



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
IEE2683 – LABORATORIO DE CONTROL AUTOMÁTICO

Experiencia 1: Control de Procesos

2 Sesiones

1. Introducción

La automatización de procesos se ha masificado en los últimos años en áreas tan diversas como la minería, energía, y sistemas aeronáuticos o robóticos, debido a sus beneficios económicos y técnicos. Ejemplos de procesos automatizados incluyen el controlar la posición o velocidad de desplazamiento de mecanismos como servoválvulas, que poseen motores de corriente continua que proveen la fuerza para abrir o cerrar los pistones; o motobombas, cuya velocidad es regulada para ajustar el flujo de líquidos. El componente fundamental de un proceso automatizado es el sistema SCADA (supervisory control and data acquisition), el cual se encarga de adquirir, procesar y desplegar las mediciones de los diversos sensores, y aplicar las acciones de control a los actuadores de campo. En su forma mas simple, un sistema SCADA consta de un dispositivo de adquisición/envío de datos, una interfaz hombre-máquina, y un algoritmo de control digital.

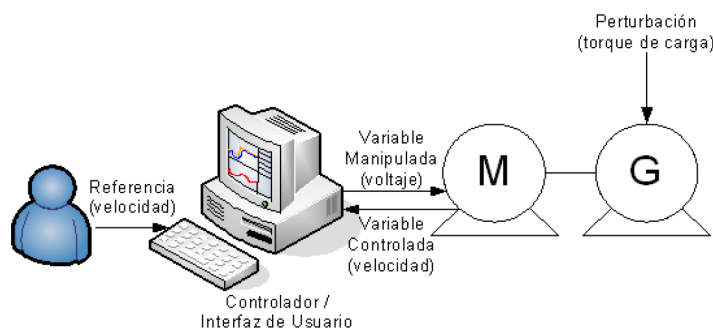


Figura 1: Sistema de control grupo motor-generador

La apertura de las tecnologías de automatización ha traído consigo una diversificación de proveedores, lo que conlleva un problema de conectividad entre equipos de planta, como los sensores y actuadores, con los controladores debido a la coexistencia de diversos protocolos de comunicación, muchas veces propietarios. Para solucionar este problema, en 1996 surge el protocolo de comunicación OPC (*Ole for Process Control*) basado en una arquitectura cliente-servidor, que busca estandarizar las comunicaciones a nivel de redes de control potenciando la interoperabilidad entre equipos de diversos fabricantes. Tras su surgimiento, OPC se ha establecido como un estándar de comunicación en el campo de control y supervisión de procesos industriales.

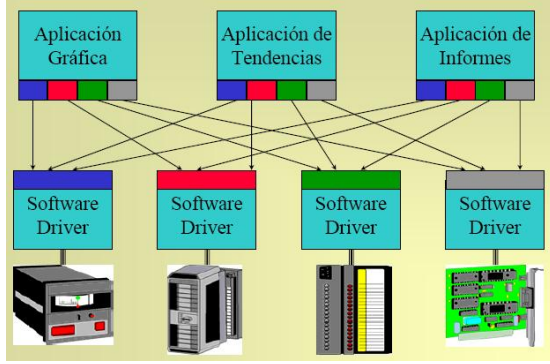


Figura 2: Problema de no contar con OPC

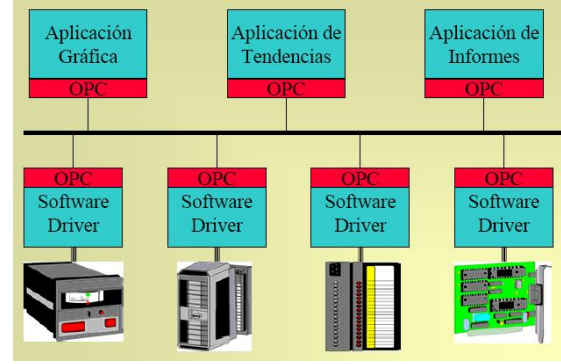


Figura 3: Ventajas de contar con OPC

La presente experiencia consiste en desarrollar un sistema SCADA simple, capaz de adquirir mediciones mediante el protocolo OPC, desplegar mediciones en una interfaz de usuario, e implementar acciones de control basadas en un controlador del tipo PID digital. El objetivo final es implementar un lazo de control simple, como el que se muestra en la Figura 1, donde el operador manipule el set-point del proceso y el sistema SCADA lleve la planta a la referencia deseada.

2. Marco Teórico

2.1. Proceso a controlar

El proceso que se debe controlar mediante el sistema SCADA corresponde al clásico sistema de los *cuatro tanques* que se muestra en la figura 4. Para efectos de la experiencia, se utilizará un simulador del sistema implementado en un servidor remoto, cuyas entradas y salidas se leen y escriben mediante el protocolo OPC.

El sistema de los cuatro tanques se ha popularizado a lo largo de los años debido a su complejidad y flexibilidad, siendo un sistema MIMO acoplado que puede variar entre un comportamiento de fase mínima y fase no mínima solo cambiando la razón del flujo que va a los tanques superiores e inferiores.

El objetivo de control es llevar los tanques 1 y 2 (tanques inferiores) a una referencia, manipulando las dos válvulas existentes. De la figura 4 se aprecia que los tanques 1 y 2 interactúan entre sí indirectamente mediante los tanques 3 y 4 (tanques superiores), generando una perturbación que hace que el sistema sea complejo de controlar.

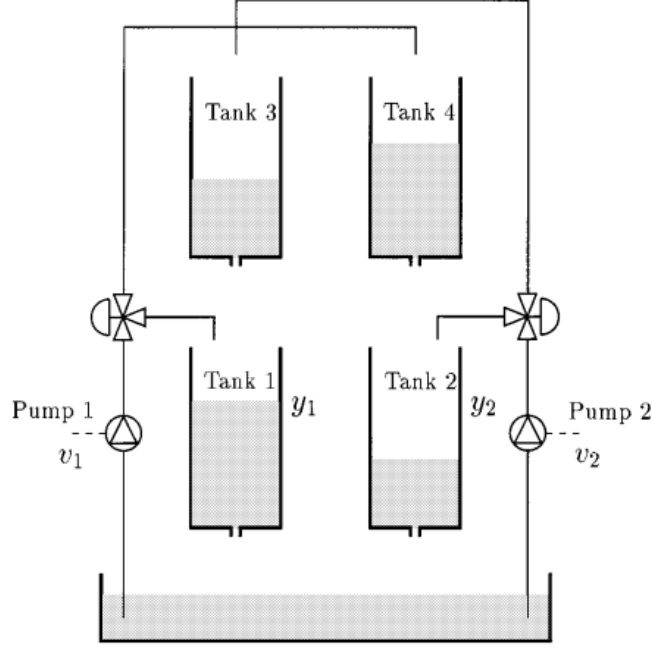


Figura 4: Sistema de los cuatro tanques

A partir de la Figura 4, un modelo en tiempo continuo del sistema viene dado por:

$$\begin{aligned}
 \frac{dh_1}{dt} &= -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1 v_1}{A_1} \\
 \frac{dh_2}{dt} &= -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2 v_2}{A_2} \\
 \frac{dh_3}{dt} &= -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2 v_2}{A_3} \\
 \frac{dh_4}{dt} &= -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1 v_1}{A_4},
 \end{aligned} \tag{1}$$

donde:

- A_i : Sección transversal del tanque
- a_i : Sección transversal del agujero
- h_i : Altura del agua del tanque i
- v_j : Voltaje aplicado a la válvula j , $j \in \{1, 2\}$
- k_j : Factor de transformación de voltaje a flujo en la válvula j
- γ_j : Razón del flujo de la válvula que llega a los tanques inferiores respecto a los superiores.

Al linealizar el sistema en torno a un cierto punto de operación x_0 y u_0 se tiene:

$$\frac{dx}{dt} = \begin{bmatrix} \frac{-1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & \frac{-1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & \frac{-1}{T_3} & 0 \\ 0 & 0 & 0 & \frac{-1}{T_4} \end{bmatrix} x + \begin{bmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2)k_2}{A_3} \\ \frac{(1-\gamma_1)k_1}{A_4} & 0 \end{bmatrix} u \quad (2)$$

$$y = \begin{bmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{bmatrix} x, \quad (3)$$

donde:

$$T_i = \frac{A_i}{a_i} \sqrt{\frac{2h_{i0}}{g}} \quad (4)$$

Luego, la función de transferencia entrada salida estará dada por:

$$G(s) = \begin{bmatrix} \frac{\gamma_1 c_1}{1+sT_1} & \frac{(1-\gamma_2)c_1}{(1+sT_3)(1+sT_1)} \\ \frac{(1-\gamma_1)c_2}{(1+sT_4)(1+sT_2)} & \frac{\gamma_2 c_2}{1+sT_2} \end{bmatrix} \quad (5)$$

donde $c_1 = T_1 k_1 \frac{k_c}{A_1}$ y $c_2 = T_2 k_2 \frac{k_c}{A_2}$.

Analizando los ceros de la función de transferencia se puede concluir que:

- $0 < \gamma_1 + \gamma_2 < 1$: Hay un cero en el semiplano derecho y el sistema es de fase no mínima.
- $1 < \gamma_1 + \gamma_2 < 2$: Ambos ceros están en el semiplano izquierdo y el sistema es de fase mínima.

2.2. Simulador

Para el desarrollo de la experiencia se utilizará un simulador programado en Python que cuenta con los siguientes parámetros:

A_1, A_3	28 [cm ²]
A_2, A_4	32 [cm ²]
a_1, a_3	0.071 [cm ²]
a_2, a_4	0.057 [cm ²]
k_1, k_2	3.33 [cm ³ /V]
g	981 [cm/s ²]
γ_1	0.7
γ_2	0.6
v_{1max}, v_{2max}	5 [Volt]
h_{max}	50 [cm]
$h_i(0) \quad \forall i$	40 [cm]

El simulador posee una interfaz gráfica que muestra los niveles y valores de las válvulas en tiempo real. Usted no debe modificar el código que implementa el sistema ni el código de la interfaz. Cabe mencionar que en el simulador las válvulas saturan cuando el voltaje es 1 o -1, lo que debe ser considerado en el diseño de su controlador.

3. OPC: Ole for Process Control

OPC surge en 1996 con el objetivo de proveer de un estándar de comunicación para el campo de control y supervisión de procesos industriales. En su primera versión, OPC estaba restringido solo para sistemas Windows y se necesitaba un servidor distinto para usar cada una de las funcionalidades del estándar. Algunas de las funcionalidades mas utilizadas en ambientes de control industrial son:

- **OPC-DA:** Para adquisición de datos en tiempo real
- **OPC-HDA:** Para trabajar con datos históricos
- **OPC- AE:** Para trabajar con alarmas y eventos

Con el fin de solucionar problemas de interoperabilidad e impulsado por la tendencia de integrar funcionalidades, el año 2008 surge **OPC UA** (Unified Architecture) que se abrió para cualquier sistema operativo, permite un modelamiento de la información a modo de clases y además unifica todas las funcionalidades anteriores en un solo servidor.

Algunos conceptos claves de OPC UA son los siguientes:

- **Namespace:** Es un espacio o *contenedor* de objetos en que se pueden compartimentalizar las distintas aplicaciones OPC en un mismo servidor. Un servidor puede tener múltiples Namespaces con su propio identificador y cada Namespace puede contener información de uno o múltiples procesos.
- **Objetos:** Un objeto es simplemente un ente que posee ciertas variables internas que se pueden leer o escribir, métodos que le dan funcionalidades definidas por el usuario y también referencias a otros objetos. Es un concepto equivalente a una clase en programación orientada a objetos.
- **Nodos:** En OPC UA todo se representa por nodos, esto incluye a los objetos, variables, métodos y referencias entre otros. Así, un objeto puede ser visto como un nodo del tipo objeto que contiene subnodos del tipo variable, método y referencias. Cabe mencionar que cada nodo tiene un identificador único denominado *NodeId*.
- **NodeId:** Este es el Id de cada nodo y permite identificarlo de manera única en todo el servidor, este Id está compuesto de tres elementos:
 - NamespaceIndex: Valor numérico que identifica cada Namespace del servidor.
 - IdentifierType: El tipo de identificador utilizado, puede ser string, bytes o números. Por defecto, OPC UA utiliza el tipo numérico.
 - Identifier: El identificador de cada nodo del tipo especificado.

Un nodo puede ser accesado por su NodeId o por su path, tal como ocurre en un sistema de archivos.

- **Suscripción a un nodo:** Para no estar leyendo constantemente los valores de las variables, OPC provee de una funcionalidad mucho más elegante llamada suscripción. Un cliente puede suscribirse a los nodos de su interés y dejar que el servidor los monitoree por él. Hay tres tipos distintos de suscripción:

- Suscripción a cambios en los valores de variables.
- Suscripción a eventos y alarmas
- Suscripción a valores agregados, que son valores que se calculan en tiempos definidos por el cliente a partir de los valores originales de las variables.

NodeId	
NamespaceIndex	2
IdentifierType	numeric
Identifier	5001

Figura 5: Formato del Identificador de un nodo

3.1. Free OPC UA

Para el desarrollo de la experiencia se utilizará la librería de código abierto FreeOPCUA, escrita en Python. Su tarea será implementar un cliente OPC UA que se conecte al servidor ya implementado y que cumpla las siguientes cuatro funciones:

1. Leer información de los niveles (variables controladas).
2. Escribir posiciones en las válvulas (variables manipuladas).
3. Escribir valores en las razones de flujo (variables γ).
4. Suscribirse a las alarmas de nivel que ya están en el servidor (se activan cuando alguno de los niveles sea menor a 10cm).

La librería se puede encontrar en: <https://github.com/FreeOpcUa/python-opcua>. Para instalar este paquete, la forma más rápida es utilizando pip, para esto debe abrir una consola de comandos y escribir *pip install opcua*.

Utilizar esta librería es bastante simple, para crear un cliente que se suscriba a un servidor en localhost por el puerto 4048 y luego acceder al nodo padre donde se encuentran todos los objetos, se puede usar el siguiente código:

```

1 from opcua import Client
2 from opcua import ua
3
4 client = Client("opc.tcp://localhost:4840/freeopcua/server/")
5 try:
6     client.connect()
7     objectsNode = client.get_objects_node()
8 except:
9     client.disconnect()

```

3.2. Estructura de datos del proceso

Como ya se mencionó anteriormente, su deber es programar el cliente OPC UA a partir del servidor ya implementado, para esto, usted debe conocer la estructura de datos presente en el servidor.

Esta estructura consiste simplemente en cuatro carpetas distintas, una carpeta que contiene todos los objetos del proceso, otra que contiene los objetos Tanques, en donde cada uno de ellos contiene la variable “h” (altura). La tercera carpeta tiene los dos objetos Válvulas y cada una de ellas contiene la variable “u” que se mide en Volts. Finalmente, la cuarta carpeta lleva el nombre de Razones y contiene dos objetos Razon los cuales contienen la variable “gamma”.

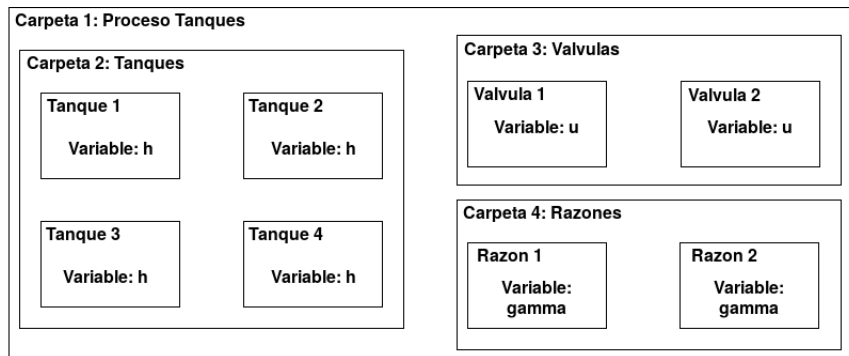


Figura 6: Estructura de datos

Por ejemplo, asumiendo que el ID del namespace es 1, para poder acceder a la variable h del tanque 1 utilizando el path, se puede usar el siguiente código:

```
1 from opcua import Client
2 from opcua import ua
3
4 client = Client("opc.tcp://localhost:4840/freeopcua/server/")
5 try:
6     client.connect()
7     objectsNode = client.get_objects_node()
8     h1 = objectsNode.get_child(['1:Proceso Tanques',
9     '1:Tanques', '1:Tanque1', '1:h'])
10    valor = h1.get_value()
11 except:
12    client.disconnect()
```

4. Interfaz gráfica (HMI)

Si bien hay múltiples librerías en python para crear interfaces gráficas como Matplotlib, Plotly y Seaborn, que pueden servir para el desarrollo de la experiencia, la mayoría de ellas no están hechas para streaming de datos, por lo que implementar esta funcionalidad se le podría complicar más de la cuenta. Por esto se recomienda fuertemente utilizar librerías que estén creadas para este propósito como Dash o Bokeh, ambas son librerías de alto nivel y permiten graficar datos a manera de streaming de forma sencilla. Adicionalmente, estas librerías tienen una serie de *Widgets* que

permiten interactuar con el programa a través de la interfaz.

Por ejemplo, utilizando Bokeh un pequeño Dashboard que muestre en el mismo gráfico los valores de un seno y un coseno simulando un stream de datos cada 0.3 segundos y un *Widget* para mostrar los valores numéricos, se puede implementar utilizando el siguiente código:

```
1 import numpy as np
2 from bokeh.models import ColumnDataSource, PreText
3 from bokeh.layouts import layout
4 from bokeh.plotting import curdoc, figure
5
6
7 # Se crea un seno
8 T = np.linspace(0,1000,1001)
9 sin = np.sin(T)
10 cos = np.cos(T)
11
12 # Se crea el DataSource
13 DataSource = ColumnDataSource(dict(time=[], sin=[], cos=[]))
14
15 # Figura
16 fig_sin = figure(title='Seno y Coseno', plot_width=600, plot_height=200, tools="
17     reset,xpan,xwheel_zoom,xbox_zoom", y_axis_location="left")
18 fig_sin.line(x='time', y='sin', alpha=0.8, line_width=3, color='blue', source=
19     DataSource, legend='Seno')
20 fig_sin.line(x='time', y='cos', alpha=0.8, line_width=3, color='red', source=
21     DataSource, legend='Coseno')
22 fig_sin.xaxis.axis_label = 'Tiempo (S)'
23 fig_sin.yaxis.axis_label = 'Valores'
24
25 # Se crea un par de Widgets
26 estilo = {'color': 'white', 'font': '15px bold arial, sans-serif', 'background-color':
27     'green', 'text-align': 'center', 'border-radius': '7px'}
28 SinText = PreText(text='Valor del Seno: 0.00', width=300, style=estilo)
29 CosText = PreText(text='Valor del Coseno: 0.00', width=300, style=estilo)
30
31 t = 0
32 def MainLoop(): # Funcion principal que se llama cada cierto tiempo para mostrar la
33     informacion
34     global t, sin, cos
35     update = dict(time=[t], sin=[sin[t]], cos=[cos[t]])
36     DataSource.stream(new_data=update, rollover=100) # Se ven los ultimos 100 datos
37     SinText.text = 'Valor del Seno: {}'.format(round(sin[t],2))
38     CosText.text = 'Valor del Coseno: {}'.format(round(cos[t],2))
39     t += 1
40
41 l = layout([
42     [fig_sin],
43     [SinText, CosText]
44 ])
45
46 curdoc().add_root(l)
47 curdoc().title = "Dashboard Ejemplo"
```



```
44 curdoc().add_periodic_callback(MainLoop, 300) # Cada 300 milisegundos se llama a la
    funcion y se actualiza el grafico
```

Suponiendo que el Script lleva por nombre "InterfazEjemplo.py", para ejecutar el código, en su consola de comandos debe escribir *bokeh serve --show InterfazEjemplo.py*. El resultado se muestra en la figura 7.

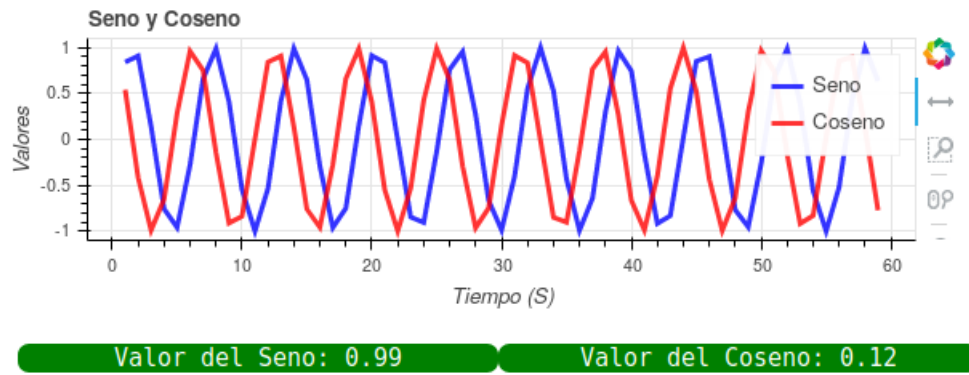


Figura 7: Interfaz Gráfica de Ejemplo

5. Desarrollo de la Experiencia

5.1. Sistema SCADA

Debe implementar un programa en Python que le permita controlar los tanques inferiores de la planta a través de un cliente OPC. Este programa debe tener las siguientes funcionalidades:

- Cliente OPC UA que se conecte al servidor utilizando la librería FreeOpcUa. El cliente debe ser capaz de leer los cuatro niveles de los tanques y escribir valores en las válvulas y en las razones de flujo. Además, debe ser capaz de suscribirse a las alarmas de nivel presentes en el servidor que se activan cuando el nivel de un tanque es menor a 10 cm.
- Interfaz gráfica que sea capaz de mostrar en tiempo real los niveles de los cuatro estanques y los voltajes aplicados a las dos válvulas. Además debe indicar visualmente cuando se genere una alarma.
- Dos modos de control:
 - **Modo Manual:** A través de la interfaz se debe poder controlar el voltaje en las válvulas y las razones de flujo de manera manual.
 - **Modo Automático:** Debe ser capaz de generar dos variables manipuladas para controlar los tanques inferiores utilizando dos referencias y uno o más controladores PID.
- En modo automatico generar dos variables manipuladas como salida de uno o más controladores PID con un ancho de banda limitado para la acción derivativa y que a la vez evite el *wind-up* de la acción integral.

- Guardar las series de tiempo de las variables manipuladas y controladas en algún tipo de archivo como csv, npy o txt.

5.1.1. Detalles a ser implementados

1. *Inputs* del usuario

- Selección de modo Automático o Manual
- Botones para guardar y dejar de guardar la información. Además, su interfaz debe dar la opción de guardar archivos con distintas extensiones.
- En modo Manual:
 - Voltaje modificable por el usuario.
 - Razones de flujo modificables por el usuario.
- En modo automático:
 - Valor de la Referencia
 - Ganancias Proporcional, Integral y Derivativa
 - Parámetros del compensador de *integral wind-up*.
 - Parámetros del filtro de componente derivativa.

2. *Outputs* al usuario:

- En modo manual despliegue de gráficos en el tiempo del nivel de cada tanque y de ambos voltajes aplicados a las válvulas, todo en gráficos separados.
- En modo automático incluir en el gráfico de los tanques que se deben controlar su respectiva referencia.
- Alarma visual cuando el nivel de alguno de los tanques baje de su valor crítico (debe ser una alarma activada por OPC).

3. Consideraciones generales:

- La aplicación deberá permitir la interacción con usuario sin interrumpir la lectura, generación o despliegue en pantalla de las señales.
- La aplicación deberá permitir almacenar los siguientes valores en los archivos antes mencionados:
 - Niveles de los cuatro Tanques.
 - Referencias de los Tanques inferiores.
 - Voltajes aplicados a las válvulas
 - Parámetros del controlador.
 - Razones de flujo.
 - Timestamp.

5.2. Experimentos

1. Demuestre el cumplimiento de cada una de las especificaciones del sistema SCADA.
2. Linealice el sistema y encuentre las matrices A,B y C en el espacio de los estados, luego encuentre los puntos de equilibrio del sistema con los parámetros dados. Finalmente simule el proceso y verifique que efectivamente el sistema se estaciona en esos puntos de equilibrio. Justifique con gráficos.
3. Encuentre la función de transferencia del sistema.
4. En modo manual aplique 2 escalones a las variables manipuladas de manera independiente (una a la vez) de distintos valores cuando el sistema está en fase mínima, es decir, cuando $1 < \gamma_1 + \gamma_2 < 2$, luego registre los valores. Posteriormente, aplique los mismos escalones sobre las variables manipuladas pero ahora cuando el sistema está en fase no mínima, es decir, $0 < \gamma_1 + \gamma_2 < 1$ y registre los valores. ¿Qué diferencias observa en el comportamiento de los tanques? ¿Cuál comportamiento cree que sería más difícil de controlar? ¿Por qué? Justifique apoyado en gráficos.
5. Con el controlador en modo automático y el sistema en Fase Mínima encuentre las ganancias del controlador PID y la ganancia del Anti wind-up que le permita llevar los tanques inferiores a alguna referencia desde un punto de equilibrio. Mida el overshoot, el setting time y el tiempo de respuesta.
6. Nuevamente, con el controlador en modo automático utilice las ganancias encontradas en el punto anterior pero con el sistema en Fase No Mínima. Grafique sus resultados.
7. Finalmente, intente encontrar las ganancias del controlador PID y ganancia Anti wind-up que le permite llevar los tanques a alguna referencia pero ahora con el sistema en Fase No mínima. Mida el overshoot, el setting time y el tiempo de respuesta.

6. Informe Final

El informe debe contener:

1. Un diagrama de flujo del programa implementado.
2. Una breve descripción funcional del programa implementado (incluya imágenes de referencia e instrucciones de uso).
3. Gráficos y análisis pedidos en las actividades de la Sección 5.2.

7. Evaluación

La nota de la experiencia se distribuye como:

- Cumplimiento de las especificaciones del programa: 40 %.
- Informe: 50 %.
- Preguntas hechas durante el desarrollo: 10 %.