Trabajo Fin de Grado

Automatización de una estación virtual en Factory I/O a través de comunicación Modbus y Matlab

Autor

Sergio Lázaro Martí

Director

Cristian Florentín Mahulea

Escuela de Ingeniería y Arquitectura.

2020

ÍNDICE

Índice de Anexos

Índice de Figuras

Resumen

En la actualidad las empresas que basan su actividad en el uso de sistemas de control exigen continuamente innovaciones que aporten una mayor eficiencia precisión y rapidez a la hora de aplicar estos sistemas. Para ello combinan sus conocimientos y experiencias productivas para desarrollar modelos de control fiables que garantizaran el correcto funcionamiento, como pude observar durante mis prácticas empresariales.

Con el objetivo de poder recrear estos modelos de control sin la necesidad de tener una estación física real, he utilizado un programa de simulación llamado Factory I/O el cual permite implementar una estación ficticia, imitando su comportamiento.
El propósito en este trabajo fin de grado es el poder mostrar al alumno como se ha realizado el desarrollo y conexión de la estación simulada, así como mostrar el comportamiento que tendría los diferentes controladores vistos en la asignatura de Ingeniería de Control en la aplicación real.

Para mostrar el comportamiento de los controladores, deberemos primeramente elaborar la estación en Factory I/O sobre la que los vamos a aplicar. En este caso hemos elegido una mezcladora industrial ya que el uso de controladores en este tipo de sistemas es fundamental en el control del dosificado y salida de producto final.

Una vez diseñada la estación debemos desarrollar un código informático que nos permitirá a través de Matlab comunicarnos con dicha estación. Seguidamente se deberá crear los diferentes tipos de controladores y obtener los códigos que nos permita aplicar la acción de esos controladores.

Este trabajo seria aplicable a las prácticas de la asignatura de Ingeniería de Control ya que el calculo de estos controladores es una faceta importante de la asignatura, permitiendo al alumno el desarrollo de los conocimientos adquiridos en este campo. El alumno solo deberá cambiar los parámetros relativos al cálculo de la acción de control.

Paralelamente al objetivo del uso del trabajo como base para las practica se ha calculado un modelo matemático de la simulación a tratar y unos controladores, partiendo de unas especificaciones propias.

Una vez obtenidos los controladores podremos llevar a cabo la prueba del sistema de control, permitiéndonos observar el funcionamiento característico propio de cada de uno de ellos y validando el correcto funcionamiento de la estación.

Como resultado de este proyecto, se puede comprobar de una forma verídica el comportamiento de los diferentes controladores creados, además de haber desarrollado una plataforma simulada a través de la cual poder interactuar con facilidad.

Capitulo I. Introducción

1.1. Motivación

La motivación parte del interés desarrollado al cursar la asignatura de Ingeniería de Control donde se plantean y se profundiza en cuestiones de automatizaciones, las cuales tienen una aplicación real en la industria.

Durante el desarrollo de mis prácticas en la empresa SFS Santiago Francos S.L. [8] dedicada a la automatización de fábricas especializadas en la industria de la alimentación pude valorar directamente las ventajas de una buena programación en los sistemas de control. Es por lo que decidí realizar mi trabajo final de grado en esta área y así aportar un trabajo personal que pueda servir de mejora en las prácticas y aplicaciones de este campo.

Asimismo, quisiera colaborar desde la idea de que una buena práctica en esta área es básica para que el alumno llegue a apreciar las ventajas que traen estos sistemas en los diferentes sectores industriales e incentive la profundización de sus conocimientos en este campo.

1.2. Objetivo del Proyecto

El principal objetivo de este trabajo se centra en la creación de una planta industrial la cual recreara una mezcladora industrial presente en muchísimos sectores industriales. En general, la principal función de una mezcladora es la homogeneización de un producto final constituido por varios componentes, para lo cual se probaran diferentes sistemas de control para realizar esta elaboración, optimizando al máximo el proceso.

En la consecución de mi objetivo he realizado la simulación de la mezcladora por medio del programa Factory I/O. Esto permitirá a un alumno en prácticas conocer la utilidad de este software ya que de una manera muy visual e interactiva nos permite comprobar los efectos del sistema de control.

En el desarrollo de este proyecto la base para conseguir este objetivo y donde reside la dificultad del proyecto ha sido el desarrollar un código en el que se integre la comunicación ente Factory I/O y Matlab. Siendo Matlab el programa base a partir del cual se desarrollará los diferentes sistemas de control. Es en ese punto donde el alumno deberá aplicar los conocimientos obtenidos sobre control y sistemas para poder comprender este trabajo aplicable a una situación real.

1.3. Estructura del documento

Esta memoria se ha estructurado siguiendo el mismo recorrido que se ha seguido a la hora de llevar a cabo el proyecto.

Inicialmente realice una búsqueda de información acerca de los programas informáticos que iban a ser necesarios durante todo el proyecto además de otros programas que finalmente descartamos ya que el enfoque que queríamos desarrollar discernía de las aplicaciones que nos permitían, (Kepserver, Modbus Ethernet OPC configurador…).

Una vez obtuvimos suficiente información y conocimientos acerca de estos programas, pasamos a la creación de la Planta Virtual en que deseábamos controlar mediante el programa Factory I/O, realizando distintas pruebas hasta finalmente llegar al modelo mostrado.

Una vez obtuvimos el modelo simulado de la planta industrial en Factory I/O deseábamos conectarla con Matlab a través del protocolo de comunicación Modbus TCP/IP por lo que nos informamos acerca de que caracterizaba este modelo y como llevar a cabo esta conexión con este modelo.

Con la planta creada y conectada a Matlab mediante Modbus TCP/IP necesitábamos el modelo de control mediante el cual lograríamos llevar a cabo el funcionamiento deseado. Primeramente, realizando el modelado matemático de nuestro sistema explicado en el Anexo I, gracias al cual podemos obtener los diferentes controladores (P, PI…). A continuación, se muestra los controladores y se implementan hasta finalmente simulándolos en Factory I/O.

Para finalizar observaremos las respuestas obtenidas por simulación en Matlab utilizando el modelo matemático linealizado y en Factory I/O utilizando la planta simulada.

Capitulo II. Programas Utilizados

2.1 Matlab

Matlab es un software matemático desarrollado y mantenido por MATHWORKS desde 1984, utilizado tanto por empresas como por la comunidad académica debido a su entorno sencillo y fácil de utilizar. Matlab cuenta con su propio lenguaje de programación llamado M-Code el cual permite trabajar con funciones, datos, matrices, desarrollo de algoritmos e incluso comunicarse con otros lenguajes de programación como C++ o Java.

Una de las características principales de Matlab es la amplia variedad de extensiones de las que se dispone como complemento del programa principal, estas son llamadas ToolBoxes, las cuales utilizaremos a lo largo del proyecto, como Simulink, Modbus Explorer, PID Tuner… como algunos ejemplos.

2.1.1. Simulink

Simulink es un entorno grafico de programación mayormente utilizado para el modelado, simulación y análisis de sistemas dinámicos. Esta plataforma nos permite hacer un modelo de forma rápida y sencilla gracias a la gran cantidad de bloques disponibles en la biblioteca. Anadir que al igual que en el programa principal, Simulink cuenta con numerosas extensiones llamadas BlockSets.

Simulink nos permite simular no solo sistemas continuos sino también nos permite analizar sistemas discretos, proporcionando un amplio rango de aplicaciones, muy utilizado para el control automático y el procesamiento de señales.

Para poder acceder a Simulink desde Matlab hay varios métodos, aunque la forma más rápida es en la pestaña HOME de Matlab seleccionar la extensión de Simulink.
Una vez hemos accedido únicamente deberemos abrir un nuevo modelo en blanco, Blank Model (Figura 1).

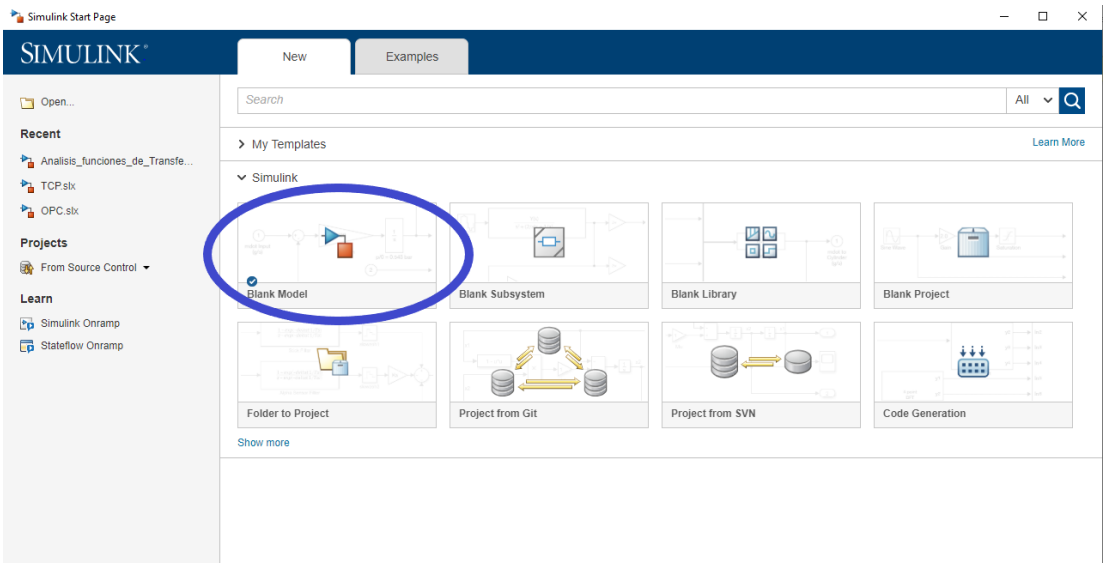


Figura 1. Apertura Simulink

Una vez nos encontremos en el entorno de trabajo podremos ir agregando los bloques que necesitemos desde la biblioteca de Simulink (Figura 2) y en el caso de no estar disponibles podremos o bien añadir una extensión o bien crear nuestro propio bloque, como explicaremos más adelante ya que será nuestro caso.

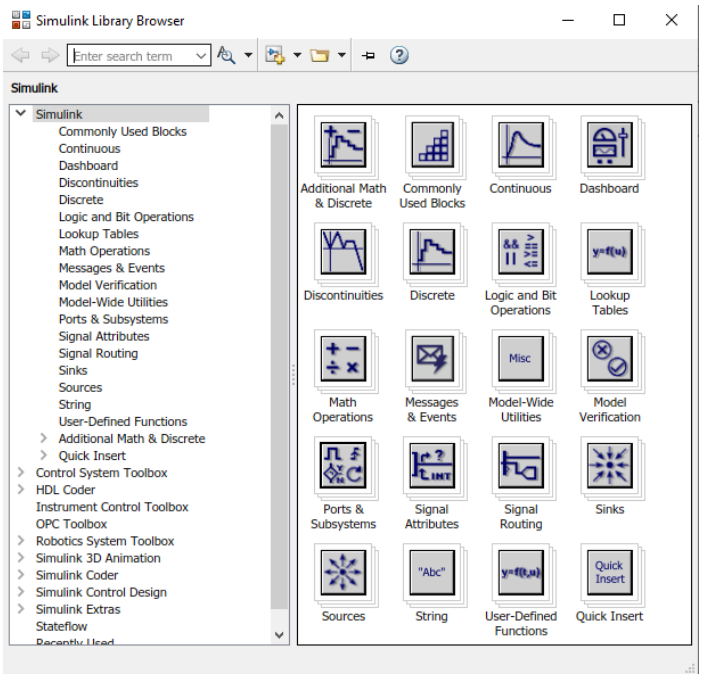


Figura 2. Bloques Simulink

2.1.2. Modbus Explorer

Esta será una de las primeras extensiones de Matlab que utilizaremos ya que permite una rápida conexión entre Factory I/O y Matlab puesto que tiene predeterminada la configuración necesaria para establecer una comunicación Modbus.

Una vez instalada la App deberemos iniciarla desde la barra de Apps. A continuación, elegimos el tipo de configuración que deseamos establecer ya que permite una conexión Serial o TCP/ IP, siendo el ultimo nuestro caso. (Figura 3)

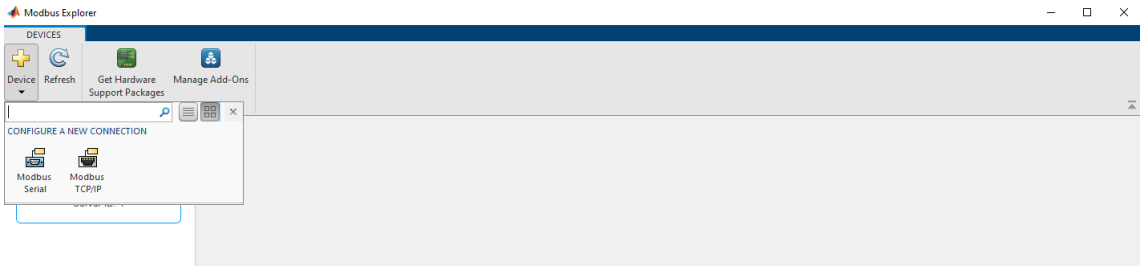


Figura 3. Modbus Explorer

Una vez elegido el tipo de conexión deberemos configurar el servidor al cual deseamos conectarnos. En esta configuración es donde podemos ver la rapidez y sencillez de la interfaz de Matlab debido a que partimos de valores ya predeterminados, como el puerto, tipo de datos o Server ID, teniendo que variar lo mínimo posible. Además de proporcionarnos la explicación pertinente de que es exactamente cada variable (Figura 4).

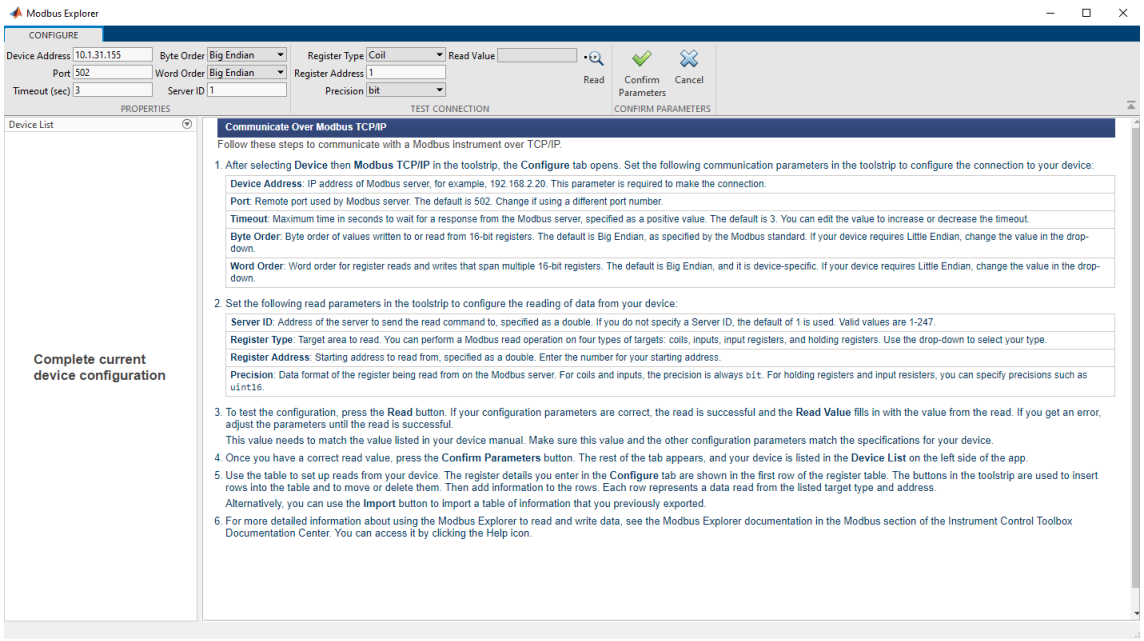


Figura 4. Conexión Modbus Explorer

Una vez conectado al servidor deseado únicamente deberemos ir añadiendo las direcciones que deseemos leer, mediante botón Insert, y ordenar las direcciones introducidas a voluntad, proporcionando una gran personalización.

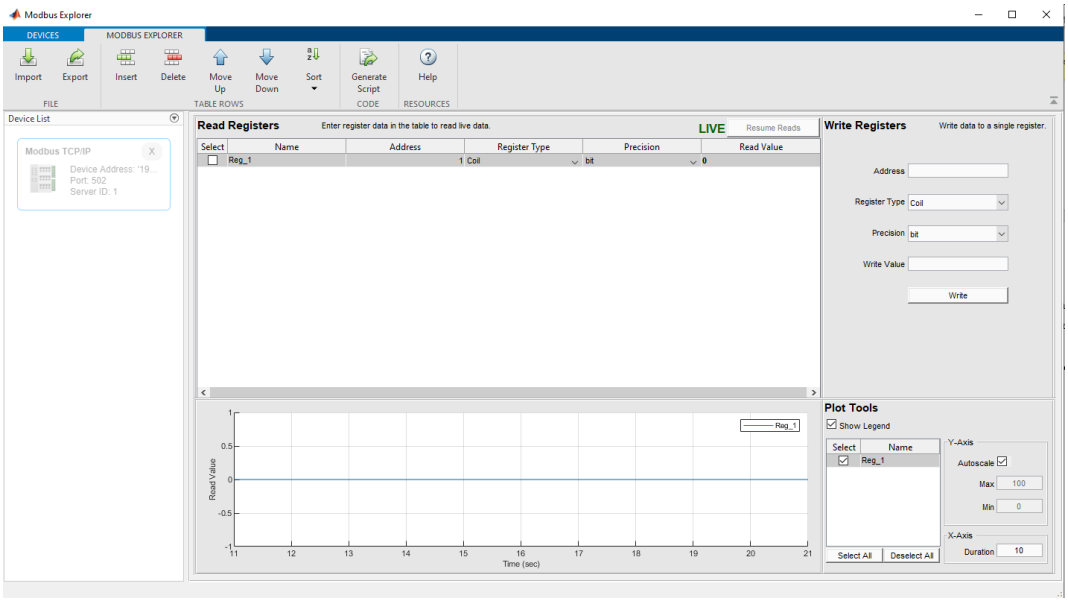


Figura 5. Conexión Modbus Explorer realizada

Una vez introducidos todos los parámetros podemos exportar la configuración como un archivo .m para un rápido cargado en futuros usos, ya que solo deberemos importar el archivo deseado.

Esta extensión nos permite realizar una rápida conexión Modbus y leer todos los valores, pero a la hora de realizar un control está limitado ya que, aunque permite cambiar ciertos valores, lo realiza de forma manual, siendo muy lento. Por lo cual lo utilizaremos solamente en la lectura de ciertas entradas.

2.2 Factory I/O

El software Factory I/O es un programa de simulación 3D el cual permite elaborar una instalación industrial de forma virtual mediante una selección de piezas ya prediseñadas. La existencia de piezas prediseñadas permite una rápida personalización de la fábrica virtual deseada, la cual puede estar constituida por escenarios de baja complejidad hasta escenarios de alta complejidad. Además, Factory I/O cuenta con 20 escenas ya diseñadas permitiéndonos concentrarnos en el aprendizaje del sistema de control.

Este programa ha sido desarrollado por la compañía Real Games con el objetivo de facilitar el aprendizaje acerca de la comunicación de los PLC's, (Programmable Logical Computer), para ello incluye paquetes de drives específicos (Allen-Bradley, Siemens, Schneider, …) permitiendo la comunicación directa con los PLC's, así como otros medios de comunicación industrial como microcontroladores, OPC, Ethernet Industrial y Modbus, el cual nosotros desarrollaremos en este proyecto.

Otro punto muy relevante de Factory I/O es que incluye tanto entradas como salidas lógicas y analógicas, permitiéndonos medir tanto el peso de un objeto como el nivel de un depósito.

2.2.1 Interfaz de creación de escenas

Como hemos comentado con anterioridad Factory I/O ofrece una amplia gama de piezas y escenas, en nuestro caso deberemos crear nuestra propia escena ya que no tomaremos ninguna de las escenas predeterminadas. En este apartado explicaremos el modo de crear una nueva escena, para ello iniciaremos el programa, abriremos un nuevo archivo y procederemos a ir colocando y conectando cada una de las piezas requeridas en una escena nueva.

Una vez nos encontremos dentro de la interfaz nos damos cuenta de su fácil manejo, ya que únicamente deberemos arrastrar las piezas deseadas desde la Paleta al espacio 3D, (Figura 6), pudiendo desplazarla horizontal y verticalmente, manteniendo pulsado la tecla V.

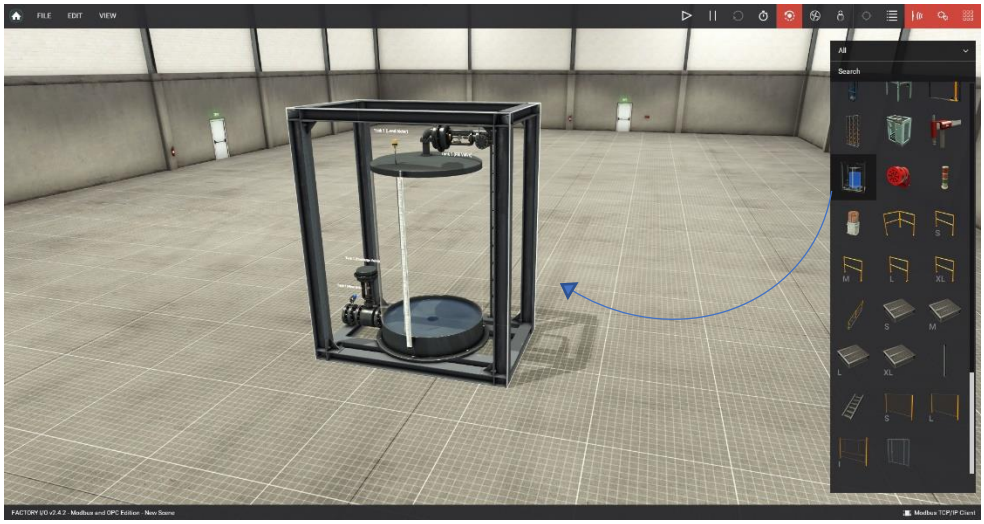


Figura 6. Creación Escena

En caso de querer fijar una altura predeterminada deberemos fijar el punto de interés, únicamente deberemos dar doble clic la altura deseada.

En el caso de añadir varios elementos iguales como estructuras o sensores podremos añadirlos de forma más rápida mediante la opción Duplicado (Alt+Izq). Y trabajar con ellos de forma más fácil mediante su agrupación (Ctrl + G), permitiendo moverlos, rotarlos (Y) y duplicarlos a la vez. Pudiendo una vez acabado el diseño desagruparlos con la opción Desagrupar (Ctrl + G).

Una vez realizado el diseño de la estación podremos guardarla o bien a través de la ToolBar ⟶ File ⟶ Save o directamente pulsando Ctrl + S

Capitulo III. Descripción de la planta

3.1 Mezcladoras en la Industria

Como se ha comentado con anterioridad durante el curso 2018/2019 realicé unas prácticas empresariales en una empresa dedicada al ámbito de la automatización, por lo que pude experimentar de primera mano el método de trabajo.

En este apartado se muestran los modelos reales de mezcladoras con los que se trabaja y diferentes formas de configurarlas. Cabe decir que el principal objetivo es el control de fábricas destinadas a la creación de piensos, por lo que la mayoría de las mezcladoras eran de sólidos.

La más utilizada era la mezcladora de tipo Hélice, debido a su alta extensión en este sector. Su funcionamiento es bastante intuitivo ya que se suministra los materiales deseados desde los silos de almacenaje, posteriormente se introducen en la mezcladora junto con los líquidos deseados, donde se acabarán mezclando de forma homogénea,

Una vez mezclado se abrirán las tajaderas depositando la mezcla o bien en un silo para posteriores procesos o directamente realizarle un proceso como podría ser la granulación.

Figura 7. Mezcladora de Hélice y su interior

Otro modelo de mezcladora es el especializado en líquido, el cual se asemejará más a nuestra estación, este se utiliza para mezclar con anterioridad los diferentes productos químicos que contendrán todos aquellos compuestos que añadirán propiedades/vitaminas al producto final.

Figura 8. Mezcladora de Líquidos

La dosificación de productos a la mezcladora esta controlada por una serie de elemento. En el caso de las mezcladoras de solidos la regulación de materias primas suele hacerse por pesaje, con la ayuda de basculas electrónicas, añadiendo todos aquellos elementos sólidos en la cantidad deseada, y posteriormente añadir los productos líquidos por medio de válvulas neumáticas con contadores volumétricos.

Figura 9. Regulación Mezcladora de Hélice

Para el control de los productos líquidos a la mezcladora el añadido del producto suele realizarse de dos formas distintas, por medio de control de válvulas electromagnéticas o mediante bombas siendo ambos casos controlados en función de los caudalímetros volumétricos (generalmente de tipo masico), las cuales miden directamente el caudal que atraviesa y actúa en función de ello. En el caso de las bombas su funcionamiento se basa en la variación de la velocidad del motor mediante la modificación de la frecuencia
Como podemos observar en la Figura 10, el control de caudales está siendo controlado mediante regulador de frecuencia de las bombas y contamos con los sensores de caudal para confirmar.

Figura 10. Cuadro control Mezcladora de Líquidos

Cabe comentar que toda esta regulación se realiza mediante conexión de 4 a 20 miliamperios, a diferencia del modelo simulado de la planta que trabaja con 0-10V debido a las limitaciones de Factory I/O. Esta regulación con Intensidad es utilizada para evitar los posibles errores de medida por caídas de tensión en la línea entre sensor y controlador, asimismo detentar error por rotura de hilo o avería de sensor ya que nos daría lecturas menores a 4 miliamperios.

3.2 Mezcladora Simulada

Una vez explicado como es el funcionamiento real de las mezcladoras en la industrial y sus componentes, nosotros deberemos construir un modelo que se asemeje lo máximo posible utilizando el software de Factory I/O.

Como hemos comentado con anterioridad Factory I/O dispone de multitud de componentes, a pesar de ello no disponemos de todos los componentes con los que nos gustaría trabajar.

Como podemos comprobar Factory I/O dispone de un tanque con sensores ya integrados, este lo utilizaremos a modo de depósito para materias primas y también a modo de mezcladora ya que no disponemos de un tanque con motor, o un motor independiente que pudiéramos incorporar. Adicionalmente incorporaremos a cada tanque su HMI (Human Machine Interface) el cual contendrá todos aquellos pulsadores, indicadores y displays necesarios para verificar el correcto funcionamiento de la estación por los supuestos operarios en planta. Podremos añadir una serie de estructuras mecánicas para dar una forma más realista a nuestra simulación, obteniendo la simulación final apreciable en la Figura 11

Figura 11. Planta final.

A continuación, vamos a explicar en detalle todos aquellos componentes que nos conforman la estación, estos se dividen en dos categorías principales: actuadores y sensores.
Los actuadores son aquellos dispositivos cuyo objetivo es crear o modificar un efecto sobre un determinado proceso automatizado mediante la transformación de energía eléctrica.
Los sensores en contraposición son aquellos que detectan esos efectos y actúan en consecuencia, transformando normalmente esos efectos en otros tipos de señales, por ejemplo, eléctricas.

3.2.1. Actuadores

El objetivo perseguido es el control del caudal de salida de la mezcladora y su adecuada concentración, por ello el principal actuador en nuestra estación serán las válvulas que nos regularán el caudal proveniente de los depósitos de materias primas, los tanque 1 y 2. Estas nos permitirán regular los caudales que entran en la mezcladora abriéndose o cerrándose, (más adelante veremos que las acciones de los controladores actúan directamente sobre ellas).

Otros actuadores muy importantes en nuestra estación son las bombas de llenado de los tanques, esto se debe a una limitación de Factory I/O. Por el momento nos es imposible conectar los tanques a través de tuberías ya que no están implementadas, esto nos obligara, como veremos más adelante, a realizar una falsa unión de ellos mediante el código informático. Debiendo hacer que el caudal introducido a la mezcladora sea la suma de las lecturas de salida de los tanques 1 y 2. Además los tanques de materia prima que suponemos de elevado tamaño, no son posibles en la simulación por lo que deberemos introducir a ellos un determinado caudal ya que si no se vaciarían.

Una vez comentados los actuadores más destacables comentar que hemos añadido en el HMI un pulsador el cual actuara como botón de pausado del proceso, deteniéndolo en caso de emergencia o para introducir una nueva consigna a la mezcladora.

A continuación, mostramos los actuadores, Figura 12

Figura 12. Actuadores

3.2.2. Sensores

Para poder llevar a cabo el control de la mezcladora vamos a requerir de una serie de sensores que nos permitan observar nuestro sistema.
En nuestro caso el bloque del tanque ya lleva incorporados los sensores que utilizaremos para llevar a cabo ese control, entre ellos están:

- Un sensor de nivel, el cual nos proporciona una señal analogía, obteniendo valores de una forma continua, permitiendo en todo momento conocer la altura.
- Sensor de caudal a la salida de los tanques, que nos permite medir directamente la variable a controlar. Al igual que el sensor de altura nos proporciona de forma continuada una señal eléctrica la cual dependerá del caudal de salida.

En la siguiente imagen podemos ver la disposición de los sensores, Figura 13

Figura 13. Sensores

Todos los sensores que utilizaremos disponen de una constante característica, la que nos relaciona la unidad medida, como por ejemplo m3/s o metros, con voltios. Comentar que la obtención de estas constantes esta explicada en más detalle en el Anexo I.

Un punto a tener en cuenta es que Factory I/O no dispone actualmente de nada parecido a un sensor de concentración, por lo que no nos resulta posible hacer una comprobación de la concentración en el tanque 3. Esta carencia de un sensor de concentración en tanque nos supone que no podemos realizar la comprobación a la salida de la mezcladora, pero lo solucionamos ya que disponemos de la concentración exacta en los tanques 1 y 2, definida en el modelo. Por ello, únicamente deberemos mantener la proporcionalidad a la hora de abrir las válvulas de salida de los tanques 1 y 2, garantizando así la concentración deseada.

3.2.3 Etiquetado

Como hemos comentado, ya conocemos aquellos actuadores y sensores disponibles en la estación, a parte de estos añadiremos aquellos elementos necesarios en la HMI como son luces de señalización y los displays que permitan a los operarios comprobar los valores de la estación.

Todos estos elementos deberemos configurarlos en las etiquetas, las cuales nos permiten vincular los valores de los elementos al controlador. Estas etiquetas están constituidas por un nombre y un valor, que dependiendo del tipo de actuador o sensor pueden ser de 3 tipos:

- Boolean: datos cuyos valores son únicamente On/Off, (0 o 1).
- Integer: contiene datos de valor completos.
- Float: datos con precisión fraccionaria.

Una vez hemos designado estas etiquetas podemos mostrarlas en el propio espacio 3D todos aquellos elementos mediante File→ Dock All Tags.

Un punto muy útil de Factory I/O es que nos permitirá forzar el funcionamiento de ciertos elementos, simplemente actuando en el panel desplegado. Al realizar esto, los valores leídos por los controladores se anularán y prevalecerá el valor forzado, controlando la escena de forma manual o dando un valor determinado a un actuador, muy útil para comprobar posibles fallos en el funcionamiento.

Finalmente, en lo referido a este apartado concluiremos con los drivers de la estación, los cuales deberernos configurar a la hora de conectar la simulación con su futuro sistema de control, en nuestro caso Matlab.

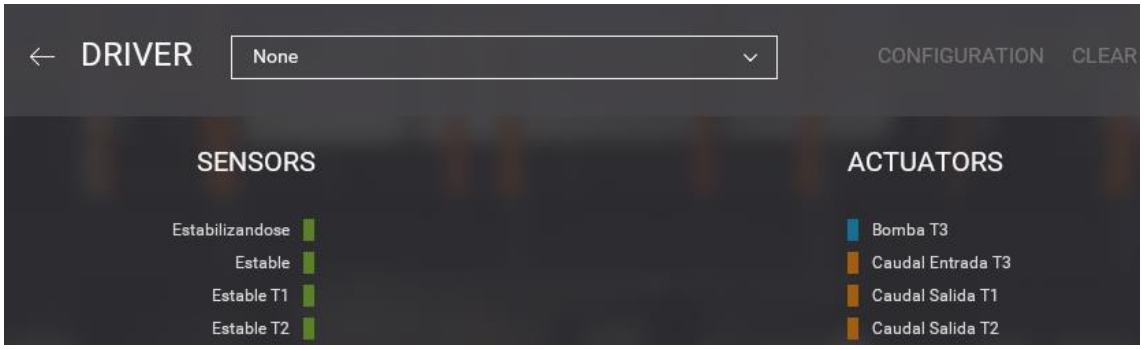


Figura 14. Etiquetado Factory I/O

Por lo cual elegimos entre la amplia variedad de paquetes drivers que ofrece Factory I/O el modelo Modbus TCP/IP Server y procedemos a asignar a cada elemento su dirección característica, ya que en un futuro necesitaremos identificar a que elemento pertenece cada valor que es leído o escrito. Quedando el siguiente modelo:

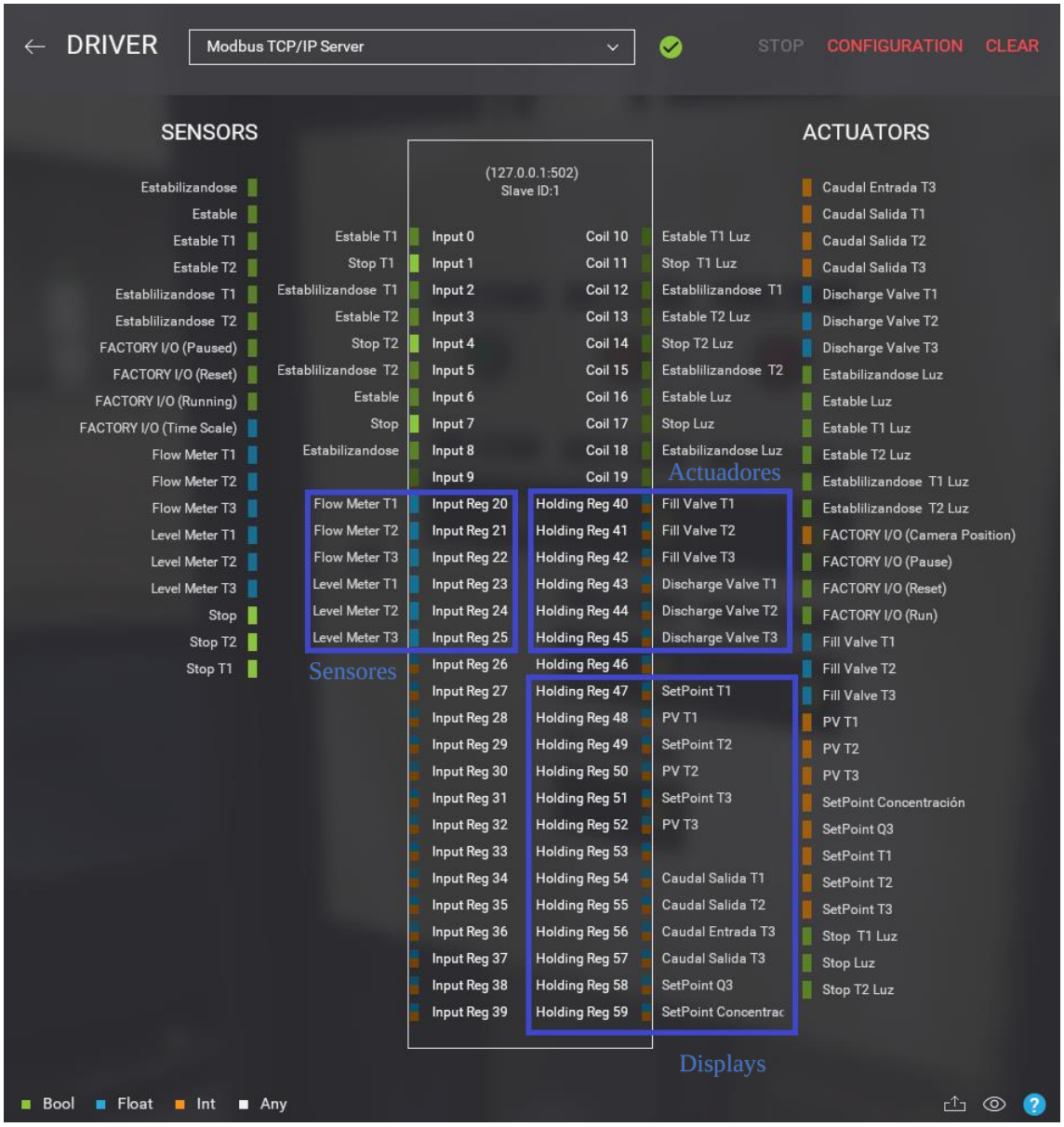


Figura 15. Drivers Factory I/O

Capitulo IV. Conexión entre programas

4.1. Modbus

4.1.1. Introducción

Modbus es un protocolo de comunicación desarrollado por la compañía Modicon en 1979 con el objetivo de comunicarse con sus PLCs. Este ofrecía una comunicación basada en el modelo cliente/servidor para diferentes redes y buses.
Aunque inicialmente fue desarrollado como un protocolo no oficial, a día de hoy sigue creciendo y tiene un gran uso en toda el área de automatización industrial, permitiendo trabajar con redes Ethernet, HDLC y TCP/IP.

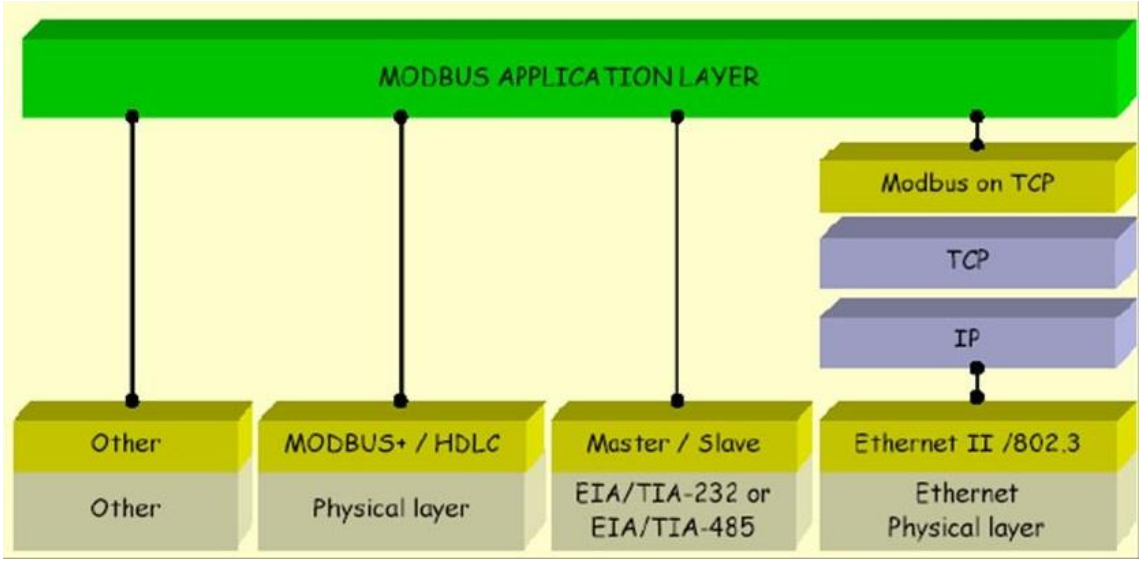


Figura 16. Aplicaciones Modbus. (fuente [6])

4.1.2. Protocolo

Modbus es un protocolo de solicitud-respuesta implementado con una relación maestro-esclavo. Un dispositivo, el esclavo, debe iniciar un requerimiento y espera una respuesta. El maestro, será el encargado de iniciar cada proceso. El protocolo abarca diferentes unidades de datos de aplicación, ya que depende de si la conexión es serial o usando las redes.

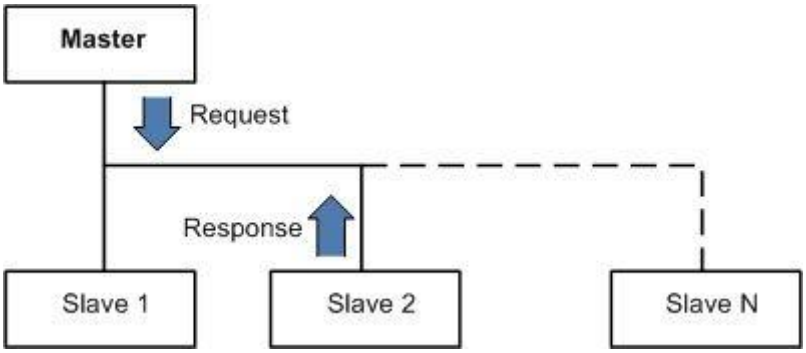


Figura 17. Maestro-Esclavo

Como hemos comentado en Modbus no solo hay un protocolo de comunicación, sino que dependiendo del tipo de comunicación utilicemos variara el modelo básico, algunos ejemplos son: Modbus RTU, Modbus ASCII, Modbus+ o Modbus TCP/IP, el cual vamos a comentar más en detalle.

Modbus TCP/IP

Este protocolo se introdujo para aprovechar las infraestructuras LAN actuales, permitiendo comunicación entre diferentes dispositivos a través de una red Ethernet TCP/IP. Aquí el esclavo se convierte en el servidor y el maestro en el controlador, aunque no se limita a un dispositivo, pudiendo haber múltiples clientes, así como múltiples servidores.

Este protocolo está basado en 4 tipos de mensajes:

- Solicitud, el cual es enviado por el cliente para iniciar conexión
- Indicación, la solicitud llega al servidor
- Respuesta, se envía respuesta por el servidor al cliente
- Confirmación, la respuesta llega al cliente

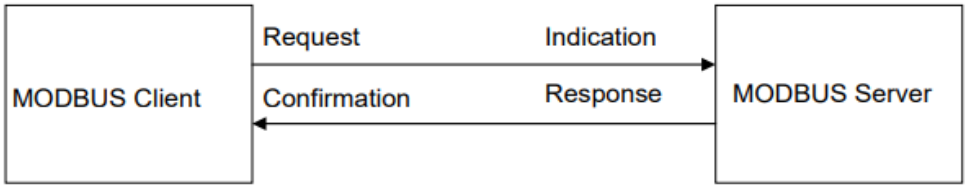


Figura 18. Modbus TCP/IP

4.2. Factory I/O & Matlab

Primeramente, deberemos saber la dirección IP de nuestro servidor simulado en Factory I/O, ya que a través de esta dirección recibiremos o enviaremos todos los datos. Hay que comentar que el modelo TCP/IP tiene una salida máxima de 512 bytes, lo que significa que puede leer 512 bytes a la vez, aunque no tiene que ser obligatoriamente de esta longitud. Además, ya que estamos trabajando con Modbus TCP/IP sabemos que siempre utilizaremos el puerto 502 y en lo referido a los datos trabajaremos con "big-endian", siendo este un modelo de escritura de datos, donde escribiremos los datos hacia la derecha.

Para trabajar con Modbus en Matlab, hay que definir un objetivo de tipo modbus utilizando el siguiente comando,

```
m = modbus('tcpip',ip,502);
```

Donde 'ip' es la IP del ordenador que ejecuta la simulación de Factory I/O y 502 el puerto que se utiliza.

Una vez conectados, como ya conocemos las direcciones de los elementos de la simulación gracias a que con anterioridad hemos realizado el etiquetado, somos capaces de leer y escribir en esas direcciones (dependiendo del tipo del tipo de dato que contenga, mostrados en la siguiente tabla);

Primary tables	Object type	Access type	Comments	Type prefix
Discrete Inputs	Single bit	Read-Only	Provided by an I/O slave	0x
Coils	Single bit	Read-Write	Can be alterable by Master	1x
Input Registers	16-bit word	Read-Only	Provided by an I/O slave	3x
Holding Registers	16-bit word	Read-Write	Can be alterable by Master	4x

Figura 19. Modelo de Datos

La lectura o escritura se realizará mediante los siguientes comandos;

- Lectura;

```
read(obj,'target',address,count,'precision');
```

Cuando realizamos una lectura debemos especificar:

- El `obj`, es el nombre del objeto Modbus, en nuestro caso será `m`.
- target, donde deberemos especificar el tipo de dato utilizado,
- La dirección en la que deseas empezar a leer, `address`.
- El número de lecturas a realizar, count.
- Precision nos permite especificar el formato de datos que está siendo leído.

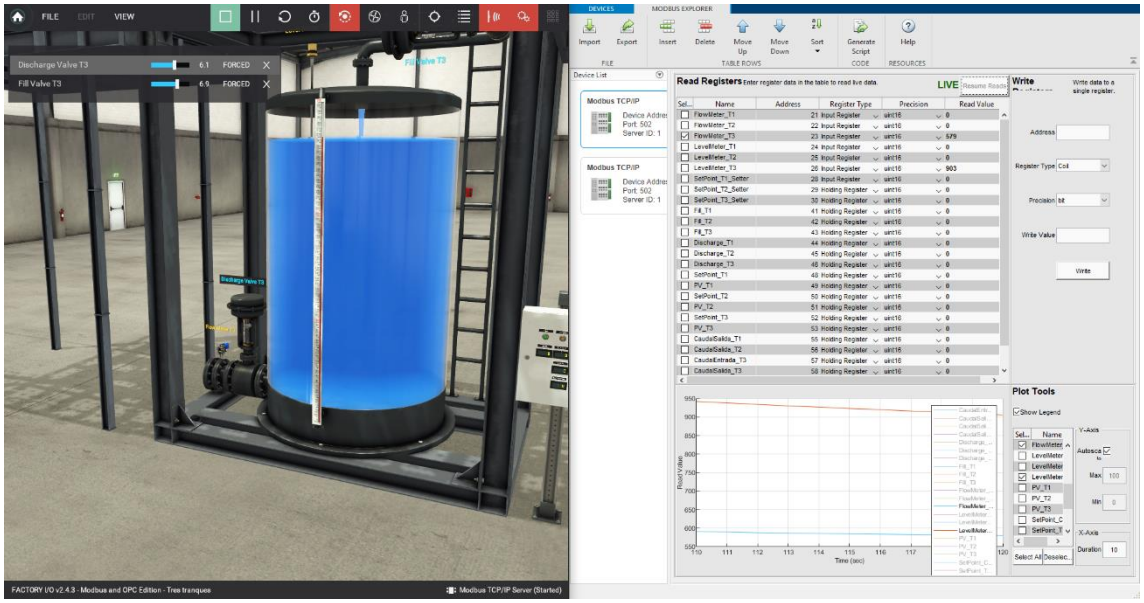


Figura 20. Comprobación Lectura

En la anterior imagen podemos comprobar que gracias a la ToolBox de Modbus Explorer, forzando la salida y entrada de líquido en un tanque, somos capaces de leer los sensores tanto de nivel como del caudal que sale.

- Escritura;

```
write(obj,'target',address,value,'precision');
```

En el caso de la escritura tenemos prácticamente los mismos parámetros con alguna variación:

- Únicamente podremos escribir valores sobre los ‘coils’ & ‘HoldingRegisters’, siendo estos actuadores.
- Y deberemos escribir el valor de las escrituras a realizar.

Uno de los inconvenientes que hemos tenido a la hora de realizar el control ha sido el escribir los valores exactos en Factory I/O y a que esta plataforma trabaja con valores de tipo ‘Uint16’ por lo que en determinadas situaciones hemos tenido que redondear nuestros valores a aplicar.

Una vez conocemos como conectarnos, leer datos y escribirlos procedemos a explicas los métodos de control probados sobre nuestra simulación

Capitulo V. Funcionamiento

5.1. Marco Teórico

Una vez hemos desarrollado la simulación y sabiendo cómo realizar la conexión entre los programas deberemos realizar un modelado matemático de la estación a controlar, gracias al cual podremos comprobar su correcto funcionamiento y diseñar los sistemas de control.

Para realizar este modelado hemos tenido que suponer una serie de Hipotesis en el sistema;

- Los depósitos que contienen las materias primas, se supone que tienen un abastecimiento permanente para la cantidad de materia prima requerida.
- Otro punto para considerar son las válvulas, ya que en la realidad su apertura no es lineal, ya que la salida tiene forma circular y por lo tanto el caudal de salida es una integral en función de la apertura de la válvula. Sin embargo, en nuestro modelo partiremos de que si lo son.

Con el objetivo de que este apartado no resulte muy extenso, el apartado de modelado matemático del modelo se presenta de forma explayada en el Anexo I, en el cual podremos observar el modelo en el espacio de estados.

Una vez obtenido el modelo matemático podemos definir el tipo de control que deseamos realizar, en nuestro caso deseamos controlar el caudal de salida de la mezcladora partiendo de unas proporciones determinadas de materia prima. Este control lo realizaremos actuando sobre las válvulas de salida de los tanques 1 y 2, para poder controlar de una forma eficiente este proceso implementaremos una serie de controladores.

Para entender el proceso de control de caudal, hay que decir que las válvulas trabajan en un rango de 0 a 10 V de DC, este voltaje variara en función del caudal introducido por comando. Cada válvula constara de su propio controlador ya que son elementos independientes, el siguiente esquema muestra la estructura de cada controlador:

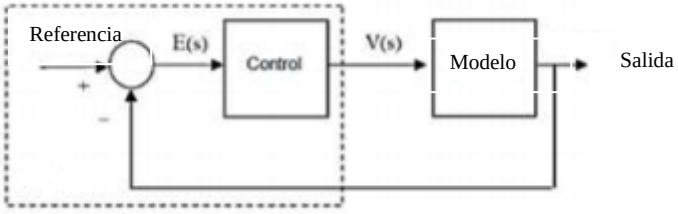


Figura 21. Esquema Control

En el esquema, E representa la señal de error, la cual es la diferencia entre los litros deseados y los que realmente salen. La zona marcada con línea discontinua representa la plataforma en la que se realiza el control.

Inicialmente un controlador calcula la diferencia entre a señal de variable de proceso (PV) y la señal de referencia establecida (SP), lo llamado Error (SP- PV). Este valor representa cuan desviado está el proceso del valor designado, por lo que el controlador deberá incrementar o disminuir su tensión de salida sobre el actuador.

Ya explicada la estructura que vamos a seguir, explicaremos los tres tipos de elementos correspondientes a controladores PID que vamos a desarrollar y probar en nuestro modelo, siendo 3 tipos diferentes:

- Proporcional. - Este elemento viene definido por una constante de proporcionalidad Kp, la cual relaciona el error con la acción de control, de modo que para errores grandes las variaciones de intensidad serán también grandes acciones.

$$u(t) = K_c * e(t)$$

  El principal problema de estos controladores que únicamente utilizan este elemento es la presencia de un error de régimen permanente distinto a cero.

- Integral. - Este elemento proporciona una mejor actuación, ya que en vez de actuar en función del error actúa en función de la integral de error, de modo que mientras la referencia no sea alcanzada la integral del error no para de acumular valores hasta llevar al sistema al punto deseado. La constante propia de este controlador se denomina Ti.

$$u(t) = \frac{K_c}{T_i}\int_0^t e(\tau)d\tau$$

- Diferencial. – Un control de acción derivativa es una acción de control que realiza un desplazamiento en la señal de salida, proporcional a la tasa la cual cambia el error. La acción derivativa alterara la señal de salida en proporción a que tan rápido cambia la señal de entrada, con el objetivo de lograr un sistema de control más suave, evitando los picos de sobre oscilación excesivos. La acción del control diferencial es proporcional a la derivada de la señal de entrada, siendo referida por la constante Td.

$$u(t) = K_c * T_d * \frac{de(t)}{dt}$$

Otro controlador que veremos, aunque únicamente según su modelado matemático, será un control por realimentación de estados. Su principio de funcionamiento es usar el estado del sistema y la entrada de referencia para comandar el proceso a través de la entrada u.

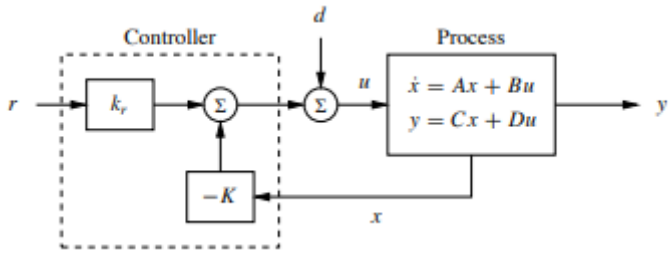


Figura 22. Esquema control por realimentación

5.2. Métodos de control

Este apartado va a ser donde expliquemos como hemos aplicado al modelo simulado aquellas Hipotesis con las que hemos partido, como hemos superado aquellas adversidades provenientes de la falta de determinados componentes, y así como la implementación de los distintos controladores mencionados en el apartado anterior.

Inicialmente vamos a comentar explicando cómo hemos solucionado la Hipotesis de los depósitos de llenado, ya que en la industria a la hora de realizar procesos de mezclado el producto suele estar almacenado en grandes silos, cuyo tamaño difiere mucho del cual nosotros disponemos. Estos silos ya están llenos y a la espera de la acción que les demande el controlador, a diferencia de nuestra simulación en la que partiremos de un estado inicial en la que todos los depósitos están vacíos.

Para soluciona esto deberemos iniciar la simulación con proceso de llenado de los tanques 1 y 2 hasta una altura de trabajo deseado, a partir de la cual podremos empezar a funcionar.

Normalmente los silos trabajan con una capacidad dependiendo del consumo, ya que por tema logística su capacidad suele ser bastante superior, preveyendo posibles paradas del proceso productivo por rotura de stock. Para solucionar esto he llevado a cabo el siguiente código:

```
%Lleno los Tanques de materia prima
while (LevelMeter_T1<=Hmax)&&(LevelMeter_T2<=Hmax)

    write(m,'holdingregs',41,1000,'uint16'); % hago que se llenen rapido hasta el nivel deseado
    write(m,'holdingregs',42,1000,'uint16');

    LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por el sensor
    LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor

    write(m,'holdingregs',49,LevelMeter_T1,'uint16');
    write(m,'holdingregs',51,LevelMeter_T1,'uint16');

    if LevelMeter_T1>=Hmax
         write(m,'holdingregs',41,0,'uint16');
    end
    if LevelMeter_T2>=Hmax
         write(m,'holdingregs',42,0,'uint16');
    end


end % Lleno de depositos de materia prima
```

Figura 23. Código llenado Materias Prima

Mediante este fragmento de código , (Figura 23), abrimos al máximo la entrada de líquido a cada tanque y comprobamos la altura de los depósitos, cuando llega al valor deseado cerramos la válvula de entrada. Como podemos observar antes de comenzar la regulación debemos llenar los silos al máximo así podremos partir de la situación ideal.
Una vez ya estemos en la regulación del sistema deberemos comprobar y corregir en caso pertinente los silos, mediante el siguiendo código;

```
%Control Tanques Materia Primas
LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por el sensor
LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor

if (LevelMeter_T1>Hmax)
    write(m,'holdingregs',41,0,'uint16');
elseif(LevelMeter_T1<Hmin)
     write(m,'holdingregs',41,1000,'uint16');
else
    Red1 = round(100*(Kin/Kout)*(SetPoint_Q1_Voltios),0);
    write(m,'holdingregs',41,Red1,'uint16');
end

if (LevelMeter_T2>Hmax)
    write(m,'holdingregs',42,0,'uint16');
elseif(LevelMeter_T2<Hmin)
    write(m,'holdingregs',42,1000,'uint16');
else
    Red2 = round(100*(Kin/Kout)*(SetPoint_Q2_Voltios),0);
    write(m,'holdingregs',42,Red2,'uint16');
end
```

Figura 24. Tanque MP en Regulación

Vemos que si alguno de los tanques de materias primas se encuentra por debajo del rango de trabajo indicaremos que deberemos abrir al máximo la válvula de entrada de materia prima ya que sino nuestro deposito se vaciaría. Así como si se encuentra lleno deberemos cerrar las válvulas ya que podría sobrealimentarse produciendo una pérdida de materia prima o un atascado de la válvula de suministro. El último apartado es debido a que nuestro tanque tiene un rango más limitado por lo que estaría continuamente oscilando la entrada de materia prima provocando una contante perturbación en el sistema ya que utilizamos como fuerza de transporte la gravedad, por lo que la presión de vaciado podría variar, por lo que introducimos el valor de referencia que debería salir, así de esta forma logramos sobrellevar el problema de suministro manteniendo una zona de trabajo.

A continuación, quiero comentar que el problema de la no linealidad de las válvulas se soluciona directamente en Factory I/O debido a que el mismo programa las considerará lineales haciendo el voltaje aplicado sea proporcional a su grado de apertura.

Una vez tenemos los tanques llenos necesitaremos conocer los valores que deseamos obtener, haciendo que nuestro programa pueda obtener los valores de referencia.
Estos deberán ser introducidos en la barra de comandos de Matlab. (Ver Anexo IV).
Conocido el estado deseado realizaremos una serie de cálculos para obtener los valores de caudal del tanque 1, Q1, y caudal del tanque 2, Q2, para que se respete la concentración deseada.
Una vez llenos los tanques de materias primas y conocidos los valores estipulados podremos comenzar a llenar la mezcladora, debiendo ser muy cuidadoso este proceso para respetar las concentraciones y a que como hemos comentado con anterioridad, es aquí donde se realizara el control de concentración.
Abriremos las válvulas de los tanques 1 y 2 comenzando la regulación. Para ello deberemos fijar el tiempo de muestreo de nuestro sistema e implementar el código necesario para cada modelo de controlador. Hay que comentar que el tiempo de muestreo que hemos utilizado es de 0.1 segundos, esto se debe a que no es posible coger valores inferiores ya que debemos dejar tiempo al procesado matemático y que, si eligiésemos valores más elevados del orden de 1 segundo, el sistema se volvería inestable

Como podemos observar aquí surge el problema de no poder conectar los tanques de forma física por lo que deberemos leer las salidas de los tanques 1 y 2 y manualmente asegurar que la entrada de la mezcladora (tanque 3) sea el valor de la suma de ambas salidas.

```
%Caudal que entra en T3 procedente de T1 y T2
QEntrada_T3= 0.3543*Voltaje_caudal_salida_1 + 0.3543*Voltaje_caudal_salida_2; %Leo salida real T1 y T2
Q = round(QEntrada_T3,0);
write(m,'holdingregs',57,Q,'uint16');% Muestro por pantalla lo que entra

Q3_aplic =(Kin/Kout)*(Voltaje_caudal_salida_1 + Voltaje_caudal_salida_2); %Obtengo entrada real T3
Q3_apli_redondeado = round(Q3_aplic,0);
write(m,'holdingregs',43,Q3_apli_redondeado,'uint16'); % aplico la accion de caudal de entrada
```

Figura 25. Caudal de entrada mezcladora

El primer controlador que vamos a tratar es el modelo más sencillo, *un control proporcional*. Para ello necesitaremos realizar la lectura de caudal a la salida del tanque 3 y compararlo con el valor fijado de referencia, obteniendo el error, esta obtención será válida para todos los controladores;

```
Ek = Q_3_ref - Q_3_real;
Ek_1 = (SetPoint_Q1_L/SetPoint_Q3_L)*Ek ;
Ek_2 = (SetPoint_Q2_L/SetPoint_Q3_L)*Ek;

% esto es debido a que contamos con 2 controladores
% independientes,por lo que si aplicasemos directamente el error Ek
% estariamos aplicando el doble de error, ademas de aplicarle las
% misma proporciones a cada uno, por lo que acabiran igualandose.
%Ek = Q_3_ref - Q_3_real;
%Ek = (Q_1_ref + Q_2_ref) - (Q_1_real + Q_2_real);
%Ek = (Q_1_ref - Q_1_real) + (Q_2_ref - Q_2_real);
%Ek = Ek_1 + Ek_2;
```

Figura 26. Calculo Error

Para poder aplicar las acciones de los controladores deberemos pasar de la forma matemática de cada controlador a código, estos pasos están descritos en el Anexo II.

A continuación, ya obtenida la constante Kc, Anexo III, obtendremos la acción y comprobaremos que esta esté entre los márgenes posible, finalmente aplicaremos la acción resultante en las válvulas 1 y 2, de forma independiente.

```
%Valor P
Uk_1 = P*Ek_1;
% creamos metodo saturacion
if (Uk_1 > 1)
    Uk_1 = 1;
elseif (Uk_1 < 0)
    Uk_1 = 0;
end

%Aplicamos valor %
Qent1 = 1000*Uk_1; % paso valor referencia factory ya que en factory no es [0-1] sino [0-1000]
Porcentaje_V1 = round(Qent1,0); % redondeo para que sea uint16 y poder aplicarlo
write(m,'holdingregs',44,Porcentaje_V1,'uint16') % aplico la accion
```

Figura 27. Proporcional 1

```
%Valor P
Uk_2 = P*Ek_2;
% creamos metodo saturacion
if (Uk_2 > 1)
    Uk_2 = 1;
elseif (Uk_2 < 0)
    Uk_2 = 0;
end
%Aplicamos valor %
Qent2 = 1000*Uk_2; % paso valor referencia factory
Porcentaje_V2 = round(Qent2,0); % redondeo para que sea uint16 y poder aplicarlo
write(m,'holdingregs',45,Porcentaje_V2,'uint16') % aplico la accion
```

Figura 28. Proporcional 2

El siguiente controlador es un PI el cual contiene tanto la acción proporcional como la integral, en este caso deberemos inicializar las variables anteriores ya que como hemos explicado dependemos de la interacción anterior para determinar la acción, siendo actualizadas una vez se ha aplicado la acción. Una vez obtenido el controlador, Anexo III, aplicamos el control y comprobamos márgenes

```
%Valor PI
Uk_1 = Uk_1_ant + 0.3*Ek_1 - 0.29922*Ek_1_ant;

%Guardo variables ciclo
Ek_1_ant = Ek_1;
Uk_1_ant = Uk_1;

% creamos metodo saturacion
if (Uk_1 > 1)
    Uk_1 = 1;
elseif (Uk_1 < 0)
    Uk_1 = 0;
end

%Aplicamos valor %
Qent1 = 1000*Uk_1; % paso valor referencia factory
Porcentaje_V1 = round(Qent1,0); % redondeo para que sea uint16 y poder aplicarlo
write(m,'holdingregs',44,Porcentaje_V1,'uint16') % aplico la accion
```

Figura 29. Controlador PI 1

```
%Valor PI
Uk_2 = Uk_2_ant + 0.3*Ek_2 - 0.29922*Ek_2_ant;

%Guardo variable ciclo
Ek_2_ant = Ek_2;
Uk_2_ant = Uk_2;

% creamos metodo saturacion
if (Uk_2 > 1)
    Uk_2 = 1;
elseif (Uk_2 < 0)
    Uk_2 = 0;
end

%Aplicamos valor %
Qent2 = 1000*Uk_2; % paso valor referencia factory
Porcentaje_V2 = round(Qent2,0); % redondeo para que sea uint16 y poder aplicarlo
write(m,'holdingregs',45,Porcentaje_V2,'uint16') % aplico la accion
```

Figura 30. Controlador PI 2

Una vez calculada la acción deberemos guardar los valores de esta interacción para la siguiente.

Adicionalmente probaremos un tipo de controlador llamado DeadBeat el cual es un controlador el cual pretende minimizar el tiempo de respuesta del sistema en bucle cerrado. Esto resulta en unas acciones de control iniciales muy grandes.

```
%Valor DB
Uk_1 = Uk_1_ant + 1.9888*Ek_1 - 1.9836*Ek_1_ant;

%Guardo variables ciclo
Ek_1_ant = Ek_1;
Uk_1_ant = Uk_1;

% creamos metodo saturación
if (Uk_1 > 1)
    Uk_1 = 1;
elseif (Uk_1 < 0)
    Uk_1 = 0;
end

%Aplicamos valor %
Qent1 = 1000*Uk_1; % paso valor referencia factory
Porcentaje_V1 = round(Qent1,0); % redondeo para que sea uint16 y poder aplicarlo
write(m,'holdingregs',44,Porcentaje_V1,'uint16') % aplico la accion
```

Figura 31. DeadBeat 1

```
%Valor DB
Uk_2 = Uk_2_ant + 1.9888*Ek_2 - 1.9836*Ek_2_ant);

%Guardo variable ciclo
Ek_2_ant = Ek_2;
Uk_2_ant = Uk_2;

% creamos metodo saturacion
if (Uk_2 > 1)
    Uk_2 = 1;
elseif (Uk_2 < 0)
    Uk_2 = 0;
end

%Aplicamos valor %
Qent2 = 1000*Uk_2; % paso valor referencia factory
Porcentaje_V2 = round(Qent2,0); % redondeo para que sea uint16 y poder aplicarlo
write(m,'holdingregs',45,Porcentaje_V2,'uint16') % aplico la accion
```

Figura 32. DeadBeat 2

Finalmente aplicaremos un control PID. La codificación de su función de transferencia ha sido obtenida partiendo de la función proporcionada por el bloque PID de simulink, proporcionando así más cercanía entre resultados. (Anexo II).

Iniciaremos las variables antes de comenzar la regulación, justo con las constantes características del controlador. Y tras realizar cada interacción deberemos guardar los valores para las próximas interacciones.

```
%Valor PID
Uk_1 = Uk_1_ant + P*( (Kd + 1)*Ek_1 + (Ki - 2*Kd - 1)*Ek_1_ant + Kd*Ek_1_ant_ant);

%Guardo Variables ciclo
Uk_1_ant = Uk_1;
Ek_1_ant_ant = Ek_1_ant;
Ek_1_ant = Ek_1;

% creamos metodo antiwindup
if (Uk_1 > 1)
    Uk_1 = 1;
elseif (Uk_1 < 0)
    Uk_1 = 0;
end

%Aplicamos valor %
Qent1 = 1000*Uk_1; % paso valor referencia factory
Porcentaje_V1 = round(Qent1,0); % redondeo para que sea uint16 y poder aplicarlo
write(m,'holdingregs',44,Porcentaje_V1,'uint16') % aplico la accion
```

Figura 33. PID 1

```
%Valor PI
Uk_2 = Uk_2_ant + P*( (Kd + 1)*Ek_2 + (Ki - 2*Kd - 1)*Ek_2_ant + Kd*Ek_2_ant_ant );

%Guardo Variables ciclo
Uk_2_ant = Uk_2;
Ek_2_ant_ant = Ek_2_ant;
Ek_2_ant = Ek_1;

% creamos metodo antiwindup
if (Uk_2 > 1)
    Uk_2 = 1;
elseif (Uk_2 < 0)
    Uk_2 = 0;
end

%Aplicamos valor %
Qent2 = 1000*Uk_2; % paso valor referencia factory
Porcentaje_V2 = round(Qent2,0); % redondeo para que sea uint16 y poder aplicarlo
write(m,'holdingregs',45,Porcentaje_V2,'uint16') % aplico la accion
```

Figura 34. PID 2

De entre estos controladores implementados el más utilizado en la industria es el controlador PID.

Adicionalmente quiero comentar que en la industria se suele añadir un control On/Off. Este control consiste en fijar un punto a partir del cual comenzara la regulación, en nuestro caso sería llenar el depósito hasta un punto deseado (el punto de equilibrio) y una vez ahí iniciar la regulación acortando significativamente los tiempos de respuesta de los sistemas.

Este control On/Off lo implementaremos a continuación del llenado de los tanques 1 y 2, llenado el tanque 3 hasta un punto designado respetándolas durante el llenado las proporciones de llenado, manteniendo la válvula de salida de la mezcladora cerrada hasta dicho punto.

```
D1 = 1000*(SetPoint_Q1_L/SetPoint_Q3_L);
D2 = 1000*(SetPoint_Q2_L/SetPoint_Q3_L);

%     %Lleno hasta punto de equilibrio T3
while (LevelMeter_T3<=Level_Q)

    write(m,'holdingregs',44,D1,'uint16');% Hago que la valvula 1 se abra para el caudal deseado
    write(m,'holdingregs',45,D2,'uint16');% Hago que la valvula 2 se abra para el caudal deseado

    write(m,'holdingregs',41,1000,'uint16'); %Introduzco al maximo materias primas T1
    write(m,'holdingregs',42,1000,'uint16'); %Introduzco al maximo materias primas T2

    %Limitador Tanques Materia Primas para que no desborden
    LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por el sensor
    LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor
    write(m,'holdingregs',49,LevelMeter_T1,'uint16');%Muestro nivel de los tanques
    write(m,'holdingregs',51,LevelMeter_T1,'uint16');

    if (LevelMeter_T1>Hmax)
        write(m,'holdingregs',41,0,'uint16');
    end
    if (LevelMeter_T2>Hmax)
        write(m,'holdingregs',42,0,'uint16');
    end

    Voltaje_caudal_salida_1 = read(m,'inputregs',21,1,'uint16'); %Leo Caudales de salida
    Voltaje_caudal_salida_2 = read(m,'inputregs',22,1,'uint16');

    Q_salida_T1=0.3543*Voltaje_caudal_salida_1;%Paso de voltios a litros
    Qs1=round(Q_salida_T1,0);

    Q_salida_T2=0.3543*Voltaje_caudal_salida_2;
    Qs2=round(Q_salida_T2,0);

    write(m,'holdingregs',55,Qs1,'uint16');%Muestro caudal salida de los tanques
    write(m,'holdingregs',56,Qs2,'uint16');

    QEntrada_T3= Q_salida_T1 + Q_salida_T2;
    Q = round(QEntrada_T3,0);
    write(m,'holdingregs',57,Q,'uint16');% Muestro por pantalla lo que entra
    Q3_aplic =(Kin/Kout)*(Voltaje_caudal_salida_1 + Voltaje_caudal_salida_2);
    Q3_apli_redondeado = round(Q3_aplic,0);
    write(m,'holdingregs',43,Q3_apli_redondeado,'uint16'); % aplico la accion
    %Leo Nivel T3 y lo muestro
    LevelMeter_T3 = read(m,'inputregs',26,1,'uint16');
    write(m,'holdingregs',53,LevelMeter_T3,'uint16');
end
```

Figura 35. Control On/Off

5.3. Validación

En este apartado vamos a comprobar el correcto funcionamiento de la estación con los diversos controladores implementados.

Proporcional

Inicialmente vamos a mostrar un video en el que podremos apreciar su funcionamiento. En el video mostraremos como realizamos los procesos de llenado de los tanques de materias primas. Una vez alcanzado el llenado se ejecutará la regulación, mostrando las características determinadas de este tipo de control.
En el siguiente enlace mostramos un video de la simulación:

https://youtu.be/k0snJUX3kWc

Como hemos observado en este control, Podemos observar el principal problema, el cual es el error en régimen permanente, el cual nunca es cero. Siendo en nuestro caso un caudal final de salida 190 m^3/s , frente a los 200 fijados.
Si comprobamos el controlador mediante simulink ocurre lo mismo, existencia de un error en régimen permanente, en lo referido al tiempo de respuesta se ve condicionado igualmente por la saturación impuesta por el límite físico del sistema.

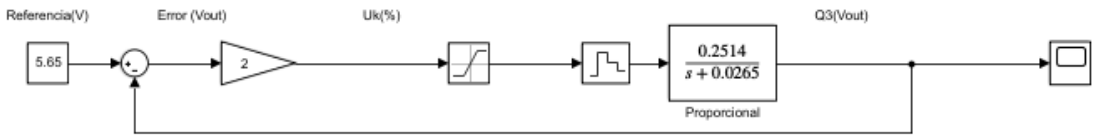


Figura 36. Proporcional Simulink

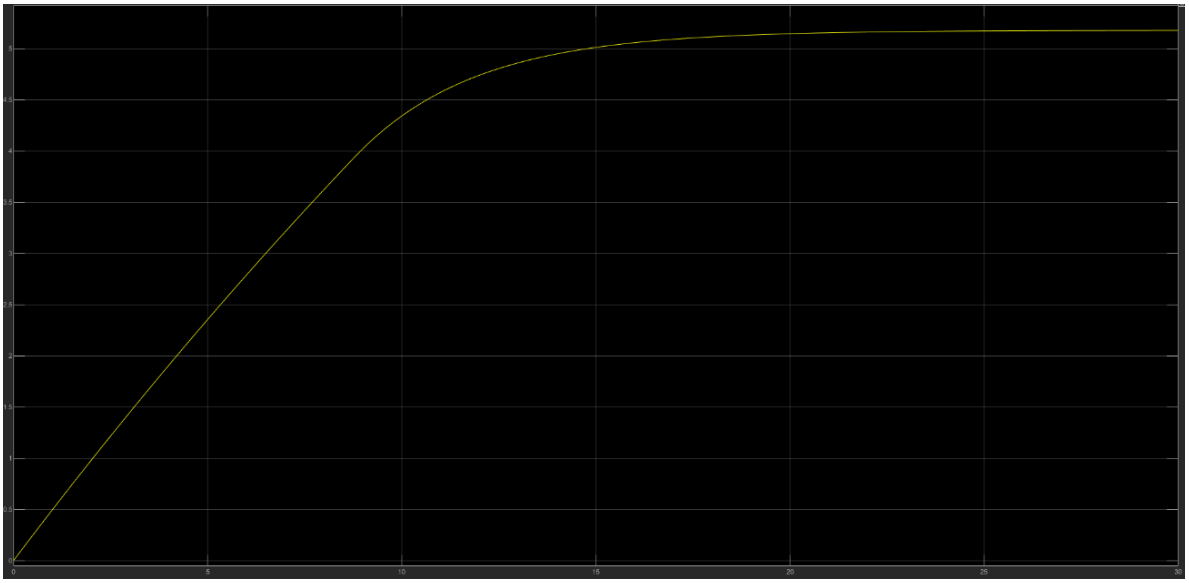


Figura 37. Proporcional Simulink Respuesta

Proporcional e Integral (PI)

Al igual que con el controlador proporcional comenzaremos con el llenado de los tanques 1 y 2, seguido de la regulación del sistema.

El video del control lo podemos ver en el siguiente enlace:

https://youtu.be/rQPu3mY1Nv8

Como vemos a diferencia del controlador Proporcional hemos sido capaces de eliminar el error en régimen permanente, fijando el valor designado.

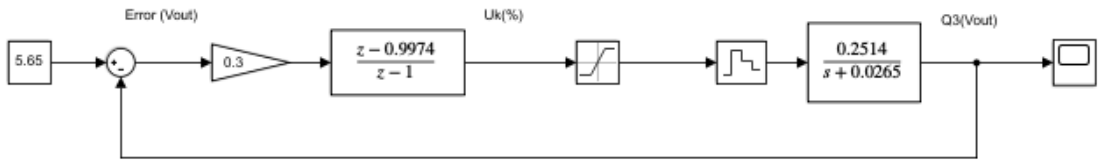


Figura 38. PI Simulink

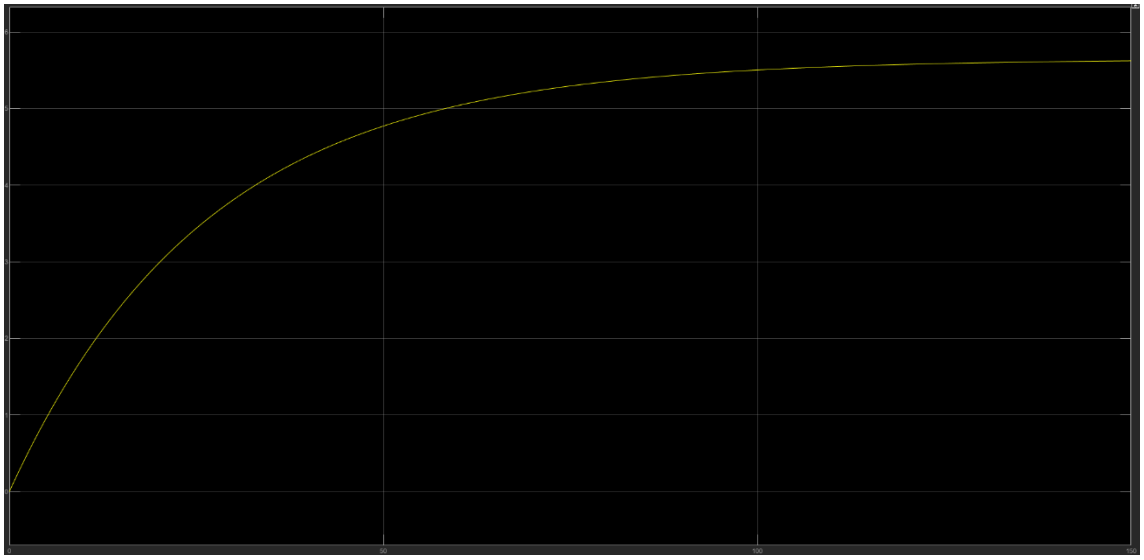


Figura 39. PI Simulink Respuesta

Mediante las ToolBox Modbus Explorer hemos podido medir el tiempo de respuesta siendo:

- *Tr = 110 segundos*

DeadBeat

Este controlador es un tipo de control digital que he incluido ya que coloca todos los polos de lazo cerrado en el origen, logrando una mejora en el tiempo de respuesta y eliminando el error de posición en el régimen permanente.

El video del control lo podemos ver en el siguiente enlace:

https://youtu.be/QPPntJ1Z2tk

En el video hemos podido comprobar que logra la eliminación del error de posición y mejora el tiempo de respuesta de la estación.

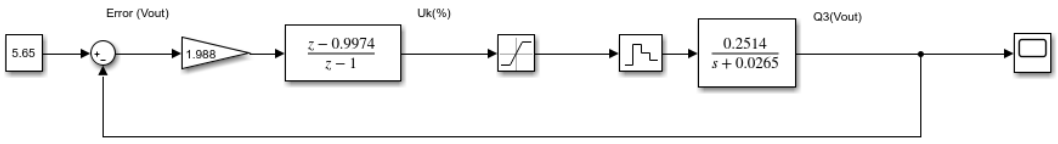


Figura 40. DeadBeat Simulink

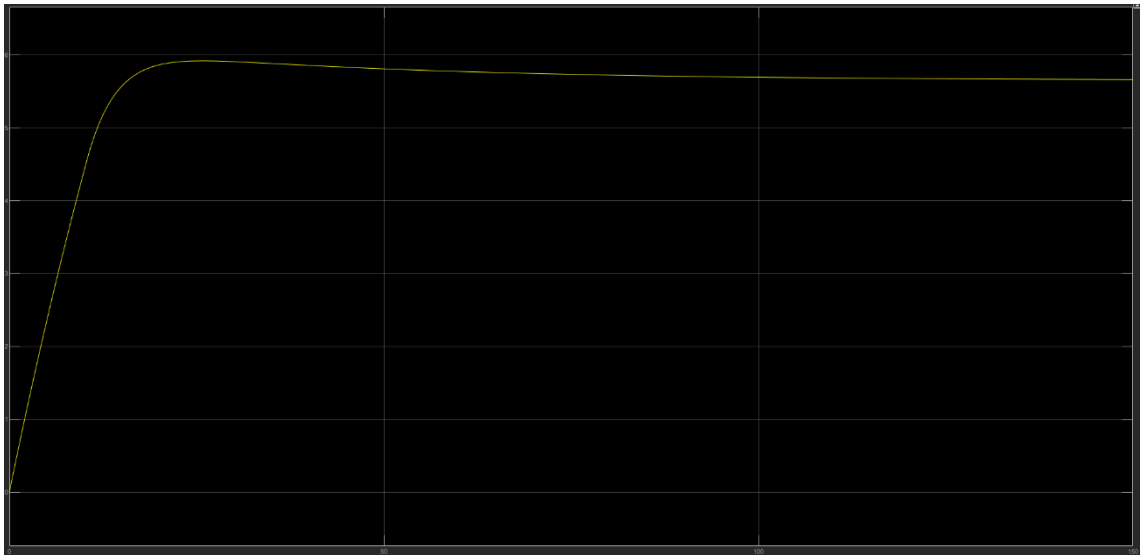


Figura 41. DeadBeat Simulink Respuesta

Mediante la ToolBox Modbus Explorer hemos podido obtener el tiempo de respuesta siendo:

- *Tr = 78 segundos*

Proporcional, Integral y Derivativo (PID)

En el siguiente enlace podemos ver un video de la simulación siendo controlada por un controlador PID:

https://youtu.be/Kxeidra8e8Q

Como conocemos el controlador PID es el más complejo y adecuado ya que logra corregir el error en régimen permanente y logra alcanzar el régimen permanente en un menor tiempo que el PI.

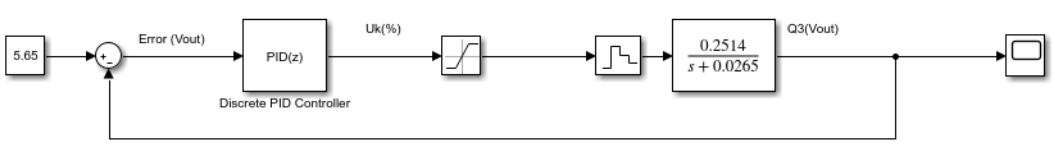


Figura 42. PID Simulink

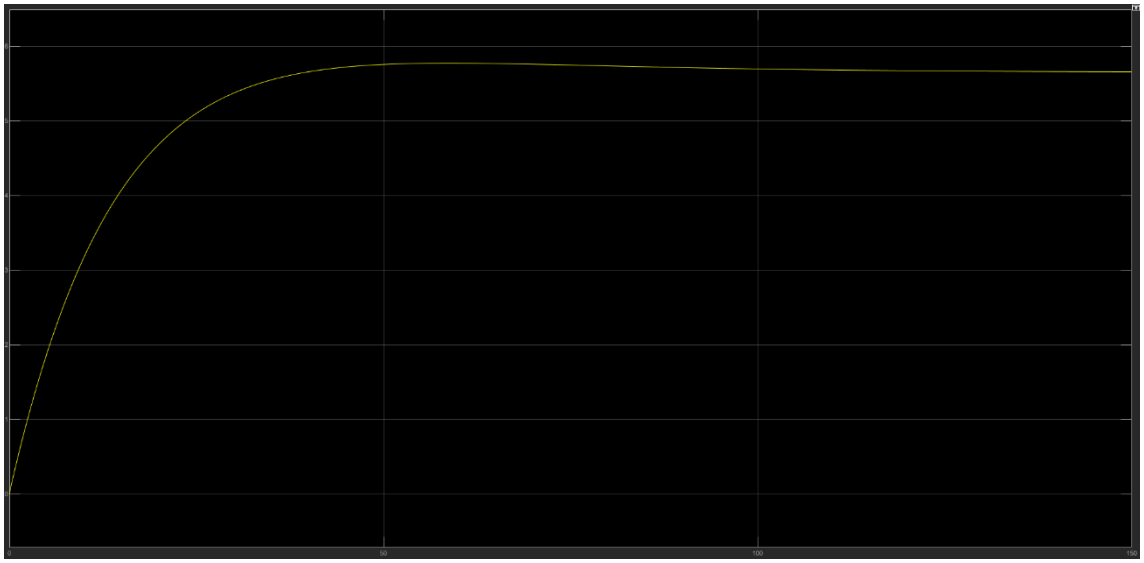


Figura 43. PID Simulink Respuesta

Estas mejoras las podemos observar claramente ya que tiene un tiempo de respuesta considerablemente inferior, medido con la ToolBox Modbus Explorer obtenemos:

- *Tr = 28 segundos*

En el siguiente enlace podemos ver un video donde implementamos el controlador On/Off

https://youtu.be/e8iqtBAS8rM

Por todos los motivos mencionados y la comprobación realizada podemos entender porque es el mas utilizado en la industria. Debido a su rápida respuesta y optimizar los procesos productivos lo máximo posible.

Realimentación de estados (L)

Este modelo no lo hemos llegado a aplicar pero como podemos observar lograríamos un sistema, el cual se estabilizaría al igual que el PID rápidamente y permitirá un control más simplificado de los dos sistemas en paralelo.

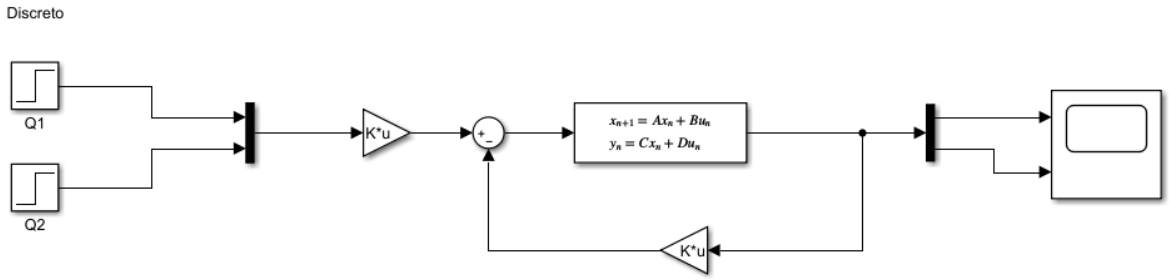


Figura 44. Control Realimentación de estados.

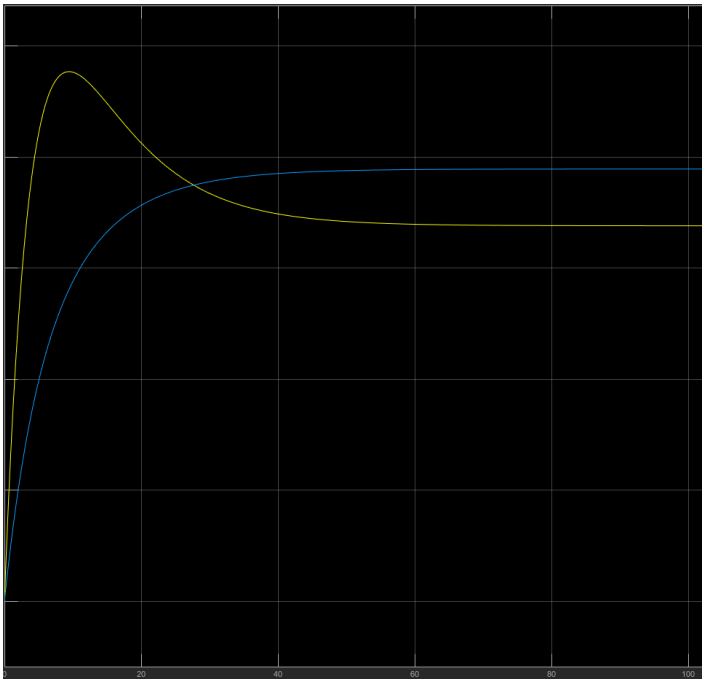


Figura 45. Señal con Realimentación de estados

Capítulo VI. Bibliografía

[1] Manual de Factory I/O.
https://docs.factoryio.com/manual/

[2] Descripción de estaciones y elementos Factory I/O.
https://docs.factoryio.com/manual/parts/stations/

[3] Descripción acerca de la ToolBox Modbus Explorer
https://www.mathworks.com/help/instrument/configure-a-connection-in-the-modbus-explorer.html

[4] Información acerca de Modbus Comunication en Matlab.
https://www.mathworks.com/help/instrument/modbus-communication.html

[5] Historia e información sobre el desarrollo y funcionamiento de Modbus
https://en.wikipedia.org/wiki/Modbus

[6] Información acerca de configuración Modbus.
https://modbus.org/

[7] Diapositivas de la asignatura de Ingeniería de Control, Grado en Ingeniería de Tecnologías Industriales 2018-2019.

[8] SFS Santiagos Francos S.L., Mezcladoras en la Industria y su funcionamiento
http://www.santiagofrancos.es/sfs.php

Anexo I: Modelado Matemático

En este anexo vamos a explicar todos aquellos procedimientos que hemos llevado a cabo para la obtención del modelo en el que nos apoyaremos para la obtención de los controladores.

En la siguiente figura mostramos el sistema el cual deseamos modelizar, esta modelización nos permitirá un control de caudal de salida de la mezcladora, así como su concentración, y comprobaremos si también fuese posible un control de la altura de esta. Los sensores de los que dispondremos son un sensor de caudal (Kq), sensor de altura (Kh) y sensor de concentración (Kc).

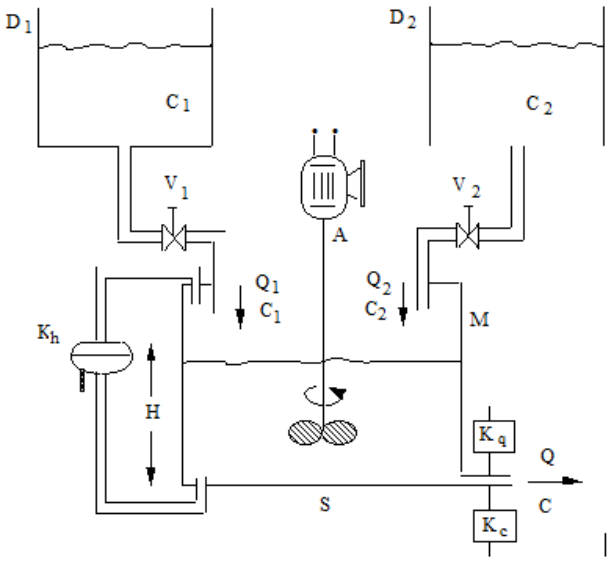


Figura 46. Modelo Mezcladora

Inicialmente deberemos conocer las constantes del sistema, estas son proporcionadas por Factory I/O, ya que disponemos de todos los datos referido al depósito en la página oficial del software ;(https://docs.factoryio.com/manual/parts/stations/).

Tank

Liquid tank including two control valves and a capacitive level sensor which can be used to control the flow of liquid in and out of the tank. Control valves are controlled by pneumatic actuators that can be positioned with signals between 0 and 10 V. Note that capacitive sensors can also be used to detect liquid levels. The liquid tank is primarily intended to be used for level and flow control using a PID.

- Height: 3 m
- Diameter: 2 m
- Discharge pipe radius: 0.125 m
- Max. input flow: 0.25 m³/s
- Max. output flow: 0.3543 m³/s
- Capacitive sensors can detect liquid

Por lo que obtenemos:

- $A = \pi * r^2 = \pi * 1^2 = \pi\, m^2$
- $Kt = Constante\ de\ salida\ libre\ del\ fluido$

  $Q = Kt * \sqrt{H} \rightarrow Q_{max} = Kt * \sqrt{H_{max}}$

  $0.3543\ {}^{m^3}/_{s} = Kt * \sqrt{3}\, m^{1/2}$

  $Kt = \dfrac{0.3543\ \frac{m^3}{s}}{\sqrt{3}\ m^{\frac{1}{2}}} = 0.204\ \dfrac{m^{\frac{5}{2}}}{s}$

Además, nos indica que las señales de los sensores tienen un rango de 0 a 10V por lo que podemos calcular:

- Vq=Kq*Q→ $Vq_{max} = Kq * Q_{max} \rightarrow Kq = \frac{Vq_{max}}{Q_{max}}$

  $Kq = \dfrac{10\ V}{0.3543\ \frac{m^3}{s}} = 28.224\ \dfrac{V}{{}^{m^3}/_{s}}$

- $Vh = Kh * H$

  $Kq = \dfrac{10\ V}{3\ m} = 3.33\ \dfrac{V}{m}$

- $Vc = Kc * C$

  En este punto debemos comentar que en nuestro modelo las concentraciones de los depósitos de materias primas ya están definidas con C1= 1 $\frac{\text{kmol}}{{}^{m^3}/_{s}}$ y C2= 2$\frac{\text{kmol}}{{}^{m^3}/_{s}}$, por lo que nuestro sensor solo deberá medir un rango de 1 Kmol

  Comentado esto sacamos que;

  $Kq = \dfrac{10\ V}{1\ \frac{\text{kmol}}{{}^{m^3}/_{s}}} = 10\ \dfrac{V}{\frac{\text{kmol}}{{}^{m^3}/_{s}}}$

Una vez hemos obtenido todas las constantes requeridas procederemos al modelizado.

Primero comentaremos las ecuaciones que rigen el comportamiento del sistema:

Por parte del caudal de salida, este se relaciona con la altura:

$$\boldsymbol{Q = K_t\sqrt{H}}$$

La variación de volumen el depósito está relacionada con el caudal que entra y sale del depósito:

$$\boldsymbol{\frac{dV}{dt} = \frac{d}{dt}(SH) = S\frac{dH}{dt} = Q_1 + Q_2 - Q}$$

En lo referido a la concentración del flujo a la salida, tendremos que los caudales de entrada por sus concentraciones menos el caudal de salida por su concentración se debe equilibrar con la variación de volumen por la concentración del depósito:

$$\boldsymbol{\frac{d}{dt}(VC) = \frac{d}{dt}(SHC) = S\frac{d}{dt}(HC) = Q_1C_1 + Q_2C_2 - QC}$$

De estas ecuaciones la correspondiente al caudal a la salida es una ecuación no lineal, por lo que antes de obtener las ecuaciones de estado deberemos linealizar el resto en torno a un punto de equilibrio. Para ello, la forma general de linealizar una ecuación es:

$$\begin{aligned}\boldsymbol{y = f(x_1, x_2, \dots x_n)} \\ &\boldsymbol{\approx f(x_{10}, x_{20}, \dots, x_{n0}) + \left.\frac{\partial f}{\partial x_1}\right|_{(x_{10},x_{20},\dots,x_{n0})}(x_1 - x_{10}) + \left.\frac{\partial f}{\partial x_2}\right|_{(x_{10},x_{20},\dots,x_{n0})}(x_2 - x_{20})} \\ &\boldsymbol{+ \cdots + \left.\frac{\partial f}{\partial x_n}\right|_{(x_{10},x_{20},\dots,x_{n0})}(x_n - x_{n0})}\end{aligned}$$

Aplicando esto a las ecuaciones no lineales, tenemos, para el caudal de salida en relación con la altura del depósito:

$$\boldsymbol{Q = K_t\sqrt{H} \approx K_t\left(\sqrt{H_0} + \frac{1}{2\sqrt{H_0}}(H - H_0)\right) = K_t\sqrt{H_0} + \frac{K_t}{2\sqrt{H_0}}(H - H_0) = Q_0 + \frac{K_t}{2\sqrt{H_0}}(H - H_0)}$$

La expresión que define el equilibrio de concentraciones contiene varios términos no lineales, por lo que hemos de linealizar cada uno de ellos por separado para dar lugar a la ecuación completa:

$$S\frac{d}{dt}(HC) = SH\frac{dC}{dt} + SC\frac{dH}{dt} = Q_1C_1 + Q_2C_2 - QC \rightarrow$$

$$\begin{cases} SH\dfrac{dC}{dt} \approx SH_0\dfrac{dC}{dt}\Big|_0 + S\dfrac{dC}{dt}\Big|_0 (H - H_0) + SH_0\left(\dfrac{dC}{dt} - \dfrac{dC}{dt}\Big|_0\right) = S(H - H_0)\dfrac{dC}{dt}\Big|_0 + SH_0\dfrac{dC}{dt} \\ SC\dfrac{dH}{dt} \approx SC_0\dfrac{dH}{dt}\Big|_0 + S\dfrac{dH}{dt}\Big|_0 (C - C_0) + SC_0\left(\dfrac{dH}{dt} - \dfrac{dH}{dt}\Big|_0\right) = S(C - C_0)\dfrac{dH}{dt}\Big|_0 + SC_0\dfrac{dH}{dt} \end{cases}$$

$$QC \approx Q_0C_0 + C_0(Q - Q_0) + Q_0(C - C_0)$$

$$\rightarrow S(H - H_0)\frac{dC}{dt}\Big|_0 + SH_0\frac{dC}{dt} + S(C - C_0)\frac{dH}{dt}\Big|_0 + SC_0\frac{dH}{dt}$$

$$= Q_1C_1 + Q_2C_2 - \left(Q_0C_0 + C_0(Q - Q_0) + Q_0(C - C_0)\right)$$

Por otra parte, en el punto de equilibrio se verifica que:

$$S\frac{dH}{dt}\Big|_0 = 0 = Q_{10} + Q_{20} - Q_0$$

$$SH_0\frac{dC}{dt}\Big|_0 + SC_0\frac{dH}{dt}\Big|_0 = 0 = Q_{10}C_1 + Q_{20}C_2 - Q_0C_0$$

Y a partir de estas dos ecuaciones se puede establecer:

$$\frac{dC}{dt}\Big|_0 = -\frac{C_0}{H_0}\frac{dH}{dt}\Big|_0 = 0$$

Además, deberemos realizar los siguientes cambios de variables:

$$h = H - H_0, \qquad q = Q - Q_0, \qquad q_1 = Q_1 - Q_{10}, \qquad q_2 = Q_2 - Q_{20}, \qquad c = C - C_0$$

y reorganizar las ecuaciones, por lo que quedara para la altura del deposito

$$\boldsymbol{Q - Q_0 = q = \frac{K_t}{2\sqrt{H_0}}(H - H_0) = \frac{K_t}{2\sqrt{H_0}}h}$$

Para el equilibrio de caudales:

$$S\frac{dH}{dt} = S\frac{d(h + H_0)}{dt} = S\frac{dh}{dt} = (q_1 + Q_{10}) + (q_2 + Q_{20}) - (q + Q_0) = q_1 + q_2 - q + Q_{10} + Q_{20} - Q_0 = q_1 + q_2 - q$$

Y para el equilibrio de caudales:

$$\begin{aligned} S(H - H_0)\frac{dC}{dt}\Big|_0 + SH_0\frac{dC}{dt} + S(C - C_0)\frac{dH}{dt}\Big|_0 + SC_0\frac{dH}{dt} &= SH_0\frac{d(c + C_0)}{dt} + SC_0\frac{d(h + H_0)}{dt} \\ = SH_0\frac{dc}{dt} + SC_0\frac{dh}{dt} &= Q_1C_1 + Q_2C_2 - \big(Q_0C_0 + C_0(Q - Q_0) + Q_0(C - C_0)\big) \\ = Q_1C_1 + Q_2C_2 - \big(Q_{10}C_1 + Q_{20}C_2 + C_0(Q - Q_0) + Q_0(C - C_0)\big) \\ = (Q_1 - Q_{10})C_1 + (Q_2 - Q_{20})C_2 - C_0(Q - Q_0) - Q_0(C - C_0) \\ = q_1C_1 + q_2C_2 - C_0q - Q_0c \end{aligned}$$

Quedando finalmente las siguientes ecuaciones linealizadas:

$$S\frac{dh}{dt} = q_1 + q_2 - q$$

$$q = \frac{K_t}{2\sqrt{H_0}}h$$

$$SC_0\frac{dh}{dt} + SH_0\frac{dc}{dt} = q_1C_1 + q_2C_2 - qC_0 - cQ_0$$

Esto nos permitirá construir el¡¡las ecuaciones de estado, las cuales tienen la siguiente forma:

$$\dot{\mathrm{x}} = \mathrm{Ax} + \mathrm{Bu}$$

$$\mathrm{y} = \mathrm{Cx} + \mathrm{Du}$$

Despejando de las ecuaciones obtenemos:

$$\frac{dq}{dt} = \dot{q} = -\frac{K_t}{2S\sqrt{H_0}}q + \frac{K_t}{2S\sqrt{H_0}}q_1 + \frac{K_t}{2S\sqrt{H_0}}q_2$$

$$\frac{dc}{dt} = \dot{c} = -\frac{Q_0}{SH_0}c + \frac{C_1 - C_0}{SH_0}q_1 + \frac{C_2 - C_0}{SH_0}q_2$$

$$\frac{dh}{dt} = \dot{h} = -\frac{1}{S}q + \frac{1}{S}q_1 + \frac{1}{S}q_2$$

Quedando la forma matricial:

$$\begin{pmatrix}\dot{q}\\ \dot{c}\\ \dot{h}\end{pmatrix} = \begin{pmatrix} -\frac{K_t}{2S\sqrt{H_0}} & 0 & 0 \\ 0 & -\frac{Q_0}{SH_0} & 0 \\ -\frac{1}{S} & 0 & 0 \end{pmatrix}\begin{pmatrix}q\\ c\\ h\end{pmatrix} + \begin{pmatrix} \frac{K_t}{2S\sqrt{H_0}} & \frac{K_t}{2S\sqrt{H_0}} \\ \frac{C_1 - C_0}{SH_0} & \frac{C_2 - C_0}{SH_0} \\ \frac{1}{S} & \frac{1}{S} \end{pmatrix}\begin{pmatrix}q_1\\ q_2\end{pmatrix} = \mathrm{A}\begin{pmatrix}q\\ c\\ h\end{pmatrix} + \mathrm{B}\begin{pmatrix}q_1\\ q_2\end{pmatrix}$$

Donde las salidas del sistema se corresponden con:

- Vq=Kq∗Q
- $Vh = Kh * H$
- $Vc = Kc * C$

$$\begin{pmatrix}V_q\\ V_c\\ V_h\end{pmatrix} = \begin{pmatrix}\mathrm{Kq} & 0 & 0\\ 0 & Kc & 0\\ 0 & 0 & Kh\end{pmatrix}\begin{pmatrix}q\\ c\\ h\end{pmatrix} + \begin{pmatrix}0 & 0\\ 0 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}q_1\\ q_2\end{pmatrix} = \mathrm{C}\begin{pmatrix}q\\ c\\ h\end{pmatrix} + \mathrm{D}\begin{pmatrix}q_1\\ q_2\end{pmatrix}$$

El punto de equilibrio que vamos a utilizar para poder obtener el modelo matemático es:

$$H_0 = 1.5\ m, \qquad Q_0 = 0.25\ \frac{m^3}{s}, \qquad Q_{10} = 0.125\ \frac{m^3}{s}, \qquad Q_{20} = 0.125\ \frac{m^3}{s}, \qquad C_0 = 1.5\ \frac{\mathrm{kmol}}{m^3/s}$$

Sustituyendo los valores, podemos obtener las matrices A, B, C y D, este procedimiento y los siguiente los realizaremos con la ayuda de Matlab.

```
%Datos
Kt=0.204;      %m^(5/2)/s
S=3.1416;      %m2
Ho=1.5;        %m
Qo=0.25;       %m3/s
Co=1.5;        %Kmol/m3
C1=1;          %Kmol/m3
C2=2;          %Kmol/m3
Kq = 28.2246; %V/m3s
Kc = 10;       %V/ Kmol/m3
Kh = 3.33;     %V/m

%Matrices
A = [-(Kt/(2*S*(Ho^0.5)))     0           0;
    0                      -(Qo/(S*Ho))   0;
    -1/S                   0           0]
B = [(Kt/(2*S*(Ho^0.5)))      (Kt/(2*S*(Ho^0.5)));
    (C1-Co)/(S*Ho)            (C2-Co)/(S*Ho);
    1/S                        1/S]
C = [Kq 0    0;
    0   Kc   0;
    0   0    Kh]
D = [0  0;
    0   0;
    0   0]
```

Obteniendo:

```
A =                                    B =

   -0.0265         0         0          0.0265    0.0265
         0   -0.0531         0         -0.1061    0.1061
   -0.3183         0         0          0.3183    0.3183


C =                                    D =

   28.2246         0         0               0         0
         0   10.0000         0               0         0
         0         0    3.3300               0         0
```

Una vez obtenidas las matrices procederemos a comprobar que efectivamente podemos trabajar con este modelo de estados, esto lo comprobaremos mediante la controlabilidad y observabilidad.

La controlabilidad de un sistema se relaciona con la viabilidad de llevar el estado del sistema desde cualquier estado inicial a cualquier estado final, por medio de una señal de entrada apropiada, en un tiempo finito. Para ello, definimos la matriz de controlabilidad:

$$\zeta = (\mathrm{B}\ \mathrm{AB}\ \ldots\ \mathrm{A}^{n-1}\mathrm{B})$$

La observabilidad del sistema se relaciona con la viabilidad de reconstruir el estado de un sistema observando la salida de este durante un intervalo de tiempo finito. Para ello, se define la matriz de observabilidad:

$$\theta = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{pmatrix}$$

Para ambas matrices n es el número de variables del estado. Y para que ambas condiciones se cumplan el rango de las correspondientes matrices ha de ser igual a n. Por lo que en nuestro caso su rango deberá ser 3.

Comprobamos esto con Matlab:

```
% CONTROLABILIDAD
Zeta = [B A*B A^2*B];
rango_Zeta = rank(Zeta);

% OBSERVABILIDAD
Tita = [C; C*A; C*A^2];
rango_Tita = rank(Tita);
```

```
rango_Zeta =
     2

rango_Tita =
     3
```

Es nos indica que, aunque el sistema es observable no es controlable. Esto es debido a que el caudal de salida y la altura del depósito son proporcionales, por lo que nunca podremos fijar valores independientes a cada uno.

Esto nos obliga a cambiar nuestras ecuaciones de estado quedando:

$$\begin{pmatrix} \dot{q} \\ c \end{pmatrix} = \begin{pmatrix} -\frac{K_t}{2S\sqrt{H_0}} & 0 \\ 0 & -\frac{Q_0}{SH_0} \end{pmatrix} \begin{pmatrix} q \\ c \end{pmatrix} + \begin{pmatrix} \frac{K_t}{2S\sqrt{H_0}} & \frac{K_t}{2S\sqrt{H_0}} \\ \frac{C_1 - C_0}{SH_0} & \frac{C_2 - C_0}{SH_0} \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = A \begin{pmatrix} q \\ c \end{pmatrix} + B \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}$$

$$\begin{pmatrix} V_q \\ V_c \end{pmatrix} = \begin{pmatrix} Kq & 0 \\ 0 & Kc \end{pmatrix} \begin{pmatrix} q \\ c \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = C \begin{pmatrix} q \\ c \\ h \end{pmatrix} + D \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}$$

Adicionalmente hay que comentar que nosotros no podemos directamente introducir el caudal, sino que controlamos la válvula, por lo que deberemos añadir la relación

$$q_1 = \%_{v1} * Q$$

$$q_2 = \%_{v2} * Q$$

Quedando el siguiente sistema:

$$\begin{pmatrix}\dot{q}\\ c\end{pmatrix} = \begin{pmatrix}-\frac{K_t}{2S\sqrt{H_0}} & 0\\ 0 & -\frac{Q_0}{SH_0}\end{pmatrix}\begin{pmatrix}q\\ c\end{pmatrix} + \begin{pmatrix}\frac{K_t}{2S\sqrt{H_0}} & \frac{K_t}{2S\sqrt{H_0}}\\ \frac{C_1 - C_0}{SH_0} & \frac{C_2 - C_0}{SH_0}\end{pmatrix} * \begin{pmatrix}Q & 0\\ 0 & Q\end{pmatrix}\begin{pmatrix}\%_{v1}\\ \%_{v2}\end{pmatrix} = \mathrm{A}\begin{pmatrix}q\\ c\end{pmatrix} + \mathrm{B}\begin{pmatrix}\%_{v1}\\ \%_{v2}\end{pmatrix}$$

$$\begin{pmatrix}V_q\\ V_c\end{pmatrix} = \begin{pmatrix}\mathrm{Kq} & 0\\ 0 & Kc\end{pmatrix}\begin{pmatrix}q\\ c\end{pmatrix} + \begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}\%_{v1}\\ \%_{v2}\end{pmatrix} = \mathrm{C}\begin{pmatrix}q\\ c\\ h\end{pmatrix} + \mathrm{D}\begin{pmatrix}\%_{v1}\\ \%_{v2}\end{pmatrix}$$

```
A =                         B =

   -0.0265         0         0.0089    0.0089
         0   -0.0531        -0.0357    0.0357


C =                         D =

   28.2246         0              0         0
         0   10.0000              0         0
```

Comprobamos Controlabilidad y observabilidad

```
%Comprobar controlabilidad y observabilidad

Zeta = [B A*B];
rango_Zeta = rank(Zeta);

Tita = [C; C*A];
rango_Tita = rank(Tita);
```

```
rango_Zeta =

     2

rango_Tita =

     2
```

Una vez comprobado que podemos trabajar sobre este modelo, calculamos las funciones de trasferencia resultando:

```
%Sacamos funciones de transferencia

[num1, den1] = ss2tf(A,B,C,D,1);
[num2, den2] = ss2tf(A,B,C,D,2);

G_11 = tf(num1(1,:),den1);
G_12 = tf(num2(1,:),den2);
G_21 = tf(num1(2,:),den1);
G_22 = tf(num2(2,:),den2);

G = [G_11 G_12;
     G_21 G_22]%Matriz funciones de transferencia
```

Resultando:

```
G =

  From input 1 to output...
           0.2514 s + 0.01334
   1:  --------------------------
       s^2 + 0.07956 s + 0.001406

          -0.3565 s - 0.009451
   2:  --------------------------
       s^2 + 0.07956 s + 0.001406

  From input 2 to output...
           0.2514 s + 0.01334
   1:  --------------------------
       s^2 + 0.07956 s + 0.001406

          0.3565 s + 0.009451
   2:  --------------------------
       s^2 + 0.07956 s + 0.001406
```

Como vemos las funciones no están simplificadas, por lo que podemos aplicar la anulación de polos y ceros obteniendo la G(s) final:

$$\boldsymbol{G(s)} = \begin{pmatrix} \dfrac{\mathbf{0.2514}}{\boldsymbol{s}+\mathbf{0.0265}} & \dfrac{\mathbf{0.2514}}{\boldsymbol{s}+\mathbf{0.0265}} \\ \dfrac{\mathbf{-0.3565}}{\boldsymbol{s}+\mathbf{0.0531}} & \dfrac{\mathbf{0.3565}}{\boldsymbol{s}+\mathbf{0.0531}} \end{pmatrix}$$

El control de este modelo se ha de realizar mediante computación, por lo que para poder llevar a cabo el diseño de las leyes de control el sistema se deberá discretizar.

El discretizado lo realizaremos considerando un tiempo de muestreo de Tm=0.1 segundos.

```
%Discretizamos el sistema

T=0.1;
[F,G] = c2d(A,B,T);
```

```
F =                    G =

    0.9974         0       0.0009    0.0009
         0    0.9947      -0.0036    0.0036
```

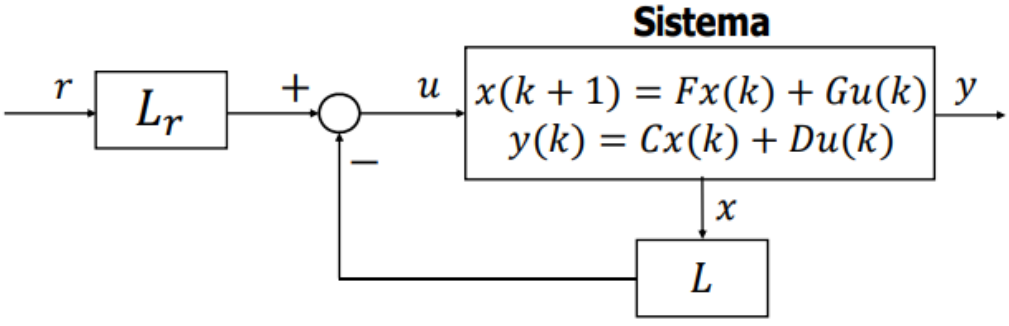
Y comprobamos que al discretizar se siguen manteniendo las condiciones de controlabilidad y observabilidad en el sistema. Esto lo comprobamos de la misma forma que en el modelo continuo.

```
ZetaD = [G F*G];
rango_ZetaD = rank(ZetaD);

TitaD = [C; C*F];
rango_TitaD = rank(TitaD);
```

```
rango_ZetaD =

     2

rango_TitaD =

     2
```

Donde sique siendo un sistema tanto controlable como observable.

El siguiente paso en el modelado matemático es la obtención de la matriz L y Lr, las cual nos serviría para llevar a cabo un control de todo el sistema por realimentación del estado



Primero deberemos calcular la matriz L, en función de la posición deseada de los polos del bucle cerrado.

$$|sI - A + B * L| = 0$$

$$|\boldsymbol{sI - A + BL}| = \left|\begin{pmatrix} \boldsymbol{s} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{s} \end{pmatrix} - \begin{pmatrix} \boldsymbol{-0.0265} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{-0.0531} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0.0089} & \boldsymbol{0.0089} \\ \boldsymbol{-0.0357} & \boldsymbol{0.0357} \end{pmatrix} * \begin{pmatrix} \boldsymbol{l_1} & \boldsymbol{l_2} \\ \boldsymbol{l_3} & \boldsymbol{l_4} \end{pmatrix}\right|$$

$$= \left|\begin{pmatrix} \boldsymbol{s + 0.0265 + 0.0265 * l_1 + 0.0265 * l_3} & \boldsymbol{0.0265 * l_2 + 0.0265 * l_4} \\ \boldsymbol{-0.106 * l_1 + 0.106 * l_3} & \boldsymbol{s + 0.0531 - 0.106 * l_2 + 0.106 * l_4} \end{pmatrix}\right|$$

$$\begin{aligned} = \boldsymbol{0.00047 * l_1 - 0.00094 * l_2 - 0.000635 * l_2 * l_3 + 0.00047 * l_3 + 0.000635 * l_1} \\ \boldsymbol{* l_4 + .00094 * l_4 + s^2 + s} \\ \boldsymbol{* (0.0089 * l_1 - 0.0357 * l_2 + 0.0089 * l_3 + 0.0357 * l_4 + 0.0796)} \\ \boldsymbol{+ 0.0014} \end{aligned}$$

Sacamos los polos del sistema:

$$|sI - A| = (s + 0.0265) * (s + 0.0531)$$

$$s = -0.0265 \quad ; \quad s = -0.0531$$

Tenemos el siguiente sistema de ecuaciones para los dos polos:

$$\boldsymbol{D(s) = 0}$$

$$\left.\boldsymbol{\frac{dD(s)}{dt}}\right|_{\boldsymbol{s}} = \boldsymbol{0}$$

Resolvemos este sistema de ecuaciones de 4 incógnitas y 4 ecuaciones gracias a la ayuda del programa EES (Enginner equation solver), Obteniendo la siguiente matriz L

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{0.3163} & \boldsymbol{0.3742} \\ \boldsymbol{0.3551} & \boldsymbol{0.6155} \end{pmatrix}$$

Y a continuación sacamos la matriz Lr ya que queremos que:

$$\lim_{z \to 1} G_{(z)} = 1 \rightarrow L_r = ((C - DL)(I - F + GL)^{-1}G + D)^{-1}$$

$$\boldsymbol{L_r} = \begin{pmatrix} \boldsymbol{0.0307} & \boldsymbol{0.0176} \\ \boldsymbol{0.0282} & \boldsymbol{0.0531} \end{pmatrix}$$

Anexo II: Código Controladores

En este anexo vamos a desarrollar el proceso de obtención del código que necesitaremos para aplicar las acciones de los controladores en el programa de Matlab.
Sabiendo que U(z) es la acciona aplicar por el controlador y E(z) = Referencia – Salida;
Donde la referencia es el caudal de salida que nosotros hemos fijado en la mezcladora
Y la salida es el caudal real que está saliendo.

Proporcional

$$\mathbf{C(z) = Kc}$$

$$\mathbf{\frac{U(z)}{E(z)} = Kc}$$

$$\mathbf{U(z) = Kc * E(z)}$$

Proporcional Integral

$$\mathbf{C(z) = Kc\left(\frac{z-a}{z-1}\right)}$$

$$\mathbf{\frac{U(z)}{E(z)} = Kc\left(\frac{z-a}{z-1}\right)}$$

$$\mathbf{U(z) * (z-1) = Kc * (z-a) * E(z)}$$

$$\mathbf{U(z) * \left(1-z^{-1}\right) = Kc * \left(1-a*z^{-1}\right) * E(z)}$$

$$\mathbf{U_k - U_{k-1} = Kc * E_k - Kc * a * E_{k-1}}$$

$$\mathbf{U_k = U_{k-1} + Kc * E_k - Kc * a * E_{k-1}}$$

Proporcional Integral y Derivativo (PID)

$$\boldsymbol{C(z) = Kc(\ 1 + + I * T * \frac{1}{z-1} + \frac{D}{T} * \frac{z-1}{z})}$$

$$\boldsymbol{P = Kc}$$
$$\boldsymbol{Ki = \ I * T}$$
$$\boldsymbol{Kd = \ D/T}$$

$$\boldsymbol{C(z) = P(\ 1 + + Ki * \frac{1}{z-1} + \mathbf{Kd} * \frac{\mathbf{z-1}}{\mathbf{z}})}$$

$$\mathbf{= P\left(\frac{(z-1)z\ +\ Ki * z + Kd * (z-1)^2}{(z-1)z}\right) =}$$

$$\mathbf{= P\left(\frac{z^2 - z + Ki * z + Kd * z^2 - 2 * Kd * z + Kd}{(z-1)z}\right) =}$$

$$\mathbf{= P\left(\frac{(\ Kd + 1) * z^2 + (Ki - 2 * Kd - 1) * z + Kd}{(z-1)z}\right)}$$

$$\mathbf{\left(z^2 - z\right) * U_k = E_k * P * ((\ Kd + 1) * z^2 + (Ki - 2 * Kd - 1) * z + Kd)}$$

$$\mathbf{\left(1 - z^{-1}\right) * U_k = E_k * P * ((\ Kd + 1) + (Ki - 2 * Kd - 1) * z^{-1} + Kd * z^{-2})}$$

$$\mathbf{U_k - \ U_{k-1}\ = P * (\ (\ Kd + 1) * E_k + (Ki - 2 * Kd - 1) * E_{k-1} + Kd * E_{k-2})}$$

$$\mathbf{U_k = \ U_{k-1} + \ P * (\ (\ Kd + 1) * E_k + (Ki - 2 * Kd - 1) * E_{k-1} + Kd * E_{k-2})}$$

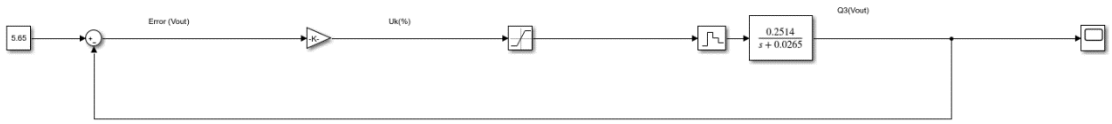
Anexo III: Obtención de Controladores

Para la obtención de los controladores requerimos de la función de transferencia de la base ya discretizada, junto con todos los elementos que conforman los diferentes cambios de referencia.

El cálculo de los controladores proporcional y proporcional integral se realizará mediante un desarrollo matemático, expuesto a continuación:

Proporcional

Para la obtención de este controlador debemos tener en cuenta que nuestro sistema tiene un límite de saturación, ya que el tanque tiene una entrada de caudal limitada. Debido a esto aumentar el valor de K al máximo aceptable no supone una mejora notable en el controlador, ya que hay un tiempo mínimo de llegado del depósito.



Para la obtención de un rango de estabilidad deberos sacar la Función en bucle cerrado del sistema y calcular entre que valore de K el sistema es estable.

$$F_{BC} = \frac{2K * BoG(z)}{1 + 2K * BoG(z)} = \frac{2K * \frac{0.2514}{z - 0.9974}}{1 + 2K * \frac{0.2514}{z - 0.9974}}$$

$$= \frac{2K * 0.2514}{(z - 0.994) + 2K * 0.2514} = \frac{0.2028 * K}{z + (0.5028K - 0.9974)}$$

Para que se cumpla el criterio de estabilidad de Routh: $|(0.5028K - 0.9974)| < 1$

Por lo que nos queda: $-0.005 < K < 3.97$

Por otro lado, sabemos que el sistema saturará cuando Uk > 1, lo que se da cuando

$$U_{Kmax} = K * Error_{max} \rightarrow K \leq \frac{U_{Kmax}}{Error_{max}} = \frac{1}{7.056} = 0.1417$$

Por lo que si K > 0.1417, el tiempo de respuesta se ve condicionado por el límite físico del sistema, Por esta razón elegimos un valor medio para K, siendo K = 2.

DeadBeat

Aplicamos especificaciones características de este controlador:

$$F_{BC} = \frac{K}{z}$$

Ep = 0, implicara $F_{BC}|_{(z=1)} = 1 \rightarrow \frac{K}{1} = 1 \rightarrow K = 1$

$$F_{BC} = \frac{1}{z}$$

$$F_{BC} = \frac{2R(z) * BoG(z)}{1 + 2R(z) * BoG(z)}$$

$$(1 + 2 * R(z) * BoG(z)) * F_{BC} = 2 * R(z) * BoG(z)$$
$$F_{BC} + 2 * R(z) * BoG(z) * F_{BC} = 2 * R(z) * BoG(z)$$
$$F_{BC} = -2 * R(z) * BoG(z) * F_{BC} + 2 * R(z) * BoG(z)$$
$$F_{BC} = R(z) * (2 * BoG(z) * (1 - F_{BC})$$

$$R(z) = \frac{F_{BC}}{1 - F_{BC}} * \frac{1}{2 * BoG(z)}$$

$$R(z) = \frac{\frac{1}{z}}{1 - \frac{1}{z}} * \frac{1}{2 * \frac{0.2514}{z - 0.9974}} = \frac{1}{z - 1} * \frac{z - 0.9974}{0.5028} = 1.988 * \frac{z - 0.9974}{z - 1}$$

Proporcional e Integral (PI)

$$C(s) = Kd * \frac{1 + \tau s}{s}$$

$$G(s) = \frac{K}{1 + \tau s} = \frac{0.2514}{s + 0.0265} = \frac{9.486}{1 + 37.73s}$$

$$F_{BA} = C(s) * G(s) = 2 * Kd * \frac{1 + \tau s}{s} * \frac{K}{1 + \tau s} = \frac{2 * K * Kd}{s}$$

$$F_{BC} = \frac{C(s) * G(s)}{1 + C(s) * G(s)} = \frac{2 * K * Kd}{s + 2 * K * Kd} = \frac{1}{1 + \frac{1}{2 * K * Kd} s} =$$

Fijamos especificaciones

$$tr = \frac{tr_{BA}}{10} = 20\ segundos$$

$$tr = 3 * \tau = 3 * \frac{1}{2 * K * Kd}$$

$$Kd = \ 3 * \frac{1}{2 * K * tr} = 0.008$$

$$C(s) = 0.008 * \frac{1 + 37.73s}{s}$$

Discretizamos mediante Tustin, con tiempo de muestreo T=0.1;

$$C(s) = 0.008 * \frac{1 + 37.73\frac{2}{0.1} * \frac{z-1}{z+1}}{\frac{2}{0.1} * \frac{z-1}{z+1}} = 0.008 * \frac{(z+1) + 754.6 * (z-1)}{(z-1)}$$

$$= 0.008 * \frac{755.6 * (z - 0.9974)}{(z-1)} = 0.3 * \frac{z - 0.9974}{z-1}$$

Proporcional, Integral y Derivador (PID)

Este controlador lo hemos calculado directamente con Matlab ya que el propio bloque de PID en simulink contiene la herramienta TUNE nos permite probar diferentes valores para las constantes y ajustar el controlar para que se adapte a nuestras especificaciones. El controlador final elegido es:

$$C(z) = 0.627 * (\ 1 + \ 0.035 * T * \frac{1}{z-1} + \frac{0.05}{T} * \frac{z-1}{z})$$

Donde T es el tiempo de muestreo, T= 0.1 seg.

Anexo IV: Implementación Factory I/O

Código Espacio de Estados

```
% Modelo controlable y observable
Kt=0.204;      %m^(5/2)/s
S=3.1416;      %m2
Ho=1.5;        %m
Qo=0.25;       %m3/s
Co=1.5;        %Kmol/m3
C1=1;          %Kmol/m3
C2=2;          %Kmol/m3
Kq = 28.2246; %V/m3s
Kc = 10;       %V/ Kmol/m3


 A = [-(Kt/(2*S*(Ho^0.5)))      0;
     0                          -(Qo/(S*Ho))];

 B1 = [(Kt/(2*S*(Ho^0.5)))       (Kt/(2*S*(Ho^0.5)));
       ((C1-Co)/(S*Ho))         ((C2-Co)/(S*Ho))];

 B2 = [0.336 0;
       0 0.336];

 B = B1*B2;

 C = [Kq 0;
     0   Kc];

 D = [0  0;
     0   0];

 I = [1  0;
      0  1];

%Comprobar controlabilidad y observabilidad

Zeta = [B A*B];
rango_Zeta = rank(Zeta);

Tita = [C; C*A];
rango_Tita = rank(Tita);

%Sacamos funciones de transferencia

[num1, den1] = ss2tf(A,B,C,D,1);
[num2, den2] = ss2tf(A,B,C,D,2);

G_11 = tf(num1(1,:),den1);
G_12 = tf(num2(1,:),den2);
G_21 = tf(num1(2,:),den1);
G_22 = tf(num2(2,:),den2);

G = [G_11 G_12;
     G_21 G_22];%Matriz funciones de transferencia

p = roots(den1); %Raices de la ecuacion
```

```
%Con estos pasos podemos simplificar la G(s) y escribirla mejor
%LAS PODEMOS INTRODUCIR EN SIMULINK

z_11 = roots(num1(1,:));
z_12 = roots(num2(1,:));
z_21 = roots(num1(2,:));
z_22 = roots(num2(2,:));

k_11 = num1(1,2)/den1(1);
k_12 = num2(1,2)/den2(1);
k_21 = num1(2,2)/den1(1);
k_22 = num2(2,2)/den2(1);

%Discretizamos el sistema

T=0.1;
[F,G] = c2d(A,B,T);


%Volvemos a comprobar controlabilidad y observabilidad

ZetaD = [G F*G];
rango_ZetaD = rank(ZetaD);

TitaD = [C; C*F];
rango_TitaD = rank(TitaD);
```

Código Proporcional

```
%Programa P

    IPent = 'Ip:'; %Pido Ip como comando
    ip = input(IPent,'s');

    m = modbus('tcpip',ip,502); %Me conecto a la ip proporcionada

    disp('Conectado');

    Kin= 40;
    Kout=28.224;
    Hmin=900;
    Hmax=980;

    Conent = 'Setpoint(1000~2000 mol/m3):'; %Pido Setpoint de forma manual
    Con = input(Conent);
    write(m,'holdingregs',60,Con,'uint16'); %Muestro Setpoint_Concentración por↙
pantalla

    C1=1000; % mol/m3
    C2=2000; % mol/m3
    C3=Con;  % mol/m3

    % Q3*C3=Q1*C1+Q2*C2
    % Q3=Q1+Q2

    %Para poder hacer u control solo de concentracion deber fijar el
    %valor de cauda deseado

    Q3 = 'Caudal deseado de salida(0~250 l/s):';
    SetPoint_Q3_L = input(Q3);
    SetPoint_Q3_m3s= SetPoint_Q3_L/1000;
    SetPoint_Q3_Voltios = Kout*SetPoint_Q3_m3s;

    %Saco los valores de Q1 y Q2 con los valores asignados
    A = [C1 C2; 1 1];
    B = [C3*SetPoint_Q3_m3s; SetPoint_Q3_m3s ];
    S = inv(A)*B;
    Q1T = S(1,1);
    Q1_L = 1000*Q1T;
    Q1= round(Q1_L,0);

    Q2T = S(2,1);
    Q2_L = 1000*Q2T;
    Q2 = round(Q2_L,0);

    disp('Pulsar Stop para desconectar');

    %Aseguro valvulas de salida cerradas
    write(m,'holdingregs',44,0,'uint16');
    write(m,'holdingregs',45,0,'uint16');
    write(m,'holdingregs',46,0,'uint16');

    %Tanques de materia prima
    LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por↙
```

```
el sensor
    LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por↙
el sensor

    %Calculo altura equilibrio T3
    Level_Q = ((SetPoint_Q3_m3s)^2)/(((0.204)^2)*0.003);

    %Escribo Setpoint Altura para Caudal salida
    Set3 = round(Level_Q,0);
    write(m,'holdingregs',52,Set3,'uint16');

    %Escribo Setpoint Caudal salida
    write(m,'holdingregs',59,SetPoint_Q3_L ,'uint16');

    %Lleno los Tanques de materia prima
    while (LevelMeter_T1<=Hmax)&&(LevelMeter_T2<=Hmax)

        write(m,'holdingregs',41,1000,'uint16'); % hago que se llenen rapido hasta el↙
nivel deseado
        write(m,'holdingregs',42,1000,'uint16');

        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido↙
por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido↙
por el sensor

        write(m,'holdingregs',49,LevelMeter_T1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T1,'uint16');

        if LevelMeter_T1>=Hmax
            write(m,'holdingregs',41,0,'uint16');
        end
        if LevelMeter_T2>=Hmax
            write(m,'holdingregs',42,0,'uint16');
        end


    end % Lleno de depositos de materia prima

    %TANQUE 1
    SetPoint_Q1_L = Q1;
    SetPoint_Q1_m3s= SetPoint_Q1_L/1000;
    SetPoint_Q1_Voltios = Kout*SetPoint_Q1_m3s;

    %TANQUE 2
    SetPoint_Q2_L = Q2;
    SetPoint_Q2_m3s= SetPoint_Q2_L/1000;
    SetPoint_Q2_Voltios = Kout*SetPoint_Q2_m3s;

    %TANQUE 3
    %Calculo Nivel T3 y lo muestro
    LevelMeter_T3 = read(m,'inputregs',26,1,'uint16');
    write(m,'holdingregs',53,LevelMeter_T3,'uint16');

    D1 = 1000*(SetPoint_Q1_L/SetPoint_Q3_L);
```

```
    D2 = 1000*(SetPoint_Q2_L/SetPoint_Q3_L);

    %Comienzo la regulacion
    T=0.1;% Fijo tiempo de muestreo

    %Fijo valores controlador
    P = 2;

    write(m,'holdingregs',46,1000,'uint16');%Abro valvula salida T3 para que comience
regulacion

    while(1) %Regulacion
        tic
        write(m,'holdingregs',46,1000,'uint16');%Abro valvula salida T3 para que
comience regulacion
        %Control Tanques Materia Primas
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido
por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido
por el sensor

        if (LevelMeter_T1>Hmax)
            write(m,'holdingregs',41,0,'uint16');
        elseif(LevelMeter_T1<Hmin)
             write(m,'holdingregs',41,1000,'uint16');
        else
            Red1 = round(100*(Kin/Kout)*(SetPoint_Q1_Voltios),0);
            write(m,'holdingregs',41,Red1,'uint16');
        end

        if (LevelMeter_T2>Hmax)
            write(m,'holdingregs',42,0,'uint16');
        elseif(LevelMeter_T2<Hmin)
            write(m,'holdingregs',42,1000,'uint16');
        else
            Red2 = round(100*(Kin/Kout)*(SetPoint_Q2_Voltios),0);
            write(m,'holdingregs',42,Red2,'uint16');
        end

        %TANQUE 3

        %Calculo Nivel T3 y lo muestro
        LevelMeter_T3 = read(m,'inputregs',26,1,'uint16');
        write(m,'holdingregs',53,LevelMeter_T3,'uint16');

        Q_3_ref = SetPoint_Q3_Voltios; %Voltios que deberian de salir de T3
        Voltaje_caudal_salida_3 = read(m,'inputregs',23,1,'uint16');
        Q_3_real = Voltaje_caudal_salida_3/100;%Voltios que realmente salen de T3

        Ek = Q_3_ref - Q_3_real;
        Ek_1 = (SetPoint_Q1_L/SetPoint_Q3_L)*Ek ;
        Ek_2 = (SetPoint_Q2_L/SetPoint_Q3_L)*Ek;

        % esto es debido a que contamos con 2 controladores
        % independientes,por lo que si aplicasemos directamente el error Ek
```

```
        % estariamos aplicando el doble de error, ademas de aplicarle las
        % misma proporciones a cada uno, por lo que acabiran igualandose.
        %Ek = Q_3_ref - Q_3_real;
        %Ek = (Q_1_ref + Q_2_ref) - (Q_1_real + Q_2_real);
        %Ek = (Q_1_ref - Q_1_real) + (Q_2_ref - Q_2_real);
        %Ek = Ek_1 + Ek_2;

        %TANQUE 1

        %Aplico controlador Proporcional  al T1
        Q_1_ref = SetPoint_Q1_Voltios; %voltios que deberian de salir de T1 (Q1)
        Voltaje_caudal_salida_1 = read(m,'inputregs',21,1,'uint16');
        Q_1_real = Voltaje_caudal_salida_1/100; %Voltios que realmente salen de T1

        %Error_1 = Q_1_ref - Q_1_real; %en Vout

        %Q1_Voltios = %Valvula*Q_1_max_voltios
        % Lo que vamos a controlar el el porcentaje de valvula abierto

        %Valor P
        Uk_1 = P*Ek_1;
        % creamos metodo antiwindup
        if (Uk_1 > 1)
            Uk_1 = 1;
        elseif (Uk_1 < 0)
            Uk_1 = 0;
        end

        %Aplicamos valor %
        Qent1 = 1000*Uk_1; % paso valor referencia factory ya que en factory no es [0-↙
1] sino [0-1000]
        Porcentaje_V1 = round(Qent1,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',44,Porcentaje_V1,'uint16') % aplico la accion

        %Calculo caudal salida T1 y lo muestro
        Caudal_salida_1 = round(0.3543*Voltaje_caudal_salida_1,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',55,Caudal_salida_1,'uint16');

        %Calculo Nivel T1 y lo muestro
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16');
        write(m,'holdingregs',49,LevelMeter_T1,'uint16');


        %TANQUE 2

        %Aplico Proporcional
        Q_2_ref = SetPoint_Q2_Voltios; %Voltios que deberian de salir de T2
        Voltaje_caudal_salida_2 = read(m,'inputregs',22,1,'uint16');
        Q_2_real = Voltaje_caudal_salida_2/100; %Voltios que realmente salen de T2

        %Error_2 = Referencia_2 - Salida_2; %en Vout

        %Valor P
```

```
        Uk_2 = P*Ek_2;
        % creamos metodo antiwindup
        if (Uk_2 > 1)
            Uk_2 = 1;
        elseif (Uk_2 < 0)
            Uk_2 = 0;
        end
        %Aplicamos valor %
        Qent2 = 1000*Uk_2; % paso valor referencia factory
        Porcentaje_V2 = round(Qent2,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',45,Porcentaje_V2,'uint16') % aplico la accion

        %Calculo caudal salida T2 y lo muestro
        Caudal_salida_2 = round(0.3543*Voltaje_caudal_salida_2,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',56,Caudal_salida_2,'uint16');

        %Calculo Nivel T2 y lo muestro
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T2,'uint16');


        %Caudal que entra en T3 procedente de T1 y T2
        QEntrada_T3= 0.3543*Voltaje_caudal_salida_1 + 0.3543*Voltaje_caudal_salida_2;
        Q = round(QEntrada_T3,0);
        write(m,'holdingregs',57,Q,'uint16');% Muestro por pantalla lo que entra

        Q3_aplic =(Kin/Kout)*(Voltaje_caudal_salida_1 + Voltaje_caudal_salida_2);
        Q3_apli_redondeado = round(Q3_aplic,0);
        write(m,'holdingregs',43,Q3_apli_redondeado,'uint16'); % aplico la accion

        %Caudal que realmente sale por T3
        Caudal_salida_3 = round(0.3543*Voltaje_caudal_salida_3,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',58,Caudal_salida_3,'uint16');

        %Metodo de parado del programa
        Stop = read(m,'inputs',8,1);

        if (Stop==0) % si pulsas S se dejan de leer datos del servidor y se inicializa↙
a cero para el siguiente valor
            disp('Stop Pulsado');
            %Llenados a 0
                write(m,'holdingregs',41,0,'uint16');
                write(m,'holdingregs',42,0,'uint16');
                write(m,'holdingregs',43,0,'uint16');
            %Vaciados al Maximo
                write(m,'holdingregs',44,1000,'uint16');
                write(m,'holdingregs',45,1000,'uint16');
                write(m,'holdingregs',46,1000,'uint16');
            %Setpoint a 0
                write(m,'holdingregs',48,0,'uint16');
                write(m,'holdingregs',50,0,'uint16');
                write(m,'holdingregs',52,0,'uint16');
```

```
            write(m,'holdingregs',59,0,'uint16');
            write(m,'holdingregs',60,0,'uint16');
        %valvulas a 0
            write(m,'holdingregs',44,0,'uint16');
            write(m,'holdingregs',45,0,'uint16');
            write(m,'holdingregs',46,0,'uint16');
        %Caudal entrada T3 a 0
            write(m,'holdingregs',57,0,'uint16');
        %PV a 0
            write(m,'holdingregs',49,0,'uint16');
            write(m,'holdingregs',51,0,'uint16');
            write(m,'holdingregs',53,0,'uint16');
        %Caudales de salida a 0
            write(m,'holdingregs',55,0,'uint16');
            write(m,'holdingregs',56,0,'uint16');
            write(m,'holdingregs',58,0,'uint16');
        %Cierro programa
            break;
    end

%Espero hasta tiempo de muestreo
while toc<T
end
end
```

Código Proporcional e Integral (PI)

```
%Programa PI

    IPent = 'Ip:'; %Pido Ip como comando
    ip = input(IPent,'s');

    m = modbus('tcpip',ip,502); %Me conecto a la ip proporcionada
    disp('Conectado');

    Kin= 40;
    Kout=28.224;
    Hmin=900;
    Hmax=980;

    Conent = 'Setpoint(1000~2000 mol/m3):'; %Pido Setpoint de forma manual
    Con = input(Conent);
    write(m,'holdingregs',60,Con,'uint16'); %Muestro Setpoint_Concentración por↙
pantalla

    C1=1000; % mol/m3
    C2=2000; % mol/m3
    C3=Con;  % mol/m3

    % Q3*C3=Q1*C1+Q2*C2
    % Q3=Q1+Q2

    %Para poder hacer u control solo de concentracion deber fijar el
    %valor de cauda deseado

    Q3 = 'Caudal deseado de salida(0~250 l/s):';
    SetPoint_Q3_L = input(Q3);
    SetPoint_Q3_m3s= SetPoint_Q3_L/1000;
    SetPoint_Q3_Voltios = Kout*SetPoint_Q3_m3s;

    %Saco los valores de Q1 y Q2 con los valores asignados
    A = [C1 C2; 1 1];
    B = [C3*SetPoint_Q3_m3s; SetPoint_Q3_m3s ];
    S = inv(A)*B;
    Q1T = S(1,1);
    Q1_L = 1000*Q1T;
    Q1= round(Q1_L,0);

    Q2T = S(2,1);
    Q2_L = 1000*Q2T;
    Q2 = round(Q2_L,0);

    disp('Pulsar Stop para desconectar');

    %Aseguro valvulas de salida cerradas
    write(m,'holdingregs',44,0,'uint16');
    write(m,'holdingregs',45,0,'uint16');
    write(m,'holdingregs',46,0,'uint16');

    %Tanques de materia prima
    LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por↙
el sensor
```

```
    LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor
%     write(m,'holdingregs',48,H,'uint16');
%     write(m,'holdingregs',50,H,'uint16');


    %Calculo altura equilibrio T3
    Level_Q = ((SetPoint_Q3_m3s)^2)/(((0.204)^2)*0.003);

    %Escribo Setpoint Altura para Caudal salida
    Set3 = round(Level_Q,0);
    write(m,'holdingregs',52,Set3,'uint16');

    %Escribo Setpoint Caudal salida
    write(m,'holdingregs',59,SetPoint_Q3_L ,'uint16');


    %Lleno los Tanques de materia prima
    while (LevelMeter_T1<=Hmax)&&(LevelMeter_T2<=Hmax)

        write(m,'holdingregs',41,1000,'uint16'); % hago que se llenen rapido hasta el nivel deseado
        write(m,'holdingregs',42,1000,'uint16');

        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor

        write(m,'holdingregs',49,LevelMeter_T1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T1,'uint16');

        if LevelMeter_T1>=Hmax
             write(m,'holdingregs',41,0,'uint16');
        end
        if LevelMeter_T2>=Hmax
             write(m,'holdingregs',42,0,'uint16');
        end


    end % Lleno de depositos de materia prima

    %TANQUE 1
    SetPoint_Q1_L = Q1;
    SetPoint_Q1_m3s= SetPoint_Q1_L/1000;
    SetPoint_Q1_Voltios = Kout*SetPoint_Q1_m3s;

    %TANQUE 2
    SetPoint_Q2_L = Q2;
    SetPoint_Q2_m3s= SetPoint_Q2_L/1000;
    SetPoint_Q2_Voltios = Kout*SetPoint_Q2_m3s;

    %Comienzo la regulacion
    T=0.1; % Fijo tiempo de muestreo
```

```
    %Inicializo variables

    Ek_1_ant = 0;
    Ek_2_ant = 0;

    Uk_1_ant = 0;
    Uk_2_ant = 0;


    write(m,'holdingregs',46,1000,'uint16'); %Abro valvula salida T3 para que comience↙
regulacion

    while(1)
        tic

        %Limitador Tanques Materia Primas para que no desborden
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido↙
por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido↙
por el sensor

        if (LevelMeter_T1>Hmax)
            write(m,'holdingregs',41,0,'uint16');
        elseif(LevelMeter_T1<Hmin)
             write(m,'holdingregs',41,1000,'uint16');
        else
            Red1 = round(100*(Kin/Kout)*(SetPoint_Q1_Voltios),0);
            write(m,'holdingregs',41,Red1,'uint16');
        end

        if (LevelMeter_T2>Hmax)
            write(m,'holdingregs',42,0,'uint16');
        elseif(LevelMeter_T2<Hmin)
            write(m,'holdingregs',42,1000,'uint16');
        else
            Red2 = round(100*(Kin/Kout)*(SetPoint_Q2_Voltios),0);
            write(m,'holdingregs',42,Red2,'uint16');
        end

        %TANQUE 3

        %Calculo Nivel T3 y lo muestro
        LevelMeter_T3 = read(m,'inputregs',26,1,'uint16');
        write(m,'holdingregs',53,LevelMeter_T3,'uint16');

        Q_3_ref = SetPoint_Q3_Voltios; %Voltios que deberian de salir de T3
        Voltaje_caudal_salida_3 = read(m,'inputregs',23,1,'uint16');
        Q_3_real = Voltaje_caudal_salida_3/100;%Voltios que realmente salen de T3

        Ek = Q_3_ref - Q_3_real;
        Ek_1 = (SetPoint_Q1_L/SetPoint_Q3_L)*Ek ;
        Ek_2 = (SetPoint_Q2_L/SetPoint_Q3_L)*Ek;

        % esto es debido a que contamos con 2 controladores
        % independientes,por lo que si aplicasemos directamente el error Ek
```

```
        % estariamos aplicando el doble de error, ademas de aplicarle las
        % misma proporciones a cada uno, por lo que acabiran igualandose.
        %Ek = Q_3_ref - Q_3_real;
        %Ek = (Q_1_ref + Q_2_ref) - (Q_1_real + Q_2_real);
        %Ek = (Q_1_ref - Q_1_real) + (Q_2_ref - Q_2_real);
        %Ek = Ek_1 + Ek_2;


        %TANQUE 1
        %Aplico controlador Proporcional e Integral al T1
        Q_1_ref = SetPoint_Q1_Voltios; %voltios que deberian de salir de T1 (Q1)
        Voltaje_caudal_salida_1 = read(m,'inputregs',21,1,'uint16');
        Q_1_real = Voltaje_caudal_salida_1/100; %Voltios que realmente salen de T1

        %Error_1 = Q_1_ref - Q_1_real; %en Vout

        % Lo que vamos a controlar el el porcentaje de valvula abierto

        %Valor PI
        Uk_1 = Uk_1_ant + 0.3*Ek_1 - 0.29922*Ek_1_ant;

        %Guardo variables ciclo
        Ek_1_ant = Ek_1;
        Uk_1_ant = Uk_1;

        % creamos metodo saturacion
        if (Uk_1 > 1)
            Uk_1 = 1;
        elseif (Uk_1 < 0)
            Uk_1 = 0;
        end

        %Aplicamos valor %
        Qent1 = 1000*Uk_1; % paso valor referencia factory
        Porcentaje_V1 = round(Qent1,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',44,Porcentaje_V1,'uint16') % aplico la accion

        %Calculo caudal salida T1 y lo muestro
        Caudal_salida_1 = round(0.3543*Voltaje_caudal_salida_1,0); % al multiplicar↙
por 0.3543 paso de voltios de rango[0-1000] a l/s
        write(m,'holdingregs',55,Caudal_salida_1,'uint16');

        %Calculo Nivel T1 y lo muestro
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16');
        write(m,'holdingregs',49,LevelMeter_T1,'uint16');


        %TANQUE 2

        %Aplico Proporcional
        Q_2_ref = SetPoint_Q2_Voltios; %Voltios que deberian de salir de T2
        Voltaje_caudal_salida_2 = read(m,'inputregs',22,1,'uint16');
        Q_2_real = Voltaje_caudal_salida_2/100; %Voltios que realmente salen de T2
```

```
        %Error_2 = Referencia_2 - Salida_2; %en Vout

        %Valor PI
        Uk_2 = Uk_2_ant + 0.3*Ek_2 - 0.29922*Ek_2_ant;

        %Guardo variable ciclo
        Ek_2_ant = Ek_2;
        Uk_2_ant = Uk_2;

        % creamos metodo saturacion
        if (Uk_2 > 1)
            Uk_2 = 1;
        elseif (Uk_2 < 0)
            Uk_2 = 0;
        end

        %Aplicamos valor %
        Qent2 = 1000*Uk_2; % paso valor referencia factory
        Porcentaje_V2 = round(Qent2,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',45,Porcentaje_V2,'uint16') % aplico la accion

        %Calculo caudal salida T2 y lo muestro
        Caudal_salida_2 = round(0.3543*Voltaje_caudal_salida_2,0); % al multiplicar↙
por 0.3543 paso de voltios de rango[0-1000] a l/s
        write(m,'holdingregs',56,Caudal_salida_2,'uint16');

        %Calculo Nivel T2 y lo muestro
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T2,'uint16');

        %Caudal que entra en T3 procedente de T1 y T2
        QEntrada_T3= 0.3543*Voltaje_caudal_salida_1 + 0.3543*Voltaje_caudal_salida_2;
        Q = round(QEntrada_T3,0);
        write(m,'holdingregs',57,Q,'uint16');% Muestro por pantalla lo que entra

        Q3_aplic =(Kin/Kout)*(Voltaje_caudal_salida_1 + Voltaje_caudal_salida_2);
        Q3_apli_redondeado = round(Q3_aplic,0);
        write(m,'holdingregs',43,Q3_apli_redondeado,'uint16'); % aplico la accion

        %Caudal que realmente sale por T3
        Caudal_salida_3 = round(0.3543*Voltaje_caudal_salida_3,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',58,Caudal_salida_3,'uint16');

        %Metodo de parado del programa
        Stop = read(m,'inputs',8,1);

        if (Stop==0) % si pulsas S se dejan de leer datos del servidor y se inicializa↙
a cero para el siguiente valor
            disp('Stop Pulsado');
            %Llenados a 0
                write(m,'holdingregs',41,0,'uint16');
                write(m,'holdingregs',42,0,'uint16');
                write(m,'holdingregs',43,0,'uint16');
```

```
        %Vaciados al Maximo
            write(m,'holdingregs',44,1000,'uint16');
            write(m,'holdingregs',45,1000,'uint16');
            write(m,'holdingregs',46,1000,'uint16');
        %Setpoint a 0
            write(m,'holdingregs',48,0,'uint16');
            write(m,'holdingregs',50,0,'uint16');
            write(m,'holdingregs',52,0,'uint16');
            write(m,'holdingregs',59,0,'uint16');
            write(m,'holdingregs',60,0,'uint16');
        %valvulas a 0
            write(m,'holdingregs',44,0,'uint16');
            write(m,'holdingregs',45,0,'uint16');
            write(m,'holdingregs',46,0,'uint16');
        %Caudal entrada T3 a 0
            write(m,'holdingregs',57,0,'uint16');
        %PV a 0
            write(m,'holdingregs',49,0,'uint16');
            write(m,'holdingregs',51,0,'uint16');
            write(m,'holdingregs',53,0,'uint16');
        %Caudales de salida a 0
            write(m,'holdingregs',55,0,'uint16');
            write(m,'holdingregs',56,0,'uint16');
            write(m,'holdingregs',58,0,'uint16');
        %Cierro programa
            break;
    end

%Espero hasta tiempo de muestreo
while toc<T
end
end
```

DeadBeat

```
    IPent = 'Ip:'; %Pido Ip como comando
    ip = input(IPent,'s');

    m = modbus('tcpip',ip,502); %Me conecto a la ip proporcionada
    disp('Conectado');

    Kin= 40;
    Kout=28.224;
    Hmin=900;
    Hmax=980;

    Conent = 'Setpoint(1000~2000 mol/m3):'; %Pido Setpoint de forma manual
    Con = input(Conent);
    write(m,'holdingregs',60,Con,'uint16'); %Muestro Setpoint_Concentración por↙
pantalla

    C1=1000; % mol/m3
    C2=2000; % mol/m3
    C3=Con;  % mol/m3

    % Q3*C3=Q1*C1+Q2*C2
    % Q3=Q1+Q2

    %Para poder hacer u control solo de concentracion deber fijar el
    %valor de cauda deseado

    Q3 = 'Caudal deseado de salida(0~250 l/s):';
    SetPoint_Q3_L = input(Q3);
    SetPoint_Q3_m3s= SetPoint_Q3_L/1000;
    SetPoint_Q3_Voltios = Kout*SetPoint_Q3_m3s;

    %Saco los valores de Q1 y Q2 con los valores asignados
    A = [C1 C2; 1 1];
    B = [C3*SetPoint_Q3_m3s; SetPoint_Q3_m3s ];
    S = inv(A)*B;
    Q1T = S(1,1);
    Q1_L = 1000*Q1T;
    Q1= round(Q1_L,0);

    Q2T = S(2,1);
    Q2_L = 1000*Q2T;
    Q2 = round(Q2_L,0);

    disp('Pulsar Stop para desconectar');

    %Aseguro valvulas de salida cerradas
    write(m,'holdingregs',44,0,'uint16');
    write(m,'holdingregs',45,0,'uint16');
    write(m,'holdingregs',46,0,'uint16');

    %Tanques de materia prima
    LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por↙
el sensor
```

```
    LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor
%     write(m,'holdingregs',48,H,'uint16');
%     write(m,'holdingregs',50,H,'uint16');


    %Calculo altura equilibrio T3
    Level_Q = ((SetPoint_Q3_m3s)^2)/(((0.204)^2)*0.003);

    %Escribo Setpoint Altura para Caudal salida
    Set3 = round(Level_Q,0);
    write(m,'holdingregs',52,Set3,'uint16');

    %Escribo Setpoint Caudal salida
    write(m,'holdingregs',59,SetPoint_Q3_L ,'uint16');


    %Lleno los Tanques de materia prima
    while (LevelMeter_T1<=Hmax)&&(LevelMeter_T2<=Hmax)

        write(m,'holdingregs',41,1000,'uint16'); % hago que se llenen rapido hasta el nivel deseado
        write(m,'holdingregs',42,1000,'uint16');

        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor

        write(m,'holdingregs',49,LevelMeter_T1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T1,'uint16');

        if LevelMeter_T1>=Hmax
            write(m,'holdingregs',41,0,'uint16');
        end
        if LevelMeter_T2>=Hmax
            write(m,'holdingregs',42,0,'uint16');
        end


    end % Lleno de depositos de materia prima

    %TANQUE 1
    SetPoint_Q1_L = Q1;
    SetPoint_Q1_m3s= SetPoint_Q1_L/1000;
    SetPoint_Q1_Voltios = Kout*SetPoint_Q1_m3s;

    %TANQUE 2
    SetPoint_Q2_L = Q2;
    SetPoint_Q2_m3s= SetPoint_Q2_L/1000;
    SetPoint_Q2_Voltios = Kout*SetPoint_Q2_m3s;

    %Comienzo la regulacion
    T=0.1; % Fijo tiempo de muestreo
```

```
    %Inicializo variables

    Ek_1_ant = 0;
    Ek_2_ant = 0;

    Uk_1_ant = 0;
    Uk_2_ant = 0;


    write(m,'holdingregs',46,1000,'uint16'); %Abro valvula salida T3 para que comience↙
regulacion

    while(1)
        tic

        %Limitador Tanques Materia Primas para que no desborden
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido↙
por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido↙
por el sensor

        if (LevelMeter_T1>Hmax)
            write(m,'holdingregs',41,0,'uint16');
        elseif(LevelMeter_T1<Hmin)
             write(m,'holdingregs',41,1000,'uint16');
        else
            Red1 = round(100*(Kin/Kout)*(SetPoint_Q1_Voltios),0);
            write(m,'holdingregs',41,Red1,'uint16');
        end

        if (LevelMeter_T2>Hmax)
            write(m,'holdingregs',42,0,'uint16');
        elseif(LevelMeter_T2<Hmin)
            write(m,'holdingregs',42,1000,'uint16');
        else
            Red2 = round(100*(Kin/Kout)*(SetPoint_Q2_Voltios),0);
            write(m,'holdingregs',42,Red2,'uint16');
        end

        %TANQUE 3

        %Calculo Nivel T3 y lo muestro
        LevelMeter_T3 = read(m,'inputregs',26,1,'uint16');
        write(m,'holdingregs',53,LevelMeter_T3,'uint16');

        Q_3_ref = SetPoint_Q3_Voltios; %Voltios que deberian de salir de T3
        Voltaje_caudal_salida_3 = read(m,'inputregs',23,1,'uint16');
        Q_3_real = Voltaje_caudal_salida_3/100;%Voltios que realmente salen de T3

        Ek = Q_3_ref - Q_3_real;
        Ek_1 = (SetPoint_Q1_L/SetPoint_Q3_L)*Ek ;
        Ek_2 = (SetPoint_Q2_L/SetPoint_Q3_L)*Ek;

        % esto es debido a que contamos con 2 controladores
        % independientes,por lo que si aplicasemos directamente el error Ek
```

```
        % estariamos aplicando el doble de error, ademas de aplicarle las
        % misma proporciones a cada uno, por lo que acabiran igualandose.
        %Ek = Q_3_ref - Q_3_real;
        %Ek = (Q_1_ref + Q_2_ref) - (Q_1_real + Q_2_real);
        %Ek = (Q_1_ref - Q_1_real) + (Q_2_ref - Q_2_real);
        %Ek = Ek_1 + Ek_2;


        %TANQUE 1
        %Aplico controlador Proporcional e Integral al T1
        Q_1_ref = SetPoint_Q1_Voltios; %voltios que deberian de salir de T1 (Q1)
        Voltaje_caudal_salida_1 = read(m,'inputregs',21,1,'uint16');
        Q_1_real = Voltaje_caudal_salida_1/100; %Voltios que realmente salen de T1

        %Error_1 = Q_1_ref - Q_1_real; %en Vout

        % Lo que vamos a controlar el el porcentaje de valvula abierto

        %Valor PI
        Uk_1 = Uk_1_ant + 1.9888*Ek_1 - 1.9836*Ek_1_ant;

        %Guardo variables ciclo
        Ek_1_ant = Ek_1;
        Uk_1_ant = Uk_1;

        % creamos metodo antiwindup
        if (Uk_1 > 1)
            Uk_1 = 1;
        elseif (Uk_1 < 0)
            Uk_1 = 0;
        end

        %Aplicamos valor %
        Qent1 = 1000*Uk_1; % paso valor referencia factory
        Porcentaje_V1 = round(Qent1,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',44,Porcentaje_V1,'uint16') % aplico la accion

        %Calculo caudal salida T1 y lo muestro
        Caudal_salida_1 = round(0.3543*Voltaje_caudal_salida_1,0); % al multiplicar↙
por 0.3543 paso de voltios de rango[0-1000] a l/s
        write(m,'holdingregs',55,Caudal_salida_1,'uint16');

        %Calculo Nivel T1 y lo muestro
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16');
        write(m,'holdingregs',49,LevelMeter_T1,'uint16');


        %TANQUE 2

        %Aplico Proporcional
        Q_2_ref = SetPoint_Q2_Voltios; %Voltios que deberian de salir de T2
        Voltaje_caudal_salida_2 = read(m,'inputregs',22,1,'uint16');
        Q_2_real = Voltaje_caudal_salida_2/100; %Voltios que realmente salen de T2
```

```
        %Error_2 = Referencia_2 - Salida_2; %en Vout

        %Valor PI
        Uk_2 = Uk_2_ant + 4*(0.4972*Ek_2 - 0.4959*Ek_2_ant);

        %Guardo variable ciclo
        Ek_2_ant = Ek_2;
        Uk_2_ant = Uk_2;

        % creamos metodo antiwindup
        if (Uk_2 > 1)
            Uk_2 = 1;
        elseif (Uk_2 < 0)
            Uk_2 = 0;
        end

        %Aplicamos valor %
        Qent2 = 1000*Uk_2; % paso valor referencia factory
        Porcentaje_V2 = round(Qent2,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',45,Porcentaje_V2,'uint16') % aplico la accion

        %Calculo caudal salida T2 y lo muestro
        Caudal_salida_2 = round(0.3543*Voltaje_caudal_salida_2,0); % al multiplicar↙
por 0.3543 paso de voltios de rango[0-1000] a l/s
        write(m,'holdingregs',56,Caudal_salida_2,'uint16');

        %Calculo Nivel T2 y lo muestro
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T2,'uint16');

        %Caudal que entra en T3 procedente de T1 y T2
        QEntrada_T3= 0.3543*Voltaje_caudal_salida_1 + 0.3543*Voltaje_caudal_salida_2;
        Q = round(QEntrada_T3,0);
        write(m,'holdingregs',57,Q,'uint16');% Muestro por pantalla lo que entra

        Q3_aplic =(Kin/Kout)*(Voltaje_caudal_salida_1 + Voltaje_caudal_salida_2);
        Q3_apli_redondeado = round(Q3_aplic,0);
        write(m,'holdingregs',43,Q3_apli_redondeado,'uint16'); % aplico la accion

        %Caudal que realmente sale por T3
        Caudal_salida_3 = round(0.3543*Voltaje_caudal_salida_3,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',58,Caudal_salida_3,'uint16');

        %Metodo de parado del programa
        Stop = read(m,'inputs',8,1);

        if (Stop==0) % si pulsas S se dejan de leer datos del servidor y se inicializa↙
a cero para el siguiente valor
            disp('Stop Pulsado');
            %Llenados a 0
                write(m,'holdingregs',41,0,'uint16');
                write(m,'holdingregs',42,0,'uint16');
                write(m,'holdingregs',43,0,'uint16');
```

```
        %Vaciados al Maximo
            write(m,'holdingregs',44,1000,'uint16');
            write(m,'holdingregs',45,1000,'uint16');
            write(m,'holdingregs',46,1000,'uint16');
        %Setpoint a 0
            write(m,'holdingregs',48,0,'uint16');
            write(m,'holdingregs',50,0,'uint16');
            write(m,'holdingregs',52,0,'uint16');
            write(m,'holdingregs',59,0,'uint16');
            write(m,'holdingregs',60,0,'uint16');
        %valvulas a 0
            write(m,'holdingregs',44,0,'uint16');
            write(m,'holdingregs',45,0,'uint16');
            write(m,'holdingregs',46,0,'uint16');
        %Caudal entrada T3 a 0
            write(m,'holdingregs',57,0,'uint16');
        %PV a 0
            write(m,'holdingregs',49,0,'uint16');
            write(m,'holdingregs',51,0,'uint16');
            write(m,'holdingregs',53,0,'uint16');
        %Caudales de salida a 0
            write(m,'holdingregs',55,0,'uint16');
            write(m,'holdingregs',56,0,'uint16');
            write(m,'holdingregs',58,0,'uint16');
        %Cierro programa
            break;
    end

%Espero hasta tiempo de muestreo
while toc<T
end
end
```

Código Proporcional, Integral y Derivador (PID)

```
%Programa PID

    IPent = 'Ip:'; %Pido Ip como comando
    ip = input(IPent,'s');

    m = modbus('tcpip',ip,502); %Me conecto a la ip proporcionada
    disp('Conectado');

    Kin= 40;
    Kout=28.224;
    Hmin=900;
    Hmax=980;

    Conent = 'Setpoint(1000~2000 mol/m3):'; %Pido Setpoint de forma manual
    Con = input(Conent);
    write(m,'holdingregs',60,Con,'uint16'); %Muestro Setpoint_Concentración por↙
pantalla

    C1=1000; % mol/m3
    C2=2000; % mol/m3
    C3=Con;  % mol/m3

    % Q3*C3=Q1*C1+Q2*C2
    % Q3=Q1+Q2

    %Para poder hacer u control solo de concentracion deber fijar el
    %valor de cauda deseado

    Q3 = 'Caudal deseado de salida(0~250 l/s):';
    SetPoint_Q3_L = input(Q3);
    SetPoint_Q3_m3s= SetPoint_Q3_L/1000;
    SetPoint_Q3_Voltios = Kout*SetPoint_Q3_m3s;

    %Saco los valores de Q1 y Q2 con los valores asignados
    A = [C1 C2; 1 1];
    B = [C3*SetPoint_Q3_m3s; SetPoint_Q3_m3s ];
    S = inv(A)*B;
    Q1T = S(1,1);
    Q1_L = 1000*Q1T;
    Q1= round(Q1_L,0);

    Q2T = S(2,1);
    Q2_L = 1000*Q2T;
    Q2 = round(Q2_L,0);

    disp('Pulsar Stop para desconectar');

    %Aseguro valvulas de salida cerradas
    write(m,'holdingregs',44,0,'uint16');
    write(m,'holdingregs',45,0,'uint16');
    write(m,'holdingregs',46,0,'uint16');

    %Tanques de materia prima
    LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por↙
el sensor
```

```
    LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor

    %Calculo altura equilibrio T3
    Level_Q = ((SetPoint_Q3_m3s)^2)/(((0.204)^2)*0.003);

    %Escribo Setpoint Altura para Caudal salida
    Set3 = round(Level_Q,0);
    write(m,'holdingregs',52,Set3,'uint16');

    %Escribo Setpoint Caudal salida
    write(m,'holdingregs',59,SetPoint_Q3_L ,'uint16');

    %Lleno los Tanques de materia prima
    while (LevelMeter_T1<=Hmax)&&(LevelMeter_T2<=Hmax)

        write(m,'holdingregs',41,1000,'uint16'); % hago que se llenen rapido hasta el nivel deseado
        write(m,'holdingregs',42,1000,'uint16');

        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor

        write(m,'holdingregs',49,LevelMeter_T1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T1,'uint16');

        if LevelMeter_T1>=Hmax
             write(m,'holdingregs',41,0,'uint16');
        end
        if LevelMeter_T2>=Hmax
             write(m,'holdingregs',42,0,'uint16');
        end


    end % Lleno de depositos de materia prima

    %TANQUE 1
    SetPoint_Q1_L = Q1;
    SetPoint_Q1_m3s= SetPoint_Q1_L/1000;
    SetPoint_Q1_Voltios = Kout*SetPoint_Q1_m3s;

    %TANQUE 2
    SetPoint_Q2_L = Q2;
    SetPoint_Q2_m3s= SetPoint_Q2_L/1000;
    SetPoint_Q2_Voltios = Kout*SetPoint_Q2_m3s;

    %Comienzo la regulacion
    T = 0.1;% Fijo tiempo de muestreo

    %Inicializo variables

    Ek_1_ant = 0;
    Ek_2_ant = 0;
```

```
    Ek_1_ant_ant = 0;
    Ek_2_ant_ant = 0;
    Uk_1_ant = 0;
    Uk_2_ant = 0;

    %Fijo valores controlador

    K=0.627;
    Ti=0.035;
    Td=0.05;

    P = K;
    Ki = Ti*T;
    Kd = Td/T;

    write(m,'holdingregs',46,1000,'uint16');%Abro valvula salida T3 para que comience
regulacion

    while(1)
        tic

        %Limitador Tanques Materia Primas para que no desborden
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido
por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido
por el sensor

        if (LevelMeter_T1>Hmax)
            write(m,'holdingregs',41,0,'uint16');
        elseif(LevelMeter_T1<Hmin)
             write(m,'holdingregs',41,1000,'uint16');
        else
            Red1 = round(100*(Kin/Kout)*(SetPoint_Q1_Voltios),0);
            write(m,'holdingregs',41,Red1,'uint16');
        end

        if (LevelMeter_T2>Hmax)
            write(m,'holdingregs',42,0,'uint16');
        elseif(LevelMeter_T2<Hmin)
            write(m,'holdingregs',42,1000,'uint16');
        else
            Red2 = round(100*(Kin/Kout)*(SetPoint_Q2_Voltios),0);
            write(m,'holdingregs',42,Red2,'uint16');
        end

        %TANQUE 3
        %Calculo Nivel T3 y lo muestro
        LevelMeter_T3 = read(m,'inputregs',26,1,'uint16');
        write(m,'holdingregs',53,LevelMeter_T3,'uint16');

        Q_3_ref = SetPoint_Q3_Voltios; %Voltios que deberian de salir de T3
        Voltaje_caudal_salida_3 = read(m,'inputregs',23,1,'uint16');
        Q_3_real = Voltaje_caudal_salida_3/100;%Voltios que realmente salen de T3

        Ek = Q_3_ref - Q_3_real;
```

```
        Ek_1 = (SetPoint_Q1_L/SetPoint_Q3_L)*Ek ;
        Ek_2 = (SetPoint_Q2_L/SetPoint_Q3_L)*Ek;

        % esto es debido a que contamos con 2 controladores
        % independientes,por lo que si aplicasemos directamente el error Ek
        % estariamos aplicando el doble de error, ademas de aplicarle las
        % misma proporciones a cada uno, por lo que acabiran igualandose.
        %Ek = Q_3_ref - Q_3_real;
        %Ek = (Q_1_ref + Q_2_ref) - (Q_1_real + Q_2_real);
        %Ek = (Q_1_ref - Q_1_real) + (Q_2_ref - Q_2_real);
        %Ek = Ek_1 + Ek_2;


        %TANQUE 1
        %Aplico controlador Proporcional e Integral al T1
        Q_1_ref = SetPoint_Q1_Voltios; %voltios que deberian de salir de T1 (Q1)
        Voltaje_caudal_salida_1 = read(m,'inputregs',21,1,'uint16');
        Q_1_real = Voltaje_caudal_salida_1/100; %Voltios que realmente salen de T1

        %Error_1 = Q_1_ref - Q_1_real; %en Vout

        % Lo que vamos a controlar el el porcentaje de valvula abierto

        %Valor PID
        Uk_1 = Uk_1_ant + P*( (Kd + 1)*Ek_1 + (Ki - 2*Kd - 1)*Ek_1_ant +↙
Kd*Ek_1_ant_ant);

        %Guardo Variables ciclo
        Uk_1_ant = Uk_1;
        Ek_1_ant_ant = Ek_1_ant;
        Ek_1_ant = Ek_1;

        % creamos metodo antiwindup
        if (Uk_1 > 1)
            Uk_1 = 1;
        elseif (Uk_1 < 0)
            Uk_1 = 0;
        end

        %Aplicamos valor %
        Qent1 = 1000*Uk_1; % paso valor referencia factory
        Porcentaje_V1 = round(Qent1,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',44,Porcentaje_V1,'uint16') % aplico la accion

        %Calculo caudal salida T1 y lo muestro
        Caudal_salida_1 = round(0.3543*Voltaje_caudal_salida_1,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',55,Caudal_salida_1,'uint16');

        %Calculo Nivel T1 y lo muestro
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16');
        write(m,'holdingregs',49,LevelMeter_T1,'uint16');
```

```
        %TANQUE 2
        %Aplico Proporcional
        Q_2_ref = SetPoint_Q2_Voltios; %Voltios que deberian de salir de T2
        Voltaje_caudal_salida_2 = read(m,'inputregs',22,1,'uint16');
        Q_2_real = Voltaje_caudal_salida_2/100; %Voltios que realmente salen de T2

        %Error_2 = Referencia_2 - Salida_2; %en Vout

        %Valor PI
        Uk_2 = Uk_2_ant + P*( (Kd + 1)*Ek_2 + (Ki - 2*Kd - 1)*Ek_2_ant +↙
Kd*Ek_2_ant_ant );

        %Guardo Variables ciclo
        Uk_2_ant = Uk_2;
        Ek_2_ant_ant = Ek_2_ant;
        Ek_2_ant = Ek_1;

        % creamos metodo antiwindup
        if (Uk_2 > 1)
            Uk_2 = 1;
        elseif (Uk_2 < 0)
            Uk_2 = 0;
        end

        %Aplicamos valor %
        Qent2 = 1000*Uk_2; % paso valor referencia factory
        Porcentaje_V2 = round(Qent2,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',45,Porcentaje_V2,'uint16') % aplico la accion

        %Calculo caudal salida T2 y lo muestro
        Caudal_salida_2 = round(0.3543*Voltaje_caudal_salida_2,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',56,Caudal_salida_2,'uint16');

        %Calculo Nivel T2 y lo muestro
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T2,'uint16');

        %Caudal que entra en T3 procedente de T1 y T2
        QEntrada_T3= 0.3543*Voltaje_caudal_salida_1 + 0.3543*Voltaje_caudal_salida_2;
        Q = round(QEntrada_T3,0);
        write(m,'holdingregs',57,Q,'uint16');% Muestro por pantalla lo que entra

        Q3_aplic =(Kin/Kout)*(Voltaje_caudal_salida_1 + Voltaje_caudal_salida_2);
        Q3_apli_redondeado = round(Q3_aplic,0);
        write(m,'holdingregs',43,Q3_apli_redondeado,'uint16'); % aplico la accion

        %Caudal que realmente sale por T3
        Caudal_salida_3 = round(0.3543*Voltaje_caudal_salida_3,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',58,Caudal_salida_3,'uint16');

        %Metodo de parado del programa
        Stop = read(m,'inputs',8,1);
```

```
        if (Stop==0) % si pulsas S se dejan de leer datos del servidor y se inicializa↙
a cero para el siguiente valor
            disp('Stop Pulsado');
            %Llenados a 0
                write(m,'holdingregs',41,0,'uint16');
                write(m,'holdingregs',42,0,'uint16');
                write(m,'holdingregs',43,0,'uint16');
            %Vaciados al Maximo
                write(m,'holdingregs',44,1000,'uint16');
                write(m,'holdingregs',45,1000,'uint16');
                write(m,'holdingregs',46,1000,'uint16');
            %Setpoint a 0
                write(m,'holdingregs',48,0,'uint16');
                write(m,'holdingregs',50,0,'uint16');
                write(m,'holdingregs',52,0,'uint16');
                write(m,'holdingregs',59,0,'uint16');
                write(m,'holdingregs',60,0,'uint16');
            %valvulas a 0
                write(m,'holdingregs',44,0,'uint16');
                write(m,'holdingregs',45,0,'uint16');
                write(m,'holdingregs',46,0,'uint16');
            %Caudal entrada T3 a 0
                write(m,'holdingregs',57,0,'uint16');
            %PV a 0
                write(m,'holdingregs',49,0,'uint16');
                write(m,'holdingregs',51,0,'uint16');
                write(m,'holdingregs',53,0,'uint16');
            %Caudales de salida a 0
                write(m,'holdingregs',55,0,'uint16');
                write(m,'holdingregs',56,0,'uint16');
                write(m,'holdingregs',58,0,'uint16');
            %Cierro programa
                break;
        end

    %Espero hasta tiempo de muestreo
    while toc<T
    end
    end
```

Código Proporcional, Integral y Derivador (PID) con On/Off

```
%Programa PID

    IPent = 'Ip:'; %Pido Ip como comando
    ip = input(IPent,'s');

    m = modbus('tcpip',ip,502); %Me conecto a la ip proporcionada
    disp('Conectado');

    Kin= 40;
    Kout=28.224;
    Hmin=900;
    Hmax=980;

    Conent = 'Setpoint(1000~2000 mol/m3):'; %Pido Setpoint de forma manual
    Con = input(Conent);
    write(m,'holdingregs',60,Con,'uint16'); %Muestro Setpoint_Concentración por↙
pantalla

    C1=1000; % mol/m3
    C2=2000; % mol/m3
    C3=Con;  % mol/m3

    % Q3*C3=Q1*C1+Q2*C2
    % Q3=Q1+Q2

    %Para poder hacer u control solo de concentracion deber fijar el
    %valor de cauda deseado

    Q3 = 'Caudal deseado de salida(0~250 l/s):';
    SetPoint_Q3_L = input(Q3);
    SetPoint_Q3_m3s= SetPoint_Q3_L/1000;
    SetPoint_Q3_Voltios = Kout*SetPoint_Q3_m3s;

    %Saco los valores de Q1 y Q2 con los valores asignados
    A = [C1 C2; 1 1];
    B = [C3*SetPoint_Q3_m3s; SetPoint_Q3_m3s ];
    S = inv(A)*B;
    Q1T = S(1,1);
    Q1_L = 1000*Q1T;
    Q1= round(Q1_L,0);

    Q2T = S(2,1);
    Q2_L = 1000*Q2T;
    Q2 = round(Q2_L,0);

    disp('Pulsar Stop para desconectar');

    %Aseguro valvulas de salida cerradas
    write(m,'holdingregs',44,0,'uint16');
    write(m,'holdingregs',45,0,'uint16');
    write(m,'holdingregs',46,0,'uint16');

    %Tanques de materia prima
    LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por↙
el sensor
```

```
    LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor

    %Calculo altura equilibrio T3
    Level_Q = ((SetPoint_Q3_m3s)^2)/(((0.204)^2)*0.003);

    %Escribo Setpoint Altura para Caudal salida
    Set3 = round(Level_Q,0);
    write(m,'holdingregs',52,Set3,'uint16');

    %Escribo Setpoint Caudal salida
    write(m,'holdingregs',59,SetPoint_Q3_L ,'uint16');

    %Lleno los Tanques de materia prima
    while (LevelMeter_T1<=Hmax)&&(LevelMeter_T2<=Hmax)

        write(m,'holdingregs',41,1000,'uint16'); % hago que se llenen rapido hasta el nivel deseado
        write(m,'holdingregs',42,1000,'uint16');

        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor

        write(m,'holdingregs',49,LevelMeter_T1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T1,'uint16');

        if LevelMeter_T1>=Hmax
            write(m,'holdingregs',41,0,'uint16');
        end
        if LevelMeter_T2>=Hmax
            write(m,'holdingregs',42,0,'uint16');
        end


    end % Lleno de depositos de materia prima

    %TANQUE 1
    SetPoint_Q1_L = Q1;
    SetPoint_Q1_m3s= SetPoint_Q1_L/1000;
    SetPoint_Q1_Voltios = Kout*SetPoint_Q1_m3s;

    %TANQUE 2
    SetPoint_Q2_L = Q2;
    SetPoint_Q2_m3s= SetPoint_Q2_L/1000;
    SetPoint_Q2_Voltios = Kout*SetPoint_Q2_m3s;

    %TANQUE 3
    %Calculo Nivel T3 y lo muestro
    LevelMeter_T3 = read(m,'inputregs',26,1,'uint16');
    write(m,'holdingregs',53,LevelMeter_T3,'uint16');

    D1 = 1000*(SetPoint_Q1_L/SetPoint_Q3_L);
    D2 = 1000*(SetPoint_Q2_L/SetPoint_Q3_L);
```

```
    %Lleno hasta punto de equilibrio T3
    while (LevelMeter_T3<=Level_Q)

        write(m,'holdingregs',44,D1,'uint16');% Hago que la valvula 1 se abra para el caudal deseado
        write(m,'holdingregs',45,D2,'uint16');% Hago que la valvula 2 se abra para el caudal deseado

        write(m,'holdingregs',41,1000,'uint16'); %Introduzco al maximo materias primas T1
        write(m,'holdingregs',42,1000,'uint16'); %Introduzco al maximo materias primas T2

        %Limitador Tanques Materia Primas para que no desborden
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido por el sensor
        write(m,'holdingregs',49,LevelMeter_T1,'uint16');%Muestro nivel de los tanques
        write(m,'holdingregs',51,LevelMeter_T1,'uint16');

        if (LevelMeter_T1>Hmax)
            write(m,'holdingregs',41,0,'uint16');
        end
        if (LevelMeter_T2>Hmax)
            write(m,'holdingregs',42,0,'uint16');
        end

        Voltaje_caudal_salida_1 = read(m,'inputregs',21,1,'uint16'); %Leo Caudales de salida
        Voltaje_caudal_salida_2 = read(m,'inputregs',22,1,'uint16');

        Q_salida_T1=0.3543*Voltaje_caudal_salida_1;%Paso de voltios a litros
        Qs1=round(Q_salida_T1,0);

        Q_salida_T2=0.3543*Voltaje_caudal_salida_2;
        Qs2=round(Q_salida_T2,0);

        write(m,'holdingregs',55,Qs1,'uint16');%Muestro caudal salida de los tanques
        write(m,'holdingregs',56,Qs2,'uint16');

        QEntrada_T3= Q_salida_T1 + Q_salida_T2;
        Q = round(QEntrada_T3,0);
        write(m,'holdingregs',57,Q,'uint16');% Muestro por pantalla lo que entra
        Q3_aplic =(Kin/Kout)*(Voltaje_caudal_salida_1 + Voltaje_caudal_salida_2);
        Q3_apli_redondeado = round(Q3_aplic,0);
        write(m,'holdingregs',43,Q3_apli_redondeado,'uint16'); % aplico la accion
        %Leo Nivel T3 y lo muestro
        LevelMeter_T3 = read(m,'inputregs',26,1,'uint16');
        write(m,'holdingregs',53,LevelMeter_T3,'uint16');
    end


    %Comienzo la regulacion
```

```
    T = 0.1;% Fijo tiempo de muestreo

    %Inicializo variables

    Ek_1_ant = 0;
    Ek_2_ant = 0;
    Ek_1_ant_ant = 0;
    Ek_2_ant_ant = 0;
    Uk_1_ant = SetPoint_Q1_m3s/0.336;
    Uk_2_ant = SetPoint_Q2_m3s/0.336;

    %Fijo valores controlador

    K=0.64031;
    Ti=0.1224;
    Td=0.2262;

    P = K;
    Ki = Ti*T;
    Kd = Td/T;

    write(m,'holdingregs',46,1000,'uint16');%Abro valvula salida T3 para que comience↙
regulacion

    while(1)
        tic

        %Limitador Tanques Materia Primas para que no desborden
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16'); % Leo El nivel del T1 leido↙
por el sensor
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16'); % Leo El nivel del T2 leido↙
por el sensor

        if (LevelMeter_T1>Hmax)
            write(m,'holdingregs',41,0,'uint16');
        elseif(LevelMeter_T1<Hmin)
             write(m,'holdingregs',41,1000,'uint16');
        else
            Red1 = round(100*(Kin/Kout)*(SetPoint_Q1_Voltios),0);
            write(m,'holdingregs',41,Red1,'uint16');
        end

        if (LevelMeter_T2>Hmax)
            write(m,'holdingregs',42,0,'uint16');
        elseif(LevelMeter_T2<Hmin)
            write(m,'holdingregs',42,1000,'uint16');
        else
            Red2 = round(100*(Kin/Kout)*(SetPoint_Q2_Voltios),0);
            write(m,'holdingregs',42,Red2,'uint16');
        end

        %TANQUE 3
        %Calculo Nivel T3 y lo muestro
        LevelMeter_T3 = read(m,'inputregs',26,1,'uint16');
        write(m,'holdingregs',53,LevelMeter_T3,'uint16');
```

```
        Q_3_ref = SetPoint_Q3_Voltios; %Voltios que deberian de salir de T3
        Voltaje_caudal_salida_3 = read(m,'inputregs',23,1,'uint16');
        Q_3_real = Voltaje_caudal_salida_3/100;%Voltios que realmente salen de T3

        Ek = Q_3_ref - Q_3_real;
        Ek_1 = (SetPoint_Q1_L/SetPoint_Q3_L)*Ek ;
        Ek_2 = (SetPoint_Q2_L/SetPoint_Q3_L)*Ek;

        % esto es debido a que contamos con 2 controladores
        % independientes,por lo que si aplicasemos directamente el error Ek
        % estariamos aplicando el doble de error, ademas de aplicarle las
        % misma proporciones a cada uno, por lo que acabiran igualandose.
        %Ek = Q_3_ref - Q_3_real;
        %Ek = (Q_1_ref + Q_2_ref) - (Q_1_real + Q_2_real);
        %Ek = (Q_1_ref - Q_1_real) + (Q_2_ref - Q_2_real);
        %Ek = Ek_1 + Ek_2;


        %TANQUE 1
        %Aplico controlador Proporcional e Integral al T1
        Q_1_ref = SetPoint_Q1_Voltios; %voltios que deberian de salir de T1 (Q1)
        Voltaje_caudal_salida_1 = read(m,'inputregs',21,1,'uint16');
        Q_1_real = Voltaje_caudal_salida_1/100; %Voltios que realmente salen de T1

        %Error_1 = Q_1_ref - Q_1_real; %en Vout

        % Lo que vamos a controlar el el porcentaje de valvula abierto

        %Valor PID
        Uk_1 = Uk_1_ant + P*( (Kd + 1)*Ek_1 + (Ki - 2*Kd - 1)*Ek_1_ant +↙
Kd*Ek_1_ant_ant);

        %Guardo Variables ciclo
        Uk_1_ant = Uk_1;
        Ek_1_ant_ant = Ek_1_ant;
        Ek_1_ant = Ek_1;

        % creamos metodo antiwindup
        if (Uk_1 > 1)
            Uk_1 = 1;
        elseif (Uk_1 < 0)
            Uk_1 = 0;
        end

        %Aplicamos valor %
        Qent1 = 1000*Uk_1; % paso valor referencia factory
        Porcentaje_V1 = round(Qent1,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',44,Porcentaje_V1,'uint16') % aplico la accion

        %Calculo caudal salida T1 y lo muestro
        Caudal_salida_1 = round(0.3543*Voltaje_caudal_salida_1,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',55,Caudal_salida_1,'uint16');
```

```
        %Calculo Nivel T1 y lo muestro
        LevelMeter_T1 = read(m,'inputregs',24,1,'uint16');
        write(m,'holdingregs',49,LevelMeter_T1,'uint16');


        %TANQUE 2
        %Aplico Proporcional
        Q_2_ref = SetPoint_Q2_Voltios; %Voltios que deberian de salir de T2
        Voltaje_caudal_salida_2 = read(m,'inputregs',22,1,'uint16');
        Q_2_real = Voltaje_caudal_salida_2/100; %Voltios que realmente salen de T2

        %Error_2 = Referencia_2 - Salida_2; %en Vout

        %Valor PI
        Uk_2 = Uk_2_ant + P*( (Kd + 1)*Ek_2 + (Ki - 2*Kd - 1)*Ek_2_ant +↙
Kd*Ek_2_ant_ant );

        %Guardo Variables ciclo
        Uk_2_ant = Uk_2;
        Ek_2_ant_ant = Ek_2_ant;
        Ek_2_ant = Ek_1;

        % creamos metodo antiwindup
        if (Uk_2 > 1)
            Uk_2 = 1;
        elseif (Uk_2 < 0)
            Uk_2 = 0;
        end

        %Aplicamos valor %
        Qent2 = 1000*Uk_2; % paso valor referencia factory
        Porcentaje_V2 = round(Qent2,0); % redondeo para que sea uint16 y poder↙
aplicarlo
        write(m,'holdingregs',45,Porcentaje_V2,'uint16') % aplico la accion

        %Calculo caudal salida T2 y lo muestro
        Caudal_salida_2 = round(0.3543*Voltaje_caudal_salida_2,0); % al multiplicar↙
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',56,Caudal_salida_2,'uint16');

        %Calculo Nivel T2 y lo muestro
        LevelMeter_T2 = read(m,'inputregs',25,1,'uint16');
        write(m,'holdingregs',51,LevelMeter_T2,'uint16');

        %Caudal que entra en T3 procedente de T1 y T2
        QEntrada_T3= 0.3543*Voltaje_caudal_salida_1 + 0.3543*Voltaje_caudal_salida_2;
        Q = round(QEntrada_T3,0);
        write(m,'holdingregs',57,Q,'uint16');% Muestro por pantalla lo que entra

        Q3_aplic =(Kin/Kout)*(Voltaje_caudal_salida_1 + Voltaje_caudal_salida_2);
        Q3_apli_redondeado = round(Q3_aplic,0);
        write(m,'holdingregs',43,Q3_apli_redondeado,'uint16'); % aplico la accion

        %Caudal que realmente sale por T3
```

```
        Caudal_salida_3 = round(0.3543*Voltaje_caudal_salida_3,0); % al multiplicar
por 0.3543 paso de voltios de rando[0-1000] a l/s
        write(m,'holdingregs',58,Caudal_salida_3,'uint16');

        %Metodo de parado del programa
        Stop = read(m,'inputs',8,1);

        if (Stop==0) % si pulsas S se dejan de leer datos del servidor y se inicializa
a cero para el siguiente valor
            disp('Stop Pulsado');
            %Llenados a 0
                write(m,'holdingregs',41,0,'uint16');
                write(m,'holdingregs',42,0,'uint16');
                write(m,'holdingregs',43,0,'uint16');
            %Vaciados al Maximo
                write(m,'holdingregs',44,1000,'uint16');
                write(m,'holdingregs',45,1000,'uint16');
                write(m,'holdingregs',46,1000,'uint16');
            %Setpoint a 0
                write(m,'holdingregs',48,0,'uint16');
                write(m,'holdingregs',50,0,'uint16');
                write(m,'holdingregs',52,0,'uint16');
                write(m,'holdingregs',59,0,'uint16');
                write(m,'holdingregs',60,0,'uint16');
            %valvulas a 0
                write(m,'holdingregs',44,0,'uint16');
                write(m,'holdingregs',45,0,'uint16');
                write(m,'holdingregs',46,0,'uint16');
            %Caudal entrada T3 a 0
                write(m,'holdingregs',57,0,'uint16');
            %PV a 0
                write(m,'holdingregs',49,0,'uint16');
                write(m,'holdingregs',51,0,'uint16');
                write(m,'holdingregs',53,0,'uint16');
            %Caudales de salida a 0
                write(m,'holdingregs',55,0,'uint16');
                write(m,'holdingregs',56,0,'uint16');
                write(m,'holdingregs',58,0,'uint16');
            %Cierro programa
                break;
        end

    %Espero hasta tiempo de muestreo
    while toc<T
    end
    end
```