UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN DE CONTROL DE PROCESO INDUSTRIAL SOBRE PANTALLA HMI

INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA

TRABAJO FINAL DE GRADO

Adil Bouchakri Zaamy / Adil.bouchakri@estudiantat.upc.edu

Junio del 2021

Director del TFG: Fernandez Sobrino, Ángel
Co-Director del TFG: Delgado Prieto, Miguel

Yo, Adil Bouchakri Zaamy, declaro que el trabajo realizado en esta tesis de grado es completamente mi propio trabajo, que ninguna parte de esta tesis de grado se ha tomado del trabajo de otras personas sin darles crédito y que todas las referencias se han citado claramente.

Entiendo que una infracción a esta declaración me deja sujeto a acciones disciplinarias por parte de la Universitat Politècnica de Catalunya - BarcelonaTECH.

Adil Bouchakri Zaamy		29/06/2020
Nombre del Estudiante	Firma	Data

Título de la Tesis: DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN DE CONTROL DE PROCESO INDUSTRIAL SOBRE PANTALLA HMI.

SUMARIO

SUMARIO FIGURAS

SUMARIO TABLAS

1.Introducción

1.1 Abstract

En este proyecto se ha realizado el diseño e implementación de una aplicación de control de proceso industrial sobre pantalla HMI en el laboratorio de automatización ubicado en la Universidad Politécnica de Cataluña (UPC) de Terrassa.

Los distintos sistemas de control que diseñaremos y del cual partiremos para hacer nuestro proyecto son los siguientes:

- Lazo abierto

Figura 1. Lazo abierto

- Lazo cerrado sin control PID

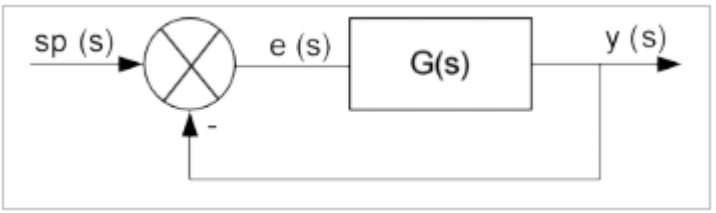


Figura 2. Lazo cerrado sin control PID

- Lazo cerrado con control PID

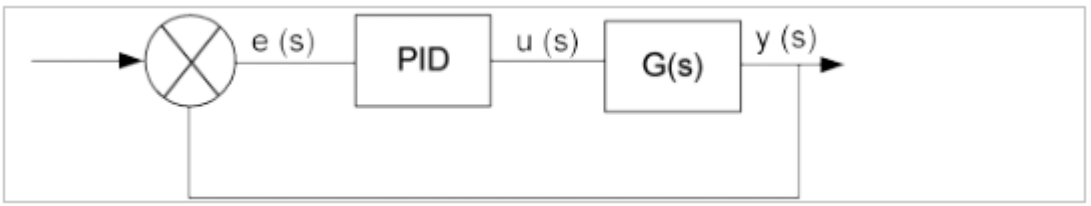


Figura 3. Lazo cerrado con control PID

El proyecto consta, por un lado, de la programación del PLC, dónde se situarán los distintos tipos de lazo, que recibirá distintas instrucciones (Marcha, Paro, Consigna, Tipo de lazo, etc ...) de una aplicación HMI y por el otro, la lectura del estado de la planta y actuación sobre ella mediante Modbus, utilizando el entorno Node-Red.

In this project, the design and implementation of an industrial control process application on HMI screen has been carried out in the automation laboratory located at the Polytechnic University of Catalonia (UPC) in Terrassa.

The different control systems that we will design and from which we will start to carry out our project are the following:

- Open loop system:

Figura 1 bis. Open loop system

- Closed loop without PID control

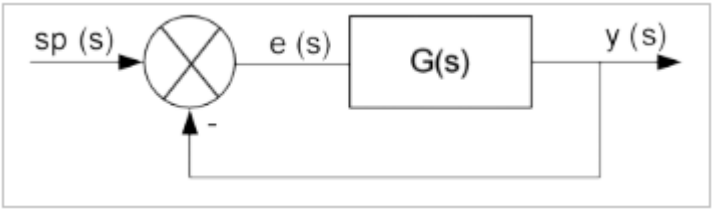


Figura 2 bis. Closed loop system without PID control

- Closed loop with PID control

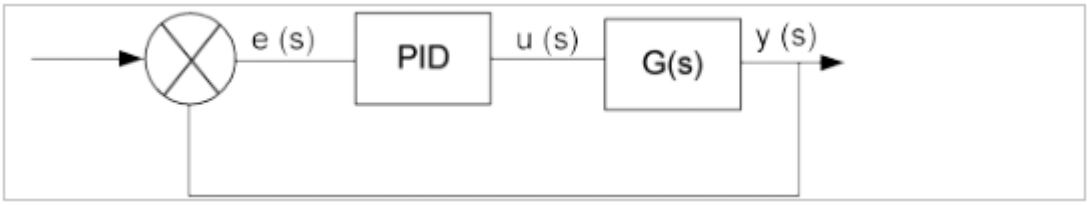


Figura 3 bis. Closed loop with PID control

The project consists, on one hand, of the programming of the PLC, where the different loop types will be located, which will receive different instructions (Start, Stop, Setpoint, Type of loop, etc ...) from an HMI application. And through the other, reading the status of the plant and acting on it through Modbus, using the Node-Red environment.

1.2 Objetivo

El objetivo principal de este proyecto es el diseño y puesta en marcha de de una aplicación soportada por una interfaz gráfica para tareas de control de un proceso industrial. Para lograr dichas tascas necesitaremos estudiar y comprender el funcionamiento de los distintos sistemas de control aplicados.

Uno de los factores más importantes a tener en cuenta son las comunicaciones, almacenaje y visualización de datos. El enviar y recibir datos constantemente será uno de los puntos a analizar en este proyecto ya que la planta de nuestro sistema estará situada en un entorno de simulación digital (Node Red), donde podremos realizar diferentes tipos de pruebas.

Para lograr el objetivo principal, una vez comprendido el funcionamiento del sistema de control, necesitaremos hacer el estudio de las siguientes etapas: conceptualización de la estructura de comunicación necesaria, puesta en marcha del dispositivo HMI, diseño de la aplicación, implementación y validación del sistema de control.

1.3 Alcance

Para conseguir realizar el objetivo marcado, en este proyecto se realizará una serie de tareas, que definirán el alcance. En primer lugar, es necesario buscar información y hacer un repaso sobre la teoría de control, automatización y comunicaciones necesarias para el proyecto.

Comprender la estructura de nuestro sistema de control, entender y relacionar sus distintas partes con los diferentes dispositivos y aplicaciones necesarias es un punto muy importante.

Una vez comprendidos estos conceptos procedemos a realizar las siguientes tareas:

- Discretización de un sistema de primer orden, en este caso aprovecharemos una ecuación obtenida en la Práctica 9 (realizada en la UPC de Terrassa).
- Definir la arquitectura de programación de los distintos sistemas de control mediante instrucciones del PLC.
- Complementar el programa de control con una aplicación HMI mediante la pantalla MAGELIS.
- Entender y revisar los buses de comunicaciones entre la pantalla HMI - PLC y PLC-NodeRed. Definir la arquitectura y el flujo de información.
- Utilizar el entorno Node-RED para implementar la planta y comunicaciones necesarias, además de aprovechar este entorno para la visualización de los resultados mediante el dashboard.
- Realizar un ensayo sobre la comunicación entre “planta”-PLC y estudiar el comportamiento de las dinámicas de comunicación y planta..

1.4 Justificación

La elección de este trabajo nace de un interés propio por las asignaturas de automatización que se cursan en el grado de Ingeniería Electrónica Industrial y Automática en la UPC de Terrassa. Con este proyecto, desde un punto de vista personal, pretendía hacer un repaso y aplicar los distintos conocimientos que adquirí al cursar el grado mencionado anteriormente, además de profundizar y aprender más sobre este ámbito de la Automática.

Desde un principio mi idea era hacer un estudio enfocado principalmente en la programación Ladder, pero se me ofrece un trabajo que abarca mucho más, desde discretizar procesos de control de primer orden mediante instrucciones del PLC y realizar una aplicación HMI, hasta estudiar el comportamiento de las comunicaciones entre PLC y Node Red.

Este proyecto supone una gran oportunidad de aprendizaje, ya que me ofrece adentrarme en el mundo de los PLC's y los distintos protocolos de comunicación que permiten la conexión con diferentes dispositivos y softwares, además de ver el comportamiento de los sistemas de control aplicados a este ámbito.

Por otro lado, aparte de tener un motivo de aprendizaje propio, la finalidad de este proyecto es crear un sistema de muestreo digital provisto por un sistema de control por PLC. Hoy en día las empresas optan por este tipo de control por la facilidad y la optimización que ofrecen a sus operaciones. El poder modificar parámetros operativos mediante una pantalla táctil, seguir el estado del sistema y obtener máxima información de los procesos, visualizar condiciones de alarma y modificar los parámetros del sistema de toma de muestras a través de un software sin necesidad de un hardware específico ni personal, permiten una notoria optimización de recursos, tiempo y actividad laboral entre otras razones.

Estas distintas razones me han dado la motivación necesaria para seleccionar este proyecto y querer aprender aún más sobre este mundo de la automática.

2. Consideraciones previas

A lo largo de este apartado se introducirán todos los elementos básicos y necesarios para la construcción del proyecto. Se ha de tener una base teórica sólida para la realización de los puntos más importantes. Para ello, se introducirá el concepto de la Pirámide CIM, que resume la importancia y la necesidad de clasificar los distintos dispositivos en los procesos industriales. En dicha pirámide se podrá relacionar los elementos de este proyecto y nos permitirá estructurarlo de manera firme.

2.1 Pirámide CIM

A lo largo del proceso de producción, las comunicaciones industriales nos permiten tener un flujo de información entre el controlador y los diferentes dispositivos. Teniendo en cuenta los distintos y numerosos sistemas de comunicación, se ha optado por la creación de una pirámide jerárquica con la que se podrá implementar los protocolos necesarios hasta conseguir una comunicación completa, partiendo del nivel físico hasta llegar al nivel más alto de red.

La pirámide mencionada anteriormente, se le denomina “La pirámide de automatización CIM (Computer integrated Manufacturing) “, se trata de un sistema piramidal donde se agrupan jerárquicamente todos los equipos, dispositivos, sensores, etc…, involucrados en los procesos de fabricación, aspirando a una optimización de la producción, actividad laboral, mejorar la calidad y conseguir la mayor eficiencia posible en las diferentes secciones de la empresa.

La pirámide está formada por cinco niveles, donde cada subsistema de cada nivel ha de tener comunicación directa con los subsistemas del mismo nivel, de los niveles inferiores y de los superiores, ayudando así a la integración de los procesos de producción con los de gestión.

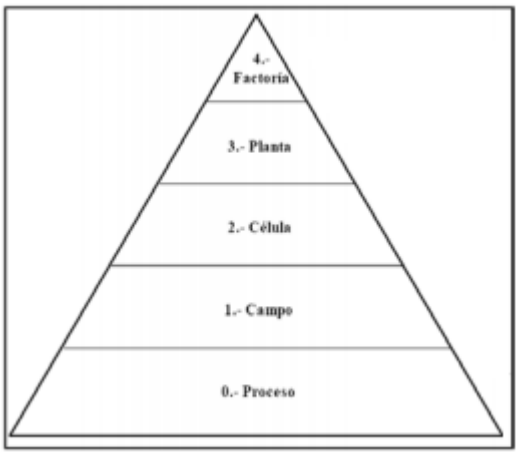


Figura 4. Pirámide CIM

- **Nivel 0: Nivel de Proceso**

  En este nivel se realiza todo el trabajo físico, donde se ejecutan las órdenes de los elementos que controlan el proceso. Este trabajo se realiza mediante distintos dispositivos, tanto de medida usando sensores, como de mando a través de actuadores (motores, válvulas, actuadores hidráulicos y neumáticos). Una de las características de los sensores y actuadores es que suelen ser dispositivos que han de ser controlados por otros elementos, por ello son elementos que se involucran directamente con el proceso de producción.

- **Nivel 1: Nivel de Campo**

  En este nivel se hallan los distintos elementos de control automático, que se encargan de gestionar los dispositivos del nivel 0 (sensores y actuadores) proporcionándoles información para realizar ciertas tareas, a la vez que informan del estado del proceso al nivel 2. Entre estos elementos nos encontramos con los PLC'S (controladores lógicos programables), Robots o UTR (unidades terminales remotas), DCS (sistemas de control distribuïdos) y tarjetas de control o los PAC (controladores de automatización programables).

- **Nivel 2: Nivel de Célula**

  El nivel de célula se encarga de, una vez que recibe órdenes de los niveles superiores, lanzar órdenes de ejecución de manera sencilla, a los dispositivos del nivel 1, además de recoger la información del proceso a través de dicho nivel.
  En este nivel encontramos SCADA (Supervisory Control And Data Adquisition), un sistema que se encarga de coordinar el proceso de manera sincronizada. Suele tener incorporado un HMI para el control de las funciones de manera remota.

- **Nivel 3: Nivel de Planta**

  Este nivel se realiza la monitorización de todo el proceso de fabricación a través de ordenadores o pantallas industriales, permitiendo así observar la información de la planta y así poder prevenir los posibles fallos o modificar cualquier proceso necesario para obtener el resultado deseado. Además de poder optimizar los recursos basándose en los resultados e información obtenidos.

- **Nivel 4: Nivel de Fábrica o Gestión**

  El último nivel de la pirámide, como su pròpio nombre indica, se trata de una gestión de la producción completa de la empresa, que se conoce como planificación de recursos empresariales (ERP). Básicamente se trata de un nivel donde se intercambia información en volúmenes muy altos y a tiempos justos, por lo que las comunicaciones han de ser las mejores (Ethernet con protocolos TCP/IP) para obtener una gran eficiencia en la monitorización de los distintos niveles.

2.2 Controladores lógicos programables

2.2.1 Definicion

Un PLC (Programmable Logic Controller) se trata de un dispositivo programable con la función de automatizar procesos industriales mediante el control de los dispositivos del nivel de proceso (sensores y actuadores) y enviando la información necesaria a los niveles superiores, según la jerarquía de la pirámide CIM estudiada en el apartado anterior, donde el PLC se encuentra situado en el nivel de campo.

Los PLC 'S son dispositivos de alta fiabilidad y aportan una serie de ventajas y magnitud de usos. Entre ellos el poder comunicarse con múltiples dispositivos a través de sus módulos de entrada para recoger la información recibida de los sensores, procesar dicha información y hacer un control sobre los actuadores, a través de los módulos de salida, modificando los parámetros necesarios mediante un panel de operador.

Por otro lado, los PLC 'S están diseñados para poder aguantar condiciones adversas y se pueden programar con bastante facilidad mediante lenguajes de programación sencillos y comprensibles. Estos aspectos mencionados hacen del PLC un dispositivo imprescindible en el entorno industrial.

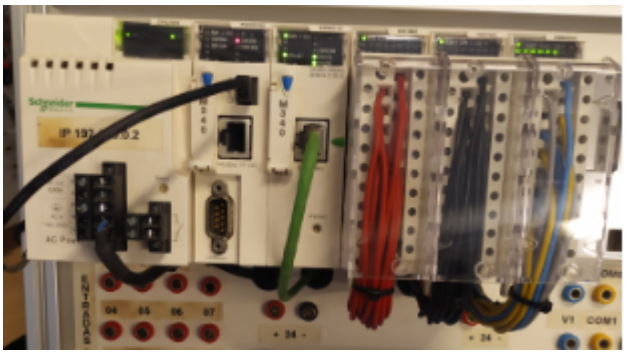


Figura 5. PLC

2.2.2 Estructura

Existen distintos tipos de PLC 'S dependiendo de los diferentes módulos que lo forman y de sus funciones, capacidad, memoria, etc.... . Pero la estructura básica de un PLC se puede resumir en los siguientes componentes:

- **Fuente de alimentación:**

  Se trata de un elemento imprescindible para el PLC, básicamente porque se encarga de suministrar la potencia necesaria para que el sistema funcione, dicha potencia puede variar entre ±5V, ±12V y ±24V, que son los valores más frecuentemente utilizados.

- **CPU:**

  La CPU (unidad central de procesamiento) se encarga de procesar las distintas instrucciones del PLC, además de realizar un testeo frecuente del plc, permitiendo encontrar y prevenir los errores que puedan haber. La CPU se compone de un microprocesador que a través de operaciones lógicas permite realizar distintas funciones.

- **Módulos de entrada / salida:**

  Los módulos de E/S se encargan básicamente de controlar las distintas señales de entrada y salida de un sistema con las del PLC. Estos módulos pueden ser analógicos o digitales dependiendo de la necesidad del operador. Los módulos de entrada se encargan de recoger la información a través de los sensores y enviarla a la CPU para procesarla según el programa definido, una vez procesada, se envía al módulo de salida que se encarga de controlar los actuadores para que realicen las acciones necesarias.

- **Módulo de Memoria:**

El módulo de memoria se encarga básicamente de guardar las instrucciones y datos de manera provisional o permanente y su capacidad de almacenamiento pueden ser de distintos tamaños (16, 32 128, 256 Kb...). Existen dos tipos de memoria:

- **Memorias volátiles (RAM)**

  Una memoria volátil es aquella que necesita de un suministro de energía constante a través de una fuente de alimentación, una vez se deja de proporcionar dicha energía, se pierde la información .
  La memoria RAM se encarga de almacenar los programas, ejecutarlos y durante su elaboración se le permite al usuario modificarlo constantemente.

- **Memorias no volátiles (EPROM y EEPROM):**

  Una memoria no volátil es aquella que conserva la información aun cuando no se le proporciona energía. Se caracteriza por ser una memoria solo de lectura además de que el borrado y la programación se puede realizar eléctricamente.

2.2.3 Ciclo de funcionamiento del PLC

La mayoría de PLC 'S cuentan con un funcionamiento cíclico y secuencial, es decir, mientras el autómata recibe potencia de la fuente de alimentación, las instrucciones y operaciones tendrán lugar y se repetirán continuamente una tras otra de manera cíclica. A este proceso se le denomina ciclo de SCAN.

Una vez conectado el PLC se realiza un autodiagnóstico para verificar el correcto funcionamiento de los circuitos que lo integran, esto ocurre de manera rápida y en caso de error el PLC lo comunica al usuario a través de una señal luminosa. Después del diagnóstico se realiza una lectura de las entradas para determinar cuál de ellas se encuentran encendidas o apagadas, dicha información se envía a la CPU y se guarda en la memoria, seguidamente la CPU ejecuta el programa creado por el usuario de manera secuencial, siguiendo el orden determinado y se generando nuevas señales de salida. Finalmente se actualiza el registro de salidas de manera simultánea y se envían a los actuadores para realizar la tarea determinada. Al final de cada ciclo del PLC, se inicia uno nuevo.

2.3 SCADA

El SCADA (Supervisory Control And Data Acquisition) se trata de un software instalado en un ordenador (Máster o MTU), conectados con distintos dispositivos (PLC Y RTU) permitiendo un control y automatización industrial en los procesos productivos. Este software se encuentra en el nivel de célula según la jerarquía de la pirámide CIM, donde se reciben las

órdenes de los niveles superiores y a través del SCADA se realiza un almacenamiento de datos, supervisión, control de calidad, etc. .

Figura 6. Pantalla HMI

Un sistema SCADA suele estar formado por los siguientes componentes:

- **HMI (Human Machine Interface):**

  Se trata de una interfaz gráfica que conecta al operario con la máquina presentando los datos del proceso. Las pantallas HMI permiten optimizar el proceso industrial mediante la digitalización y organización de datos. A través de este panel de control el operario podrá realizar las funciones de control, controlar y observar el proceso y las respuestas que presenta a tiempo real.

- **Sistema de supervisión o MTU:**

  Es un sistema de supervisión con la función de recopilar información leyendo los datos mesurados del proceso y enviar las instrucciones necesarias mediante comandos.

- **Unidades Terminales Remotas (RTU):**

  Las RTU con microprocesadores conectados remotamente que permiten leer el estado de una acción, enviar la información para que se procese y convertir las señales recibidas en datos digitales que se enviarán al sistema de supervisión.

- **PLC** (Descrito en el apartado anterior *"2.2Controladores lógicos programables"*)

  Autómatas programables.

- **Sistemas de comunicación:**

  Se encargan de conectar todos los componentes del SCADA.

2.4 Protocolos de comunicación

El bus de campo es un sistema de transmisión de información desarrollado en los '90 por un solo cable con la intención de simplificar las instalaciones de máquinas industriales.

2.4.1 Modelo de Comunicaciones

- **Maestro/Esclavo:**

  Se refiere a una relación de comunicación donde el maestro inicia y controla una sesión con el/los esclavo/s. Se trata de un esquema pregunta-respuesta donde el maestro determina la dirección del tráfico de datos y la temporización de este, mientras que el esclavo sólo tiene la función de resolver, no puede empezar ni una comunicación ni temporizar.

- **Cliente/Servidor:**

  La relación Cliente/Servidor es una relación donde el servidor da soporte a los clientes con los que está conectado, dicho de otra manera, el cliente se encarga de hacer una petición de servicio al servidor, este realiza el servicio pedido y devuelve la información en forma de respuesta.

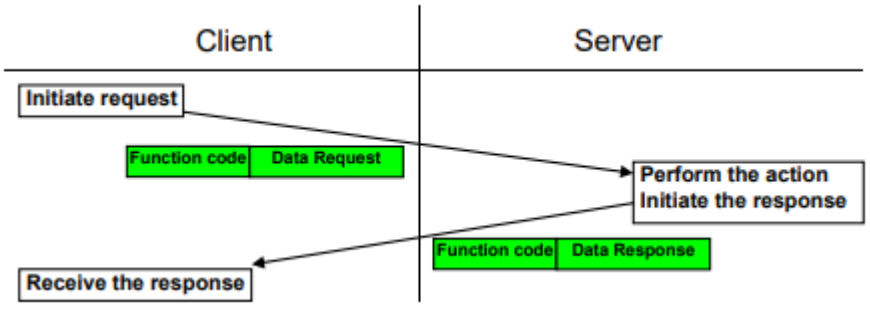


Figura 6. Ejemplo modelo de comunicación cliente/servidor

- **Productor/Consumidor:**

  La relación del Productor/Consumidor se basa en la relación maestro-esclavo. Se trata de un proceso sincronizado donde el productor almacena datos en un buffer y el consumidor a su ritmo va recogiendo y usando estos datos, permitiendo así un almacenamiento temporal de los datos, ya que cada uno trabaja a su ritmo.

2.4.2 Protocolo CAN

El sistema Control Area Network fué desarrollado por Bosch en el año 1986 con la finalidad de reducir los hilos conductores de los vehículos. Se trata de un bus multi-master que permite conectar todo tipo de dispositivos y aplicaciones inteligentes sin necesidad de un host. Este protocolo permite descomponer la información recibida en mensajes para enviarlos por tramos una vez procesados, es bidireccional, cada nodo del sistema es independiente, es decir, eliminar uno no supondrá un problema al sistema CAN, ya que el resto seguirán interconectados, además de que los mensajes cuentan con orden de prioridad. En este sentido, gracias a sus características, es un protocolo necesario para coordinar las infinitas funcionalidades que se necesitan en la industria.

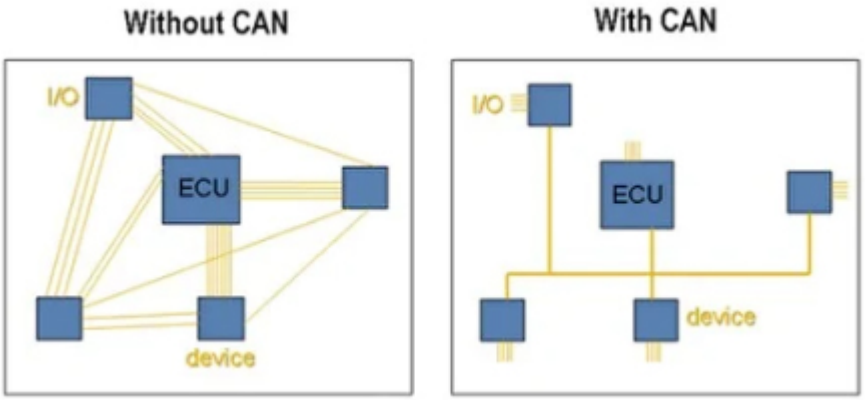


Figura 7. Protocolo CAN

Las velocidades de transmisión de este protocolo van de 50 Kbps a 1 Mbps con un volumen de información de 60 bits de datos de usuario.
Por otro lado, basados en este protocolo CAN, podemos encontrarnos con el bus de campo CANOpen soportado en el campo de la fabricación de maquinaria. Y el DeviceNet con el que es posible la conexión de 64 nodos como máximo con velocidades de 125 a 500 Kbps.

2.4.3 Protocolo Modbus

Se trata de un protocolo de comunicación abierto de solicitud-respuesta basado en la arquitectura maestro/esclavo (RTU) o cliente/servidor (TCP/IP), es decir que la comunicación se hace en pares, el maestro inicia una solicitud y espera una respuesta del esclavo. Este protocolo fue creado por Modicon a finales de los ‘70 para la comunicación entre los PLC ‘S y actualmente se ha convertido en uno de los protocolos más usados en el ámbito industrial, ya que el uso del protocolo está libre de derechos (gratuito y público), es fácil de implementar, maneja datos sin suponer restricciones y permite una comunicación sencilla con distintos tipos de dispositivos y aplicaciones.

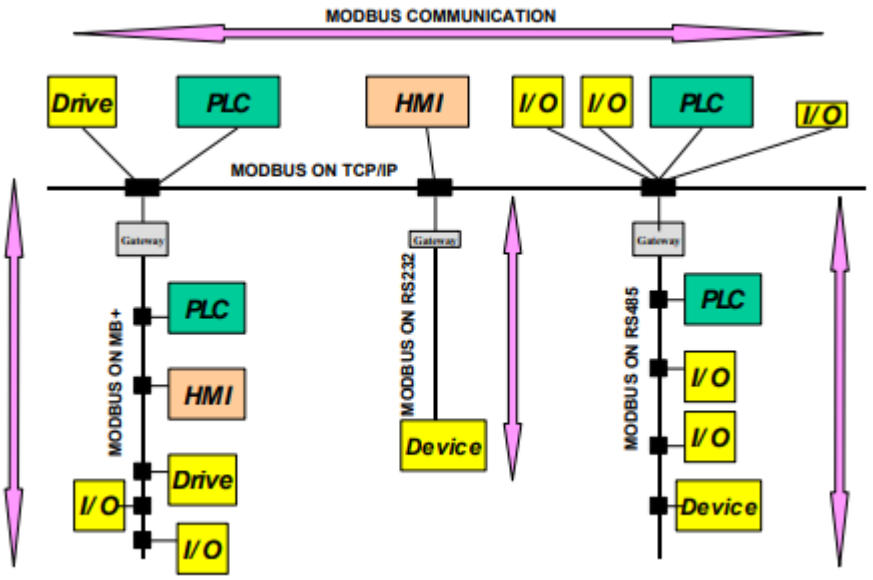


Figura 8. Ejemplo comunicación Modbus

En una red modbus estándar podemos encontrar un maestro y hasta 247 esclavos, cada uno con una dirección única y un intercambio de datos de hasta 1200 m de distancia, además de usar los protocolos RS232/RS485/RS422, unos protocolos de transferencia de datos para el control de la comunicación serie, que aportan una gran simplicidad y restringe la cantidad de esclavos necesarios en la transmisión, ya que cuantos más esclavos más lenta es la comunicación.

La interfaz de comunicación Modbus se crea alrededor de mensajes que son independientes del tipo de interfaz física, permitiendo actualizar fácilmente la estructura del hardware industrial sin necesidad de hacer cambios en el software. Cada mensaje tiene la misma estructura de cuatro elementos básicos para facilitar el análisis de su contenido. El formato de los mensajes varía dependiendo del protocolo Modbus usado. La estructura básica de la trama modbus está compuesta por una ADU (Unidad de Datos de Aplicación), que incluye una PDU (Unidad de Datos de Protocolo).

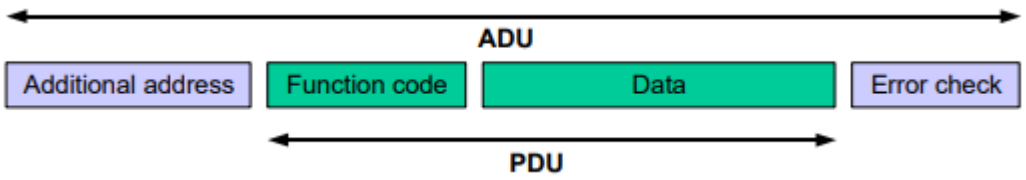


Figura 9. Formato de los mensajes Modbus

- **Additional address (Dirección de la estación) :**

  Indica el destino de la solicitud y su longitud varía dependiendo del tipo de variante modbus se esté utilizando, para RTU 8 bits y 2 Bytes para ASCII, con direcciones

válidas de 0-247, siendo el 0 una dirección de difusión y del 1-247 los esclavos activos.

- **Function code (Código de función):**
  El código de función es principalmente la solicitud del maestro que informa de la dirección y el tipo de acción que el esclavo ha de realizar. Por ejemplo, el código de función *04* es una solicitud donde el esclavo ha de leer registros de entrada.

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**		**14**	6.14
			Write File record	**21**		**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Server ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19
			CANopen General Reference	**43**	**13**	**2B**	6.20

Figura 10. Código de función

- **Data (Datos):**

  El apartado de “Data” contiene la información de solicitud por parte del maestro para realizar cierta acción o la respuesta por parte del esclavo.

- **Error check (Verificación de redundancia):**

  Existen dos tipos de verificación de error dependiendo del modelo Modbus usado.
  Por un lado, se usa el modo RTU para el entramado de caracteres. El campo de verificación de errores contiene un valor entre 16 bits usado como dos bytes de 8 bits. Para el error check se usa la verificación de redundancia cíclica (CRC) que se implementa en el último campo del mensaje, al realizar esto, el byte de orden inferior

del campo se añade al primero, seguido por el byte de orden superior (el último que se envía en el mensaje).
Por otro lado, en el modo ASCII, el campo de comprobación contiene dos caracteres ASCII. Para el error check se usa la verificación de redundancia longitudinal que se realiza en el contenido del mensaje.

2.4.4 Protocolo Modbus TCP/IP

Modbus TCP/IP es una variante de la familia Modbus de protocolos de comunicación para la supervisión y control de equipos de automatización. Se trata de un protocolo de Modbus RTU con una interfaz TCP que se ejecuta en Ethernet. TCP/IP se trata del protocolo de internet y de control de transmisión que proporciona el medio de transmisión de mensajes Modbus TCP/IP, además de permitir un gran número de conexiones simultáneas.

Modbus TCP/IP permite mantener el control de los datos a través de una conexión que pueda ser identificada,supervisada y cancelada sin que el cliente o servidor tomen una acción específica, dàndole a este protocolo una gran tolerancia a los cambios de rendimiento de la red, y permite la aplicación de elementos de seguridad como son los cortafuegos y servidores proxy.

TCP/IP permite a los ordenadores que se intercambien bloques de datos binarios. La función de TCP es asegurar que se reciban sin ningún problema o error los paquetes de datos, mientras que IP tiene la función de asegurar que los mensajes sean abordados y colocados correctamente. En definitiva, Modbus TCP/IP permite combinar la red física Ethernet con la red estándar (TCP/IP) y con el protocolo de aplicación Modbus con la finalidad de transportar los datos entre dispositivos compatibles.

La estructura trama de la comunicación Modbus TCP/IP es la siguiente:

MBAP	Función	Datos
7	1	n

Figura 11. Estructura trama de la comunicación Modbus TCP/IP

El encabezado MBAP (Modbus Application Protocol) tiene un tamaño de 7 bytes, y esta compuesto de lo siguiente:

- **Transaction identifier**: Se encarga de la sincronización de los mensajes entre cliente y servidor. Tiene una longitud de 2 bytes.

- **Protocol**: El valor 0 indica Modbus TCP/IP y consta de 2 bytes.

- **Length**: Se trata de la cantidad de bytes en la trama. Tiene una longitud de 2 bytes.

- **Unit Identifier**: Es utilizado para la identificación del esclavo remoto en una red Modbus RTU. Tiene una longitud de 1 bytes.

Para completar la trama, como observamos en su estructura, por un lado, se encuentra el campo llamado "función", que como ya hemos explicado en el apartado de Modbus, indica el código de función y cuenta con una longitud de 2 bytes, por el otro lado y como último campo de esta trama, se encuentran los "Datos", el cual contiene la información de solicitud por parte del maestro y cuya longitud depende del tipo de mensaje.

2.4.5 Protocolo Ethernet

Ethernet, conocido como Acceso Múltiple con Escucha Portadora y Detección de Colisiones (CSMA/CD), es un estándar de redes de área local para computadoras, el cual según la norma IEEE 802.3 basada en la Ethernet de Xerox se ha convertido en el método con más abasto para las interconexiones entre computadores en redes de proceso de datos. Ethernet se basa en el cableado y señalización del nivel físico y el formato de tramas de datos del nivel de enlace de datos según el modelo OSI.

En diversos buses de campo como Profibus y Modbus se ha adoptado Ethernet como red apropiada para los niveles superiores, buscando en ello un soporte para las comunicaciones industriales.

El formato de trama Ethernet es el siguiente:

Preámbulo	Delimitador de inicio de trama	MAC de destino	MAC de origen	Etiqueta	Ethertype	Payload	CRC	Gap
7 Bytes	1 Byte	6 Byte	6 Bytes	(4 Bytes)	2 Bytes	De 46 (o 42) hasta 1500 Bytes	4 Bytes	12 Bytes

Tabla 1. Formato de trama Ethernet

- Al primer campo se le denomina **preámbulo**, se encarga de marcar el inicio de la trama y tiene el objetivo de que el dispositivo que lo recibe detecte una nueva trama y se sincronice.
- El frame empieza a partir del **Delimitador de inicio de trama**.
- El campo de **MAC de destino** indica la dirección física del dispositivo que recibe los datos.

- El campo de **MAC de origen** indica la dirección física del dispositivo de origen de los datos.
- El campo de **Etiqueta** es opcional e indica si la red pertenece a una VLAN o prioridad.
- **Ethernetype** es un campo que indica el protocolo con el que están encapsulados los datos que contiene el payload en el momento en que se use un protocolo de capa superior.
- El campo de **Payload** contiene todos los datos de la red o cabeceras si se trata de protocolos de capas superiores. Las tramas enanas (mensajes inferiores a 64 Bytes) informan de si hay mensajes dañados y parcialmente transmitidos.
- El campo de **CRC** se refiere a un campo de secuencia de comprobación mediante un verificador de control de redundancia cíclica (CRC). Si después del cálculo del CRC de toda la trama, el valor es 0 la trama es válida.
- Finalmente, el último campo denominado **Gap** ofrece un espacio entre tramas.

El protocolo Ethernet/IP es un estándar de red de comunicación en niveles para aplicaciones de automatización industrial con la capacidad de manejar cantidades de datos grandes a velocidades de 10 Mbps hasta 1500 bytes por paquete. Este protocolo se basa en los protocolos estándar TCP/IP y es muy importante para los sistemas de automatización y control ya que es un tipo de red fácil de configurar, operar, amplia, que cuenta con una gran fiabilidad y rendimiento inherente.

2.5 Software de Programación Unity Pro XL

Unity Pro XL se trata de un software de programación y configuración de autómata para la ejecución de cierto programa. Unity admite las plataformas hardware Modicon M340, Premium, Atrium y Quantum.

El software IEC 61131-3, Unity Pro XL permite obtener una mayor productividad y facilidad gracias a su amplio conjunto de nuevas funcionalidades. IEC 61131-3 define los lenguajes de programación más corrientes y define los estándares los lenguajes gráficos y textuales para PLC 'S.

2.5.1 Lenguajes de programación

Unity Pro proporciona lenguajes de programación diversos para la creación del programa del usuario. Dichos lenguajes cumplen con la norma IEC 61131-3. Dependiendo de las necesidades del usuario se puede elegir el lenguaje más óptimo o, si es necesario, se pueden utilizar juntos para la construcción de un mismo programa.

Diagrama de contactos (LD)

El diagrama de contactos o lenguaje ladder es un lenguaje de programación gráfico que permite una fácil adaptación del usuario gracias a su estructura basada en esquemas eléctricos de control clásicos.

La estructura de una sección LD corresponde a un circuito de conmutadores relé. El lado izquierdo del editor LD corresponde a la barra de alimentación (conductor L de un circuito corriente) a la que han de ir conectados los distintos objetos LD. En el lado contrario se encuentra la barra de alimentación derecha a la que se conectan tanto directa o indirectamente las bobinas y salidas FFB, de esta manera el conjunto de la conexión de estos elementos se llega a establecer un flujo de corriente.

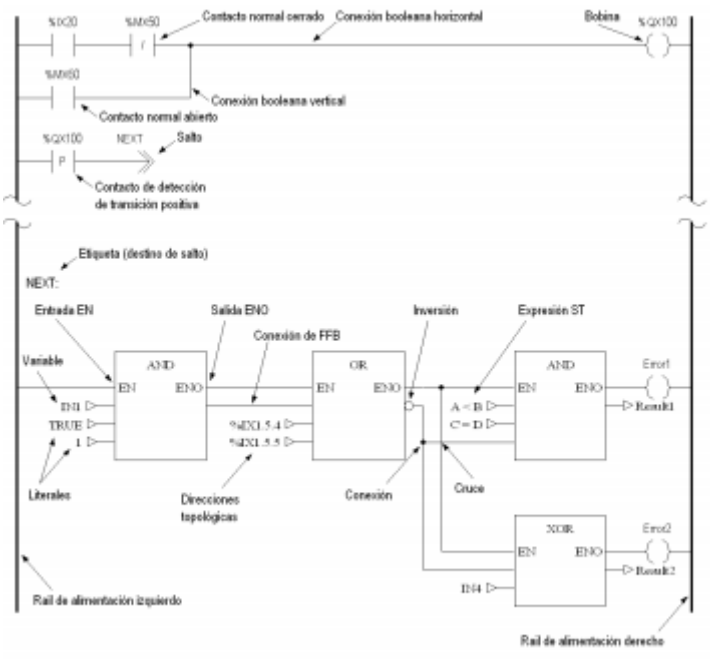


Figura 12. Representación diagrama de flujo ladder

Diagrama de bloques de funciones (FBD)

El diagrama de bloques de funciones así como el lenguaje LD, permite la programación gráfica de los bloques de función, donde las distintas variables de entrada y salida se conectan a estos bloques mencionados mediante líneas de conexión. Los bloques de función están basados en puertas lógicas.

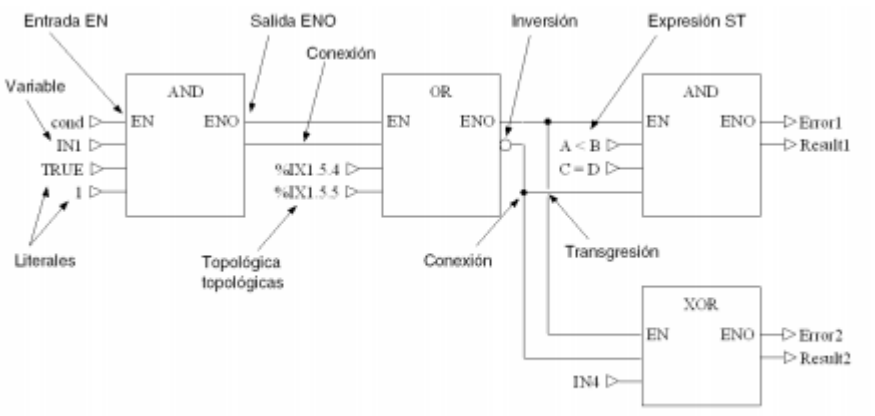


Figura 13. Representación de una sección FBD

Texto estructurado (ST)

El lenguaje de programación de texto estructurado, es un lenguaje de alto nivel que permite estructurar series de instrucciones de manera cómoda y rápida, con la posibilidad de generar documentos en HTML, TeX, docbook. ST permite llamar bloques de funciones, ejecutar asignaciones, ejecutar funciones, ejecutar instrucciones de forma condicional y repetir instrucciones.

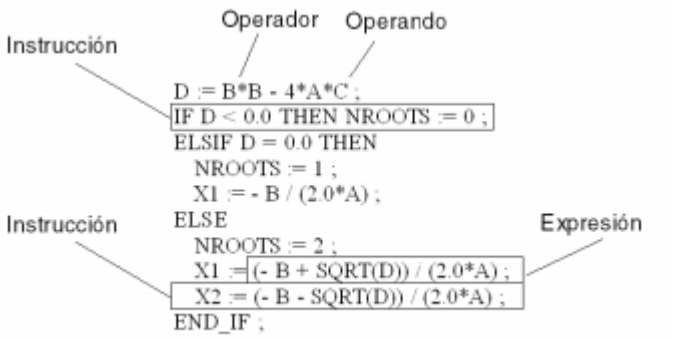


Figura 14. Representación de una sección ST

Lista de instrucciones (IL)

El lenguaje de lista de instrucciones es un lenguaje de bajo nivel bastante parecido al lenguaje ensamblador y está compuesto por una lista de instrucciones que se procesa en el PLC.

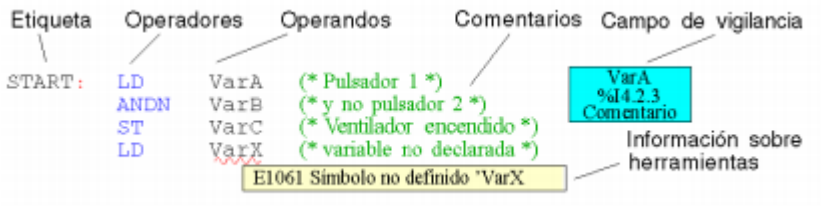


Figura 15. Representación de una sección IL

Bloques de función secuencial (SFC)

Una sección SFC es una "máquina de estado", donde el estado representa el paso activo, y las transiciones el comportamiento de las conexiones. Pasos y transiciones se vinculan mediante conexiones direccionales y no es posible vincular dos pasos directamente sin una transición de por medio. A cada paso le corresponden ciertas acciones o ninguna. A cada transición le corresponde una condición de transición.

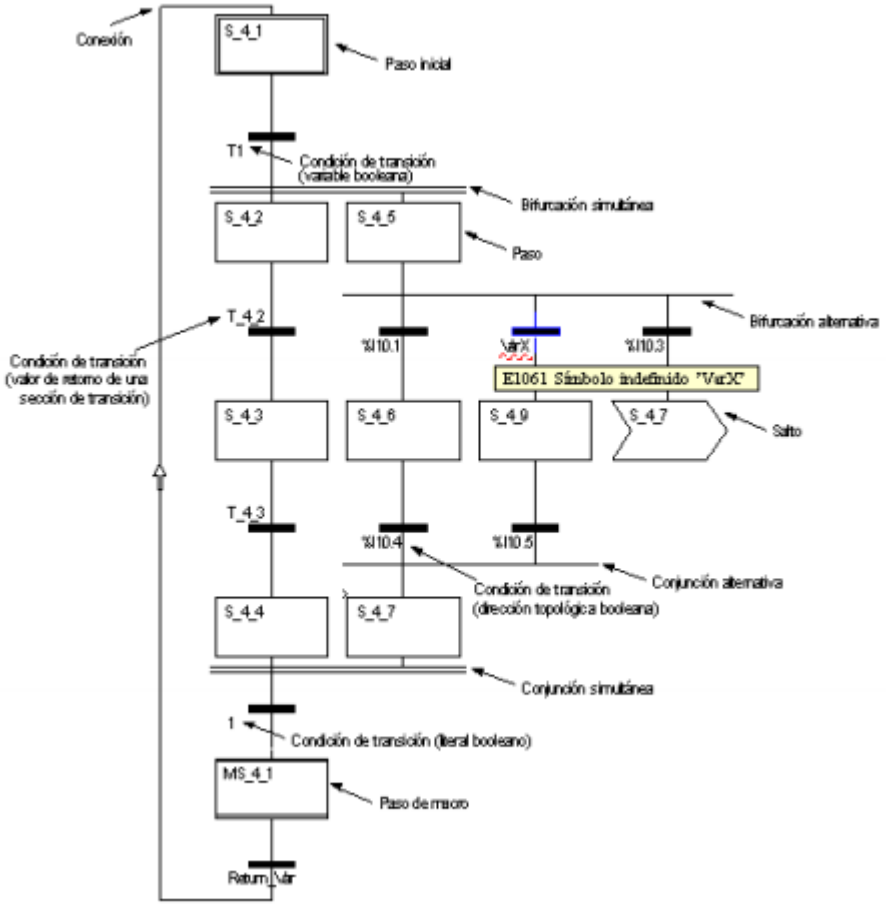


Figura 16. representación de una sección IL

2.5.2 Creación del proyecto

Al crear un nuevo proyecto en Unity, se le ofrece al usuario la selección de una plataforma (M340, Premium, Quantum) y procesador.

Menú Fichero ➔ Nuevo

Figura 17. Nuevo proyecto unity

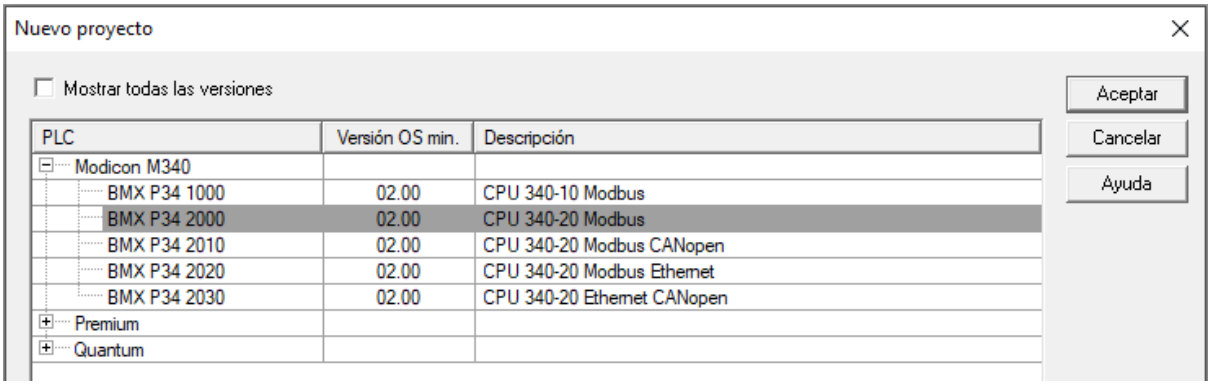


Figura 18. Seleccion de plataforma Unity

Una vez seleccionado la plataforma, se nos presenta la ventana principal del programa:

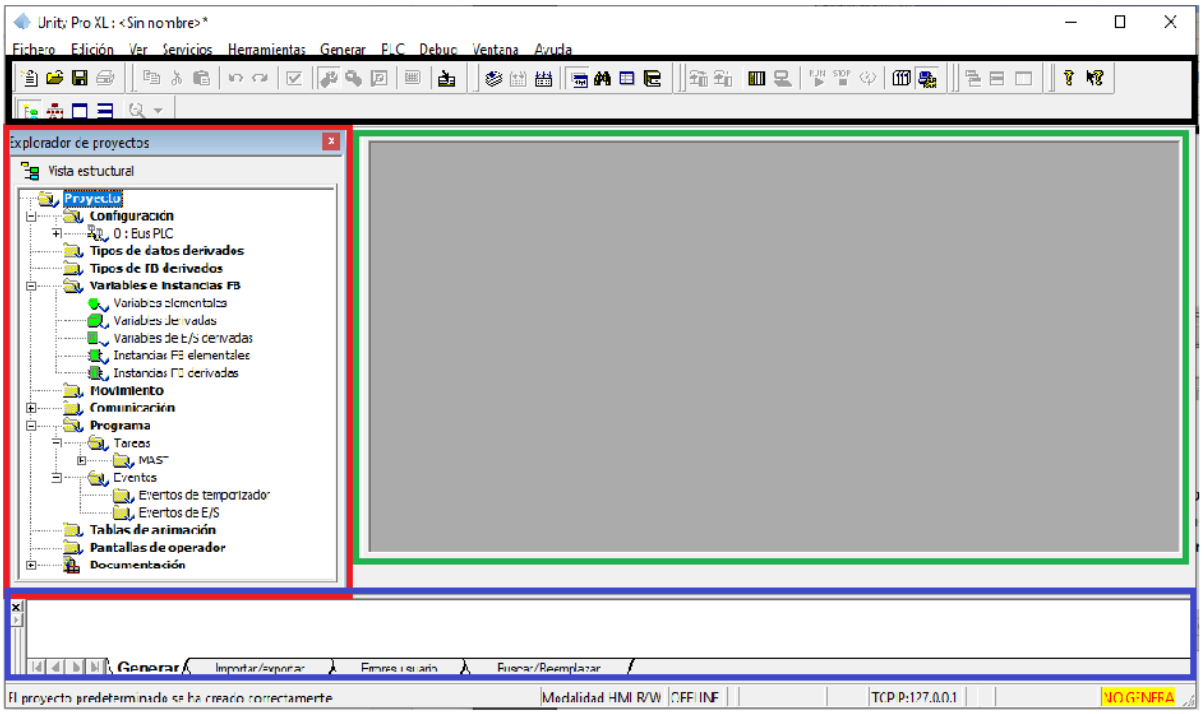


Figura 19. Ventana principal Unity

Donde, como podemos observar, se divide en distintas partes, cada una de ellas con su función.

En el recuadro **negro** se halla el menu e iconos que sirven para acceder a las distintas funciones de Unity.

El recuadro **rojo** o explorador de proyectos representa el árbol del proyecto, es decir, permite acceder y modificar las distintas partes de un proyecto.

En el recuadro **azul** o ventana de resultados podremos ver la información relacionada con las distintas operaciones que realicemos, aparecen todo tipo de errores y advertencias del programa realizado o de otras operaciones.
El recuadro verde se trata de un espacio donde se visualizarán las pantallas de las distintas herramientas, pantallas de conexión, secciones del programa, etc..., y donde se podrán editar de manera sencilla.

2.5.3 Explorador de proyectos

Bastidor

En el apartado de ***Configuración*** del ***Explorador de proyectos*** se puede configurar un PLC modificando el tamaño del bastidor añadiendo o eliminando, si es el caso, los distintos módulos necesarios para el usuario.

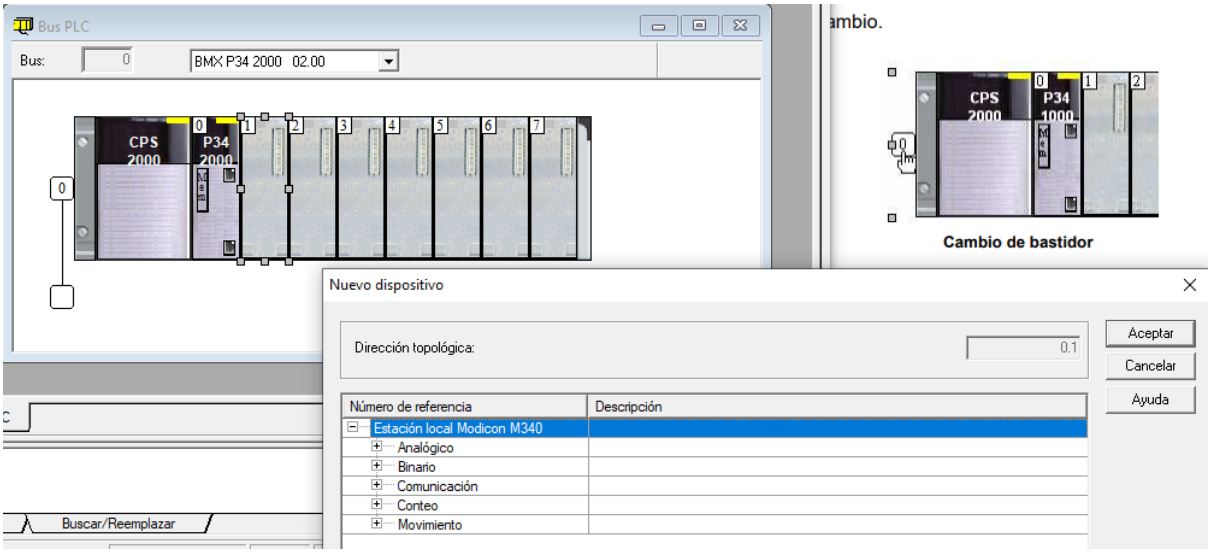


Figura 20. configuración del bastidor

Editor de datos

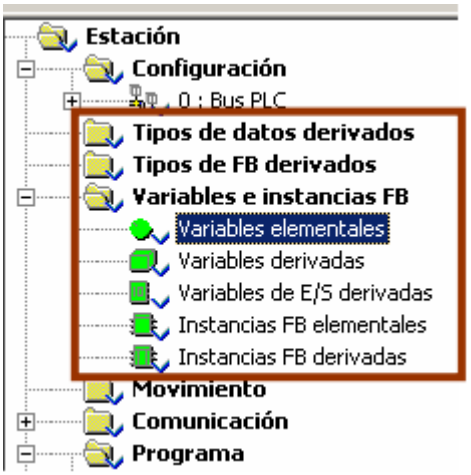


Figura 21. editor de datos

Al seleccionar cualquiera de las opciones marcadas dentro del recuadro podremos acceder al editor de datos donde aparecen 4 pestañas, en dicha pestaña podremos crear /modificar /visualizar distintas variables, datos, instancias y bloques de función de usuario.

Para nuestro proyecto nos enfocaremos en la pestaña de variables:

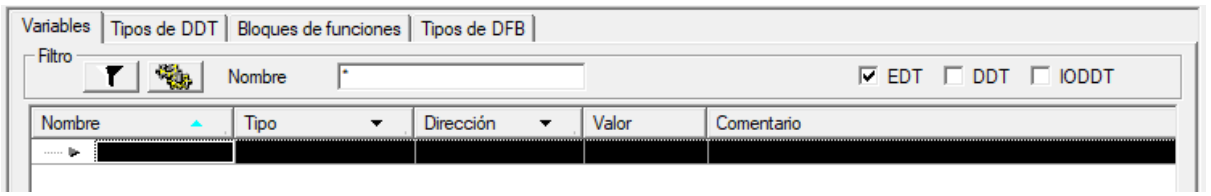


Figura 22. Pestaña de variables

Como hemos indicado anteriormente, la ventana de variables permite crear o modificar variables. Las variables han de tener como mínimo un nombre y un tipo de datos. Es posible añadirle una dirección y un valor por defecto:

- Las variables están formadas por un **Nombre**, el cual una vez introducido, la variable por defecto será de tipo BOOL.

- En el apartado **Tipo** podremos seleccionar el tipo de dato de la variable, los básicos en Unity son:
    - **Integers**(enteros): No tienen parte decimal y pueden ser negativos.
    - **Floats**(decimales): Tienen parte decimal y pueden ser negativos.
    - **Booleans**(valores lógicos): Representan si una sentencia es Verdad(1)/Falso(0).
    - **Strings**(cadenas de texto): Son caracteres y símbolos de escritura.

- En el campo de **Dirección** se puede escribir una dirección física:

Dirección física	Descripción
%I	Entrada digital
%Q	Salida digital
%IW	Entrada analogica
%QW	Salida analogica

Tabla 2. Campo de dirección

o en una dirección de memoria:

Dirección de memória	Descripción
%M (Memory)	Bit de memoria (0 o 1)
%MW (Memory Word)	Palabra de memoria (16 bits)
%KW (Constant Word)	Palabra constante (no se puede modificar durante la ejecución del PLC)
%S (System Bit)	Bit de sistema
%SW (System Word)	Palabra de sistema

Tabla 3. Dirección de memória

- En el campo de **Valor** se puede escribir un valor numérico, cadena de bit, cadena de caracteres… , dependiendo de qué tipo de dato sea la variable.

- En el último campo (**Comentario**) se puede escribir cualquier comentario respecto a una variable

Comunicación

Este apartado permite asignar una dirección IP a un PLC. Primero se ha de crear una conexión de red.

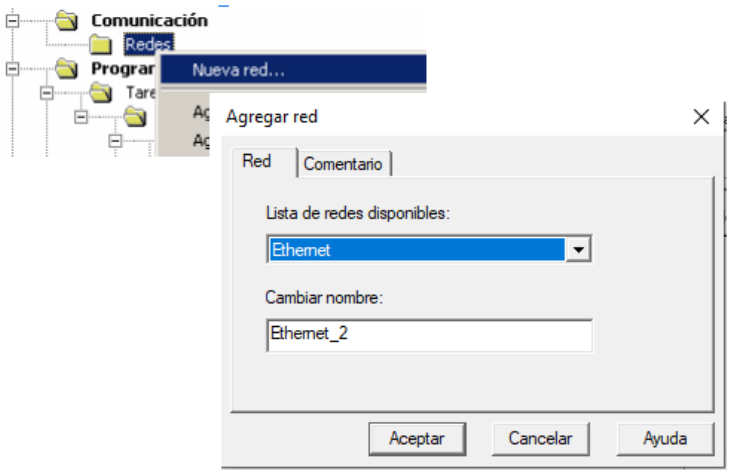


Figura 23. Creación de red/conexión

Configurar dicha conexión eligiendo la familia del modelo y escribiendo la dirección IP, máscara subred y puerta de enlace, además de que ofrece servicios de comunicación para que el PLC funcione como servidor de dirección IP, que intercambie datos mediante el servicio Global Data… .

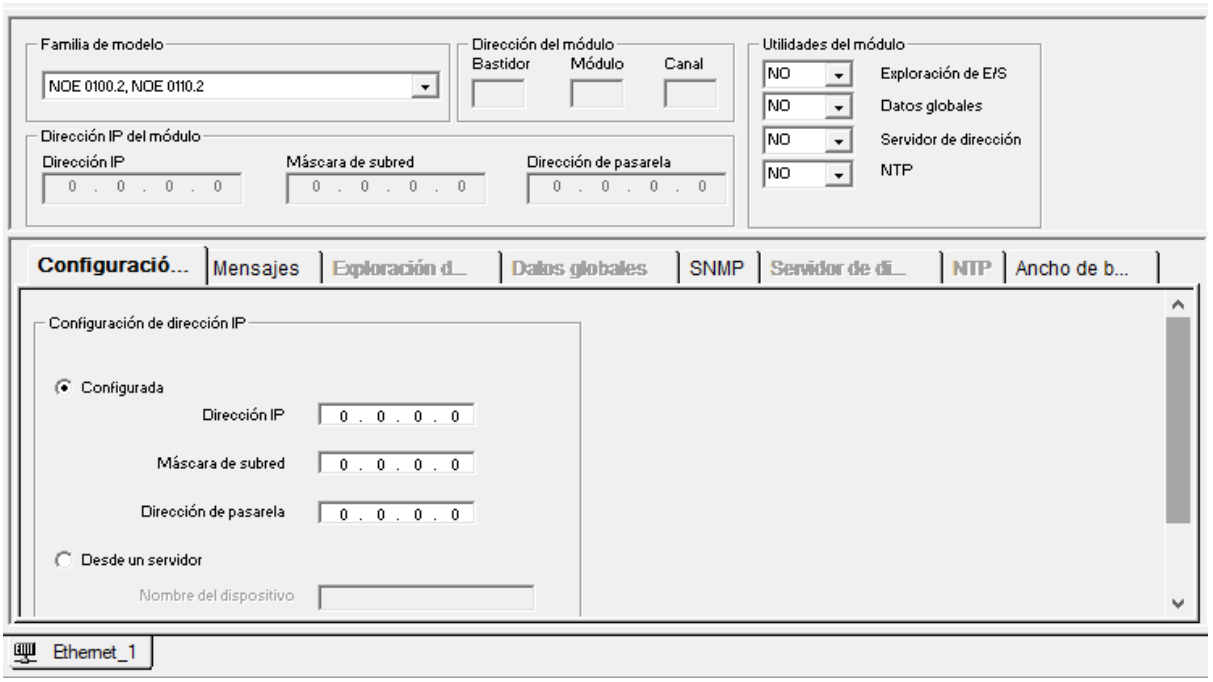


Figura 24. Configuración de la conexión

Finalmente se ha de asignar la conexión a un módulo Ethernet o puerto Ethernet integrado a la CPU.

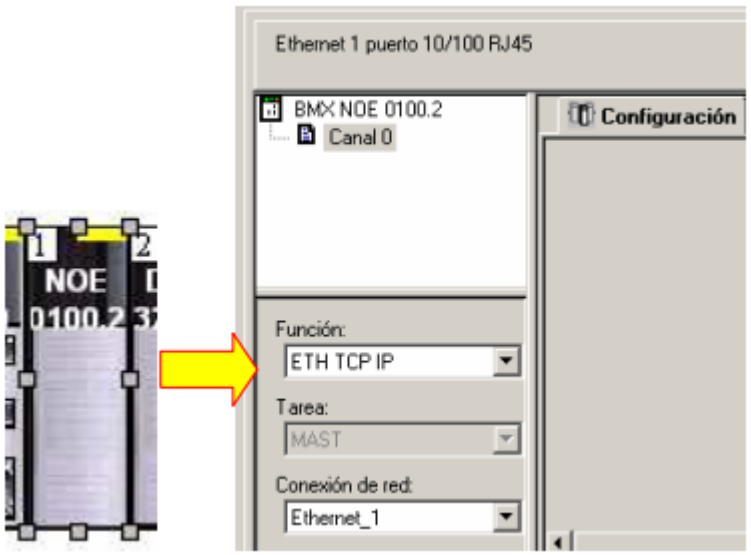


Figura 25. Asignación de la conexión

Secciones / Secciones SR

La lógica del programa se creará en las unidades de programa autónomas, es decir, en las secciones. Estas se ejecutan en el mismo orden en las que se presentan en el explorador de

proyectos y se pueden programar en los distintos lenguajes mencionados anteriormente, con la condición de que el lenguaje se admita en la tarea.

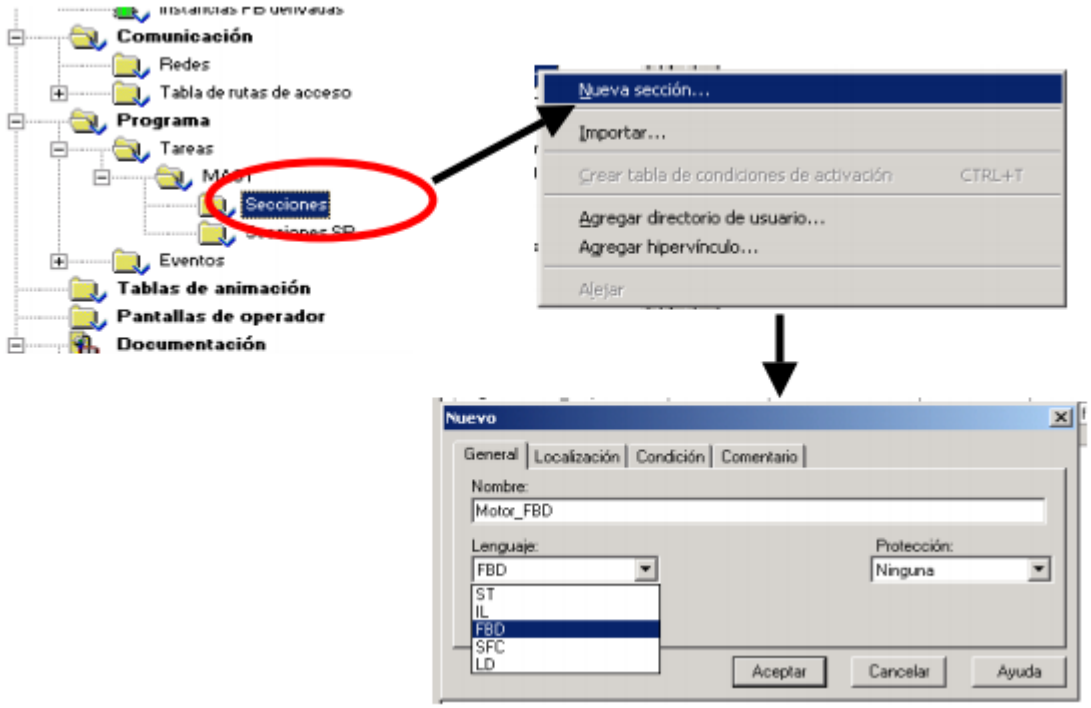


Figura 26. Creación de una sección

Las secciones se vinculan a una tarea, una misma sección no puede pertenecer simultáneamente a varias tareas.

En este proyecto se utilizará únicamente las secciones LD, como se observará en la imagen siguiente, estas secciones contienen una cuadrícula de fondo que divide la sección en líneas y columnas. El lenguaje de programación LD está orientado en celdas, sólo se puede colocar un objeto por celda. Dichos objetos u elementos aparecerán en el menú superior una vez seleccionado el lenguaje LD.

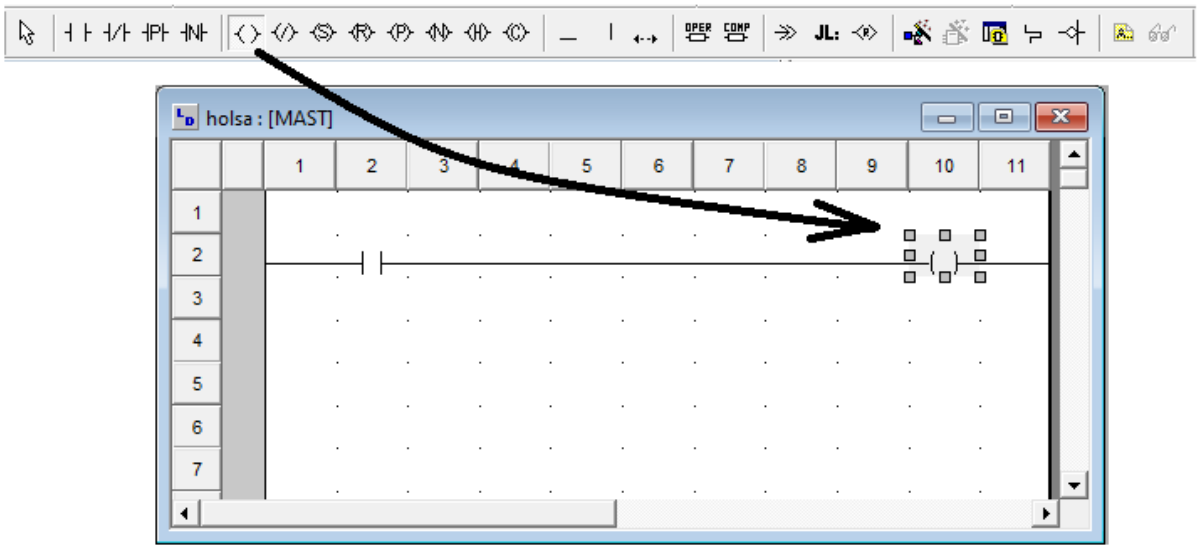


Figura 27. Ejemplo sección LD

Tablas de animación

Las tablas de animación permiten visualizar el valor de las variables una vez el PLC está conectado y en modo run, además te otorga la posibilidad de modificar el valor de una variable y forzar los valores, hasta que se cancele el forzado, de las señales con direccionamiento como %I y %Q.

En la siguiente imagen se indica cómo se crea una tabla de animación:

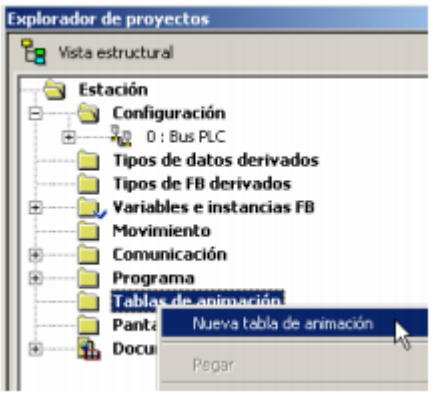


Figura 28. Creación de tabla de animación

Para la modificación del valor de cierta variable, se ha de pulsar el botón con el nombre “Modificación”, después hacer click en el campo valor de la variable e introducir el valor que queramos, es muy importante validar el valor pulsando la tecla Enter, sinó no se guardará el valor introducido.

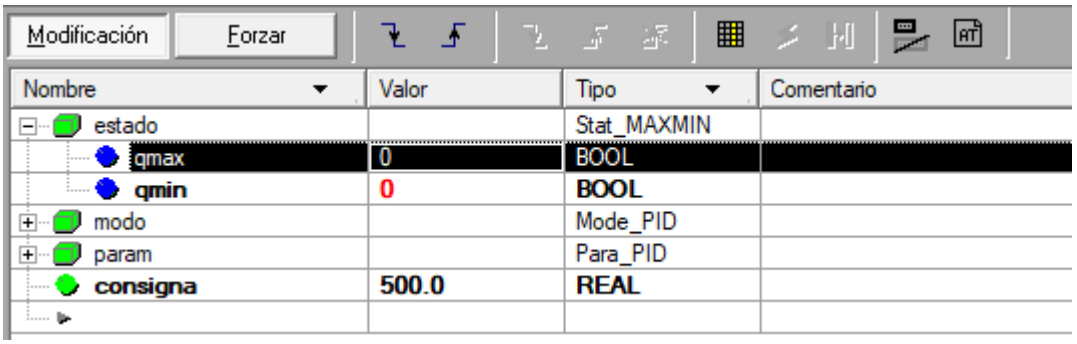


Figura 29. Modificación variables de la tabla de animación

Por otro lado, para forzar las señales se ha de seleccionar el botón "Forzar", hacer click sobre el campo valor y escribir el nuevo valor o cambiarlo mediante los iconos:

Figura 30. Iconos "Forzar"

Forzar a 0, Forzar a 1 y cancelar el forzado, respectivamente.

2.6 Software de Programación Vijeo Designer 6.2

El software de programación Vijeo és un software de configuración HMI que permite crear aplicaciones de diálogo del operario para el control de los sistemas de automatización.

Una vez creado y configurado un proyecto vijeo aparece la pantalla de edición del programa:

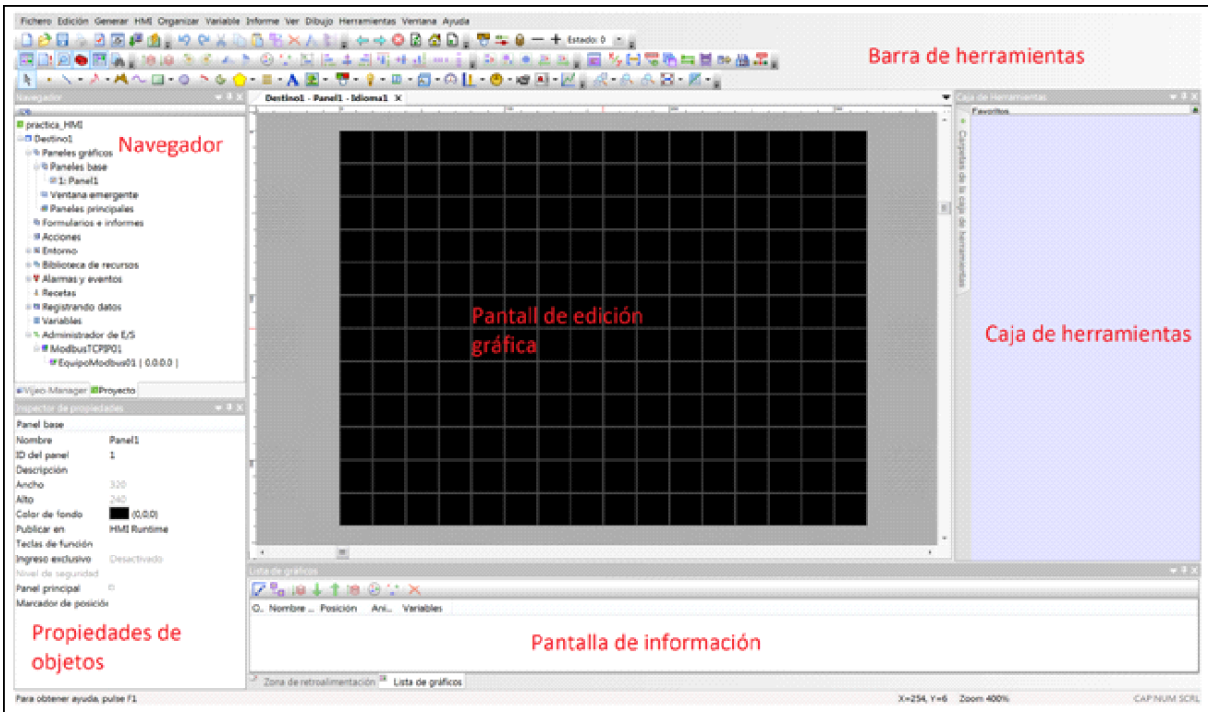


Figura 31. Barra de edición de un proyecto vijeo

Esta pantalla está formada por distintas partes que permiten al usuario la creación del proyecto.

- **Barra de herramientas**: Es el menú e iconos que sirven para acceder a las distintas funciones de Vijeo.

- **Navegador**: Representa el árbol del proyecto, es decir, permite acceder y modificar las distintas partes de un proyecto, además de configurar las variables, conexiones, entre otras.

- **Propiedades de objetos:** Permite modificar las propiedades de los elementos seleccionados, ya sea el color, tamaño, posicionamiento en la pantalla, … .

- **Pantalla de información**: Pantalla de resultados donde podremos ver la información relacionada con las distintas operaciones que realicemos, aparecen todo tipo de errores y advertencias del programa realizado o de otras operaciones.

- **Caja de herramientas**: En este apartado encontraremos una amplia librería formada por todas las familias de los objetos/iconos que se podrán utilizar para la realización del programa.

- **Pantalla de edición**: Se trata de un espacio cuadriculado en el cual podremos arrastrar los distintos elementos que utilizaremos en el programa, en la misma pantalla se pueden visualizar los elementos y modificarlos. Los elementos presentes en la pantalla de edición gráfica serán los que se muestran en la pantalla del HMI.

2.6.1 Variables compartidas

Para compartir las variables con con el proyecto Unity, se han de vincular las variables de usuario del proyecto Unity.

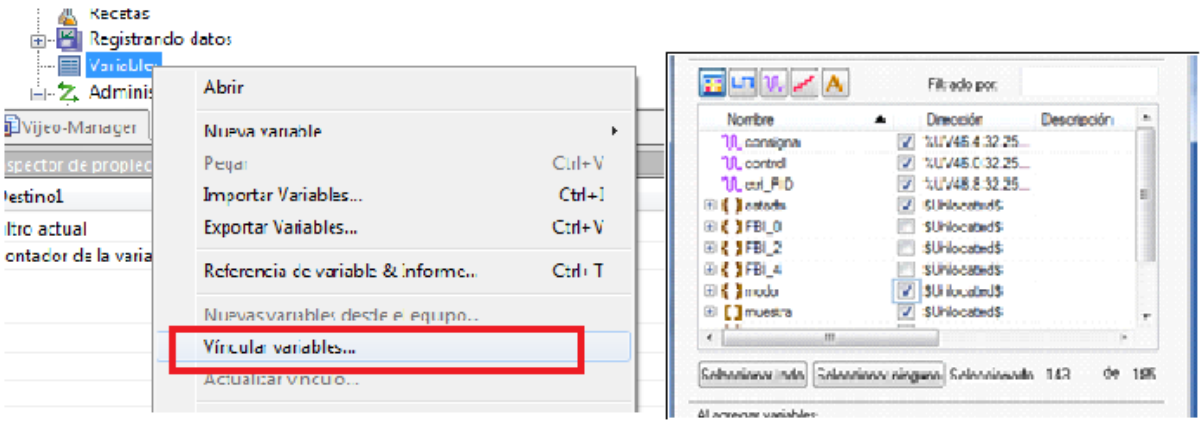


Figura 32. Vinculación de variables

Es de suma importancia haber exportado los archivos desde Unity (en el caso de mi proyecto se ha exportado en archivo .XVM) previamente, una vez realizado esto, desde la ventana del navegador de vijeo se podrá vincular las variables y seleccionar las necesarias para el programa.

2.7 Software de Programación Node-Red

Node-Red es una herramienta de programación que facilita la conexión con distintos dispositivos hardware, API y servicios en línea, creada por Nick O'Leary y Dave Conway-Jones del grupo de Servicios de Tecnologías Emergentes de IBM en 2013.Esta plataforma permite tener una experiencia de programación más visual proporcionando un editor basado en un navegador que facilita la interconexión de flujos a través de una amplia biblioteca de nodos logrando simplificar bastante la tarea de programar.

Node-Red se ha convertido en el estándar open-source, disponible en github, creado para la gestión y procesado de datos a tiempo real en entornos de IOT (Internet Of Things). Está basado en Node.js y la librería de JavaScript D3.js, ya que estos proporcionan la potencia suficiente para que Node-Red sea fiable.

Mediante la URL http://localhost:1880 se puede acceder al editor de nodos Node-RED y añadiendo a la misma URL " /ui " podremos acceder al dashboard de Node-RED.

2.7.1 Editor de flujo

Como hemos indicado anteriormente, Node-Red proporciona un editor basado en un navegador que permite la conexión de flujos mediante una paleta formada por una amplia gama de nodos.

Un flujo Node-RED funciona pasando mensajes entre nodos. Los mensajes son objetos JavaScript simples que pueden tener cualquier conjunto de propiedades. Por defecto, los mensajes tienen la propiedad ***payload*** , que puede ser del tipo object, string, Array etc.

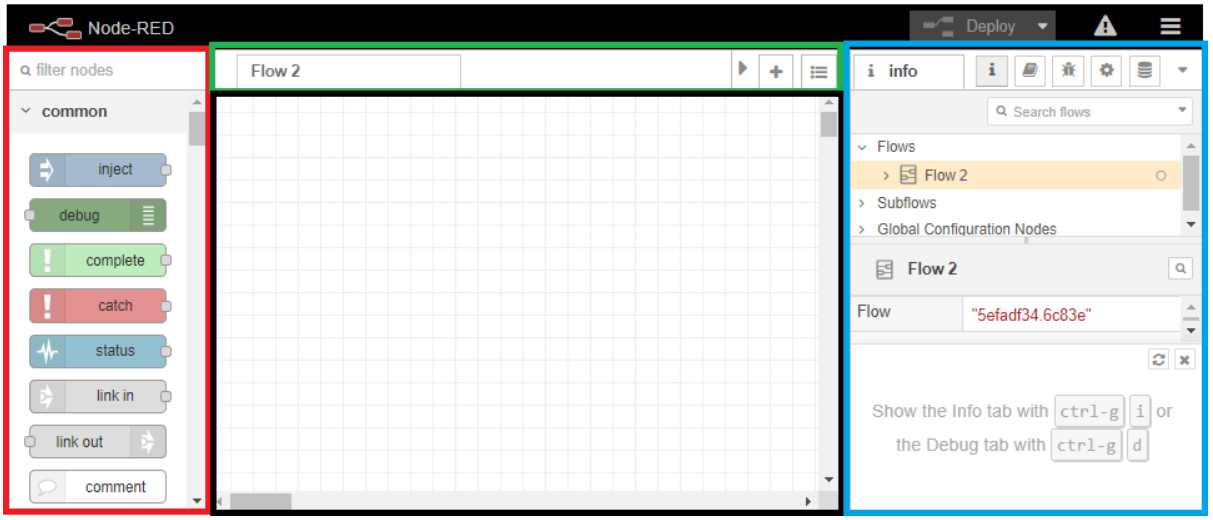


Figura 33. Navegador Node-Red

Los nodos se arrastran desde la parte izquierda de la interfaz gráfica, donde se pueden observar estructurados los distintos nodos, hacia la **parte central**, donde podremos conectarlos entre ellos para realizar una tarea concreta. Todos estos nodos se organizan en flujos o flows situados en la parte superior de la interfaz gráfica.

En la parte derecha de la interfaz gráfica se sitúa un apartado donde podremos visualizar desde la información y características de cada uno de los nodos, los nodos que forman parte de cada flujo, hasta los valores de salida mediante el uso de un nodo *debug*.

Por otro lado, en la parte superior derecha tenemos un recuadro ***Deploy*** que nos permitirá ejecutar los nodos seleccionados, además de otras opciones como ejecutar solamente los nodos o flows que han sido modificados, o reiniciar los flows que se estén ejecutando en ese instante.

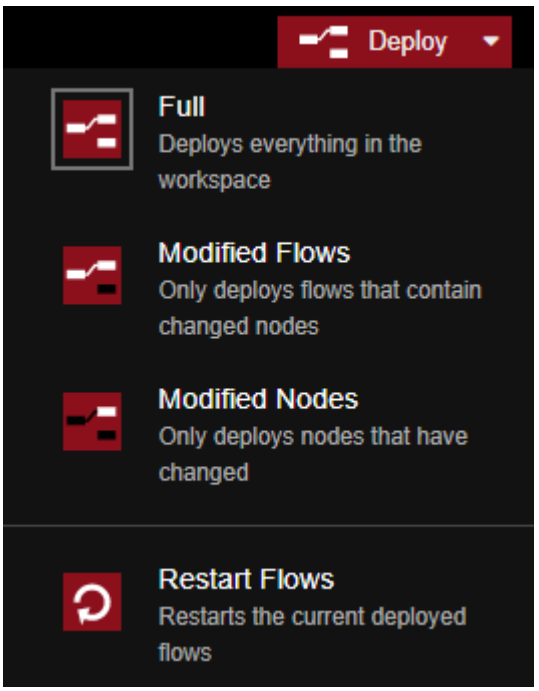


Figura 34. Opcion Deploy

Además junto al Deploy, existe una opción con una figura de tres barras horizontales, que mediante un click, despliega una lista de opciones que el usuario puede seleccionar, como por ejemplo, importar o exportar el flow, buscar un flow, configurar nodos, … etc .

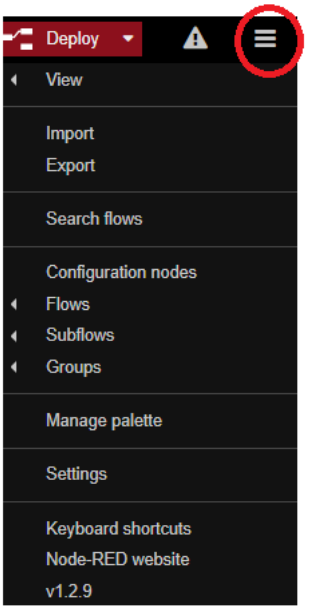


Figura 35. Lista de acciones Node-Red

En la opción ***Manage palette*** de esta lista se podrán descargar los distintos nodos que no vengan incluidos por defecto en Node-Red.

2.7.2 Nodos

En este apartado se explicarán las características de los distintos nodos que se utilizaran en este proyecto. Los nodos son:

Inject: Inyectar un mensaje dentro del flow ya sea manualmente o configurado previamente para que lo realice durante ciertos periodos.

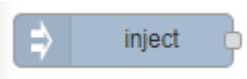


Figura 36. Incect Node

Function: Permite programa dicho nodo mediante Javascript.

Figura 37. Function node

Debug: Muestra las propiedades del mensaje del nodo seleccionado.

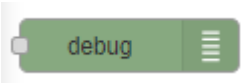


Figura 38. Debug node

Modbus Nodes: En el proyecto usaremos el ***Modbus Flex Getter*** y el ***Modbus Flex Write***. Estos nodos nos permiten leer o escribir la información de las bobinas, entradas o registros a la velocidad del mensaje entrante.

Figura 39. Modbus Flex Nodes

Buffer Parser: Este nodo permite convertir una matriz de números enteros (Int16) o un buffer en un objeto de valor según la especificación que se le asigne.

Figura 40. Buffer parser node

Chart: De la librería dashboard, mediante este nodo, se podrán visualizar los valores seleccionados en un formato de gráfica.

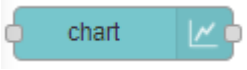


Figura 41. Chart node

3. Aspectos de diseño del sistema de control

El concepto de control es bastante amplio, abarcando desde un simple interruptor controlando el encendido de una bombilla hasta sistemas más complejos como el control del piloto automático de un avión. Este control se podría definir como la manipulación de las magnitudes de un sistema llamado *planta* a través de otro sistema llamado *sistema de control*.

El objetivo de este proyecto es la implementación de una aplicación de ***control de proceso industrial*** sobre una pantalla HMI, para ello, necesitaremos conocer el concepto de control industrial y estudiar los distintos sistemas de control que existen.

El sistema de control industrial es un término general que abarca distintos tipos de control e instrumentos necesarios para el control de procesos industriales.

3.1 Sistemas de control

Un sistema de control se puede interpretar como la combinación de elementos que actúan sobre una planta o proceso, tratando de fijar o variar alguno de sus parámetros, en el transcurso del tiempo.

Según hemos indicado anteriormente, el objetivo de un sistema de control es el de gobernar la respuesta de una ***planta*** a través de accionamientos, sin que el operador tenga que intervenir con los valores de la salida y solo tenga que manipular la consigna.

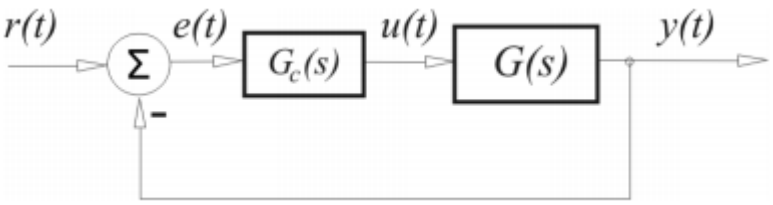


Figura 42. Ejemplo sistema de control

Para explicarlo de otra manera, podríamos decir que un controlador es un dispositivo que utiliza la señal de error *e(t)*, la procesa y genera una acción de control *u(t)*. En la figura x , podemos observar como el proceso *G(s)* (***planta***) es controlado por ***controlador*** *Gc(s)* dando lugar a la salida y(t).

La señal *r(t)* se denomina referencia e indica el estado que se desea conseguir en la salida del sistema *y(t)*.

La señal de error *e(t),* es la señal que le indica al controlador la diferencia que hay entre el estado que se quiere conseguir y el estado real del sistema.

El control automático está integrado por operaciones básicas:

- **Medida**: La medida de la variable a controlar se realiza mediante un sensor, que puede ser combinado con un transmisor.

- **Decisión**: Con la medición, el controlador debe decidir qué hacer el error para mantener la variable en el valor deseado.

- **Acción**: Generalmente esta acción se realiza al final del control, y es el resultado de la decisión del controlador.

Dependiendo de si la acción de control dependa o no de la salida del sistema a controlar, los sistemas de control pueden ser de lazo abierto o de lazo cerrado.

3.1.1 Sistema de control de lazo abierto

El sistema de control de lazo abierto es un sistema en el cual la salida no afecta a la acción de control, es decir, en un sistema de control abierto no se mide ni la salida ni se realimenta para compararla con la entrada para que el controlador pueda ajustar la condición de control. Estas características permiten que este tipo de sistemas sean sencillos y fáciles de implementar, eso sí, la precisión depende de la previa calibración del sistema.

Figura 43. Sistema de control de lazo abierto

Como podemos observar, el error (*e(s)*) coincide con el valor del punto de consigna(*sp(s)*).

3.1.2 Sistema de control de lazo cerrado

En el caso de los sistemas de control de lazo cerrado, son sistemas donde la acción de control está en función de la señal de salida. Se alimenta al controlador con la señal de error, el cual es la diferencia entre la señal de entrada y la señal de realimentación con la finalidad de reducir el error y conseguir que la señal de salida se ajuste lo máximo al valor deseado.

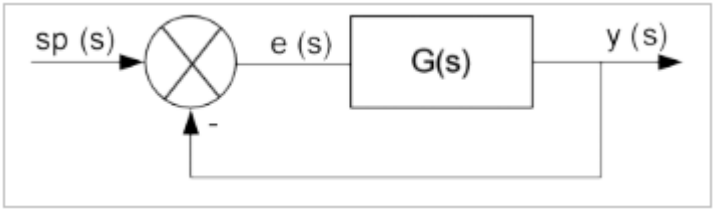


Figura 44. Sistema de control de lazo cerrado

Observando la estructura de la *Figura 44,* podemos deducir que *e(s)=sp(s)-y(s).*

3.1.3 Sistema de control de lazo cerrado con control PID

El regulador o controlador PID es un dispositivo que permite controlar un sistema en lazo cerrado para conseguir el estado de salida real. Este algoritmo PID está compuesto por tres parámetros que proporcionan una acción Proporcional, Integral y derivativa. El valor del controlador Proporcional depende del error actual, el Integral depende de errores anteriores y el Derivativo es una predicción de los errores futuros.Ajustando los valores de estas tres variables, el controlador permite realizar un control adaptado a las necesidades del proceso. La suma de estas tres acciones le dan nombre al controlador PID.

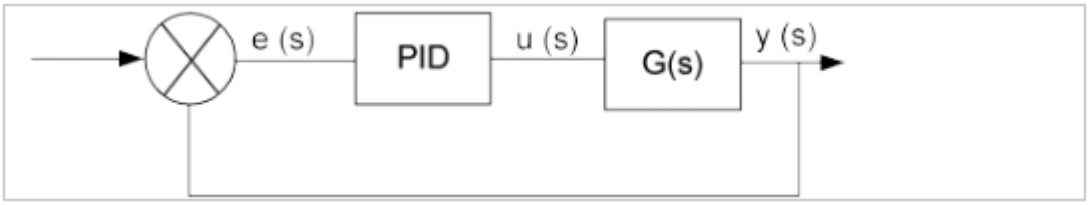


Figura 45. Sistema de control de lazo cerrado con control PID

Como podemos observar en la figura x, *e(s)* corresponde a la salida del control PID *u(s)*.

Acción de control Proporcional

Esta acción de control, es proporcional a la señal de error *e(t)*. Básicamente se multiplica la señal de error por una constante Kp. Dicha acción minimiza el error del sistema.

Aumentar la acción proporcional Kp permite aumentar la velocidad de respuesta del sistema y disminuir el error del sistema en régimen permanente, pero también aumenta la inestabilidad del sistema, por lo que se ha de buscar un equilibrio para conseguir el mínimo error sin que haya una gran inestabilidad.

En las siguiente imagen, *Figura 46*, se puede observar el comportamiento del sistema en respuesta al aumento de la constante Kp.

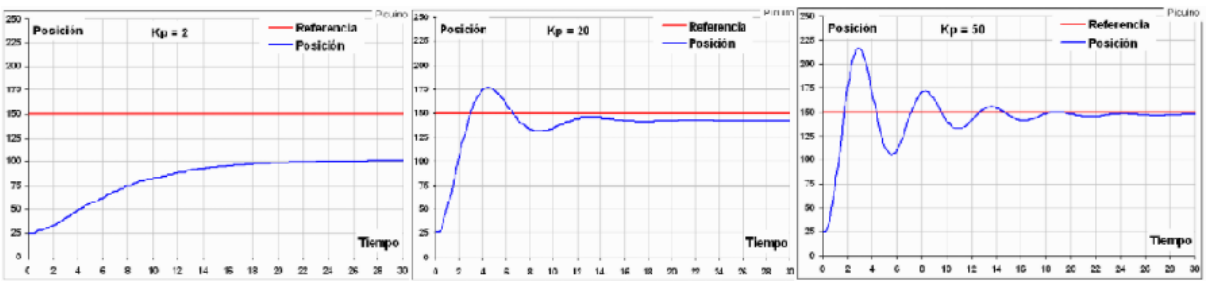


Figura 46. Ejemplo acción proporcional

Las Kp de las gráficas son 2, 20, 50, respectivamente. La línea roja representa a la señal de referencia y la azul a la del sistema.

En la *Figura 46*, podemos deducir que la mejor opción es la Kp=20, es decir la gráfica del medio, donde el error es mínimo y el sistema no es tan inestable.

Acción de control Integral

La acción de control Integral tiene la función de disminuir el error en estado estacionario provocado por los pequeños errores o perturbaciones que se van sumando a medida que pasa el tiempo y que el control proporcional no ha sido capaz de corregir. Como en el caso de la acción proporcional, hay una desventaja también al utilizar este control y es que al aumentar la Ki, a parte de disminuir el error del sistema en régimen permanente, aumenta la inestabilidad del sistem a causa de la suma de una cierta inercia al sistema

En las siguiente imagen, *Figura 47*, se puede observar el comportamiento del sistema en respuesta al aumento de la constante Ki.

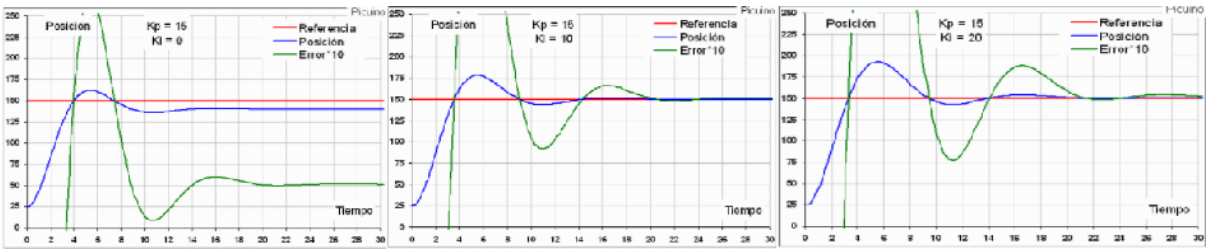


Figura 47. Acción de control integral

Las Ki de las gráficas son 0, 10 y 20 respectivamente, mientras que Kp se mantiene constante en 15, así podemos ver como el control integral funciona. La señal de color verde es el error, se puede ver como se reduce el error mientas aumenta la acción integral, pero también se observa como aumnenta la inestabilidad.

Acción de control Derivativa

La acción derivativa se utiliza cuando hay un cambio en el valor absoluto del error, si no es así, solo se usan las acciones proporcional e integral. Como su propio nombre indica este control es la derivada de la señal del error, es decir la “velocidad” del error.

Cuando un sistema se aproxima al punto de referencia, al moverse a una gran “velocidad” a causa de la acción proporcional, este se pasará de largo a causa de la inercia, provocando un sobreimpulso y a su vez oscilaciones. Con la acción derivativa se podrá evitar esta situación ya que se encarga de reconocer la velocidad a la que el sistema se acerca a la

referencia y así poder prevenir el sobreimpulso. De esta manera, cuando se aumenta la constante derivativa Kd, aumenta la estabilidad del sistema disminuyendo la velocidad del sistema. Por otro lado, este control no afecta al error en régimen permanente.

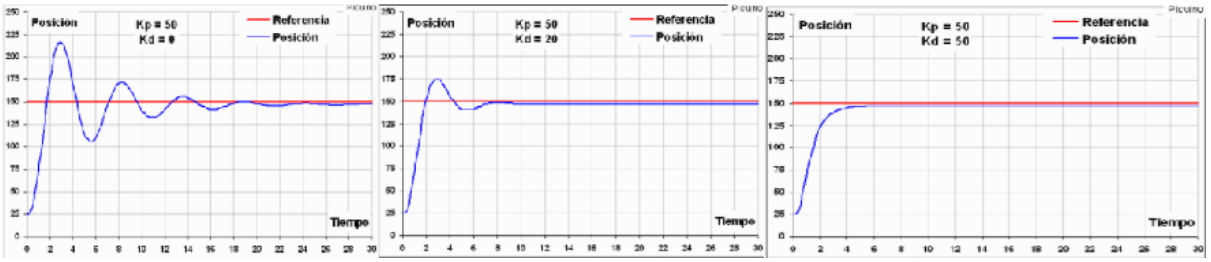


Figura 48. Acción de control derivativa

Las Ki de las gráficas son 0, 20 y 50 respectivamente, mientras que Kp se mantiene constante en 50.
Como podemos observar en la *Figura 48*, la acción derivativa está corrigiendo el sobreimpulso provocado por la acción Kp, a su vez, el sistema tarda un poco más en llegar la señal de referencia.

3.1.4 Ejemplos de Sistemas de control reales

Podemos encontrar todo tipo de sistemas de control en nuestra vida cotidiana, desde la más sencilla acción de encender una luz hasta los distintos dispositivos complejos.

En el caso de los sistemas de control en lazo abierto, podemos encontrar múltiples objetos con los que interactuamos que representan este tipo de control. Por ejemplo el control de potencia de un microondas, donde el controlador es el dispositivo que enciende y apaga el horno a cierta potencia durante un tiempo específico. En este tipo de sistemas, al ser abiertos, no hay feedback del estado del sistema y la corrección del error no se realiza mediante la comparación del valor de salida con el de referencia.

Por otro lado, los sistemas de control en lazo cerrado se pueden representar mediante el funcionamiento de distintos aparatos cotidianos como son el frigorífico inteligente, un servomotor, horno eléctrico, etc.. los cuales mediante un sensor que tienen incorporado, pueden medir el estado del sistema y realizar el control con más precisión.

Otro ejemplo de un sistema en lazo cerrado es la acción de tomarse una ducha, donde el objetivo de control es mantener la temperatura del agua en un valor deseado (valor de referencia).

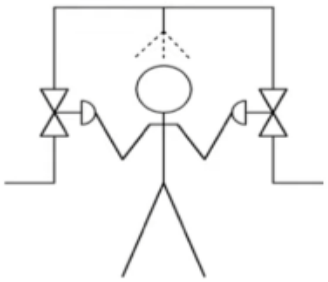


Figura 49. Ejemplo ducha

La planta corresponde a la acción general de tomarse la ducha, no hay que confundirlo con la persona que se ducha o el comportamiento de la regadera.

En este caso la variable de salida es la temperatura que queremos controlar.

Las variables de entrada son el flujo de agua caliente y de agua fría. Son las que podemos modificar para conseguir el control necesario.

El sensor de este sistema estaría representado por la persona que mediante la mano mide la temperatura del agua.

El cerebro de la persona representa al controlador, que compara la temperatura del agua con la temperatura de confort a la que quiere llegar. En función de esta temperatura, el cerebro manda a la otra mano para que ajuste la temperatura mediante la palanca. Esta secuencia de medir la temperatura y ajustarla se repite secuencialmente hasta ajustarla a la temperatura de referencia.

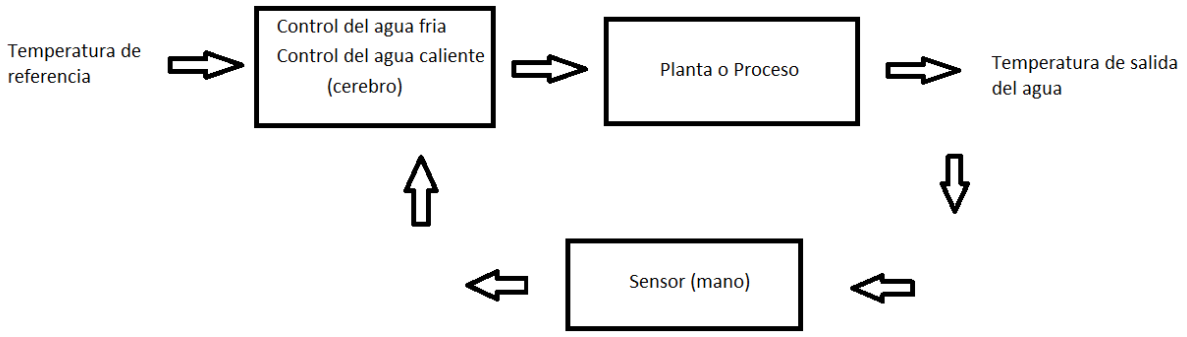


Figura 50. Ejemplo ducha - control de lazo cerrado

3.2 Método de sintonía de Ziegler y Nichols

Existe una gran cantidad de combinaciones de los distintos parámetros del controlador PID que permiten hacer una buena corrección del error de control, y más combinaciones aún que lo harán de manera incorrecta. Al proceso de selección de los parámetros de un controlador se le denomina sintonía.

La sintonía de los controladores PID se ha ido realizando a partir de las distintas características estimadas del proceso, mediante prueba y error o mediante ciertos criterios de sintonía comprobados empíricamente.

Las reglas de sintonía más utilizadas y eficaces que permiten ajustar los parámetros del controlador de forma empírica sin necesidad de conocer las ecuaciones de la planta o sistema controlado, fueron desarrollados por Ziegler y Nichols en 1942.

El método de sintonía de Ziegler y Nichols permite definir las ganancias proporcional, integral y derivativa siguiendo ciertas reglas empíricas en las cuales el criterio de sintonía es la razón de decaimiento de ¼, es decir, los valores determinados intentan conseguir que en un sistema realimentado a una respuesta escalón tenga un sobreimpulso del 25%, un valor con buenas características de rapidez y estabilidad para los distintos sistemas.

En el método de Ziegler y Nichols se proponen dos métodos de sintonía dependiendo del tipo de lazo, nosotros nos enfocaremos en el lazo cerrado que es donde el PID estará situado.

Tipo	Estructura	K_c	K_i	K_d
P	K_c	$0.5\ K_{cu}$		
PI	$K_c\left(1+\frac{K_i}{s}\right)$	$0.45\ K_{cu}$	$1.2\ /\ P_n$	
PID	$K_c\left(1+\frac{K_i}{s}+K_d\ s\right)$	$0.6\ K_{cu}$	$2\ /\ P_n$	$P_n\ /\ 8$

Tabla 4. Fórmulas de sintonía de Ziegler y Nichols (lazo cerrado)

En la Tabla 5 anterior podemos observar las distintas fórmulas para determinar los valores de los parámetros del controlador PID, dependiendo del tipo de control se esté realizando.

3.3 Ecuación de la planta

La planta corresponde al sistema físico que realiza una acción, el cual el controlador se encarga de regular para que realice dicha acción de manera correcta.

La ecuación de la planta que se usará en este proyecto corresponde a una que se ha utilizado en la práctica 9 de la asignatura de automatización que se cursa en la UPC de Terrassa, la información que se adjuntará a continuación corresponde en gran parte a la del documento de dicha práctica, dónde se explica de manera sencilla y concisa el procedimiento que se sigue hasta llegar a la ecuación de la planta.

3.3.1 Discretización de un sistema de primer orden

Los sistemas de primer orden con retardo son sistemas frecuentemente usados en los procesos industriales. En este proyecto se simularán estos procesos en dispositivos discretos implementando un algoritmo discreto que se aproxime al sistema continuo en estudio.

Para el desarrollo teórico de esta ecuación empezaremos con un sistema de primer orden, el qual se representa mediante su función de transferencia G(s) de la siguiente manera:

$$G(s)=\frac{Y(s)}{U(s)}=\frac{K}{\tau s+1}$$

Figura 51. Ecuación de primer orden

Dónde:

- ***y(t)*** es la salida del sistema y ***Y(s)*** su transformada de Laplace.
- ***u(y)*** es la entrada del sistema y ***U(s)*** su transformada de Laplace.
- ***K*** es la ganancia del sistema.
- **τ** es la constante de tiempo del sistema.

El modelo temporal de la función de transferencia se puede conseguir a través de la ecuación anterior.

$$\frac{dy(t)}{dt}+\frac{1}{\tau}y(t)=\frac{K}{\tau}u(t)$$

Figura 52. Modelo temporal

Con la fórmula anterior, despejando, podemos llegar a conocer la derivada de la salida a partir del valor actual de la salida y de la entrada, quedando la ecuación de la siguiente manera:

$$\frac{dy(t)}{dt}=\frac{1}{\tau}(Ku(t)-y(t))$$

Figura 53. Derivada de la salida I

Necesitaremos realizar una aproximación discreta por el método de Euler de la derivada de la salida a partir de la ecuación anterior.

$$\frac{dy}{dt}(n)=\dot{y}(n)=\frac{y(n+1)-y(n)}{dt}$$

Figura 54. Derivada de la salida II

Donde:

- $\dot{y}(n)$ corresponde a la derivada de la salida en el muestreo n.
- ***y(n)*** es la salida del sistema en la muestra n.
- ***y(n+1)*** es la salida del sistema en la siguiente muestra.
- ***dt*** corresponde al tiempo de muestreo ***T*** del sistema.

De esta manera podemos deducir que la salida y(n+1), será:

$$y(n+1)=\dot{y}(n)T+y(n)$$

Figura 55. Salida de la siguiente muestra

Por otro lado, sustituyendo en la fórmula de la derivada de la salida la variable $\dot{y}(n)$ tendremos la siguiente ecuación:

$$\dot{y}(n)=\frac{1}{\tau}(Ku(n)-y(n))$$

Figura 56. derivada de la salida mediante muestras

Finalmente, con las dos ecuaciones anteriores se obtiene la discretización del sistema continuo de primer orden sustituyendo el valor de la derivada de la salida del sistema de la segunda ecuación en la primera. Esta ecuación será la que aplicaremos en nuestro proyecto más adelante y queda de la siguiente manera:

$$\begin{cases} y(n+1)=\dot{y}(n)T+y(n) \\ \dot{y}(n)=\frac{1}{\tau}(Ku(n)-y(n)) \end{cases} \Rightarrow \boxed{y(n+1)=\frac{1}{\tau}(Ku(n)-y(n))T+y(n)}$$

Figura 57. Ecuación de primer orden discretizada

Teniendo en cuenta que el periodo de muestreo del T es de 1 segundo, obtenemos:

$$y(n+1)=0.244*e+0.951\,y(n)$$

Figura 58. Ecuación de la planta

Donde **e** es la señal de control del proceso. Esta señal variará dependiendo de la estructura de control, en el apartado 3.1 Sistemas de control se vé claramente el valor de la señal en cada una de las estructuras.

4. Diseño e implementación de la aplicación de control

Una vez asimilada y entendida la teoría que engloba las distintas etapas de este proyecto, llega el momento de diseñar e implementar la aplicación de control con los dispositivos mencionados en los apartados anteriores.

A continuación, se representará el diseño del flujo de trabajo de la aplicación de control donde implementaremos los distintos programas.

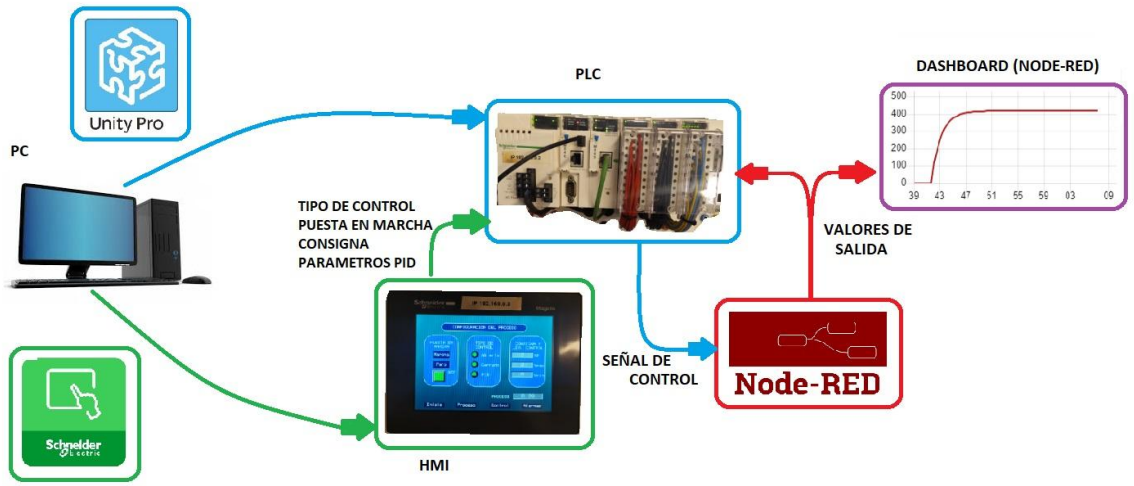


Figura 59. Flujo de información del proyecto

En la figurax podemos ver el flujo de la información del proyecto y los distintos dispositivos/ programas que influyen. El proceso de control empezará cuando carguemos los programas en los dos dispositivos, el programa Unity al PLC y el VIJEO a la pantalla HMI.

Una vez realizado esto, el operario, mediante la pantalla HMI, seleccionará el tipo de control, el valor de referencia/consigna, la puesta en marcha del programa del PLC, los valores de los parámetros PID, etc… . El PLC recibe esta información del HMI y procederá a enviar los valores del control a la planta.

La planta está situada en Node-Red. Este software recibirá los valores de control del PLC, realizará ciertos cálculos mediante la ecuación del sistema y enviará el valor de la salida del sistema al PLC (realimentación), para que se calcule el error mediante la comparación del valor de referencia y la salida del sistema de Node-Red.

Node-Red cuenta con la aplicación Dashboard, con la que podremos visualizar el comportamiento de la señal de salida y la podremos comparar con el valor de referencia.

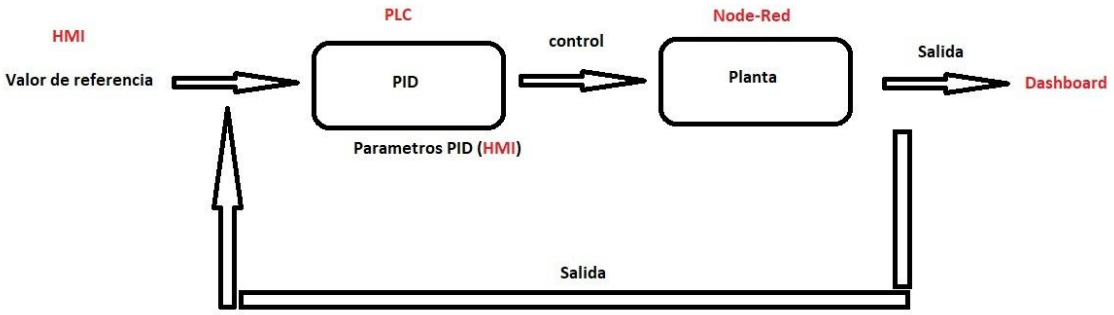


Figura 60. Ejemplo aplicado de un sistema en lazo cerrado con PID

En la imagen anterior, se representa de manera sencilla la aplicación de los dispositivos a un sistema de control de lazo cerrado con PID.

4.1 Comunicaciones industriales

En este apartado se explicará cómo se conectan los distintos dispositivos entre sí y qué tipos de protocolos de comunicación se necesitan para cada uno de estos dispositivos. En el siguiente diagrama de flujo se representan las comunicaciones industriales utilizadas en este proyecto.

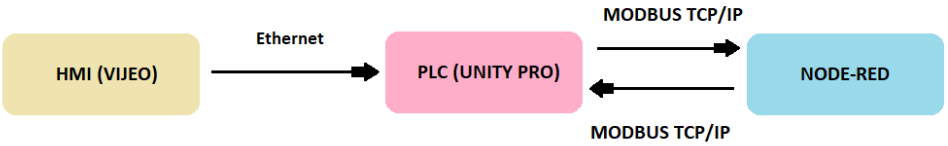


Figura 61. Comunicaciones entre dispositivos

Como podemos observar en el diagrama de flujo, la información se transmite inicialmente por la pantalla HMI al PLC mediante el protocolo de comunicación Ethernet. El PLC necesita los valores de entrada del HMI para compilar las instrucciones necesarias y poder enviar las señales de salida. En este caso el PLC envía los valores de control mediante MODBUS TCP/IP a Node-Red, donde se encuentra situada la planta. De la misma manera, los valores de salida de Node-Red se envían mediante el mismo protocolo (MODBUS TCP/IP) al PLC.

La conexión entre HMI y PLC se realiza mediante un módulo que se configura previamente en el programa Unity Pro XL, dicho módulo permite la conexión a través de un cable coaxial, un par trenzado o fibra óptica. En cuanto a la comunicación PLC y Node-Red se realiza a través de una red IP y con TCP como transporte, mediante la capa física Ethernet.

4.2 Instrucciones del programa Unity Pro XL

La base de este proyecto se construye sobre el programa Unity Pro XL, aquí estará situado el programa principal que se encargará de poner en marcha la aplicación de control y de configurar el tipo de lazo de control.

Lo primero de todo, para empezar el proyecto se han de crear las variables necesarias para realizar el control.

Nombre	Tipo	Dirección	Valor	Comentario
consigna	REAL			
control	REAL	%MW180		
ctrl_PID	REAL			
gain	REAL			
salidanodo	REAL	%MW160		

Figura 62. Variables elementales

Como se puede observar, sólo la variable ***"control"*** y ***"salidanodo"*** tiene una dirección, esto es necesario para que el software Node-Red pueda acceder al valor de estas variables mediante el protocolo Modbus TCP/IP y se pueda realizar el intercambio de información con la planta.
Una vez realizado esto, crearemos una sección, a la que llamaremos ***"main_prog"***, donde se encontrará situado el programa principal del proceso de control.

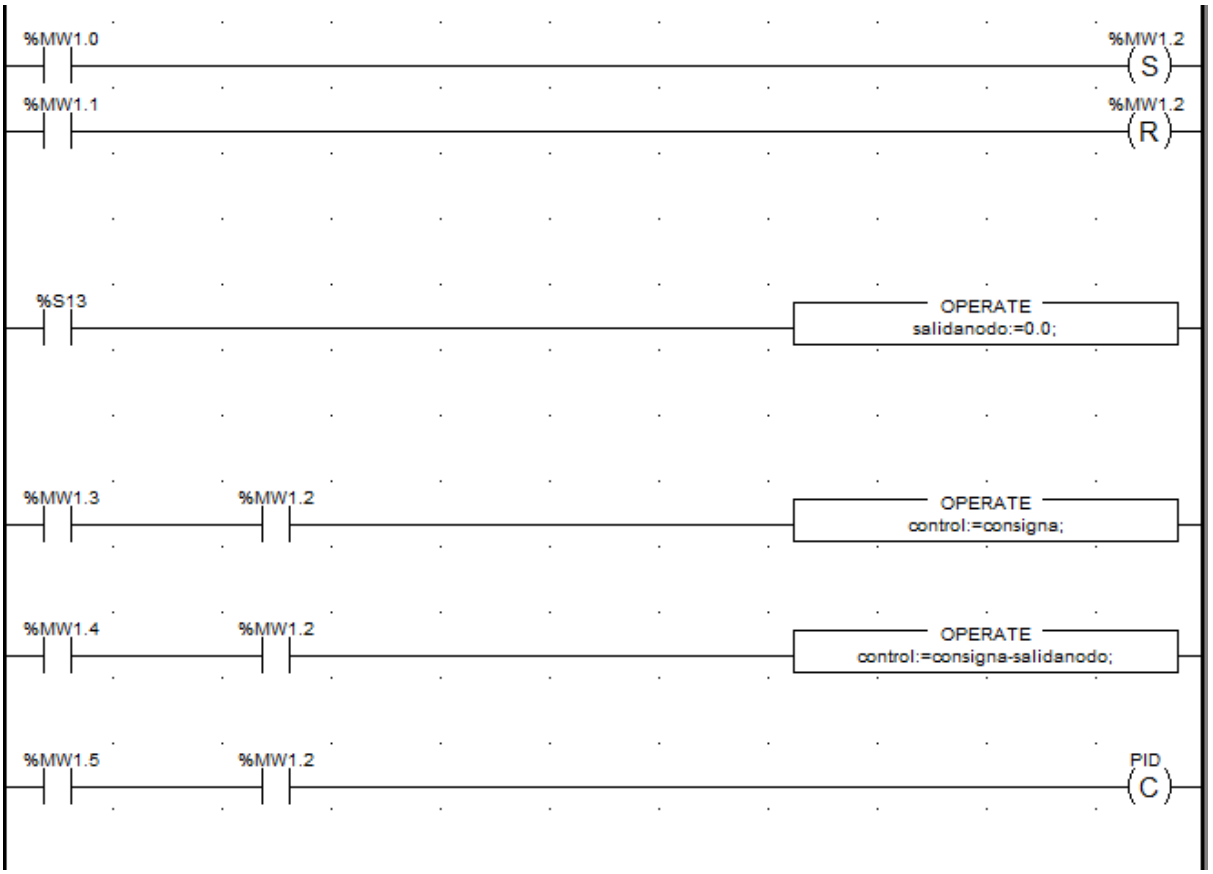


Figura 63. Programa principal

Como podemos observar en el programa principal, la sección comienza con la puesta en marcha o paro del proceso.

Figura 64. Puesta en marcha del programa

Mediante el contacto %MW1.0, podremos poner en set la bobina %MW1.2. Por otro lado, mediante el contacto %MW1.1, podremos poner en reset la misma bobina.

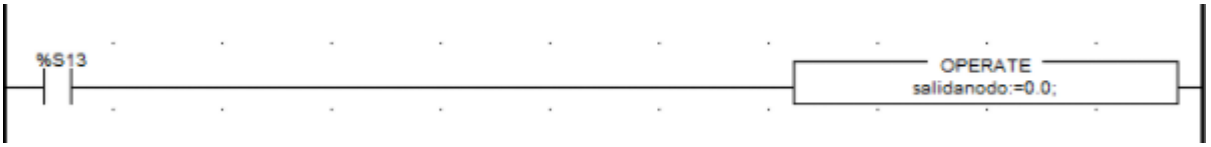


Figura 65. Inicializar "salidanodo"

Mediante el contacto %S13, que se pondrá a 1 durante el primer ciclo de RUN, pondremos a 0 la salida del proceso durante el primer ciclo de SCAN.

Figura 66. Selección del tipo de control

Como se puede observar en la *Figura 65*, dependiendo del contacto seleccionado y si el proceso está en marcha o no (estado del contacto %MW1.2), podremos seleccionar el tipo de control que se realizará. De la siguiente manera:

- Mediante el contacto %MW1.3 seleccionaremos el control en lazo abierto.
- Mediante el contacto %MW1.4 seleccionaremos el control en lazo cerrado.
- Mediante el contacto %MW1.5 seleccionaremos el control en lazo cerrado con PID.

En el caso de la selección del control en lazo cerrado con controlador PID, como se podrá observar en la *Figura 65*, la bobina no es una bobina normal ni de set (S) ni de reset (R), sino que tiene una C. La C de la bobina del PID significa que es una bobina de llamada. Esta bobina se encargará de llamar a al subprograma llamado PID.

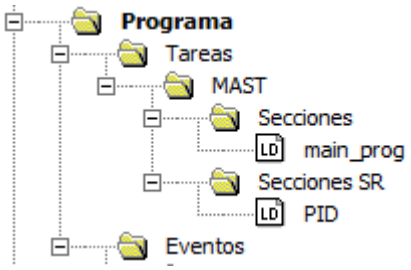


Figura 67. Secciones y secciones SR

Como podemos observar, para programar un subprograma se ha de realizar mediante la creación de una sección SR, de esta manera las secciones podrán llamar a estos subprogramas a través de la bobina (C) .

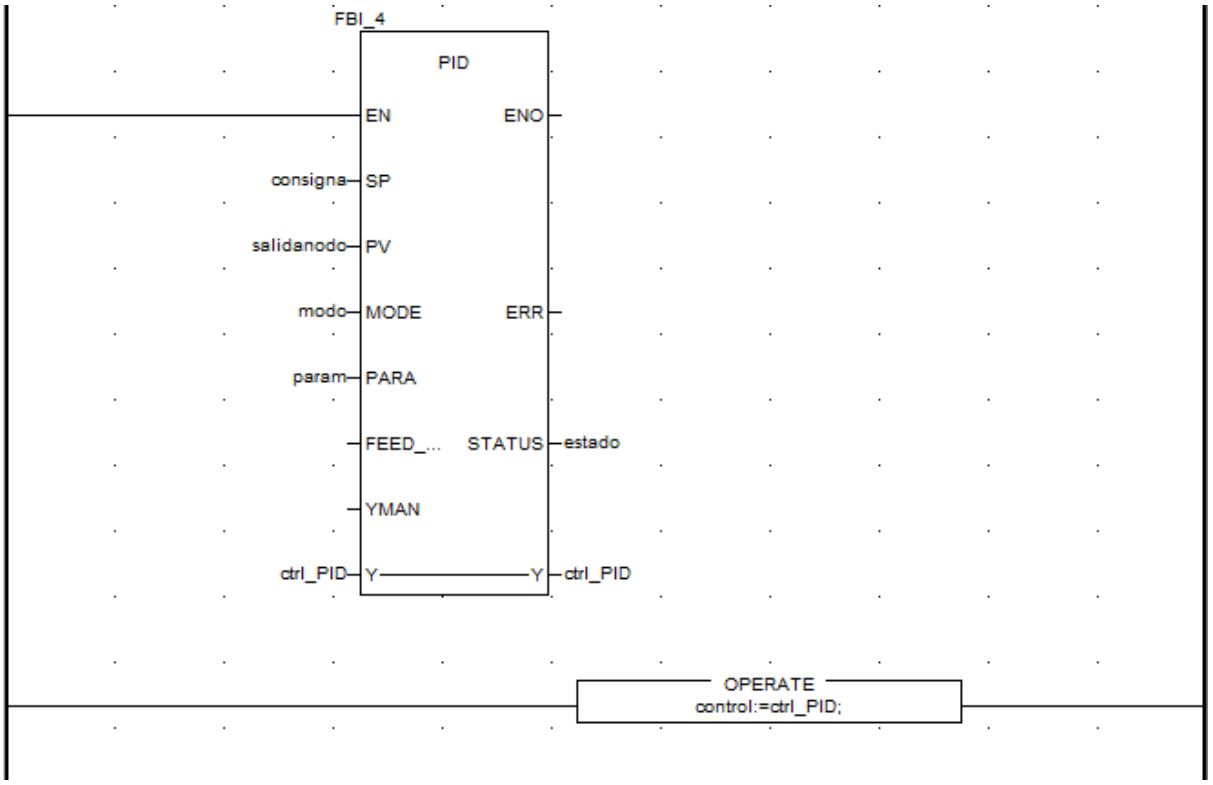


Figura 68. Sección SR para el control PID

En la sección SR llamada PID, se encontrará el controlador PID. El bloque FBI_4 que se observa en la Figura 67 permite realizar el control PID (ctrl_PID) al que en la siguiente línea de LD igualamos a la variable control (la que usaremos para enviar la información a la planta).

Al crear el bloque PID, se crearán ciertas variables derivadas con las que podremos configurar el controlador.

Nombre	Tipo	Dirección	Valor	Comentario
estado	Stat_MAXMIN			
qmax	BOOL			
qmin	BOOL			
modo	Mode_PID			
man	BOOL			
halt	BOOL			
en_p	BOOL			
en_i	BOOL			
en_d	BOOL			
d_on_pv	BOOL			
param	Para_PID			
gain	REAL			
ti	TIME			
td	TIME			
td_lag	TIME			
ymax	REAL			
ymin	REAL			

Figura 69. Parámetros PID

Se puede configurar tanto el tipo de control (P, PI, PD, PID) que se quiere realizar, como el valor de los parámetros PID, entre otros.

4.3 Aplicación HMI

En el caso de la aplicación HMI, se desarrollará una aplicación multipantalla que nos permitirá la evaluación del proceso en estudio.

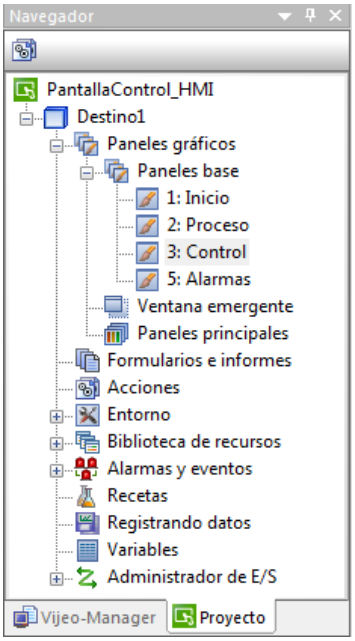


Figura 70. Navegador vijeo

Dicha aplicación está formada por:

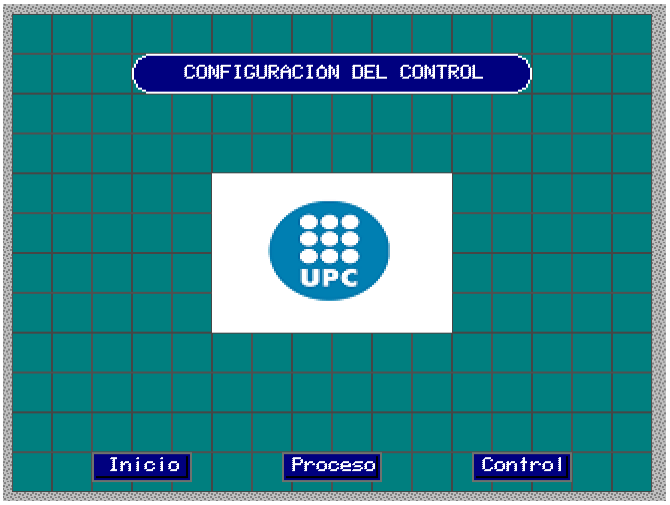


Figura 71. Pantalla de inicio vijeo

Una pantalla de inicio en la que mediante los botones de la parte inferior se puede acceder a las otras pantallas.

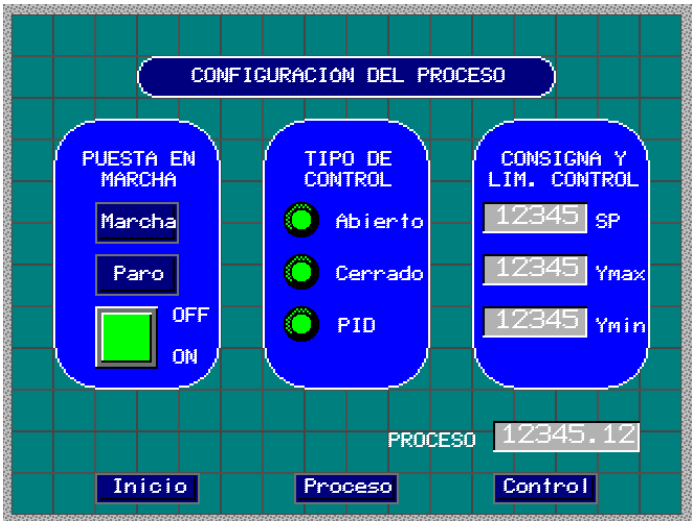


Figura 72. Pantalla de proceso vijeo

Una pantalla en la que podremos configurar el proceso de control, es decir, podremos poner en marcha, configurar el tipo de lazo de control e introducir la consigna y el límite de control.

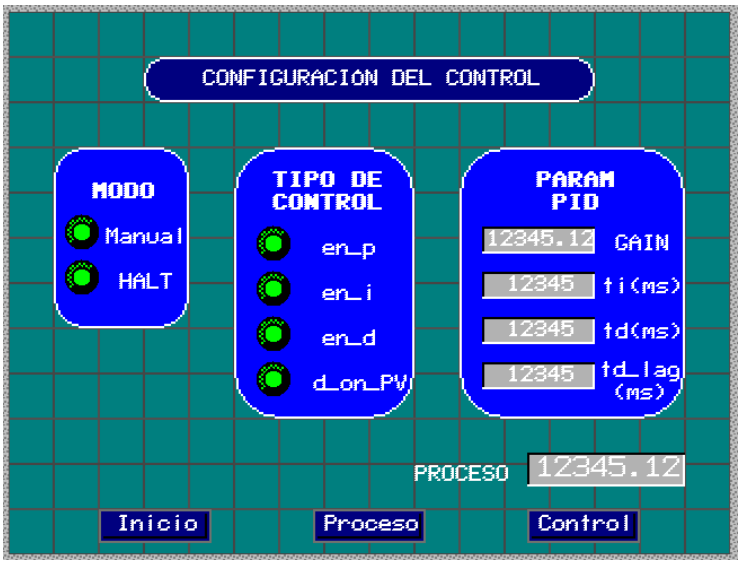


Figura 73. Pantalla de control vijeo

Finalmente, una pantalla donde podremos configurar e introducir los distintos parámetros del controlador PID.

Desde cada pantalla, presionando los botones de la parte inferior se puede acceder a las otras pantallas.

4.4 Diagrama de flujo Node-Red

El diagrama de flujo de Node-Red consta de dos partes. Por un lado tenemos un flujo liderado por un Inject que se encargará de marcar la dinámica de las comunicaciones, es decir, dependiendo del intervalo de inyección de dicho nodo, la velocidad de transmisión de información entre Node-Red y el PLC será más rápida o más lenta. A continuación se muestran los nodos que forman el flujo de comunicaciones:

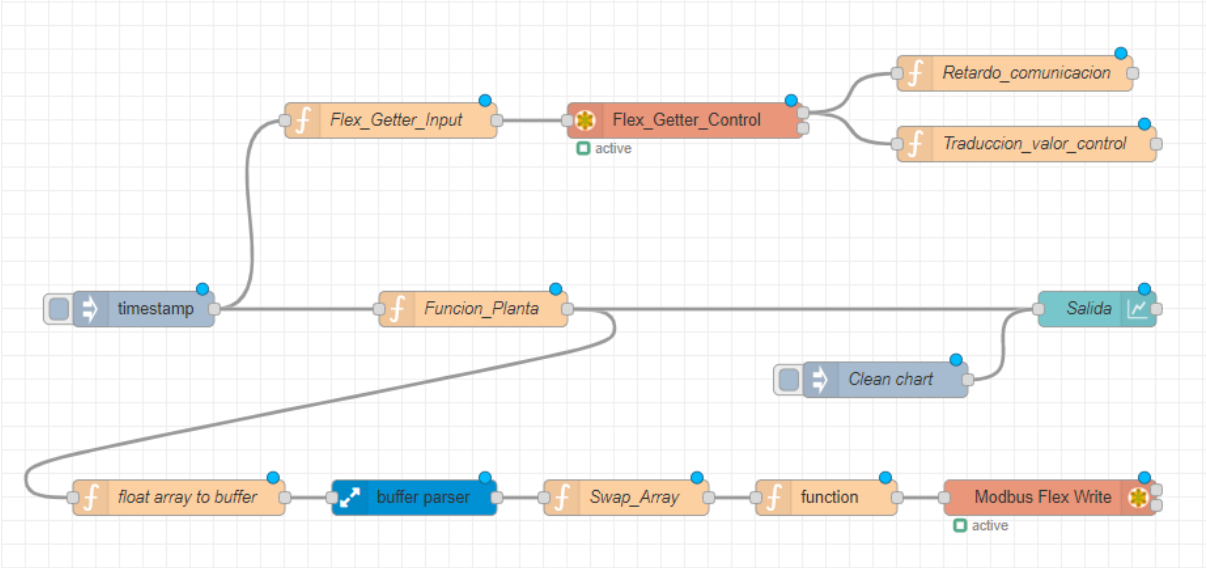


Figura 74. Dinámica de las comunicaciones

Como se ha indicado anteriormente, este flujo permitirá enviar y recibir información del PLC. Esto es posible gracias a los nodos modbus, los cuales se han de descargar porque no forman parte de los nodos que aporta Node-Red por defecto.

Por otro lado, mediante otro nodo Inject podremos controlar la dinámica de la planta de la misma manera que con la de las comunicaciones, fijando intervalos de inyección del flujo.

Los nodos que permitirán crear la planta y controlar su dinámica son los siguientes:

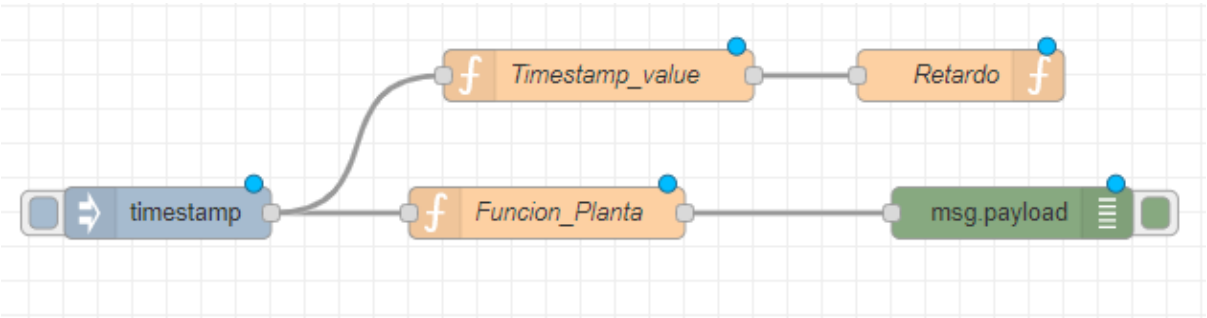


Figura 75. Dinámica de la planta

Como se podrá observar, la planta se encuentra en un nodo de función.

Es muy importante tener en cuenta que todos estos nodos, tanto los de la comunicación como los de la planta, se encuentran en un mismo flow, es decir, en un mismo espacio de trabajo.

La planta puede formar parte del flujo de la dinámica de comunicaciones perfectamente (no hace falta utilizar dos Injects), pero se ha decidido hacerlo de esta manera con la finalidad

de realizar un estudio más profundo y observar el comportamiento de estas dos dinámicas por separado.

4.4.1 Flujo de la dinámica de comunicaciones

Las comunicaciones con el PLC se realizan mediante dos nodos modbus. El nodo ***"modbus flex getter"*** permite leer la información de las bobinas, entradas o registros a la velocidad del mensaje entrante.

Por otro lado, mediante el nodo ***"modbus flex write"*** se podrá enviar la información de las bobinas, entradas o registros a la velocidad del mensaje entrante.

Para conectarte con el PLC, los nodos modbus se han de conectar a Modbus TCP, esto se realiza desde los propios nodos, en el apartado de server, se ha de configurar de la siguiente manera:

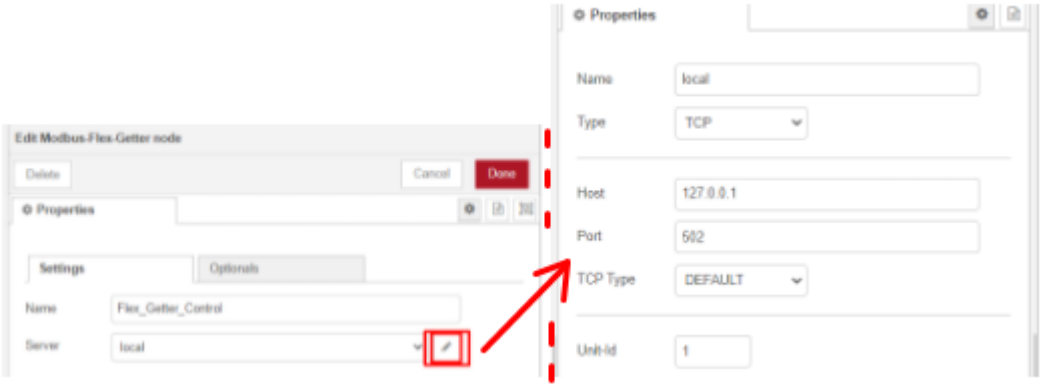


Figura 76. Configuracion del server

El "Host" puede ser el local (127.0.0.1) para simulación o en el caso de estar en el laboratorio, se ha de introducir la IP del PLC.

4.4.2 Configuración de los nodos de lectura

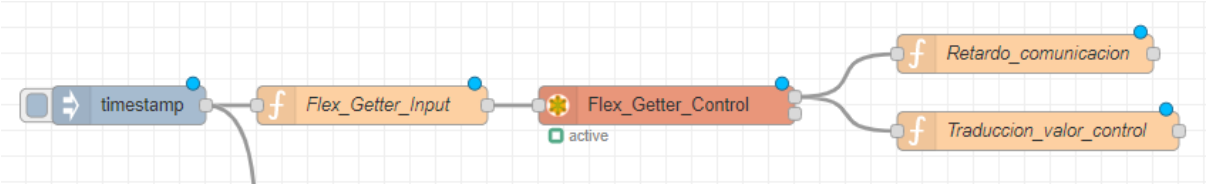


Figura 77. Nodos de lectura

A parte de la configuración del servidor del nodo ***"modbus flex getter"***, este necesita de ciertos parámetros de entrada para conectarse a modbus y leer la información necesaria. Dichos parámetros serán configurados en un nodo de función de la siguiente manera:

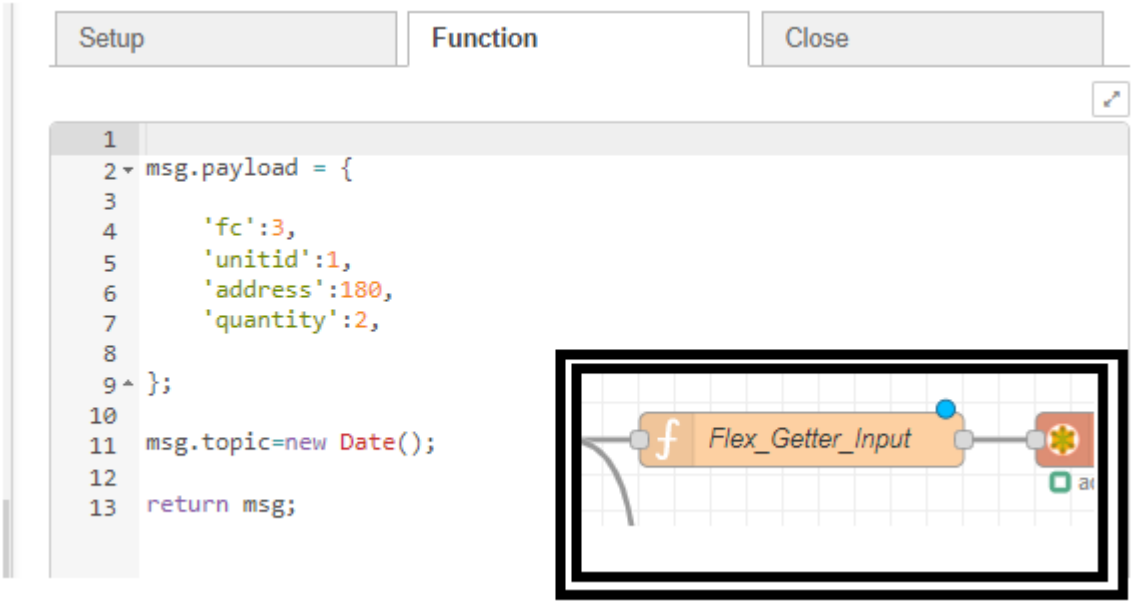


Figura 78. Configuración de los inputs del flex getter

Estos son los apartados que se han de configurar:

- Necesitamos hacer la lectura de "holding registers", por lo que ***'fc'*** ha de ser igual a 3.
- ***"unitid"*** (0..255 tcp | 0..247 serial).
- ***"address"*** (dirección de lectura) es igual a 180. Esta dirección es la que se le ha asignado a la variable ***"control"*** en el PLC.
- ***"quantity"*** la igualamos a 2. Esto se debe a que modbus intercambia paquetes de datos de 16 bits y la lectura que estamos realizando es de una variable REAL, la cual ocupa un espacio en memoria de 32 bits.

El nodo ***"modbus flex getter"*** devolverá el valor de la variable del PLC en el siguiente formato, por ejemplo [17105,54193], que es la traducción en decimal del conjunto de bits. Pero a nosotros nos interesa el valor real, para ello, se realizará la traducción de estos valores mediante un nodo de función.

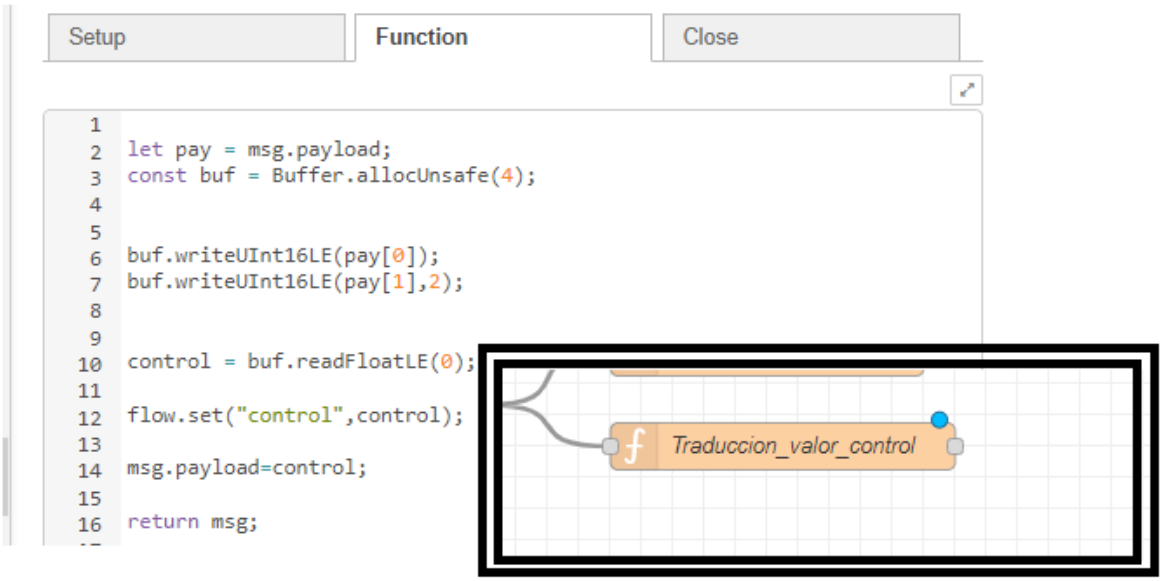


Figura 79. Traduccion valor de control

Explicación del código:

En la línea de código número **2** igualamos el ***msg.payload*** a la variable ***pay***. ***Msg.payload*** es igual al valor de salida del ***"modbus flex getter"***, de esta manera, si ***msg.payload***, como en el ejemplo anterior, es [17105,54193], entonces ***pay***=[17105,54193].

En la línea de código número **3** creamos una constante ***buf***, la cual será igual a "***Buffer.allocUnsafe(4)"***. Esta instrucción permite crear un nuevo ***buffer object***, en este caso de 4 Bytes, los necesarios para poder guardar el valor del ***pay***. Hay que tener en cuenta que 4 Bytes es equivalente a los 32 bits.

Un **buffer** es un espacio de memoria que permite almacenar temporalmente datos de entrada o salida.

En las siguientes dos líneas de código, la 6 y 7, se utilizará la instrucción "***Buf.writeUInt16LE(pay[0])"*** y "***Buf.writeUInt16LE(pay[1],2)"***. Esta instrucción permite escribir los bits especificados en ***Little Endian*** en el ***buffer***.

El formato ***Little Endian*** es aquel que ordena los Bytes del menos significativo al más significativo.

Para expresar ciertos datos se ha de tener en cuenta los Bytes más significativos (MSB) y los menos significativos (LSB).

sintaxis:

Buffer.writeUInt16LE(value, offset)

- **Value:** Es el valor integral que se escribirá en el ***Buffer***.
- **Offset:** Se trata de un valor integral y representa el número de Bytes que se saltaran antes de empezar a escribir el valor.

De esta manera una vez realizadas estas 4 líneas de código, se habrá conseguido convertir los dos mensajes de 16 bits cada uno en uno solo de 32 bits.

Una vez realizado esto, es necesario traducir estos valores para poder hacer los cálculos de la salida con los valores reales. Mediante la función ***Buffer.readFloatLE()*** de la línea 10, se podrá hacer la lectura de un float number de 32 bits con un formato Endian específico, en este caso el Little Endian funciona correctamente. El valor de ***control*** será igual al valor real de la variable ***control*** del PLC.

En la línea de código número 12, mediante la instrucción ***"flow.set("control",control)"*** se guardará el valor de la variable control en una variable global llamada control, de esta manera podremos usar el valor de esta variable en los distintos nodos del flow sin necesidad de que estén conectados.

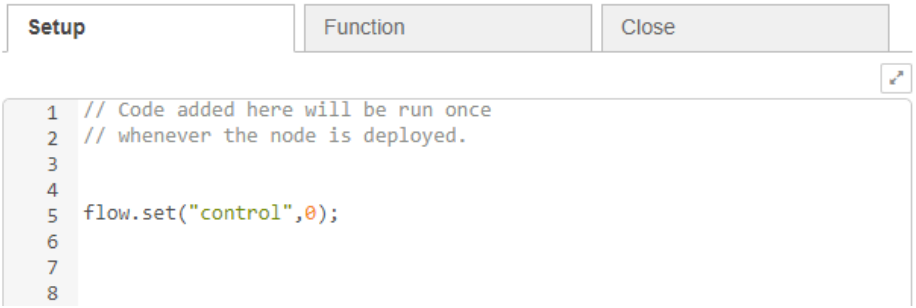


Figure 80. Declaración de variables generales

Como podemos ver en la imagen, en la pestaña Setup de cada nodo se pueden declarar las variables globales e inicializarlas.

Finalmente, en las últimas dos líneas de código,primero igualamos la variable "***control"*** al ***msg.payload*** del nodo, para finalmente con la instrucción ***return msg***, poder enviar el valor por la salida del nodo.

Por otro lado, como se puede observar en la *Figura 81* (la de comunicaciones), tenemos configurado otro nodo de función. Este nodo tendrá la función de devolver el valor del retardo de la comunicación, es decir, la diferencia de tiempo entre que el PLC envía el

valor y que Node-Red lo recibe. Esto nos servirá más adelante para realizar un estudio de la dinámica de las comunicaciones.

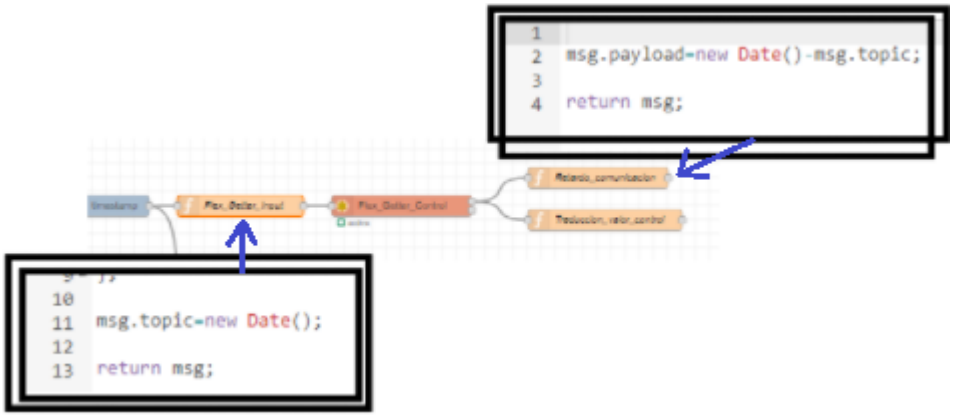


Figure 81. Retardo de la comunicación

"new Date()" permite guardar el valor de la fecha en el instante en el que se ejecuta. De esta manera, al usar esta instrucción antes y después del ***flex getter*** podremos calcular el retardo de las comunicaciones mediante una sencilla resta.

4.4.3 Configuración de los nodos de lectura

En este apartado se explicará cómo se configuran los distintos nodos para enviar los valores de la salida de la planta al PLC. Esto se realizará mediante ***modbus flex write*** ya configurado préviamente. Se usará el mismo Inject con el que se realiza la lectura.

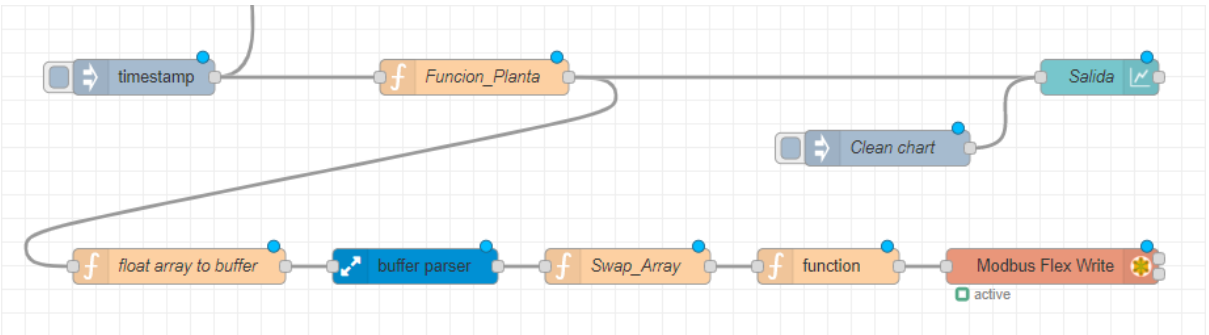


Figura 82. Nodos de escritura

Una vez presionado el **Inject** y realizado el cálculo de la función de la planta que observamos en la *Figura 81* (dinámica de la planta), mediante el nodo **Función_Planta** que observamos en la imagen anterior, se recogerá el valor de la función de la planta para enviarlo.

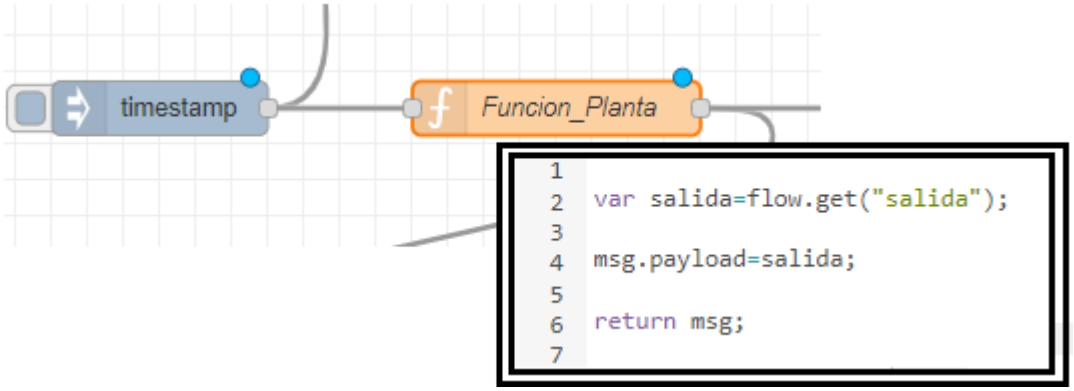


Figura 83. Funcion planta

Como podemos observar en el código de la imagen, primero mediante un ***flow.get()*** obtenemos el valor de la salida de la planta, para después enviarlo por la salida de este nodo.

Hay que tener en cuenta que ***"salida"***, la que está dentro del get, es una variable global, es decir, el cálculo de esta variable se ha realizado en el nodo Función-Planta, que se explicará más adelante, que forma parte de la dinámica de la planta, en ese propio nodo se ha inicializado como variable global para que luego se pueda utilizar en los otros nodos.

Una vez que tenemos el valor de la salida, hay que convertir este valor en un vector de dos enteros de 16 bits, es decir, hay que realizar el procedimiento contrario al del apartado ***"4.4.3 Configuración de los nodos de lectura",*** ya que como podemos recordar, modbus trabaja con paquetes de información de 16 bits, y el valor con el que trabajamos nosotros es un valor Real de 32 bits.

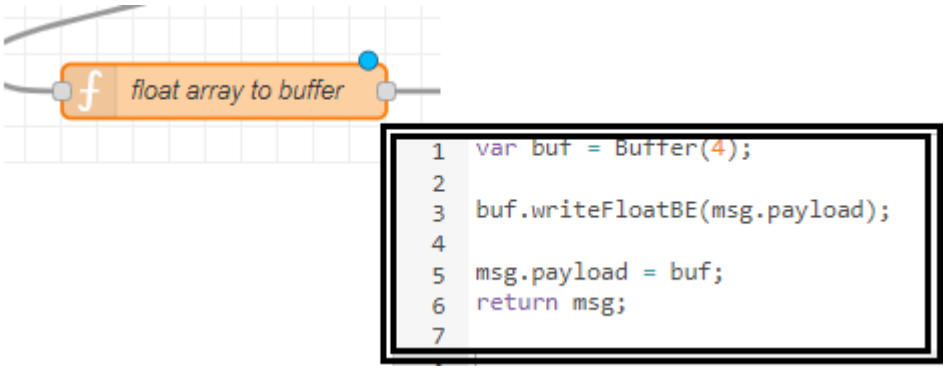


Figura 84. Float into buffer

Primero de todo, crearemos un **Buffer** de 4 bytes donde mediante una instrucción "***buffer.writeFloatBE()"*** , escribiremos el valor de la salida ("***msg.payload=salida"*** en esa

línea de código) en una variable que crearemos a la que llamaremos ***"buf"***. Esta variable buf será la que enviaremos por la salida del nodo.

En este caso, en vez de realizar la conversión mediante código, se utilizará un nodo llamado **Buffer Parser.**

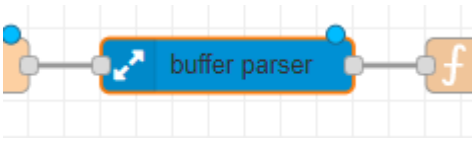


Figure 85. Nodo buffer parser

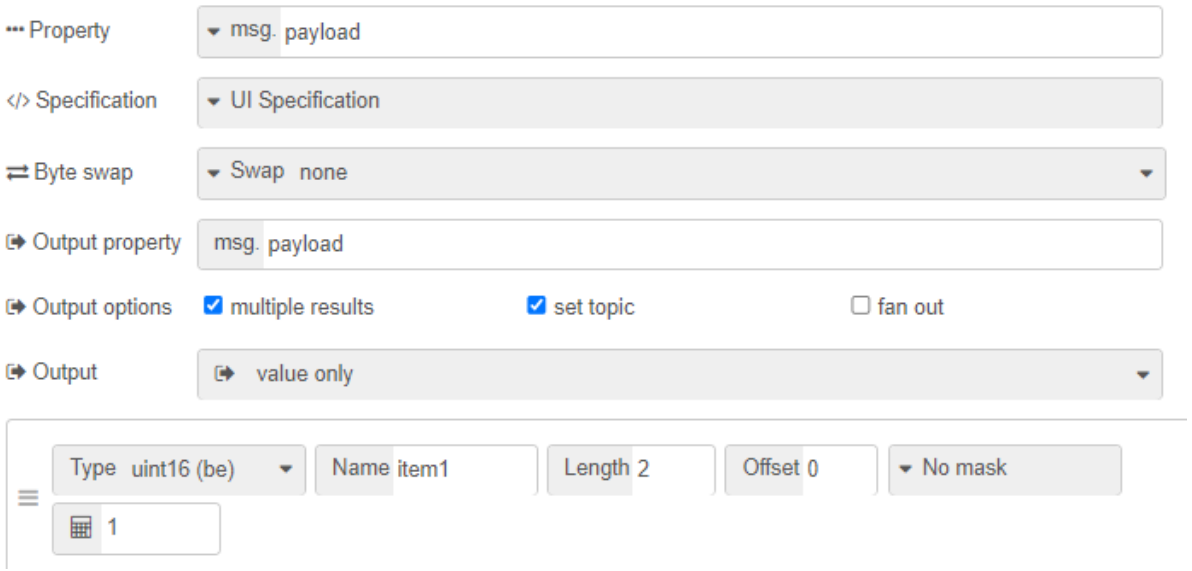


Figura 86. Configuración buffer parser

El nodo **Buffer Parser** tal y como lo hemos configurado en la imagen anterior, permitirá convertir el buffer de 32 bits del valor de salida, en un vector de dos valores de 16 bits.

Dicho vector nos devolverá los valores intercambiados de posición, por lo que necesitaremos ordenarlo.

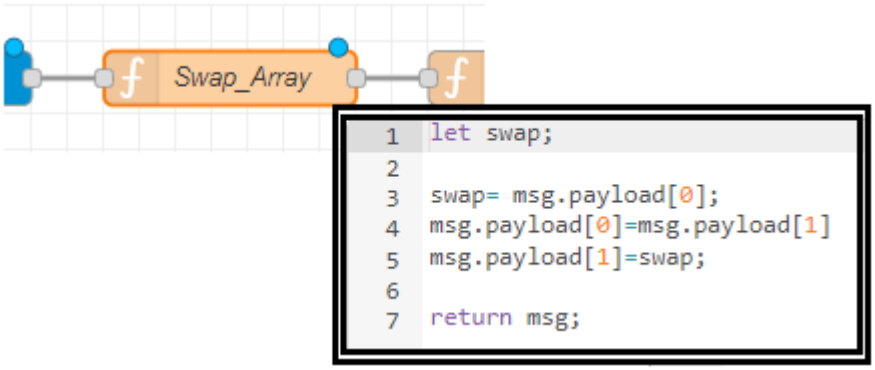


Figura 87. Nodo Swap array

Mediante un nodo de función ***Swap Array*** podremos crear un código que nos permita cambiar el orden del vector. Para ello necesitaremos a parte de los dos valores del vector, una tercera variable que llamaremos "***swap"*** para realizar el intercambio de posición. De esta manera primero igualamos el valor de la primera posición del vector "***msg.payload[0]"*** a "***swap",*** de esta manera ya podremos igualar el valor de la segunda posición del vector "***msg.payload[1]"*** en la primera posición del vector. Con esto se habrá intercambiado el orden de la segunda posición a la primera sin ningún problema. Finalmente igualamos el valor de "***swap"***, que será el valor de la primera posición, a la segunda posición del vector.

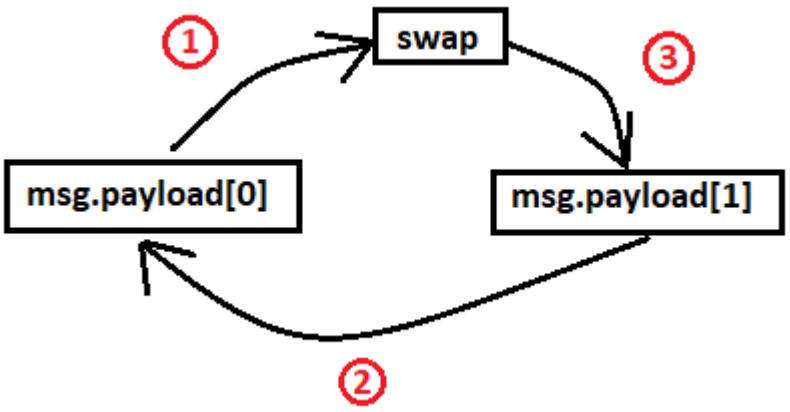


Figura 88. Representación del intercambio de valores

Finalmente, se enviará el valor de la salida mediante el nodo "***modbus flex write"*** que necesitará de un nodo de función para configurar los parámetros de entrada con tal de realizar la comunicación con el valor de salida del PLC.

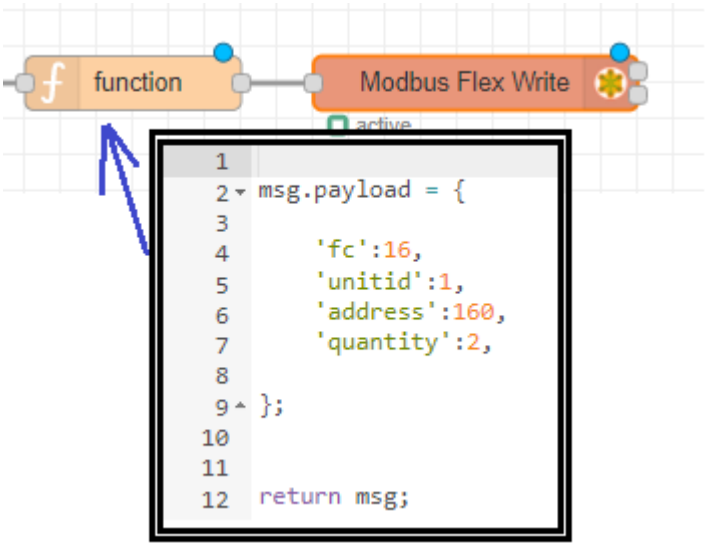


Figura 89. Configuración flex write

Estos son los apartados que se han de configurar:

- Necesitamos hacer la escritura de "multiple registers", por lo que ***'fc'*** ha de ser igual a 16.
- ***"unitid"*** (0..255 tcp | 0..247 serial).
- ***"address"*** (dirección de lectura) es igual a 160. Esta dirección es la que se le ha asignado a la variable ***"salidanodo"*** en el PLC.
- ***"quantity"*** la igualamos a 2. Esto se debe a que modbus intercambia paquetes de datos de 16 bits.

4.4.4 Flujo de la dinámica de la planta

Mediante el uso de otro Inject podremos realizar el estudio de la dinámica de la planta. La ecuación de la planta al estar en Node-Red tiene una dinámica muchísimo más rápida que la de las comunicaciones, es decir, el cálculo del valor de la salida se puede realizar más veces comparado con la velocidad de la comunicación entre Node-Red y PLC.

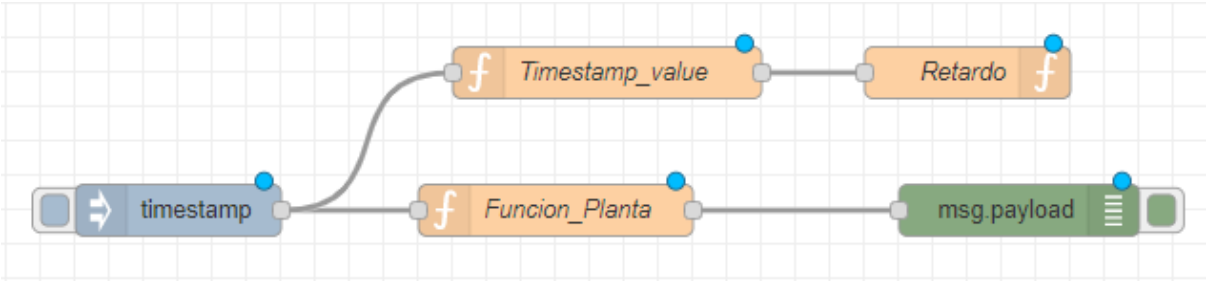


Figura 90. Nodos de la dinámica de la planta

La ecuación de la planta la aplicaremos en el nodo ***"Funcion_Planta"***de la imagen anterior, se configurará de la siguiente manera:

Primero inicializamos la variable global ***"salida"*** a 0:

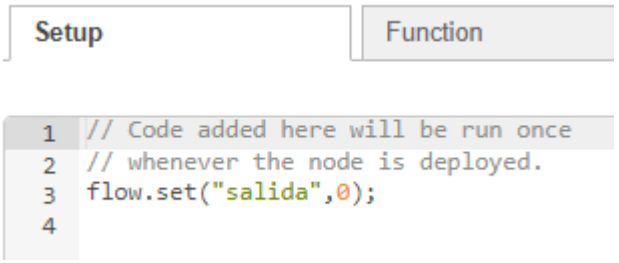


Figura 91. Variable global salida

A continuación crearemos un código en el cual se realizará el cálculo del valor de la salida mediante la ecuación de la planta:

```
1
2  let salida=flow.get("salida");
3  let retardo=flow.get("retardo");
4  let control=flow.get("control");
5
6  if(retardo>3000){
7
8      salida=0.244*control+0.951*salida;
9
10 }
11
12 else {
13
14     salida=0;
15
16 }
17
18 flow.set("salida",salida);
19
20 return msg;
21
```

Figura 92. Configuración función de la planta

Necesitaremos el valor de la ***salida***, del ***control*** y una variable ***retardo*** que nos permitirá realizar un retraso temporal de la salida con tal de conseguir un mejor control.

Mediante una ***if,*** podremos realizar el retardo con la condición de que hasta que esta variable no sea mayor de 3 s, que la salida sea igual a 0. Una vez que el retardo es mayor, se

ejecutará la ecuación de la planta. Guardaremos el valor de la salida para el siguiente cálculo de la salida, es decir, la ecuación de la planta depende del valor de la salida anterior.

La configuración del retardo la realizaremos mediante dos nodos de función codificados de la siguiente manera:

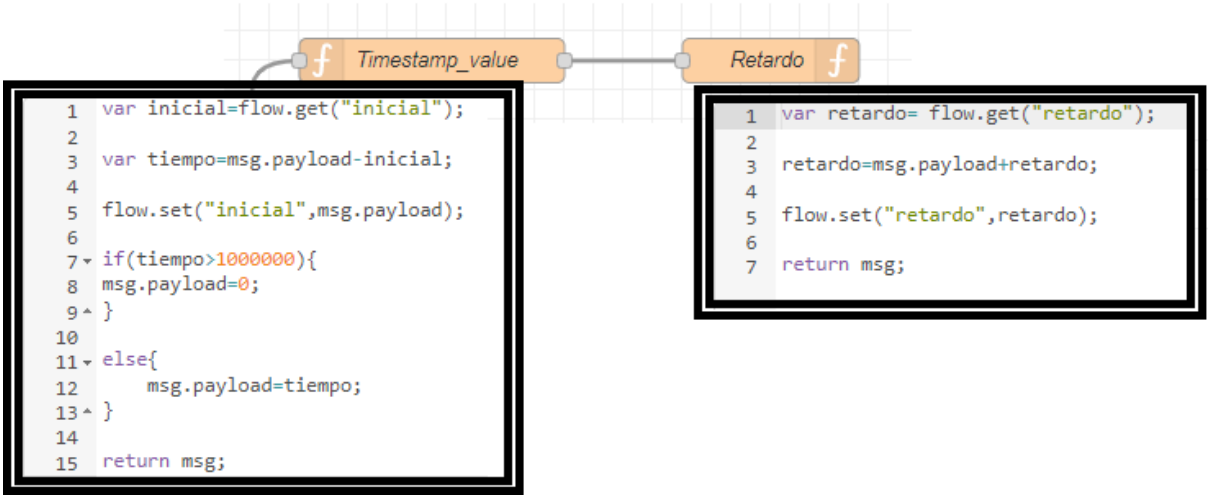


Figura 93. Retardo

El msg.payload del inject tiene el formato timestamp. Timestamp es una marca temporal, una secuencia de caracteres que denotan la fecha y hora de un determinado evento. En este caso nos devolverá el tiempo en milisegundos des del 01/01/1970.

El primer nodo nos interesa que nos devuelva el valor de los intervalos que se determinan en el Inject. Por lo que crearemos una variable tiempo que será la diferencia entre el valor timestamp de ese instante con el del anterior, de esta manera se nos devolverá constantemente el valor del intervalo.

Inicialmente usaremos un "**if**" con la condición de que si el tiempo es mayor a 100000 (un valor mayor al de los intervalos pero menor al timestamp), que devuelva msg.payload=0, de esta manera conseguiremos realizar correctamente el cálculo de la diferencia. Esto se realiza para que de alguna manera se guarde el primer valor del timestamp, porque si no guardamos este primer valor, la diferencia será constantemente 0.

En la segunda ejecución del nodo tiempo será menor que 100000 por lo que se ejecuta el **else** y msg.payload será igual a tiempo, que es la diferencia, es decir, será igual al intervalo.

Ejemplo de ejecución:

intervalo del Inject = 1 s

1 r Inject--->

(setup del nodo) ----> inicial=0 (variable global)
msg.payload=timestamp ----> msg.payload=1235498**7912**
tiempo=msg.payload-inicial ----> tiempo=1235498**7912**-0
flow.set (...) ----> inicial=1235498**7912**
if (...) / else(...) ----> msg.payload=0
return msg ----> **msg=0**

2 ndo Inject--->

msg.payload=timestamp ----> msg.payload=1235498**8912**
tiempo=msg.payload-inicial ----> tiempo=1235498**8912**-1235498**7912**
flow.set (...) ----> inicial=1235498**8912**
if (...) / **else(...)** ----> msg.payload=tiempo // tiempo=1000ms
return msg ----> **msg=1000**

3 er Inject--->

msg.payload=timestamp ----> msg.payload=1235498**9912**
tiempo=msg.payload-inicial ----> tiempo=1235498**9912**-1235498**8912**
flow.set (...) ----> inicial=1235498**9912**
if (...) / **else(...)** ----> msg.payload=tiempo // tiempo=1000ms
return msg ----> **msg=1000**

… (msg será constantemente 1000ms. El msg que enviará este nodo será el valor del intervalo.)

Una vez realizado esto, seguiremos con el siguiente nodo **Retardo**, como podemos observar, se realiza una suma del valor entrante del nodo ***Timestamp Value*** más el valor de la constante **retardo** que se irá actualizando por cada ejecución del nodo, haciéndose cada vez mayor.

Siguiendo con el ejemplo anterior, el nodo **Retardo** devolverá estos valores:

1 r Inject--->
(setup del nodo) ----> retardo=0 (variable global)
msg.payload=msg (***Timestamp_Value***) ----> msg.payload=0
retardo=msg.payload+retardo ----> retardo=0+0=0
flow.set (...) ----> retardo=0
return msg ----> **msg=0;**

2 ndo Inject--->

msg.payload=msg (***Timestamp_Value***) ----> msg.payload=1000ms
retardo=msg.payload+retardo ----> retardo=1000+0=1000ms

flow.set (...) ----> retardo=1000
return msg ----> **msg=1000;**

3 er Inject--->

msg.payload=msg (***Timestamp_Value***) ----> msg.payload=1000ms
retardo=msg.payload+retardo ----> retardo=1000+1000=2000ms
flow.set (...) ----> retardo=2000
return msg ----> **msg=2000;**
… (msg será 3000, 4000, 5000...)

De esta manera realizaremos un contador que nos servirá como marcador del retardo.

4.4.5 Dashboard

Mediante un nodo chart de la librería dashboard de Node-Red podremos muestrear la gráfica de los valores de la salida de la planta.

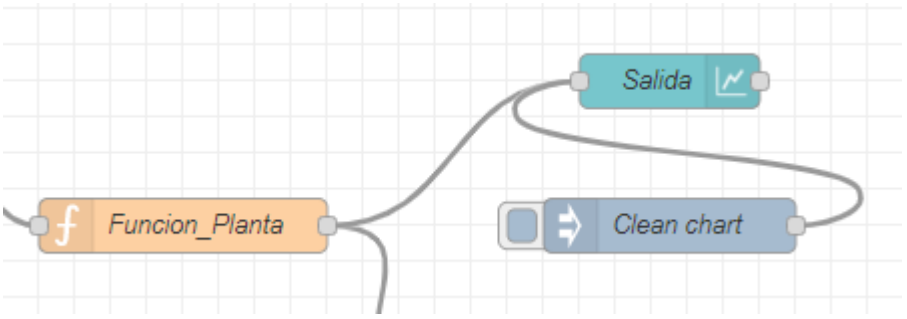


Figura 94. Dashboard

Haciendo doble click sobre el nodo ***Salida*** se podrá configurar la gráfica de la salida.

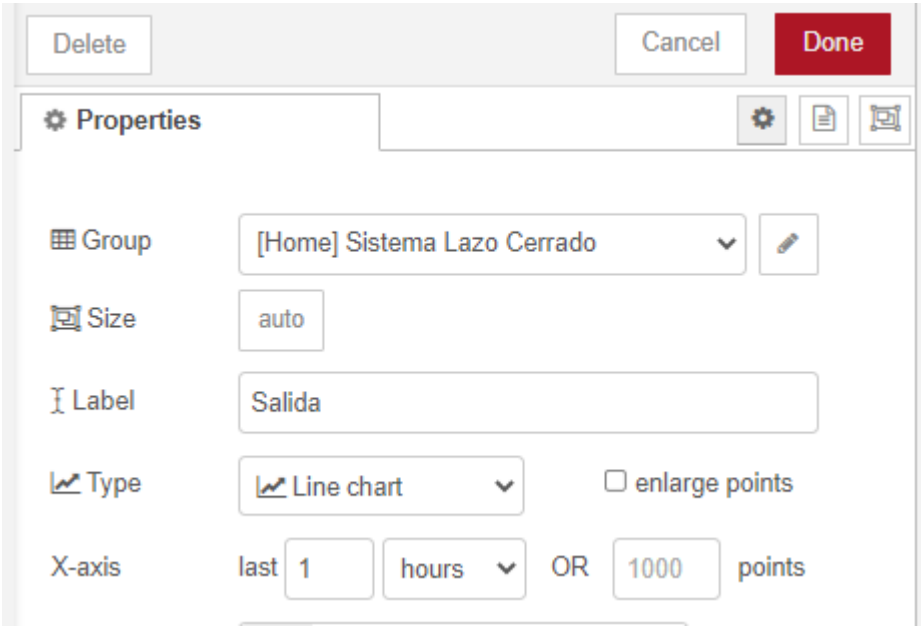


Figura 95. Propiedades del nodo Salida

En esta pantalla de propiedades se permite la configuración de los elementos que forman parte de la gráfica que queremos mostrar. Este es un ejemplo de la gráfica configurada:

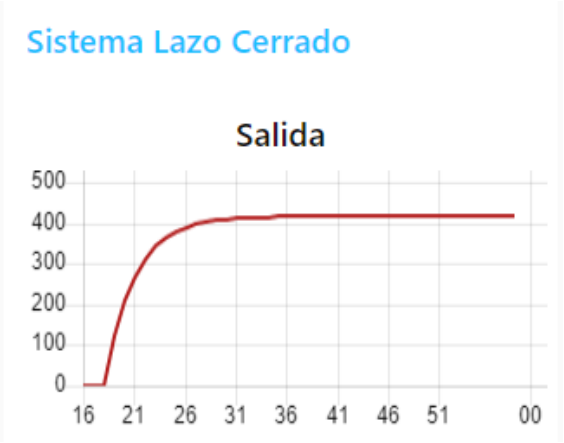


Figura 96. Ejemplo gráfica dashboard

Por otro lado, también se implementa un nodo Inject que permitirá limpiar la gráfica de la salida.

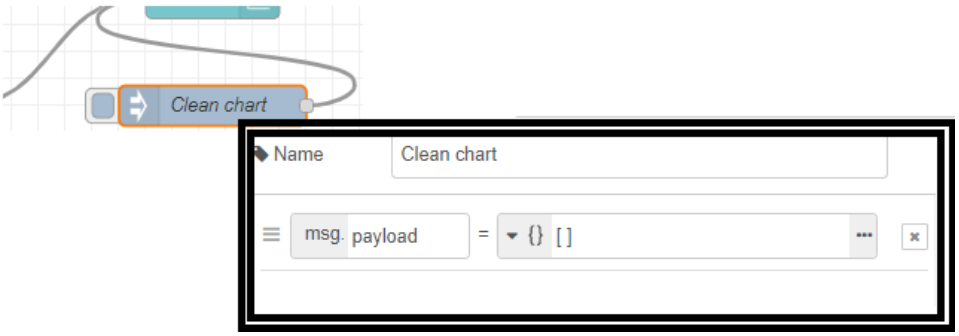


Figura 97. Clear chart

Este nodo inyectará un ***msg.payload*** con un vector vacío permitiendo así dejar la gráfica en blanco.

Atencion!!

En el laboratorio, el software Node-RED es de una versión anterior donde no se encuentra el apartado **SETUP** para inicializar las variables globales, lo que no permitirá que el programa funcione correctamente.

Para solucionar esto, se configurará otro nodo Inject conectado a un nodo de función dónde se inicializan las variables globales. Dicho nodo se ha de configurar para que se inyecte antes que los otros nodos Inject.

Figura 98. Inicializar variables globales

5. Validación y Ensayos

Este apartado consta de las distintas pruebas que se han realizado durante la programación de los distintos dispositivos del proyecto con el objetivo de validar el funcionamiento del proceso de control diseñado.

La validación del proceso de control se realizará desde un punto de vista funcional y operacional. Las pruebas funcionales nos permitirán comprobar el funcionamiento de de las pequeñas configuraciones o bloques individuales mientras que con la validación operacional podremos validar todo el conjunto. En este apartado se combinarán estos dos puntos de vista para conseguir una explicación más completa. Se presentarán explicaciones funcionales y comprobaciones operacionales con tal de representar de mejor manera el funcionamiento correcto del proceso de control.

Mediante el uso de las tablas de animación podremos visualizar los valores de cada variable por separado, esto nos facilitará el proceso de validación.

Las pruebas las realizaremos en dos partes, por un lado, se comprobará que las variables seleccionadas en la pantalla HMI afecten a las del PLC, es decir, que todas las variables de control estén vinculadas correctamente, y por otro lado, se realizarán las mismas pruebas, en este caso, con la conexión PLC / Node-RED, comprobando que el direccionamiento sea el correcto mediante el intercambio de información.

5.1 Validación variables PLC / HMI

Para la primera prueba de validación entre PLC/HMI, se comprobó que todas las direcciones de las variables de control están conectadas y direccionadas correctamente. Dicha comprobación se puede justificar mediante la visualización de la activación de las variables del PLC una vez seleccionadas en la pantalla HMI.

Para esta validación se ha observado primero la puesta en marcha del proceso. Funciona correctamente.

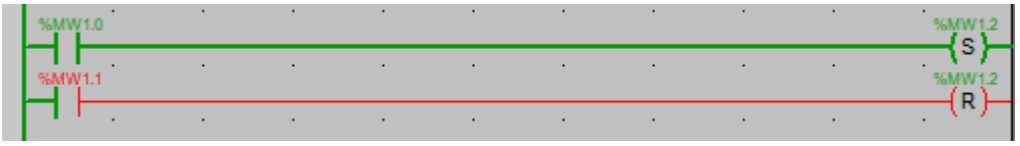


Figura 99. Validación de la puesta en marcha

Después se ha comprobado que las tres variables de control se activan adecuadamente y como se había previsto, lo hacen.

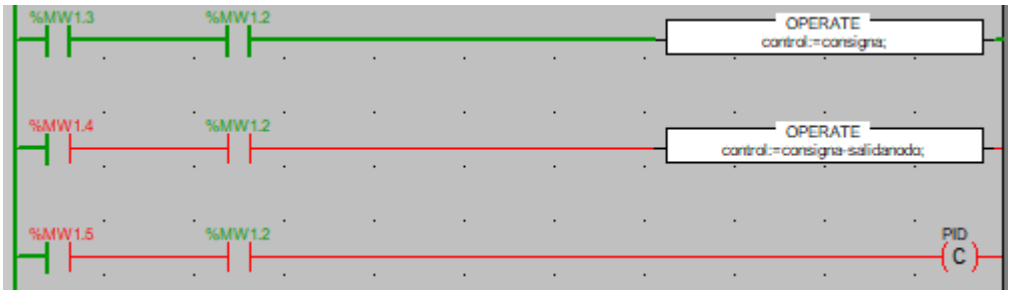


Figura 100. Validación proceso de control

A continuación se adjuntará un ejemplo completo para la validación del proceso de control entre PLC/HMI.

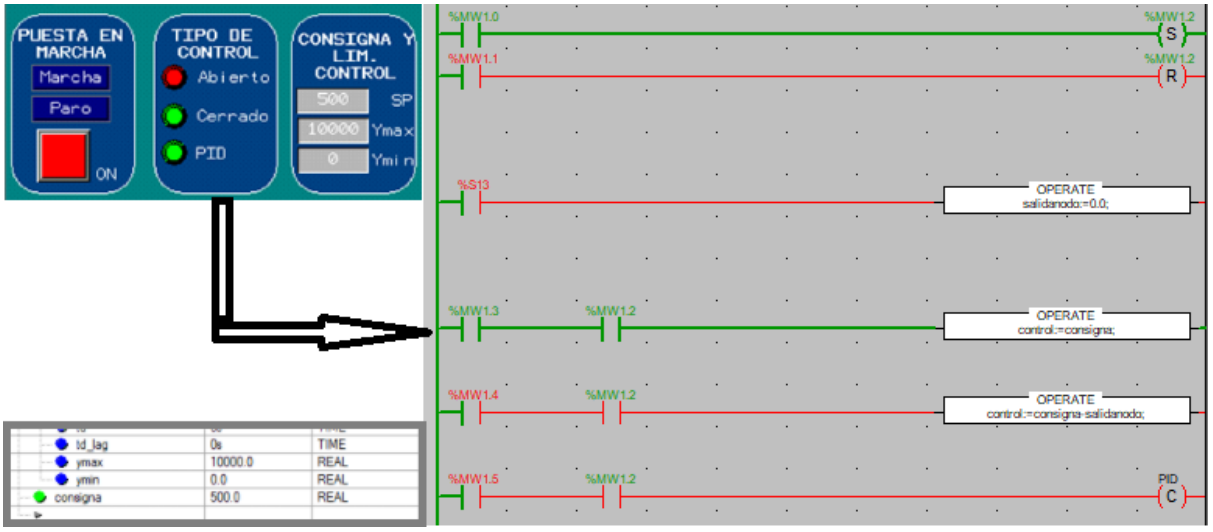


Figura 101. Validación de las variables

Como podemos observar, el proceso funciona correctamente.

La segunda prueba consta de la validación de los parámetros del PID. Igual que en la primera prueba, a través de la pantalla HMI, se ha introducido el tipo de control y los parámetros correspondientes. Todas las variables se han vinculado correctamente y el proceso funciona perfectamente. A través de la tabla de animación en Unity se puede ver como los valores se han enviado sin problemas.

param		Para_PID
gain	4.0	REAL
ti	966ms	TIME
td	662ms	TIME
td_lag	0s	TIME
ymax	10000.0	REAL
ymin	0.0	REAL
consigna	500.0	REAL

Figura 102. Validación parámetros PID

Tal y como se ha explicado en el apartado anterior, esta validación se ha realizado variable por variable para comprobar el funcionamiento de estas de manera individual.

5.2 Validación variables PLC / Node-RED

Para la validación de estas variables compartidas entre Node-RED y el PLC se realizarán dos pruebas.

En la primera prueba se ha realizado la validación del envío de información del PLC y la lectura y procesado de la información mediante Node-RED. Este proceso funciona correctamente como podremos ver a continuación, con el valor de la consigna del ejemplo anterior.

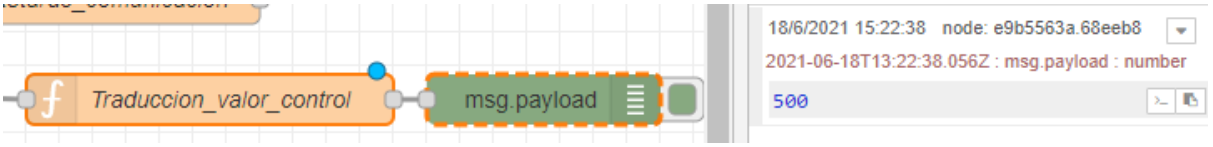


Figura 103. Validación valores PLC/Node-RED

Para la segunda prueba, se realizará el mismo proceso, pero en este caso, Node-RED envía la información y el PLC realiza la lectura. Este proceso funciona correctamente como podremos ver a continuación.

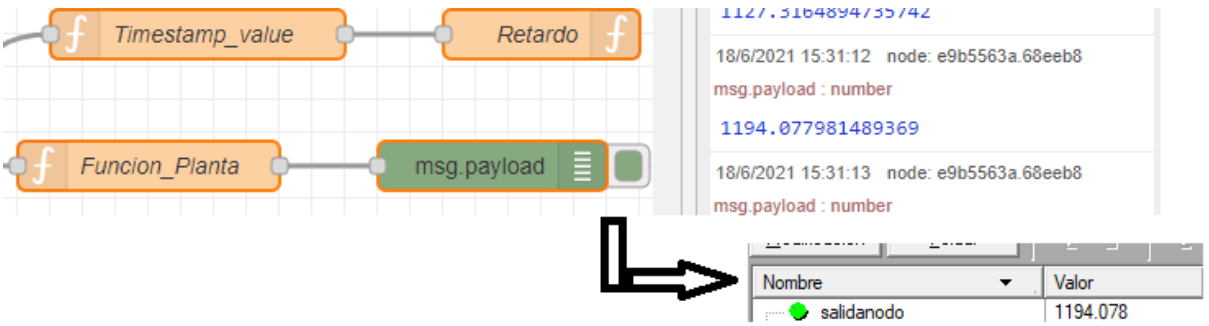


Figura 104. Validación valores Node-RED/PLC

Por otro lado, se han realizado ciertas validaciones para comprobar que se respetan las operaciones del PLC para el cálculo de los distintos lazos.

Primero se ha realizado una prueba mediante la configuración del proceso de control para realizar un control en lazo abierto y ha funcionado correctamente ya que el valor de control permanece constante .

Para la segunda prueba, se realiza la validación del proceso de control en lazo cerrado. Esto ha funcionado correctamente y se puede comprobar viendo que el valor de control va aumentando hasta estabilizarse en el valor de la consigna.

Para la tercera validación, se ha realizado el control en lazo cerrado con PID e igual que los casos anteriores mediante los valores de salida se ha comprobado el correcto funcionamiento de este proceso.

5.3 Ensayos

Para este subapartado de ensayos se ha decidido hacer un estudio de la curva de reacción del proceso de control. Se configurará el proceso mediante la pantalla HMI para realizar el control de los distintos tipos de lazo . De esta manera se podrá verificar que la aplicación no solo funciona sino que los valores adquiridos son los indicados y lógicos.

Para la realización de estas pruebas se ha tomado como ejemplo el proceso descrito en la Práctica 9 que se realiza en la UPC (Terrassa). Los valores de consigna y límites de la salida son los siguientes:

Figura 105. Valor de control

Respuesta en lazo abierto:

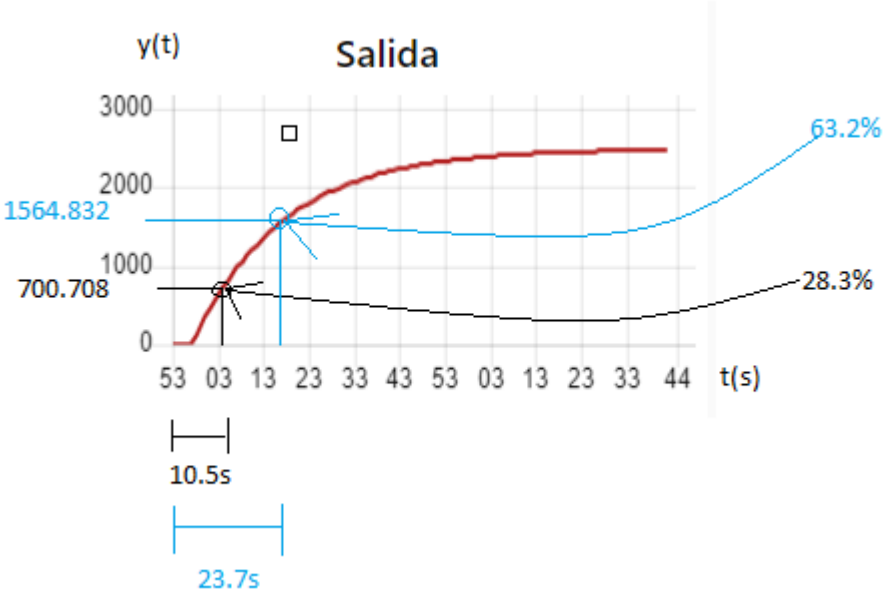
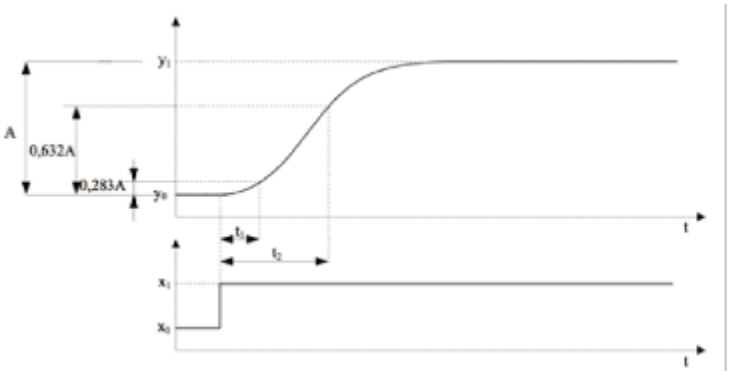


Figura 106. Respuesta en lazo abierto

Como podemos observar en la imagen de la respuesta en lazo abierto, el sistema no logra estabilizarse en el valor deseado a causa de que no hay realimentación.

Se han indicado ciertos puntos que representan los tiempos cuando la respuesta alcanza el 28,3% (t_1) y el 63,2% (t_2) de su valor final. A partir de las características de esta respuesta, se determinarán los parámetros de la curva de reacción (K, τ_o ,τ_p) .



$$\tau_p = 1{,}5 \cdot (t_2 - t_1)$$
$$\tau_o = (t_2 - \tau_p)$$

Figura 107. Parámetros de la curva de reacción

Constante de ganancia del proceso (K). Se define este valor como la relación existente entre el incremento de la señal de salida respecto al incremento de la señal de la consigna, resultado de aplicar la señal en escalón.

Constante de tiempo del sistema (τ_p).

Tiempo muerto o retardo (τ_0). Es el tiempo que transcurre entre el instante en que se produce el cambio de escalón de la consigna y el punto en que la tangente cruza con el valor inicial de la variable controlada (y_0).

Mediante los valores de la curva de reacción se podrá calcular los valores característicos del proceso.

K. División entre el incremento de y (desde 0 hasta el valor de estabilización) y el valor de incremento de consigna (500):

K= 2476/500 =**4.952**

t_1. Tiempo en que la respuesta es un 28,3% del valor final:

t_1= 10.5s

t_2. Tiempo en que la respuesta es un 63,2% del valor final:

t_2=23.7s

Cálculo de los parámetros:

τ_p = 19.8

τ_0 = 3.9

Los valores adquiridos en este apartado son necesarios para más tarde calcular los valores de los parámetros del controlador PID.

Respuesta en Lazo Cerrado:

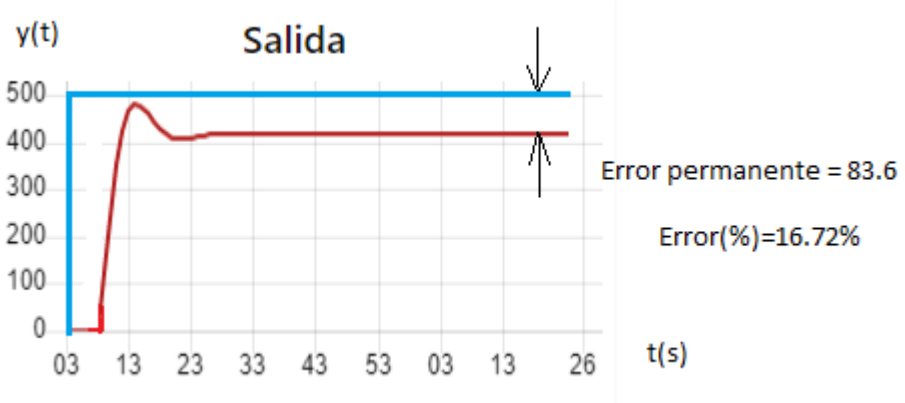


Figura 108. Respuesta en lazo cerrado

En esta gráfica podemos observar la respuesta de la planta y deducir que el resultado es lógico ya que como ya he mencionado anteriormente, el control en lazo cerrado se basa en la realimentación que permite aproximar el valor de la salida al valor deseado. Como podemos observar también, existe un error permanente de un 16,72%.

El error permanente o estacionario es aquella diferencia que existe entre el valor deseado y el real, una vez que el real se estabiliza.

Respuesta en Lazo Cerrado con un controlador PID:

Como hemos mencionado anteriormente, con los parámetros de la curva de reacción calculados en el lazo abierto y mediante el método de sintonía de Ziegler y Nichols podremos calcular los parámetros del controlador PID.

Tipo de controlador	**K_p**	**T_i**	**T_d**
P	**$\tau_p/(K\tau_0)$**		
PI	**$0,9(\tau_p/(K\tau_0)$**	**$\tau_0/0,3$**	
PID	**$1,2(\tau_p/(K\tau_0))$**	**$2\tau_0$**	**$0,5\tau$**

Tabla 5. Ecuaciones parámetros PID

Sustituyendo los parámetros de la curva de reacción obtenemos lo siguiente:

Tipo de controlador	**K_p**	**T_i**	**T_d**
P	**1.046**		
PI	**0.94**	**13s**	
PID	**1.2552**	**7.8s**	**1.95s**

Tabla 6. Valores parámetros PID

Una vez tenemos los valores de los parámetros del controlador PID, iremos seleccionando mediante la pantalla el tipo de control que queremos realizar, se introducirán los parámetros para cada uno de los controles e iremos visualizando el comportamiento de la salida de la planta.

Para saber si los resultados que obtenemos son lógicos, en la siguiente tabla se podrán visualizar las características de cada una de las constantes del controlador PID.

Respuesta lazo cerrado	**Tiempo de subida**	**sobre-pico**	**Tiempo de establecimiento**	**Error**
Kp	Disminuye	Aumenta	No cambia mucho	Disminuye
Ki	Disminuye	Aumenta	Aumenta	Se elimina
Kd	No cambia mucho	Disminuye	Disminuye	No cambia mucho

Tabla 7. Características de las constantes del controlador PID

Control P

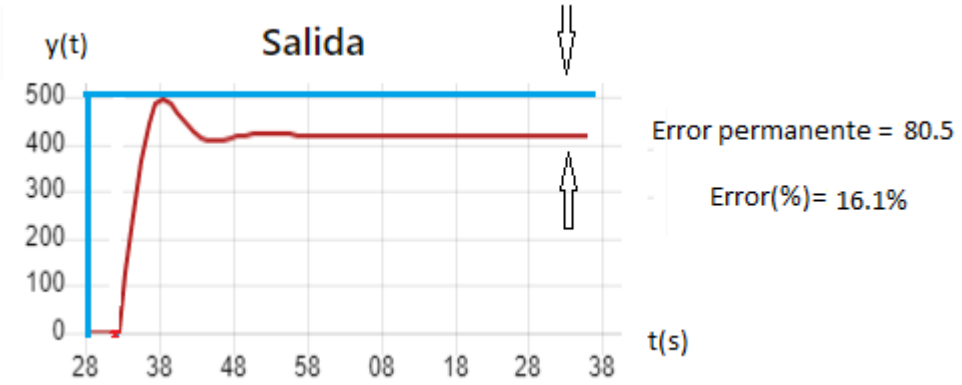


Figura 109. Respuesta del sistema con controlador P

Como podemos observar, mediante la comparación con la salida del sistema en lazo cerrado sin controlador, podemos ver que aumenta el sobre-pico y disminuye el error. Esto cumple con las características mencionadas anteriormente, aunque las diferencias sean mínimas, si pusiéramos una Kp mayor se acentuará más este comportamiento.

Control PI

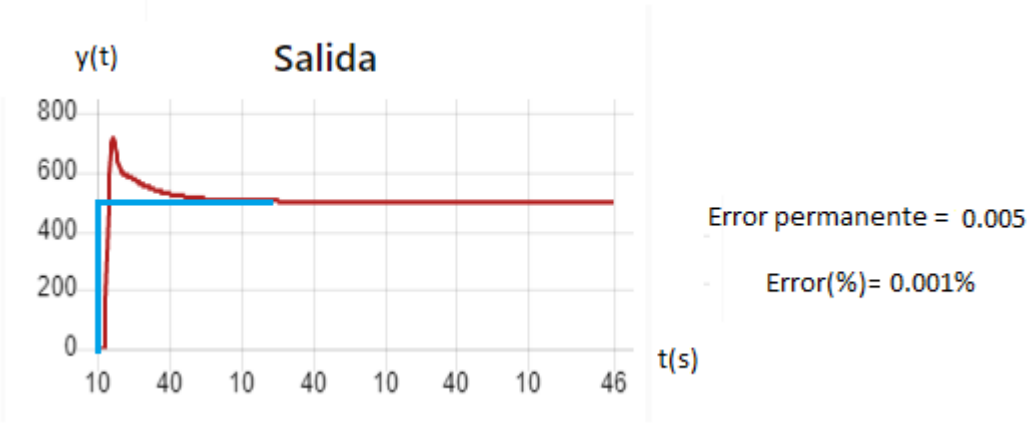


Figura 110. Respuesta del sistema con controlador PI

En este caso podemos ver claramente que se cumplen las características de los parámetros introducidos. Vemos como implementando el parámetro I el tiempo de subida es muchísimo menor, aumenta drásticamente el sobre-pico, el tiempo de establecimiento es mayor y se consigue eliminar el error.

Control PID

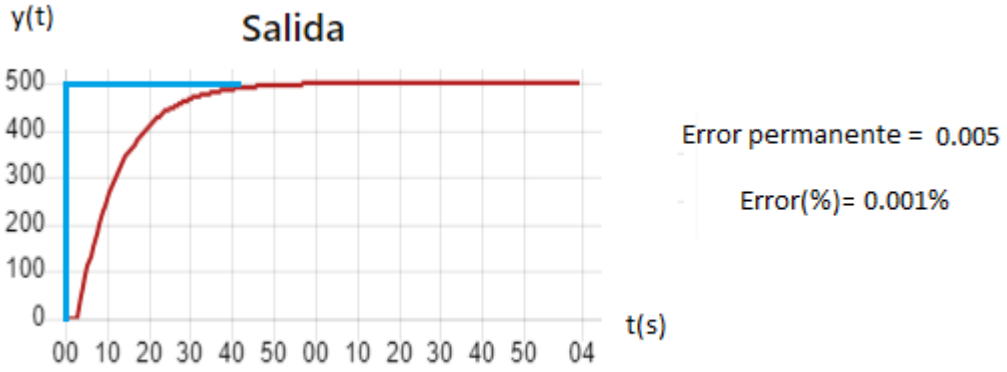


Figura 111. Respuesta del sistema con controlador PID

Aquí ya podemos observar un control más exacto y un comportamiento de la salida más estable y preciso. Como podemos ver, al aplicar el parámetro D conseguimos disminuir el sobre-pico y el tiempo de establecimiento.

Una vez habiendo realizado todos estos ensayos podemos llegar a la conclusión de que el proceso de control funciona correctamente y que los parámetros del controlador PID hacen su trabajo.

5.4 Pruebas sobre la planta

Finalmente, para cerrar el apartado de Validación y Ensayos se ha decidido hacer un estudio más allá de solo comprobar el funcionamiento del proceso de control, sino también realizar ciertas pruebas para comprobar los límites de nuestro proceso. Con estos límites nos referimos a probar de forzar la capacidad de las comunicaciones implementadas y de la planta.

Este estudio se realizará en dos partes, por un lado se estudiará el comportamiento de la dinámica de las comunicaciones y por el otro el de la planta.

Para forzar estos límites se jugará con los intervalos de inyección en Node-RED, ya que ahí se sitúa toda la parte de las comunicaciones y la planta en sí. Mediante el forzado de los valores de estos intervalos, para que sean cada vez más pequeños, se conseguirá realizar un intercambio de información cada vez más rápido, y lo mismo con el cálculo de la función de la planta.

Este estudio nos permite ver los límites de nuestro proceso y hasta qué punto se puede implementar en uno real, ya que algunos controles reales se han de realizar acorde con un sistema que puede ser muy rápido y para no omitir información de la salida del sistema, el intercambio de información ha de ser rápido.

5.4.1 Dinámica de las comunicaciones

Los ensayos que hemos realizado para comprobar los límites de la dinámica de las comunicaciones han sido a través de la configuración de los nodos donde se realiza la lectura del valor de control des del PLC.

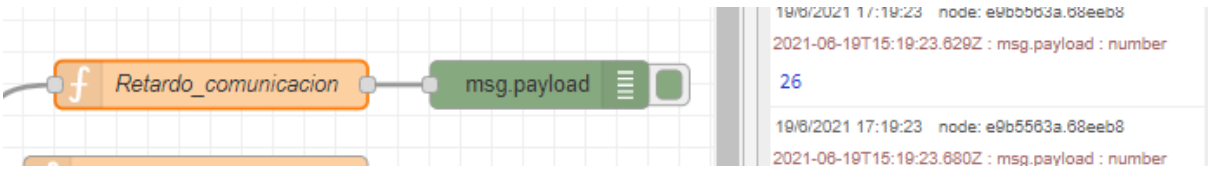


Figura 112. Nodo retardo comunicación

El nodo "***Retardo_comunicación"*** podremos visualizar el tiempo que ocurre entre que se realiza el Inject con la finalidad de realizar la lectura del valor de la variable ***control*** del PLC, hasta que el valor de la variable de control llega. Para realizar correctamente la lectura, este tiempo mencionado ha de rondar entre los 2 ms-26 ms como máximo.

Mediante las pruebas que hemos realizado hemos llegado a la conclusión de que el límite de la dinámica de las comunicaciones sucede cuando el Inject está configurado con intervalos de ***0.05 segundos***. Para tiempos más pequeños, el retardo de la comunicación se

hace cada vez más grande a causa de que se va acumulando el error del tiempo entre que Node-RED realiza la lectura hasta que se recibe el valor.

En el caso de realizar la misma prueba en el laboratorio y usando todos los dispositivos, sale que el límite de la dinámica de las comunicaciones sucede cuando el Inject está configurado con intervalos de ***0.03 segundos***.

5.4.2 Dinámica de la Planta

En el caso de la dinámica de planta, para ver cómo se comporta la salida del sistema en consecuencia a la disminución de los intervalos de inyección en Node-RED para aumentar la velocidad del cálculo de la ecuación del sistema, se realizará mediante la visualización de la señal de salida a través del dashboard de Node-RED.

Estas pruebas se han realizado empezando directamente con el proceso de control del sistema en lazo cerrado, esto se debe a que en el caso del lazo abierto no se podrá visualizar ningún cambio ya que el valor de la variable ***control*** en este caso es constante.

Pruebas sobre el sistema en lazo cerrado

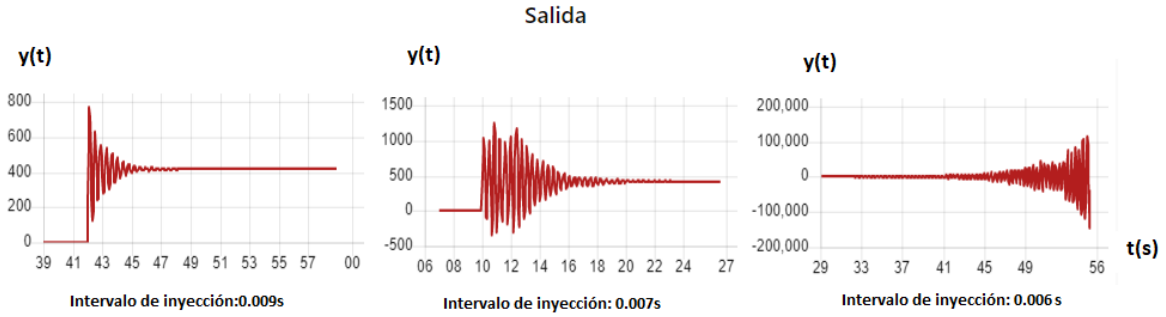


Figura 113. Graficas sistema lazo cerrado

Como podemos observar, el hecho de que la dinámica del sistema sea más rápida que la de las comunicaciones provoca sobre-picos cada vez más grandes, aumenta las oscilaciones y provoca que el tiempo de establecimiento sea mayor, hasta llegar al punto de que los sobreimpresos son tan drásticos que el sistema no consigue estabilizarse nunca.

Estas pruebas se han realizado de manera simulada, es decir, con el PLC en modo simulación y sin usar la pantalla HMI sino forzando los bits mediante las tablas de animación del Unity.

En el laboratorio y mediante el uso de la pantalla HMI para realizar el control, aunque el intercambio de información es más rápido (0.03s), al usarse el mismo software (Node-RED) para la simulación de la planta, salen los mismos valores para la dinámica del sistema.

Pruebas sobre el sistema en lazo cerrado con controlador PID

En este caso se presentarán los resultados del control realizado en el laboratorio, donde el límite de la dinámica del sistema está en 0.04s, es decir, la dinámica es 0.01s más lenta a causa de añadir el controlador PID.

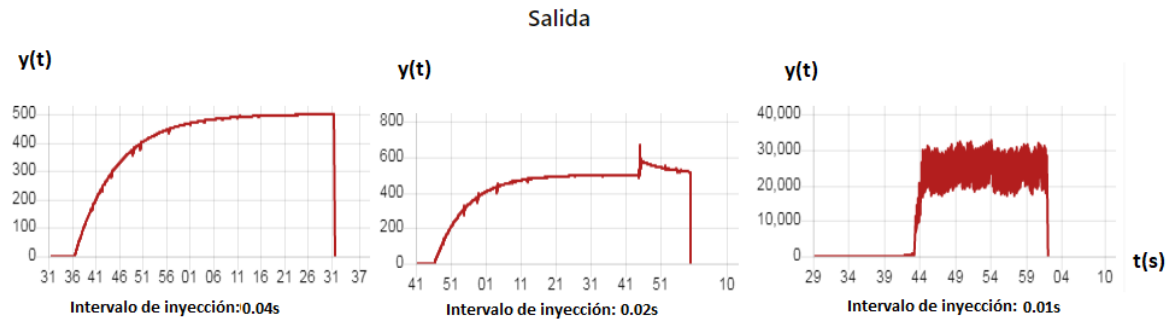


Figura 114. Gráficas lazo cerrado con PID

Como podemos observar el controlador PID no permite que la diferencia entre la dinámica del sistema y la de las comunicaciones sea tan grande. Con una diferencia de 0.03s el controlador ya devuelve valores del sistema bastante grandes y nunca se llega a realizar el control correctamente.

Finalmente, hay que tener en cuenta que en los sistemas reales existen límites que provocan una disminución en la capacidad del controlador para conseguir la respuesta deseada. Por mucho que se aumente la acción proporcional, llegará un momento en que el accionador se saturara y no podrá dar más de sí. Por ejemplo, en un sistema de control de temperatura, si la potencia máxima que se puede suministrar es de 1000 vatios, el controlador no podrá proporcionar más potencia para conseguir más rapidez de calentamiento ya que el límite está en 1000 vatios.

Por lo tanto hay que tener en cuenta que la velocidad de respuesta de los sistemas reales tiene ciertos límites que el control no podrá superar.

6.Conclusión

Habiendo finalizado el proyecto y realizado la simulación del proceso de control, se puede afirmar que se ha logrado realizar con éxito el diseño y la implementación de la aplicación de control de proceso industrial sobre la pantalla HMI, ya que se han conseguido validar todos los objetivos definidos para dicha aplicación.

Como objetivo principal del proyecto se ha conseguido estudiar, entender y configurar los distintos dispositivos que forman parte del proceso de control. Se ha podido implementar la aplicación de control en el PLC y controlarlo mediante una pantalla HMI. Esto me ha permitido aplicar los conocimientos adquiridos en el grado de Ingeniería Electrónica Industrial y Automática sobre la programación Ladder y la configuración de un HMI.

Por otro lado, se ha conseguido implementar la función de la planta en el software Node-RED y visualizar los resultados mediante el dashboard. Esto me ha permitido conocer este software y me ha hecho ver desde otro punto de vista como es la programación y cómo implementarla de manera más sencilla y visual.

Por último y no por ello menos importante, las comunicaciones. La configuración de las comunicaciones entre Node-RED/PLC y PLC/HMI me han supuesto un reto y me han hecho ver la importancia de cada uno de los protocolos de comunicación implementados.

En cuanto a las posibles mejoras para los posibles trabajos en un futuro, a continuación propondré ciertos puntos a implementar que podrían suponer un avance al proyecto.

1. Automatizar aún más el proceso de control para adquirir las gráficas automáticamente sin necesidad de que el operario tenga que intervenir.

2. Limitar los resultados de la salida a ciertas muestras donde se pueda visualizar por completo el comportamiento de la salida y que el proceso no sea infinito hasta que el operario lo pare.

3. Implementar por ejemplo el software Influxdb para la creación de una base de datos que recoja los valores de la salida de la planta y visualizarlos mediante Grafana. Esto supondría una mejora del proceso, ya que el dashboard de node red presenta gráficas sencillas y en ocasiones colapsa si es que la dinámica del proceso es muy rápida.

4. Añadir alarmas o que la aplicación avise cuando el proceso supere ciertos límites marcados por los valores máximos y mínimos de la salida de la planta.

7.Bibliografía

Pirámide CIM

≫ Pirámide CIM de Automatización Industrial 🤖, 2021. ATEC ENERGY BLOG [online], Available from: https://www.blog.atec-energy.com/2019/04/piramide-de-automatizacion-cim.html[Accessed 21 June 2021].

Salazar Serna & Correa Ortiz, 2013

Controladores Lógicos Programables

Controlador lògic programable - Viquipèdia, l'enciclopèdia lliure, 2021. Ca.wikipedia.org [online], Available from: https://ca.wikipedia.org/wiki/Controlador_l%C3%B2gic_programable[Accessed 21 June 2021].

Estructura de un PLC: Modulos de Memoria, 2021. Instrumentacion y Automatizacion Industrial[online], Available from: https://instrumentacionycontrol.net/estructura-de-un-plc-modulos-de-memoria/ [Accessed 21 June 2021].

Funcionamiento de un PLC, 2021. Electrinblog.wordpress.com [online], Available from: https://electrinblog.wordpress.com/2016/04/26/post-4/[Accessed 21 June 2021].

2021. Ieec.uned.es [online]. 2021. Available from: http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE6_1_1.pdf .[Accessed 21 June 2021],

ANUNZIA SOLUCIONS TECNOLÒGIQUES, S.L., 2021, PLCs | ACE, Automatisme i Control Elèctric. Acesa.es [online]. 2021. Available from:https://www.acesa.es/cs/p1/plcs [Accessed 21 June 2021].

Protocolos de comunicación

Problema productor-consumidor - Wikipedia, la enciclopedia libre, 2021. Es.wikipedia.org [online],Available from: https://es.wikipedia.org/wiki/Problema_productor-consumidor [Accessed 21 June 2021].

Modbus TCP/IP

Modbus - Wikipedia, la enciclopedia libre, 2021. Es.wikipedia.org [online], Available from: https://es.wikipedia.org/wiki/Modbus [Accessed 21 June 2021].

Modbus: Qué es y cómo funciona | Comunicaciones Industriales, 2021. aula21 | Formación para la Industria [online], Available from: https://www.cursosaula21.com/modbus-que-es-y-como-funciona [Accessed 21 June 2021].

2021. Modbus.org [online] , Available from: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf [Accessed 21 June 2021].

Ethernet

Ethernet/IP - Protocolo de red en niveles para aplicaciones de automatización industrial, 2021. Default [online], Available from: https://www.siemon.com/es/home/support/education/white-papers/03-10-13-ethernet-ip [Accessed 21 June 2021].

Qué es el protocolo Ethernet Industrial y cómo funciona | Red Informática, 2021. aula21 | Formación para la Industria [online], Available from: https://www.cursosaula21.com/que-es-ethernet-industrial/ [Accessed 21 June 2021].

Software Unity

2021. Lra.unileon.es [online], Available from: http://lra.unileon.es/sites/lra.unileon.es/files/Documents/plc/Unity_Pro/Manuales_Unity/Manual_Unity.pdf [Accessed 21 June 2021].

2021. Instrumentacionycontrol.net [online], Available from: https://instrumentacionycontrol.net/wp-content/uploads/2017/11/IyCnet_GuiaRapidaUnityPRO-min.pdf [Accessed 21 June 2021].

2021. Lra.unileon.es [online],Available from: http://lra.unileon.es/sites/lra.unileon.es/files/Documents/plc/Unity_Pro/Manuales_Unity/Unity_Manual%20de%20Referencia.pdf [Accessed 21 June 2021].
IEC 61131-3 - Wikipedia, la enciclopedia libre, 2021. Es.wikipedia.org [online],Available from: https://es.wikipedia.org/wiki/IEC_61131-3https://es.wikipedia.org/wiki/IEC_61131-3 [Accessed 21 June 2021].

Software Node-RED

Node-RED, 2021. Nodered.org [online],Available from: https://nodered.org/ [Accessed 21 June 2021].

Qué es Node-RED, 2021. Aprendiendo Arduino [online], Available from: https://aprendiendoarduino.wordpress.com/2020/03/05/que-es-node-red/ [Accessed 21 June 2021].

Aspectos de diseño de sistemas de control

Regulación automática - Wikipedia, la enciclopedia libre, 2021. Es.wikipedia.org [online],Available from: https://es.wikipedia.org/wiki/Regulaci%C3%B3n_autom%C3%A1tica [Accessed 21 June 2021].

Controlador PID - Control Automático - Picuino, 2021. Picuino.com [online],Available from: https://www.picuino.com/es/arduprog/control-pid.html [Accessed 21 June 2021].

MOYA, SAMUEL, 2021, Conceptos Básicos: Sistemas de Control | ISA Sección Central México. ISA Sección Central México [online]. Available from: https://www.isamex.org/intechmx/index.php/2018/12/24/conceptos-basicos-sistemas-de-control/ .2021. [Accessed 21 June 2021].

Controlador PID - Control Automático - Picuino, 2021. Picuino.com [online],Available from: https://www.picuino.com/es/arduprog/control-pid.html [Accessed 21 June 2021].

2021. Dea.unsj.edu.ar [online], Available from: http://dea.unsj.edu.ar/control2/ControladoresPID.pdf [Accessed 21 June 2021].

Introducción al Control De Procesos. (2020, 30 abril). YouTube. Available from: https://www.youtube.com/watch?v=P7wzaMH96Dg