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. Uno de los componentes fundamentales 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 clásica, un sistema SCADA consta de un dispositivo de adquisición/envio de datos, una interfaz hombre-máquina, y un algoritmo de control digital.

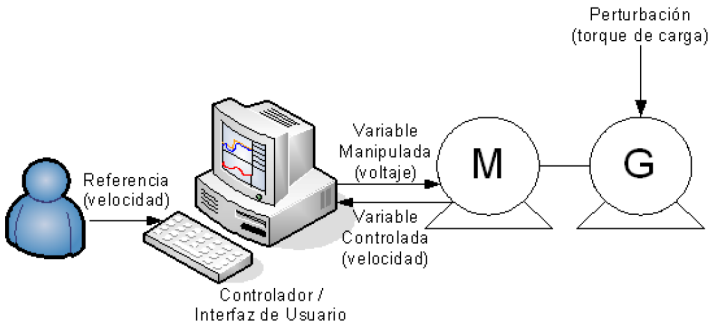

Figura 1: Ejemplo de sistema SCADA para un grupo motor-generador.

La apertura de las tecnologías de automatización ha traido 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.

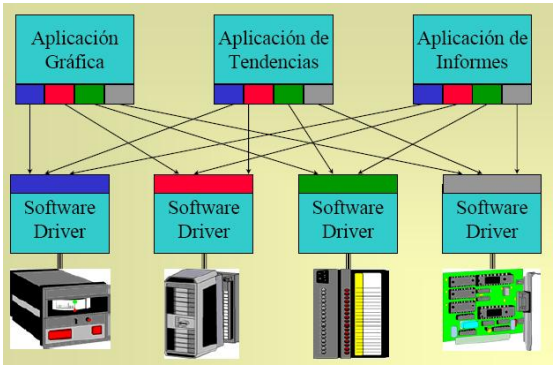

Figura 2: Problema de no contar con OPC

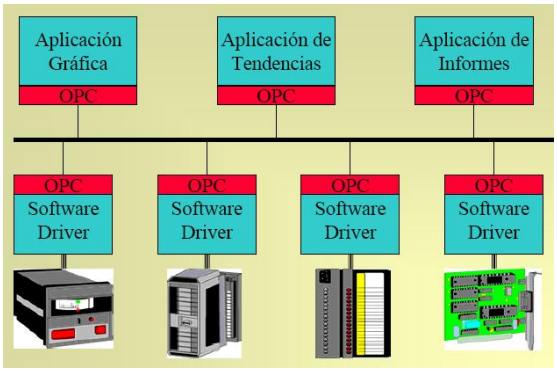

Figura 3: Ventajas de contar con OPC

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 desarrollado lleve la planta a la referencia deseada.

2. Marco Teórico

2.1. Proceso a controlar

El proceso a controlar en esta experiencia es el sistema de los *Cuatro Tanques* que se muestra en la Figura 4.

Este sistema de laboratorio se ha popularizado a lo largo de los años debido a la complejidad que presenta, siendo un sistema MIMO acoplado que puede variar entre un comportamiento de fase mínima y fase no mínima sólo cambiando la razón del flujo que va a los tanques superiores, en relación al que va a los inferiores.

El objetivo de control es regular el nivel de agua en los tanques 1 y 2 (tanques inferiores) a una referencia dada, manipulando las dos válvulas existentes. Se puede apreciar en la Figura 4 que los Tanques 1 y 2 interactúan indirectamente mediante los Tanques 3 y 4 (tanques superiores), modificando la perturbación del otro tanque, lo que hace que el sistema sea complejo de controlar. El sistema de los cuatro tanques se puede modelar en base a las siguientes ecuaciones:

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

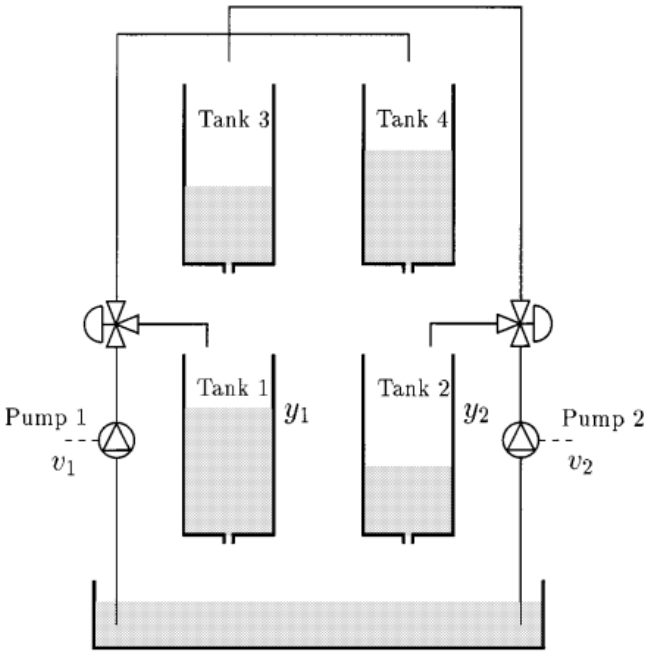

Figura 4: Sistema de los cuatro tanques

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, $\quad j \in \{1, 2\}$
- 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 punto de operación x_0 y u_0 se tiene:

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

$$y = \begin{bmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{bmatrix} x \tag{3}$$

donde:

$$T_i = \frac{A_i}{a_i} \sqrt{\frac{2h_{i0}}{g}} \tag{4}$$

Con esto, la función de transferencia entrada-salida del sistema queda dada por:

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

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

- si $0 < \gamma_1 + \gamma_2 < 1$, entonces hay un cero en el semiplano derecho y el sistema es de **Fase no mínima**.
- si $1 < \gamma_1 + \gamma_2 < 2$, entonces ambos ceros están en el semiplano izquierdo y el sistema es de **Fase mínima**

Para el desarrollo de la experiencia, considere que el simulador del proceso que debe utilizar 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
$v_{1_{max}}, v_{2_{max}}$	10 $[V]$
h_{max}	50 $[cm]$

El simulador posee una interfaz gráfica desarrollada en Python que muestra los niveles y valores de las válvulas en tiempo real, la cual viene por defecto desactivada, pero puede activarla bajo su responsabilidad si estima que le facilita el desarrollo. Usted no debe modificar el código ni del proceso ni de la interfaz.

3. OPC: Ole for Process Control

OPC surge 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 sólo para sistemas Windows y, si bien, se tenían distintas funcionalidades, se necesitaba un servidor distinto para usar cada una de ellas. Algunas de las funcionalidades disponibles 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

Para garantizar una operación más eficiente y transversal, el año 2008 surge **OPC UA** (Unified Arquitecture) que se abrió para cualquier sistema operativo, permite un modelamiento de la información a modo de clases y 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 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.

  Uno puede accesar a un nodo por su NodeId o por su posición path, tal como un sistema de archivos.
- **Subscripción a un nodo**: Para no estar leyendo constantemente los valores de las variables, OPC provee de una funcionalidad mucho más elegante llamada subscripción. Un cliente puede subscribirse a ciertos nodos de interés y dejar que el servidor los monitoree por él. Hay tres tipos distintos de subscripción:
  - Subscripción a cambios en los valores de variables.
  - Subscripción a eventos y alarmas
  - Subscripción a valores agregados, que son valores que se calculan en tiempos definidos por el cliente a partir de los valores originales de las variables.

3.1. Free OPC UA

En esta 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 lea la información de los niveles y escriba posiciones en las válvulas en volts y de las razones de flujo

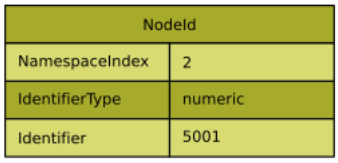

Figura 5: Formato del Identificador de un nodo

(variables γ), además de esto usted deberá incluir la funcionalidad alarmas de OPC UA cuando el nivel de los estanques baje de un nivel crítico.

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 subscriba a un servidor en localhost por el puerto 4048 y luego accesar al nodo padre donde se encuentran todos los objetos, se puede usar el siguiente código:

```
from opcua import Client
from opcua import ua

client = Client("opc.tcp://localhost:4840/freeopcua/server/")
try:
  client.connect()
  objectsNode = client.get_objects_node()
except:
  client.disconnect()
```

3.2. Estructura de datos del proceso

Para poder programar el cliente OPC UA a partir del servidor ya implementado, se debe conocer la estructura de datos presente en el servidor.

La estructura generada para el sistema de los cuatro tanques consiste en cuatro carpetas distintas anidadas según se muestra en la Figura 6: i) carpeta que contiene todos los objetos del proceso; ii) carpeta que contiene los objetos Tanques, en donde cada uno de ellos contiene las variable "h" (altura); iii) carpeta que contiene los dos objetos Válvulas y cada una de ellas contiene la variable "u" que se mide en Volts; y iv) carpeta llamada Razones y que contiene dos objetos "Razon" los cuales contienen la variable "gamma".

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

```
from opcua import Client
from opcua import ua

client = Client("opc.tcp://localhost:4840/freeopcua/server/")
try:
  client.connect()
```

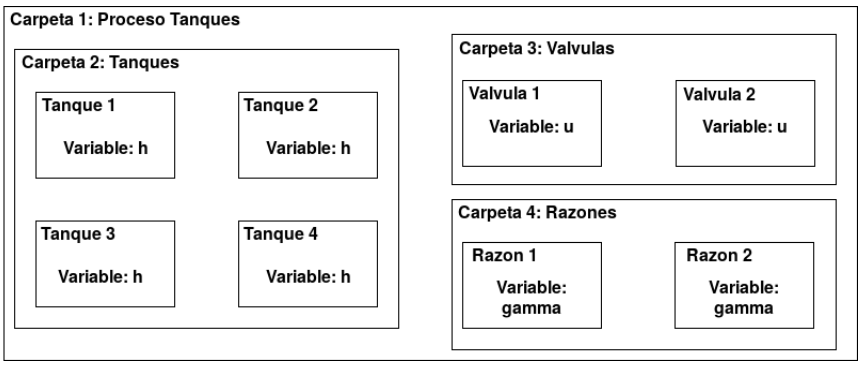

Figura 6: Estructura de datos

```
  objectsNode = client.get_objects_node()
  h1 = objectsNode.get_child(['1:Proceso_Tanques',
  '1:Tanques', '1:Tanque1', '1:h'])
  valor = h1.get_value()
except:
  client.disconnect()
```

4. Interfaz gráfica

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 puede ser complejo. 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 gráficar datos a manera de streaming de manera simple, además tienen una serie de *Widgets* que permiten interactuar con el programa a través de la interfaz.

5. Desarrollo de la Experiencia

5.1. Sistema SCADA

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

- Cliente OPC UA que se conecte al servidor ya implementado, 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 generar alarmas con OPC que se activen cuando uno o más de los niveles sea más bajo que un límite definido por usted.
- Debe desarrollar una 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 de OPC.

- Debe tener dos modos de control:
  - **Modo Manual**: A través de la interfaz 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.
- La interfaz debe incluir la funcionalidad de guardar las series de tiempo de las variables manipuladas y controladas en algún tipo de archivo como csv, npy o txt.

5.1.1. Características generales

1. *Inputs* del usuario
   - Selección de modo Automático o Manual
   - Cantidad máxima de muestras a almacenar en memoria RAM y botón para almacenar las muestras en un archivo.
   - En modo Manual:
     - Voltaje constante modificable por el usuario.
     - Razones de flujo constates modificables por el usuario.
   - En modo automático:
     - Valor de la Referencia
     - Ganancias Proporcional, Integral y Derivativa
     - Parámetros del compensador de *integral wind-up*.
2. *Outputs* al usuario:
   - En modo manual despliegue de gráficos en el tiempo del nivel de cada tanque en gráficos separados y de ambos voltajes aplicados a las válvulas en un único gráfico.
   - En modo automático incluir la referencia en los gráficos de los tanques que se deben controlar.
   - 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 en disco en formato texto una secuencia finita de los siguientes datos:
     - Niveles de los cuatro Tanques.
     - Referencias de los Tanques inferiores.
     - Voltajes aplicados a las válvulas
     - Parámetros del controlador.
     - Razones de flujo.

- La aplicación deberá 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.

5.2. Experimentos

1. Demuestre el cumplimiento de cada una de las especificaciones del sistema SCADA.

2. Linealice el sistema y muestre las matrices A,B y C en el espacio de estados, luego encuentre sus puntos de equilibrio 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 iniciando desde los puntos de equilibrio encontrados anteriormente. 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. Evidencia del cumplimiento de las funcionalidades detalladas en la Sección 5.1.

4. Gráficos y análisis pedidos en las actividades de la Sección 5.2.