi 4 4Gb.

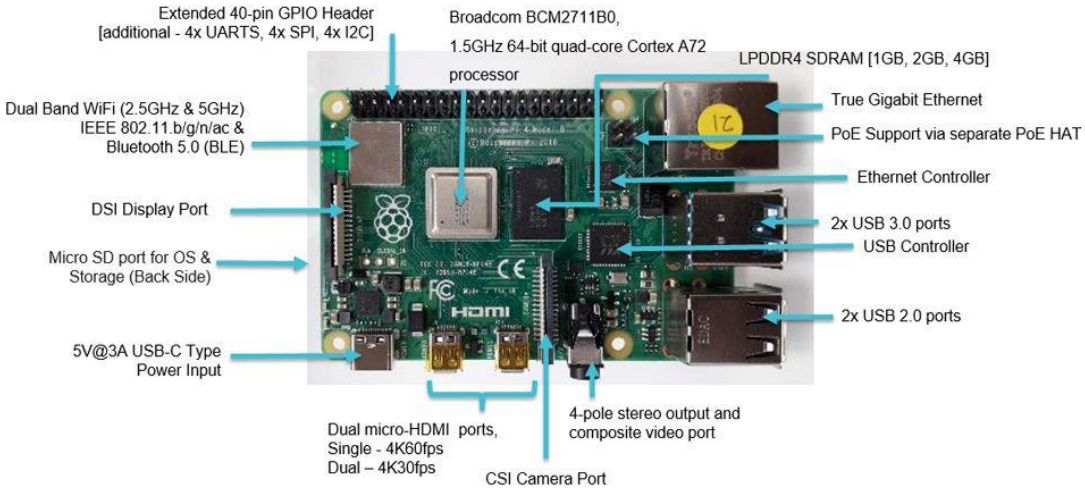
A continuación, se hace referencia a una imagen y un diagrama de bloques del sistema embebidos en mención.

Figura II-12 Raspberry Pi 4 Model B+ 4GB



Fuente: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

Figura II-13 Raspberry Pi 4 Periféricos



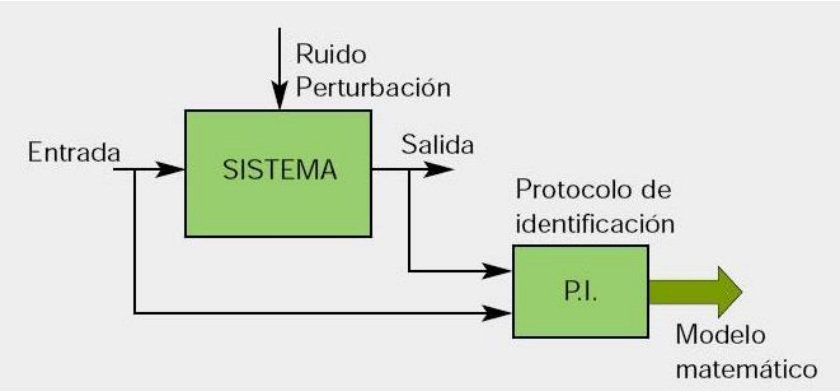
Fuente: https://omniretro.com/tecnologia/raspberry-pi-4-especificaciones-caracteristicas/

2.3.3 Algoritmos de identificación

Ésta es la primera parte de la investigación, donde se obtiene el tipo de modelo matemático que se pretende identificar. Según los diferentes autores pueden existir varias clasificaciones como, por ejemplo: estocásticos o deterministas; estáticos o dinámicos; de parámetros concentrados o distribuidos, lineales o no lineales; tiempo continuo o tiempo discreto; paramétricos y no paramétricos, etc. Cada autor aborda el tema de la identificación según su visión.

Ahora debemos tener en cuenta que, una identificación, trata de una técnica para construir un modelo de un sistema de control o de un proceso, tomando en cuenta las variables de entrada y salida del sistema a identificar. Como se muestra en la Figura II-10 se tiene un diagrama de bloques de un proceso de identificación.

Figura II-14 Proceso de Identificación de un sistema



Fuente : http://www.tecnicaindustrial.es/TIFrontal/a-1408-introduccion-identificacion-sistemas.aspx

Según Sedano et al (Javier Sedano Franco; José Ramón Villar Flecha, 2020) indican que la identificación se puede realizar tomando en cuenta la estructura del modelo, y de su comportamiento físico o no del mismo. Donde se pueden distinguir algunos tipos de estructuras de los modelos:

- Estructuras de Tipo Black-box: donde los parámetros del modelo no tienen una interpretación física.
- Estructuras de Tipo Gray-box: donde algunas partes del sistema pueden ser modeladas teniendo en cuenta principios fundamentales, y otras partes son como una caja negra.

- Estructuras de Tipo White-box: la estructura del modelo se obtiene a partir de leyes fundamentales. Los parámetros tienen una interpretación física.

Hay varias formas de catalogar los modelos matemáticos, para esto se ha realizado un resumen que sería de la siguiente manera:

- Modelos de tipo mentales, los cuales no requieren un formalismo matemático.

- Modelos de tipo no paramétricos, estos tienen como principal característica el uso de gráficos, diagramas u otro tipo de representaciones que describen sus propiedades, para esto se emplean funciones de excitación como, por ejemplo: respuesta al impulso, respuesta al escalón, o a la frecuencia.

- Modelos de tipo paramétricos o matemáticos, son aquellos en los cuales se puede describir las relaciones entre las variables de entrada y salida del sistema mediante uno o varios sistemas de ecuaciones matemáticas como, por ejemplo: ecuaciones diferenciales en función al tiempo o en ecuaciones en diferencias para sistemas discretos.

Ahora en función del tipo de sistema y de la representación matemática a emplearse, dichos sistemas pueden ser clasificados de la siguiente manera:

- Determinísticos: cuando existe una relación entre entradas y salidas mediante una ecuación; se estudia la relación entre la entrada y la salida.

- Estocásticos: cuando este posee un cierto grado de incertidumbre, estos solo pueden ser definidos mediante técnicas probabilísticas.

- Dinámicos: cuando las salidas varían con el tiempo, donde el valor actual de la salida está en función al tiempo transcurrido desde la aplicación de una señal de entrada.

- Estáticos: cuando en un sistema la salida depende únicamente de la entrada, la relación es independiente del tiempo entre las entradas y salidas.

- Continuos: se trata de sistemas que trabajan con señales continuas y se expresan mediante ecuaciones diferenciales.

- Discretos: son aquellos que trabajan con señales muestreadas, y se pueden expresar mediante el uso de ecuaciones en diferencias.

El método empleado en el presente trabajo de investigación es la identificación no paramétrica. Este modelo tiene como principal característica que realiza la identificación mediante la construcción de experimentos.
Algunas consideraciones que se deben tomar en cuenta para la correcta identificación del modelo son las siguientes:

- El modelo se desarrolla siempre a partir de una serie de aproximaciones y, por lo tanto, siempre va a ser una representación parcial del modelo original, donde se debe tomar en cuenta el porcentaje de error.

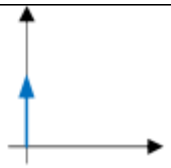
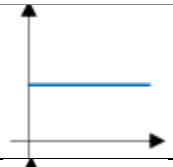
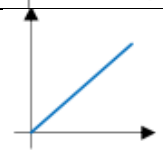
- El modelo debe tener siempre la relación entre la simplicidad del método, así como la necesidad de obtener los detalles más esenciales del sistema a identificar.

Existen algunas técnicas de identificación de los sistemas de tipo no paramétricas como: análisis a la respuesta transitoria, el análisis de correlación entre señales de entrada y salida y las técnicas frecuenciales.

2.3.4 Análisis de la respuesta transitoria

Las señales de excitación de tipo no periódicas que pueden ser utilizadas para evaluar la respuesta transitoria son: impulsos de diferente amplitud y de corta duración, función escalón y función rampa.

Tabla II-3 Señales de prueba

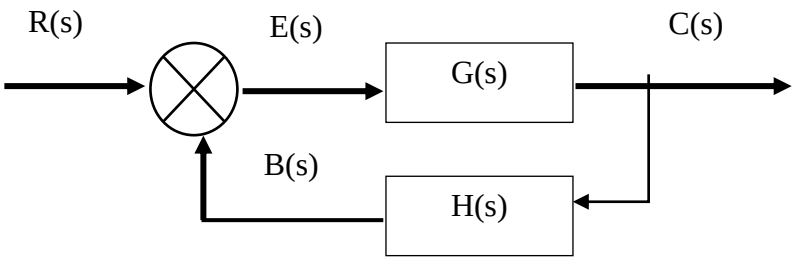
Tipo de Señal	Representación	Gráfica
Impulso	$\delta(t)$	
Escalón	$u_s(t)$	
Rampa	$r(t) = tu_s(t)$	

Fuente: Elaboración Propia.

2.3.5 Análisis de la respuesta impulso unitario

Esta metodología consiste en aplicar como entrada al proceso una señal de tipo impulso. Pero primero se debe hacer unos alcances acerca de las nomenclaturas y diagramas a utilizar.

Figura II-15 Diagrama de Bloques de un sistema de lazo cerrado



Fuente: Elaboración propia

Donde:

R(s) : Señal de referencia o consigna

C(s) : Señal de salida (Variable controlada)

B(s) : Señal de retroalimentación

E(s) : Señal de error

H(s) : Función de transferencia de retorno

G(s) : Función de transferencia de la planta

El impulso unitario viene definido por la función: $X(s) = 1$; entonces, a la salida del sistema se obtendrá:

$$Y(s) = X(s).\frac{K}{\tau s + 1} = 1.\frac{K}{\tau s + 1} = \frac{K}{\tau s + 1} \qquad \textit{(II-1)}$$

Donde:

τ : Constante de Tiempo.

K : Ganancia del estado estacionario del sistema

Suponiendo K=1 se obtiene:

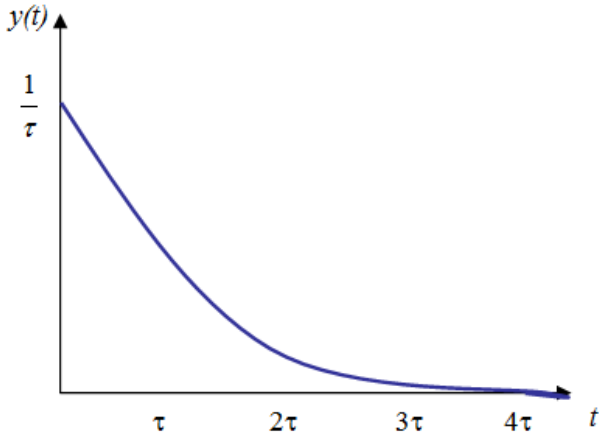
$$Y(s) = \frac{1/\tau}{s + 1/\tau} \qquad \textit{(II-2)}$$

Aplicando la trasformada inversa de la Laplace se tiene:

$$y(t) = \mathcal{L}^{-1}\{Y(s)\} = 1/\tau\, \mathcal{L}^{-1}\left\{\frac{1}{s + 1/\tau}\right\} \qquad \textit{(II-3)}$$

$$y(t) = \frac{1}{\tau}.e^{-t/\tau} \qquad (II\text{-}4)$$

Gráfica II-1 Respuesta al impulso



Fuente: http://plantscontrol.blogspot.com/2012/02/8_14.html

2.3.6 Análisis de la respuesta escalón unitario

Una señal también utilizada es la respuesta a un escalón, esta señal tiene una ventaja, que es mucho más sencilla su generación. Esta señal se puede describir como:

$$u(t) = \begin{cases} \alpha\ , & t > 0 \\ 0\ , & t < 0 \end{cases} \qquad (II\text{-}5)$$

Si α=1 entonces la amplitud está definida por X(s) = $1/s$; por tanto, se obtiene:

$$Y(s) = X(s).\frac{K}{\tau s + 1} = \frac{1}{s}.\frac{K}{\tau s + 1} = \frac{K}{s(\tau s + 1)} \qquad (II\text{-}6)$$

Descomponiendo a Y(s) en fracciones simples:

$$Y(s) = \frac{a_1}{s} + \frac{a_2}{\tau s + 1} \qquad (II\text{-}7)$$

Calculando los avalores de a_1 y a_2 :

$$a_1 = \left[s.\frac{K}{s(\tau s+1)}\right]_{s=0} = \left[\frac{K}{\tau s+1}\right]_{s=0} = K \qquad \textit{(II-8)}$$

$$a_2 = \left[(\tau s+1).\frac{K}{s(\tau s+1)}\right]_{s=-1/\tau} = \left[\frac{K}{s}\right]_{s=-1/\tau} = -K.\tau \qquad \textit{(II-9)}$$

Por lo tanto:

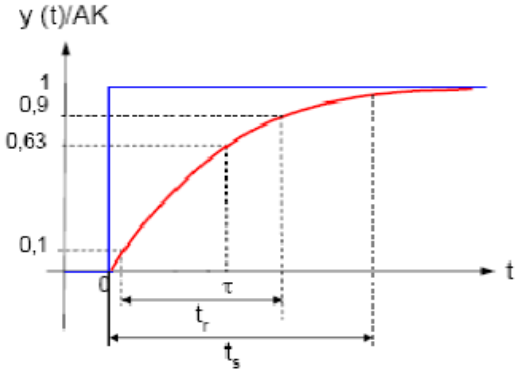
$$Y(s) = \frac{K}{s} - \frac{K\tau}{(\tau s+1)} = \frac{K}{s} - \frac{K}{s+1/\tau} \qquad \textit{(II-10)}$$

Aplicando la trasformada inversa de la Laplace se tiene:

$$y(t) = \mathcal{L}^{-1}\{Y(s)\} = K\mathcal{L}^{-1}\left\{\frac{1}{s}\right\} - K\mathcal{L}^{-1}\left\{\frac{1}{s+1/\tau}\right\} \qquad \textit{(II-11)}$$

$$y(t) = K.(1 + e^{-t/\tau}) \qquad \textit{(II-12)}$$

Gráfica II-2 Respuesta al escalón



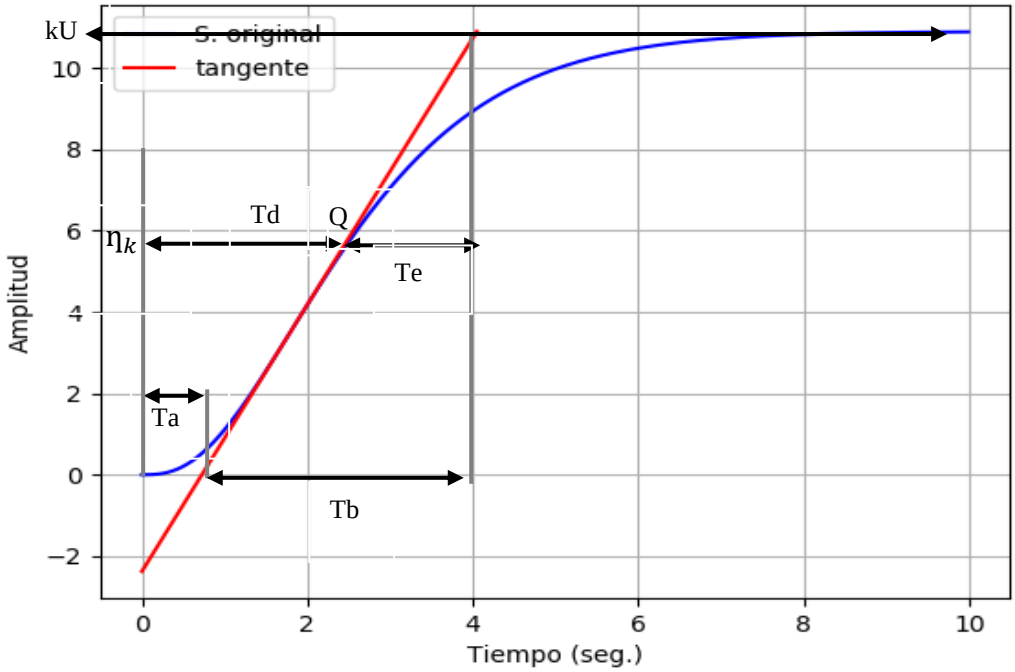
Fuente:
http://instrumentacionunillanos2014.blogspot.com/2014/10/caracteristicas-dinamicas-de-los.html

2.3.7 Método de Strejc

Este método fue desarrollado por Vladimir Strejc, ingeniero mecánico de la Universidad Técnica de Czech, quien trabajo durante los 10 primeros años como ingeniero de diseño de sistemas de control en deferentes industrias, el método en mención trata de la identificación de sistemas de polos múltiples, mediante los parámetros T_a , T_b , T_d y T_e, los cuales son obtenidos cuando se aplica una señal escalón a un sistema.

Para obtener dichos parámetros se basa en trazar una línea recta tangente a la respuesta escalón del sistema, la cual va a estar superpuesta sobre el área de pendiente del sistema a identificar, de tal forma que el parámetro T_a se obtiene con el corte de la recta tangente con el eje de abscisas y el origen T_b se obtiene desde el punto de corte de T_a con el corte de kU una paralela al eje de abscisas en el punto donde la respuesta se encuentra estable.

Gráfica II-3 Método Strejc



Fuente: Elaboración Propia

Después de graficar las rectas se puede obtener el valor de T_a y T_b, para luego calcular el coeficiente de $^{T_a}/_{T_b}$. Con este valor se logra obtener el valor de "n", donde "n" es el orden del sistema, para el cálculo de este valor se hace uso de la Tabla II-4.

Tabla II-4 Coeficientes de Strejc

n	$^{T_a}/_{T_b}$	$^{T_e}/_{T_b}$	η_k	$^{T_a}/_{\tau}$	$^{T_b}/_{\tau}$	$^{T_d}/_{\tau}$	$^{T_e}/_{\tau}$
1	0	1	0	0	1	0	1
2	0.104	0.736	0.264	0.282	2.718	1	2.000
3	0.218	0.677	0.323	0.805	3.695	2	2.500
4	0.319	0.647	0.353	1.425	4.463	3	2.888
5	0.410	0.629	0.371	2.100	5.119	4	3.219

Fuente (Angel, Martínez Bueno; Jorge, Pomares, 2011): Identificación experimental de sistemas

El método en mención va a ser implementado en un sistema embebido, se trata de una identificación no paramétrica de lazo abierto, donde se identifican sistemas hasta de orden 5.

- Polos reales múltiples con retardo

  Se aplica el mismo procedimiento de polos reales múltiples, pero tomando ahora la consideración de un retardo al comienzo.

$$G(s) = \frac{K}{(1+\tau_1 s)(1+\tau_2 s)(1+\tau_3 s)\ldots..(1+\tau_n s)} \qquad \textit{(II-13)}$$

  Ahora la función de transferencia será de la siguiente manera:

$$G(s) = \frac{K}{(1+\tau s)^n}.e^{-Ts} \qquad \textit{(II-14)}$$

  Donde:

  K = Ganancia del sistema $K = \frac{\Delta y}{\Delta u}$

  $\boldsymbol{\tau}$ = Constante de Tiempo

  T = Retardo

  n = número de polos del sistema

$$G(s) = \frac{K}{(1+\tau s)^n}.e^{-Ts} \qquad \textit{(II-15)}$$

  El procedimiento para la identificación es el siguiente:

  1. Se debe excitar a la planta con una señal escalón, para obtener la respuesta del sistema y calcular la Gancia K en el régimen estacionario.
  2. La respuesta es un vector que llevará el nombre de P(t).
  3. Se realiza la P´(t) y P´´(t), donde ahora P´´(t)=0, donde se puede obtener la posición del punto de inflexión T_d.
  4. Se evalúa P´(T_d) =m, donde se obtiene el valor de la pendiente y también su intercepto con el eje de las ordenadas.

$$Y_0 = P(T_d) \quad - mT_d \qquad \textit{(II-16)}$$

5. Con esto se obtiene la recta tangente en el punto Q.

$$y(t) \quad = mx + y_0 \qquad \textit{(II-17)}$$

6. Se evalúa tanto y(t)=0 y y(t)=uK, con esto se calculan los valores de T_a y T_b respectivamente.

$$T_b = T_c - T_a \qquad \textit{(II-18)}$$

$$T_e = T_c - T_d \qquad \textit{(II-19)}$$

7. Luego se calculan los cocientes T_a/T_b y T_e/T_b y se obtiene el orden n.

n	T_a/T_b	T_e/T_b	η_k
1	0	1	0
2	0.104	0.736	0.264
3	0.218	0.677	0.323
4	0.319	0.647	0.353
5	0.410	0.629	0.371

2.3.8 Software Open Source

Según Torres et al (Elsa Gladys Torres Saybay, Arturo Francisco Ordoñez Peña, Rodrigo Víctor Alarcón A. y Melvin Leonardo López Franco, 2016), en su publicación hace uso de una definición, donde comenta que la palabra open source una expresión de la lengua inglesa y que puede traducirse como “fuente abierta”. La principal característica de estos programas es tener el acceso a su código fuente, con lo cual permite hacer las modificaciones por parte de otros programadores.

En este trabajo de investigación se empleó la herramienta Python, la cual tiene un sin número de librerías orientas a trabajos de investigación, modelamiento,

tratamiento de señales, etc. Existen organizaciones como Open Source Initiative (OSI), es una organización dedicada la promoción de código abierto.

Figura II-16 Logotipo de Open Source

Fuente: https://opensource.org/logo-usage-guidelines

- **Características de un software open source:**
  Según el portal web Tecnología y Negocios (Negocios, 2020) , nos comenta que los usuarios de este tipo de software tienen las siguientes particularidades.

  a. Están a la espera de licencias públicas, por ende, que no requiere códigos de activación.
  b. Esperan las nuevas actualizaciones y posibles parches de forma periódica.
  c. Esperan la ayuda de una comunidad o blog donde pueden conseguir ayuda entre ellos mismos.

- **Ventajas de un software open source:**

  a. Los software de tipo open source cuentan con la posibilidad modificar, compartir e investigar el código fuente de una aplicación.

b. Por otro lado, el uso de software de tipo open source promueven el trabajo en equipo y la colaboración entre sus miembros. Con esto supone una mayor rapidez de implementación y tener una gran variedad de herramientas.

c. Por lo tanto, los software de tipo open source le pertenece a toda la comunidad o suscriptores, de tal manera que los nuevos desarrollos y/o actualizaciones dependen de la comunidad.

- **Desventajas de un software open source:**

a. Se deben tomar en cuenta al momento de desarrollar una aplicación, que no existe soporte técnico o un lugar donde colocar una inconformidad.

b. Estos programas, no tienen un soporte o empresa a la cual se le pueda realizar algún tipo de reclamación.

c. La posibilidad de no continuidad de los programas, por lo que puede entrar a la categoría de desuso. Aquí es donde se sufren muchos inconvenientes para las migraciones a un nuevo programa.

2.3.9 Python

Una definición de este lenguaje se encuentra en (https://openwebinars.net/blog/que-es-python/, 2020) donde dice que este lenguaje es multiplataforma que tiene como características poseer un código legible y limpio. Esa es una de las razones de su éxito, es uno de los lenguajes de iniciación de muchos desarrolladores, por lo que es estudiado en escuelas de formación y universidades a nivel mundial.

En la actualidad este lenguaje de programación es uno de los más usados para trabajar con grandes volúmenes de datos, también es usado a nivel científico y para aplicaciones en ingeniería, por su gran variedad de bibliotecas y cuenta con una comunidad muy activa de programadores que constantemente están realizando modificaciones.

Figura II-17 Logotipo de Python

Fuente: https://www.python.org/

Las librerias utilizadas para la presente investigación son:

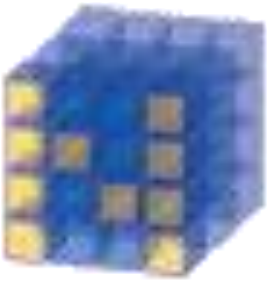
- Scipy : Librería para el manejo de rutinas numéricas, integración numérica, interpolación, optimización, álgebra lineal y estadísticas, para mayor información en detalle se encuentra en: https://www.scipy.org/scipylib/index.html

Figura II-18 Logotipo de Scipy

Fuente: https://www.scipy.org/scipylib/index.html

- Numpy : Librería para el manejo de rutinas para cálculos científicos, como matriz N-dimensional, integración de código C / C ++ y Fortran, Álgebra lineal y transformada de Fourier. Para mayor información en detalle se encuentra en: https://numpy.org/

Figura II-19 Logotipo de Numpy



Fuente: https://numpy.org/

- Matplotlib : Librería para el manejo de rutinas para realizar graficas en 2D, gran cantidad formatos impresos. Para mayor información en detalle se encuentra en: https://matplotlib.org/

Figura II-20 Logotipo de Matplotlib

Fuente: https://matplotlib.org/

- SymPy : Librería para el manejo de rutinas para matemática simbólica. Trabaja como un un sistema de álgebra computacional con todas las funciones. Para mayor información en detalle se encuentra en: https://www.sympy.org/en/index.html

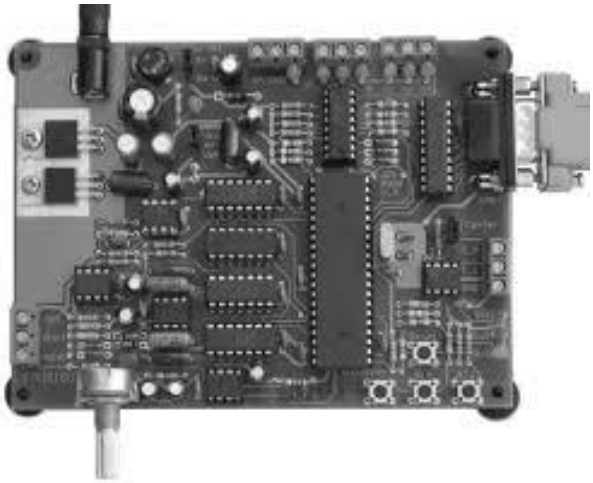
Figura II-21 Logotipo de Sympy

Fuente: https://www.sympy.org/en/index.html

2.3.10 Hardware in Loop

Según Hernández et al (HERNÁNDEZ-MEDRANO, Israel*†, PINEDA-MARTÍNEZ, Alejandro Gabriel, TAPIA-TINOCO,, 2017) es un sistema embebido utilizado para el desarrollo y comprobación de sistemas en tiempo real, el cual interactúa con el medio a través de sus interfaces puede capturar señales de los sensores y enviar una señal hacia los actuadores. En algunos casos, se requieren interfaces para el manejo de diferentes niveles de tensiones e intensidades de corriente.

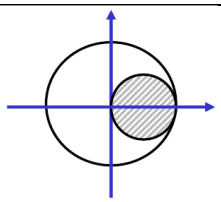
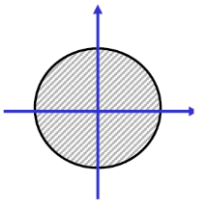
Figura II-22 Emulador de Plantas SISO



Fuente: https://pdfs.semanticscholar.org/3c75/d19f6747db9f05b746f9edde0e96bb2be489.pdf

En la tesis de Velo Sánchez (Esteban, 2014) hace una emulación de las funciones de transferencia en el tiempo continuo y discreto y realiza la transformación del Plano S al Plano Z hace uso de las aproximaciones de Tustin y Euler.

Tabla II-5 Tabla Comparativa de tipos de aproximación

Descripción	Aproximación de Euler	Aproximación de Tustin
Grado de similitud	Bajo	Alto
Oscilante o Inestable	Puede llegar a ser estable	Mantiene el estado del sistema original
Región de Estabilidad		

Fuente: Elaboración Propia

Como se muestra en el Tabla II-5, para nuestra investigación se implementó la aproximación de Tustin.

Esta es la segunda parte de la investigación, consiste en la implementación de un algoritmo que nos permite realizar la emulación de un sistema hasta de orden 5.

2.3.11 Aproximación de Tustin, bilineal o trapezoidal

Se trata de hacer la sustitución de un término en una función transferencia en tiempo continuo, la variable s:

$$s = \frac{2}{T}.\frac{z-1}{z+1} \qquad (II\text{-}20)$$

Donde:

T: Tiempo de muestro (Segundos)

- Para funciones de transferencia de 1er Orden se tiene:

$$G(s) = \frac{a_1 s + a_0}{b_1 s + b_0} \qquad (II\text{-}21)$$

Donde:

a0, a1, b0, b1: Son los coeficientes de la FT

Reemplazando (II 20) en (II 21) se tiene:

$$G(z) = \frac{(2a_1 + a_0 T)z - 2a_1 + a_0 T}{(2b_1 + b_0 T)z - 2b_1 + b_0 T} \qquad (II\text{-}22)$$

- Para funciones de transferencia de 2do Orden se tiene:

$$G(s) = \frac{a_2 s^2 + a_1 s + a_0}{b_2 s^2 + b_1 s_1 - b_0} \qquad (II\text{-}23)$$

Donde:

a0, a1, a2, b0, b1, b2: Son los coeficientes de la FT

Reemplazando (II 20) en (II 23) se tiene:

$$G(z) = \frac{(4a_2 + 2a_1 T + a_0 T^2)z^2 + (2a_0 T^2 - 8a_1 T)z + 4a_2 + a_0 T^2 - 2a_1 T}{(4b_2 + 2b_1 T + b_0 T^2)z^2 + (2b_0 T^2 - 8b_1 T)z + 4b_2 + b_0 T^2 - 2b_1 T} \qquad (II\text{-}24)$$

- Para funciones de transferencia de 3er Orden se tiene:

$$G(s) = \frac{a_3 s^3 + a_2 s^2 + a_1 s + a_0}{b_3 s^3 + b_2 s^2 + b_1 s_1 - b_0} \qquad (II\text{-}25)$$

Donde:

a0, a1, a2, a3, b0, b1, b2, b3: Son los coeficientes de la FT

Reemplazando (II 20) en (II 25) se tiene:

$$G(z) = \frac{(8a_3 + 4a_2 T + 2a_1 T^2 + a_0 T^3)z^3 + (-24a_3 - 4a_2 T + 2a_1 T^2 - 3a_0 T^3)z^2}{(8b_3 + 4b_2 T + 2b_1 T^2 + b_0 T^3)z^3 + (-24b_3 - 4b_2 T + 2b_1 T^2 - 3b_0 T^3)z^2}$$

$$+ \frac{(24a_3 - 4a_2 T - 2a_1 T^2 + 3a_0 T^3)z + (-8a_3 + 4a_2 T^2 - 2a_1 T^2 + a_0 T^3)}{(24b_3 - 4b_2 T - 2b_1 T^2 + 3b_0 T^3)z + (-8b_3 + 4b_2 T^2 - 2b_1 T^2 + b_0 T^3)} \qquad (II\text{-}26)$$

2.3.12 Transformación a ecuaciones en diferencias

Después de la aplicación de la aproximación de Tustin se tiene la función de transferencia en el Plano Z, luego para poder implementar estas funciones en un sistema embebido se debe convertir a ecuaciones en diferencias. Este paso es muy sencillo. Tan solo hacen falta los parámetros anteriormente calculados y la transformada Z inversa de la ecuación 10.2.0.1 (donde i es el instante de

muestreo actual y k un número natural que hace referencia a un estado anterior):

$$z^{-k}X(z) = X(i-k) \qquad (II\text{-}27)$$

Donde:

i : es el instante del muestreo actual

k : un estado anterior

Para un caso general se tendría de la siguiente manera para un sistema de 2do Orden:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{a_2z^2 + a_1z + a_0}{b_2z^2 + b_1z_1 - b_0} = \frac{a_2 + a_1z^{-1} + a_0z^{-2}}{b + b_1z^{-1} + b_0z^{-2}} \qquad (II\text{-}28)$$

Operando se obtiene:

$$Y(k) = \frac{a_2}{b_2}x(k) + \frac{a_1}{b_2}x(k-1) + \frac{a_0}{b_2}x(k-2) - \frac{b_1}{b_2}y(k-1) - \frac{b_0}{b_2}y(k-2) \qquad (II\text{-}29)$$

2.3.13 Interfaz gráfica Hardware

El sistema embebido debe contar con una pantalla industrial para la visualización de las gráficas durante y después del proceso de identificación del sistema, así como también para el proceso de emulación. Este dispositivo debe contar también con una uy buena resolución, de un tamaño apropiado para que el sistema a implementar sea portátil.

Tabla II-6 Tabla Comparativa de TouchScreen

Descripción	Eyoyo	SunFounder	NewSoul – Touch	Metal Shell KF12
Tamaño	8,0 pulgadas	10,1 pulgadas	12,3 pulgadas	12,0 pulgadas
Resolución	1280x800	1280x800	1600 x 1200	1024x768
Peso	35.00 onzas	32.50 onzas	32.66 onzas	36.00 onzas
Conectividad	2 HDMI	1 HDMI 01 Micro USB 01 VGA	USB HDMI DVI VGA audio salida de audio	USB HDMI DVI VGA
Case	Standard	Standard	Robusto	Robusto
Precio US$D	60.00	139.00	179.00	259.00

Fuente: Elaboración Propia

Después de haber realizado una búsqueda de diferentes tipos de pantallas de tipo touchscreen y según la Tabla II-5 se visualiza que la mejor opción para el desarrollo de la investigación es NewSoul – Touch.

Figura II-23 Industrial Touch-Panel

Fuente: https://www.amazon.com/-/es/Newsoul-pantalla-pulgadas-resoluci%C3%B3n-sistemas/dp/B07P8ZCJCG /

2.3.14 Interfaz gráfica Software

En el sistema embebido debe contar con una interfaz gráfica, GUI o un tipo de presentación de los resultados amigable para la visualización de la identificación del sistema, así como la interfaz de la etapa de la emulación de la función de transferencia identificada.

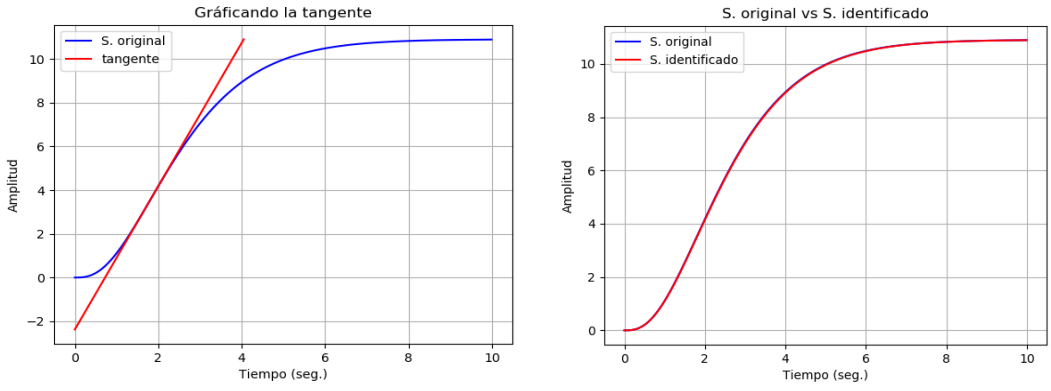
Entre las principales interfaces gráficas para la visualización de los resultados de la investigación se tiene:

Tabla II-7 Tabla Comparativa de TouchScreen

Descripción	Eyoyo	SunFounder	NewSoul – Touch	Metal Shell KF12
Tamaño	8,0 pulgadas	10,1 pulgadas	12,3 pulgadas	12,0 pulgadas
Resolución	1280x800	1280x800	1600 x 1200	1024x768
Peso	35.00 onzas	32.50 onzas	32.66 onzas	36.00 onzas
Conectividad	2 HDMI	1 HDMI 01 Micro USB 01 VGA	USB HDMI DVI VGA audio salida de audio	USB HDMI DVI VGA
Case	Standard	Standard	Robusto	Robusto
Precio US$D	60.00	139.00	179.00	259.00

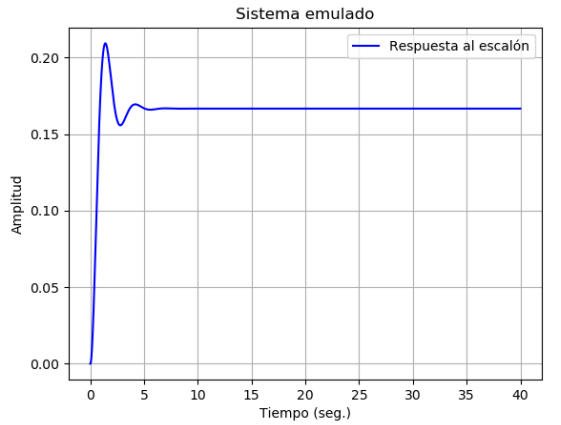
Fuente: Elaboración Propia

Gráfica II-4 Gráficas de la Identificación del sistema



Fuente: Elaboración Propia

Gráfica II-5 Gráficas del sistema emulado



Fuente: Elaboración Propia

2.4 Definición de términos básicos

Tabla II-8 Tabla de Términos

Termino	Descripción / Significado
ADC	(Analog Digital Convertion) Conversión de Análogo a Digital
ARMAX	Auto-Regressive Moving Average with eXogenous inputs
ARMv7	Familia de Procesadores ARM
ARX	Auto-Regressive with eXogenous inputs
BJ	Box-Jenkins
CAN	(Controller Área Network) Protocolo de comunicaciones basado en topología Bus
CNC	(Computer Numerical Control) Control Numérico Computarizado
CPLD	(Complex programmable logic device) Dispositivo Lógico Programable Complejo.
DAC	(Digital Analog Convertion) Conversión de Digital a Análogo
DSP	(Digital Signal Processing) Procesamiento Digital de Señal.
e(t)	Señal de error
FPGA	(Field Programmable Gate Array) Arreglo de Compuertas Programables en Campo.
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
HIL	(Hardware In the Loop) Simulación de Hardware en el Lazo.
HMI	Human Machine Interface
I2C	Inter-Integrated Circuit
IEC	(International Electrotechnical Commission) Comisión Eletrotécnica Internacional
Ks/S	Kilosamples per Second
LABVIEW	Laboratory Virtual Instrument Engineering Workbench
LCD	(Liquid Crystal Display)
LTI	Linear Time-Invariant, es un sistema lineal invariante en el tiempo
LUT	(Look-Up Tables) Tabla de Consulta o arreglo asociativo.

MATLAB ®	MATrix LABoratory, laboratorio de matrices
MicroSD	Micro Secure Digital (microSD)
MIMO	Mutiple Input Multiple Output
NOOBS	New out of the box software
OE	Output-Error
OEM	Original Equipment Manufacturer
OPAMPS	Operational Amplificator
OS	Operating System
OTP	(One-Time Password) Memorias programables una sola vez.
PC	Personal Computer
PCB	Printed Circuit Board
PD	Controlador Proporcional Derivativo.
PEM	Predictor error Method
PI	Controlador Proporcional Integral.
PID	Controlador Proporcional Integral Derivativo
PLA	(Programmable Logic Array) Arreglo Lógico Programable
PLC	(Programmable Logic Controller) Controlador Lógico Programable
PLD	(Programmable Logic Device) Dispositivo Lógico Programable.
PV	Process Value
PWM	(Pulse Wide Modulation) Modulación por Ancho de Pulso
RAR	Formato de compresión de datos
RBPi4	Raspberry Pi 4 Model B 4GB
RS232	(Recommended Standard 232) Protocolo para intercambio de datos en forma serial.
RTLAB	Simulación en tiempo real de modelos de Simulink (Matlab®).
SBC	Single Board Computer
SISO	Single Input Single Output
SP	SetPoint
SPI	Serial Peripheral Interface Bus
SRAM	(Static Random-Access Memory) Memoria de Acceso Aleatorio Estática.

TCP/IP	(Transmission Control Protocol / Internet Protocol) Protocolo de Control de Transmision / Protocolo de Internet
TOUCHSCREEN	Pantalla Táctil
TTY	Teletypewriter
u(t)	Señal de entrada de un proceso
UART	(Universal Asynchronous Receiver / Transmitter) Receptor / Transmisor Universal Asíncrono.
USB	(Universal Serial Bus) Bus Serial Universal
VHDL	Sigla que representa la combinación de HSIC (Very High Speed Integrated Circuit) y HDL (Hardware Description Language)
VHDL-AMS	Ambiente de simulación en lenguaje de descripción
ZIP	Formato de compresión de datos

Fuente: Elaboración propia

III VARIABLES E HIPÓTESIS

3.1 Definición de las variables

- X1: Identificación de un sistema SISO hasta Orden 5.
- X2: Emulación de un sistema SISO hasta Orden 5.
- Y: Sistema portátil con herramientas open source.

3.2 Operacionalización de las variables

Para la definición conceptual de las variables se tomó en cuenta las siguientes consideraciones:

Para la variable.

• X1: Identificación de un sistema SISO hasta Orden 5 y haciendo uso de herramientas de software de tipo open source, las métricas para la validación son las siguientes:

a) Desarrollo de Algoritmos
   - Identificación de sistemas tipo SISO hasta Orden 5.
     i. Con un error cuadrático medio de 0.002.

Tabla III-1 Operacionalización de la variable X1

Variable	Definición Conceptual	Dimensión	Indicador	Instrumento	
X1: Identificación de un sistema SISO hasta Orden 5	Algoritmo de Identificación por el metodo de Strejc	Grado de similitud	0.002 de error cuadrático medio	Método de Error Cuadrático Medio	Vector de datos.

Fuente: Elaboración propia

Para la variable.

• X2: Emulación de un sistema SISO hasta Orden 5 y haciendo uso de herramientas de software de tipo open source, las métricas para la validación son las siguientes:

a) Desarrollo de Algoritmos

- Emulación de sistemas SISO hasta Orden 5..
    i. Con un error cuadrático medio de 0.002.

Tabla III-2 Operacionalización de la variable X2

Variable	Definición Conceptual	Dimensión	Indicador	Instrumento	
X2: Emulación de un sistema SISO hasta Orden 5.	Algoritmo de Emulación por el metodo de Tustin	Grado de similitud	0.002 de error cuadrático medio	Método de Error Cuadrático Medio	Vector de datos.

Fuente: Elaboración propia

Para la variable.

Y: Sistema portátil con herramientas open source.

Implementación de los algoritmos de identificación y emulación en un sistema embebido, las métricas para la validación son las siguientes:

a) Implementar los algoritmos en el sistema embebido:
   - Implementación del algoritmo de identificación.
     i. Con un error cuadrático medio de 0.002.
   - Implementación del algoritmo de emulación.
     i. Con un error cuadrático medio de 0.002.

Tabla III-3 Operacionalización de la variable Y

Variable	Definición Conceptual	Dimensión	Indicador	Instrumento	
Y: Sistema portátil con herramientas open source	Implementación del Algoritmo de Identificación y emulación de un sistema tipo SISO de Orden 5	Grado de similitud	0.002 de error cuadrático medio	Método de Error Cuadrático Medio	Vector de datos.

Fuente: Elaboración propia

3.3 Hipótesis

3.3.1 Hipótesis general

Si es posible desarrollar un sistema portátil para la identificación y emulación de procesos industriales de tipo SISO hasta Orden 5 sin la necesidad de interferir en el sistema.

3.3.2 Hipótesis especificas

- Si es posible realizar la identificación de un sistema de tipo SISO hasta Orden 5 para el sistema embebido cumpliendo con los tiempos de muestro.

- Si es posible implementar el algoritmo de emulación de un sistema de tipo SISO hasta Orden 5 para el sistema embebido cumpliendo con los tiempos de muestro.

IV METODOLOGIA

4.1 Tipo de investigación

La investigación de la presente tesis tiene las siguientes características:

- Es experimental, en el sentido de realizar diversas pruebas para la identificación y emulación de procesos o plantas en un sistema embebido, para la presente investigación es necesario realizar varias pruebas, afinar valores y lograr la similitud más próxima con el sistema en cuestión.

- Es cuantitativa, en el sentido que los datos adquiridos por las diferentes interfaces y acondicionadores son analizados y posteriormente procesados para cumplir con los objetivos de la investigación.

- Es comparativa, en el sentido que, para determinar el porcentaje de aproximación del sistema emulado con el sistema real, se valida con sistemas conocidos de tal forma que los resultados de las pruebas de campo y de laboratorio son próximas entre sí.

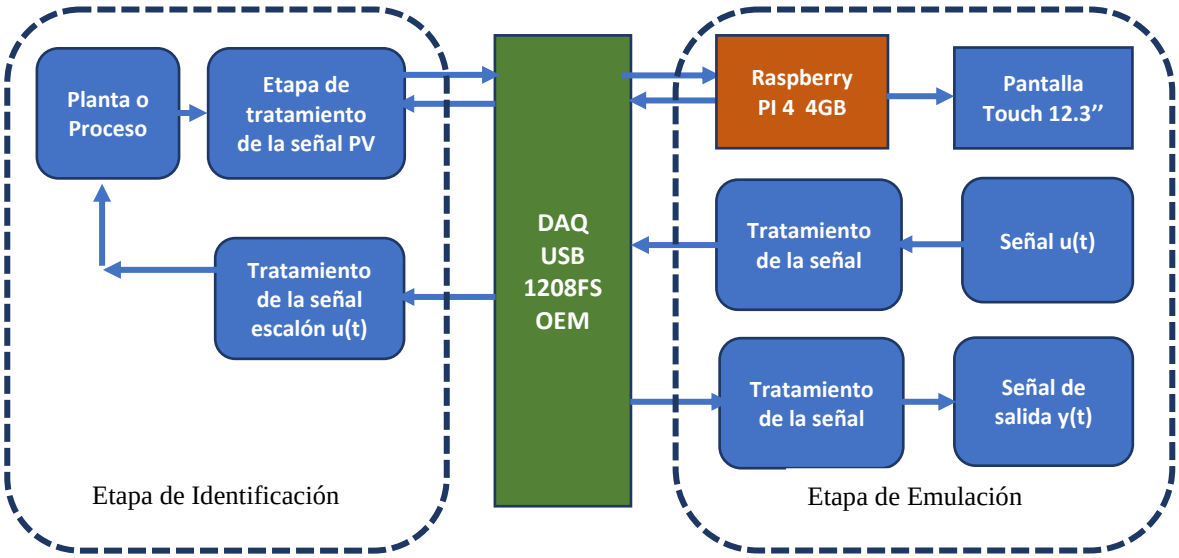
4.2 Diseño de la investigación

En este inciso debe destacarse que existen varias etapas en la presente investigación. Todas las partes de sistema implementado comparten entre sí varios tipos de tecnología. Consta de los siguientes módulos:

- Etapa de tratamiento de la señal.
- Etapa de acondicionamiento de la señal.
- Tarjeta de adquisición de datos.
- Single Board Computer (Raspberry Pi 4 4GB).
- Etapa de generación de la señal escalón u(t).
- Etapa de generación de la señal de salida y(t).
- Etapa de visualización de las funciones.

De tal manera que el siguiente diagrama de bloques se muestra el sistema embebido implementado.

Figura IV-1 Diagrama de Bloques de la Investigación



Fuente: Elaboración Propia

A continuación, se describen cada una de las etapas del trabajo de investigación, en cada una de ellas detallan el principio de funcionamiento, etapas de acondicionamiento, de procesamiento, algoritmos y visualización de las diferentes señales.

4.2.1 Etapa de identificación del sistema.

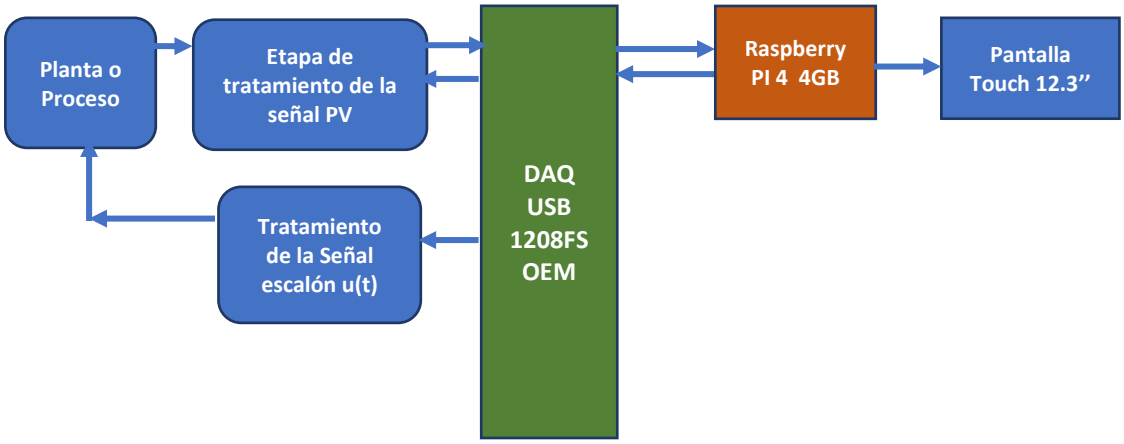
Está etapa cuenta con una tarjeta electrónica que realiza el acondicionamiento de la señal que proviene del proceso, planta o sistema. Después de realizar esta etapa la señal de salida del sistema ingresa a la entrada de la tarjeta de adquisición para posteriormente ser procesada por la Single Board Computer (Raspberry Pi 4 4GB). Todas las partes de sistema implementado comparten entre sí varios tipos de tecnología siendo de la siguiente manera:

- Etapa de acondicionamiento (tratamiento) de la señal escalón u(t).
- Etapa de acondicionamiento (tratamiento) de la señal del proceso PV.
- Tarjeta de adquisición de datos.
- Single Board Computer (Raspberry Pi 4 4GB).

- Etapa de visualización de las gráficas y resultados.

De tal manera que en el diagrama de bloques se muestra esta etapa implementada.

Figura IV-2 Diagrama de bloques de la Etapa de Identificación

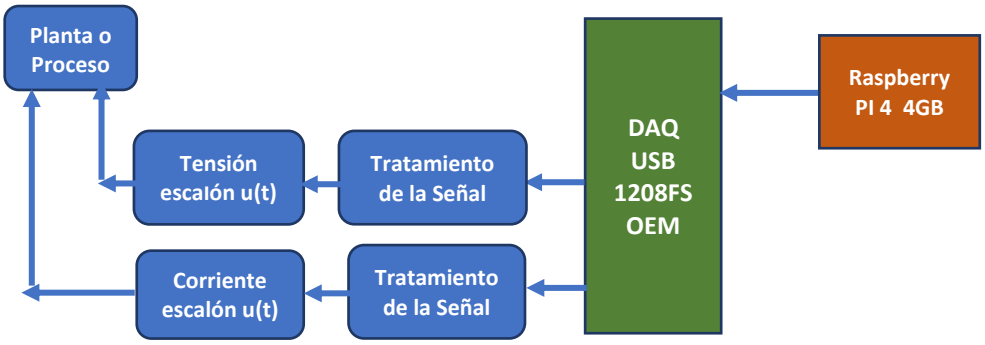


Fuente: Elaboración Propia

4.2.1.1 Etapa de acondicionamiento de la señal de entrada u(t).:

En esta etapa se genera la señal u(t) de salida para la excitación del sistema o proceso a identificar. En tal sentido, existe una etapa de acondicionamiento de la señal de salida de la tarjeta de adquisición a niveles de voltaje y corriente más usados en la industria. Una vez generada esta señal por el Raspberry y enviada por la tarjeta de adquisición vía USB a través de esta etapa de acondicionamiento, se puede excitar la planta, con la finalidad de obtener la curva característica del sistema a identificar. A continuación, se muestra en la Figura IV-3 el diagrama de bloques de la etapa en mención.

Figura IV-3 Diagrama de bloques de la señal u(t)

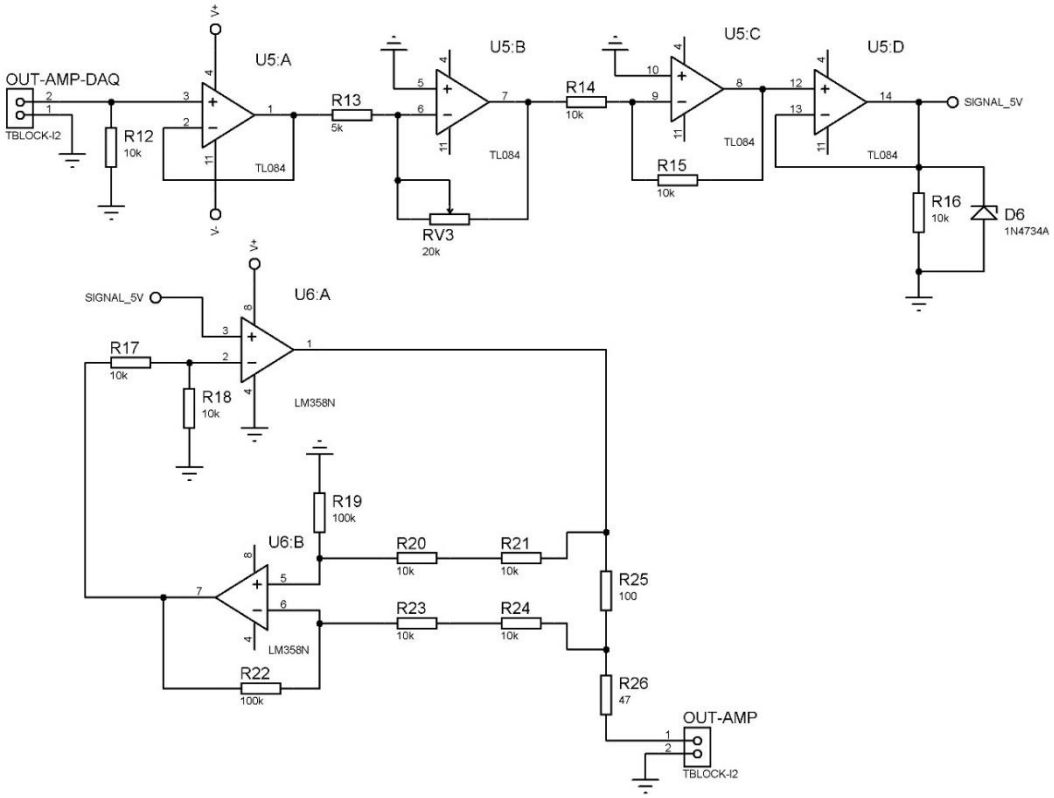


Fuente: Elaboración Propia

Esta etapa consta también de un circuito electrónico el cual realiza la función de seguidor de tensión y acondicionador de la señal para los diferentes tipos de señales (en tensión y corriente) en los niveles establecidos (0-10v o de 4-20mA).

Para la señal de u(t) en corriente se tiene el siguiente diseño electrónico.

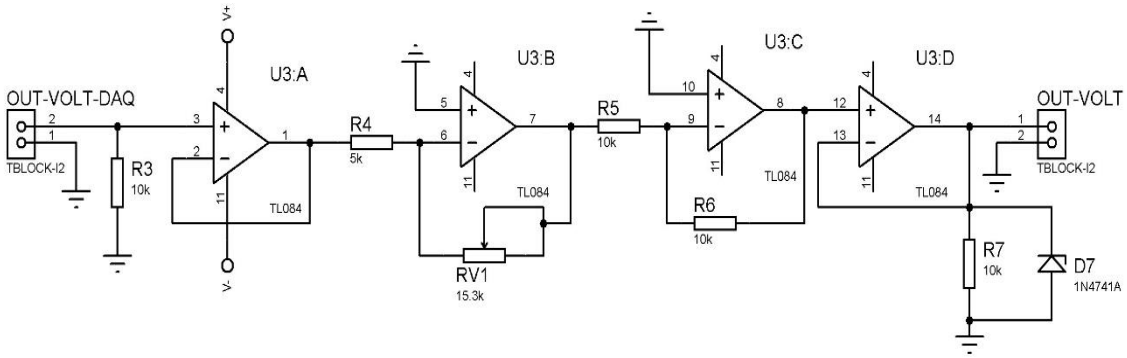
Figura IV-4 Circuito de acondicionamiento de u(t) en corriente (4-20mA)



Fuente: Elaboración Propia

Para la señal de u(t) en tensión se tiene el siguiente diseño electrónico.

Figura IV-5 Circuito de acondicionamiento de u(t) en tensión (0-10V)

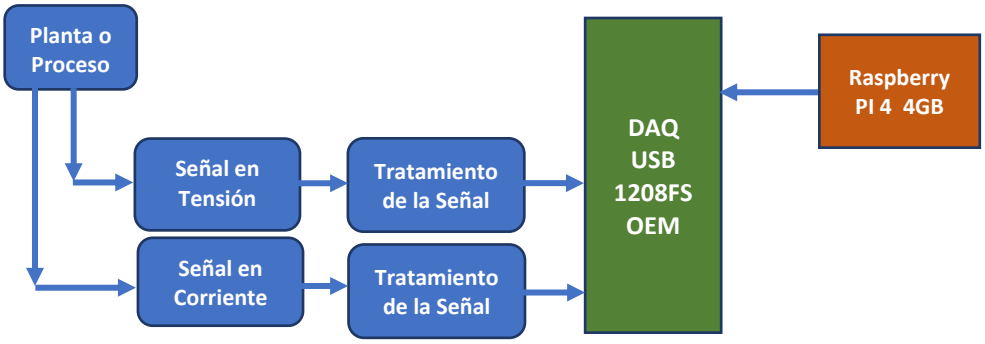


Fuente: Elaboración Propia

4.2.1.2 Etapa de acondicionamiento de la señal de salida del sistema original

Después de la etapa de excitación del sistema se debe sensar esta salida en términos de nivel de voltaje o de corriente según el tipo de variable o el tipo de proceso industrial. Capturada la señal en los niveles antes indicados se diseñó un circuito electrónico de acondicionamiento, con la intención que esta señal pueda ser digitalizada por la tarjeta de adquisición de datos, para luego ser enviada a la Single Board Computer (Raspberry Pi 4 4GB) vía USB para su almacenamiento y posterior identificación.

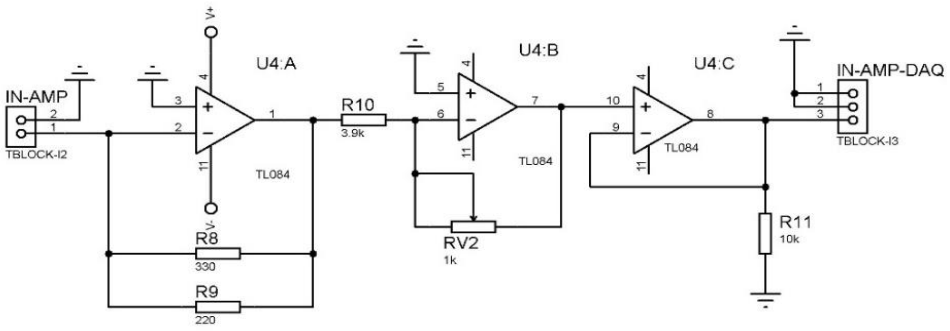
Figura IV-6 Diagrama de bloques de la etapa de tratamiento de la señal



Fuente: Elaboración Propia

Para la señal de salida del proceso en corriente se tiene el siguiente diseño electrónico.

Figura IV-7 Circuito de acondicionamiento de la salida de (4-20mA) a voltaje



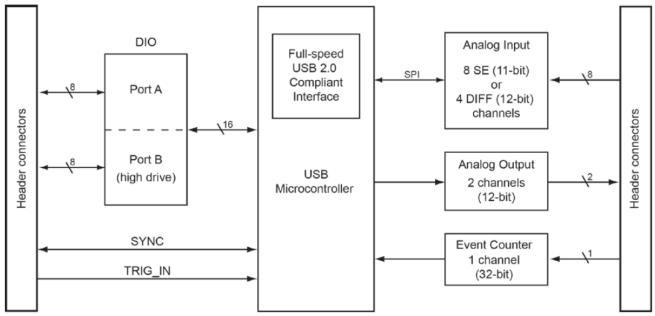
Fuente: Elaboración Propia

Para la señal de salida del proceso en tensión se tiene el siguiente diseño electrónico.

4.2.1.3 Etapa de adquisición de datos.

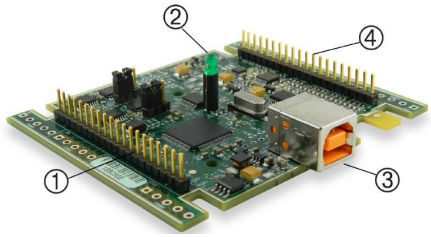
Esta etapa se realizó mediante el uso de una tarjeta de adquisición de datos que trabaja con varios rangos de muestreo, desde los 6.25Ks/S hasta 50Ks/S.

Figura IV-8 Diagrama de Bloques de USB-1208FS-Plus-OEM



Fuente: https://www.mccdaq.com/pdfs/manuals/USB-1208FS-Plus-OEM.pdf

Figura IV-9 Tarjeta de Adquisición de Datos



1. Conector para los pines del 21 al 40. 2. Indicador Led de encendido. 3. Conector USB. 4. Conector para los pines del 1 al 20

Fuente: https://www.mccdaq.com/usb-data-acquisition/USB-1208FS.aspx

La tarjeta de adquisición de datos USB DAQ 1208FS OEM cuenta con driver y librerías para poder adquirir la señal, el fabricante cuenta con librerías para los siguientes lenguajes de programación:

- C\ C++\ C#.
- Matlab.
- Labview
- Visual Basic
- Python.

Así como drivers, como se mencionó en la Tabla II-2 donde se indicó los sistemas operativos en los cuales el fabricante tiene soporte.

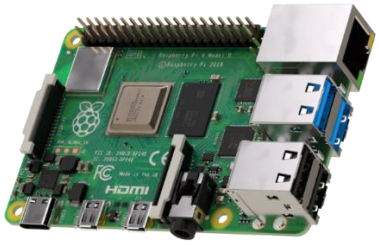
El dispositivo en mención fue configurado con los siguientes parámetros:

- Entradas Analógicas en un rango: 0-10V.
- Salidas analógicas con un rango: 0 a 4.096 V
- Tiempo de muestro con un rango: 10-2000ms.

4.2.1.4 Etapa Single Board Computer (Raspberry Pi 4 4GB).

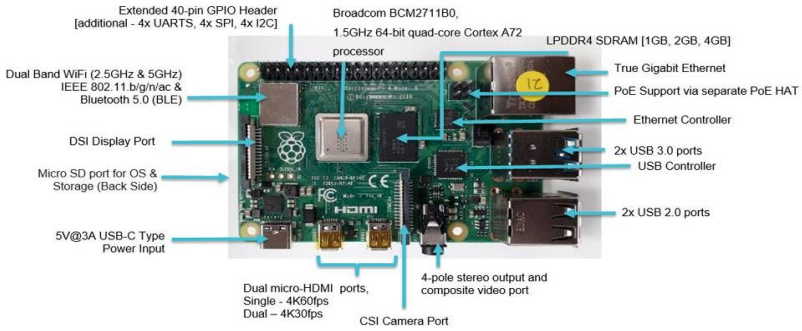
A continuación, se hace referencia a una imagen y un diagrama de bloques del sistema embebido en mención.

Figura IV-10 Raspberry Pi 4 Model B+ 4GB



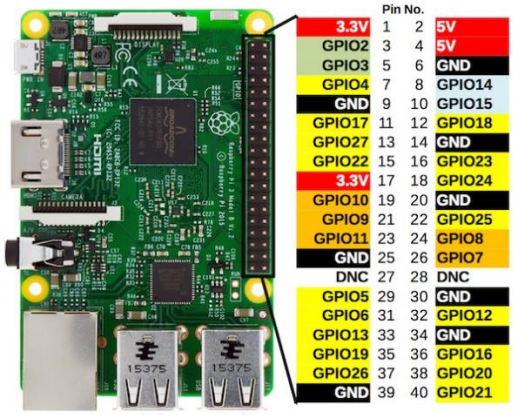
Fuente: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

Figura IV-11 Raspberry Pi 4 4GB Periféricos



Fuente: https://omniretro.com/tecnologia/raspberry-pi-4-especificaciones-caracteristicas/

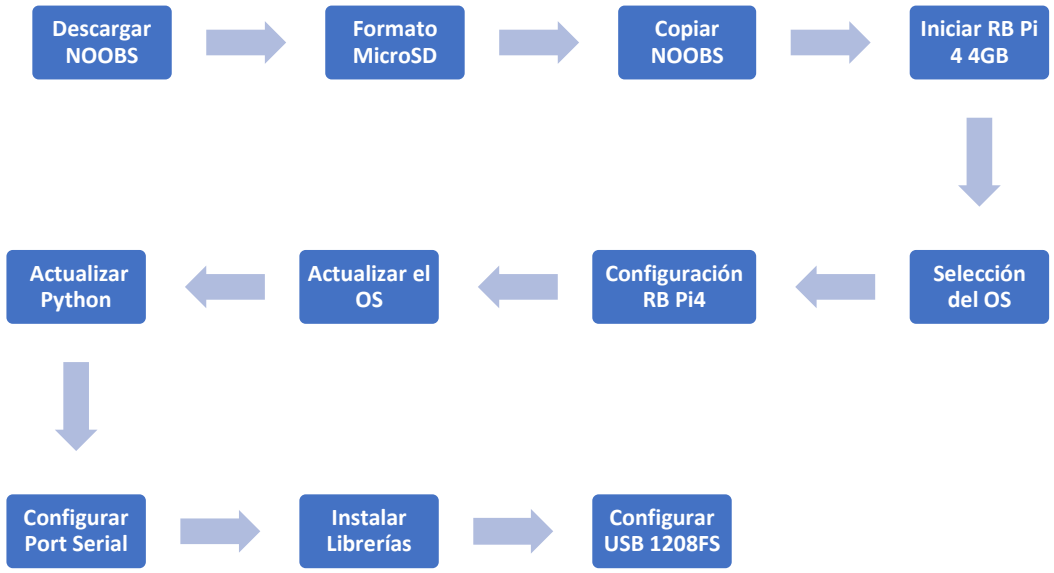
Figura IV-12 GPIO de Raspberry Pi 4 4GB



Fuente: http://www.raspberrypirobotics.com/raspberry-pi-gpio-access/

A continuación, el diagrama de boques de la configuración e instalación del sistema operativo, aplicaciones y actualizaciones de la single board computer.

Figura IV-13 Diagrama de Bloques de configuración, instalación de librerías y actualizaciones

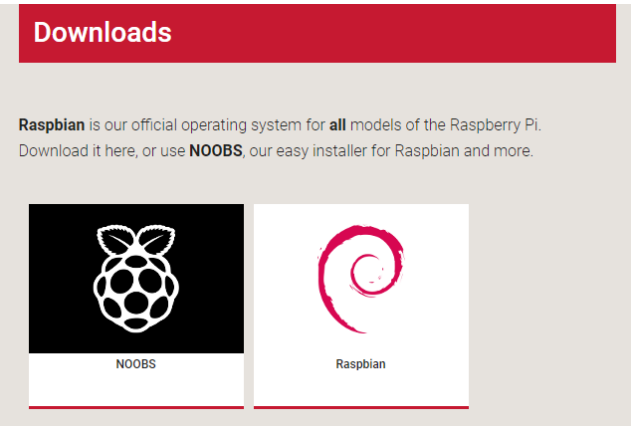


Fuente: Elaboración propia.

A continuación, se describe cada uno de los bloques.

- **Descargar NOOBS**, se realizó la descarga del aplicativo en la siguiente dirección: https://www.raspberrypi.org/downloads/, como se aprecia en la figura IV-14 se procede a seleccionar NOOBS haciendo un click.

Figura IV-14 Selección de NOOBS



Fuente: Elaboración propia

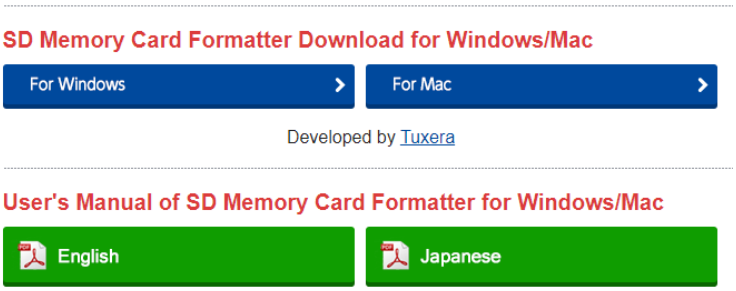
Ahora, se procede a seleccionar NOOBS en el formato ZIP, como se muestra en la siguiente figura.

Figura IV-15 Selección de NOOBS Offline

Fuente: Elaboración propia

- **Formato MicroSDHC**, se realizó la descarga del siguiente aplicativo SDFormat, para el formateo de la MicroSDHC, que se encuentra en la siguiente dirección: https://www.sdcard.org/downloads/formatter/.

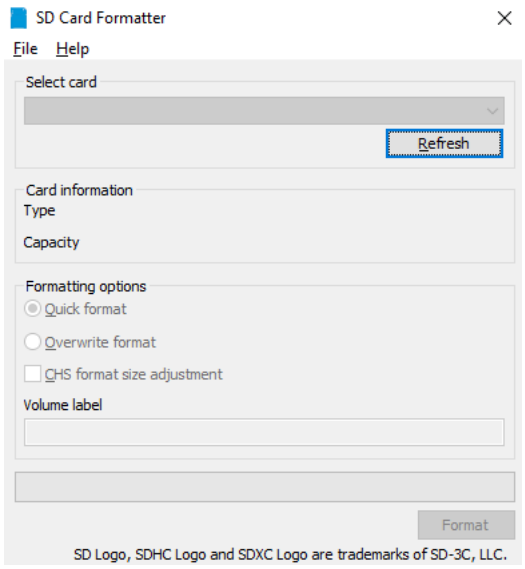
Figura IV-16 Descarga de la aplicación SDHC Card Formatter



Fuente: https://www.sdcard.org/downloads/formatter/

Después de la descarga se procedió a ejecutar el aplicativo, donde se tiene la siguiente pantalla.

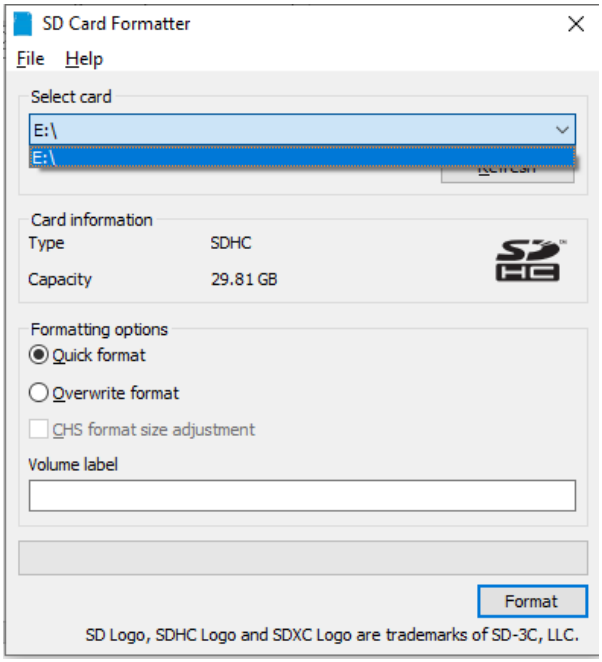
Figura IV-17 Aplicación SDHC Card Formatter



Fuente: Elaboración propia

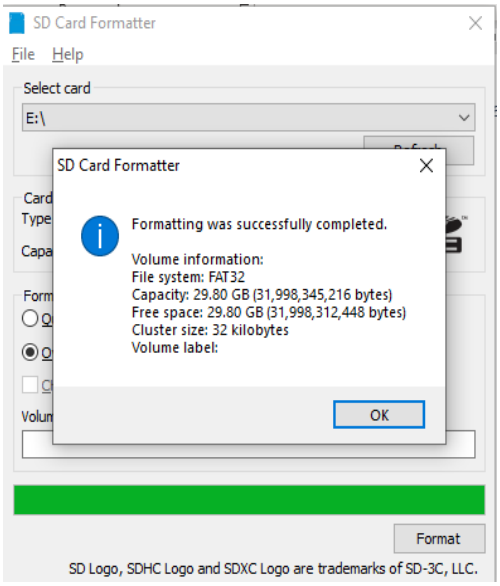
Se colocó la tarjeta MicroSDHC de 32GB, luego se pasó a seleccionar unidad que se va a formatear.

Figura IV-18 Selección de la unidad a formatear



Fuente: Elaboración propia

Figura IV-19 MicroSDHC con formato

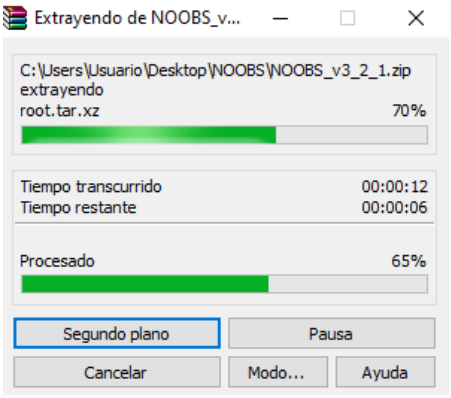


Fuente: Elaboración propia.

Al terminar este proceso, se hace clic en aceptar para continuar con las siguientes fases.

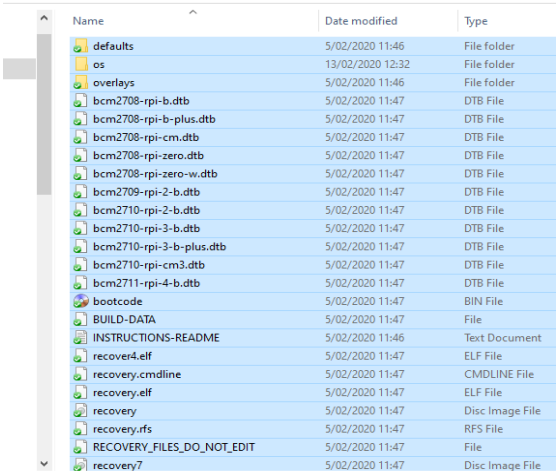
- **Copiar NOOBS**, después del formateo de la memoria se procedió a realizar la copia del archivo NOOBS_v3_3_1.ZIP en la memoria. Para este paso primero se descomprimió el archivo NOOBS_v3_3_1 y luego los archivos se copiaron en la tarjeta micro SDHC.

Figura IV-20 Descomprimir carpeta NOOBS



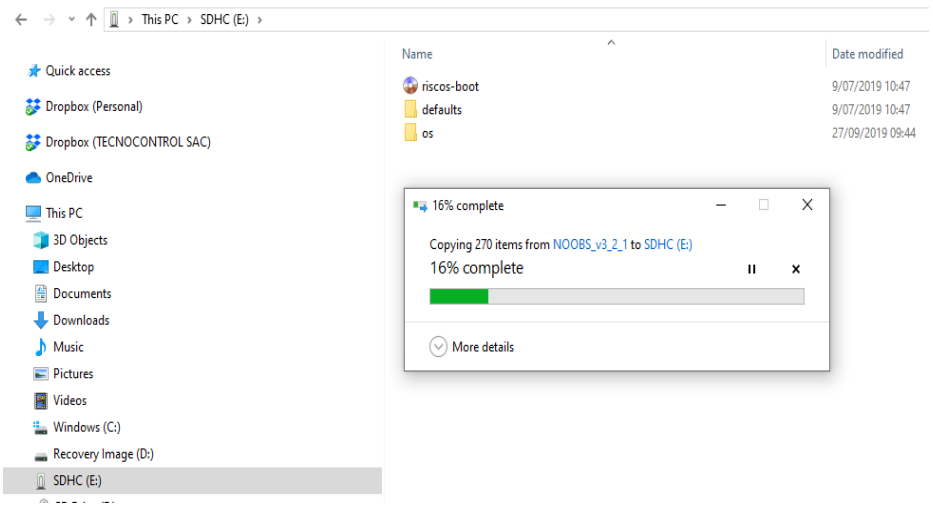
Fuente: Elaboración propia

Figura IV-21 Carpeta NOOBS extraída



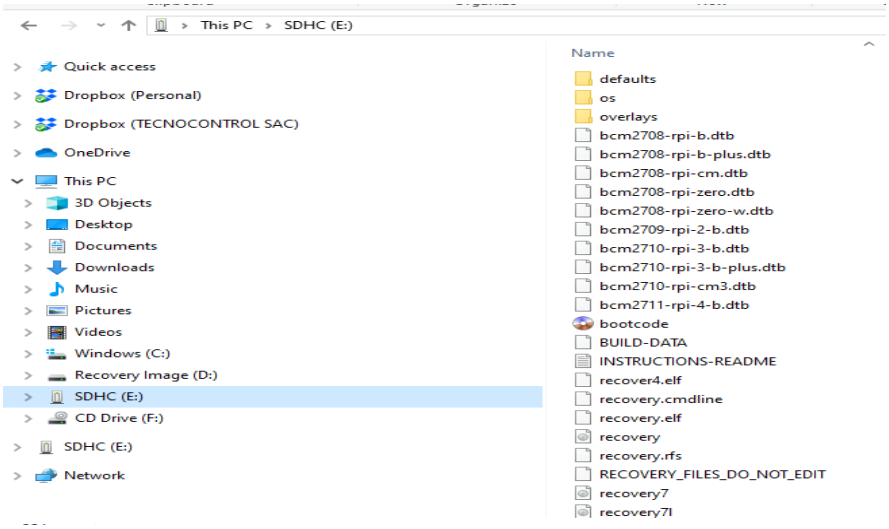
Fuente: Elaboración propia

Figura IV-22 Copiar los archivos de carpeta NOOBS hacia la MicroSDHC



Fuente: Elaboración propia

Figura IV-23 Archivos copiados en la MicroSDHC



Fuente: Elaboración propia

- **Iniciar Raspberry Pi 4 4GB**

  Para el primer arranque de la single board computer se debe tomar en cuenta varias consideraciones, que a continuación se enumeran.

  - Colocar los disipadores en la Raspberry Pi 4 4G.

Figura IV-24 Instalación de los disipadores

Fuente: Elaboración propia

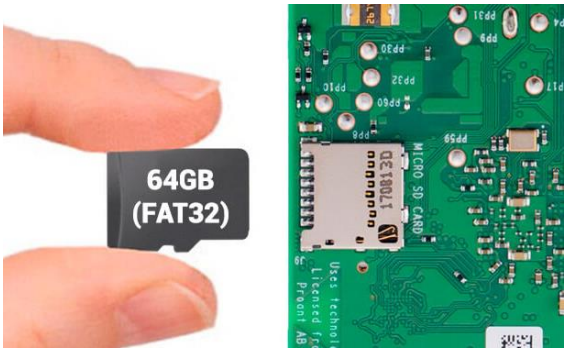
  - Insertar la Raspberry Pi 4 4GB en un case.

Figura IV-25 Instalación del Case

Fuente: Elaboración propia

  - Insertar la Tarjeta MicroSD de 16GB como mínimo en la ranura de la Raspberry que se encuentra en la parte posterior de la tarjeta.

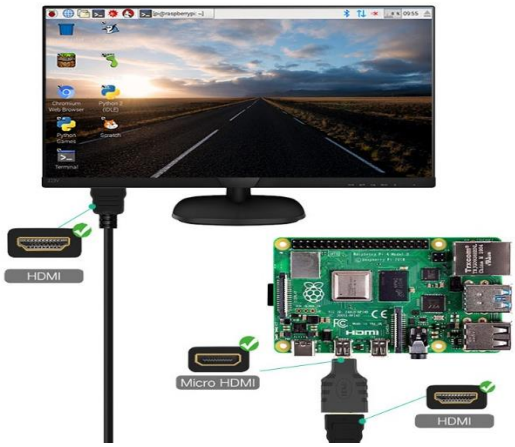
Figura IV-26 Instalación de la MicroSD



Fuente: Elaboración propia

- Conectar el cable Micro-HDMI desde la Raspberry Pi 4 4GB hacia el monitor.

Figura IV-27 Conexión Micro HDMI al monitor



Fuente: Elaboración propia

- Conectar el mouse y teclado hacia la Raspberry Pi 4 4GB.
- Al final colocar el cable de alimentación USB-C a la Raspberry Pi 4, deben tener en cuenta que la alimentación de este sistema requiere una fuente de 3 amperios.

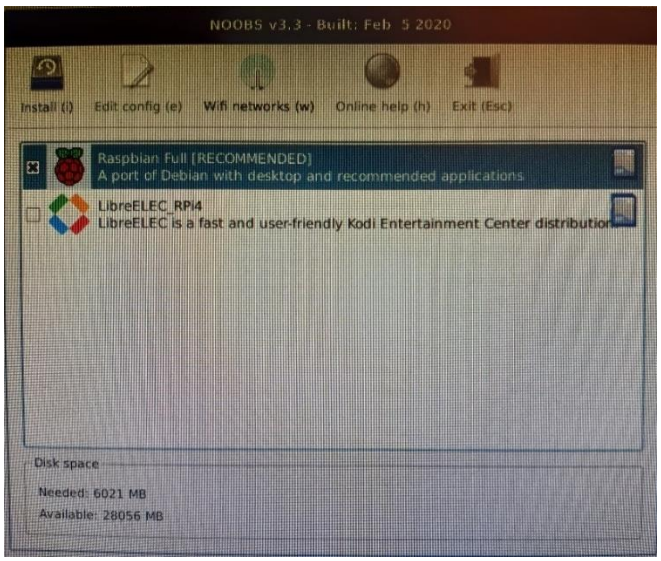
Figura IV-28 Fuente de Alimentación

Fuente: https://comohacer.eu/tienda/fuente-alimentacion-raspberry-pi-4/

- **Selección del Sistema Operativo OS**

  Después de realizar todos los pasos antes descritos, va a salir una pantalla con la siguiente información:

Figura IV-29 Selección del Sistema Operativo



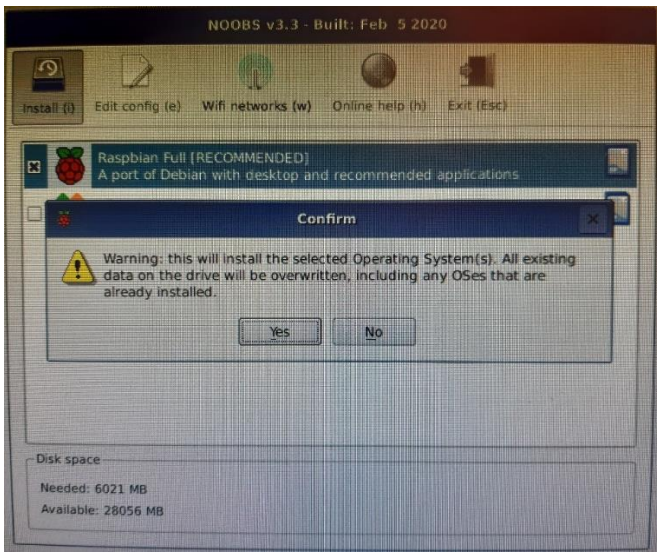
Fuente: Elaboración propia.

Seleccionar el sistema operativo Raspbian y hacer un click en Install.

- **Instalación del Sistema Operativo OS**

  Ahora va a salir una pantalla con la siguiente información:

Figura IV-30 Instalación del Sistema Operativo



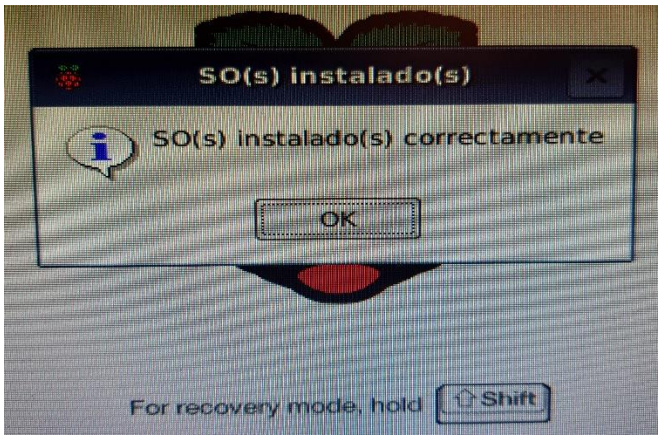
Fuente: Elaboración propia.

Se confirma y empieza la instalación del sistema operativo

Figura IV-31 Inicio de la Instalación

Fuente: Elaboración propia.

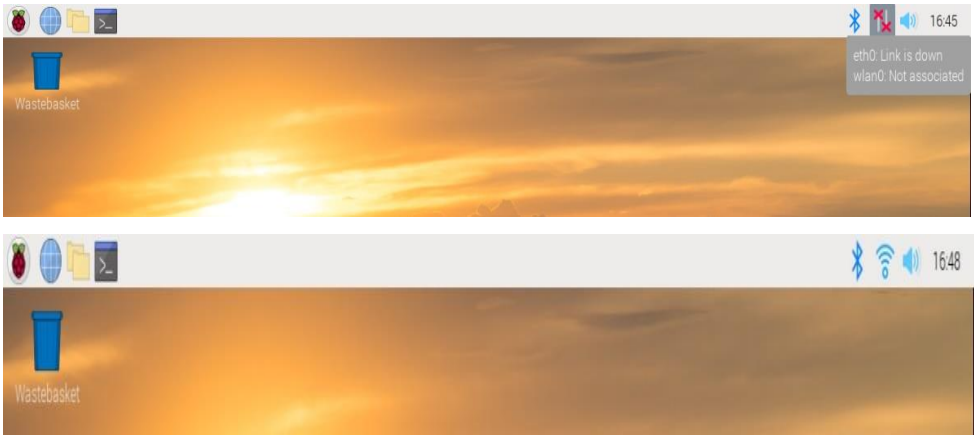
Figura IV-32 Final de la Instalación



Fuente: Elaboración propia.

- **Configuración del Raspberry Pi 4 4GB**
    1. Configuración de la Red Wifi.
        a. Hacer click en el icono de redes inalámbricas.

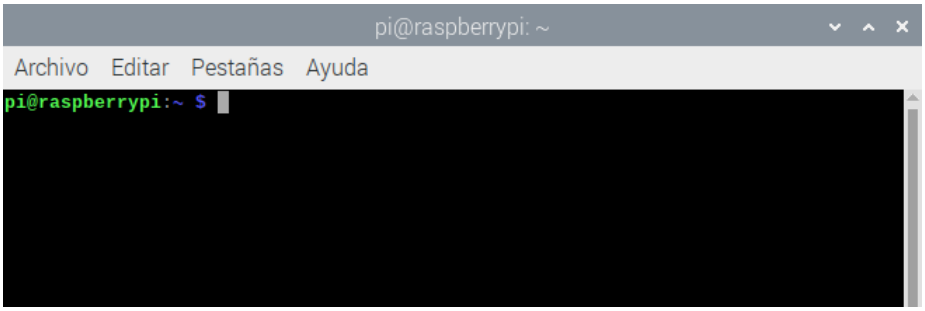
Figura IV-33 Configuración del Wifi



Fuente: Elaboración propia

    2. Ingresar a la línea de comando

Figura IV-34 Línea de Comando

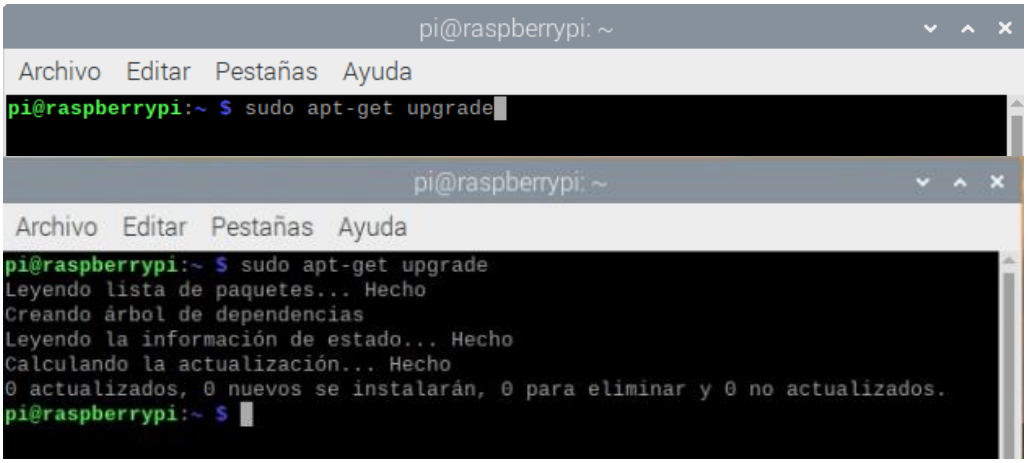


Fuente: Elaboración propia

- **Actualización del OS**

  sudo apt-get upgrade/y

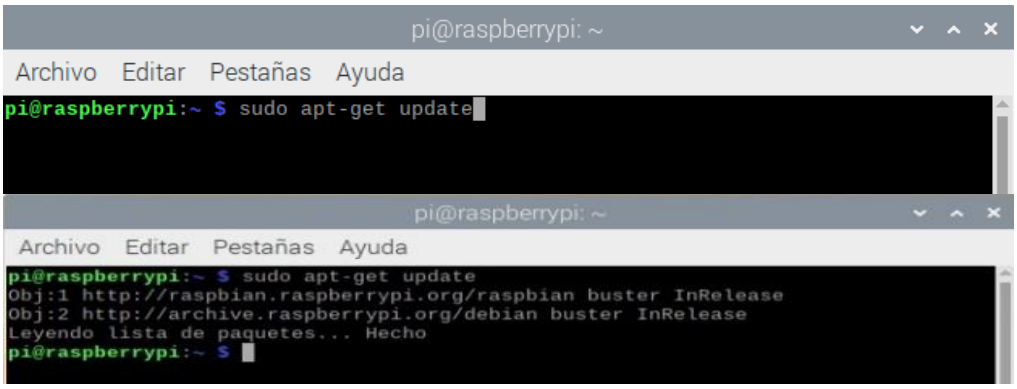
Figura IV-35 Upgrade del OS



Fuente: Elaboración Propia

sudo apt-get update /y

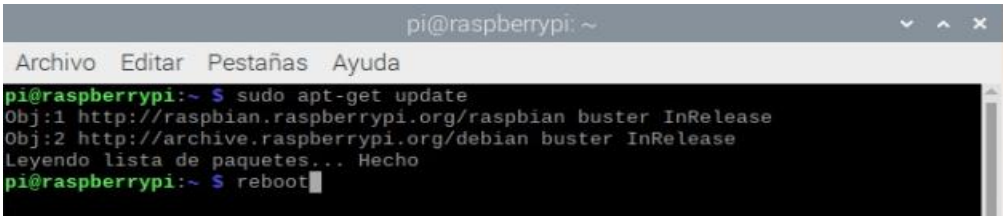
Figura IV-36 Update del SO



Fuente: Elaboración Propia

reboot

Figura IV-37 Reboot del SO



Fuente: Elaboración Propia

- **Actualización del Python**

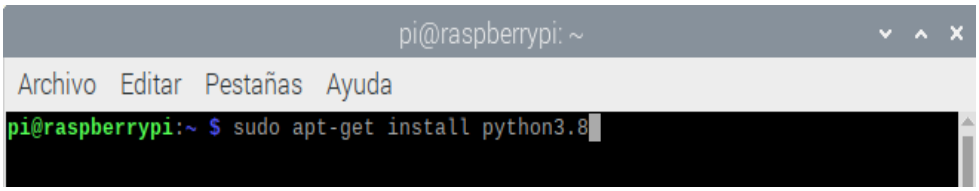
Ingresar a la línea de comando

Figura IV-38 Línea de Comando

Fuente: Elaboración propia

sudo apt-get install python3.8

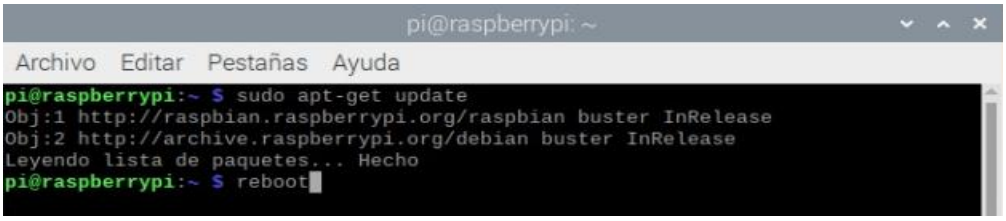
Figura IV-39 Actualización de Python



Fuente: Elaboración propia

Reboot

Figura IV-40 Reboot del SO

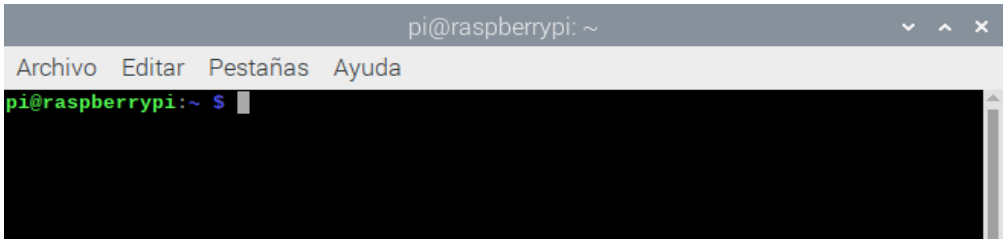


Fuente: Elaboración Propia

- **Configuración del Puerto Serial**,

Ingresar a la línea de comando

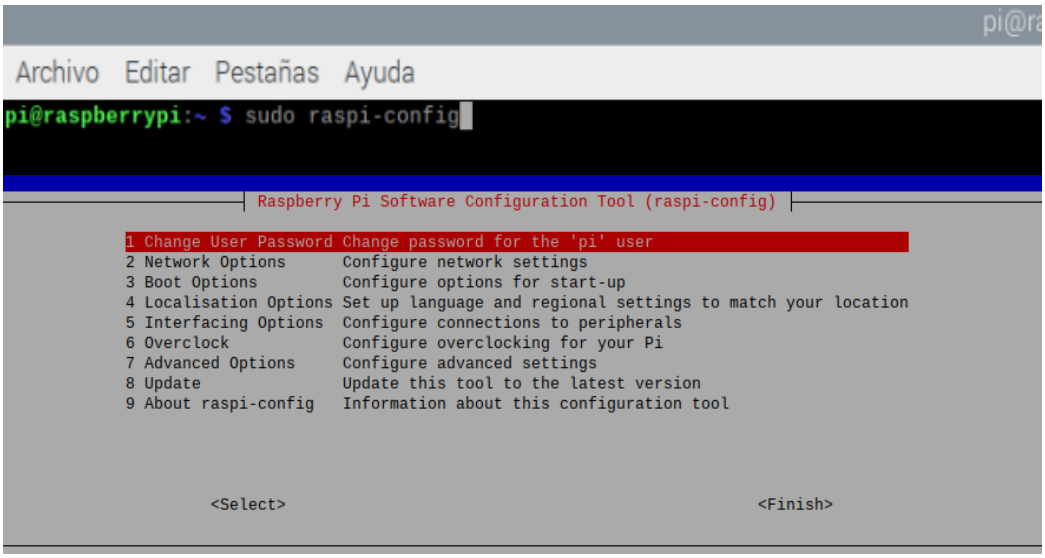
Figura IV-41 Línea de Comando



Fuente: Elaboración propia

sudo raspi-config

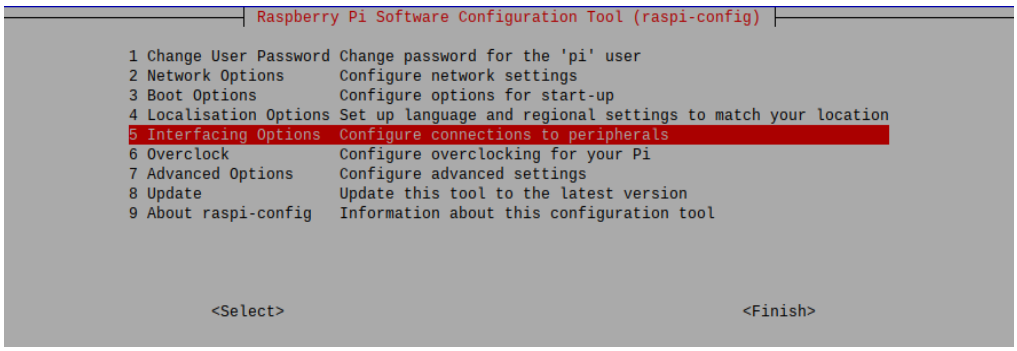
Figura IV-42 Configuración del Raspberry



Fuente: Elaboración propia

option 5 opciones de interface

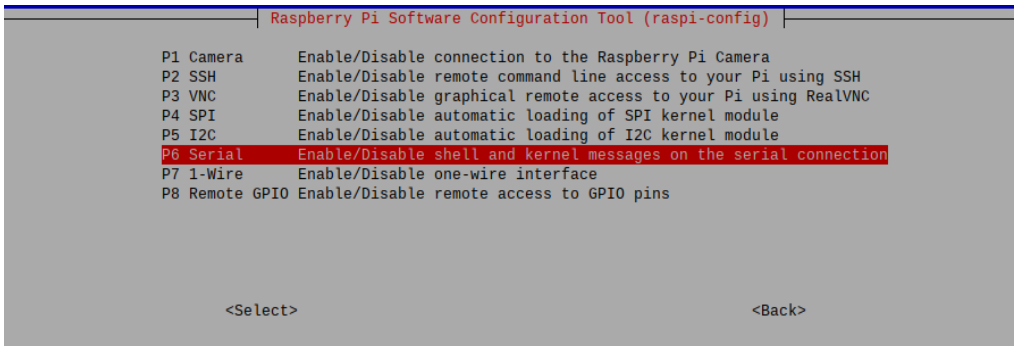
Figura IV-43 Opciones de Interface



Fuente: Elaboración propia

option 6 serial

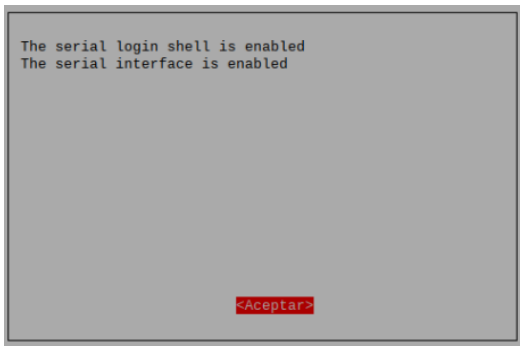
Figura IV-44 Opción Puerto Serial



Fuente: Elaboración propia

Yes

Figura IV-45 Opción Aceptar

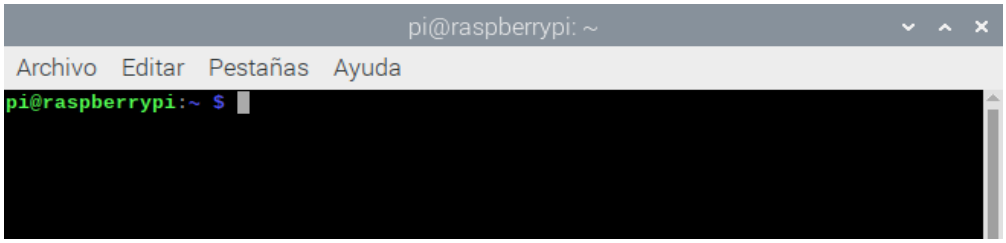


Fuente: Elaboración propia

- **Instalar Librerías**

  Ingresar a la línea de comando

Figura IV-46 Línea de Comando



Fuente: Elaboración propia

pip3 install matplotlib

pip3 install scipy

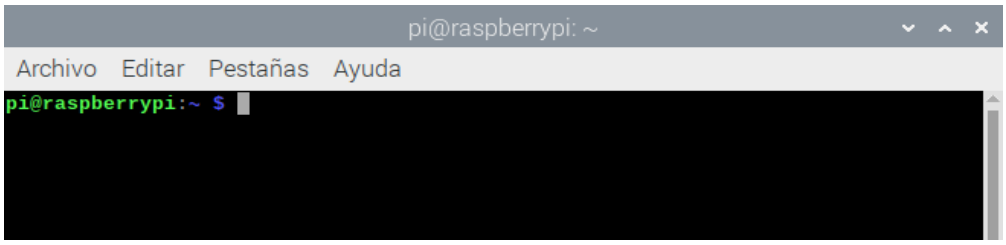
pip3 install numpy

pip3 install sympy

- **Configuración USB 1208FS**

Ingresar a la línea de comando

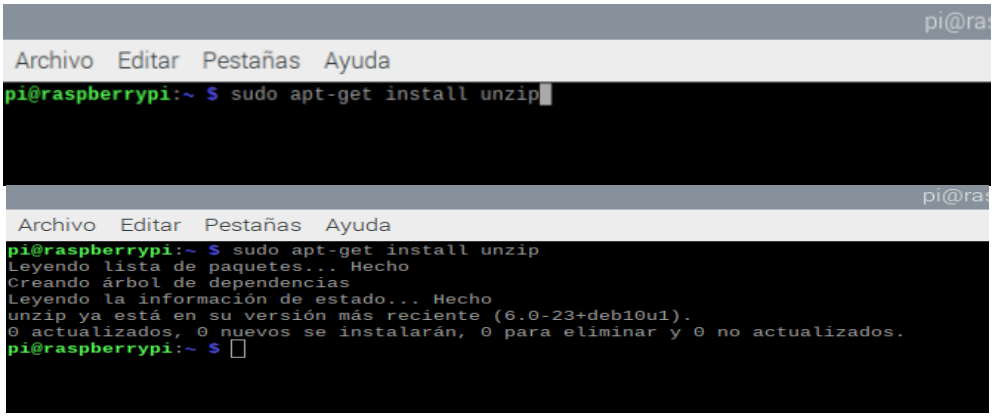
Figura IV-47 Línea de Comando



Fuente: Elaboración propia

sudo apt-get install unzip

Figura IV-48 Instalación de UnZip



Fuente: Elaboración propia

Pasos para la instalación de los Driver de la USB DAQ.

unzip Linux_Drivers-master.zip -d ~pi
sudo apt-get install libusb-1.0-0 libusb-1.0-0-dev
sudo cp 61-mcc.rules /etc/udev/rules.d/99-mcc.rules
git clone git://github.com/signal11/hidapi.git
sudo apt-get install libudev-dev libfox-1.6-dev autotools-dev autoconf automake libtool
cd ~pi/hidapi

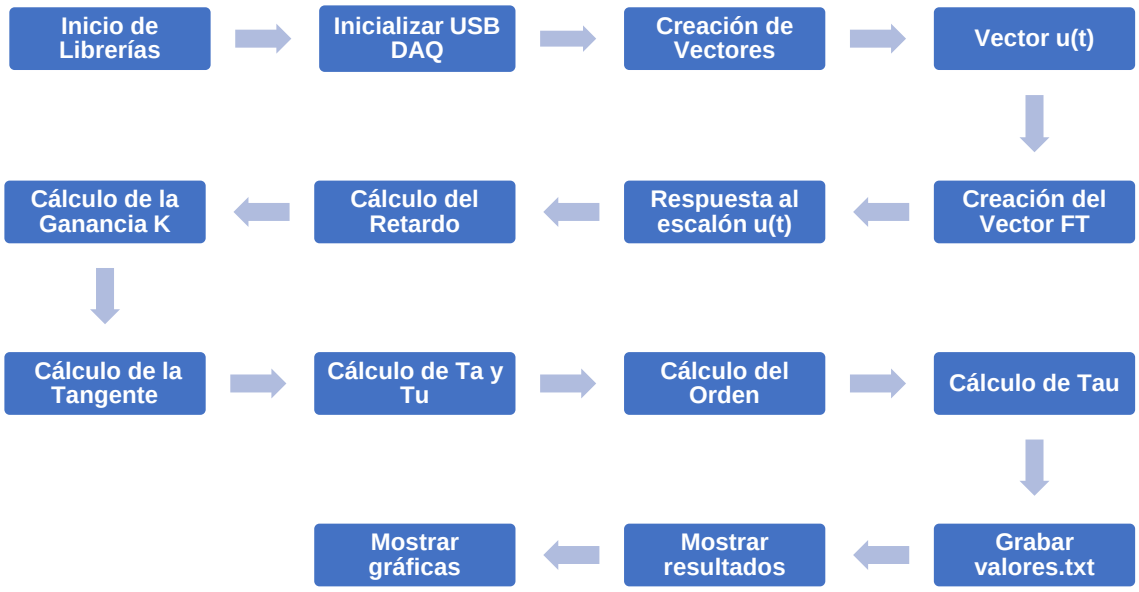
./bootstrap
./configure
  make
  sudo make install
reboot
cd ~pi/usb/mcc-libusb

```
make
sudo make install
sudo ldconfig
cd ~pi/mcc-libusb
./test-usb1608FS
```

4.2.1.5 Algoritmo de Identificación Variable:

A continuación, el diagrama de bloques del algoritmo de identificación realizado en Python:

Figura IV-49 Diagrama de Bloques del Algoritmo de Identificación



Fuente: Elaboración propia.

- Inicio de Librerías.

```
from usb_1208FS import *

import scipy.signal as ssg
import matplotlib.pyplot as plt
import numpy as np
from math import *
import RPi.GPIO as GPIO
import serial
import time
import sympy as sym
```

- Inicializar la USB DAQ

- Inicio de los vectores del Numerador y Denominador.

```
coeff_N = polinomio_N.coeffs()
coeff_D = polinomio_D.coeffs()
N = []
D = []
for i in coeff_N:
    N.append(float(i))

for i in coeff_D:
    D.append(float(i))
```

- Inicio del Vector u(t).

```
#Señal escalón preliminar
escalon_aux = escalon_amp*np.ones_like(np.arange(0,T-delay_sys,Ts))

#Delay del sistema
delay = np.zeros_like(np.arange(0,delay_sys,Ts))

#Señal escalón completa
escalon_u = np.append(delay,escalon_aux)
```

- Construcción de la FT.

```
tf_sistema = ssg.lti(N,D)
```

- Respuesta al escalón u(t).

```
[t_y, vector_muestras, yc] = ssg.lsim(tf_sistema, escalon_u,vector_tiempo)
T = len(vector_muestras)/(1/Ts)
t_y = list(np.arange(0,T,Ts))
```

- Cálculo del retardo.

```
umbral_tau =0.1
max_vector = max(vector_muestras)
min_vector = min(vector_muestras)
rango = max_vector - min_vector
umbral_rango = rango*umbral_tau/100
```

```
umbral_real = umbral_rango + min_vector

index_umbral = 0
cont_index = 0

for i in vector_muestras:
    if i >= umbral_real:
        index_umbral = cont_index
        break
    cont_index = cont_index +1

tau = (index_umbral-1)*Ts
```

- Cálculo de la ganancia.

```
index_moda = 0
aux_moda = 0
aux_moda_anterior = 0
aux_umbral_k = round(0.01*len(vector_muestras))
array_k = range(len(vector_muestras)-aux_umbral_k,len(vector_muestras),1)

for i in array_k:
    for j in array_k:
        if vector_muestras[j] == vector_muestras[i]:
            aux_moda = aux_moda+1

    if aux_moda > aux_moda_anterior:
        index_moda = j

k_aux = vector_muestras[index_moda]/escalon_amp
```

- Cálculo de la tangente.

```
vector_normalizado = vector_muestras
primer_diff = np.diff(vector_normalizado)/np.diff(t_y)
segundo_diff = np.diff(primer_diff)/np.diff(t_y[0:len(t_y)-1])
aux_inflex = np.where(segundo_diff < 0)
inflex = aux_inflex[0][0]
A = primer_diff[inflex]*t_y[inflex] - vector_normalizado[inflex];
tanget = []
for i in t_y:
    tanget.append(primer_diff[inflex]*i - A)

tan_k = 0
tan_zero = 0
index_tan_k = 0
index_tan_zero = 0
for i in tanget:
    if i>=escalon_amp*K:
        tan_k = i
        break
    index_tan_k = index_tan_k + 1

for i in tanget:
    if i>=0:
        tan_zero = i
        break
    index_tan_zero = index_tan_zero + 1
```

```
seg_lim_k_tan = index_tan_k/(1/Ts)
y_tan = list(np.arange(0,seg_lim_k_tan,Ts))
x_tan = tanget[0:index_tan_k]
```

- Cálculo de Ta y Tu.

```
Ta = (index_tan_k - index_tan_zero)/(1/Ts)
Tu = index_tan_zero/(1/Ts)
tu_ta = Tu/Ta
```

- Cálculo de Tau

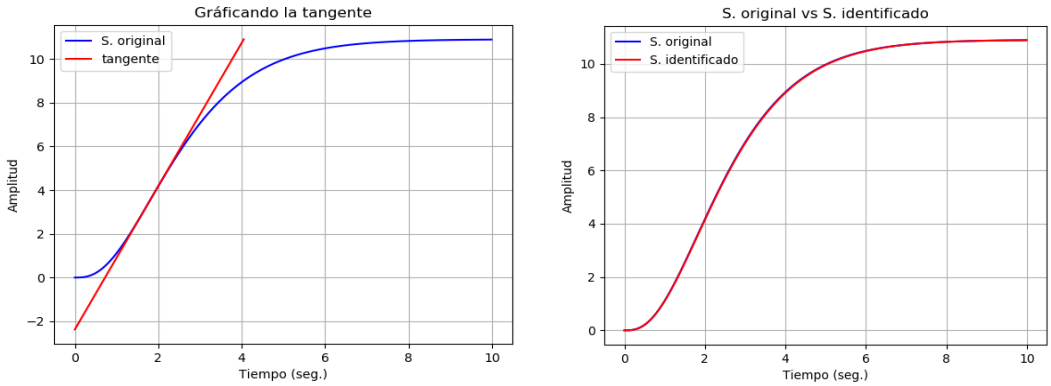
```
strejc_tau1 = [1,2.718,3.695,4.463,5.119,5.699]
strejc_tau2 = [0,0.282,0.805,1.425,2.1,2.811]
if index_strejc_n != 0:
    tau_aux1 = Ta/strejc_tau1[index_strejc_n]
    tau_aux2 = Tu/strejc_tau2[index_strejc_n]
else:
    tau_aux1 = tau_aux2 = Ta/1
tau_calc = round((tau_aux1+tau_aux2)/2,3)
```

4.2.1.6 Etapa de visualización de gráficas y resultados

El sistema embebido cuenta con una interfaz gráfica para la visualización de la identificación del sistema. Para esta etapa se usó varias librerías como MatplotLib.

La primera librería permite la visualización de los diferentes tipos de gráficos (Líneas, Barras, circular, entre otras). En los diferentes planos (t, xy, s y z). La segunda librería sirvió para el diseño gráfico (presentación de la aplicación en el sistema embebido) del sistema de identificación donde se muestran las gráficas y los resultados de la identificación.

Gráfica IV-1 Gráficas de la Identificación del sistema



Fuente: Elaboración Propia

4.2.2 Etapa de emulación del sistema

Está etapa de emulación también cuenta con una tarjeta electrónica que realiza el acondicionamiento de la señal de la función de transferencia o sistema emulado. En esta etapa se ingresan los coeficientes del numerador y denominador, así como también el tiempo de muestro, después de esto, se ejecuta el algoritmo para la emulación del sistema cuya salida del proceso se enviado a la tarjeta de adquisición de datos vía USB. Luego la señal pasa por una etapa de acondicionamiento a niveles de tensión o corriente según sea el caso.

Sin embargo, para que el equipo en cuestión pueda tener una señal de salida es necesaria la señal de entrada que excite al sistema. Por eso, es necesaria una etapa que convierta la señal de entrada por parte de controlador industrial a señales de tensión o de corriente que sean admitidas por la tarjeta de adquisición.

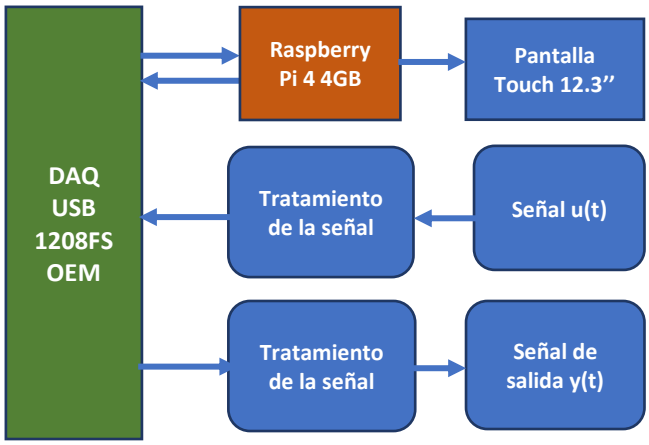
Todas las partes de sistema implementado comparten entre sí varios tipos de tecnología y serían las siguientes:

- Etapa de acondicionamiento (tratamiento) de la señal escalón
- Etapa de acondicionamiento (tratamiento) de la señal del proceso

- Tiempo de muestro.
- Single Board Computer (Raspberry Pi 4 4GB).
- Etapa de visualización de las gráficas y resultados

De tal manera que en los siguientes diagramas de bloques se muestra el sistema embebido implementado.

Figura IV-50 Diagrama de bloques de la etapa de Emulación



Fuente: Elaboración Propia

4.2.2.1 Etapa de acondicionamiento de la señal de entrada u(t)

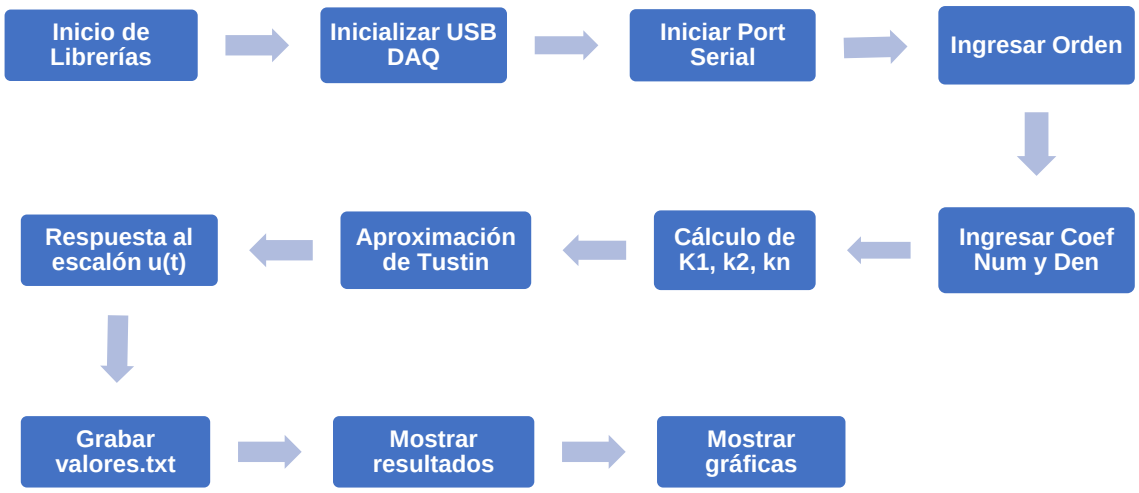
Se consideró que para obtener la respuesta del sistema este debe excitarse con una señal de entrada u(t), en este caso vendría a ser la señal de un controlador industrial. En tal sentido, mediante el diseño de una tarjeta electrónica se procede a adaptar dicha señal para ser capturada por la tarjeta de adquisición de datos, con la finalidad de excitar a la función de transferencia emulada y generar la salida y(t).

4.2.2.2 Etapa de acondicionamiento de la señal de salida y(t)

Una vez los datos (coeficientes del numerador y del denominador) del sistema a ser emulado han sido ingresados correctamente o el proceso de identificación de la planta concluyó con éxito se tiene todo listo para el proceso de emulación. Una vez que la señal u(t) es generada por el controlador y es leída por la tarjeta

de adquisición de datos mediante una etapa de acondicionamiento de la señal, el sistema genera una señal de salida a niveles de tensión o corriente según el tipo de proceso. Para emular todo el proceso también se ingresa el tiempo de muestreo, así como el tiempo final de la emulación. Como se mencionó anteriormente el método de aproximación implementado es la aproximación de Tustin, a continuación, el diagrama de bloques del algoritmo de emulación:

Figura IV-51 Diagrama de bloques del Algoritmo de Emulación



Fuente: Elaboración propia.

4.2.3 USB DAQ Tarjeta de Adquisición de Datos

Como se mencionó en el apartado 4.2.1.3 se usó la Tarjeta de adquisición de datos de Computing Measurement, con las siguientes consideraciones:

- Entradas Analógicas en un rango: 0 a 10V.
- Salidas analógicas con un rango: 0 a 4.096 V
- Tiempo de muestro con un rango: 10-2000ms.

4.2.4 Single Board Computer

Como se mencionó en el apartado 4.2.1.4. se usó la Raspberry Pi 4 de 4GB. Para el intercambio de datos entre la Raspberry Pi 4 y la tarjeta de adquisición se realizó mediante una comunicación USB.

4.2.5 Pantalla Touchscreen 12,3'' HDMI

La Raspberry Pi 4 presenta las gráficas de la identificación del sistema, emulación. Así mismo, presentará diversos botones y comandos de entrada para el seteo y configuración de las funcionalidades del equipo. Todo ello, será mostrado en una pantalla de 12,3" pulgadas de dimensión con características touch para la facilidad de manejo por parte del usuario. Así mismo, debe considerarse que se utiliza el medio de comunicación HDMI, soportado con la Raspberry, dotando al equipo de una calidad de imagen superior a la provista comúnmente por VGA.

Figura IV-52 Industrial Touch-Panel

Fuente: https://www.linux.com/tutorials/6-industrial-touch-panel-computers-based-raspberry-pi/

4.3 Población y muestra

4.3.1 Delimitación

A continuación, se mencionan las delimitaciones de la presente investigación:

- La presente investigación está dada para plantas, sistemas y procesos industriales que cumplan con la teoría de SISO (Single Input Single Output), esto quiere decir que se trata de procesos de una sola variable de entrada y salida.

- En el proceso de identificación se debe tener en cuenta el tempo de reacción del sistema a identificar, se trata de sistemas que tengan tiempos de reacción de 40ms como mínimo.

- Para el proceso de emulación de la función de transferencia se hace uso de la aproximación de Tustin, por tener una mejor estabilidad.

- El tiempo de muestreo para la presente investigación, se trabajó con tiempos de muestro desde 10ms a 1000ms, también tener en cuenta el tiempo de adquisición y procesamiento que será hasta 3600 segundos.

4.3.2 Ubicación

La presente investigación tiene como ubicación cualquier sistema o proceso de tipo SISO donde se pueda tener acceso a las variables de entrada y salida, teniendo en cuenta las consideraciones de adquisición y acondicionamiento de las señales, con la intención de tener una señal que contenga un nivel mínimo de ruido y poder aplicar los métodos propuestos por esta tesis.

4.3.3 Población

La población para esta investigación son las plantas industriales de tipo SISO, esto quiere decir para procesos o plantas de tienen una variable de entrada y una variable de salida, que se encuentren en los niveles de tensión y corriente standard y cuyo tiempo de respuesta sea mayor a 50ms. La población para esta investigación también poder ser sistemas emulados de los procesos o sistemas.

4.3.4 Tamaño de la muestra

Para la presente investigación el tamaño de la muestra es variable, se configura de acuerdo con el tipo de proceso o sistema a identificar o a emular.

Se tienen vectores con muestras para los procesos de validación entre 2000 a 8000 muestras, el sistema de identificación, así como el de emulación tienen una capacidad de hasta 36000 muestras.

4.4 Técnicas e instrumentos de recolección de la información.

4.4.1 Técnicas de recolección de la información.

- Tarjeta de adquisición de datos USB DAQ 1208FS, para la captura de señales.
- Se implementó vectores con sistemas de diferentes ordenes en Matlab.
- Se implementó vectores con sistemas de diferentes ordenes en Python.

4.4.2 Instrumentos de recolección de la información.

Los instrumentos utilizados en la presente investigación fueron los siguientes:

- Software de Simulación de circuitos electrónicos (Proteus)
- Software de Programación y simulación (Matlab)
- Software de Programación y simulación (Python)

- Software de Programación de Controladores Lógicos (Tia Portal)
- Equipo de medición de señales en tensión Multímetro (Fluke 111).
- Equipo de medición y generación de señales en corriente de 4-20mA (Process Calibrator HY H873).
- Equipo de Medición en tensión Osciloscopio (Hantek DSO5102B).

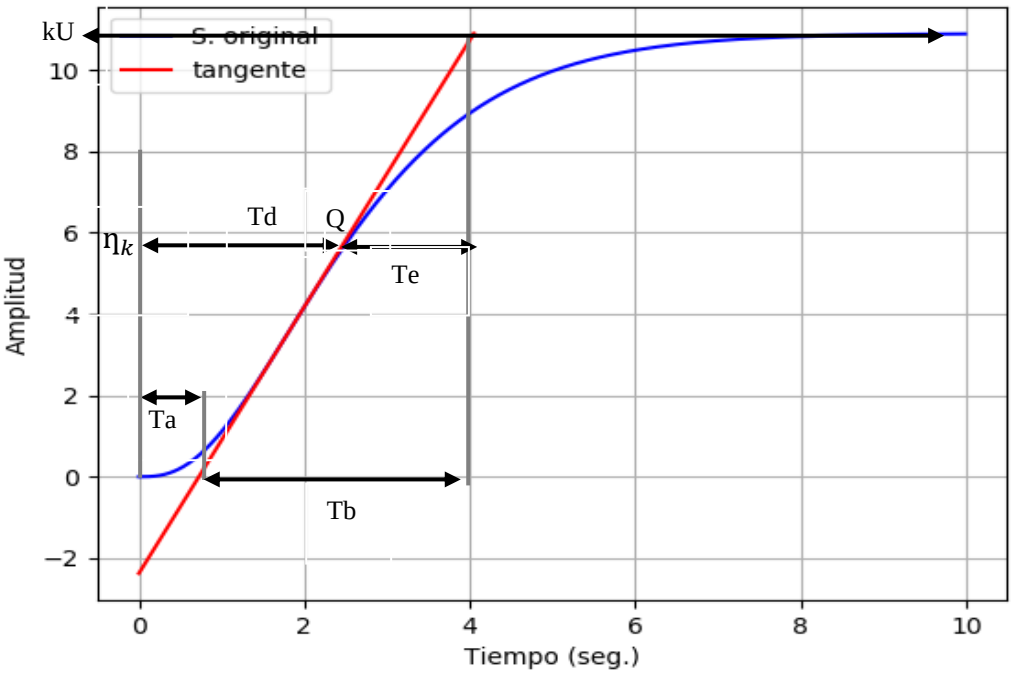
4.5 Procedimiento de recolección de datos

Para la presente investigación el procedimiento de recolección de datos fue mediante el uso de diferentes software de simulación como de programación, así como el uso de diferentes instrumentos de medición y generación de señales, con lo cual se tuvo todos los datos tanto para la validación del proceso de identificación así como para el proceso de emulación.

4.6 Procedimiento estadístico y análisis de datos

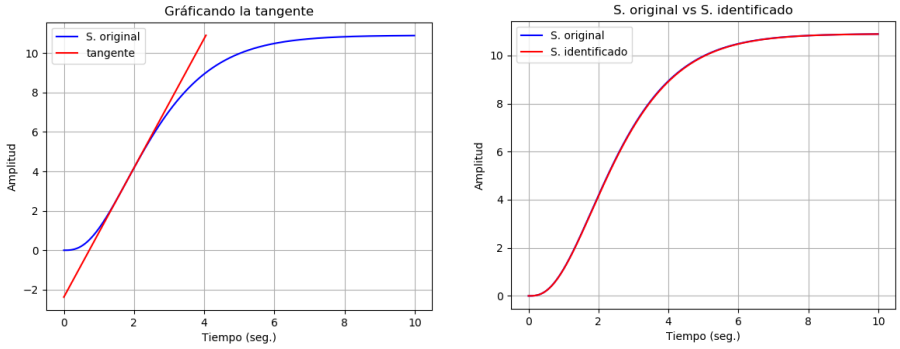
Para el proceso de identificación de los sistemas de tipo SISO de polos múltiples se aplicara el método de Strejc, que consiste en un método grafico donde se traza una línea recta de pendiente máxima, la cual va a estar superpuesta sobre el área de pendiente del sistema a identificar, de tal forma que el parámetro T_u se obtiene con el corte del eje de abscisas y el valor del parámetro Ta se obtiene con el corte de una paralela al eje de abscisas en el punto donde la respuesta se encuentra estable.

Gráfica IV-2 Método Strejc



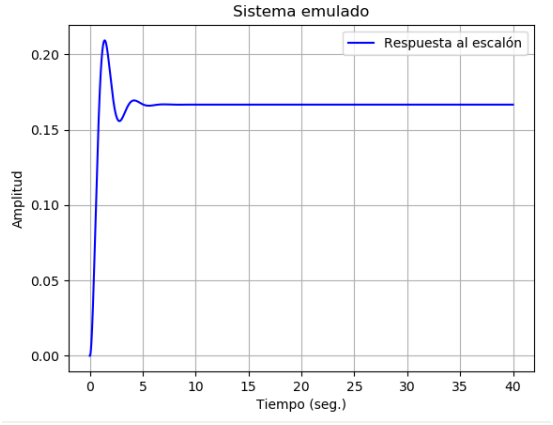
Fuente: Elaboración Propia

Gráfica IV-3 Identificación del sistema



Fuente: Elaboración Propia

Gráfica IV-4 Sistema emulado



Fuente: Elaboración Propia

Tabla IV-1 Coeficientes de Strejc

N	T_a/τ	T_u/τ	T_u/T_a
1	1	0	0
2	2.718	0.282	0.104
3	3.695	0.805	0.218
4	4.463	1.425	0.319
5	5.119	2.100	0.410

Fuente (Angel, Martínez Bueno; Jorge, Pomares, 2011): Identificación experimental de sistemas

V RESULTADOS

5.1 Resultados descriptivos

A continuación, el detalle de los resultados indicando las métricas y sus validaciones.

Se tienen las siguientes variables:

- X1: Desarrollo del algoritmo de identificación de un sistema SISO hasta Orden 5, haciendo uso de herramientas de software de tipo open source.
- X2: Desarrollo del algoritmo de emulación de un sistema SISO hasta Orden 5, haciendo uso de herramientas de software de tipo open source.
- Y: Desarrollo de un sistema portátil. Implementación de los algoritmos de identificación y emulación en un sistema embebido.

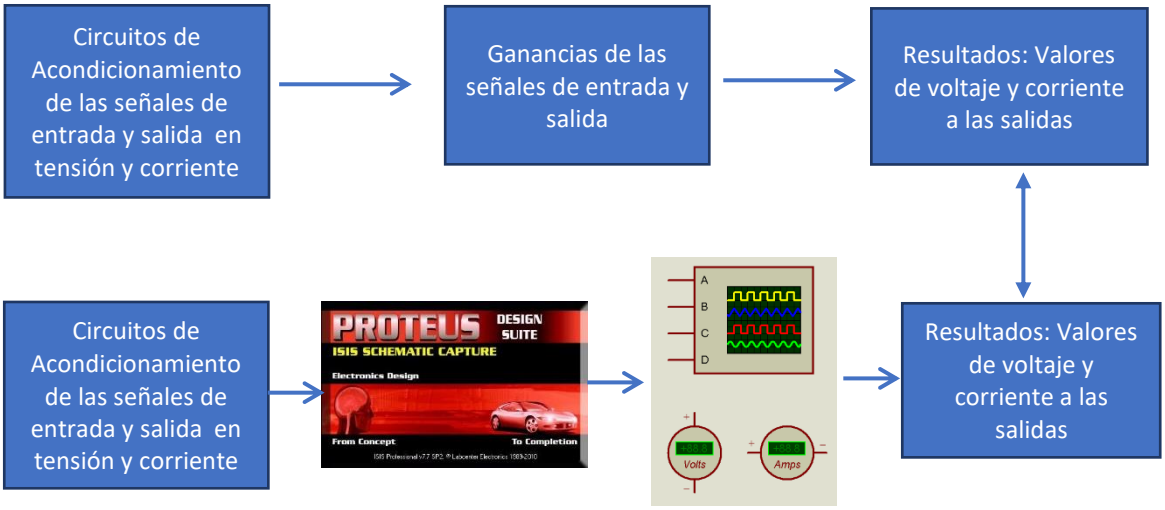
5.1.1 Desarrollo de las interfaces de entrada y salida:

Desarrollo de las interfaces de acondicionamiento de las señales de entrada y salida para la identificación y emulación de un sistema SISO, las métricas para la validación son las siguientes:

a) Los rangos de entrada y salida analógica:
   - Nivel de tensión: 0-10v.
     i. Con un error menor a 5%
   - Nivel de corriente: 4-20mA.
     i. Con un error menor a 5%

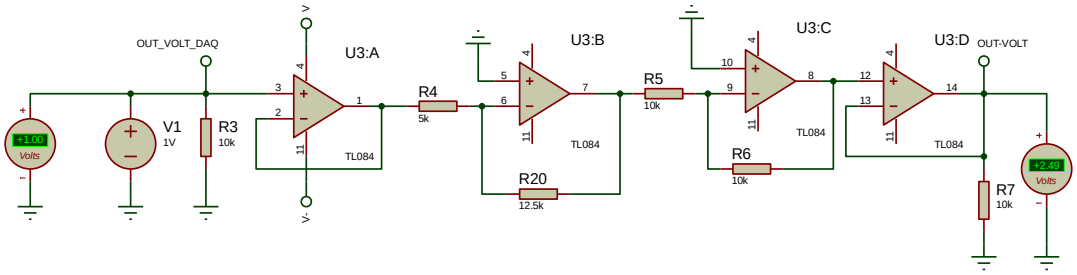
A continuación, el diagrama de bloques para la validación de X1:

Figura V-1 Diagrama de Bloques Validación



Fuente: Elaboración Propia

Figura V-2 Simulación del circuito de Salida 0-4V a 0-10V



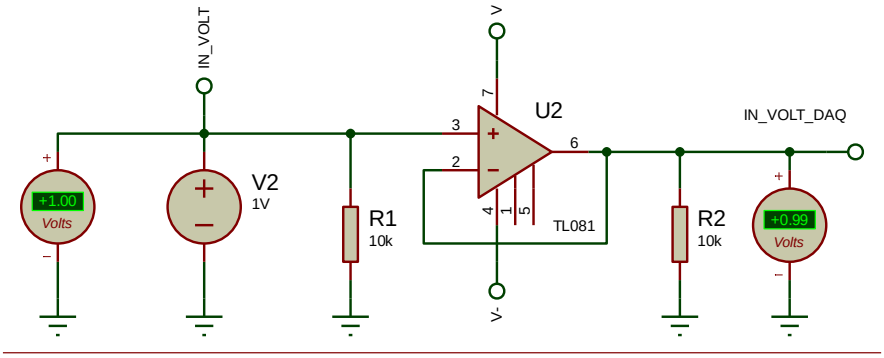
Fuente: Elaboración propia

Tabla V-1 Tabla de datos simulados de salida 0-4V a 0-10V

Tensión de Entrada (V) Simulación	Tensión de Salida (V) Simulación	Valor esperado (V)	Error %
0.50	1.24	1.25	0.806
1.00	2.49	2.50	0.402
1.50	3.74	3.75	0.267
2.00	4.99	5.00	0.200
2.50	6.24	6.25	0.160
3.00	7.49	7.50	0.134
3.50	8.74	8.75	0.114
4.00	9.99	10.00	0.100

Fuente: Elaboración propia

Figura V-3 Simulación del circuito de Entrada 0-10V



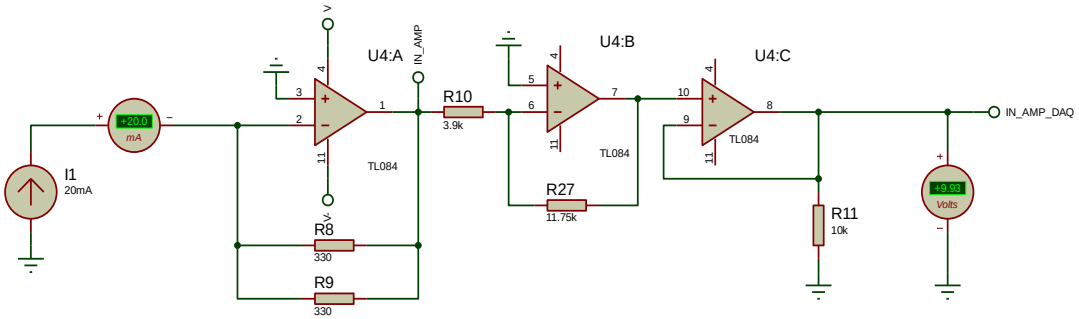
Fuente: Elaboración propia

Tabla V-2 Tabla de datos simulados de entrada de 0-10V

Tensión de Entrada (V) Simulación	Tensión de Salida (V) Simulación	Valor esperado (V)	Error %
0.50	0.49	0.50	2.041
1.00	0.99	1.00	1.010
1.50	1.49	1.50	0.671
2.00	1.99	2.00	0.503
2.50	2.49	2.50	0.402
3.00	2.99	3.00	0.334
3.50	3.49	3.50	0.287
4.00	3.99	4.00	0.251
4.50	4.49	4.50	0.223
5.00	4.98	5.00	0.402
5.50	5.49	5.50	0.182
6.00	5.98	6.00	0.334
6.50	6.49	6.50	0.154
7.00	6.99	7.00	0.143
7.50	7.48	7.50	0.267
8.00	7.99	8.00	0.125
8.50	8.49	8.50	0.118
9.00	8.99	9.00	0.111
9.50	9.49	9.50	0.105
10.00	9.99	10.00	0.100

Fuente: Elaboración propia

Figura V-4 Simulación del circuito de 4-20mA a 0-10V



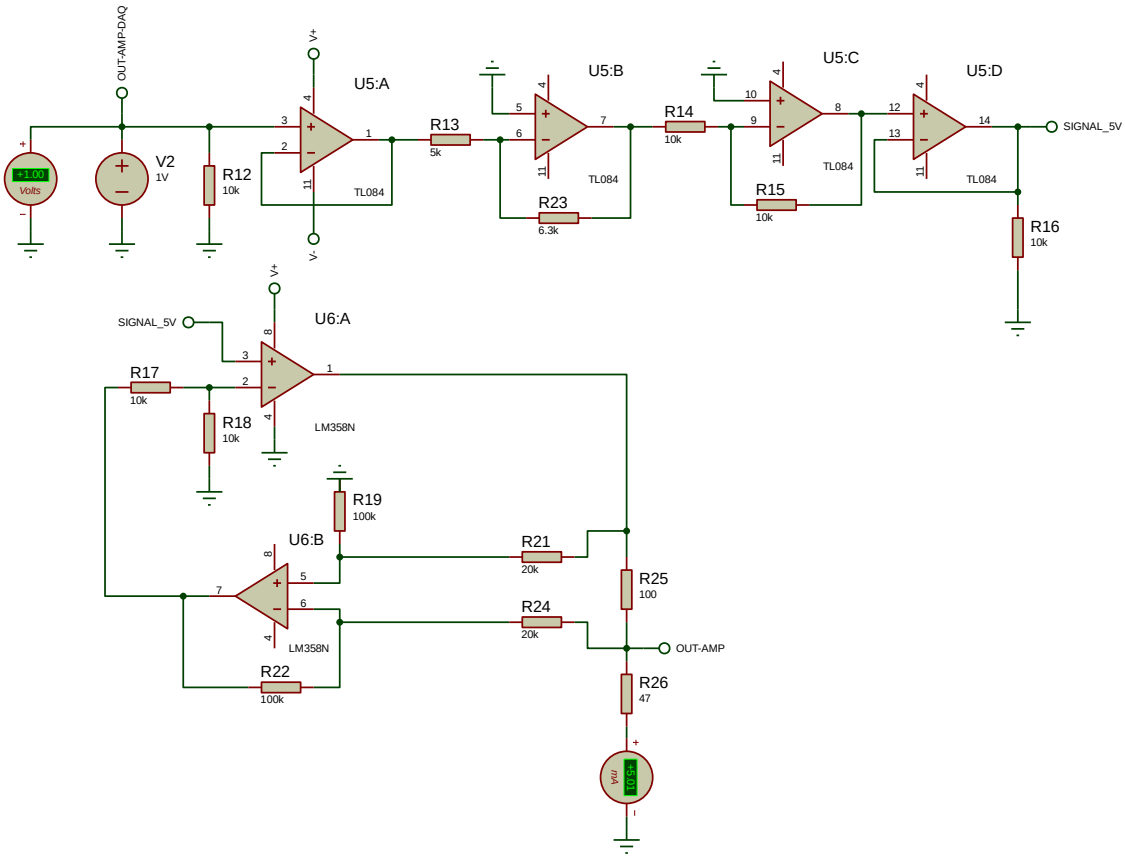
Fuente: Elaboración propia

Tabla V-3 Tabla de datos simulados de 4-20mA a 0-10V

Corriente de Entrada (mA) Simulación	Tensión de Salida (V) Simulación	Valor esperado (V)	Error %
4.00	1.98	2.00	1.010
6.00	2.97	3.00	1.010
8.00	3.97	4.00	0.756
10.00	4.96	5.00	0.806
12.00	5.95	6.00	0.840
14.00	6.95	7.00	0.719
16.00	7.94	8.00	0.756
18.00	8.94	9.00	0.671
20.00	9.93	10.00	0.705

Fuente: Elaboración propia

Figura V-5 Simulación del circuito de 0-4V a 4-20mA



Fuente: Elaboración Propia.

Tabla V-4 Tabla de datos simulados de 0-4V a 4-20mA

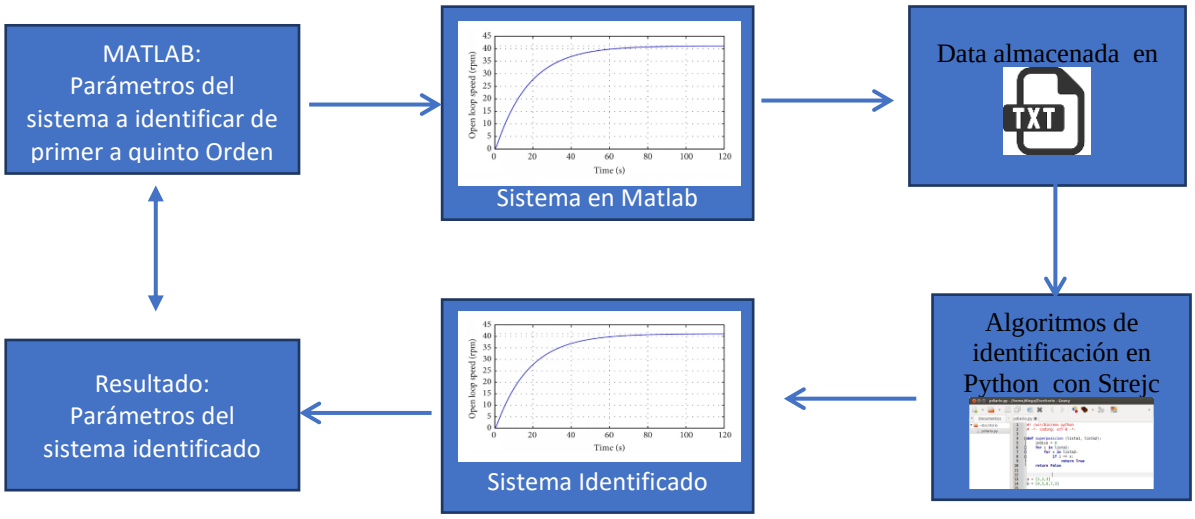
Tensión de Entrada (V) Simulación	Corriente de Salida (mA) Simulación	Valor esperado (mA)	Error %
0.80	4.01	4.00	-0.249
1.00	5.01	5.00	-0.200
1.50	7.54	7.50	-0.531
2.00	10.10	10.00	-0.990
2.50	12.60	12.50	-0.794
3.00	15.10	15.00	-0.662
3.50	17.70	17.50	-1.130
4.00	20.20	20.00	-0.990

Fuente: Elaboración Propia.

5.1.2 Variable X1:

X1: Desarrollo del algoritmo de identificación de un sistema SISO hasta Orden 5, haciendo uso de herramientas de software de tipo open source, a continuación, el diagrama de bloques del proceso de validación y las métricas para la validación son las siguientes:

Figura V-6 Diagrama de Bloques Validación de la Variable X1



Fuente: Elaboración Propia

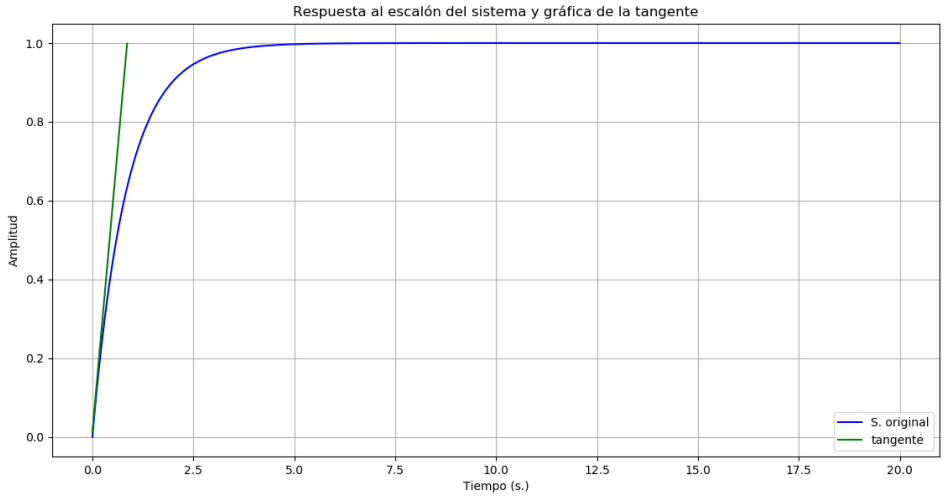
a) Desarrollo de Algoritmos de identificación

- Para Orden 1:

  Mediante la herramienta de Matlab se generó el archivo primerOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

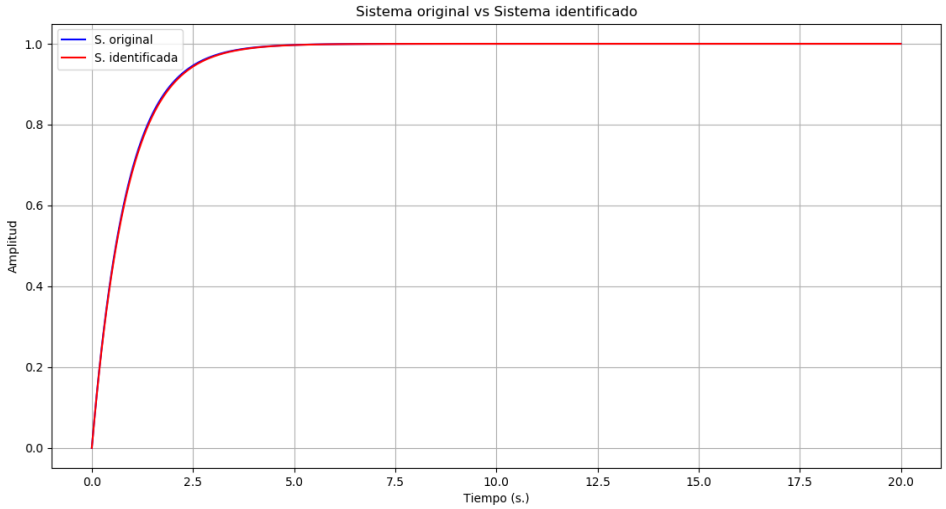
$$G(s) = \frac{1}{0.856 * S + 1} \qquad (V\text{-}1)$$

Gráfica V-1 Respuesta al Escalón Orden 1



Fuente: Elaboración Propia

Gráfica V-2 Sistema Identificado de Orden 1



Fuente: Elaboración Propia

$$G(s) = \frac{1 * e^{-0,03*s}}{0.899 * S + 1} \qquad (V\text{-}2)$$

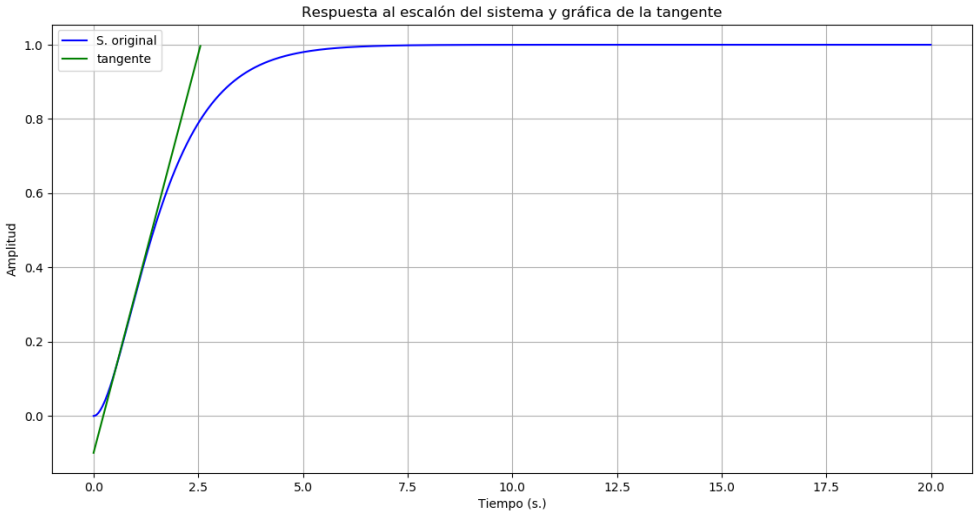
- Para Orden 2:

  Mediante la herramienta de Matlab se generó el archivo segundoOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

$$G(s) = \frac{1}{0.7327 * S^2 + 1.7120 * S + 1} \qquad (V\text{-}3)$$

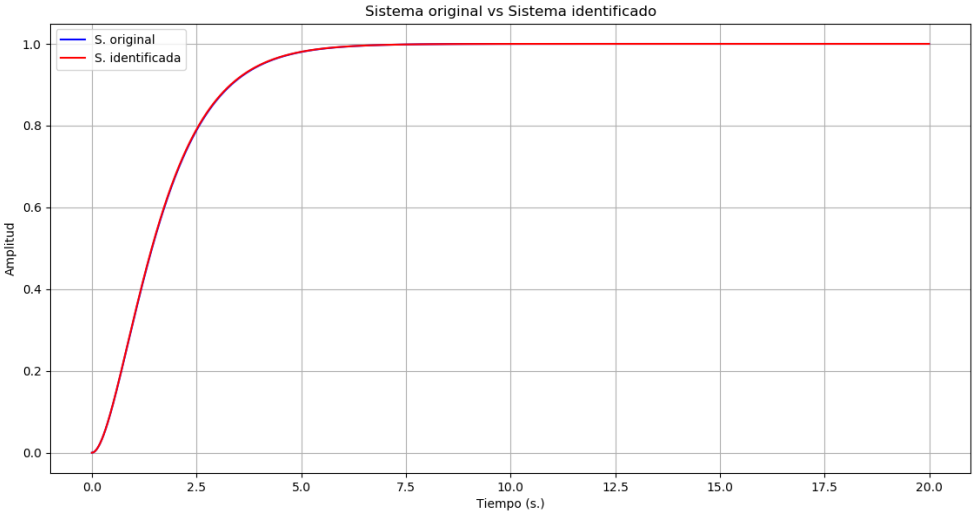
$$G(s) = \frac{1}{(0.856 * S + 1)^2} \qquad (V\text{-}4)$$

Gráfica V-3 Respuesta al Escalón Orden 2



Fuente: Elaboración Propia

Gráfica V-4 Sistema Identificado de Orden 2



Fuente: Elaboración Propia

$$G(s) = \frac{1 * e^{-0{,}03*s}}{(0.856 * S + 1)^2} \qquad (V\text{-}5)$$

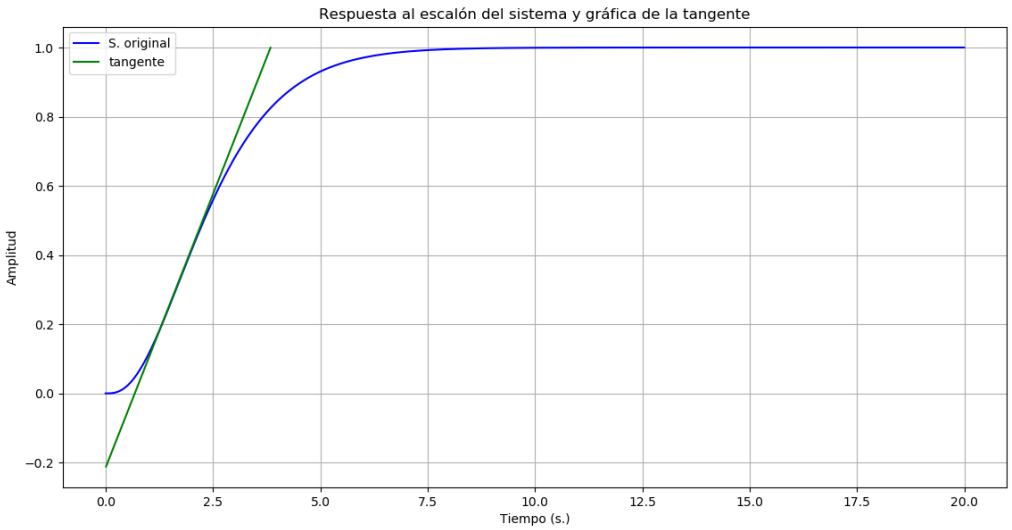
- Para Orden 3:

  Mediante la herramienta de Matlab se generó el archivo tercerOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

$$G(s) = \frac{1}{0.6272 * S^3 + 2.1982 * S^2 + 2.5680S + 1} \qquad (V\text{-}6)$$

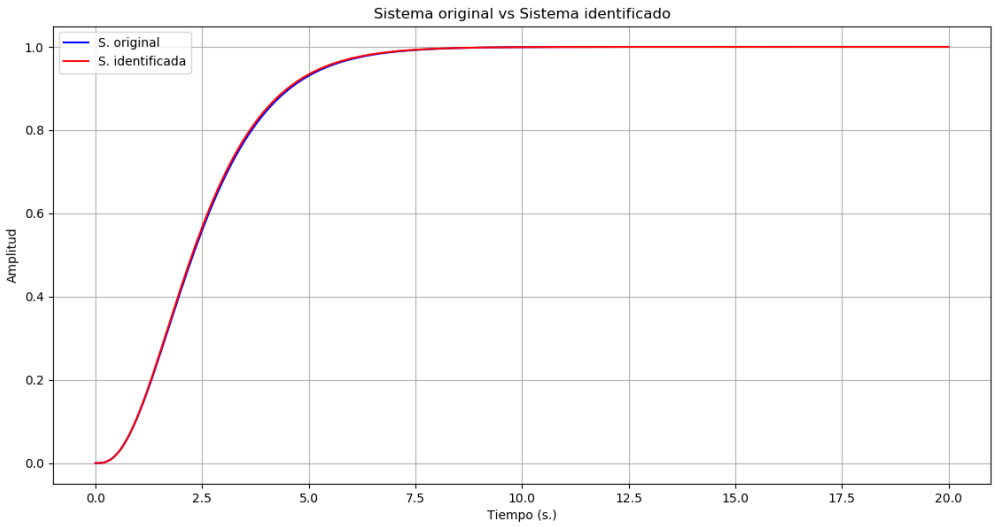
$$G(s) = \frac{1}{(0.856 * S + 1)^3} \qquad (V\text{-}7)$$

Gráfica V-5 Respuesta al Escalón Orden 3



Fuente: Elaboración Propia

Gráfica V-6 Sistema Identificado de Orden 3



Fuente: Elaboración Propia

$$G(s) = \frac{1 * e^{-0,03*s}}{(0.855 * S + 1)^3} \qquad (V\text{-}8)$$

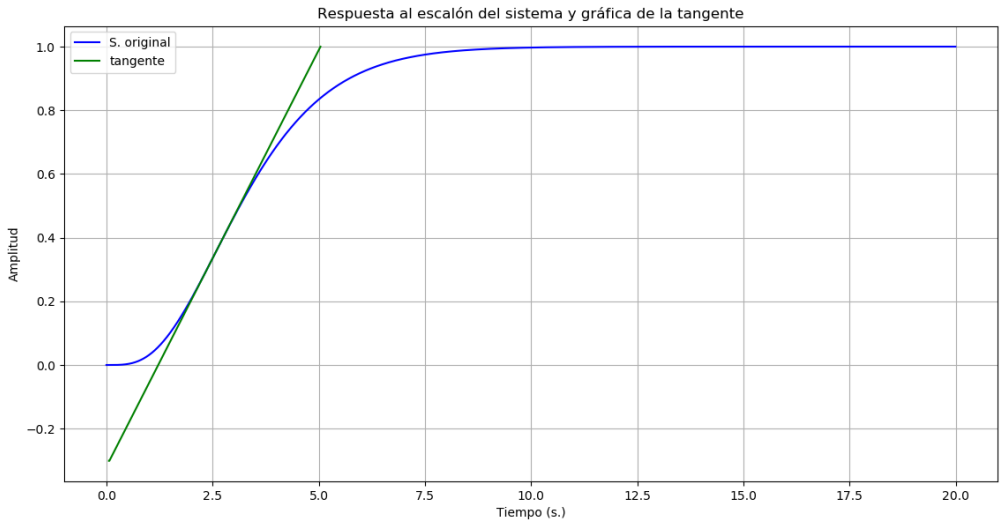
- Para Orden 4:

  Mediante la herramienta de Matlab se generó el archivo cuartoOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

$$G(s) = \frac{1}{0.5369 * S^4 + 2.5089 * S^3 + 4.3964 * S^2 + 3,4240 * S + 1} \qquad (V\text{-}9)$$

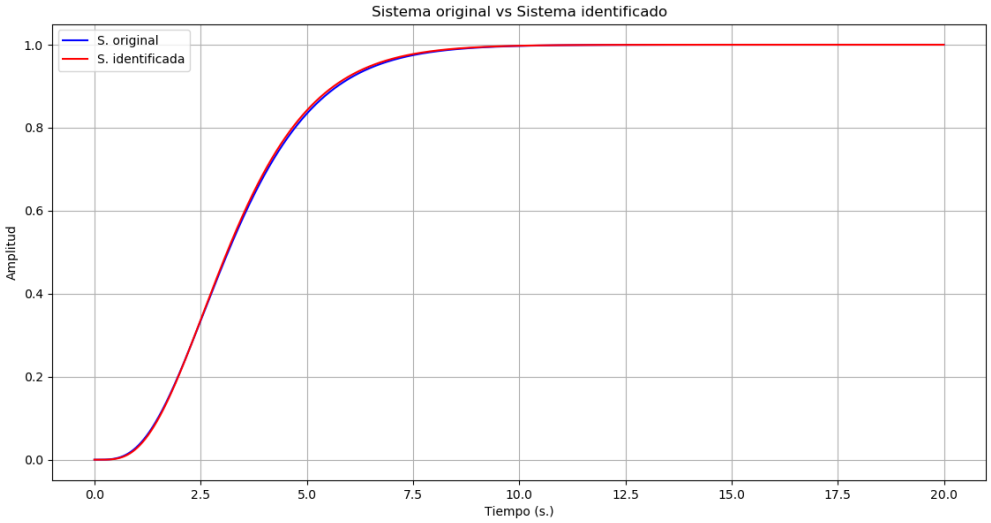
$$G(s) = \frac{1}{(0.856 * S + 1)^4} \qquad (V\text{-}10)$$

Gráfica V-7 Respuesta al Escalón Orden 4



Fuente: Elaboración Propia

Gráfica V-8 Sistema Identificado de Orden 4



Fuente Elaboración Propia

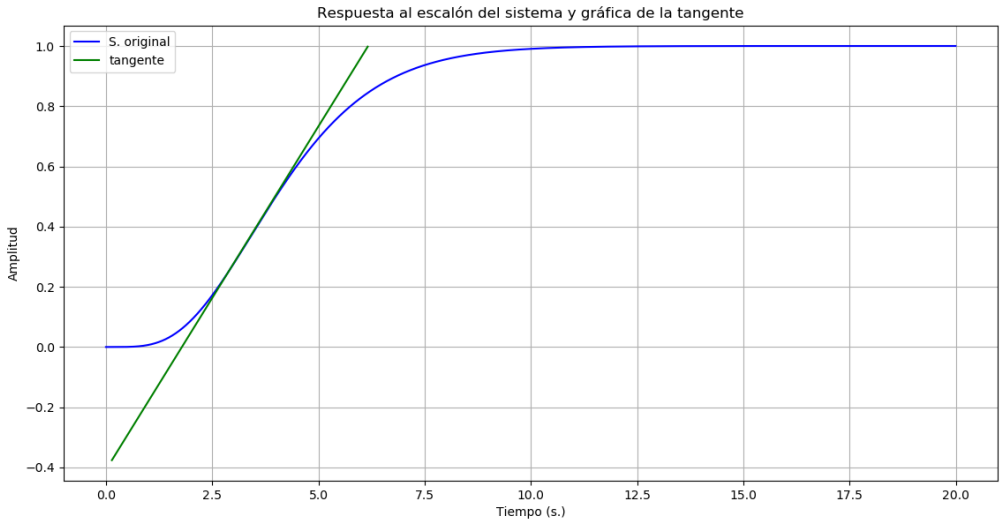
$$G(s) = \frac{1 * e^{-0,01*s}}{(0.848 * S + 1)^4} \qquad (V\text{-}11)$$

- Para Orden 5:

  Mediante la herramienta de Matlab se generó el archivo quintoOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

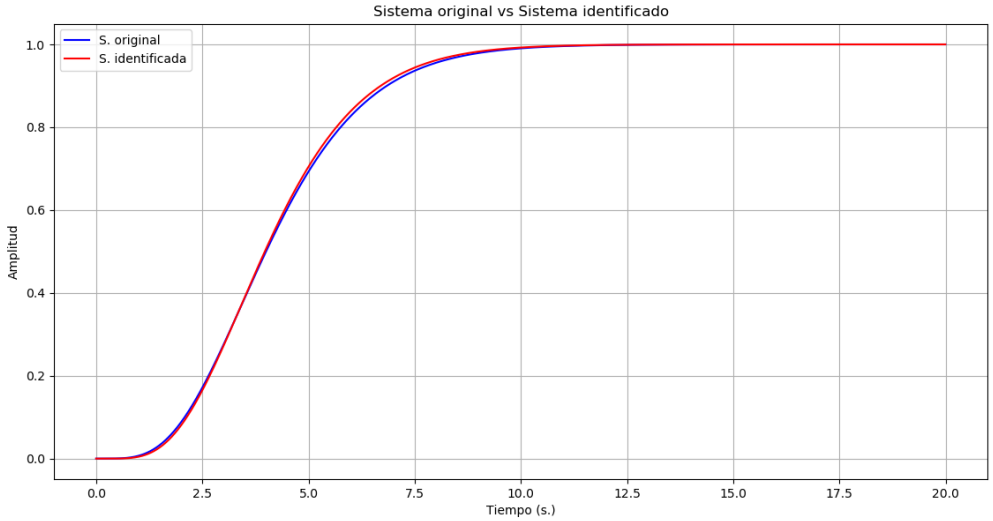
$$G(s) = \frac{1}{(0.856 * S + 1)^5} \qquad (V\text{-}12)$$

Gráfica V-9 Respuesta al Escalón Orden 5



Fuente: Elaboración Propia

Gráfica V-10 Sistema Identificado de Orden 5



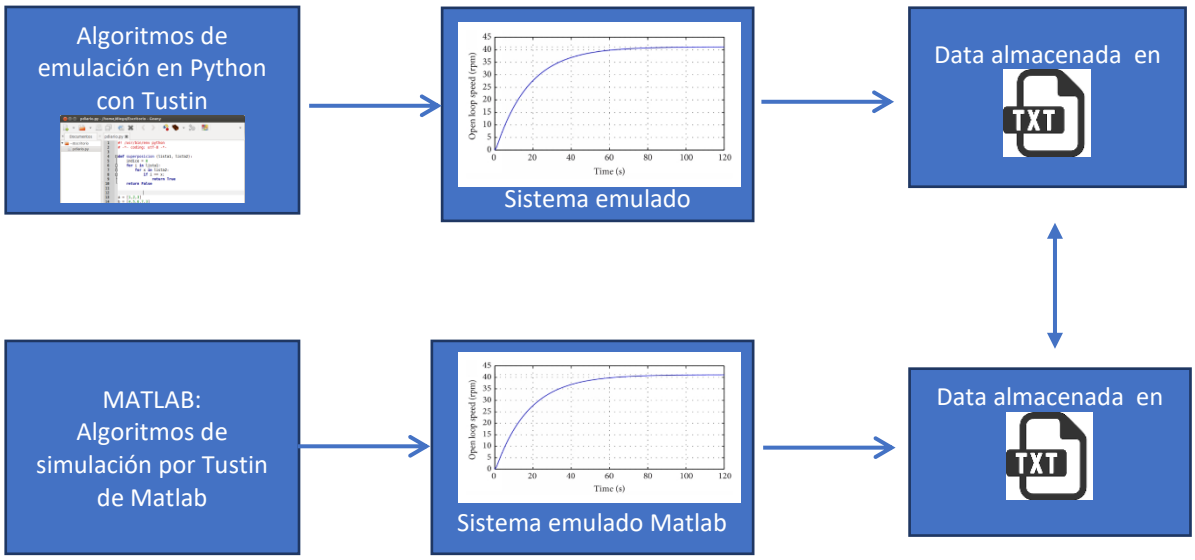
Fuente: Elaboración Propia

$$G(s) = \frac{1 * e^{0,09*s}}{(0.828 * S + 1)^5} \qquad (V\text{-}13)$$

5.1.3 Variable X2:

X2: Desarrollo del algoritmo de emulación de un sistema SISO hasta Orden 5, haciendo uso de herramientas de software de tipo open source, a continuación, el diagrama de bloques del proceso de validación y las métricas para la validación son las siguientes:

Figura V-7 Diagrama de Bloques Validación de la Variable X2



Fuente: Elaboración Propia

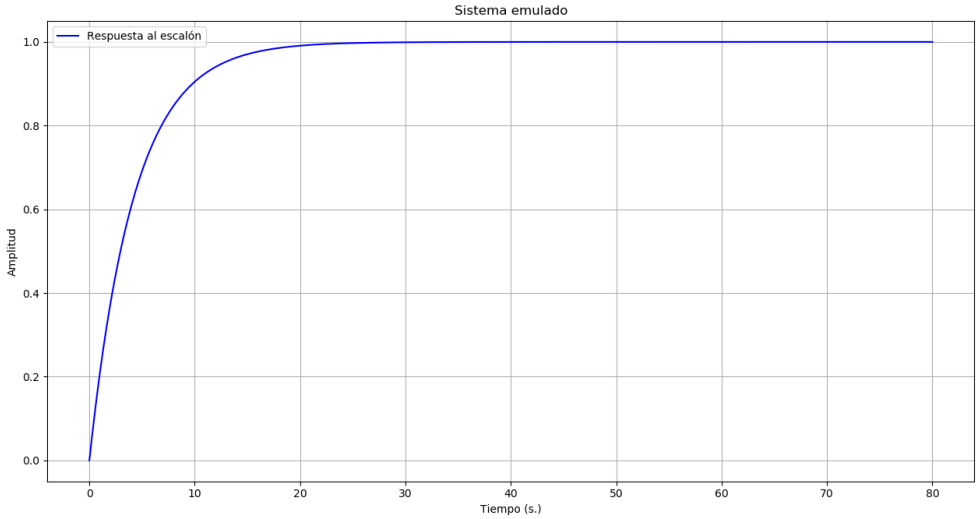
a) Desarrollo de algoritmos de emulación:

- Para Orden 1:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. El modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{4.25 * S + 1} \qquad (V\text{-}14)$$

Gráfica V-11 Sistema Emulado de Orden 1



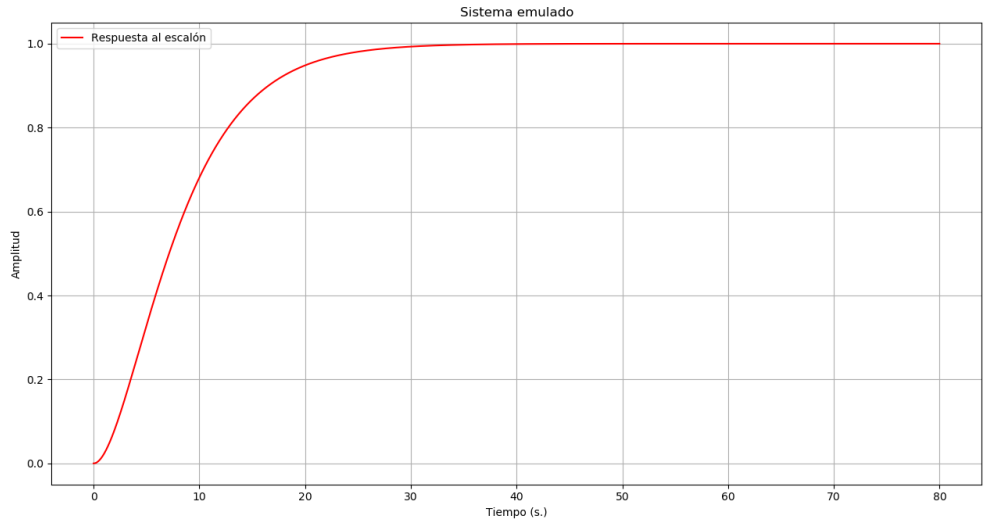
Fuente: Elaboración Propia.

- Para Orden 2:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. El modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{18.0625 * s^2 + 8.5 * s + 1} \qquad (V\text{-}15)$$

Gráfica V-12 Sistema Emulado de Orden 2



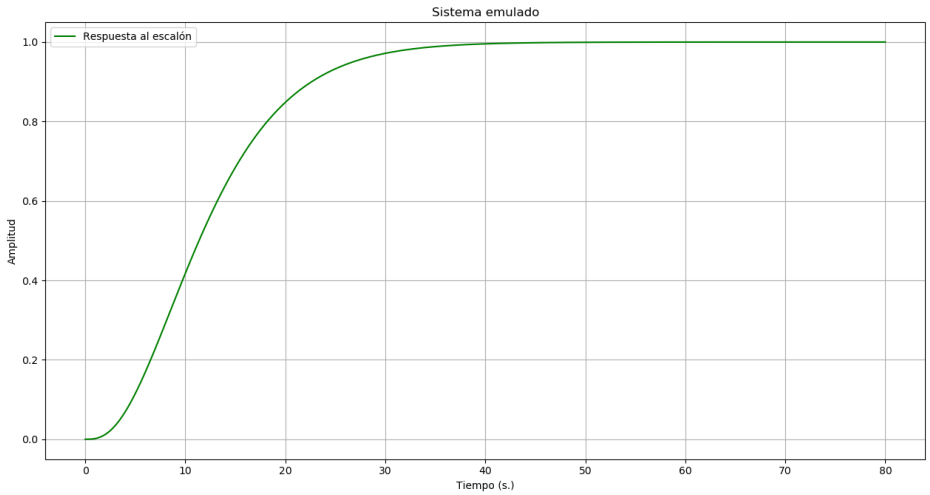
Fuente: Elaboración Propia

- Para Orden 3:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. EL modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{76.765625 * s^3 + 54.1875 * s^2 + 12.75 * s + 1} \quad (V\text{-}16)$$

Gráfica V-13 Sistema Emulado de Orden 3



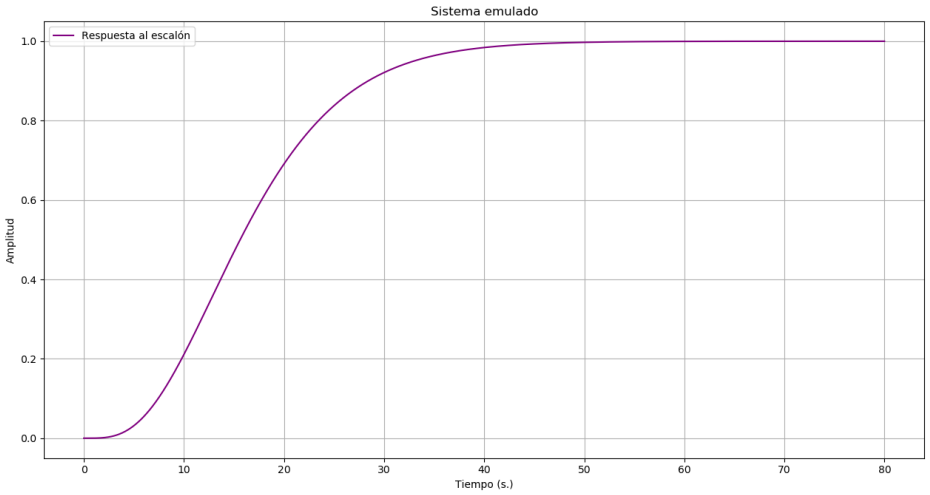
Fuente: Elaboración Propia

- Para Orden 4:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. El modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{326.25390625 * s^4 + 307.0625 * s^3 + 108.375 * s^2 + 17 * s + 1} \quad (V\text{-}17)$$

Gráfica V-14 Sistema Emulado de Orden 4



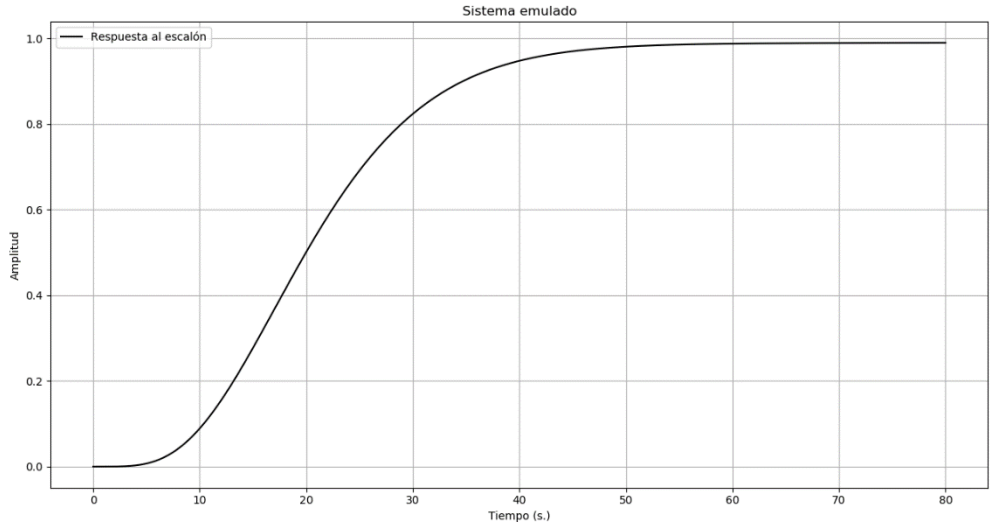
Fuente: Elaboración Propia

- Para Orden 5:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. El modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{(4.25 * s + 1)^5} \qquad (V\text{-}18)$$

Gráfica V-15 Sistema Emulado de Orden 5



Fuente: Elaboración Propia

5.1.4 Variable Y

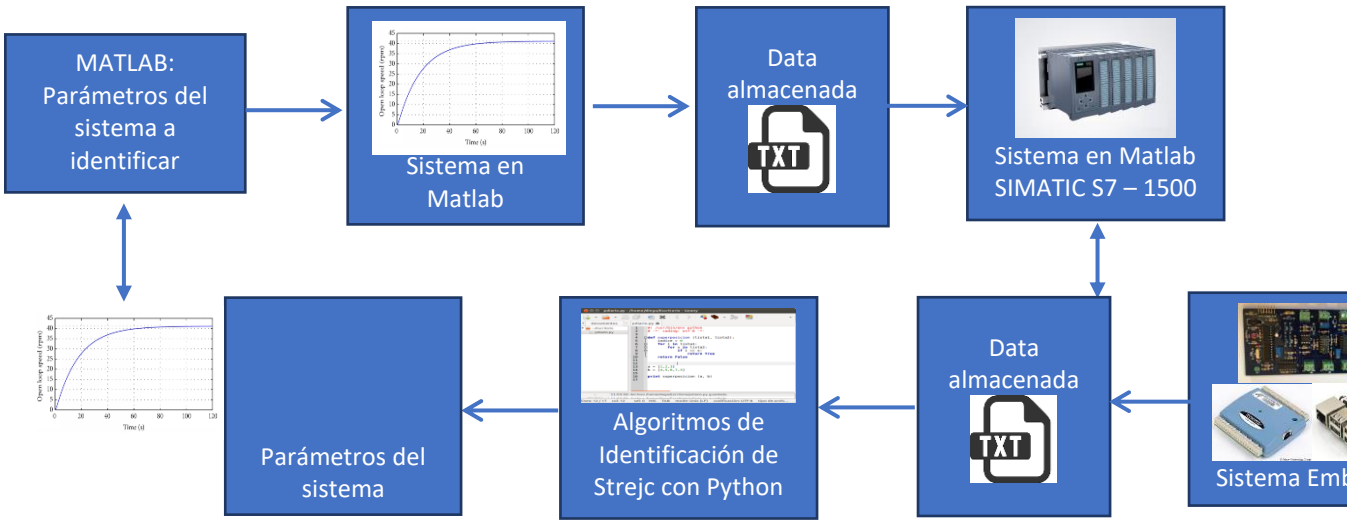
Desarrollo de un sistema portátil:

Y: Implementación de los algoritmos de identificación y emulación en un sistema embebido para la captura y visualización de las gráficas de entrada y salida, las métricas para la validación son las siguientes:

a) Implementar los algoritmos en el sistema embebido:

- Implementación del algoritmo de identificación.
    i. Hasta con 4 decimales.

Figura V-8 Diagrama de Bloques Validación de la Variable Y (Identificación)



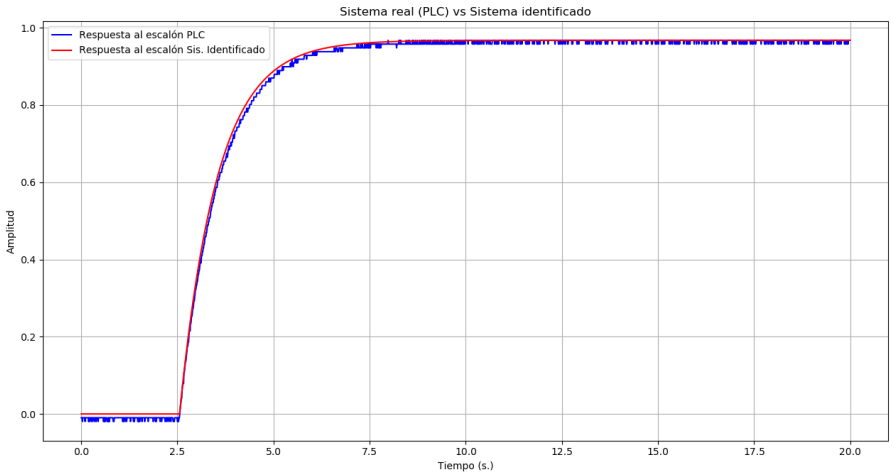
Fuente: Elaboración Propia

Para Orden 1

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 1 al PLC S7-1500 el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 1 fue generado por el Matlab y posteriormente cargado como vector en el PLC S7-1500.

Gráfica V-16 Sistema de Orden 1 Identificado



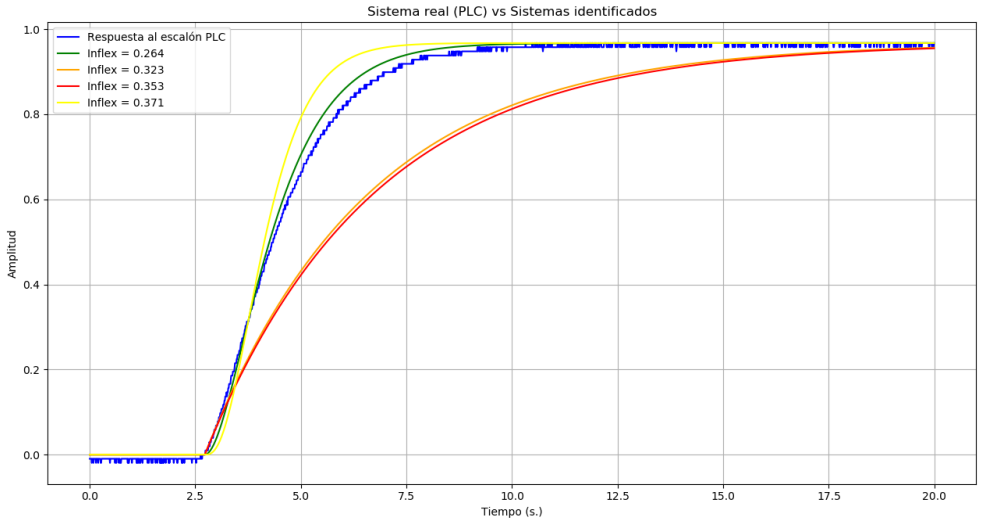
Fuente: Elaboración Propia

Para Orden 2

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 2 al PLC S7-1500 el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 2 fue generado por el Matlab y posteriormente cargado como vector en el PLC S7-1500.

Gráfica V-17 Sistema de Orden 2 Identificado



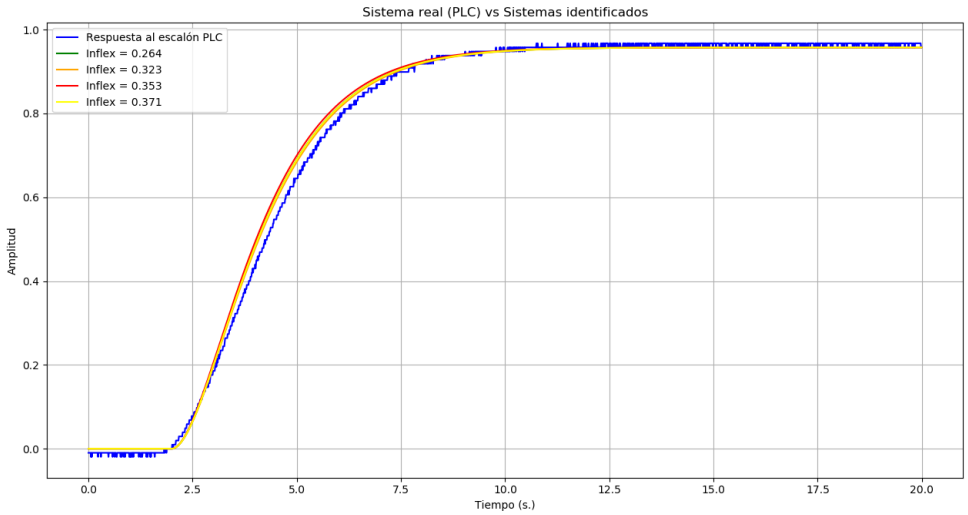
Fuente: Elaboración Propia

Para Orden 3

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 3 al PLC S7-1500 el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 3 fue generado por el Matlab y posteriormente cargado como vector en el PLC S7-1500.

Gráfica V-18 Sistema de Orden 3 Identificado



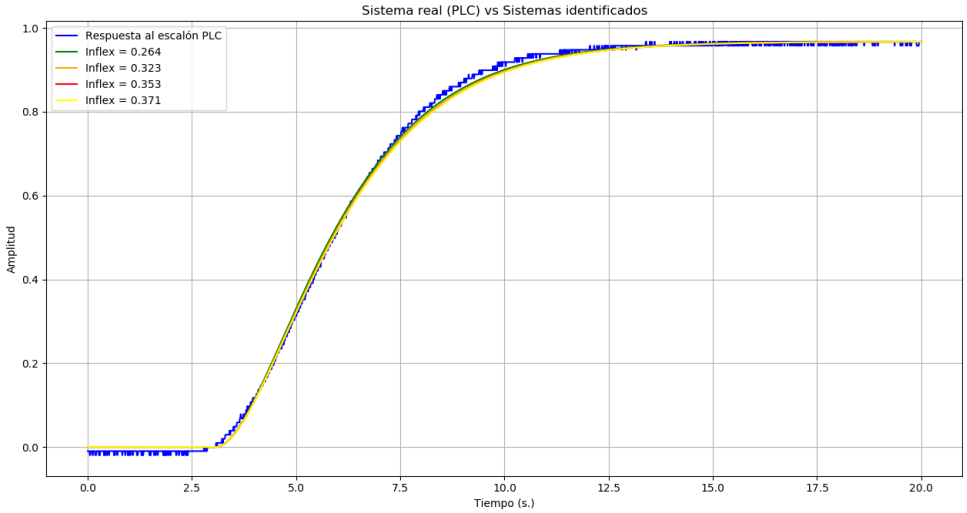
Fuente: Elaboración Propia

Para Orden 4

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 4 al PLC S7-1500 el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 4 fue generado por el Matlab y posteriormente cargado como vector en el PLC S7-1500.

Gráfica V-19 Sistema de Orden 4 Identificado



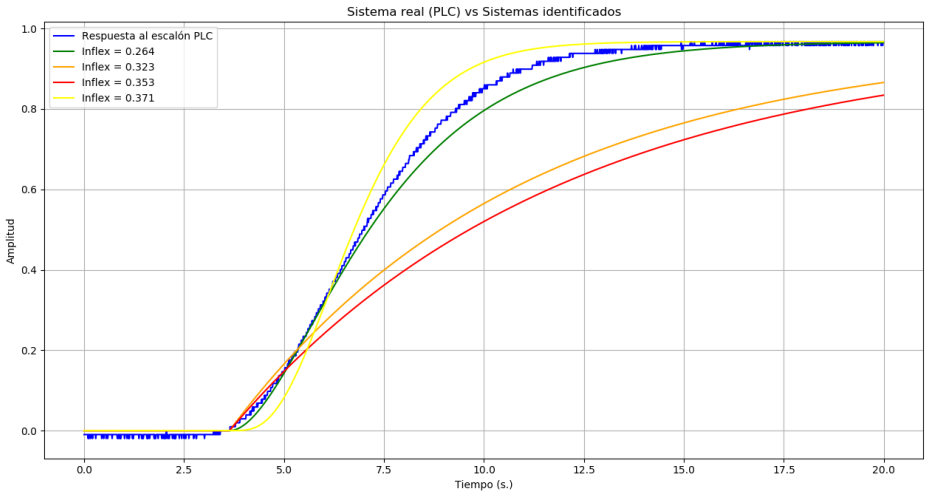
Fuente: Elaboración Propia

Para Orden 5

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 5 al PLC S7-1500 el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 5 fue generado por el Matlab y posteriormente cargado como vector en el PLC S7-1500.

Gráfica V-20 Sistema de Orden 5 Identificado



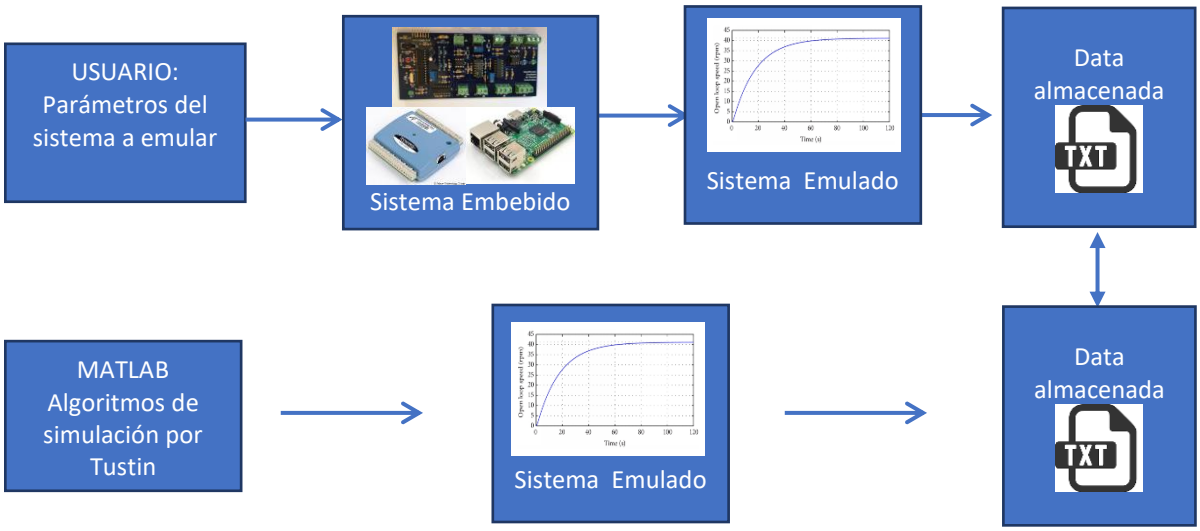
Fuente: Elaboración Propia

b) Implementar los algoritmos en el sistema embebido:

· Implementación del algoritmo de emulación.

i. Hasta con 4 decimales.

Figura V-9 Diagrama de Bloques Validación de la Variable Y (Emulación)



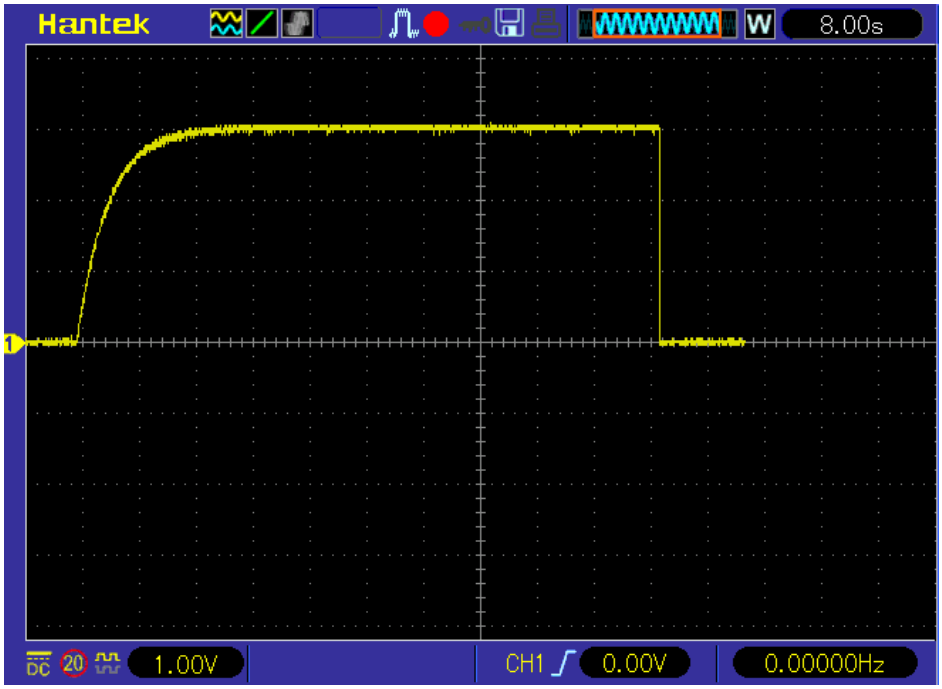
Fuente: Elaboración Propia

Para Orden 1

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 1 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 1 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.

Gráfica V-21 Sistema de Orden 1 emulado por el usuario



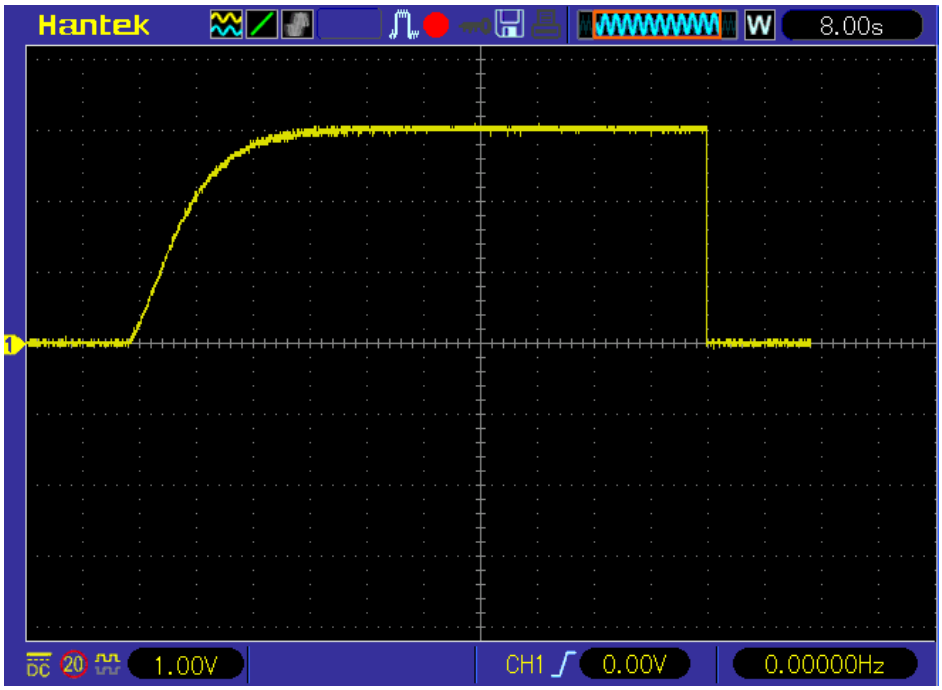
Fuente: Elaboración Propia

Para Orden 2

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 2 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 2 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.

Gráfica V-22 Sistema de Orden 2 emulador por el usuario



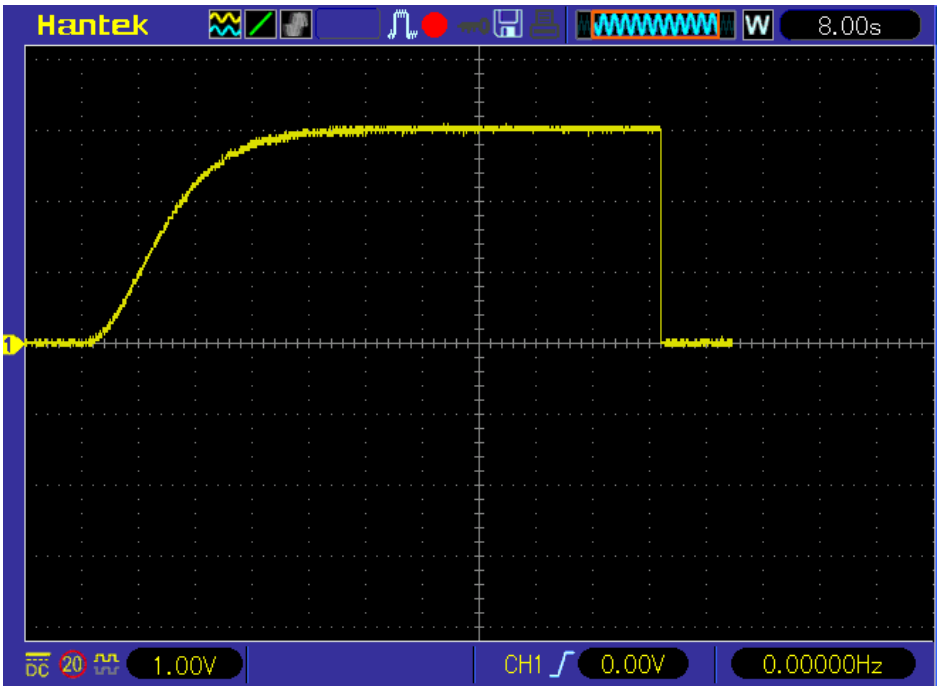
Fuente: Elaboración Propia

Para Orden 3

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 3 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 3 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.

Gráfica V-23 Sistema de Orden 3 emulado por el usuario



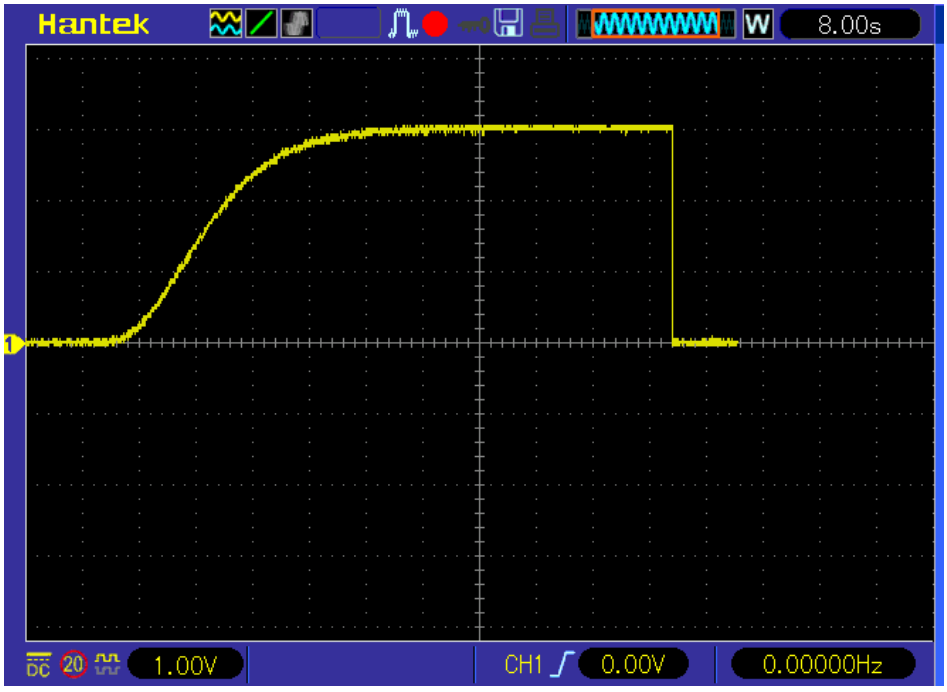
Fuente: Elaboración Propia

Para Orden 4

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 4 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 4 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.

Gráfica V-24 Sistema de Orden 4 emulado por el usuario



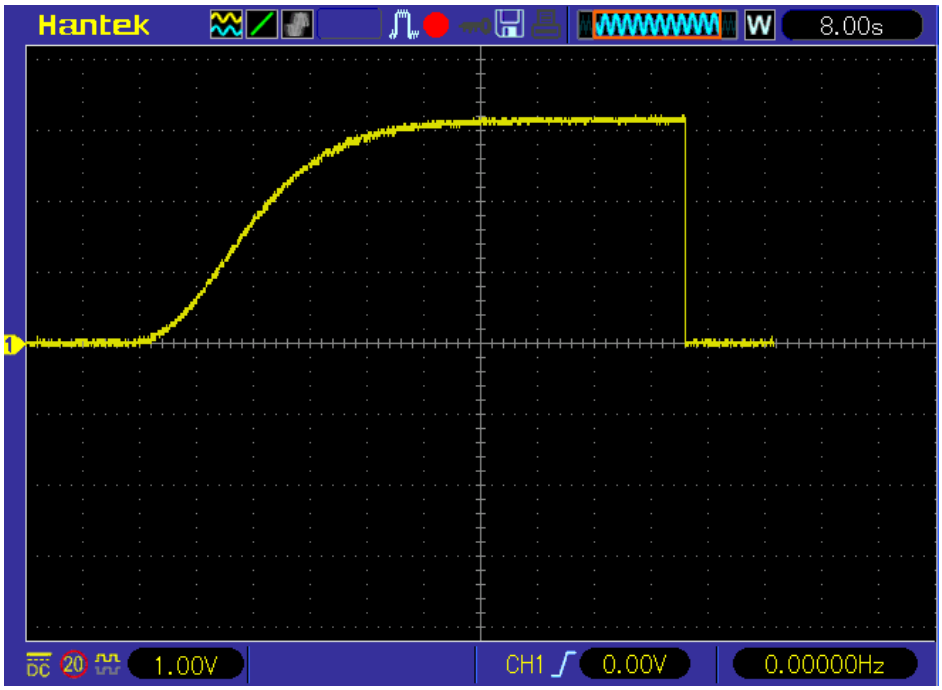
Fuente: Elaboración Propia

Para Orden 5

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 5 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 5 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.

Gráfica V-25 Sistema de Orden 5 emulado por el usuario



Fuente: Elaboración Propia

5.2 Resultados inferenciales

5.2.1 Desarrollo de las interfaces de entrada y salida:

Desarrollo de las interfaces de acondicionamiento de las señales de entrada y salida para la identificación y emulación de un sistema SISO, las métricas para la validación son las siguientes:

a) Los rangos de entrada y salida analógica:

- Nivel de tensión: 0-10v.
    i. Con un error menor a 5%
- Nivel de corriente: 4-20mA.
    i. Con un error menor a 5%

Tabla V-5 Tabla de datos simulados de salida 0-4V a 0-10V

Tensión de Entrada (V) Simulación	Tensión de Salida (V) Simulación	Valor esperado (V)	(Real-Simulado) ^2	Error %
0.50	1.24	1.25	0.000100	0.806
1.00	2.49	2.50	0.000100	0.402
1.50	3.74	3.75	0.000100	0.267
2.00	4.99	5.00	0.000100	0.200
2.50	6.24	6.25	0.000100	0.160
3.00	7.49	7.50	0.000100	0.134
3.50	8.74	8.75	0.000100	0.114
4.00	9.99	10.00	0.000100	0.100

Promedio del % de Error	0.27302
Media del % de Error	0.18033
Desviación del % de Error	0.23713
ECM	0.00010

Fuente: Elaboración propia

Con los resultados obtenidos después del proceso de simulación del diseño del circuito de acondicionamiento de la señal de 0-4V a 0-10V se muestra que el diseño simulado está dentro de la métrica propuesta (menor al 5%), adicionalmente se tienen otras métricas como la media, desviación y error cuadrático medio donde se nota que hay consistencia en los resultados obtenidos.

Tabla V-6 Tabla de datos del circuito de acondicionamiento de salida 0-4V a 0-10V

Tensión de Entrada (V) PLC S7-1500	Tensión de Salida (V) Fluke 111	Valor esperado (V)	(Real-Simulado) ^2	Error %
0.50	1.244	1.25	0.000036	0.482
1.00	2.493	2.50	0.000049	0.281
1.50	3.743	3.75	0.000049	0.187
2.00	4.992	5.00	0.000064	0.160
2.50	6.239	6.25	0.000121	0.176
3.00	7.500	7.50	0.000000	0.000
3.50	8.75	8.75	0.000000	0.000
4.00	10.00	10.00	0.000000	0.000

Promedio del % de Error	0.16084
Media del % de Error	0.16828
Desviación del % de Error	0.16745
ECM	0.00004

Fuente: Elaboración propia

Con los resultados obtenidos después del proceso de implementación del circuito de acondicionamiento de la señal de 0-4V a 0-10V se muestra que el circuito implementado está dentro de la métrica propuesta (menor al 5%), adicionalmente se tienen otras métricas como la media, desviación y error cuadrático medio donde se nota que hay consistencia en los resultados obtenidos.

Tabla V-7 Tabla de datos simulados de entrada de 0-10V

Tensión de Entrada (V) Simulación	Tensión de Salida (V) Simulación	Valor esperado (V)	(Real-Simulado) ^2	Error %
0.50	0.49	0.50	0.000100	2.041
1.00	0.99	1.00	0.000100	1.010
1.50	1.49	1.50	0.000100	0.671
2.00	1.99	2.00	0.000100	0.503
2.50	2.49	2.50	0.000100	0.402
3.00	2.99	3.00	0.000100	0.334
3.50	3.49	3.50	0.000100	0.287
4.00	3.99	4.00	0.000100	0.251
4.50	4.49	4.50	0.000100	0.223
5.00	4.98	5.00	0.000400	0.402
5.50	5.49	5.50	0.000100	0.182
6.00	5.98	6.00	0.000400	0.334
6.50	6.49	6.50	0.000100	0.154
7.00	6.99	7.00	0.000100	0.143
7.50	7.48	7.50	0.000400	0.267
8.00	7.99	8.00	0.000100	0.125
8.50	8.49	8.50	0.000100	0.118
9.00	8.99	9.00	0.000100	0.111
9.50	9.49	9.50	0.000100	0.105
10.00	9.99	10.00	0.000100	0.100

Promedio del % de Error	0.16207
Media del % de Error	0.12516
Desviación del % de Error	0.08230
ECM	0.00017

Fuente: Elaboración propia

Con los resultados obtenidos después del proceso de simulación del diseño del circuito de acondicionamiento de la señal de 0-10V se muestra que el diseño simulado está dentro de la métrica propuesta (menor al 5%), adicionalmente se tienen otras métricas como la media, desviación y error cuadrático medio donde se nota que hay consistencia en los resultados obtenidos.

Tabla V-8 Tabla del circuito de acondicionamiento de 0-10V

Tensión de Entrada (V) PLC S7-1500	Tensión de Salida (V) Fluke 111	Valor esperado (V)	(Real-Simulado) ^2	Error %
0.50	0.499	0.50	0.000001	0.200
1.00	0.997	1.00	0.000009	0.301
1.50	1.497	1.50	0.000009	0.200
2.00	1.995	2.00	0.000025	0.251
2.50	2.494	2.50	0.000036	0.241
3.00	2.992	3.00	0.000064	0.267
3.50	3.493	3.50	0.000049	0.200
4.00	3.992	4.00	0.000064	0.200
4.50	4.491	4.50	0.000081	0.200
5.00	4.990	5.00	0.000100	0.200
5.50	5.489	5.50	0.000121	0.200
6.00	5.988	6.00	0.000144	0.200
6.50	6.487	6.50	0.000169	0.200
7.00	6.990	7.00	0.000100	0.143
7.50	7.490	7.50	0.000100	0.134
8.00	7.990	8.00	0.000100	0.125
8.50	8.49	8.50	0.000100	0.118
9.00	8.99	9.00	0.000100	0.111
9.50	9.49	9.50	0.000100	0.105
10.00	9.99	10.00	0.000100	0.100
Promedio del % de Error		0.13745		
Media del % de Error		0.12516		
Desviación del % de Error		0.03811		
ECM		0.00011		

Fuente: Elaboración propia

Con los resultados obtenidos después del proceso de implementación del circuito de acondicionamiento de la señal de 0-10V se muestra que el diseño implementado está dentro de la métrica propuesta (menor al 5%), adicionalmente se tienen otras métricas como la media, desviación y error cuadrático medio donde se nota que hay consistencia en los resultados obtenidos.

Tabla V-9 Tabla de datos simulados de 4-20mA a 0-10V

Corriente de Entrada (mA) Simulación	Tensión de Salida (V) Simulación	Valor esperado (V)	(Real-Simulado) ^2	Error %
4.00	1.98	2.00	0.000400	1.010
6.00	2.97	3.00	0.000900	1.010
8.00	3.97	4.00	0.000900	0.756
10.00	4.96	5.00	0.001600	0.806
12.00	5.95	6.00	0.002500	0.840
14.00	6.95	7.00	0.002500	0.719
16.00	7.94	8.00	0.003600	0.756
18.00	8.94	9.00	0.003600	0.671
20.00	9.93	10.00	0.004900	0.705

Promedio del % de Error	0.80820
Media del % de Error	0.75567
Desviación del % de Error	0.12521
ECM	0.00232

Fuente: Elaboración propia

Con los resultados obtenidos después del proceso de simulación del diseño del circuito de acondicionamiento de la señal de 4-20mA a 0-10V se muestra que el diseño simulado está dentro de la métrica propuesta (menor al 5%), adicionalmente se tienen otras métricas como la media, desviación y error cuadrático medio donde se nota que hay consistencia en los resultados obtenidos.

Tabla V-10 Tabla del circuito de 4-20mA a 0-10V

Corriente de Entrada (mA) S7-1500	Tensión de Salida (V) Fluke 111	Valor esperado (V)	(Real-Simulado) ^2	Error %
4.00	1.989	2.00	0.000121	0.553
6.00	2.992	3.00	0.000064	0.267
8.00	3.996	4.00	0.000016	0.100
10.00	4.990	5.00	0.000100	0.200
12.00	6.003	6.00	0.000009	-0.050
14.00	7.009	7.00	0.000081	-0.128
16.00	8.020	8.00	0.000400	-0.249
18.00	9.020	9.00	0.000400	-0.222
20.00	10.03	10.00	0.000900	-0.299

Promedio del % de Error	0.01915
Media del % de Error	-0.04998
Desviación del % de Error	0.28373
ECM	0.00023

Fuente: Elaboración propia

Con los resultados obtenidos después del proceso de implementación del circuito de acondicionamiento de la señal de 4-20mA a 0-10V se muestra que el diseño implementado está dentro de la métrica propuesta (menor al 5%), adicionalmente se tienen otras métricas como la media, desviación y error cuadrático medio donde se nota que hay consistencia en los resultados obtenidos.

Tabla V-11 Tabla de datos simulados de 0-4V a 4-20mA

Tensión de Entrada (V) Simulación	Corriente de Salida (mA) Simulación	Valor esperado (mA)	(Real-Simulado) ^2	Error %
0.80	4.01	4.00	0.000100	-0.249
1.00	5.01	5.00	0.000100	-0.200
1.50	7.54	7.50	0.001600	-0.531
2.00	10.10	10.00	0.010000	-0.990
2.50	12.60	12.50	0.010000	-0.794
3.00	15.10	15.00	0.010000	-0.662
3.50	17.70	17.50	0.040000	-1.130
4.00	20.20	20.00	0.040000	-0.990
Promedio del % de Error		-0.693		
Media del % de Error		-0.728		
Desviación del % de Error		0.347		
ECM		0.013975		

Fuente: Elaboración Propia.

Con los resultados obtenidos después del proceso de simulación del diseño del circuito de acondicionamiento de la señal de 0-4V a 4-20mA se muestra que el diseño simulado está dentro de la métrica propuesta (menor al 5%), adicionalmente se tienen otras métricas como la media, desviación y error cuadrático medio donde se nota que hay consistencia en los resultados obtenidos.

Tabla V-12 Tabla de datos simulados de 0-4V a 4-20mA

Tensión de Salida (V) PLC S7-1500	Corriente de Entrada (mA) S7-1500	Valor esperado (mA)	(Real-Simulado) ^2	Error %
0.00	4.036	4.00	0.001296	-0.892
1.00	4.998	5.00	0.000004	0.040
1.50	7.489	7.50	0.000121	0.147
2.00	10.023	10.00	0.000529	-0.229
2.50	12.487	12.50	0.000169	0.104
3.00	15.000	15.00	0.000000	0.000
3.50	17.470	17.50	0.000900	0.172
4.00	19.986	20.00	0.000196	0.070

Promedio del % de Error	-0.074
Media del % de Error	0.055
Desviación del % de Error	0.353
ECM	0.000357

Fuente: Elaboración Propia.

Con los resultados obtenidos después del proceso de implementación del circuito de acondicionamiento de la señal de 0-4V a 4-20mA se muestra que el diseño implementado está dentro de la métrica propuesta (menor al 5%), adicionalmente se tienen otras métricas como la media, desviación y error cuadrático medio donde se nota que hay consistencia en los resultados obtenidos.

5.2.2 Variable X1:

X1: Desarrollo del algoritmo de identificación de un sistema SISO hasta Orden 5, haciendo uso de herramientas de software de tipo open source, las métricas para la validación son las siguientes:

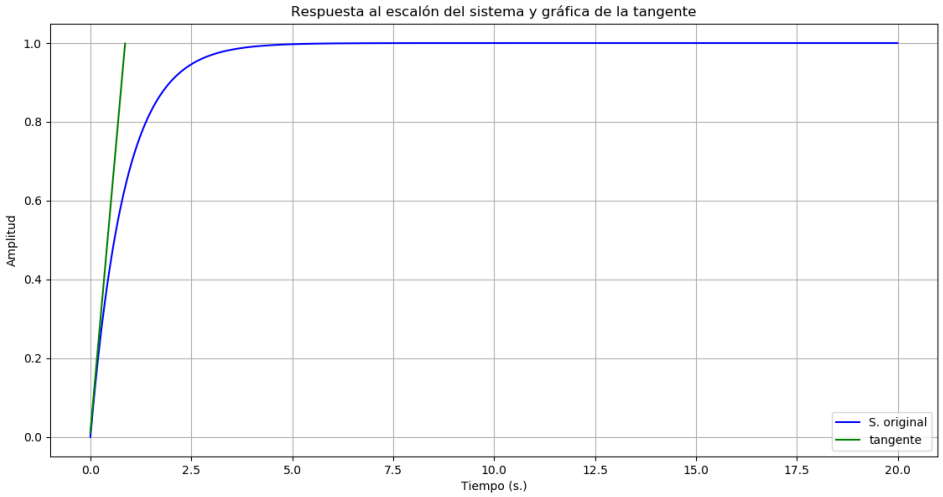
a) Desarrollo de Algoritmos de identificación

- Para Orden 1:

  Mediante la herramienta de Matlab se generó el archivo primerOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

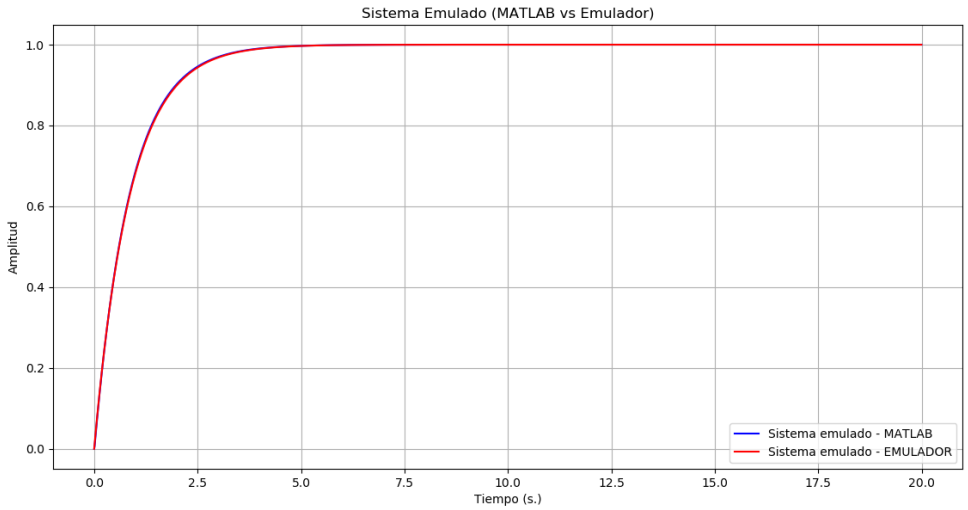
$$G(s) = \frac{1}{0.856 * S + 1} \qquad (V\text{-}19)$$

Gráfica V-26 Respuesta al Escalón Orden 1



Fuente: Elaboración Propia

Gráfica V-27 Sistema Identificado de Orden 1



Nro de datos : 2000 ECM : 1.5320307550003483e-06

Fuente: Elaboración Propia

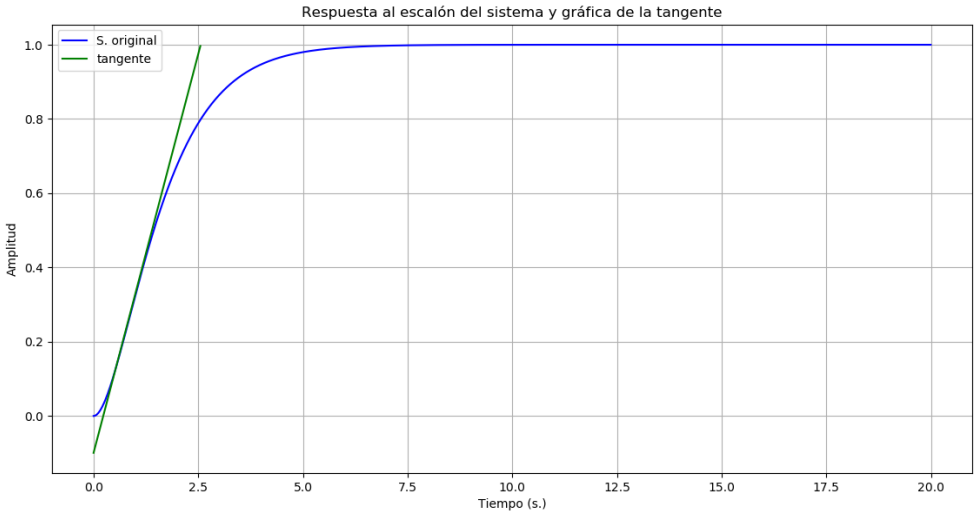
Con los resultados obtenidos después del proceso de identificación de un sistema de Orden 1, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

- Para Orden 2:

Mediante la herramienta de Matlab se generó el archivo segundoOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

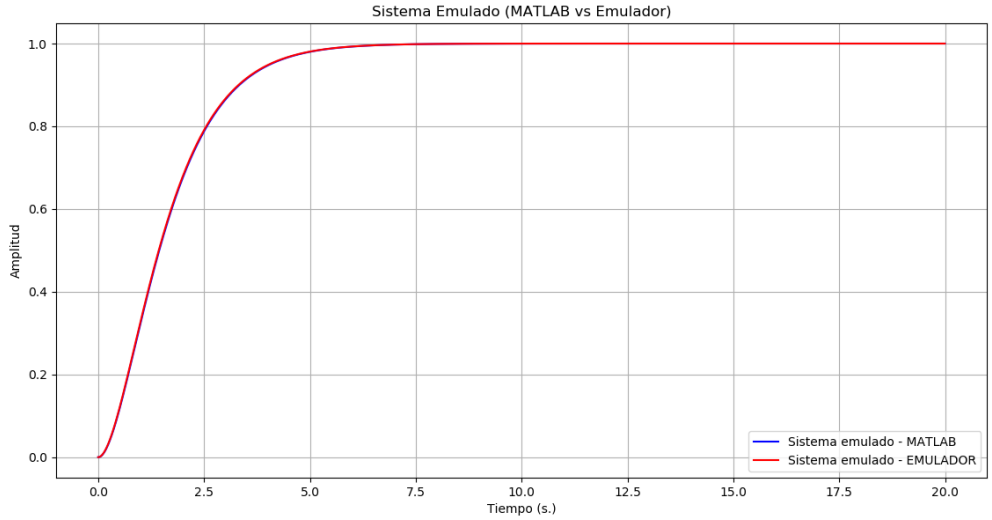
$$G(s) = \frac{1}{0.7327 * S^2 + 1.7120 * S + 1} \qquad (V\text{-}20)$$

Gráfica V-28 Respuesta al Escalón Orden 2



Fuente: Elaboración Propia

Gráfica V-29 Sistema Identificado de Orden 2



Nro de datos : 2000 ECM : 1.9482809400916138e-06

Fuente: Elaboración Propia

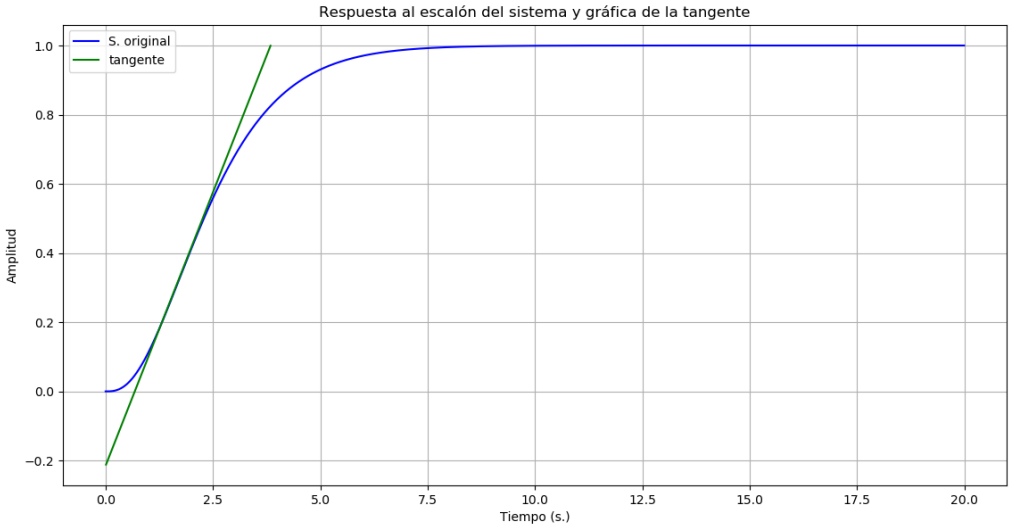
Con los resultados obtenidos después del proceso de identificación de un sistema de Orden 2, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

· Para Orden 3:

Mediante la herramienta de Matlab se generó el archivo tercerOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

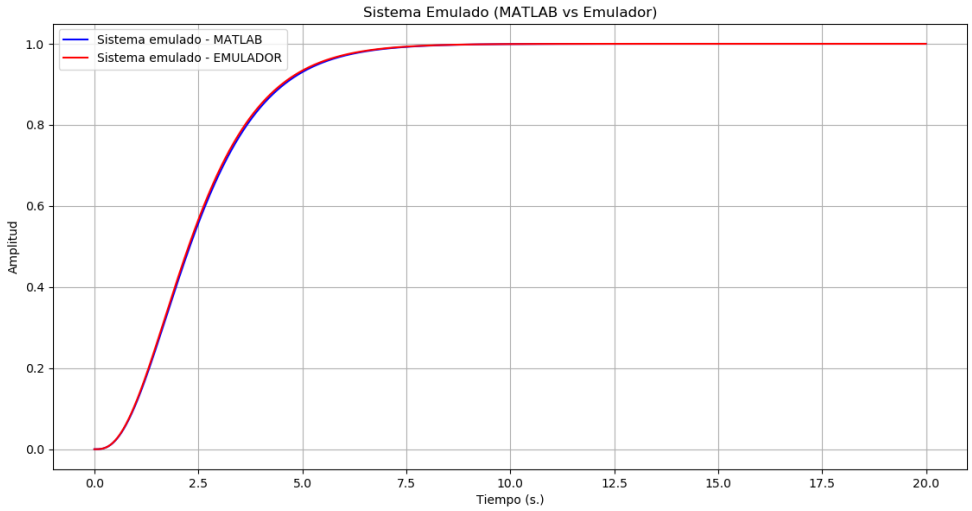
$$G(s) = \frac{1}{0.6272 * S^3 + 2.1982 * S^2 + 2.5680S + 1} \qquad (V\text{-}21)$$

Gráfica V-30 Respuesta al Escalón Orden 3



Fuente: Elaboración Propia

Gráfica V-31 Sistema Identificado de Orden 3



Nro de datos : 2000 ECM : 1.0005484383312472e-05

Fuente: Elaboración Propia

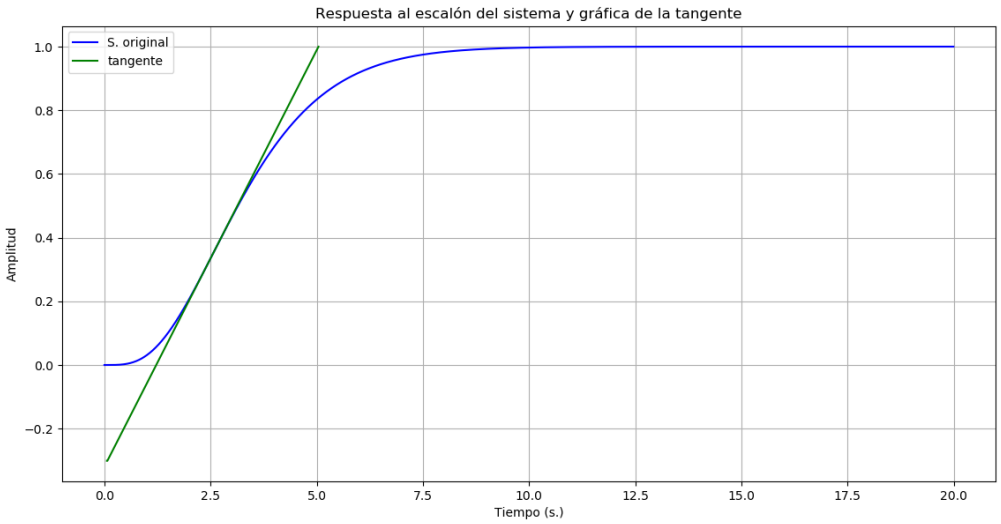
Con los resultados obtenidos después del proceso de identificación de un sistema de Orden 3, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

- Para Orden 4:

  Mediante la herramienta de Matlab se generó el archivo cuartoOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

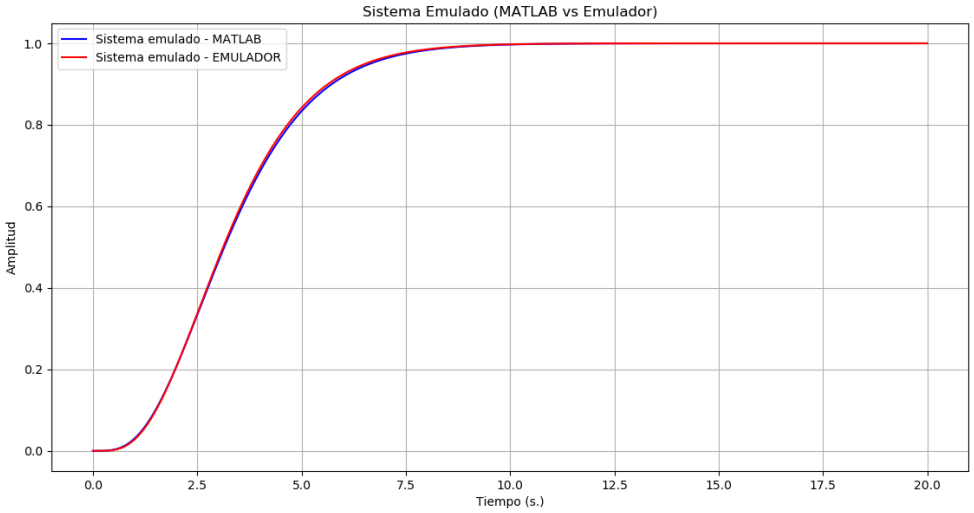
$$G(s) = \frac{1}{0.5369 * S^4 + 2.5089 * S^3 + 4.3964 * S^2 + 3{,}4240 * S \quad + 1} \qquad (V\text{-}22)$$

Gráfica V-32 Respuesta al Escalón Orden 4



Fuente: Elaboración Propia

Gráfica V-33 Sistema Identificado de Orden 4



Nro de datos : 2000 ECM : 1.2234121903681746e-05

Fuente: Elaboración Propia

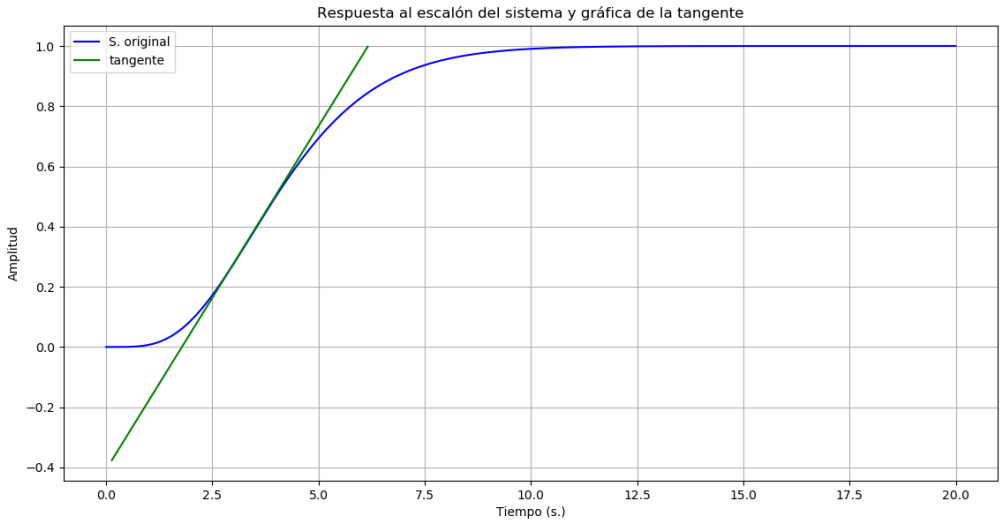
Con los resultados obtenidos después del proceso de identificación de un sistema de Orden 4, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

· Para Orden 5:

Mediante la herramienta de Matlab se generó el archivo quintoOrden3ms.txt, el cual tiene una tasa de muestreo del 3ms y un total 10000 muestras. El modelo elaborado por Matlab fue el siguiente:

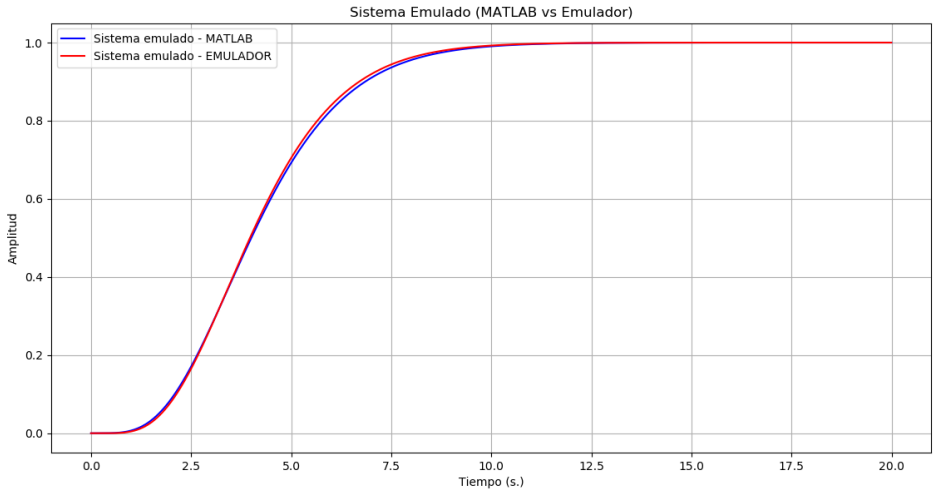
$$G(s) = \frac{1}{(0.856 * S + 1)^2} \qquad (V\text{-}23)$$

Gráfica V-34 Respuesta al Escalón Orden 5



Fuente: Elaboración Propia

Gráfica V-35 Sistema Identificado de Orden 5



Nro de datos : 2000 ECM : 1.2234121903681746e-05

Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación de un sistema de Orden 5, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

5.2.3 Variable X2:

X2: Desarrollo del algoritmo de emulación de un sistema SISO hasta Orden 5, haciendo uso de herramientas de software de tipo open source, las métricas para la validación son las siguientes:

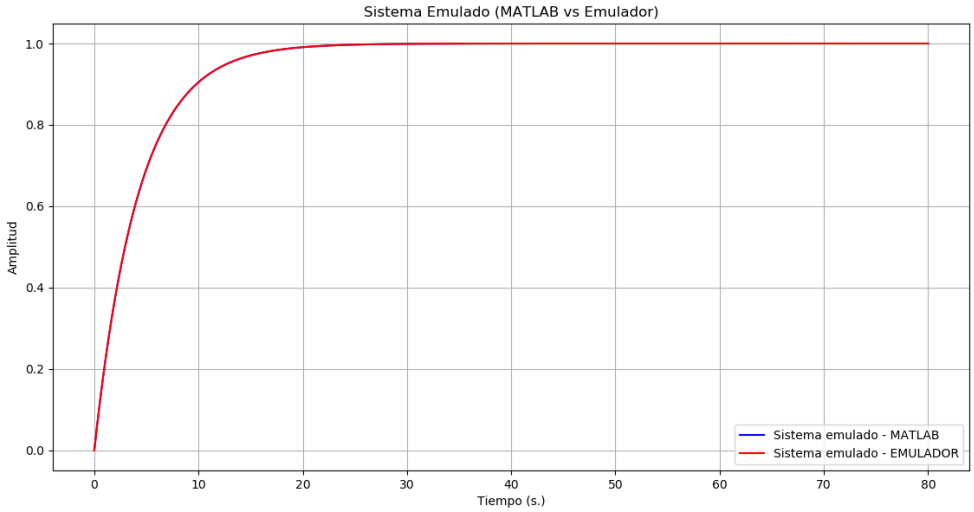
a) Desarrollo de algoritmos de emulación:

- Para Orden 1:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. El modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{4.25 * S + 1} \qquad (V\text{-}24)$$

Gráfica V-36 Sistema Emulado de Orden 1



Nro de datos : 8000 ECM : 7.108899895489959e-14

Fuente: Elaboración Propia

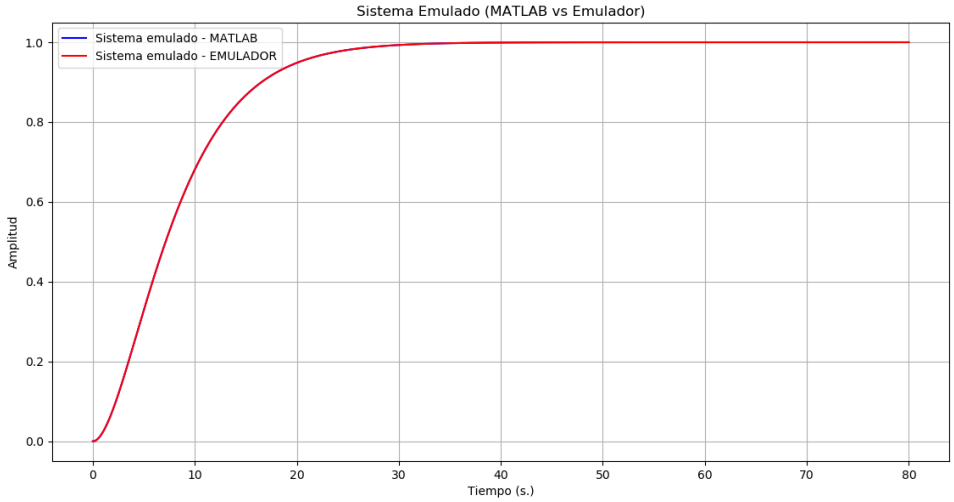
Con los resultados obtenidos después del proceso de emulación de un sistema de Orden 1, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

- Para Orden 2:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. EL modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{18.0625 * s^2 + 8.5 * s + 1} \qquad (V\text{-}25)$$

Gráfica V-37 Sistema Emulado de Orden 2



Nro de datos : 8000 ECM : 8.412175039039374e-14

Fuente: Elaboración Propia

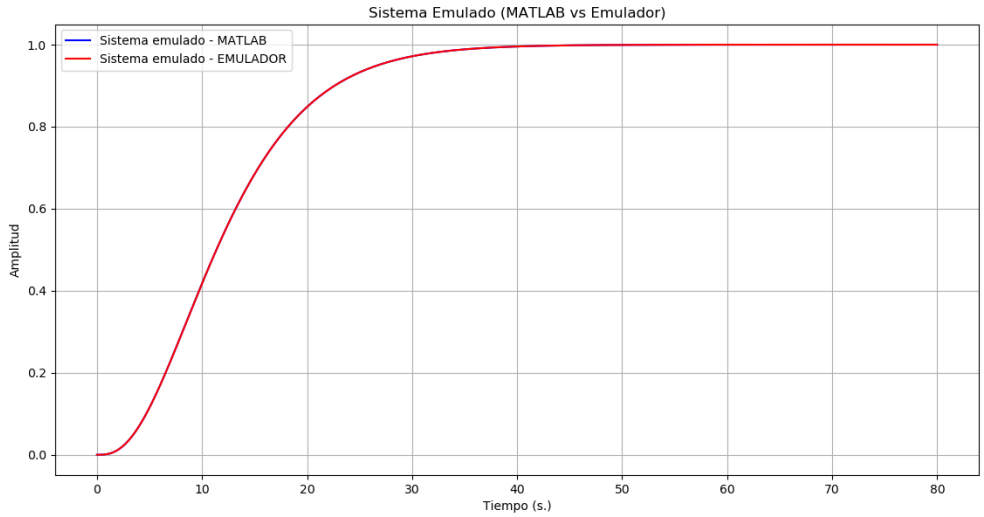
Con los resultados obtenidos después del proceso de emulación de un sistema de Orden 2, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

- Para Orden 3:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. EL modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{76.765625 * s^3 + 54.1875 * s^2 + 12.75 * s + 1} \qquad (V\text{-}26)$$

Gráfica V-38 Sistema Emulado de Orden 3



Nro de datos : 8000 ECM : 8.35308127183788e-14

Fuente: Elaboración Propia

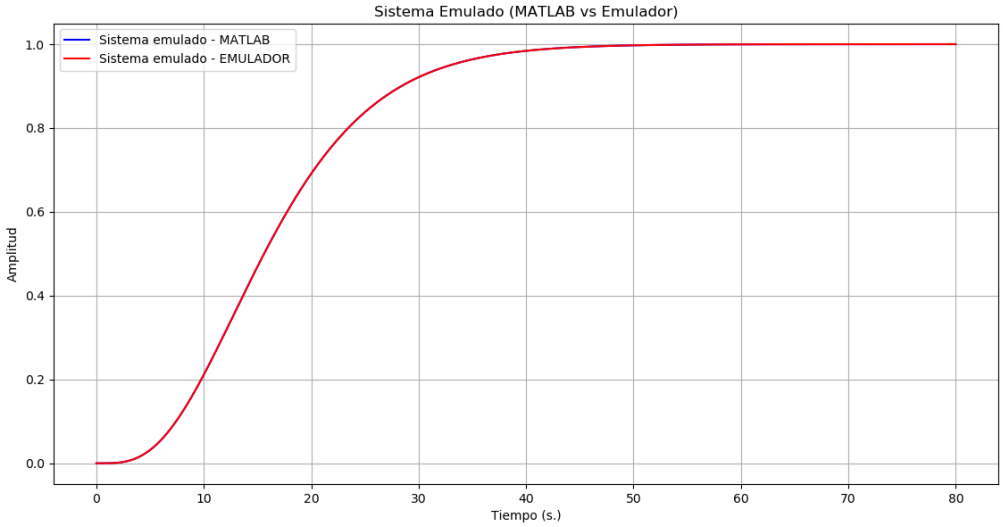
Con los resultados obtenidos después del proceso de emulación de un sistema de Orden 3, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

- Para Orden 4:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. EL modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{326.25390625 * s^4 + 307.0625 * s^3 + 108.375 * s^2 + 17 * s + 1} \qquad (V\text{-}27)$$

Gráfica V-39 Sistema Emulado de Orden 4



Nro de datos : 8000 ECM : 5.035903806307318e-10

Fuente: Elaboración Propia

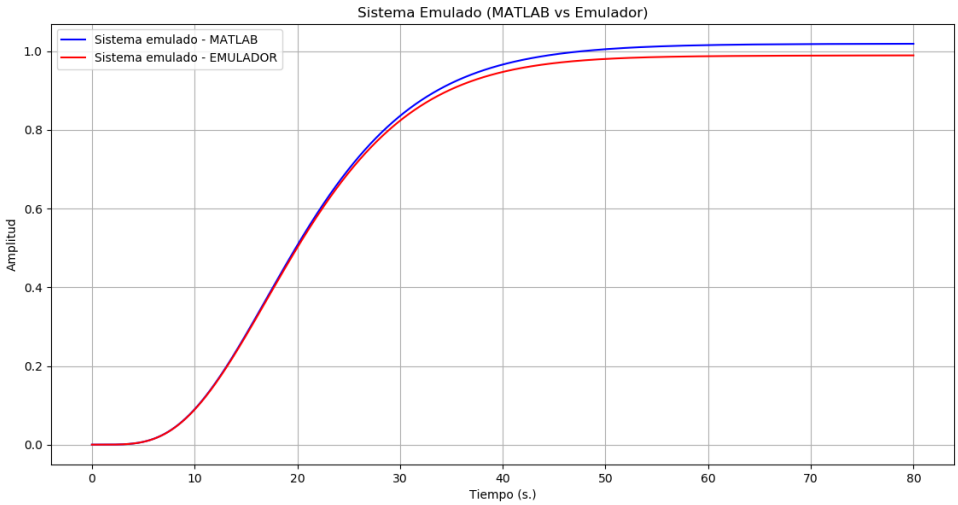
Con los resultados obtenidos después del proceso de emulación de un sistema de Orden 4, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

- Para Orden 5:

  Se utilizo el método de Aproximaciones de Tustin, con cuenta una tasa de muestreo del 10ms. EL modelo ingresado fue el siguiente:

$$G(s) = \frac{1}{(4.25 * s + 1)^5} \qquad (V\text{-}28)$$

Gráfica V-40 Sistema Emulado de Orden 5



Nro de datos : 8000 ECM : 0.0004052997512659007

Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de emulación de un sistema de Orden 5, se muestra un error cuadrático medio muy pequeño en los resultados obtenidos.

5.2.4 Variable Y

Para la variable.

y: Implementación de los algoritmos de identificación y emulación en un sistema embebido para la captura y visualización de las gráficas de entrada y salida, las métricas para la validación son las siguientes:

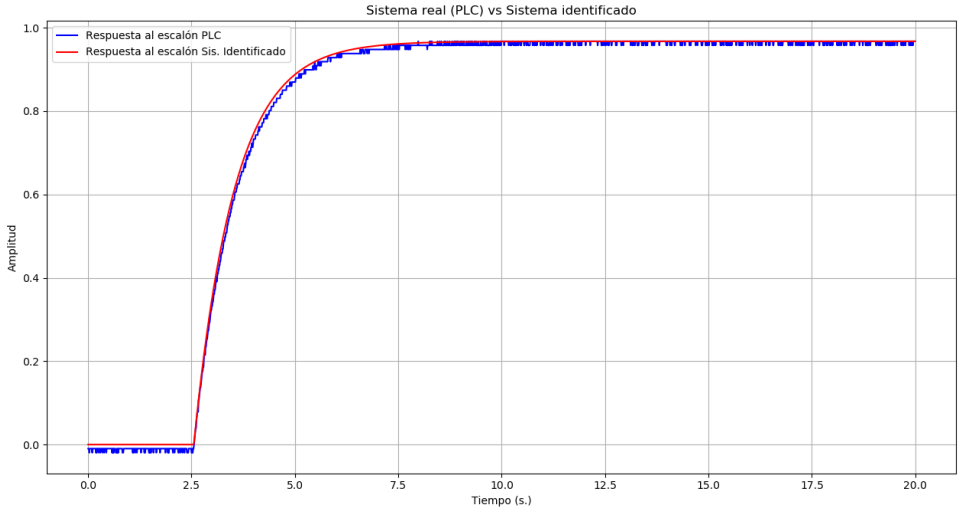
a) Implementar los algoritmos en el sistema embebido:

- Implementación del algoritmo de identificación.
    i. Hasta con 4 decimales.

Para Orden 1

Se obtuvo los siguientes resultados:

Gráfica V-41 Sistema de Orden 1 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 9.08E-05

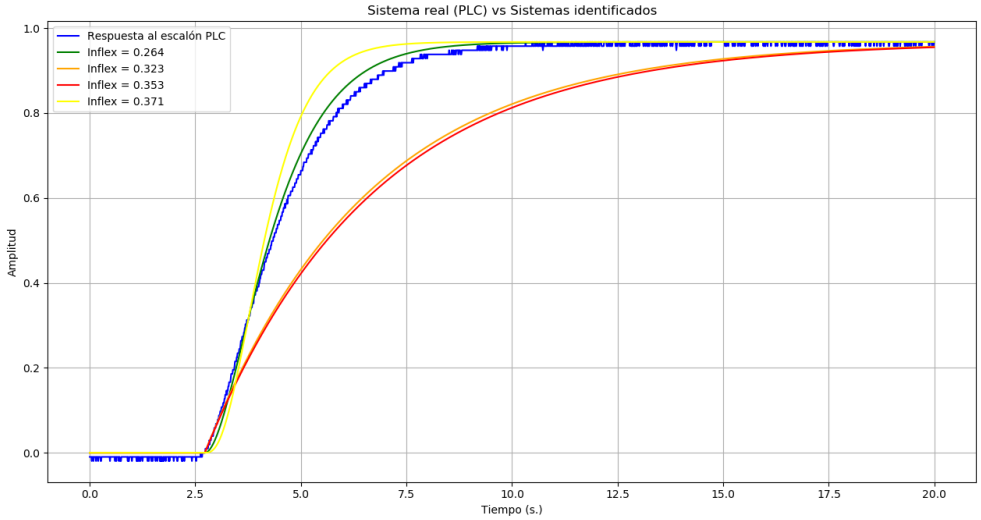
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación de un sistema de Orden 1, se muestra un error cuadrático medio muy pequeño.

Para Orden 2

Se obtuvo los siguientes resultados:

Gráfica V-42 Sistema de Orden 2 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 0.000299727496828897

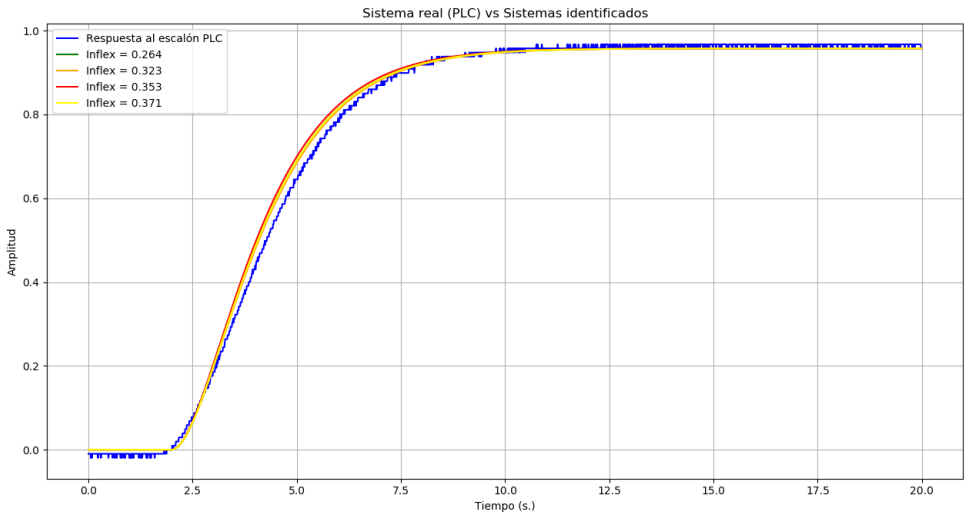
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación de un sistema de Orden 2, se muestra un error cuadrático medio muy pequeño.

Para Orden 3

Se obtuvo los siguientes resultados:

Gráfica V-43 Sistema de Orden 3 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 0.000312425968998086

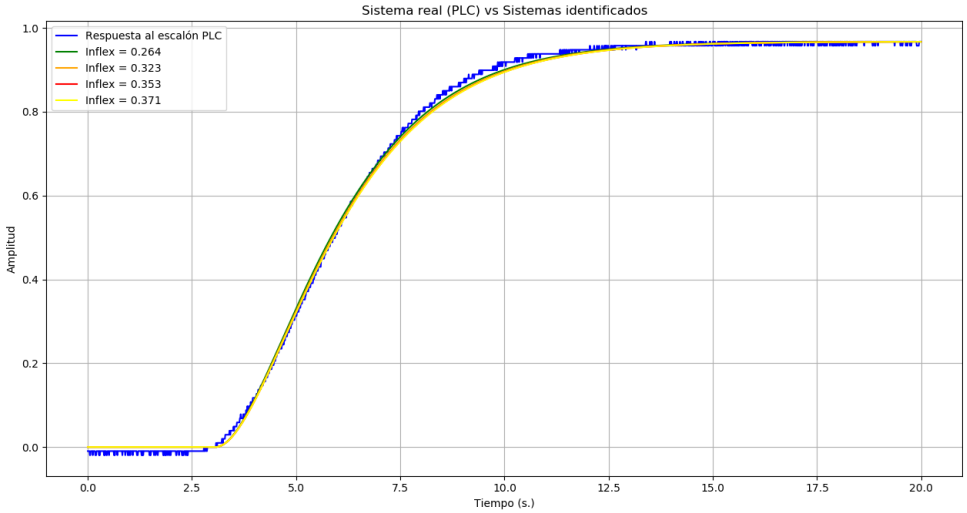
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación nos muestra que se trata de Orden 2, pero a su vez muestra un error cuadrático medio muy pequeño.

Para Orden 4

Se obtuvo los siguientes resultados:

Gráfica V-44 Sistema de Orden 4 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 0.0000970596971026697

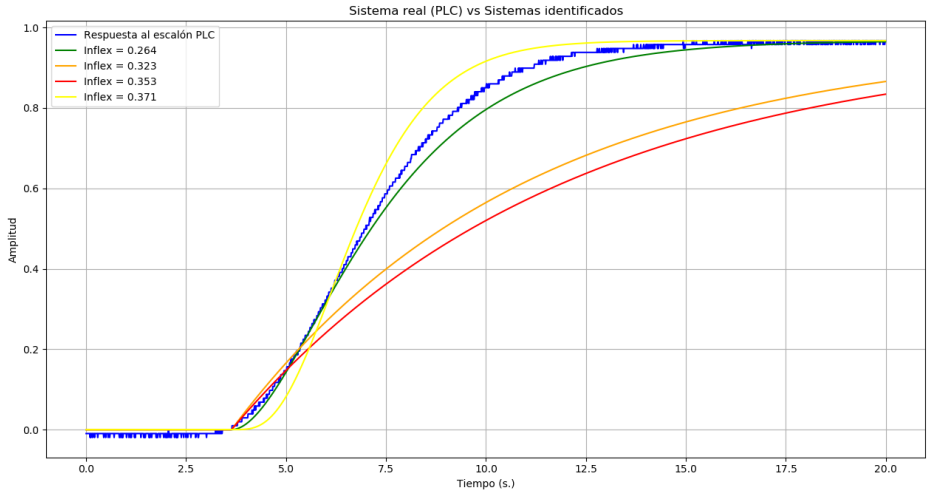
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación nos muestra que se trata de Orden 2, pero a su vez muestra un error cuadrático medio muy pequeño.

Para Orden 5

Se obtuvo los siguientes resultados:

Gráfica V-45 Sistema de Orden 5 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 0.000713667657743409

Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación nos muestra que se trata de Orden 2, pero a su vez muestra un error cuadrático medio muy pequeño.

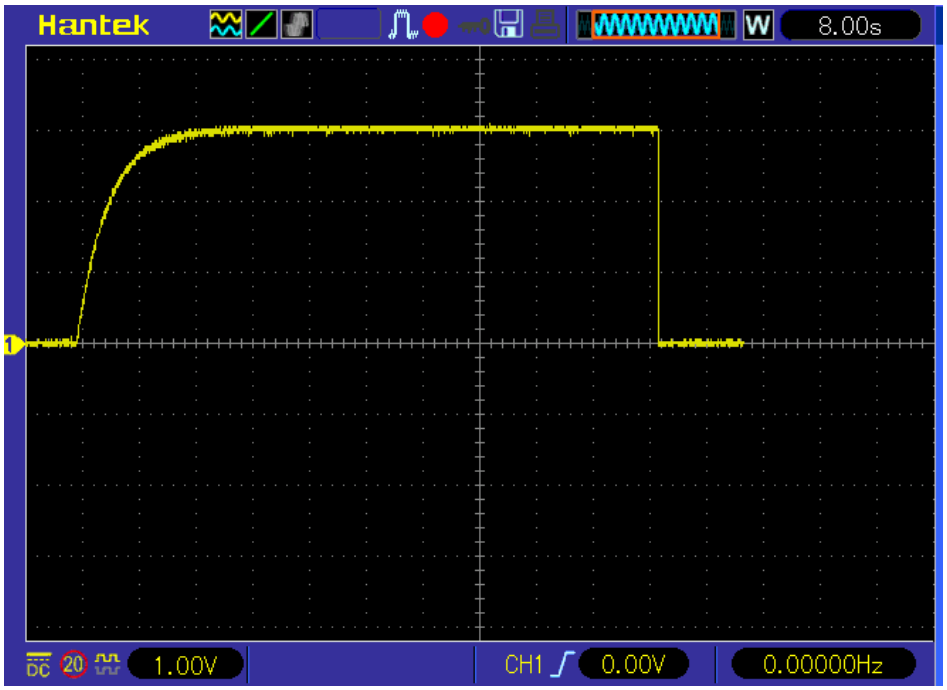
b) Implementar los algoritmos en el sistema embebido:
   - Implementación del algoritmo de emulación.
     i. Hasta con 4 decimales.

Para Orden 1

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 1 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 1 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.
- Se obtuvo las siguientes métricas.

Gráfica V-46 Sistema de Orden 1 emulado por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.000090836473234808

Fuente: Elaboración Propia

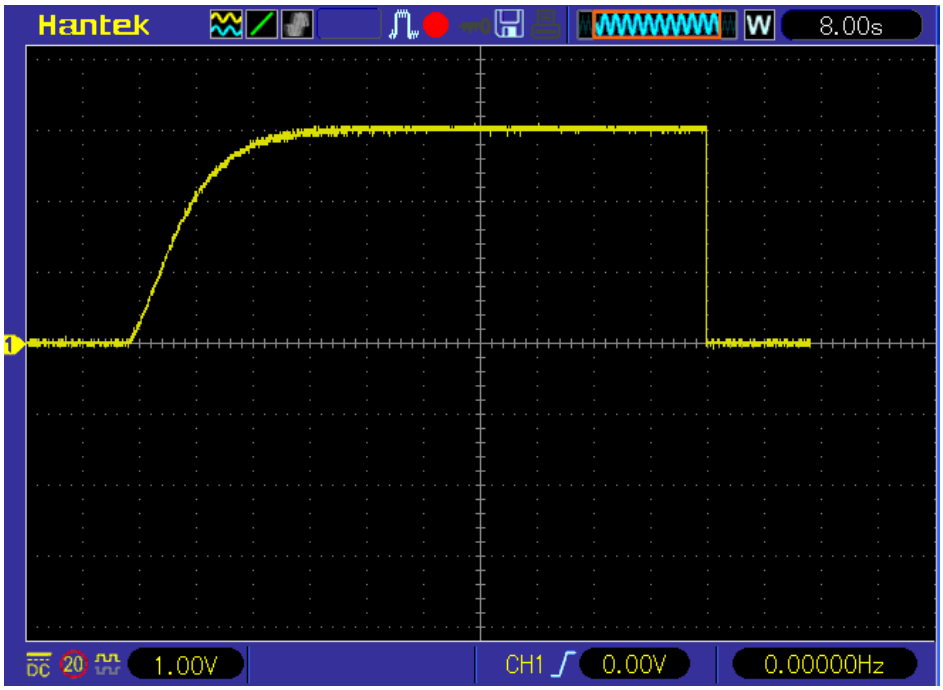
Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño.

Para Orden 2

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 2 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 2 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.
- Se obtuvo las siguientes métricas.

Gráfica V-47 Sistema de Orden 2 emulador por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.0001458159064231 89

Fuente: Elaboración Propia

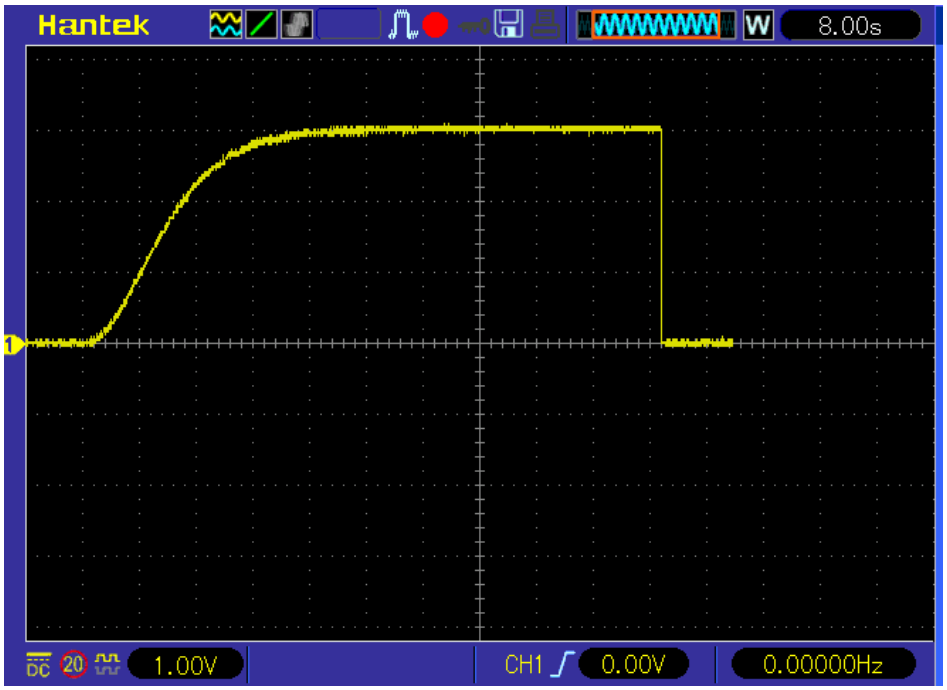
Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño.

Para Orden 3

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 3 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 3 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.
- Se obtuvo las siguientes métricas.

Gráfica V-48 Sistema de Orden 3 emulado por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.000148321617069744

Fuente: Elaboración Propia

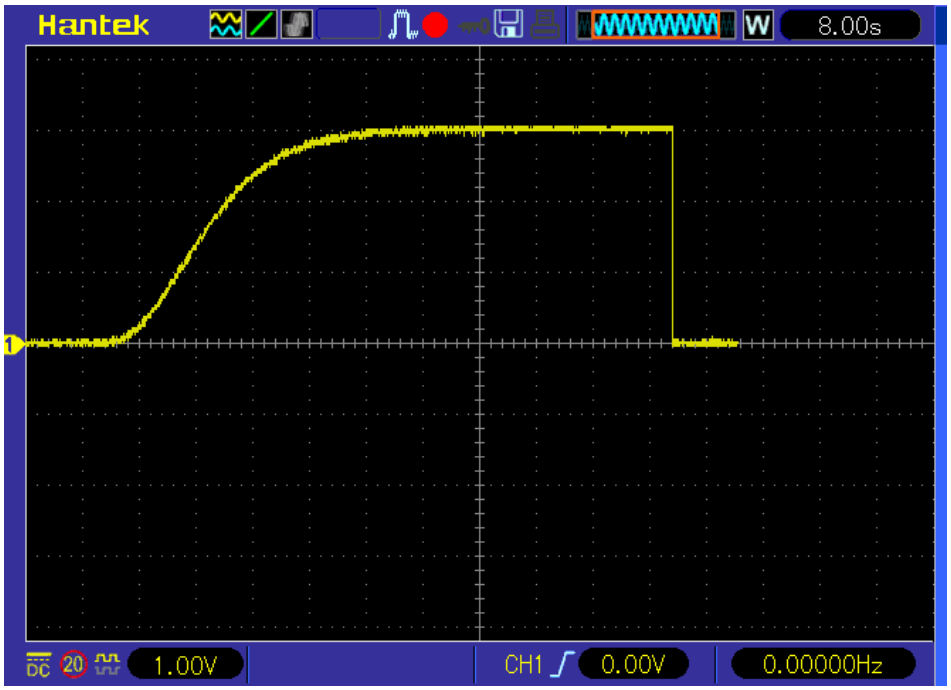
Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño.

Para Orden 4

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 4 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 4 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.
- Se obtuvo las siguientes métricas.

Gráfica V-49 Sistema de Orden 4 emulado por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.0001608187091702 72

Fuente: Elaboración Propia

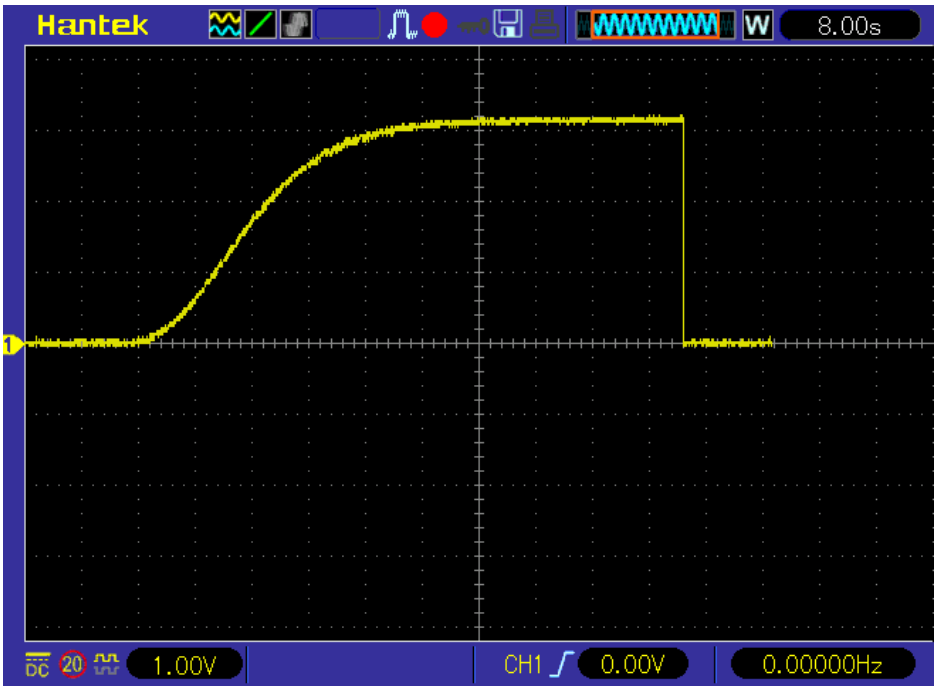
Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño.

Para Orden 5

Se procedió a realizar el proceso de identificación tomando en cuenta las siguientes consideraciones:

- Se ingreso un modelo de Orden 5 al Sistema Embebido el cual cada 10ms envía los datos a través de una salida analógica.
- Este modelo de Orden 5 fue generado por el usuario que desea emular esa función de transferencia y posteriormente es observada en el osciloscopio.
- Se obtuvo las siguientes métricas.

Gráfica V-50 Sistema de Orden 5 emulado por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.000566840763194502

Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño.

VI DISCUSIÓN DE LOS RESULTADOS

6.1 Contrastación y demostración de los resultados

6.1.1 Desarrollo de las interfaces de acondicionamiento de las señales de entrada y salida

Desarrollo de las interfaces de acondicionamiento de las señales de entrada y salida para la identificación y emulación de un sistema SISO, las métricas para la validación son las siguientes:

a) Los rangos de entrada y salida analógica:
   - Nivel de tensión: 0-10v.
     i. Con un error menor a 5%
   - Nivel de corriente: 4-20mA.
     i. Con un error menor a 5%

Tabla VI-1 Tabla de resultados simulados de salida 0-4V a 0-10V

Promedio del % de Error	0.27302
Media del % de Error	0.18033
Desviación del % de Error	0.23713
ECM	0.00010

Fuente: Elaboración propia

Tabla VI-2 Tabla de resultados del circuito de acondicionamiento de salida 0-4V a 0-10V

Promedio del % de Error	0.16084
Media del % de Error	0.16828
Desviación del % de Error	0.16745
ECM	0.00004

Fuente: Elaboración propia

Tabla VI-3 Tabla de resultados simulados de entrada de 0-10V

Promedio del % de Error	0.16207
Media del % de Error	0.12516
Desviación del % de Error	0.08230
ECM	0.00017

Fuente: Elaboración propia

Tabla VI-4 Tabla de resultados del circuito de acondicionamiento de 0-10V

Promedio del % de Error	0.13745
Media del % de Error	0.12516
Desviación del % de Error	0.03811
ECM	0.00011

Fuente: Elaboración propia

Tabla VI-5 Tabla de datos simulados de 4-20mA a 0-10V

Promedio del % de Error	0.80820
Media del % de Error	0.75567
Desviación del % de Error	0.12521
ECM	0.00232

Fuente: Elaboración propia

Tabla VI-6 Tabla de resultados del circuito de 4-20mA a 0-10V

Promedio del % de Error	0.01915
Media del % de Error	-0.04998
Desviación del % de Error	0.28373
ECM	0.00023

Fuente: Elaboración propia

Tabla VI-7 Tabla de resultados simulados de 0-4V a 4-20mA

Promedio del % de Error	-0.693
Media del % de Error	-0.728
Desviación del % de Error	0.347
ECM	0.013975

Fuente: Elaboración Propia.

Tabla VI-8 Tabla de resultados simulados de 0-4V a 4-20mA

Promedio del % de Error	-0.074
Media del % de Error	0.055
Desviación del % de Error	0.353
ECM	0.000357

Fuente: Elaboración Propia.

Con el resultado expuesto en las diferentes tablas para los diferentes circuitos de acondicionamiento tanto simulados como implementados el error es menor al 1% y los resultados son consistentes.

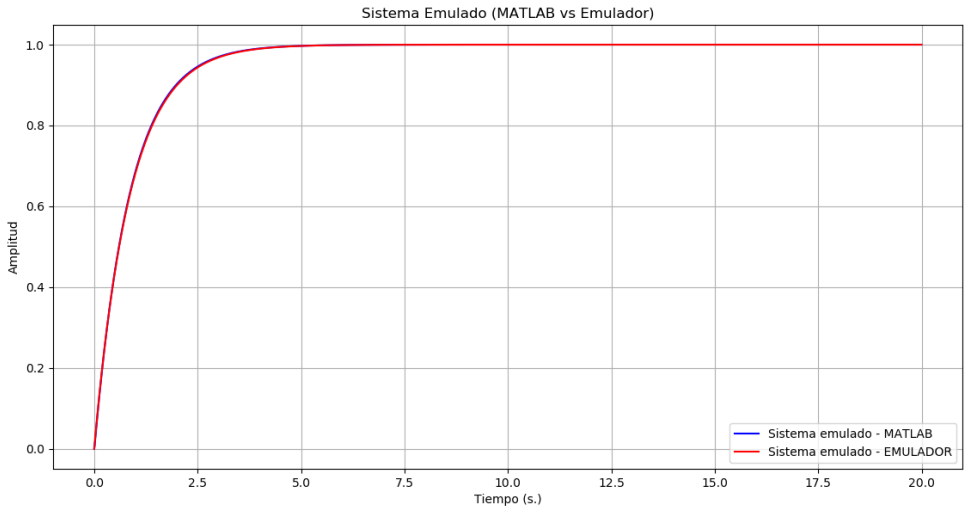
6.1.2 Variable X1:

X1: Desarrollo del algoritmo de identificación de un sistema SISO hasta Orden 5, haciendo uso de herramientas de software de tipo open source, las métricas para la validación son las siguientes:

a) Desarrollo de Algoritmos de identificación

- Para Orden 1:

Gráfica VI-1 Sistema Identificado de Orden 1

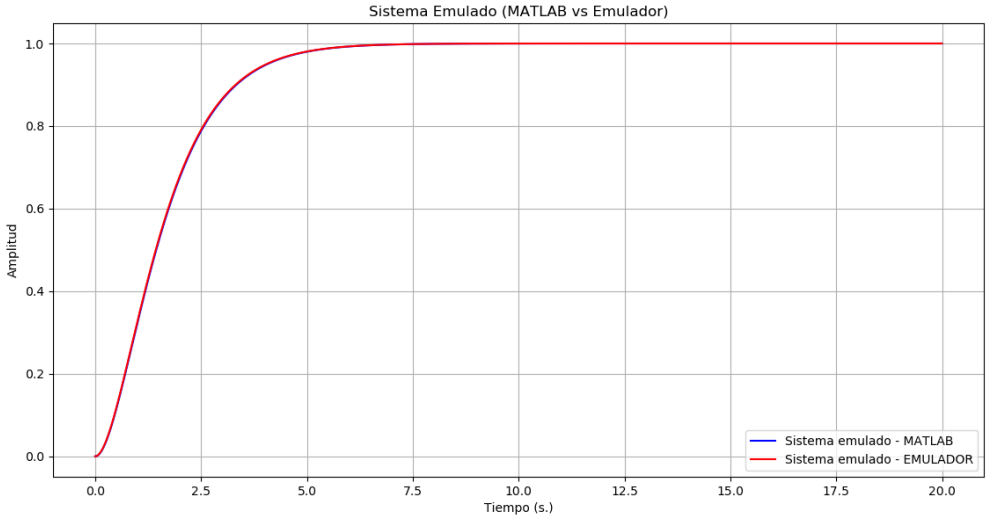


Nro de datos : 2000 ECM : 1.5320307550003483e-06

Fuente: Elaboración Propia

- Para Orden 2:

Gráfica VI-2 Sistema Identificado de Orden 2

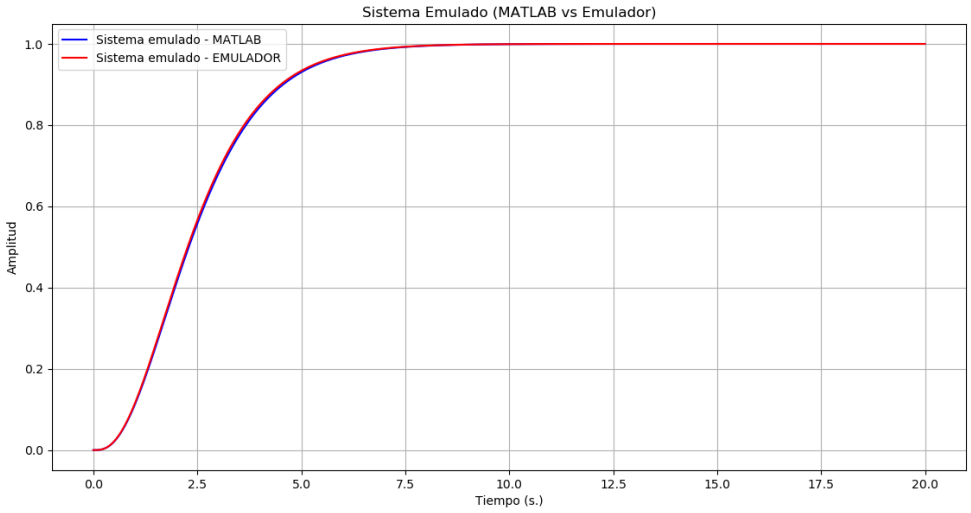


Nro de datos : 2000 ECM : 1.9482809400916138e-06

Fuente: Elaboración Propia

- Para Orden 3:

Gráfica VI-3 Sistema Identificado de Orden 3

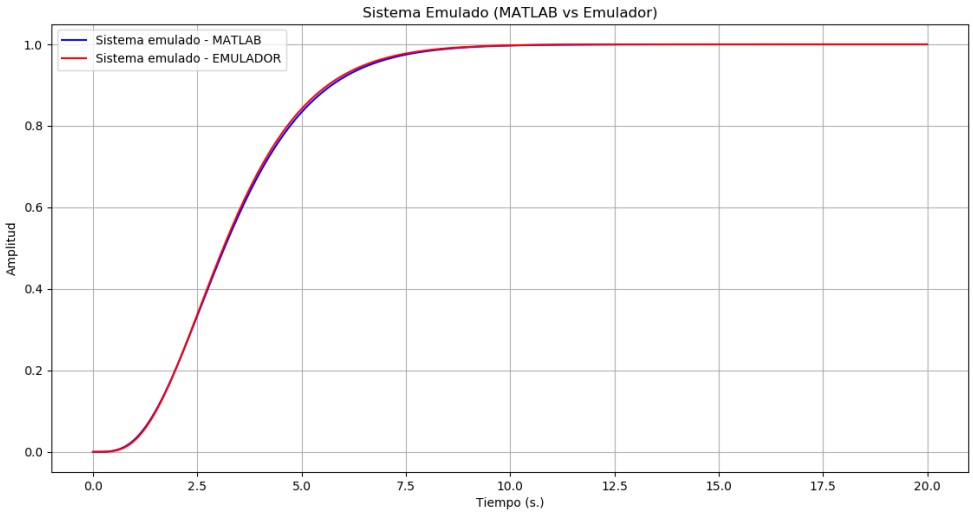


Nro de datos : 2000 ECM : 1.0005484383312472e-05

Fuente: Elaboración Propia

- Para Orden 4:

Gráfica VI-4 Sistema Identificado de Orden 4

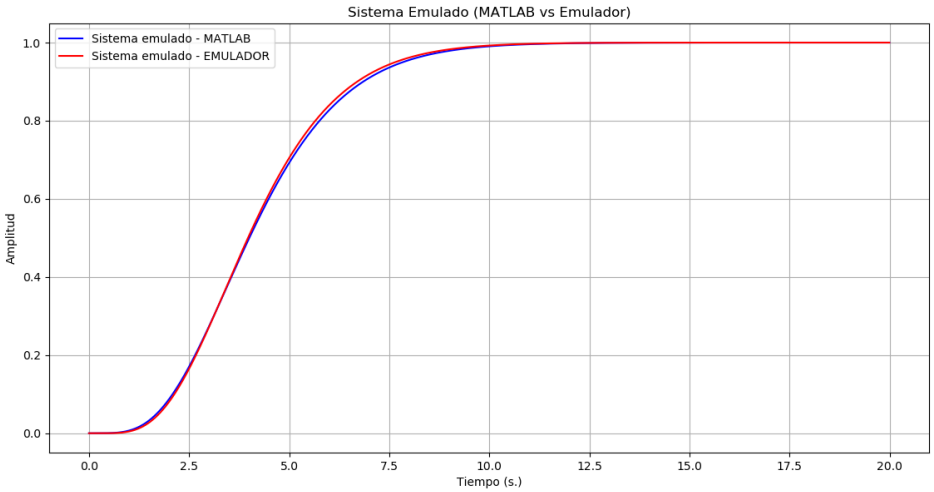


Nro de datos : 2000 ECM : 1.2234121903681746e-05

Fuente: Elaboración Propia

- Para Orden 5:

Gráfica VI-5 Sistema Identificado de Orden 5



Nro de datos : 2000 ECM : 1.2234121903681746e-05

Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación para los diferentes tipos de Orden (1,2,3,4 y 5), se demuestra un error cuadrático medio muy pequeño por lo que se ha cumplido la métrica de la hipótesis para la variable X1.

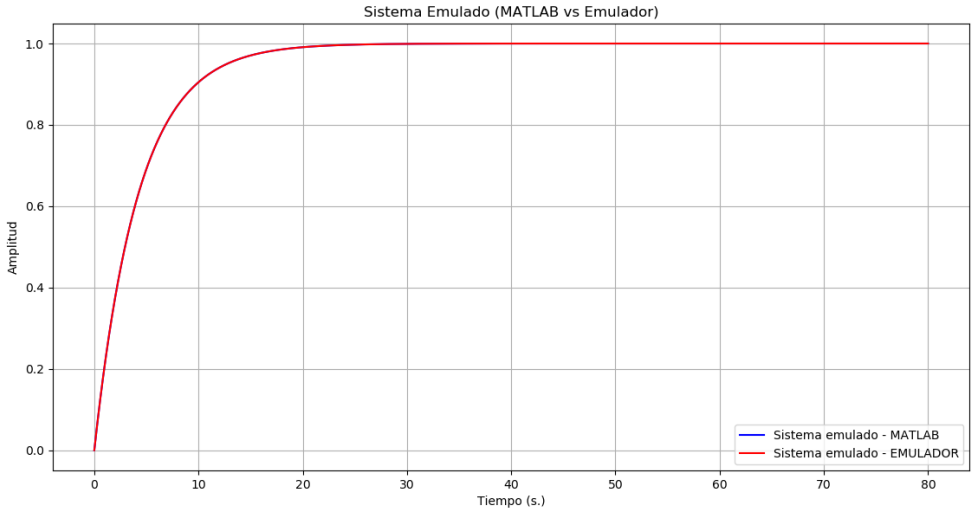
6.1.3 Variable X2:

X2: Desarrollo de los algoritmos de emulación de un sistema SISO, haciendo uso de herramientas de software de tipo open source, las métricas para la validación son las siguientes:

a) Desarrollo de algoritmos de emulación:

- Para Orden 1:

Gráfica VI-6 Sistema Emulado de Orden 1

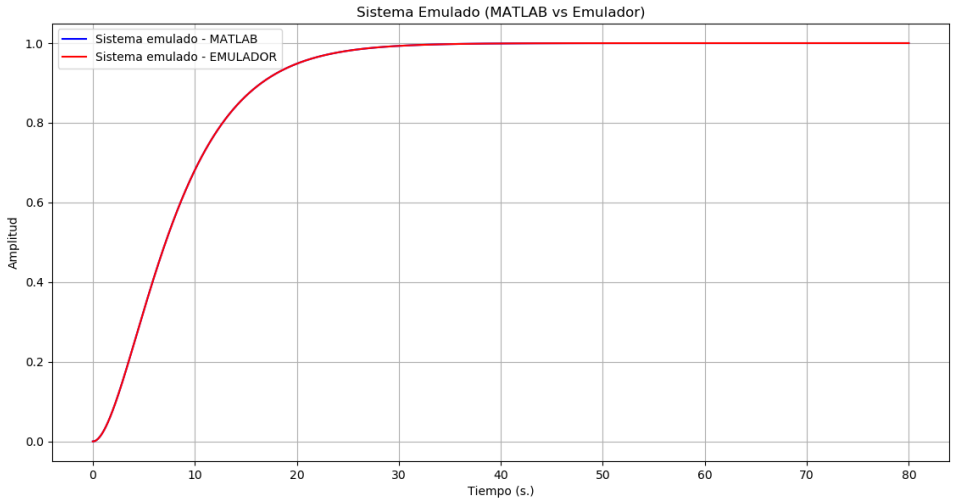


Nro de datos : 8000 ECM : 7.108899895489959e-14

Fuente: Elaboración Propia

- Orden 2:

Gráfica VI-7 Sistema Emulado de Orden 2

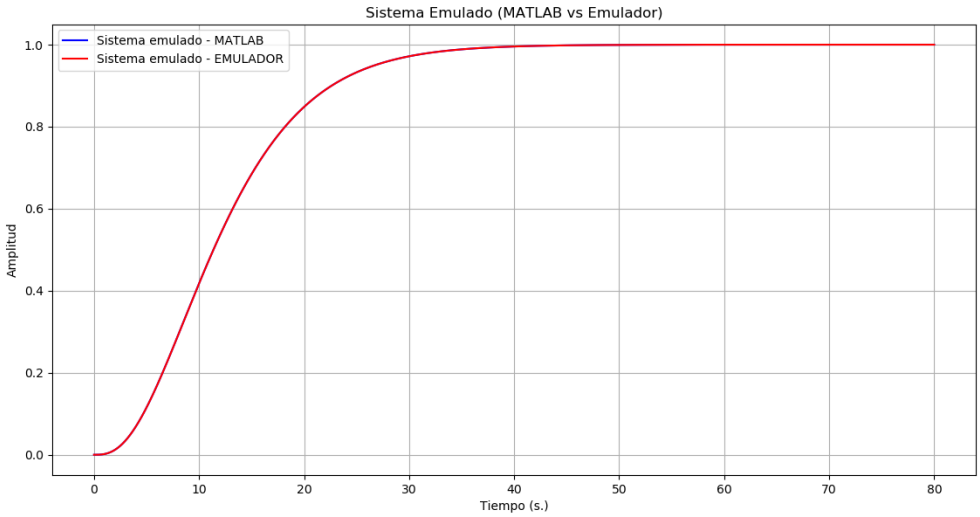


Nro de datos : 8000 ECM : 8.412175039039374e-14

Fuente: Elaboración Propia

- Para Orden 3:

Gráfica VI-8 Sistema Emulado de Orden 3

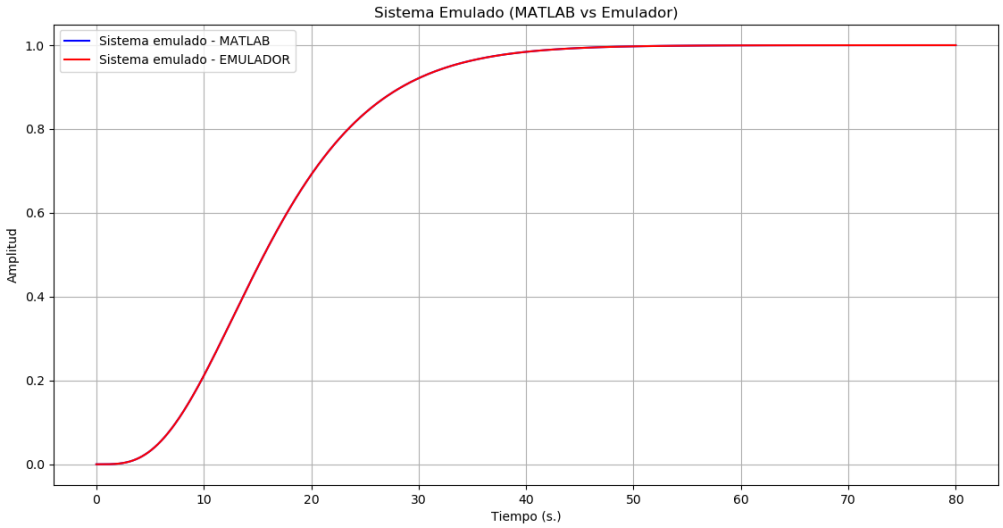


Nro de datos : 8000 ECM : 8.35308127183788e-14

Fuente: Elaboración Propia

- Para Orden 4:

Gráfica VI-9 Sistema Emulado de Orden 4

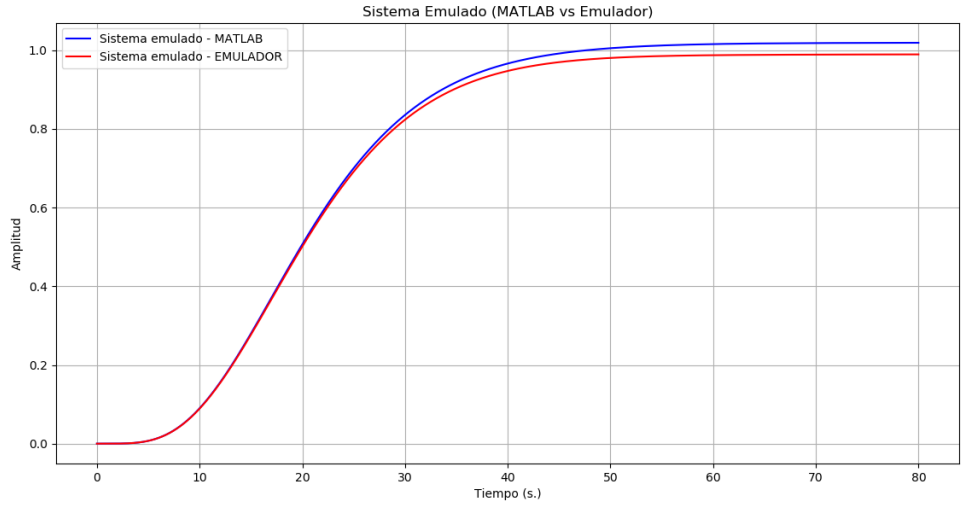


Nro de datos : 8000 ECM : 5.035903806307318e-10

Fuente: Elaboración Propia

- Para Orden 5:

Gráfica VI-10 Sistema Emulado de Orden 5



Nro de datos : 8000 ECM : 0.0004052997512659007

Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de emulación para los diferentes tipos de Orden (1,2,3,4 y 5), se demuestra un error cuadrático medio muy pequeño por lo que se ha cumplido la métrica de la hipótesis para la variable X2.

6.1.4 Variable Y

Para la variable.

y: Desarrollo de un sistema portátil.

Implementación de los algoritmos de identificación y emulación en un sistema embebido para la captura y visualización de las gráficas de entrada y salida, las métricas para la validación son las siguientes:

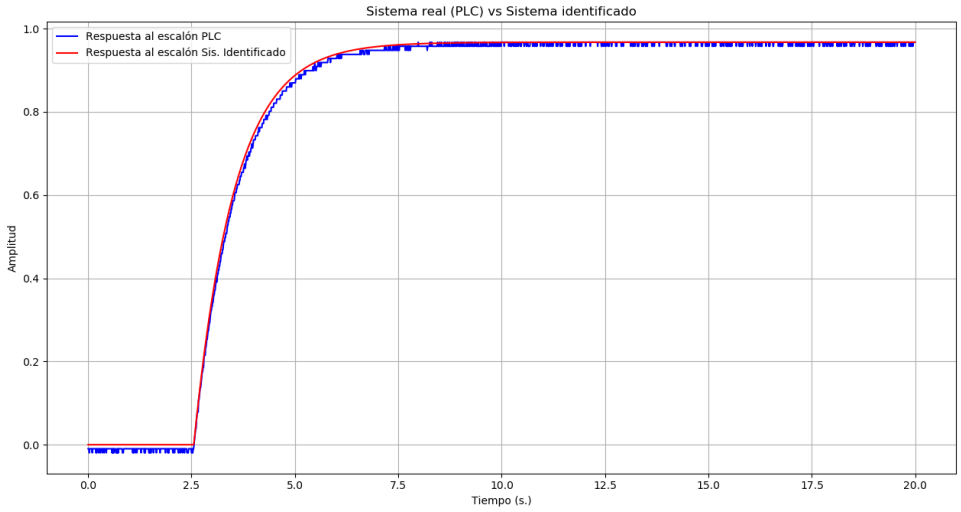
a) Implementar los algoritmos en el sistema embebido:

- Implementación del algoritmo de identificación.
  i. Hasta con 4 decimales.

Para Orden 1

Se obtuvo los siguientes resultados:

Figura VI-1 Sistema de Orden 1 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 9.08E-05

Tabla VI-9 Resultados de la identificación de Orden 1

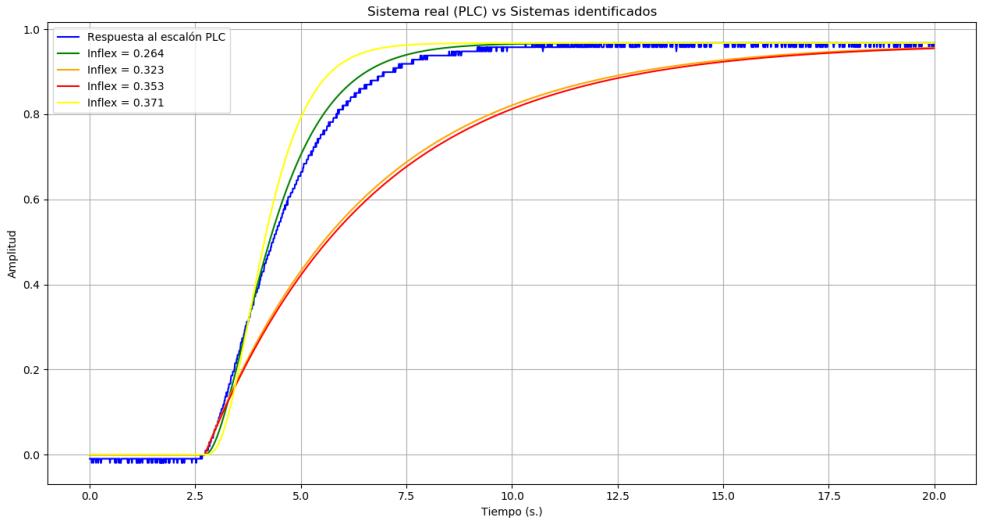
PRIMER ORDEN	
Nro de datos	2000
Inflexión	Automática
Filtro IIR	NO
Filtro Suavizado	NO
Error cuadrático medio	9.08E-05
Correlación	0.999840954
Tau	0.98
Orden	1
Delay	2.56

Fuente: Elaboración Propia

Para Orden 2

Se obtuvo los siguientes resultados:

Figura VI-2 Sistema de Orden 2 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 0.000299727496828897

Fuente: Elaboración Propia

Tabla VI-10 Resultados de la identificación de Orden 2

SEGUNDO ORDEN	Grafica Verde	Grafica Naranja	Grafica Roja	Grafica Amarilla
Nro de datos	2000	2000	2000	2000
Inflexión	0.264	0.323	0.353	0.371
Filtro IIR	SI	SI	SI	SI
Filtro Suavizado	NO	NO	NO	NO
Error cuadrático medio	0.000299727	0.015985733	0.017622461	0.001901372
Correlación	0.999227024	0.969101325	0.966581906	0.994524682
Tau	0.882	3.85	3.97	0.513
Orden	2	1	1	3
Delay	2.73	2.73	2.73	2.73

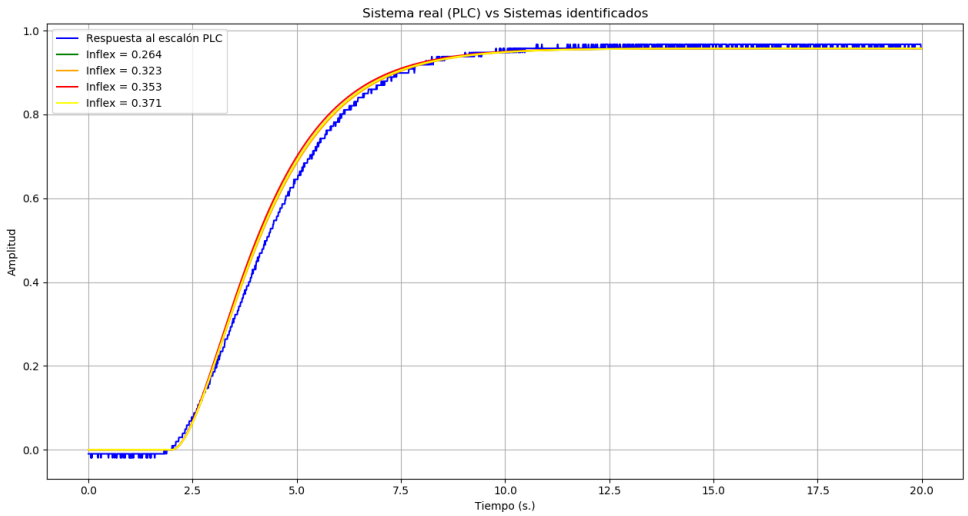
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación de un sistema de Orden 2, se muestra un error cuadrático medio muy pequeño.

Para Orden 3

Se obtuvo los siguientes resultados:

Figura VI-3 Sistema de Orden 3 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 0.0003124259689980 86

Fuente: Elaboración Propia

Tabla VI-11 Resultados de la identificación de Orden 3

TERCER ORDEN	Grafica Verde	Grafica Naranja	Grafica Roja	Grafica Amarilla
Nro de datos	2000	2000	2000	2000
Inflexión	0.264	0.323	0.353	0.371
Filtro IIR	SI	SI	SI	SI
Filtro Suavizado	SI	SI	SI	SI
Error cuadrático medio	0.000312426	0.001045149	0.000454265	0.000312426
Correlación	0.998944139	0.996626902	0.998485642	0.998944139
Tau	1.184	1.197	1.16	1.184
Orden	2	2	2	2
Delay	2	2	2	2

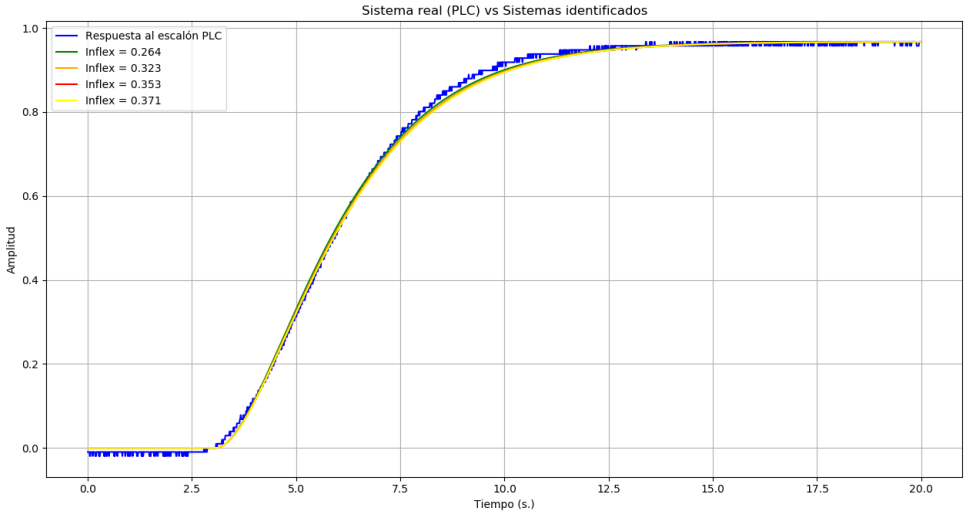
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación nos muestra que se trata de Orden 2, pero a su vez muestra un error cuadrático medio muy pequeño.

Para Orden 4

Se obtuvo los siguientes resultados:

Figura VI-4 Sistema de Orden 4 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 0.0000970596971026697

Fuente: Elaboración Propia

Tabla VI-12 Resultados de la identificación de Orden 4

CUARTO ORDEN	Grafica Verde	Grafica Naranja	Grafica Roja	Grafica Amarilla
Nro de datos	2000	2000	2000	2000
Inflexión	0.264	0.323	0.353	0.371
Filtro IIR	SI	SI	SI	SI
Filtro Suavizado	SI	SI	SI	SI
Error cuadrático medio	0.000097060	0.000114048	0.000132235	0.000135881
Correlación	0.999741021	0.999700934	0.999661767	0.999654047
Tau	1.596	1.619	1.631	1.633
Orden	2	2	2	2
Delay	3.07	3.07	3.07	3.07

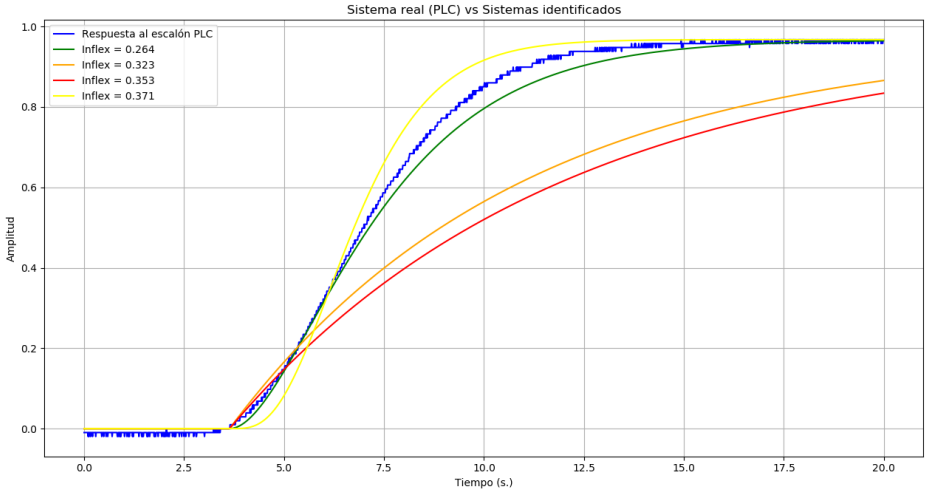
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación nos muestra que se trata de Orden 2, pero a su vez muestra un error cuadrático medio muy pequeño.

Para Orden 5

Se obtuvo los siguientes resultados:

Figura VI-5 Sistema de Orden 5 Identificado



Nro de Datos: 2000 Error Cuadrático medio: 0.000713667657743409

Fuente: Elaboración Propia

Tabla VI-13 Resultados de la identificación de Orden 5

QUINTO ORDEN	Grafica Verde	Grafica Naranja	Grafica Roja	Grafica Amarilla
Nro de datos	2000	2000	2000	2000
Inflexión	0.264	0.323	0.353	0.371
Filtro IIR	SI	SI	SI	SI
Filtro Suavizado	NO	NO	NO	NO
Error cuadrático medio	0.000713668	0.030239233	0.042872416	0.001624389
Correlación	0.998877935	0.981888694	0.97687636	0.996614026
Tau	2.017	7.26	8.25	0.83
Orden	2	1	1	4
Delay	3.64	3.64	3.64	3.64

Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de identificación nos muestra que se trata de Orden 2, pero a su vez muestra un error cuadrático medio muy pequeño.

Con los resultados obtenidos después del proceso de identificación para los diferentes tipos de Orden (1,2,3,4 y 5), se demuestra un error cuadrático medio muy pequeño por lo que se ha cumplido la métrica de la hipótesis para la variable Y.

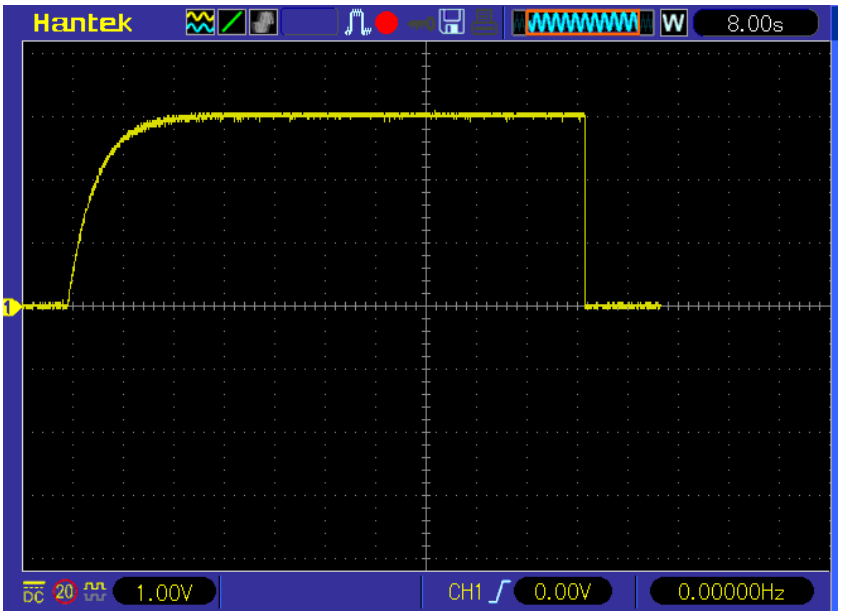
b) Implementar los algoritmos en el sistema embebido:

- Implementación del algoritmo de emulación.
    i. Hasta con 4 decimales.

Para Orden 1

- Se obtuvo las siguientes métricas.

Gráfica VI-11 Sistema de Orden 1 emulado por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.000090836473234808

Fuente: Elaboración Propia

Tabla VI-14 Resultados de emulación por el usuario de Orden 1

PRIMER ORDEN	
Nro de datos	2050
Tau	4.25
Orden	1
Delay	0
Tiempo Total	82
Error cuadrático medio	0.000090836
Correlación	0.999840954

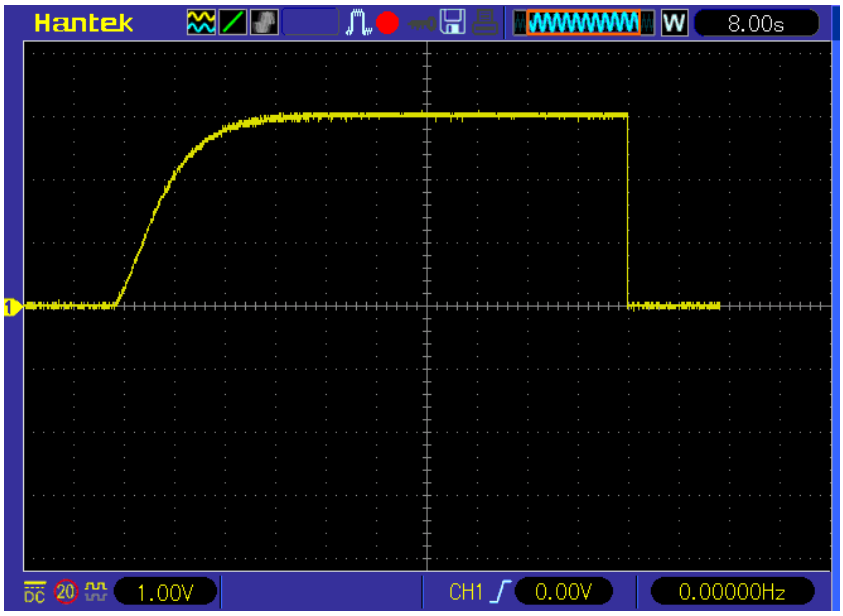
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño y una fuerte correlación que tiende a 1.

Para Orden 2

- Se obtuvo las siguientes métricas.

Gráfica VI-12 Sistema de Orden 2 emulador por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.000145815906423189

Fuente: Elaboración Propia

Tabla VI-15 Resultados de emulación por el usuario de Orden 2

SEGUNDO ORDEN	
Nro de datos	2050
Tau	4.25
Orden	2
Delay	0
Tiempo Total	82
Error cuadrático medio	0.000145815
Correlación	0.999974136

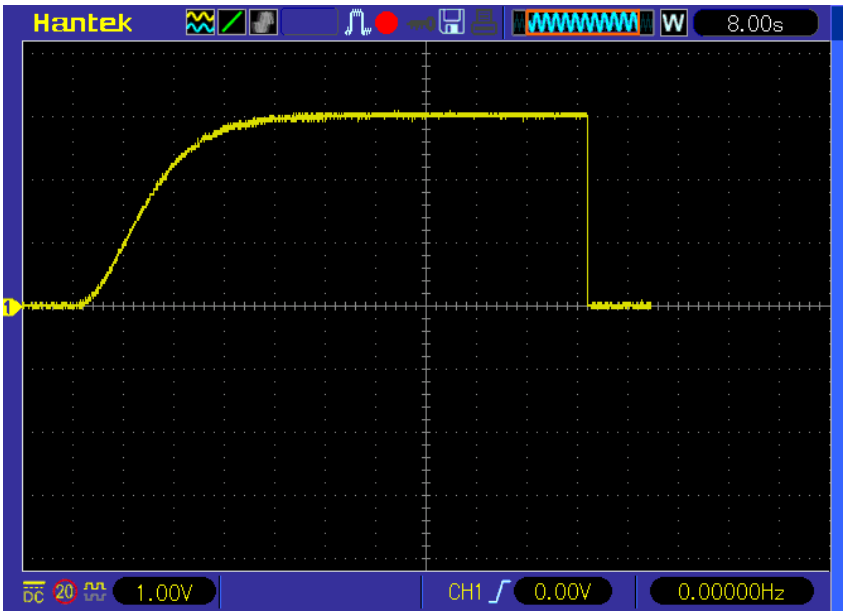
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño y una fuerte correlación que tiende a 1.

Para Orden 3

- Se obtuvo las siguientes métricas.

Gráfica VI-13 Sistema de Orden 3 emulado por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.000148321617069744

Fuente: Elaboración Propia

Tabla VI-16 Resultados de emulación por el usuario de Orden 3

TERCER ORDEN	
Nro de datos	2050
Tau	4.25
Orden	3
Delay	0
Tiempo Total	82
Error cuadrático medio	0.000148321
Correlación	0.999844544

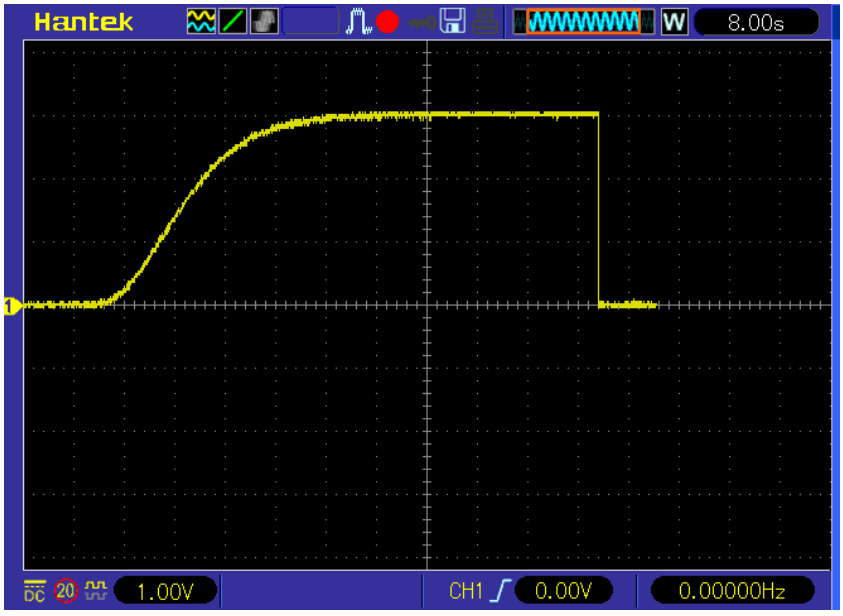
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño y una fuerte correlación que tiende a 1.

Para Orden 4

- Se obtuvo las siguientes métricas.

Gráfica VI-14 Sistema de Orden 4 emulado por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.000160818709170272

Fuente: Elaboración Propia

Tabla VI-17 Resultados de emulación por el usuario de Orden 4

CUARTO ORDEN	
Nro de datos	2050
Tau	4.25
Orden	4
Delay	0
Tiempo Total	82
Error cuadrático medio	0.000160818
Correlación	0.999845647

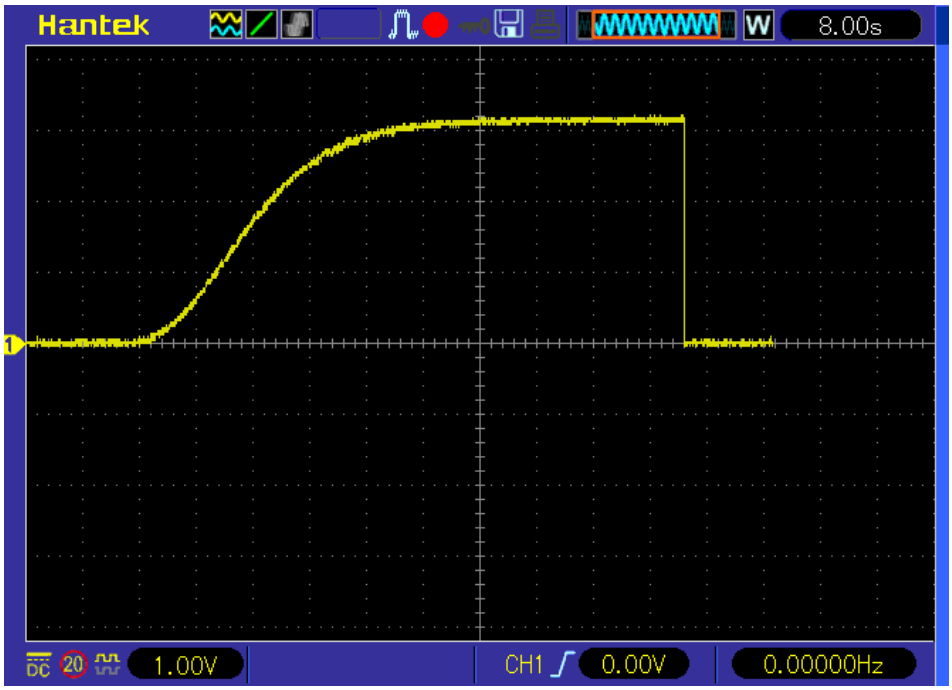
Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño y una fuerte correlación que tiende a 1.

Para Orden 5

- Se obtuvo las siguientes métricas.

Gráfica VI-15 Sistema de Orden 5 emulado por el usuario



Nro de Datos: 2050 Error Cuadrático medio: 0.000566840763194502

Fuente: Elaboración Propia

Tabla VI-18 Resultados de emulación por el usuario de Orden 5

QUINTO ORDEN	
Nro de datos	2050
Tau	4.25
Orden	5
Delay	0
Tiempo Total	82
Error cuadrático medio	0.000566840
Correlación	0.999998623

Fuente: Elaboración Propia

Con los resultados obtenidos después del proceso de emulación se muestra un error cuadrático medio muy pequeño y una fuerte correlación que tiende a 1. Con los resultados obtenidos después del proceso de emulación para los diferentes tipos de Orden (1,2,3,4 y 5), se demuestra un error cuadrático medio muy pequeño por lo que se ha cumplido la métrica de la hipótesis para la variable Y.

Tabla VI-19 Tabla Resumen

	Orden 1 ECM	Orden 2 ECM	Orden 3 ECM	Orden 4 ECM	Orden 5 ECM
X1: Identificación de un sistema SISO hasta Orden 5.	1.53203e-06	1.94828e-06	1.00054e-05	1.22341e-05	1.22341e-05
X2: Emulación de Funciones de Transferencia hasta Orden 5.	7.10889e-14	8.41217e-14	8.35308e-14	5.03590e-10	4.05299e-05
Y: Sistema portátil con herramientas open source para la identificación y emulación hasta orden 5.	9.08e-05	2.997274e-5	3.124259e-6	9.705969e-6	0.000713e-5

Fuente: Elaboración Propia

CONCLUSIONES

- Para el desarrollo del algoritmo de identificación de un sistema SISO hasta Orden 5 y haciendo uso de herramientas de software de tipo open source, se implementó un algoritmo de identificación por el método de Strejc, el cual tienen una respuesta muy buena si y solo si los datos del sistema a identificar no poseen ruido. Se obtuvo un error cuadrático medio muy bajo al momento de validación del proceso de identificación.

- Para el desarrollo del algoritmo de emulación de un sistema SISO hasta Orden 5 y haciendo uso de herramientas de software de tipo open source, se implementó un algoritmo de aproximación mediante el método de Tustin, el cual mostro un error cuadrático medio muy bajo al momento de la emulación de un proceso. En las pruebas de validación entre el sistema emulado por Matlab y el desarrollado por nuestra investigación, también arrojan valores muy pequeños de error.

- Para el desarrollo de un sistema portátil para la implementación de los algoritmos de identificación y emulación, se realizó el algoritmo de identificación (método de Strejc) en el sistema embebido y depende del punto de inflexión máxima que la curva del modelo a identificar, por lo que cualquier cambio en este vector puede tener incidencia en la identificación. Adicionalmente depende mucho de la calidad de los valores capturados. Para solucionar este inconveniente se implementó un filtro digital para el vector de muestras del sistema a identificar, donde se demostró la identificación del sistema con un error cuadrático medio muy bajo. Al momento de realizar la identificación no solo se debe evaluar el orden del sistema sino también el nivel de similitud del sistema a identificar esto se logra mediante el uso del error cuadrático medio. Donde en todos los casos de obtiene valores muy pequeños. Cuando se implementó el algoritmo de Tustin en el sistema embebido no se tuvo problemas con la señal emulada, su error cuadrático medio es muy bajo. Para el proceso emulación se comprobó que no hay necesidad de hacer

ningún tratamiento adicional, solo se envió la información mediante la salida analógica de la tarjeta de adquisición.

RECOMENDANCIONES

- Para el desarrollo de las interfaces de acondicionamiento de las señales de entrada y salida para la identificación y emulación de un sistema SISO. Es muy importante tener en cuenta la distribución de los componentes en el PCB. Se debe tener en cuenta el correcto sistema de tierra y referencia de las señales, motivo por el cual se puede tener señales con demasiado ruido. Tener instrumentos de medición y de generación de señales que midan o generen según sea el caso con error muy pequeño. Contar con fuentes de alimentación del sistema con bajo ruido. Contar un sistema de acoplamiento entre los sistemas a identificar los circuitos de acondicionamiento de tal manera que puedan reducir el ruido.

- Para la variable X1: Desarrollo del algoritmo de identificación de un sistema SISO hasta Orden 5, haciendo uso de herramientas de software de tipo open source. Se recomienda para próximas investigaciones aumentar los métodos de identificación en un mismo algoritmo y poder apreciar las prestaciones de cada uno de ellos.

- Para la variable X2: Desarrollo del algoritmo de emulación de un sistema SISO hasta Orden 5, haciendo uso de herramientas de software de tipo open source. Se recomienda para próximas investigaciones tener un tiempo de muestreo de hasta de un 1ms, con la intención de poder emular sistemas o procesos con tiempos de respuesta de reacción más rápidos.

- Para la variable Y: Desarrollo de un sistema portátil, la implementación de los algoritmos de identificación y emulación se recomienda para el proceso de identificación tener una mayor variedad de los filtros digitales con la intención de hacer un mejor tratamiento de la señal. Para ambientes con mucho ruido eléctrico el uso de fuentes de muy bajo ruido, o el uso de baterías para la alimentación de identificación.

VII REFERENCIAS BIBLIOGRÁFICAS

¿Qué es Hardware-in-the-Loop? (05 de Marzo de 2019). (National Instruments) Obtenido de https://www.ni.com/es-cr/innovations/white-papers/17/what-is-hardware-in-the-loop-.html

Andrés Ricardo, B. C. (2010). Metodología para la identificación del sistema de excitación de un generador eléctrico de potencia para propósitos de control. Bogota: Universidad Nacional de Colombia.

Angel, Martínez Bueno; Jorge, Pomares. (2011). *Identificación experimental de sistemas.* Grupos de Innovación Tecnológico-Educativa, Innovación Educativa en Automática. Alicante: Universidad de Alicante.

Arriaran, S. S. (2015). *Todo sobre sistemas embebidos.* Lima: UPC.

Computing, M. (2020). *Daq USB.* (Measurement Computing) Obtenido de https://www.mccdaq.com/data-acquisition

COMPUTING, M. (2020). *USB-1208FS-Plus-OEM*. Obtenido de https://www.mccdaq.com/pdfs/manuals/USB-1208FS-Plus-OEM.pdf

Computing, M. (2020). *WHAT IS DATA ACQUISITION*. Norton, MA 02766: https://www.mccdaq.com/index.

Elsa Gladys Torres Saybay, Arturo Francisco Ordoñez Peña, Rodrigo Víctor Alarcón A. y Melvin Leonardo López Franco. (2016). *Open Source en la educación* (2016 ed.). Cuadernos de Educación y Desarrollo. Obtenido de http://www.eumed.net/rev/atlante/2016/01/software-libre.html

Esteban, V. S. (2014). *DESARROLLO Y TESTEO DE UN EMULADOR DE PLANTAS INDUSTRIALES BASADO EN ARDUINO.* La Coruña: Universidade da Coruña.

Fatuev, Victor A.; Mishin Anton A. (2019). Realization of Optimal Identification Tasks for Dynamic Systems in Real Time Scale. *2019 8th MEDITERRANEAN CONFERENCE ON EMBEDDED COMPUTING.* BUDVA, MONTENEGRO.

HERNÁNDEZ-MEDRANO, Israel*†, PINEDA-MARTÍNEZ, Alejandro Gabriel, TAPIA-TINOCO,. (2017). *Herramienta de emulación en tiempo real para*

circuitos eléctricos mediante power hardware in the Loop. Irapuato-Bolivia: Revista de Análisis Cuantitativo y Estadístico.

https://openwebinars.net/blog/que-es-python/. (2020). Obtenido de https://openwebinars.net/blog/que-es-python/

I.JAZIRI, L.CHAARABI, K.JELASSI. (2015). A remote DC motor control using Embedded Linux and FPGA. *7th International Conference on Modelling, Identification and Control.* Sousse, Tunisia.

INEI, I. N. (2019). *Demografía Empresarial en el Perú.* INEI. Lima: INEI. Obtenido de https://www.inei.gob.pe/media/MenuRecursivo/boletines/boletin_demografia_empresarial_nov2019.pdf

Instruments, N. (2020). *https://www.ni.com/es-cr.html*. (National Instruments) Obtenido de https://www.ni.com/data-acquisition/what-is/esa/

Javier Sedano Franco; José Ramón Villar Flecha. (2020). *Técnica Industrial*. Obtenido de http://www.tecnicaindustrial.es/TIFrontal/a-1408-introduccion-identificacion-sistemas.aspx

Juan Carlos, Martínez Quintero; Jaime Eduardo, Andrade Ramírez. (2013). *Implementación de controladores en sistemas retroalimentados usando electrónica embebida y simulación hardware in the loop.* Pereira: Universidad Tecnológica de Pereira.

Juan Pablo, F. A. (2010). *Desarrollo de un Sistema para la Implementación de Controladores Lineales Multivariables.* Bógota, Colombia: Universidad Nacional de Colombia.

Negocios, T. y. (2020). *Tecnología y Negocios*. Obtenido de https://volkanrivera.com/esp/2008/01/9-caracteristicas-de-los-usuarios-open-source/

Nino Vega; Pablo Parra; Luis Córdova; Joselyne Andramuño; Johnny Álvarez. (2018). Cascade Control Algorithm developed with Embedded Systems. *ICA-ACCA 2018.* Concepción, Chile.

Santosh Kumar Verma and Shyam Krishna Nagar. (2016). *Approximation and Order Reduction of Fractional Order SISO System.* Bangalore, India: IEEE. doi:10.1109/INDICON.2016.7839060

ANEXO A

Matriz de consistencia

Tabla N° 1: MATRIZ DE CONSISTENCIA.

TÍTULO: DESARROLLO DE UN SISTEMA PORTATIL PARA LA IDENTIFICACIÓN Y EMULACIÓN DE SISTEMAS SISO EN PLANTAS INDUSTRIALES.					
PROBLEMAS	**OBJETIVOS**	**HIPÓTESIS**	**VARIABLES**	**INDICADORES**	**MÉTODO**
P. PRINCIPAL: ¿Es posible desarrollar un sistema portátil para la identificación y emulación de procesos industriales de tipo SISO hasta Orden 5, mediante el uso de herramienta de tipo open source (Software y Hardware) para plantas industriales?	O. GENERAL: Desarrollar un sistema portátil con herramientas de tipo open software y hardware, que permita la identificación y emulación de un sistema tipo SISO hasta de orden 5 para plantas industriales.	H. GENERAL: Si es posible desarrollar un sistema portátil para la identificación y emulación de procesos industriales de tipo SISO hasta Orden 5 sin la necesidad de interferir en el sistema.	V.DEPENDIENTE: Y: Sistema portátil con herramientas open source	Sistema Embebido.	Error cuadrático medio

Fuente: Elaboración propia.

ANEXO B

Matriz de consistencia

Tabla N° 2: MATRIZ DE CONSISTENCIA.

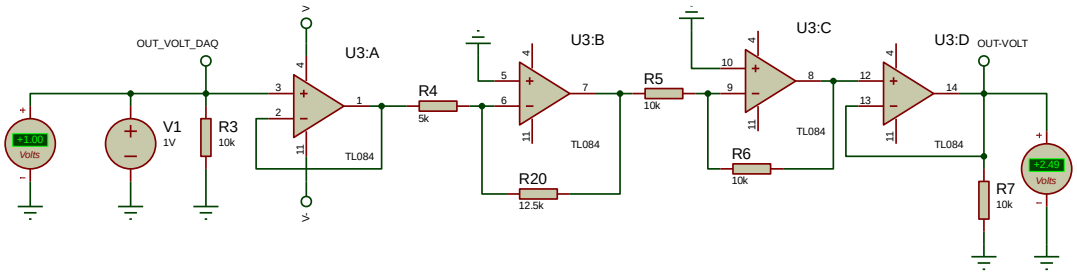
TÍTULO: DESARROLLO DE UN SISTEMA PORTATIL PARA LA IDENTIFICACIÓN Y EMULACIÓN DE SISTEMAS SISO EN PLANTAS INDUSTRIALES.					
PROBLEMAS	**OBJETIVOS**	**HIPÓTESIS**	**VARIABLES**	**INDICADORES**	**MÉTODO**
P. SECUNDARIO: P1: ¿Qué consideraciones se deberían tener el diseño de las interfaces electrónicas de entrada y salida para la identificación de un sistema de tipo SISO hasta orden 5 para lograr un sistema embebido?	O. ESPECÍFICO: O1:Desarrollar el algoritmo para la identificación de sistemas de tipo SISO hasta orden 5 haciendo uso de herramientas de tipo open source para lograr un sistema embebido.	H. ESPECIFICA: H1: Si es posible realizar la identificación de un sistema de tipo SISO hasta Orden 5 para el sistema embebido cumpliendo con los tiempos de muestro.	V.DEPENDIENTE: X1: Identificación de un sistema SISO hasta Orden 5.	Desarrollo del algoritmo de emulación de un sistema SISO hasta Orden 5.	Error cuadrático medio
P2. ¿Cuáles serían las condiciones tecnológicas de hardware y de software de tipo open source para el desarrollo de los algoritmos de emulación en un sistema embebido de tipo SISO hasta orden 5 para lograr un sistema embebido?	O2: Desarrollar el algoritmo para la emulación de sistemas de tipo SISO hasta 5 haciendo uso de herramientas de tipo open source para lograr un sistema embebido.	H2: Si es posible implementar el algoritmo de emulación de un sistema de tipo SISO hasta Orden 5 para el sistema embebido cumpliendo con los tiempos de muestro.	X2: Emulación de un sistema SISO hasta Orden 5.	Desarrollo del algoritmo de identificación de un sistema SISO hasta Orden 5.	Error cuadrático medio

Fuente: Elaboración propia.

ANEXO C

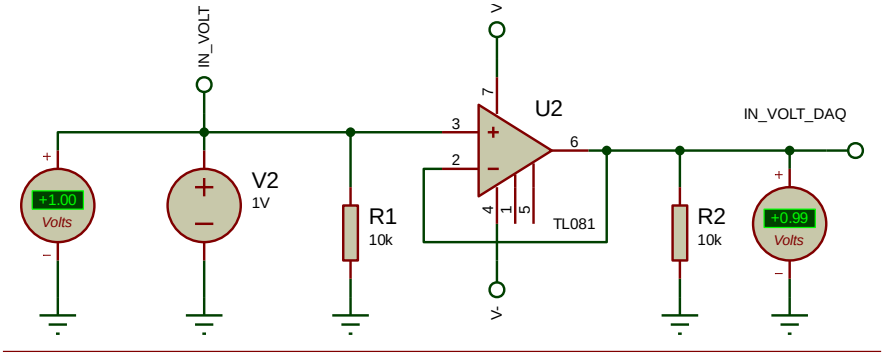
CIRCUITOS DE ACONDICIONAMIENTO

Simulación del circuito de Salida 0-4V a 0-10V



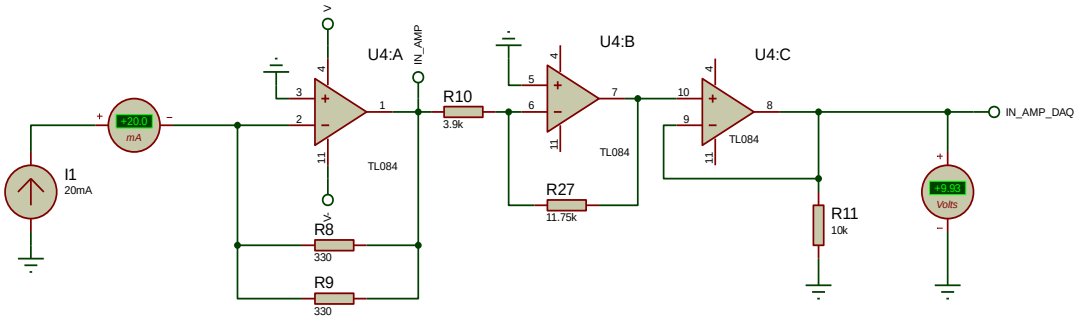
Fuente: Elaboración propia

Simulación del circuito de Entrada 0-10V



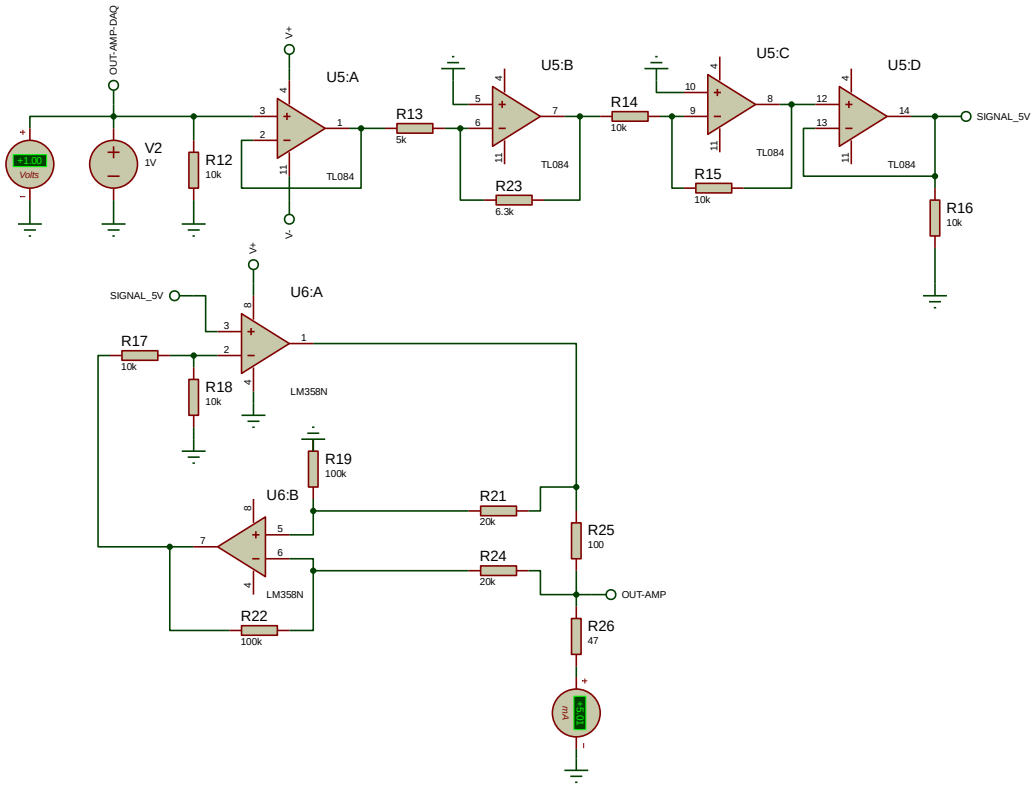
Fuente: Elaboración propia

Simulación del circuito de 4-20mA a 0-10V



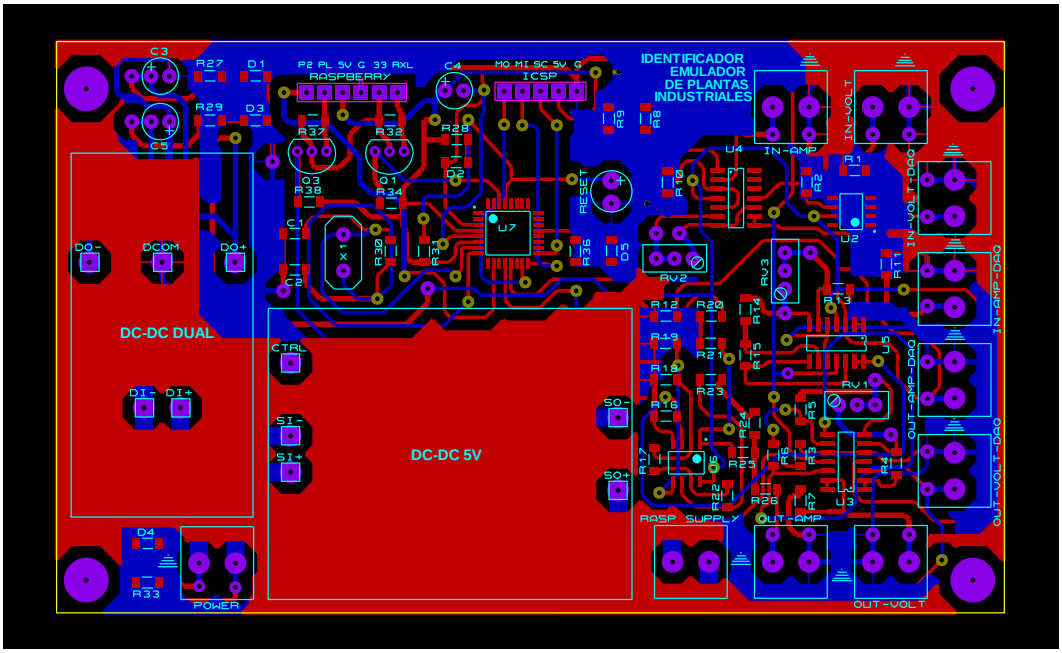
Fuente: Elaboración propia

Simulación del circuito de 0-4V a 4-20mA

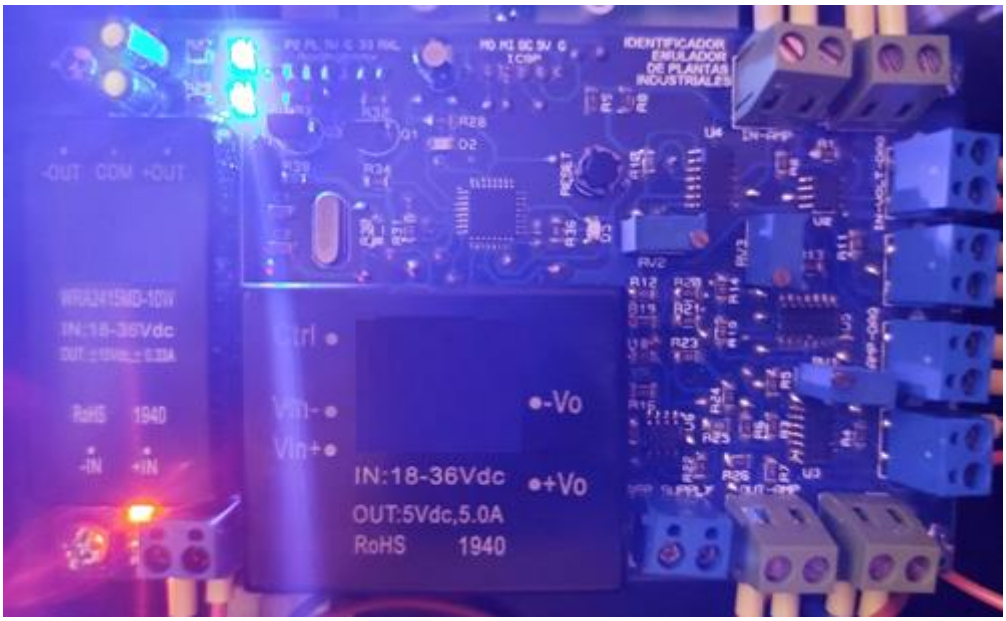


Fuente: Elaboración Propia.

Diseño PCB



PCB Implementado

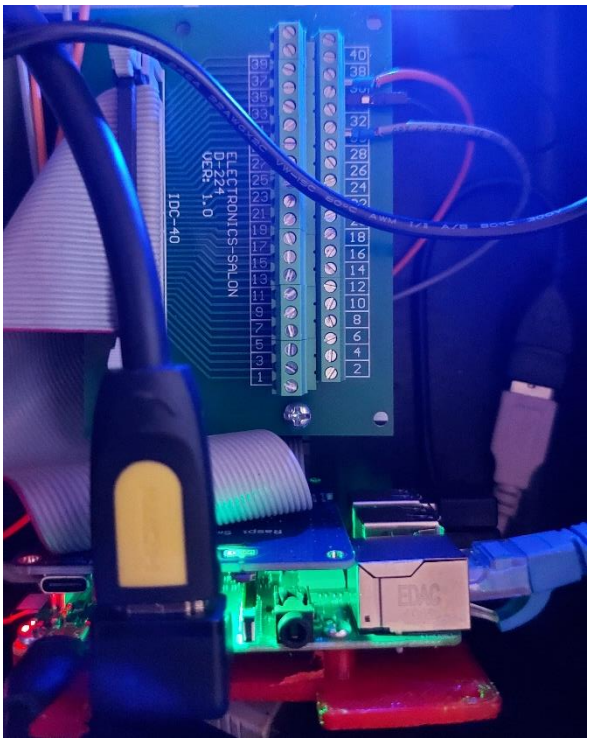


ANEXO D

INTEGRACIÓN

Tarjeta de Adquisición de datos

Raspberry PI 4

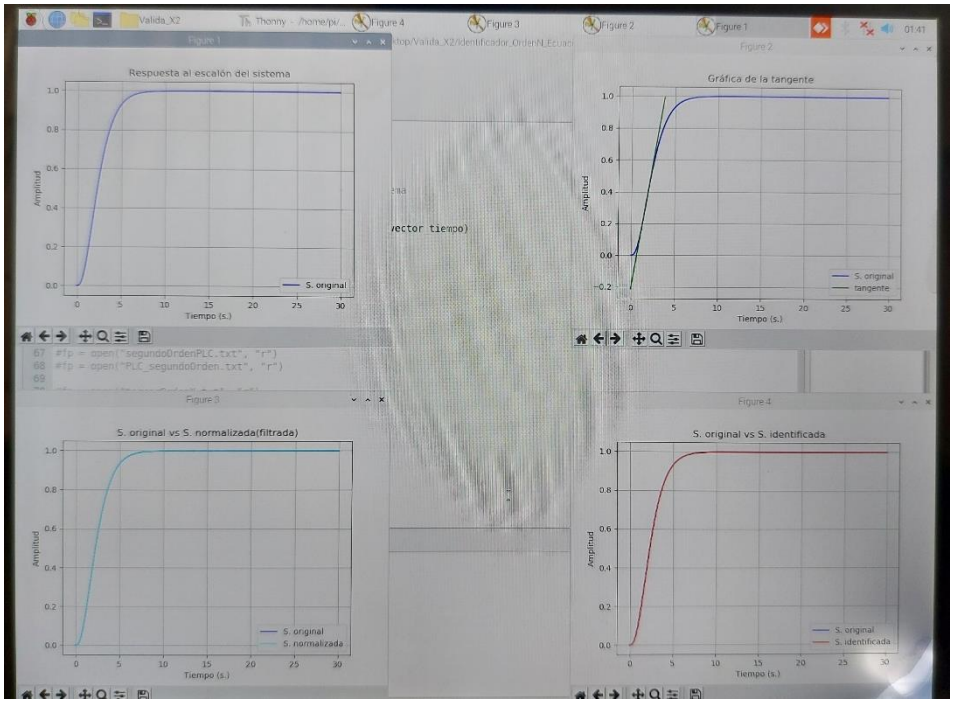


Sistema Implementado Integrado

Teclado

Pantalla

Pantalla de Interfaz



ANEXO E

#Identificador de sistemas Orden 1-5 Método Strejc

#Nikolai Vinces

```
import numpy as np                  # Librería para manejo de vectores, matrices y
                                    funciones.
import scipy.signal as ssg          # Librería para el manejo de FT
import matplotlib.pyplot as plt     # Librería para el manejo de Gráficas
import sympy as sym                 # Librería para el manejo de variables simbólicas
```

Modelamiento sistema matemático

```
s = sym.Symbol('s')                  # Variable de tipo símbolo
```

#Construcción de los vectores de numerador y denominador

```
polinomio_N = sym.Poly(ecn, s)
polinomio_D = sym.Poly(ecd, s)
coeff_N = polinomio_N.coeffs()
coeff_D = polinomio_D.coeffs()

N = [ ]
D = [ ]

for i in coeff_N:
   N.append(float(i))

for i in coeff_D:
   D.append(float(i))

#Señal escalón
vector_tiempo = np.arange(0,T,Ts)    # Vector tiempo de muestreo
escalon_u=[ ]

for i in vector_tiempo:
   if i<delay_sys:
      escalon_u.append(0)
   else:
      escalon_u.append(1)
```

#Función de transferencia y respuesta al escalón del sistema

```
tf_sistema = ssg.lti(N,D)
```

#Respuesta al escalón del sistema

```
[t_y,vector_muestras,yc] = ssg.lsim(tf_sistema,escalon_u,vector_tiempo)
```

```
#Reconstrucción vector tiempo adquirido
T = len(vector_muestras)*Ts
t_y = []
t_y = list(np.arange(0,T,Ts))

##### IDENTIFICACIÓN DEL SISTEMA #####

##Identificación del retardo
umbral_retardo =0.0001                  #Porcentaje del total de la amplitud de la señal
max_vector = max(vector_muestras)
min_vector = 0
rango = max_vector - min_vector
umbral_real = rango*(umbral_retardo/100)

index_umbral = 0
cont_index = 0

for i in vector_muestras:
    if i >= umbral_real:
        index_umbral = cont_index
        break
    cont_index = cont_index +1

retardo = (index_umbral-1)*Ts                   #Retardo en segundos
## Identificación de K

index_moda = 0
aux_moda = 0
aux_moda_anterior = 0
aux_umbral_k = round(0.01*len(vector_muestras))
array_k = range(len(vector_muestras)-aux_umbral_k,len(vector_muestras),1)

for i in array_k:
    for j in array_k:
        if vector_muestras[j] == vector_muestras[i]:
            aux_moda = aux_moda+1

    if aux_moda > aux_moda_anterior:
        index_moda = j

k_calc = vector_muestras[index_moda]/escalon_amp

#### Identificación de la tangente de la señal ####
t_y_tan=t_y[index_umbral:]
vector_normalizado = vector_muestras[index_umbral:]
```

```
#Derivadas de la señal
primer_diff = np.diff(vector_normalizado)/np.diff(t_y_tan)
segundo_diff = np.diff(primer_diff)/np.diff(t_y_tan[0:len(t_y_tan)-1])

#Inflexión
aux_inflex = np.where(segundo_diff < 0)
inflex = aux_inflex[0][0]
A = primer_diff[inflex]*t_y_tan[inflex] - vector_normalizado[inflex]

#Vector tangente
tanget = []
for i in t_y_tan:
    tanget.append(primer_diff[inflex]*i - A)
```

```
#Puntos de cruce de la tangente
tan_k = 0
tan_zero = 0
index_tan_k = 0
index_tan_zero = 0

for i in tanget:
    if i>=escalon_amp*k_calc:
        tan_k = i
        break
    index_tan_k = index_tan_k + 1

for i in tanget:
    if i>=0:
        tan_zero = i
        break
    index_tan_zero = index_tan_zero + 1

#Construir tangente para ploteo
seg_lim_k_tan = index_tan_k*Ts
x_tan = list(np.arange(retardo,seg_lim_k_tan + retardo,Ts))
y_tan = tanget[0:index_tan_k]

#Corrección data:
if(len(x_tan)>len(y_tan)):
    y_tan.insert(0,y_tan[0])

### Identificación del sistema por método de Strejc ###
Ta = (index_tan_k - index_tan_zero)/(1/Ts)
Tu = index_tan_zero/(1/Ts)
tu_ta = Tu/Ta

# Hallar el orden del sistema
strejc = [0,0.104,0.218,0.319,0.410,0.493]
res_strejc = []
for i in strejc:
    aux_n = abs(tu_ta-i)
    res_strejc.append(aux_n)

index_strejc_n = res_strejc.index(min(res_strejc))
n_calc = index_strejc_n + 1     #Orden del sistema
```

```
## Hallar el valor de tau
strejc_tau1 = [1,2.718,3.695,4.463,5.119,5.699]
strejc_tau2 = [0,0.282,0.805,1.425,2.1,2.811]
if index_strejc_n != 0:
   tau_aux1 = Ta/strejc_tau1[index_strejc_n]
   tau_aux2 = Tu/strejc_tau2[index_strejc_n]
else:
   tau_aux1 = tau_aux2 = Ta/1

tau_calc = round((tau_aux1+tau_aux2)/2,3)   #Resultado del tau

### Reconstrucción del sistema con los valores hallados ###
#Construcción de los vectores de numerador y denominador
ecn_iden = [k_calc]
ecd_iden = (1+tau_calc*s)**n_calc

polinomio_N_iden = sym.Poly(ecn_iden, s)
polinomio_D_iden = sym.Poly(ecd_iden, s)
coeff_N_iden = polinomio_N_iden.coeffs()
coeff_D_iden = polinomio_D_iden.coeffs()

N_iden = []
D_iden = []

for i in coeff_N_iden:
   N_iden.append(float(i))

for i in coeff_D_iden:
   D_iden.append(float(i))

#Señal escalón
vector_tiempo_iden = np.arange(0,T,Ts)
escalon_u_iden=[]

for i in vector_tiempo_iden:
   if i<retardo:
      escalon_u_iden.append(0)
   else:
      escalon_u_iden.append(1)

#Función de transferencia y respuesta al escalón del sistema
tf_sistema_iden = ssg.lti(N_iden,D_iden)
```

```
### Mostrar resultados ###
print('El valor de k es: ' + str(k_calc))
print('El valor de retardo es: ' + str(retardo))
print('El valor del tau: ' + str(tau_calc))
print('El orden n es: ' + str(n_calc))
print('El valor de tau1 (Strejc) es: ' + str(tau_aux1))
print('El valor de tau2 (Strejc) es: ' + str(tau_aux2))

### PLOTEO ###

## Gráfica de la señal original
plt.figure(1)
plt.plot(t_y,vector_muestras,color='blue',label='S. original')
plt.title('Respuesta al escalón del sistema')
plt.ylabel('Amplitud')
plt.xlabel('Tiempo (s.)')
plt.legend()
plt.grid()

## Gráfica de la tangente
plt.figure(2)
plt.plot(t_y,vector_muestras,color='blue',label='S. original')
plt.plot(x_tan,y_tan,color='green',label='tangente')
plt.title('Gráfica de la tangente')
plt.ylabel('Amplitud')
plt.xlabel('Tiempo (s.)')
plt.legend()
plt.grid()

## Gráfica de la señal normalizada(filtrada)
plt.figure(3)
plt.plot(t_y,vector_muestras,color='blue',label='S. original')
plt.plot(t_y_tan,vector_normalizado,color='cyan',label='S. normalizada')
plt.title('S. original vs S. normalizada(filtrada)')
plt.ylabel('Amplitud')
plt.xlabel('Tiempo (s.)')
plt.legend()
plt.grid()

##Gráfica de la señal identificada
plt.figure(4)
plt.plot(t_y,vector_muestras,color='blue',label='S. original')
plt.plot(t_y_iden,vector_muestras_iden,color='red',label='S. identificada')
plt.title('S. original vs S. identificada')
plt.ylabel('Amplitud')
```

```
plt.xlabel('Tiempo (s.)')
plt.legend()
plt.grid()
plt.show()
```

ANEXO F

INTEGRACIÓN

#Emulador de plantas industriales
#Aproximación por Tustin

#Nikolai Vinces

```
import scipy.signal as ssg
import matplotlib.pyplot as plt
import numpy as np
from math import *

#Definición de variables globales
global x,y,n
x = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]          #x->ENTRADA
y = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]          #y->ENTRADA
n = 8                                          #Número de valores simulados guardados

def main():
    orden = orden_sistema()
    Ts = float(input("Ingresar el período de muestreo en segundos: "))
    T = int(input("Ingresar el tiempo de simulación total: "))
    #delay = float(input("Ingresar el delay del sistema en segundos: "))
    #ampl_signal_u = float(input("Ingresar la amplitud de la señal escalón: "))
    ampl_signal_u=1

    file = open('segundoOrden_validaX3_python.txt', 'w')

    if (orden==1):
        [[a1,a0],[b1,b0]] = coef_1erOrden()
        [k1,k2,k3] = coef_Tustin_1erOrden(a1,a0,b1,b0,Ts)
    elif (orden==2):
        [[a2,a1,a0],[b2,b1,b0]] = coef_2doOrden()
        [k1,k2,k3,k4,k5] = coef_Tustin_2doOrden(a2,a1,a0,b2,b1,b0,Ts)
    elif (orden==3):
        [[a3,a2,a1,a0],[b3,b2,b1,b0]] = coef_3erOrden()
        [k1,k2,k3,k4,k5,k6,k7] = coef_Tustin_3erOrden(a3,a2,a1,a0,b3,b2,b1,b0,Ts)
    elif (orden==4):
        [[a4,a3,a2,a1,a0],[b4,b3,b2,b1,b0]] = coef_4toOrden()
        [k1,k2,k3,k4,k5,k6,k7,k8,k9] =
coef_Tustin_4toOrden(a4,a3,a2,a1,a0,b4,b3,b2,b1,b0,Ts)
    elif (orden==5):
        [[a5,a4,a3,a2,a1,a0],[b5,b4,b3,b2,b1,b0]] = coef_5toOrden()
```

```
        [k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11] =
coef_Tustin_5toOrden(a5,a4,a3,a2,a1,a0,b5,b4,b3,b2,b1,b0,Ts)
    else:
        print("Orden no identificado")

    #Proceso de emulación
    contIndex = 0
    sis_emul=[]
    i = 0   #i->Contador
    j = 0   #j->Contador

    for k in np.arange(0,T,Ts):
        x[i] = 1.0
        x[0] = 0.0

        if(orden==1):
            if (i==0):
                y[i] = k1*x[i]
            else:
                y[i] = k1*x[i] + k2*x[i-1] - k3*y[i-1]

        if(orden==2):
            if (i==0):
                y[i] = k1*x[i]
            elif (i==1):
                y[i] = k1*x[i] + k2*x[i-1] - k4*y[i-1]
            else:
                y[i] = k1*x[i] + k2*x[i-1] + k3*x[i-2] - k4*y[i-1] - k5*y[i-2]

        if(orden==3):
            if(i==0):
                y[i]=k1*x[i]
            elif(i==1):
                y[i]=k1*x[i]+k2*x[i-1]-k5*y[i-1]
            elif(i==2):
                y[i]=k1*x[i]+k2*x[i-1]+k3*x[i-2]-k5*y[i-1]-k6*y[i-2]
            else:
                y[i]=k1*x[i]+k2*x[i-1]+k3*x[i-2]+k4*x[i-3]-k5*y[i-1]-k6*y[i-2]-k7*y[i-3]

        if(orden==4):
            if(i==0):
                y[i]=k1*x[i]
            elif(i==1):
                y[i]=k1*x[i]+k2*x[i-1]-k6*y[i-1]
            elif(i==2):
```

```
            y[i]=k1*x[i]+k2*x[i-1]+k3*x[i-2]-k6*y[i-1]-k7*y[i-2]
        elif(i==3):
            y[i]=k1*x[i]+k2*x[i-1]+k3*x[i-2]+k4*x[i-3]-k6*y[i-1]-k7*y[i-2]-k8*y[i-3]
        else:
            y[i]=k1*x[i]+k2*x[i-1]+k3*x[i-2]+k4*x[i-3]+k5*x[i-4]-k6*y[i-1]-k7*y[i-2]-
k8*y[i-3]-k9*y[i-4]

    if(orden==5):
        if(i==0):
            y[i]=k1*x[i]
        elif(i==1):
            y[i]=k1*x[i]+k2*x[i-1]-k7*y[i-1]
        elif(i==2):
            y[i]=k1*x[i]+k2*x[i-1]+k3*x[i-2]-k7*y[i-1]-k8*y[i-2]
        elif(i==3):
            y[i]=k1*x[i]+k2*x[i-1]+k3*x[i-2]+k4*x[i-3]-k7*y[i-1]-k8*y[i-2]-k9*y[i-3]
        elif(i==4):
            y[i]=k1*x[i]+k2*x[i-1]+k3*x[i-2]+k4*x[i-3]+k5*x[i-4]-k7*y[i-1]-k8*y[i-2]-
k9*y[i-3]-k10*y[i-4]
        else:
            y[i]=k1*x[i]+k2*x[i-1]+k3*x[i-2]+k4*x[i-3]+k5*x[i-4]+k6*x[i-5]-k7*y[i-1]-
k8*y[i-2]-k9*y[i-3]-k10*y[i-4]-k11*y[i-5]

    if (i<(n-1)):

        sis_emul.append(y[i])
        if(y[i]==0):
            file.write(str(0.0)+'\n')
        else:
            file.write(str(y[i])+'\n')
        i = i+1

    else:
        sis_emul.append(y[i])
        if(y[i]==0):
            file.write(str(0.0)+'\n')
        else:
            file.write(str(y[i])+'\n')
        for j in range(0,(n-1)):
            y[j] = y[j+1]
            x[j] = x[j+1]
```

```
    ## Gráfica de las señales
    plt.figure(1)
    plt.plot(tiempo_emulado,sis_emul,color='blue',label='Respuesta al escalón')
    plt.title('Sistema emulado')
    plt.ylabel('Amplitud')
    plt.xlabel('Tiempo (seg.)')
    plt.legend()
    plt.grid()
    plt.show()

def orden_sistema():
    orden = int(input("Ingresar el orden del sistema: "))

    while (orden<1) or (orden>5):
        print("Ingresar un orden válido en el rango [1..5]")
        orden = int(input("Ingresar el orden del sistema: "))
    return orden

def coef_1erOrden():
    print("1er orden - Ingresar coeficientes del numerador")
    a1 = float(input("a1 = "))
    a0 = float(input("a0 = "))
    print("1er orden - Ingresar coeficientes del denominador")
    b1 = float(input("b1 = "))
    b0 = float(input("b0 = "))
    return [[a1,a0],[b1,b0]]

def coef_2doOrden():
    print("2do orden - Ingresar coeficientes del numerador")
    a2 = float(input("a2 = "))
    a1 = float(input("a1 = "))
    a0 = float(input("a0 = "))
    print("2do orden - Ingresar coeficientes del denominador")
    b2 = float(input("b2 = "))
    b1 = float(input("b1 = "))
    b0 = float(input("b0 = "))
    return [[a2,a1,a0],[b2,b1,b0]]

def coef_3erOrden():
    print("3er orden - Ingresar coeficientes del numerador")
    a3 = float(input("a3 = "))
    a2 = float(input("a2 = "))
    a1 = float(input("a1 = "))
    a0 = float(input("a0 = "))
    print("3er orden - Ingresar coeficientes del denominador")
```

```
    b3 = float(input("b3 = "))
    b2 = float(input("b2 = "))
    b1 = float(input("b1 = "))
    b0 = float(input("b0 = "))
    return [[a3,a2,a1,a0],[b3,b2,b1,b0]]

def coef_4toOrden():
    print("4to orden - Ingresar coeficientes del numerador")
    a4 = float(input("a4 = "))
    a3 = float(input("a3 = "))
    a2 = float(input("a2 = "))
    a1 = float(input("a1 = "))
    a0 = float(input("a0 = "))
    print("4to orden - Ingresar coeficientes del denominador")
    b4 = float(input("b4 = "))
    b3 = float(input("b3 = "))
    b2 = float(input("b2 = "))
    b1 = float(input("b1 = "))
    b0 = float(input("b0 = "))
    return [[a4,a3,a2,a1,a0],[b4,b3,b2,b1,b0]]

def coef_5toOrden():
    print("5to orden - Ingresar coeficientes del numerador")
    a5 = float(input("a5 = "))
    a4 = float(input("a4 = "))
    a3 = float(input("a3 = "))
    a2 = float(input("a2 = "))
    a1 = float(input("a1 = "))
    a0 = float(input("a0 = "))
    print("5to orden - Ingresar coeficientes del denominador")
    b5 = float(input("b5 = "))
    b4 = float(input("b4 = "))
    b3 = float(input("b3 = "))
    b2 = float(input("b2 = "))
    b1 = float(input("b1 = "))
    b0 = float(input("b0 = "))
    return [[a5,a4,a3,a2,a1,a0],[b5,b4,b3,b2,b1,b0]]
```

```
def coef_Tustin_1erOrden(a1,a0,b1,b0,Ts):
    [[az1,az0]],[bz1,bz0],dt = ssg.cont2discrete(([a1,a0],[b1,b0]),Ts)
    k1 = az1/bz1
    k2 = az0/bz1
    k3 = bz0/bz1
    return [k1,k2,k3]

def coef_Tustin_2doOrden(a2,a1,a0,b2,b1,b0,Ts):
    [[az2,az1,az0]],[bz2,bz1,bz0],dt = ssg.cont2discrete(([a2,a1,a0],[b2,b1,b0]),Ts)
    k1 = az2/bz2
    k2 = az1/bz2
    k3 = az0/bz2
    k4 = bz1/bz2
    k5 = bz0/bz2
    return [k1,k2,k3,k4,k5]

def coef_Tustin_3erOrden(a3,a2,a1,a0,b3,b2,b1,b0,Ts):
    [[az3,az2,az1,az0]],[bz3,bz2,bz1,bz0],dt =
ssg.cont2discrete(([a3,a2,a1,a0],[b3,b2,b1,b0]),Ts)
    k1=az3/bz3
    k2=az2/bz3
    k3=az1/bz3
    k4=az0/bz3
    k5=bz2/bz3
    k6=bz1/bz3
    k7=bz0/bz3
    return [k1,k2,k3,k4,k5,k6,k7]

def coef_Tustin_4toOrden(a4,a3,a2,a1,a0,b4,b3,b2,b1,b0,Ts):
    [[az4,az3,az2,az1,az0]],[bz4,bz3,bz2,bz1,bz0],dt =
ssg.cont2discrete(([a4,a3,a2,a1,a0],[b4,b3,b2,b1,b0]),Ts)
    k1=az4/bz4
    k2=az3/bz4
    k3=az2/bz4
    k4=az1/bz4
    k5=az0/bz4
    k6=bz3/bz4
    k7=bz2/bz4
    k8=bz1/bz4
    k9=bz0/bz4
    return [k1,k2,k3,k4,k5,k6,k7,k8,k9]

def coef_Tustin_5toOrden(a5,a4,a3,a2,a1,a0,b5,b4,b3,b2,b1,b0,Ts):
    [[az5,az4,az3,az2,az1,az0]],[bz5,bz4,bz3,bz2,bz1,bz0],dt =
ssg.cont2discrete(([a5,a4,a3,a2,a1,a0],[b5,b4,b3,b2,b1,b0]),Ts)
```

```
    k1=az5/bz5
    k2=az4/bz5
    k3=az3/bz5
    k4=az2/bz5
    k5=az1/bz5
    k6=az0/bz5
    k7=bz4/bz5
    k8=bz3/bz5
    k9=bz2/bz5
    k10=bz1/bz5
    k11=bz0/bz5
    return [k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11]

if __name__ =='__main__':
    main()
```