INCIDENCIA DEL USO DE LAS HERRAMIENTAS DE PLANTPAX PARA UNA PROPUESTA DE PROGRAMACIÓN Y SUPERVISIÓN EN UN CASO DE ESTUDIO

CRISTIAN CAMILO ALEGRIA CASAS
LAURA ANDREA BERMÚDEZ CÓRDOBA

Trabajo de grado en Automática Industrial

Director: M.Sc. OSCAR AMAURY ROJAS ALVARADO

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Popayán, Septiembre de 2019

CRISTIAN CAMILO ALEGRIA CASAS
LAURA ANDREA BERMÚDEZ CÓRDOBA

INCIDENCIA DEL USO DE LAS HERRAMIENTAS DE PLANTPAX PARA UNA PROPUESTA DE PROGRAMACIÓN Y SUPERVISIÓN EN UN CASO DE ESTUDIO

Tesis presentada a la Facultad de Ingeniería
Electrónica y Telecomunicaciones de la
Universidad del Cauca para la obtención del
Titulo de:

Ingeniero en Automática Industrial

Director: M.Sc. OSCAR AMAURY ROJAS ALVARADO

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Popayán, Septiembre de 2019

Agradecimientos

A Dios por siempre ayudarnos, por poner en nuestro camino todas las oportunidades y permitirnos tomar las decisiones correctas para llegar hasta este punto.

A nuestras familias por su amor incondicional, apoyo y motivación en cada momento de nuestras vidas.

Al Magíster Oscar Amaury Rojas Alvarado el cual en su labor de director nos acompañó persistentemente y facilitó todo lo necesario para concluir satisfactoriamente el proyecto.

A la Universidad del Cauca por acogernos durante todo el camino, por permitirnos aprender de excelentes profesionales y por proporcionar la infraestructura y los equipos necesarios para desarrollar nuestro proyecto.

A nuestros amigos y compañeros que nos ayudaron en la recolección de datos y a todos los que con palabras de aliento y muestras de afecto nos apoyaron durante el camino hasta culminar nuestro trabajo.

Contenido

Página

Lista de Figuras

Lista de Tablas

Introducción

La automatización de procesos es hoy en día inevitable, ya que permite obtener resultados altamente eficientes, altos niveles de producción, mejoras en la calidad, disminución de costos, entre otros [1, 2, 3]. En la actualidad, la mayoría de industrias que cuentan con una gran cantidad procesos de producción, requieren equipos o máquinas que generalmente se encuentran controladas por medio de algoritmos ejecutados en *controladores lógicos programables* (PLC) [4], esto debido a la posibilidad de realizar modificaciones en los algoritmos de manera no muy compleja y a la rápida puesta en marcha de los mismos [1, 5, 6, 7].

Controlar los procesos es de vital importancia para las industrias, por lo tanto, conocer lo que realmente está pasando en el proceso y cómo reacciona es trascendental [8, 9], por ello, paralelo al uso de algoritmos de control son utilizadas las *interfaces hombre máquina* (HMI), las cuales son empleadas para realizar una representación fiel de los procesos, permitiendo una interacción de los operadores con una representación de los equipos de la planta [10]. Las aplicaciones de supervisión se masifican cada vez más a nivel industrial, esta tendencia se debe a la necesidad de tener un control más preciso y agudo de las variables de producción contando con la información relevante de los distintos procesos en tiempo real [9, 11].

Considerando que las industrias necesitan que sus algoritmos sean flexibles y también debido a que en ocasiones la estructura utilizada en ellos puede funcionar para otros procesos, se ha optado por encapsular y reutilizar código, esto permite programar y evaluar lógicas de control de una manera más rápida y eficiente [5]. De igual manera la reutilización de conocimientos estándares sobre el diseño de una interfaz gráfica reduce significativamente el esfuerzo de desarrollo de la aplicación dando un resultado consistente en la empresa, estandarizar el desarrollo de aplicaciones de supervisión ayuda a los programadores a aprovechar código existente permitiendo aplicar la experiencia de la industria de manera efectiva [12, 13].

A medida que las soluciones de automatización han implementado el uso de PLCs e interfaces de supervisión, para los encargados de desarrollar soluciones de automatización el tiempo para implementar la solución es una variable crítica que afecta de manera directa el

costo del proyecto [14]. Otros aspectos relevantes son la búsqueda de soluciones flexibles, esto para que se adapten a los posibles cambios en los procesos de producción, también el diseño de una solución lo suficientemente intuitiva para que cualquier programador, con los conocimientos necesarios, pueda entender y trabajar sobre un algoritmo o la interfaz de un proceso [15]. Lo mencionado son factores que no se han abordado a fondo con la programación convencional.

En busca de la reducción de tiempos y mejoras en la flexibilidad de las soluciones de automatización, se han realizados trabajos que buscan ser aportes y mejoras para los factores no abordados en la programación convencional. En el año 2004 se realiza un trabajo como propuesta para el desarrollo de un programa para un PLC en el que se busca optimizar este proceso por medio del uso de un software que genera el programa de manera automática. Este software tiene como base una estructura de programación basada en el modelo físico de *ISA 88*. La aplicación busca generar el algoritmo de acuerdo a una lista de equipos de entrada/salida y una filosofía de control [14].

En otro trabajo se realizó una propuesta para crear una estructura base para el desarrollo de programas con el fin de que el programador pueda utilizar mejor el tiempo de desarrollo en la parte de la solución que es única para problema en el que se encuentra trabajando. Para esto realiza una propuesta de arquitectura base que busca ser reutilizable y extensible, esta comprende: el manejo del sistema (manual o automático), manejo de alarmas, condiciones de seguridad, dispositivos de entradas (sensores) y salidas (actuadores), entre otros. Además en el trabajo se estudia la lógica utilizada en la programación convencional y se proponen cambios en busca de conseguir mejoras en los algoritmos y en el uso de los recursos del PLC [16].

Para entrar en contexto con alternativas que pueden ayudar a disminuir los tiempos de desarrollo de una solución es necesario conocer acerca de *PlantPAx*. *Rockwell Automation* (RA) propone un sistema moderno de control distribuido denominado *PlantPAx*, el cual se encuentra diseñado considerando estándares y basado en una arquitectura integrada. Este sistema cuenta con áreas claves como: ingeniería del sistema, plataformas de control y visualización, desempeño y optimización de procesos, tecnologías *Batch*, seguridad de procesos y control crítico, gestión de activos e información de planta [17, 18, 19]. *PlantPAx* ofrece un diseño y operación eficiente gracias a que cuenta con una biblioteca de objetos de proceso predefinida, conformada de instrucciones *Add-On* las cuales son objetos

de código reutilizable que contienen lógica encapsulada. Cada objeto se proporciona como instrucción *Add-On* importable que puede compartirse entre proyectos para crear una biblioteca común de instrucciones para acelerar la ingeniería de un proyecto a otro [20, 21].

Por otra parte las aplicaciones para el diseño de interfaces cuentan con una biblioteca con elementos de pantalla (objetos globales) y *faceplates* que permiten ensamblar rápidamente grandes aplicaciones HMI con estrategias probadas, funcionalidad avanzada y resultados confiables, estas son construidas teniendo en cuenta aspectos como el color, la funcionalidad y los símbolos, aspectos referenciados en estándares internacionales. Al utilizar la biblioteca de objetos de proceso de *Rockwell Automation*, el usuario final puede configurar una aplicación de control y supervisión utilizando objetos predefinidos para funciones comunes que ayudan a reducir el tiempo de ingeniería. Estos objetos son adecuados para muchos sectores de la industria [22, 23].

Sobre el uso de *PlantPAx* pueden encontrarse casos en donde ha sido implementado y ha traído beneficios para las empresas. Uno de estos casos es el de la empresa *McEnery Automation* la cual implementó una solución con un sistema *DCS* moderno en una empresa de lubricantes y petróleo, la cual buscaba solucionar un problema de aumento de ventas frente a la capacidad de producción con la que contaban. La solución fue implementada haciendo uso de herramientas y equipos de RA, en ella se destaca que: *"la biblioteca de objetos de proceso PlantPAx de RA brindó programación consistente y objetos gráficos que redujeron los costos de implementación y minimizaron las necesidades de capacitación del operador"*. Como resultado del uso de esta solución la empresa obtuvo aumentos inmediatos en producción, capacidad y eficiencia [20].

En 2015 se desarrolló un sistema de automatización de procesos basado en la plataforma *PlantPAx* para una planta de fibra de *lyocell* con una capacidad anual de 15.000 toneladas. El sistema ha realizado operaciones simples, optimización de la planta, arranque y apagado completamente automáticos de las unidades de la planta, control y diagnóstico de fallas de funcionamiento, el registro y documentación de todas las secuencias de operaciones y datos de proceso. El sistema de automatización de procesos fue diseñado con base a la plataforma *PlantPAx* de RA logrando una estructura modular de múltiples capas para garantizar el rendimiento, la escalabilidad y la confiabilidad. Como conclusión el autor dice que: *"la plataforma de RA proporciona una biblioteca perfecta y potente para el control avanzado de procesos, por lo que algunos problemas complejos tradicionales como el control*

de secuencias, el control modular multivariable se pueden resolver fácilmente, con lo anterior se puede acortar el ciclo de desarrollo de sistemas de automatización de procesos" [24].

Otro caso fue donde la empresa *Matrix Technologies (MT)* realizó la tarea de trasladar los procesos manuales de una empresa productora de alfombras de latex a procesos totalmente automatizados bajo la plataforma de *PlantPAx*. En este proyecto *MT* brindó apoyo a otra pequeña empresa de ingeniería OEM (Original Equipment Manufacturer) para llevar acabo la solución, como consecuencia *MT* no trataba directamente con la compañía química, lo que provocaba que se enfrentaran a desafíos relacionados con los tiempos, esto porque debían trabajar con el 50 % menos del tiempo que normalmente necesitaban para completar el proyecto y la falta de información del proceso con respecto a programas de PLC o HMI existentes. La empresa *MT* realizó el programa utilizando la biblioteca de *PlantPAx*, proporcionando mejores resultados en el menor tiempo posible, esto debido a que se utilizó el código segmentado y prefabricado proporcionado por la biblioteca de procesos, esto facilitó la tarea de programar *templates* para cada dispositivo del proceso y vincular las entradas y salidas al código.

De acuerdo a los casos expuestos anteriormente es posible evidenciar buenos resultados en la implementación de un sistema *PlantPAx*, otro factor importante que puede evidenciarse es que es una buena alternativa para todo tipo de procesos, lo que lo hace una herramienta potente y de gran uso en la industria. En el siguiente estudio se busca trabajar solo con una pequeña parte de todo lo que contiene un sistema *PlantPAx*, esto es específicamente la biblioteca de objetos de proceso (herramientas de *PlantPAx*) que busca agilizar el diseño de algoritmos para PLC y HMI, dos componentes fundamentales para controlar y monitorear cualquier proceso industrial. Para analizar el uso de las herramientas descritas se plantea un caso de estudio basado en uno de los procesos más comunes y en los cuales también el uso de las herramientas de *PlantPAx* ha sido exitoso, esto es la automatización de una parte del proceso de producción de cerveza, específicamente las etapas de macerado y cocción del mosto. En el caso de estudio no solo se busca hacer uso de las herramientas descritas si no también del uso de buenas prácticas de programación de algoritmos para PLC y de diseño de HMI de alto rendimiento, todo lo anterior enfocado a trabajar en una solución más cercana a las ofrecidas en ambientes profesionales en donde es realmente importante todos los detalles del desarrollo de una propuesta de programación y supervisión para un proceso.

De acuerdo a todo lo mencionado, en el siguiente trabajo se busca resolver la pregunta de investigación: *¿Cuál es la incidencia del uso de las herramientas de PlantPAx en los tiempos de desarrollo de una propuesta de programación y supervisión?*, para ello se contrasta con el método convencional utilizado en el desarrollo de una propuesta de este tipo. En este trabajo se define como método convencional la programación de algoritmos para PLC y diseño de HMI sin el uso de herramientas que faciliten su desarrollo, haciendo que estas tareas requieran un mayor esfuerzo.

Objetivos

Objetivo general

Evaluar el efecto de la implementación de las herramientas de *PlantPAx* en los tiempos de desarrollo de una propuesta de programación y supervisión en un caso de estudio.

Objetivos específicos

- Definir un método para el desarrollo de una propuesta de programación y supervisión utilizando las herramientas de *PlantPAx*.
- Implementar el método definido para evaluar los tiempos de desarrollo de una propuesta de programación y supervisión.
- Evaluar en el caso de estudio el desempeño de la implementación de las herramientas de *PlantPAx*.

Capítulo 1

Método para desarrollar una propuesta de programación y supervisión con las herramientas de PlantPAx

Para evaluar la incidencia que tiene el uso de las herramientas de *PlantPAx*, específicamente la *Biblioteca de Objetos de Proceso*, en el desempeño de una propuesta de programación y supervisión, es necesario definir un método basado en buenas prácticas de programación y diseño de interfaz hombre máquina (HMI) que contenga los conceptos necesarios para entender cómo funcionan, cómo fueron diseñadas y cómo utilizar las herramientas que ofrece *PlantPAx*.

En este capítulo se abarcan los conceptos concernientes a buenas prácticas de programación basadas en la experiencia de algunas empresas desarrolladoras de soluciones en automatización y en pautas del fabricante *Rockwell Automation*, el cual se reconoce como proveedor mundial líder en soluciones de automatización industrial. Para los conceptos sobre diseño de HMI de alto rendimiento se consideró el estándar *ANSI/ISA-101.01-2015, Human Machine Interfaces for Process Automation Systems* y el libro *The High Performance HMI Handbook* el cual contiene los principios de mejores prácticas para evaluar, diseñar e implementar adecuadamente interfaces hombre-máquina. Al final del capítulo se realiza una introducción al uso de los objetos de la *Biblioteca de Objetos de Proceso* de *PlantPAx*.

1.1. Conceptos de buenas prácticas para programación

1.1.1. Nomenclatura del proyecto

En el desarrollo de un proyecto de programación es necesario crear y nombrar tareas, programas, rutinas, *tags* entre otros elementos importantes de la aplicación, por esto se hace necesario definir un conjunto de reglas para la elección de la secuencia de caracteres utilizada para la identificación funcional de los elementos que componen la aplicación, esto servirá para reducir el esfuerzo necesario para leer y entender los componentes de la aplicación, además de mejorar la apariencia del programa y ordenarlo, esto evitará nombres excesivamente largos o abreviaturas que no sean relevantes.

A lo largo de un proyecto de *STUDIO 5000 Logix Designer* usted define nombres para los distintos elementos que componen el proyecto, como son el controlador, las direcciones de datos o *tags*, las rutinas, los módulos de entradas y salidas (E/S), entre otros.

Para nombrar los elementos descritos se recomienda seguir las siguientes reglas en el momento de definir un nombre [25, 26, 27]:

- Los nombres solo deben contener letras, números y caracteres de subrayado, la letra "ñ" no está permitida.
- Deben empezar con una letra o un carácter de subrayado.
- El nombre debe estar compuesto de una cantidad de caracteres menor o igual a 40.
- No utilizar caracteres de subrayado consecutivos.
- Recordar que no se distingue entre mayúsculas y minúsculas.
- El carácter inicial en mayúscula, la combinación de mayúsculas y minúsculas y la separación de palabras por medio de un carácter de subrayado facilitan la lectura.

La definición de un esquema para dar nombres a las variables asociadas a los equipos de un proceso facilita la legibilidad del mismo y la modificación del programa en futuras ocasiones. A cada variable a ser identificada se le asigna una etiqueta alfanumérica o como comúnmente se le llama *tag*.

Considere como recomendación la siguiente estructura para las *tags*:

- Todas las letras de la identificación funcional son mayúsculas.
- Las 3 primeras letras corresponden a la identificación funcional del instrumento definido bajo el estándar *ISA 5.1*, los módulos de control pueden ser válvulas, motores, bombas, sensores, interruptores y transmisores asociados a una variable de proceso como temperatura, flujo, peso, entre otras.
- Después de las primeras 3 letras agregar un guion bajo para agregar un identificador numérico.
- Los siguientes 3 números hacen referencia al número del lazo de control del proceso.
- Las descripciones asociadas a las *tags* deben contener información relevante expresada de manera breve, ya que este texto se visualiza en las líneas de código donde la *tag* es usada.

Considerando las reglas mencionadas se aconseja que el nombre del controlador esté relacionado al área o equipo más relevante al que se está ejerciendo control, para esto puede seguir la siguiente estructura:

PLC_ÁREA O EQUIPO_NÚMERO DEL PLC

Un ejemplo de lo anterior es:

CLX_100_001

El anterior ejemplo hace referencia a el controlador *ControlLogix 001 del área de producción 100*.

1.1.2. Definición de tareas y rutinas

En busca de lograr optimizar los recursos del controlador, una aplicación puede ser fragmentada en varios programas, cada uno de estos bajo el control de una tarea [28]. A continuación se describen consejos para estructurar un proyecto mediante el uso de tareas, programas y rutinas.

1.1.2.1. Tareas

Los controladores *Logix5000* permiten usar múltiples tareas para programar y priorizar sus programas con base en criterios específicos. Esta función multitareas asigna el tiempo de procesamiento del controlador entre las diferentes operaciones de su aplicación, por ello tenga en cuenta [25, 29, 30]:

- El controlador ejecuta solo una tarea a la vez.
- Una tarea puede interrumpir la ejecución de otra tarea y tomar el control.
- En una tarea es posible usar múltiples programas. Sin embargo, solo un programa se ejecuta a la vez.

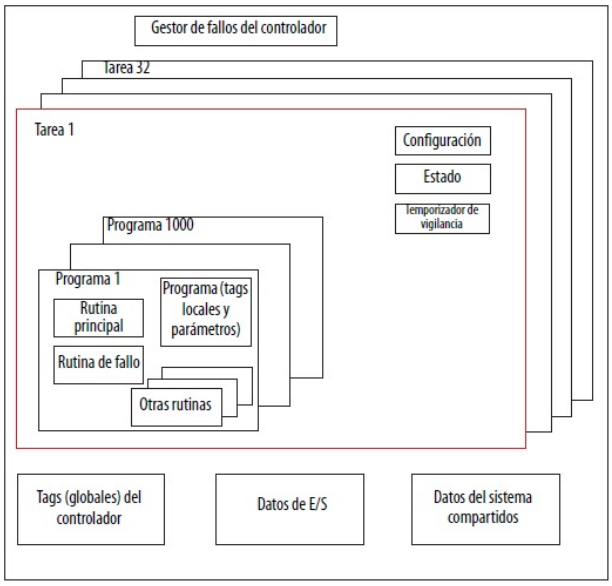


Figura 1.1: Tarea dentro de una aplicación de control.
Tomado de [29].

Las tareas proporcionan información de programación y priorización de un conjunto de uno o más programas, estas pueden ser configuradas como continuas, periódicas o por evento. La siguiente tabla explica los tipos de tareas que se pueden configurar.

Tipo de tarea	Ejecución de tarea	Descripción
Continua	Constante	La tarea continua se ejecuta en segundo plano. Todo tiempo de CPU no asignado a otras operaciones (como movimiento, comunicación y otras tareas) se utiliza para ejecutar los programas en la tarea continua. • La tarea continua se ejecuta constantemente. Una vez que la tarea continua realiza un escán completo, se reinicia inmediatamente. • Un proyecto no requiere una tarea continua. Si se usa solo pude haber una.
Periódica	• A un intervalo establecido, como cada 100 ms. • Varias veces en el escán de la otra lógica	Una tarea periódica realiza una función con un intervalo especifico. • Siempre que vence el tiempo de la tarea periódica, la tarea interrumpe las tareas de menor prioridad, se ejecuta una vez y después devuelve el control a donde estaba la tarea anterior. • Se puede configurar el periodo de tiempo de 0.1. . . 2,000,000.00 ms. El valor predeterminado es 10 ms. También depende del controlador y de la configuración.
Evento	Inmediatamente tras ocurrir un evento.	Una tarea de evento realiza una función solo cuando ocurre un (disparador de) evento especifico. El activador de la tarea de evento puede ser lo siguiente: • Cambio de estado de datos de entrada de modulo. • Un activador de tag consumido. • Una instrucción EVENT, • Un activador de eje. • Un activador de evento de movimiento.

Tabla 1.1: Tipos de tareas y frecuencia de ejecución.
Tomado de [30].

1.1.2.2. Prioridad de una tarea

Cada tarea en el controlador tiene un nivel de prioridad. El sistema operativo usa el nivel de prioridad para determinar qué tarea se debe ejecutar cuando se activan múltiples tareas. Una prioridad más alta interrumpe las tareas de prioridad más baja. La tarea continua tiene la prioridad más baja y una tarea o evento periódico la interrumpe [30, 31].

Las tareas periódicas y las de evento pueden configurarse para que se ejecuten desde la prioridad más baja de 15 hasta la prioridad más alta de 1, la cantidad de niveles de prioridad depende del controlador [30, 31].

Para asignar una prioridad a una tarea, considere lo siguiente [30]:

Acción deseada	Consejo	Notas
Esta tarea interrumpa otra tarea	Asigne un número de prioridad que sea menor que (prioridad más alta) el número de prioridad de la otra tarea	Una tarea de mayor prioridad interrumpe todas las tareas de menor prioridad. Una tarea de mayor prioridad puede interrumpir una tarea de menor prioridad varias veces.
Otra tarea para interrumpir esta tarea.	Asigne un número de prioridad que sea mayor que (prioridad más baja) el número de prioridad de la otra tarea.	Una tarea de mayor prioridad puede interrumpir una tarea de menor prioridad varias veces.
Esta tarea para compartir el tiempo del controlador con otra tarea.	Asigna el mismo número de prioridad a ambas tareas.	El controlador alterna entre cada tarea y ejecuta cada tarea durante 1 ms.

Tabla 1.2: Consideraciones para definir la prioridad de las tareas.
Tomado de [30].

1.1.2.3. Ajuste del Watchdog

Cada tarea contiene un *watchdog* (temporizador de vigilancia) que especifica cuánto tiempo puede ejecutarse una tarea antes de desencadenar una falla mayor [30].

Si el *watchdog* alcanza un valor preestablecido se produce una falla importante, de acuerdo a esto se puede detener la ejecución del programa y por consiguiente detener el proceso controlado.

Tenga presente los siguientes aspectos [30]:

- Un *watchdog* puede variar desde 1 ... 2,000,000 ms (2000 segundos). El valor predeterminado es 500 ms.
- El *watchdog* comienza a ejecutarse cuando se inicia la tarea y se detiene cuando se han ejecutado todos los programas dentro de la tarea.
- Si la tarea lleva más tiempo que el *watchdog*, se produce una falla importante (el tiempo incluye interrupciones por otras tareas).

Establezca el *watchdog* en un valor de 2 o 3 veces el período de la tarea, esto especifica cuánto tiempo puede ejecutarse una tarea antes de desencadenar una falla mayor.

Considere lo siguiente como recomendación para la programación de tareas:

- Cada tarea debe estar segmentada por medio de programas que faciliten la comprensión de la importancia de cada uno de ellos dentro del proceso.
- Nombre las tareas de acuerdo a un grupo importante de acciones que englobe el desempeño de la misma, ejemplo: equipos, secuencias, comunicaciones, verificaciones del controlador, entre otras.
- Si la tarea es periódica y el nombre de esta no es muy extenso, adicione el tiempo del *period* de cada tarea en el nombre de la misma, esto permite conocer rápidamente con qué frecuencia se ejecuta cada tarea.
- Configure la prioridad de la tarea de acuerdo a la importancia de la tarea dentro del proceso y considere las pautas del fabricante.

1.1.2.4. Rutinas

Una rutina es un conjunto de instrucciones lógicas en un solo lenguaje de programación. Las rutinas proporcionan el código ejecutable para el proyecto en un controlador. Una rutina es similar a un archivo de programa o a una subrutina en un procesador PLC o SLC [29].

Cada programa tiene una rutina principal. Esta es la primera rutina que se ejecuta cuando el controlador activa la tarea asociada y llama al programa asociado.

De manera opcional se puede especificar una rutina de fallo de programa. El controlador ejecuta esta rutina si encuentra un fallo en la ejecución de una instrucción dentro de cualquiera de las rutinas en el programa asociado [29]. Existen dos tipos de rutinas, rutina principal y subrutina [25, 29].

1. Rutina principal: en cada programa, el usuario asigna una rutina principal.

- Cuando se ejecuta el programa, se ejecuta automáticamente su rutina principal.
- Utilice la rutina principal para controlar la ejecución de las demás rutinas del programa.

2. Subrutina: Se le denomina a cualquier rutina que no sea la rutina principal o rutina de fallo.

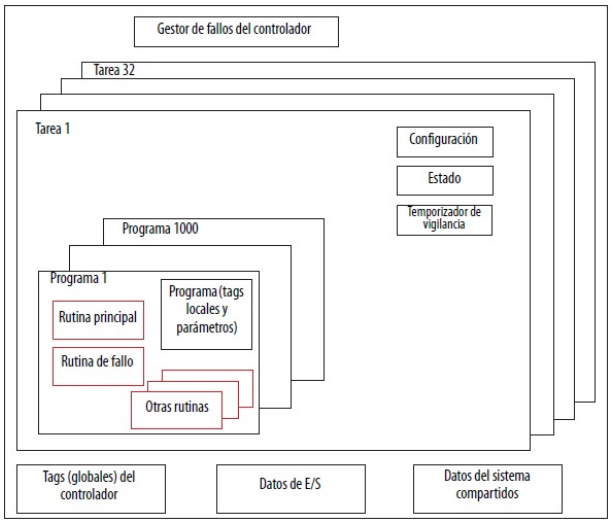


Figura 1.2: Rutinas en una aplicación de control.
Tomado de [29].

Para que el proyecto sea más fácil de desarrollar, probar y para facilitar la resolución de problemas, divídalo en rutinas (subrutinas). Considere lo siguiente [25]:

- Identifique cada sección física de la máquina o proceso.
- Asigne una rutina para cada una de esas secciones.

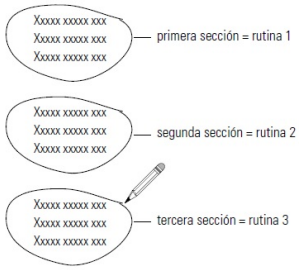


Figura 1.3: Descripción de la máquina o proceso.
Tomado de [25].

Para cada rutina, seleccione un lenguaje de programación [25]:

- Los controladores *Logix5000* le permiten usar los siguientes lenguajes: lógica de escalera, diagramas de bloques de funciones, diagrama de función secuencial, texto estructurado.
- Utilice cualquier combinación de lenguajes en el mismo proyecto.

En general, si una rutina representa:	**Entonces use este lenguaje:**
Ejecución continua o paralela de varias operaciones (que no tienen secuencia).	Lógica de escalera
Operaciones booleanas o basadas en bit.	
Operaciones lógicas complejas.	
Procesamiento de comunicación y mensajes.	
Enclavamiento de máquina.	
Operaciones que el personal de mantenimiento o servicio pueda necesitar interpretar para resolver problemas en la máquina o en el proceso.	
Control de variador y proceso continuo.	Diagrama de bloques de funciones (FBD)
Control de lazo.	
Cálculos en flujo de circuito.	
Administración de alto nivel de varias operaciones.	Diagrama de función secuencial (SFC)
Secuencias de operaciones repetitivas.	
Proceso de lote.	
Control de movimiento usando texto estructurado.	
Estado de operación de máquina.	
Operaciones matemáticas complejas.	Texto estructurado
Procesamiento especial de matriz o tabla de lazos.	
Manejo de cadenas ASCII o procesamiento de protocolo.	

Tabla 1.3: Consejos para definir el lenguaje de una rutina.
Tomado de [25].

Divida cada rutina en incrementos que tengan más significado, para esto analice la figura 1.4 [25]:

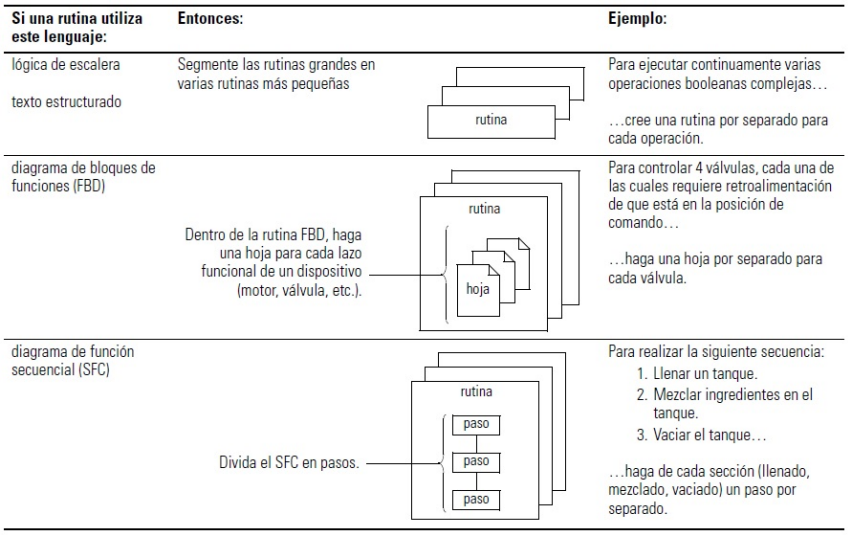
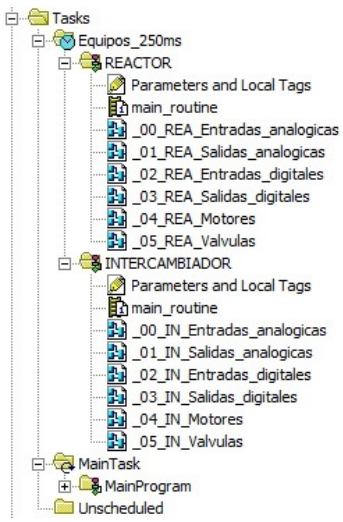


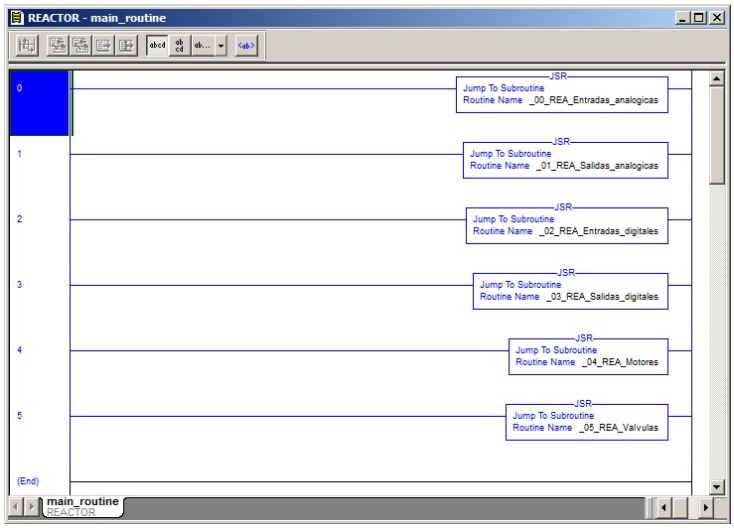
Figura 1.4: Consejos para dividir rutinas.
Tomado de [25].

Considere lo siguiente como recomendación para la programación de rutinas:

- Una tarea puede estar compuesta de varios programas, a su vez cada programa puede contener varias rutinas, pero solo una rutina principal, generalmente llamada *main_routine*. Esta rutina de cada programa debe reservarse para las acciones críticas de control del programa, utilice esta rutina solo para el llamado de otras rutinas, esto por medio del uso de instrucciones *JSR*, de ser posible mantenga esta rutina con el menor número de instrucciones posibles [32, 33]. La rutina principal por contar con un número limitado de instrucciones es generalmente programada usando *Ladder Logic*.
- Utilice rutinas para definir los modos típicos de operación y demás acciones importantes, por ejemplo: modo manual, modo automático, apagado, configuración, diagnóstico, entre otros.
- Utilice números al inicio del nombre de la rutina para dar un orden específico en el programa en el que se encuentra.
- Nombre las rutinas iniciando con las primeras letras del programa al que pertenecen.



(a) Definición de tareas, programas y rutinas.



(b) Lógica de la rutina principal (*main_ routine*) de la tarea *Equipos_ 250ms*.

Figura 1.5: Uso de buenas prácticas en la programación de PLC.
Fuente propia.

1.1.3. Interlocks o enclavamientos y permisivos

Los procesos requieren condiciones de seguridad para su normal funcionamiento y para evitar accidentes, por esto es necesario la implementación de *interlocks*. Los *interlocks* son condiciones que ejecutan acciones sobre dispositivos o secuencias dependiendo de condiciones de seguridad, proceso o permisivos, estos cambian el funcionamiento de la unidad con el fin de proporcionar un proceso sin fallas [34, 35].

Considere los siguientes términos:

- Interlocks: son condiciones que previenen, en una unidad o lógica, condiciones anómalas, estos inician o detienen el sistema para prevenir daños en equipos o accidentes con el personal.
- Permisivos: son condiciones que validan los requisitos mínimos que requiere un equipo o una unidad para trabajar [36].
- Interlocks o permisivos bypass: son condiciones necesarias para permitir un *bypass* en el proceso, generalmente usados para solucionar problemas de mantenimiento en equipos.

1.2. Conceptos de buenas prácticas para diseño de HMI de alto rendimiento

1.2.1. Consideraciones para el diseño de HMI

El diseño de HMI debe estar enfocado en las tareas importantes del proceso. Considere lo siguientes principios generales para el diseño del HMI [37]:

- El HMI es una herramienta efectiva para la seguridad y el control eficiente de los procesos.
- El HMI ayuda a la temprana detección, diagnóstico y respuesta adecuada de situaciones anormales.
- El HMI está estructurado para ayudar a los operadores a priorizar la respuesta a las principales o múltiples alteraciones simultáneas del sistema.
- Errores en las pantallas o elementos de las pantallas son inmediatamente evidentes para el operador.

El diseño de HMI deben tener en cuenta la *Ingeniería de Factores Humanos* o *HFE*.

Human Factors Engineering (HFE): "*es una disciplina científica relacionada con la comprensión de las interacciones entre los elementos humanos y otros de un sistema que aplica la teoría, los principios, los datos y los métodos de diseño para optimizar el bienestar humano y el rendimiento general del sistema*" [37].

Conceptos generales de *HFE* para incluir en el diseño son [37]:

- La manera en que el HMI funciona debe ser intuitiva para los usuarios.
- La información debe ser presentada en formas o formatos que son apropiados para los usuarios finales, ejemplo: si el operador toma lecturas de campo en metros, centímetros, pies, entre otros, entonces el HMI debe mostrar la información en este formato, en lugar de un porcentaje o rango.

- El HMI debe ser una herramienta de apoyo para poder interpretar los estados del sistema y del proceso. El HMI debe proveer al operador señales visuales y/o sonoras apropiadas a la situación, esto indica diferentes señales para el sistema cuando este funciona adecuadamente como cuando se presenta una situación de alerta.
- Distorsionar la percepción del operador sobre el proceso mediante la falta o ubicación errónea de información ha sido identificado como uno de los factores primarios en accidentes atribuidos a errores humanos.

1.2.2. Consideraciones sobre colores

Como regla general, el color debe ser usado para enfatizar información clave como alarmas y condiciones anormales o críticas. El color se debe usar de manera conservadora y coherente para denotar la información en todo el HMI [37].

Para un correcto diseño de HMI es necesario considerar las deficiencias en la percepción de colores, ya sea producto de una enfermedad u otra condición, de igual manera la visualización de la combinación de colores, el nivel de brillo y el contraste deben ser estudiados. La selección de colores debe ser distinguible para cada uno de los usuarios, por ello las pruebas de usabilidad son comunes en el diseño [37].

A pesar de que las deficiencias en la percepción de colores son una situación que se presenta por lo general en personas con enfermedades visuales, existen combinaciones exageradas de colores que su uso produce dificultades similares, además los usuarios no se encuentran libres de adquirir enfermedades visuales por avance de edad entre otros factores, por lo anterior es mejor evitar o reducir este tipo de riesgo [37].

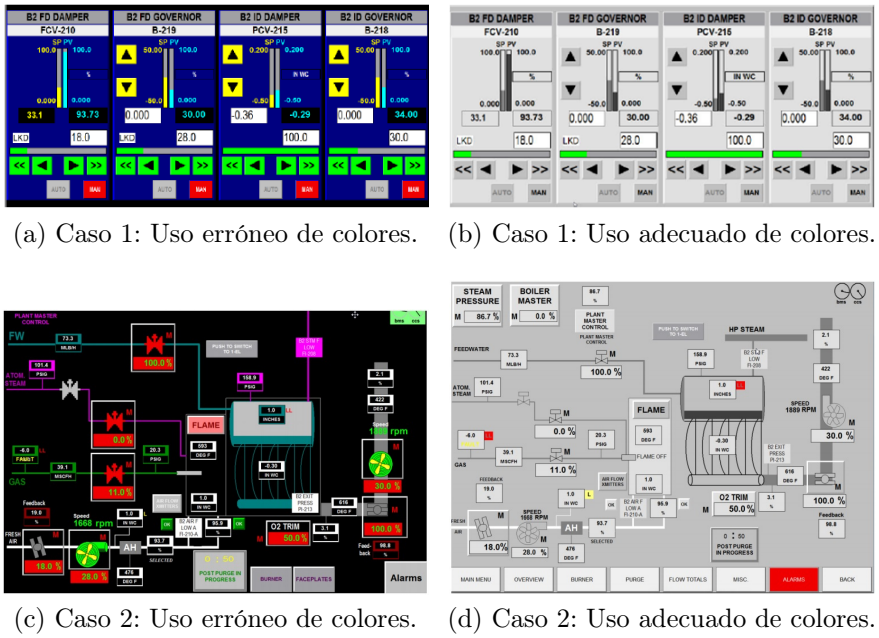
A continuación, se muestra una escala de colores desde la perspectiva de una persona con capacidad visual normal con respecto a la deficiencia visual de personas que no pueden identificar colores como rojo, verde, azul y amarillo [38].

Figura 1.6: Deficiencia de color.
Tomado de [38].

De acuerdo a *ANSI/ISA-18.2-2009* (filosofía de instalación de alarmas), los colores para presentar alarmas deben ser reservados y no usados para cualquier otro propósito, esto con la finalidad de afianzar el significado de estas para que el operador pueda responder rápidamente. No se debe confiar en el color como único indicador de una condición importante, la codificación de colores debe ser redundante a otros medios para presentar la información, incluidos la forma, el texto, el brillo, el tamaño y la textura [37].

1.2.2.1. Uso de color

Una correcta selección de colores para las pantallas del HMI es necesaria para brindar claridad a los operarios sobre el proceso, además esto contribuye con el fácil manejo de la aplicación, por otra parte, la selección errónea puede generar problemas en el desarrollo de las tareas del operario ya que puede evitar una detección temprana y precisa de situaciones anormales en el proceso [39].



(a) Caso 1: Uso erróneo de colores. (b) Caso 1: Uso adecuado de colores.

(c) Caso 2: Uso erróneo de colores. (d) Caso 2: Uso adecuado de colores.

Figura 1.7: Comparación del uso de colores en un HMI.
Tomado de [40].

A continuación se presenta una muestra de algunos colores y los usos en un HMI de alto rendimiento:

Color	RGB	Muestra	Usos definidos
Gris neutro 1	221, 221, 221		Fondo de pantalla global.
Gris claro Blanco	243, 243, 243 255, 255, 255		Indicador de equipos encendidos o en operación normal.
Gris oscuro 1	136, 136, 136		Indicador de equipos detenidos o apagados.
Gris oscuro 2	128, 128, 128		Alarmas inactivas en indicador analógico.
Gris oscuro 3 Negro	74,74,74 0, 0, 0		Texto, líneas de proceso y contorno de elementos.
Azul oscuro	0, 0, 172		Caracteres numéricos del proceso, modo de controlador, salida del controlador (O), variable de proceso (P) y trazo de tendencia variable de proceso.
Verde oscuro	0, 128, 0		Setpoint del controlador (S) y su tendencia.
Azul claro	187, 224, 227		Rango o condiciones deseadas de operación.
Azul cielo	72, 185, 253		Nivel de recipiente, otras tendencias.
Café	204, 102, 0		Indicador de posición de retroalimentación, otras tendencias.
Rojo	255, 0, 0		Alarma de prioridad uno.
Amarillo	255, 255, 0		Alarma de prioridad dos.
Naranja	255, 102, 0		Alarma de prioridad tres.
Magenta	255, 0, 255		Alarma de prioridad cuatro para diagnósticos.
Magenta oscuro	204, 0, 102		Otras tendencias.

Figura 1.8: Paleta de colores para un HMI de alto rendimiento.
Fuente propia.

Es una práctica común y efectiva usar el color como un método para diferenciar objetos y sus diversos estados en pantallas. El color no debe ser utilizado como el único método de diferenciación, además este se debe utilizar para dirigir la atención y añadir significado a la pantalla [37].

A lo largo de los años se ha acostumbrado a usar diferentes colores para representar varios estados de los elementos del proceso, considerando que el color es una propiedad que capta la atención del humano debe ser usado solo para situaciones anormales del proceso, no para estados de parada y de funcionamiento [37].

1.2.2.2. Colores para el fondo

El fondo del HMI debe tener un color opaco, generalmente gris claro [41]. Esto porque ofrece poco contraste con los demás elementos de la pantalla, lo que reduce la fatiga de los operadores. Es necesario estudiar las condiciones normales de iluminación del sitio donde estará ubicada la pantalla, esto para obtener un buen efecto con el color de fondo seleccionado [42].

Para seleccionar la tonalidad adecuada se deben realizar pruebas para validar su efecto en el sitio donde será utilizada la aplicación, para esto considere la figura 1.9 la cual muestra 16 niveles típicos dentro de la escala de gris, de la cual el gris 3 (RGB 221, 221, 221) y gris 4 (RGB 192, 192, 192) son tonalidades típicas utilizadas para los fondos de las pantallas [42].

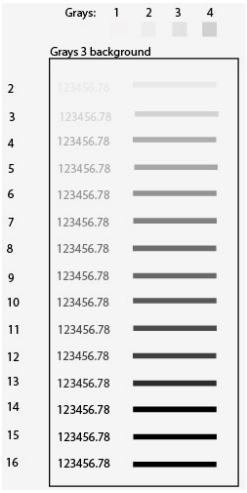


Figura 1.9: Contraste de 16 niveles típicos dentro de la escala de gris.
Tomado de [39].

1.2.3. Dinámica visual

Debido a los límites de la percepción y la cognición solo se puede usar un número limitado de colores de manera efectiva en las pantallas. Como alternativa, técnicas como el movimiento, parpadeo, brillo (destello) y condiciones de visibilidad pueden ser usadas para llamar la atención del operador hacia información específica. El uso excesivo o persistente de efectos de movimiento, parpadeo o destello pueden ser una distracción para el operador, por lo anterior considere los siguientes conceptos de diseño que deben ser empleados para administrar las animaciones [37]:

- El parpadeo debe ser reservado para elementos los cuales son destinados a aparecer y desaparecer, tales como símbolos o bordes para la identificación de alarmas no reconocidas.
- El texto y los números en sí mismos no deben moverse ni parpadear, ya que el cambio de posición o ángulo, o la aparición y desaparición alternas de los números dificultan su lectura.

- Ninguna parte de la pantalla debe parpadear o destellar a menos que sea una acción necesaria para el operador. El uso de animaciones debe ser reservada para resaltar situaciones anormales o situaciones que requieren atención del operador, ejemplo: una indicación de alarma.
- Debe ponerse a disposición del operador un medio para detener el parpadeo o el destello.

1.2.4. Presentación de texto y números

1.2.4.1. Presentación de texto

Considere lo siguiente [37]:

- El texto debe estar justificado con la dirección de lectura normal, se ha demostrado que entre el uso de letras mayúsculas y minúsculas para los títulos, el uso de letras mayúsculas facilita la legibilidad del texto por lo tanto estas se deben usar para títulos.
- Si en el display se requiere utilizar un párrafo, el ancho de texto debe ser lo suficientemente amplio para una lectura rápida, se debe tener precaución con el uso de mayúsculas en párrafos, estas deben ser utilizadas siempre y cuando se desee hacer énfasis en alguna palabra o en una frase corta ya que tener párrafos completos en mayúsculas convierte la legibilidad del texto en una tarea lenta.
- Deben evitarse las abreviaturas y acrónimos a menos que sean parte del lenguaje normal del operario o debe tratar de usar abreviaciones estandarizadas de algún glosario o lista de términos para que todos los usuarios puedan entenderlo. Se debe evitar abreviar a menos que esto sea para una mejora significativa de espacio, si este es el caso, es obligatorio garantizar que siga siendo claro para el operador y otros usuarios.
- No se debe utilizar el subrayado para hacer énfasis en el texto, el subrayado está reservado solo para hipervínculos. Si todo el texto subrayado es un hipervínculo no es necesario emplear un color por separado para los enlaces.

- Debido a que las mayúsculas y el subrayado están prohibidos para resaltar o hacer énfasis en algo, los métodos más comunes para el énfasis del texto son el tamaño, ubicación del texto, el color y los efectos del texto como negrita o cursiva.

- La visualización del texto debe verse orientada horizontalmente, la fuente seleccionada debe ser perceptible por el operador teniendo una definición clara de los caracteres, unos de los problemas más comunes están relacionados con la presentación de 1, i y L estos caracteres presentan similitud en fuentes de texto logrando no ser diferenciados.

1.2.4.2. Presentación de números

En el diseño del HMI es muy común utilizar entradas de números para diferentes configuraciones del proceso, el uso de un rango numérico excesivo requiere la utilización de notación científica para mostrar valores grandes o pequeños. Se deben suprimir los ceros principales en números enteros, pero deben ser mostrados para números entre -1 y 1 [37].

Las unidades de ingeniería deben estar disponibles como referencia para el operador, estas se pueden visualizar desde una función disponible en el faceplate como mostrar u ocultar, como información en alguna barra de herramientas o mostrarse directamente en la pantalla [37].

La presentación debe seguir generalmente la resolución de formato decimal apropiada requerida por los usuarios para realizar sus tareas. La resolución debe estar respaldada por la exactitud de la medición del proceso; las convenciones no deben exigir que las pantallas presenten más o menos dígitos significativos de los requeridos, si se usa un re escalamiento automático del formato decimal se debe tener cuidado de suprimir los cambios rápidos en el formato decimal para evitar el desplazamiento repetitivo del punto decimal [37].

Engineering range	**Decimal formatting**
100 - 9999.9	XXXX
10 - 99.99	XX.X
1 - 9.999	X.XX
0 - 0.9999	X.XXX

Tabla 1.4: Ejemplo de formato decimal numérico.
Tomado de [37].

Los números negativos deben representarse de manera coherente con la población de usuarios, generalmente entre paréntesis o con un "-" anterior al número [37].

Los números deben ser referenciados en rangos normales y en otros rangos operativos. Cuando se referencia un suceso normal y crítico, este se representa mediante el uso de un color reservado, curvas y líneas de referencia, así mismo, los valores críticos pueden ser referenciados de manera similar mediante una clara indicación de infracción del rango crítico [37].

Es recomendable que los números decimales sean utilizados con puntos y no comas. A diferencia de los textos, los formatos de los valores numéricos deben ser en negrita y con color azul oscuro, esto para lograr un correcto contraste con el fondo gris de la pantalla [37].

1.2.5. Diseño de botones

El diseño de botones debe emplear los siguientes conceptos [37]:

- El texto de la etiqueta del botón debe ser claro para los usuarios y estar claramente asociado con el botón, ya sea en él o cerca de él.
- Si el texto de la etiqueta es dinámico con las condiciones del proceso, la etiqueta debe representar la acción del botón utilizando un verbo simple o una voz activa como por ejemplo, detener, bombear, etc.
- Los botones deben ser lo suficientemente grandes como para permitir a los usuarios seleccionarlos con rapidez y precisión.
- Los botones que interactúan directamente con el proceso deben ser visualmente distintos de los botones que proporcionan vínculos de navegación o inician aplicaciones.
- Cualquier botón que no esté disponible actualmente debido al estado del proceso o el nivel de acceso del usuario que inicio sesión en el HMI debe permanecer visible. Sin embargo, cualquier botón que no esté disponible debe indicar su inalcanzabilidad temporal con una codificación visual consistente. Si se utilizan múltiples botones y la interacción requiere una secuencia determinada, los botones deben estar disponibles

sólo cuando sea apropiado o se requiera retroalimentación y se emplee un rechazo de orden incorrecto.

- Se debe proporcionar confirmación de retroalimentación para la ejecución del botón.

1.2.6. Faceplates y pop-ups

Pop-up: Un *pop-up* es también conocido como ventana emergente, es una nueva ventana que aparece de forma repentina la cual se superpone al primer plano de la pantalla para mostrar el contenido posiblemente ocultando parte o todas las demás pantallas [37].

Faceplates: Es una pantalla, parte de la pantalla o ventana emergente utilizada para monitorear u operar directamente un solo lazo de control, dispositivo, secuencia u otra entidad [37].

Considere lo siguiente para el diseño de *pop-ups* y *faceplates* [37]:

- Se debe configurar un período de espera después del cual la ventana emergente debe cerrarse si no existe ninguna interacción del usuario. El operador debe tener el control de cuando se cierran los *pop-ups* de uso especial, incluyendo la capacidad de anular el periodo de tiempo de espera configurado.
- Los *pop-ups* deben diseñarse de manera que no cubra u oculte partes importantes de la pantalla HMI, si las ventanas no se encuentran diseñadas para garantizar que no oculten la pantalla, el operador debe ser capaz de mover el pop-up.
- Si se utilizan múltiples *pop-ups*, se debe tener cuidado para asegurar que el operador es consciente del *faceplate* que se está atendiendo y la manipulación de datos que se haga desde el teclado en el *faceplate* sea el correcto.
- Todos los métodos de presentación e interacción con los *pop-ups* deben ser coherentes con el resto de HMI.
- Los *pop-ups* o *faceplates*, tendrán su fondo del mismo color de fondo que las demás pantallas. Su tamaño variará dependiendo de su función o de los elementos a los que pertenecen y se los diseñará para dar información más detallada o para el control de características sobre algún elemento o subproceso.

1.2.7. Representación de los elementos de un proceso

A continuación se nombran algunas características de un HMI de alto rendimiento en la representación de un proceso [42].

Los elementos del HMI representan fielmente el proceso y toman en consideración las pautas sobre colores. Se recomienda el uso de 2D para representar los elementos porque el uso de 3D puede afectar las velocidades de desempeño de la aplicación y requerir un hardware más robusto para soportar la aplicación. Los elementos representados en 2D evitan que los operadores enfoquen su atención en detalles irrelevantes de diseño del HMI evitando distracciones en el proceso [42].

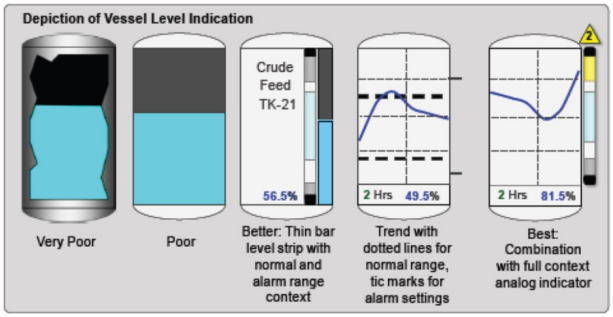
Se utiliza la escala de grises para representar tuberías, líneas, tanques y demás equipos del proceso representado deben ser de color gris oscuro, puede considerar los niveles desde 11 hasta 16 de la figura 1.9, esto permitirá destacar adecuadamente los elementos mencionados. Si se hace necesario resaltar algún elemento de la pantalla considere realizar modificaciones en el grosor o tonalidad del elemento, pero no realice cambios a otros colores. El color no debe ser usado para indicar el tipo de material o fluido (ejemplo, color azul para agua), esto para evitar distracciones del operario y mantener la seguridad del proceso. Considere el uso de otros colores y tonalidades para representar situaciones anormales de proceso o alarmas, además asegúrese de reservar los colores para uso específico (ejemplo, el uso del color amarillo solo para alarmas) [42, 43].

Considere lo siguiente para la representación de líneas de proceso [42, 43]:

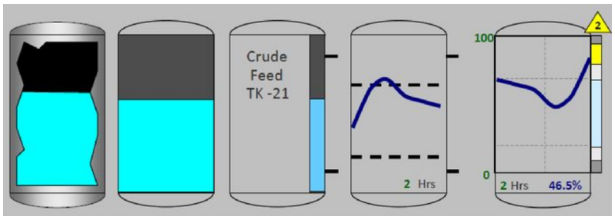
- Deben ser de color gris oscuro.
- Deben indicar la dirección de flujo del proceso.
- Para diferenciar varias líneas de proceso, realice variaciones en el grosor de estas.
- Para representar líneas de fluidos considere direccionar hacia arriba para vapores y hacia abajo para líquidos.

Considere lo siguiente para representar tanques, reactores y otros equipos de almacenamiento [42, 43]:

- Los equipos deben ser mostrados en 2 dimensiones (2D).
- El borde de la figura de los equipos debe ser de tonalidades más oscuras de gris o de color negro para contrastar con el fondo.
- El interior de los equipos puede ser del mismo color que el fondo o una tonalidad de gris que contraste ligeramente.
- Los equipos no deben tener animaciones relacionadas a otras condiciones internas del equipo, solo los cambios en el nivel.
- Se deben representar los límites de llenado y vaciado de los equipos.
- Se debe representar el nivel actual del tanque con un color un poco más brillante y el nivel restante con un color más oscuro.
- No se debe representar el nivel tanque con una franja de igual ancho a la del tanque, en cambio el espacio restante puede ser utilizado para indicar tendencias, porcentaje de llenado o *tag*.
- El interior del tanque puede mostrar las tendencias de un periodo determinado de tiempo, generalmente periodos cortos, esto según los requerimientos de los usuarios.



(a) Ejemplo 1.



(b) Ejemplo 2.

Figura 1.10: Representaciones de nivel para tanques.
Tomado de [39].

Considere lo siguiente para la representación de las válvulas de control y sus acciones [42, 43]:

- Utilice el color blanco en el cuerpo de la válvula para representar apertura y color gris oscuro para cerrado.
- Debajo del cuerpo de la válvula o a un lado, muestre el porcentaje de apertura.
- Si la válvula cuenta con retroalimentación analógica, la cabeza de la válvula se llenará con color blanco, además debajo del cuerpo de esta deben aparecer los dos valores, el de apertura de la válvula y el de retroalimentación. Para interpretar más fácilmente, estos valores deben tener colores diferentes.
- Para las válvulas solenoides u "on-off" utilice las mismas pautas de colores para indicar los estados y si cuenta con retroalimentación analógica. Debajo del cuerpo de la válvula o a un lado, mediante un texto, indique el estado, ejemplo, "abierta" o "cerrada".
- Evite ubicar barras que representen el porcentaje de apertura, en lugar de esto, si es estrictamente necesario, utilice un slider.
- Las válvulas de 3 posiciones siguen el mismo principio de colores (blanco apertura, gris oscuro cerrado) para representar la ruta de apertura de flujo.

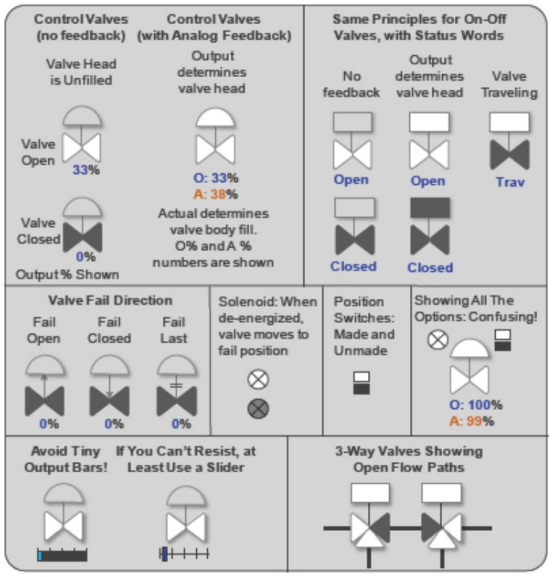
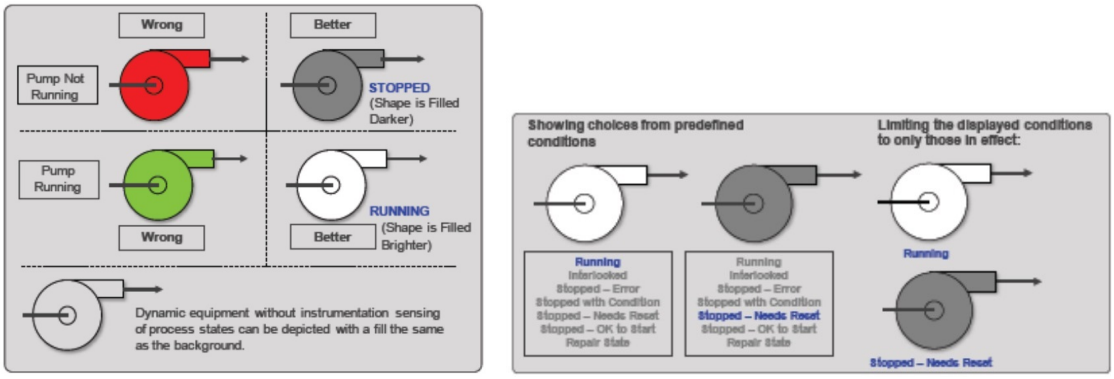


Figura 1.11: Representaciones de válvulas.
Tomado de [39].

Considere lo siguiente para la representación equipos dinámicos como motores y sus acciones [44]:

- Evite los colores brillantes como verde y rojo para representar estados como encendido y apagado.
- Utilice el color blanco y el color gris oscuro para denotar si el equipo está encendido o apagado, respectivamente.
- Si el equipo cuenta con un sensor que permita conocer el estado, debajo del elemento o a un lado, represente mediante un texto el estado del equipo, ejemplo, "encendido" o "apagado".
- Si el equipo no cuenta con un sensor que permita conocer el estado, el color del equipo puede ser el mismo que el del fondo de la pantalla.
- Siguiendo las pautas mencionadas, es posible también ubicar cerca del elemento, un recuadro con los estados posibles a los que puede llegar el equipo, el estado actual estará resaltado por medio de un color diferente, esto es mostrado en la figura 1.12b.



(a) Ejemplo 1. (b) Ejemplo 2.

Figura 1.12: Representaciones de equipos dinámicos.
Tomado de [39].

El controlador debe ser presentado e ideado con una representación física en el HMI para mejorar el modelo mental del proceso para el usuario. El controlador se asocia con los instrumentos sobre los que tiene control como válvulas o motores, no se debe mostrar demasiados parámetros pertenecientes al controlador, una representación efectiva de un

controlador en la pantalla es en donde los parámetros como el *set point*, salida del controlador y variable del proceso, son representadas. Los controladores trabajan con diferentes lazos de control al tiempo por lo tanto se debe representar cada lazo de control de manera separada brindando información acerca de su estado. Tenga en cuenta que [39]:

- Los datos mostrados en la representación del controlador son:
  1. Variable del proceso: se representa con la letra PV o P.
  2. Set point: se represa con la letra SP o S.
  3. Salida del controlador: se representa con la letra OP u O.
  4. Modo de control: los modos de control son automático (AUTO), manual (MAN) y cascada (CAS) u otros modos dependiendo del proceso y sistema de control.
- La representación del controlador consiste en un cuadrado simple que permita mostrar los 4 parámetros descritos anteriormente con sus respectivas unidades de ingeniería, el titulo puede ir con el nombre de la variable de proceso.
- Con el objetivo de diferenciar los parámetros, la variable de proceso, la salida del controlador y el modo de control tendrán un color azul oscuro y el *set point* un color verde oscuro.
- El controlador cuenta con un *faceplate* el cual permite la modificación de los 4 parámetros descritos.

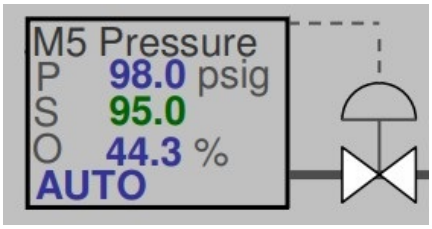


Figura 1.13: Representación de un controlador de proceso.
Tomado de [39].

1.2.8. Representación de alarmas

De acuerdo con las consideraciones para un HMI de alto rendimiento y a la norma *ANSI/ISA-101.01-2015*, se debe tratar de cumplir que el uso de colores tendrá como objetivo enfatizar información importante o diferenciar factores importantes como alarmas

o situaciones anormales. Las alarmas deben codificarse según la prioridad de la alarma teniendo en cuenta color, forma y texto. Los colores de la alarma deben ser reservados, no deben utilizarse para indicar en el HMI funciones que no estén relacionados con alarmas [37, 39].

Considere lo siguiente para la representación de alarmas [39]:

- Cuando un valor u objeto activa una alarma, el indicador de alarma debe aparecer a su lado.
- El método de parpadeo es importante para distinguir las alarmas que todavía no han sido reconocidas, el indicador parpadea mientras la alarma no ha sido reconocida y deja de parpadear después del reconocimiento, pero permanece visible mientras la condición de alarma esté activada. Las personas no detectan fácilmente los cambios de color en la visión periférica, pero los movimientos como los parpadeos sí.
- Las alarmas se deben encontrar acompañadas de formas geométricas y números. Las alertas no deben presentarse únicamente con el uso de colores. Las formas geométricas y números permiten que la alarma sea identificable de una manera más clara, rápida y eficiente.
- Si se activan varias alarmas en un momento determinado, la alarma con la más alta prioridad debe ser mostrada.
- La prioridad de la alarma es representada de manera redundante mediante la forma del indicador, el color y el número de prioridad.

El código de colores es utilizado para establecer las prioridades e identificar las situaciones anormales de manera eficiente, se recomienda implementar en un sistema 3 tipos de prioridad, existe un cuarto tipo alarma, este es de diagnóstico y se encarga de indicar un funcionamiento incorrecto de determinado instrumento que no puede ser solucionado por el operador y que por lo tanto requiere una acción de mantenimiento. La inclusión de formas geométricas y colores facilitan la identificación de las alarmas, la forma geométrica es asignada según la prioridad de la alarma. Las prioridades, colores y formas utilizadas para las alarmas son las siguientes [39]:

- Prioridad 1: Evento anormal, prioridad alta, representada con cuadrado o rectángulo de color rojo.

- Prioridad 2: Evento anormal, prioridad intermedia, representada con un triángulo de color amarillo.
- Prioridad 3: Evento anormal, prioridad media, representada con un triángulo invertido de color naranja.
- Prioridad 4: Reservado para diagnóstico, representada con un rombo de color violeta.

En la figura 1.14 se muestran 3 de los métodos más comunes de indicación de alarma que no consideran las reglas básica del uso del color, se presentan problemas en la combinación de colores, como por ejemplo, el rojo de la alarma y el azul de los números, el parpadeo de la alarma es otro problema ya que en estas presentaciones se ocultaría el valor de la alarma o no se permitiría visualizarlo de manera correcta, además de los problemas ya mencionados, la prioridad de las alarmas solo se muestra en colores y no es representada de manera redundante, como por ejemplo, con el uso de figuras geométricas y un número de prioridad sobre en ellas. Es importante resaltar que la redundancia ayuda a contrarrestar deficiencias visuales [39].

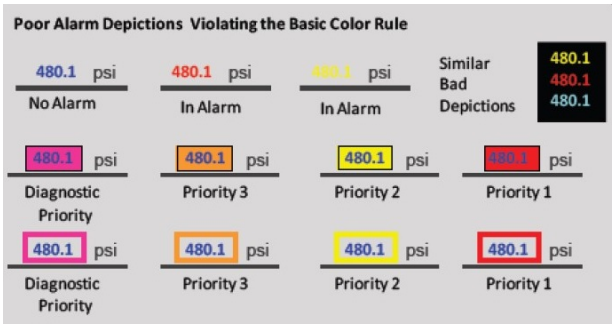


Figura 1.14: Representación inadecuada de alarmas.
Tomado de [39].

La figura 1.15 muestra el método recomendado para un HMI de alto rendimiento, este método se caracteriza por mostrar el valor de la alarma con una adecuada visualización. Cuando se presenta un evento anormal, el parpadeo generado en las figuras geométricas que se encuentran ubicadas cerca al valor no afecta la visualización de los valores mostrados, este método no posee el problema de combinación de colores. Las alertas son redundantes mediante la utilización de colores, formas y números, permitiendo ser reconocidas fácilmente [39].

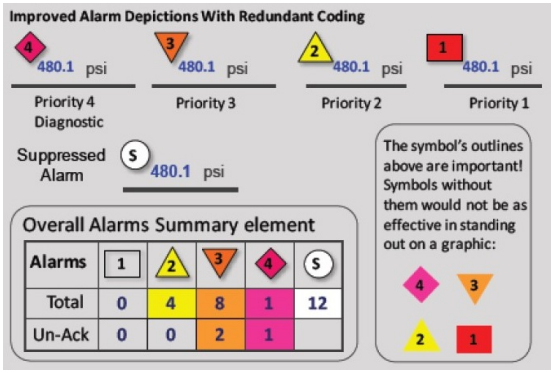


Figura 1.15: Representación adecuada de alarmas para HMI de alto rendimiento. Tomado de [39].

1.2.9. Tendencias e indicadores

Una de las diferencias más importantes de los gráficos de alto desempeño es el principio donde los valores de operación deben ser mostrados en un contexto de información y no en forma de datos sin procesar dispersos por la pantalla [39, 45].

1.2.9.1. Tendencias

Mostrar los datos del proceso de forma adecuada en los HMI es de vital importancia, esto se logra mostrando únicamente los datos que brinden información relevante, lo anterior con el fin de no saturar a los usuarios con datos e información innecesaria del proceso [44].

La representación de datos e información del proceso se realiza mediante el uso de tendencias, estas permiten mostrar datos en tiempo real o históricos en varios formatos de gráficos con respecto al tiempo. Considere lo siguiente para implementar tendencias en un HMI de alto rendimiento [39]:

- La visualización de múltiples gráficas debe ser aplicado coherentemente en donde se logre diferencia las distintas variables graficadas.
- Las tendencias deben tener una base de tiempo predeterminado conveniente de acuerdo a las condiciones que exija cada proceso.

- Las tendencias deben incorporar elementos que representen los rangos normales y anormales para el valor de tendencia para que el usuario tenga conciencia en qué estado se encuentra la variable y así realizar un control y monitoreo más efectivo. El usuario debe saber en qué estado estuvo el proceso en el pasado, como se encuentra en el presente y a donde se dirige el proceso en el futuro.
- Un gráfico apropiado de una tendencia de controlador deberá mostrar el valor actual de la variable de proceso, *set point* y la salida del controlador.

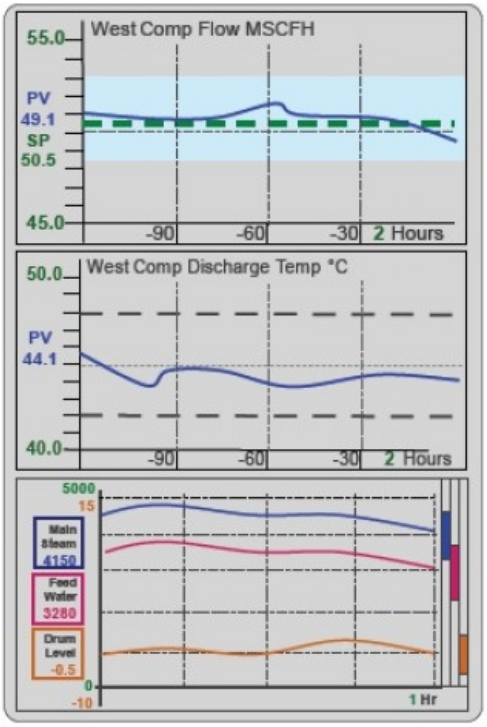


Figura 1.16: Buenas prácticas para representación de tendencias.
Tomado de [39].

1.2.9.2. Indicadores

Los datos por sí solos no siempre brindan formación, por lo tanto, por medio de un ejemplo se dará a entender la diferencia entre datos e información, en la siguiente tabla 1.5 se muestran las variables de proceso de un tanque. Como se observa a continuación, la información presentada no permite obtener conclusiones sobre el estado de las variables de proceso [39].

Tanque reactor	
Variables de proceso	**Valores**
Temperatura	50
Presión	85
Nivel	150

Tabla 1.5: Valores de las variables de proceso.
Fuente propia.

Con la tabla anterior no es posible determinar a simple vista si el proceso se encuentra operando en forma correcta, por lo tanto, será necesario tener datos adicionales como rangos y unidades de ingeniería, esto con el fin de verificar los valores dentro de unos límites establecidos.

Tanque reactor			
Variables de proceso	**Valores**	**Unidades**	**Rango**
Temperatura	50	°C	(0-100)
Presión	85	bar	(120-150)
Nivel	150	cm	(0 -100)

Tabla 1.6: Valores de las variables de proceso con más detalles.
Fuente propia.

Al agregar datos en la tabla como las unidades de ingeniería y el rango, es posible tener una idea del estado de cada variable, si es normal o anormal, partiendo del dato obtenido y su respectiva posición dentro de un rango establecido. A continuación se presenta una tabla mejorada donde se valida en menos tiempo el estado de cada variable del proceso sin la necesidad de interpretar los valores y los rangos en que se encuentra cada variable, esto debido a que es suficiente con observar los indicadores y los límites que se presentan en estos.

Tanque reactor				
Variables de proceso	**Valores**	**Unidades**	**Rango**	**Indicadores**
Temperatura	50	°C	(0-100)	
Presión	85	bar	(120-150)	
Nivel	150	cm	(0 -100)	

Tabla 1.7: Valores de las variables de proceso con más detalles e indicadores gráficos.
Fuente propia.

En el anterior ejercicio se observa que la temperatura se encuentra dentro del rango establecido, la presión está por debajo del rango y el nivel sobrepasa los límites establecidos. Los indicadores gráficos permiten verificar y entender el estado de las variables de proceso

en el menor tiempo posible, en el anterior ejemplo los números no serían necesarios siempre y cuando los indicadores se encuentren en los rangos adecuados. La información representa datos que en contexto son útiles, esto quiere decir que se deben mostrar valores que sirvan al operador para conocer cómo se encuentra el estado del proceso de forma rápida, sin la necesidad de análisis profundos [39].

En la mayoría de las interfaces se muestran grandes volúmenes de datos con una pobre presentación que no ayudan al usuario a tener un control y monitoreo eficaz de la situación, ya que los números por sí solos no proveen la posibilidad de conocer el estado del proceso a simple vista. Los operadores deberían tener un alto trabajo mental para memorizar los rangos de operación normales de los diferentes procesos dificultando su trabajo y generando posibles riesgos. Por el contrario, si los datos o valores son acompañados por información como rangos de operación normal del proceso, el trabajo de los operadores será más fácil, eficaz y completo, aportando así también a la rápida detección de situaciones anormales [38].

Una mejora en la representación gráfica del ejemplo planteado anteriormente sería utilizar indicadores que incluyan colores codificados para zonas seguras, zonas de prevención y zonas de valores críticos que denoten una emergencia o alarma, tanto los valores de la variable del proceso como los valores de sus límites por alarmas también son representados en un indicador analógico. Este indicador tiene la propiedad de mostrar la posición actual e indica los rangos de las alarmas en un color gris. De esta forma se puede identificar fácilmente si el valor está cerca de activar una alarma o está en un rango aceptable. Si se llega activar una alarma en estos rangos, el color se tornará de gris a amarillo o rojo de acuerdo al tipo de alarma, dando la posibilidad de identificar rápidamente el problema [44].

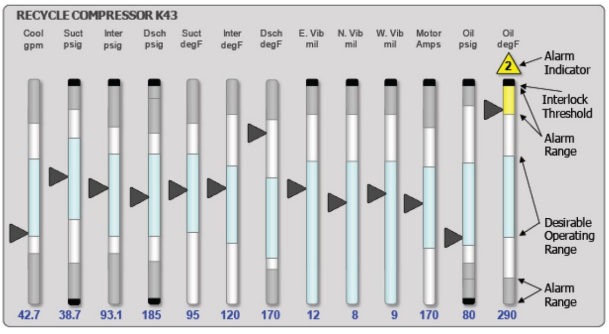


Figura 1.17: Representación de indicadores analógicos de la información.
Tomado de [39].

La recomendación es que siempre que sea posible se debe utilizar gráficos analógicos como herramienta de interpretación rápida de las variables de procesos, el cerebro interpreta una pantalla analógica más rápido ya que puede ver fácilmente donde está el valor y lo que significa. Los gráficos analógicos deben contar con un puntero que señale el rango en el que se encuentra, esto permite conocer claramente los limites superior e inferior además de proporcionar un indicador claro del rango de trabajo normal [45]. Por ejemplo, el tacómetro que utilizan los automóviles indica las revoluciones por minuto, si se observa con un indicador analógico se tiene la idea de cuánto se está esforzando el motor y por ende si éste sobrepasa un límite alto, mientras que con una representación digital que muestra solo el valor de las revoluciones, es imposible conocer en qué rango se está trabajando [38].

Figura 1.18: Comparación de indicador analógico y digital.
Tomado de [38].

1.2.10. Cambio de idioma

La función de cambio de idioma o "*Language switching*" permite a los operadores visualizar las cadenas de textos del HMI en hasta 40 idiomas diferentes. Esta función puede ser activada en el momento que la aplicación está siendo ejecutada [46, 47].

La opción de cambio de idioma permite exportar las cadenas de textos desarrolladas en la aplicación para ser traducidas en diferentes idiomas y posteriormente ser importadas. Esto permite re utilizar la aplicación en diferentes países, también ofrece la posibilidad de importar componentes de aplicaciones desarrolladas en otros idiomas a una aplicación con múltiples idiomas [46, 47].

A continuación se describen los tipos de cadenas de texto que pueden ser mostrados en diferentes idiomas cuando la aplicación se está ejecutando [46, 47]:

- Títulos, subtítulos, texto de información sobre herramientas, variables integradas de

fecha y hora, variables integradas numéricas.

- Mensajes locales.
- Texto de objetos.
- Cadenas de texto definidas para alarmas y otras funciones de *FactoryTalk View*.

Para programar una aplicación que soporte múltiples idiomas considere lo siguiente [46, 47]:

- La manera en que los operadores cambiarán de idioma en el momento que la aplicación se esté ejecutando. Para esto considere la creación de botones que al ser presionados cambiarán el idioma del HMI a cada uno de los idiomas soportados.
- Si los operadores son los encargados de cambiar el idioma de la aplicación, asegúrese de que estos tengan privilegios de seguridad para acceder a las pantallas que cuentan con los botones de cambio de idioma.
- Se recomienda el uso de banderas referentes a los idiomas soportados para cada botón de cambio de idioma, esto evita confusiones en el cambio del título que identifica el botón.

(a) Botones de cambio de idioma.

(b) Botones después del cambio a inglés.

Figura 1.19: Representación adecuada de botones de cambio de idioma.
Fuente propia.

1.3. Optimización de código reutilizable mediante el uso de instrucciones Add-On

1.3.1. Conceptos básicos

1.3.1.1. Instrucciones Add-On

A medida que se desarrolla un programa para PLC es posible observar que parte de la lógica programada puede ser re utilizada dentro del mismo proyecto o puede servir de base para otros, así que en busca de optimizar la reutilización de código se recomienda el uso de las instrucciones *Add-On*.

Las instrucciones *Add-On* permiten encapsular la lógica común en un proyecto en forma de conjuntos de instrucciones reutilizables, estas instrucciones funcionan de igual manera a las incorporadas en los entornos de programación de PLCs. El uso de instrucciones *Add-On* permite ahorrar tiempo de programación ya que ofrece la posibilidad de reutilizar código haciendo que exista consistencia con otros proyectos donde se utilice la misma lógica en el algoritmo [48, 49, 50].

La lógica definida para una instrucción *Add-On* puede ser programada en cualquiera de los lenguajes soportados, lógica de escalera (*Ladder*), diagramas de bloques funcionales y texto estructurado. Estas instrucciones permiten encapsular algoritmos complejos con la posibilidad de visualizar la lógica en cada una de las instancias creadas, además ofrecen una interfaz sencilla para configurar y usar la lógica que contiene, esto por medio de la definición de parámetros y la posibilidad de consultar las ayudas con las que cuentan las instrucciones [48, 49, 50].

Las instrucciones *Add-On* también ofrecen la posibilidad de proteger la lógica que contienen, esto evita modificaciones no deseadas facilitando el soporte a un algoritmo en donde han sido implementadas y protegiendo la propiedad intelectual [48, 49, 50].

1.3.1.2. Terminología

La siguiente figura define los términos más relevantes para entender la estructura de una instrucción *Add-On* [50]:

Term	Definition
Argument	An argument is assigned to a parameter of an Add-On Instruction instance. An argument contains the specification of the data used by an instruction in a user program. An argument can contain the following: • A simple tag (for example, L101) • A literal value (for example, 5) • A tag structure reference (for example, Recipe.Temperature) • A direct array reference (for example, Buffer[1]) • An indirect array reference (for example, Buffer[Index+1]) • A combination (for example, Buffer[Index+1].Delay)
Parameter	Parameters are created in the Add-On Instruction definition. When an Add-On Instruction is used in application code, arguments must be assigned to each required parameter of the Add-On Instruction.
InOut parameter	An InOut parameter defines data that can be used as both input and output data during the execution of the instruction. Because InOut parameters are always passed by reference, their values can change from external sources during the execution of the Add-On Instruction.
Input parameter	For an Add-On Instruction, an Input parameter defines the data that is passed by value into the executing instruction. Because Input parameters are always passed by value, their values cannot change from external sources during the execution of the Add-On Instruction.
Output parameter	For an Add-On Instruction, an Output parameter defines the data that is produced as a direct result of executing the instruction. Because Output parameters are always passed by value, their values cannot change from external sources during the execution of the Add-On Instruction.
Passed by reference	When an argument is passed to a parameter by reference, the logic directly reads or writes the value that the tag uses in controller memory. Because the Add-On Instruction is acting on the same tag memory as the argument, other code or HMI interaction that changes the argument's value can change the value while the Add-On Instruction is executing.
Passed by value	When an argument is passed to a parameter by value, the value is copied in or out of the parameter when the Add-On Instruction executes. The value of the argument does not change from external code or HMI interaction outside of the Add-On Instruction itself.
Module reference parameter	A module reference parameter is an InOut parameter of the MODULE data type that points to the Module Object of a hardware module. You can use module reference parameters in both Add-on Instruction logic and program logic. Since the module reference parameter is passed by reference, it can access and modify attributes in a hardware module from an Add-On Instruction.

Figura 1.20: Términos usados en las instrucciones *Add-On*.
Tomado de [50].

1.3.1.3. Componentes

A continuación se realiza una breve descripción de los componentes más relevantes de una instrucción *Add-On* [50]:

- **General information:** contiene la información que se ingresa en el momento de crear la instrucción. La pestaña cuenta con detalles como: *description*, *type*, *revisión*, *revision note* y *vendor*.
- **Parameters:** es el conjunto de variables que sirven para configurar la instrucción u obtener los resultados entregados por esta. Existe 3 maneras en que pueden usarse los parámetros, como entrada (*Input*), se ubican en la parte izquierda de la instrucción, como salida (*Output*), se ubican en la parte derecha de la instrucción o como entrada/salida (*InOut*).

- **Local Tags:** es el conjunto de *tags* que utiliza la lógica dentro de la instrucción.
- **Logic routine:** la rutina lógica define la funcionalidad principal de la instrucción. Es el código que se ejecuta cada vez que se llama a la instrucción.
- **Help:** el nombre, la revisión, la descripción y las definiciones de parámetros se utilizan para crear automáticamente la ayuda de las instrucciones. Utilice *Extended Description Text* para proporcionar documentación de ayuda adicional para la instrucción *Add-On*.

1.3.2. Creación de instrucciones Add-On

En esta sección de creará la instrucción *P_FT_Primer_Orden_AOI* la cual permitirá simular la función de transferencia de un sistema de primer orden más tiempo muerto. Esta instrucción permite además agregar un disturbio al sistema con el fin de poder probar un controlador posteriormente.

Para crear una instrucción *Add-On*, en el entorno de *Logix Designer* haga clic derecho sobre la carpeta de *Add-On Instructions* y seleccione *New Add-On Instruction*, entonces se desplegará la ventana para crear la nueva instrucción, para finalizar haga clic en *OK*. Para el caso de *P_FT_Primer_Orden_AOI* se agrega la siguiente información:

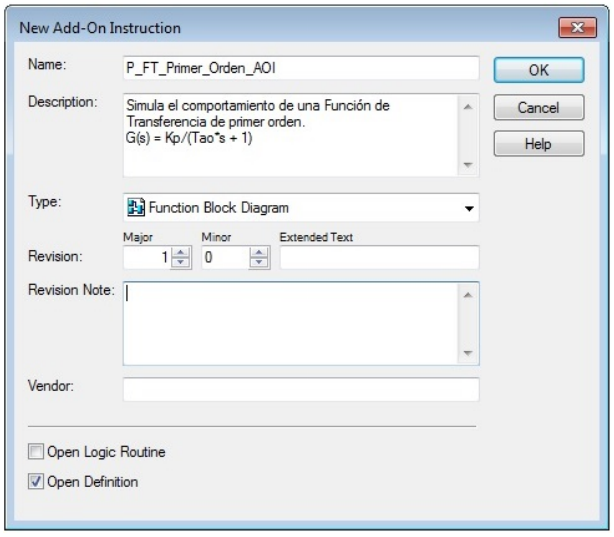
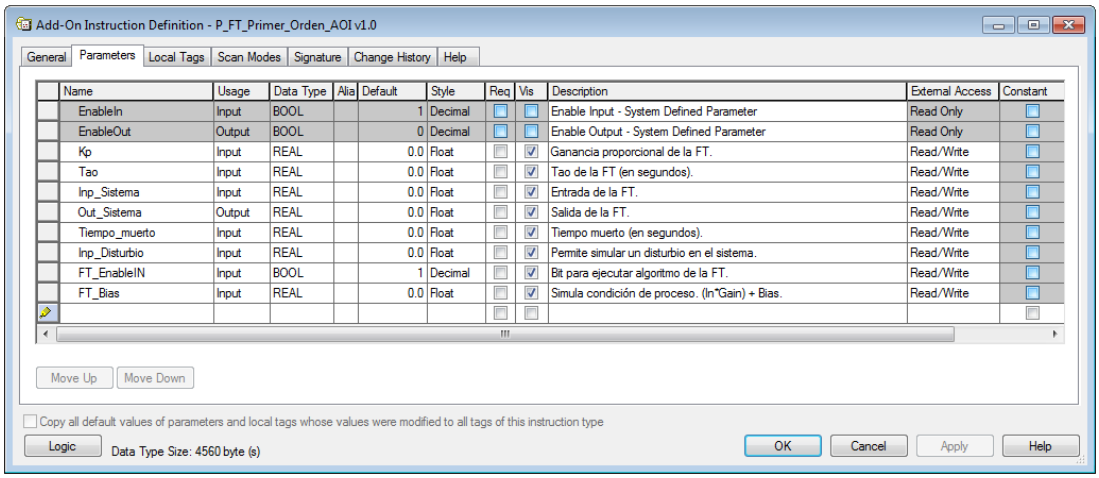


Figura 1.21: Configuración de la instrucción *Add-On*.
Fuente propia.

Una vez creada la instrucción esta aparecerá en la carpeta de *Add-On Instructions* de su proyecto de *Logix Designer*, desde aquí puede cambiar la configuración de la nueva instrucción *Add-On*.

El siguiente paso es crear los parámetros de la instrucción, para ello haga doble clic sobre la instrucción creada, esto desplegará la ventana *Add-On Instruction Definition*, haga clic en la pestaña *Parameters*. Agregue tantos parámetros como sea necesario, para ello debe asignar: *Name* (Nombre), *Usage* (cómo se usará, como *Input*, *Output* o *InOut*), *Data Type* (tipo de dato), *Default* (si el parámetro tendrá un valor por defecto), *Style* (depende del tipo de dato seleccionado), *Vis* (si el parámetro será o no visible en la instrucción), *Description* (descripción), *External Access*, entre otros. Para el caso de *P_FT_Primer_Orden_AOI* se crearon los siguientes parámetros:



Name	Usage	Data Type	Alia	Default	Style	Req	Vis	Description	External Access	Constant
EnableIn	Input	BOOL		1	Decimal	☐	☐	Enable Input - System Defined Parameter	Read Only	☐
EnableOut	Output	BOOL		0	Decimal	☐	☐	Enable Output - System Defined Parameter	Read Only	☐
Kp	Input	REAL		0.0	Float	☐	☑	Ganancia proporcional de la FT.	Read/Write	☐
Tao	Input	REAL		0.0	Float	☐	☑	Tao de la FT (en segundos).	Read/Write	☐
Inp_Sistema	Input	REAL		0.0	Float	☐	☑	Entrada de la FT.	Read/Write	☐
Out_Sistema	Output	REAL		0.0	Float	☐	☑	Salida de la FT.	Read/Write	☐
Tiempo_muerto	Input	REAL		0.0	Float	☐	☑	Tiempo muerto (en segundos).	Read/Write	☐
Inp_Disturbio	Input	REAL		0.0	Float	☐	☑	Permite simular un disturbio en el sistema.	Read/Write	☐
FT_EnableIN	Input	BOOL		1	Decimal	☐	☑	Bit para ejecutar algoritmo de la FT.	Read/Write	☐
FT_Bias	Input	REAL		0.0	Float	☐	☑	Simula condición de proceso. (In*Gain) + Bias.	Read/Write	☐

Figura 1.22: Parámetros de la instrucción *Add-On*.
Fuente propia.

El siguiente paso es crear las *tags* locales que se usarán dentro de la lógica de la instrucción, para esto haga clic sobre la pestaña *Local Tags* y agregue tantas *tags* como sea necesario, este paso se puede llevar a cabo a medida que se programe la lógica para agregar solo lo necesario o incluso las *tags* pueden ser creadas directamente desde el entorno donde se programa la lógica. Para el caso de *P_FT_Primer_Orden_AOI* se crearon las siguientes *tags*:

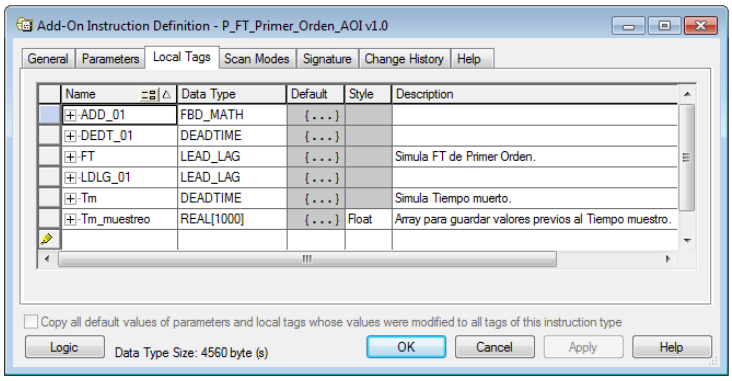


Figura 1.23: *Tags* de la instrucción *Add-On*.
Fuente propia.

Para terminar la configuración haga clic en *OK* y guarde los cambios hechos en el proyecto haciendo clic en *Save* (icono de disquete) en la esquina superior izquierda.

El siguiente paso es programar la lógica de la instrucción, para esto haga doble clic en la instrucción y haga clic en el botón *Logic* ubicado en esquina inferior izquierda o haga clic sobre la instrucción para desplegar las opciones *Parameters and Local Tags* y *Logic*. La instrucción *P_FT_Primer_Orden_AOI* fue creada con *Function Block Diagram* como tipo, esto define el lenguaje en el que será programada la lógica de la instrucción. Para el caso de *P_FT_Primer_Orden_AOI* la lógica fue programada de la siguiente manera:

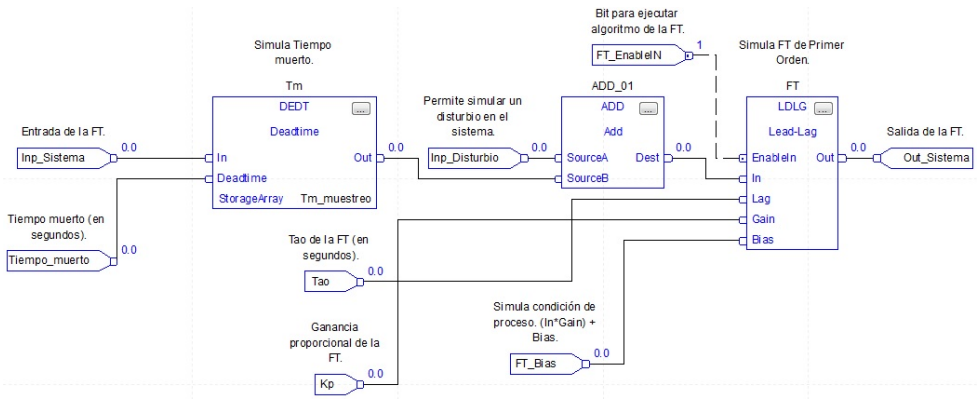


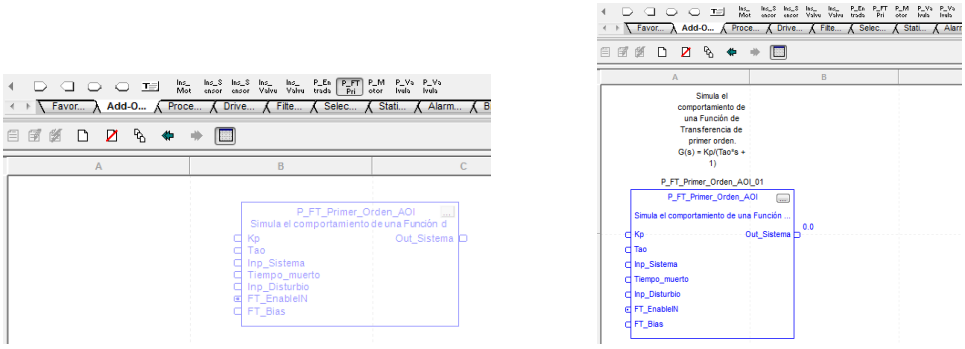
Figura 1.24: Lógica de la instrucción *Add-On*.
Fuente propia.

Con lo anterior la instrucción ya está lista para crear una instancia de esta y utilizarla dentro de la lógica del programa.

1.3.3. Uso de instrucciones Add-On

Para utilizar una instrucción *Add-On* lo primero es hacer una instancia de esta. A continuación se muestra la creación de instancias de *P_FT_Primer_Orden_AOI* en un programa.

Para crear una instancia de la instrucción se selecciona la rutina donde estará ubicada y se arrastra la instrucción desde la pestaña *Add-On* de la barra de instrucciones ubicada en la parte superior del entorno de *Logix Designer*, ver figura 1.25a, hasta el espacio de la lógica de la rutina. Para el caso de *P_FT_Primer_Orden_AOI* la instancia fue creada en una rutina de tipo *Function Block Diagram* de la siguiente manera:

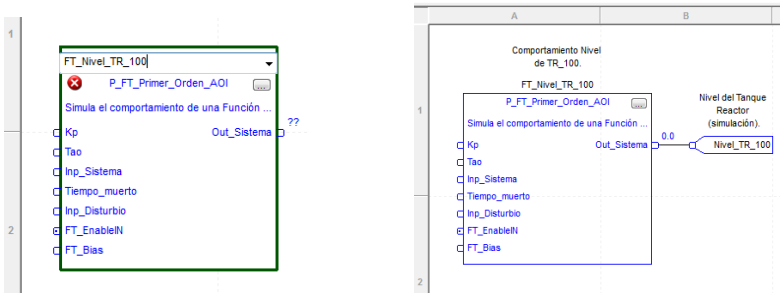


(a) Ubicación de instrucción *Add-On*.

(b) Instrucción *Add-On* lista para configurar parámetros.

Figura 1.25: Creación de instancia de instrucción *Add-On*.
Fuente propia.

En la figura 1.25b se muestra la instrucción lista para ser configurada. Lo primero es crear otra *tag* y asociarla a la instrucción, esto porque se creó una *tag* local automáticamente al arrastrar la instrucción al *Function Block Editor* y al ser local no se tendrá acceso a la instancia desde otra rutina de otro programa del proyecto. Para esto haga clic sobre la *tag* (en este caso: *P_FT_Primer_Orden_AOI_01*) y escriba otro nombre, ver figura 1.26a. En este momento aún la *tag* no existe y por eso aparece el símbolo de error, para crearla haga clic derecho sobre la nueva *tag* y seleccione *New*, con esto se desplegará la ventana *New Tag*, configure el alcance de la *tag* para que pueda ser utilizada en todo el programa, para esto en la sección *Scoope* seleccione el nombre del proyecto, esto quiere decir que la nueva *tag* tiene como alcance todo el proyecto y por esto puede ser utilizada en cualquier parte del mismo. Para el caso de *P_FT_Primer_Orden_AOI* la instancia fue creada con la siguiente *tag*, ver figura 1.26b:



(a) Cambio de *tag* de la instancia.

(b) *Tag* asociada correctamente a instancia de la instrucción.

Figura 1.26: Instancia de instrucción *Add-On*.
Fuente propia.

La figura 1.26b muestra como mediante *Function Block Diagram* es más fácil entender qué parámetros de entrada y salida tiene configurados la instancia, en este caso el parámetro de salida *Out_Sistema* es asociada a la *tag Nivel_TR_100*.

Una de las ventajas del uso de instrucciones *Add-On* es que permiten la reutilización de código, en la siguiente figura se muestra como la instrucción ha sido usada para modelar diferentes variables dentro un proceso.

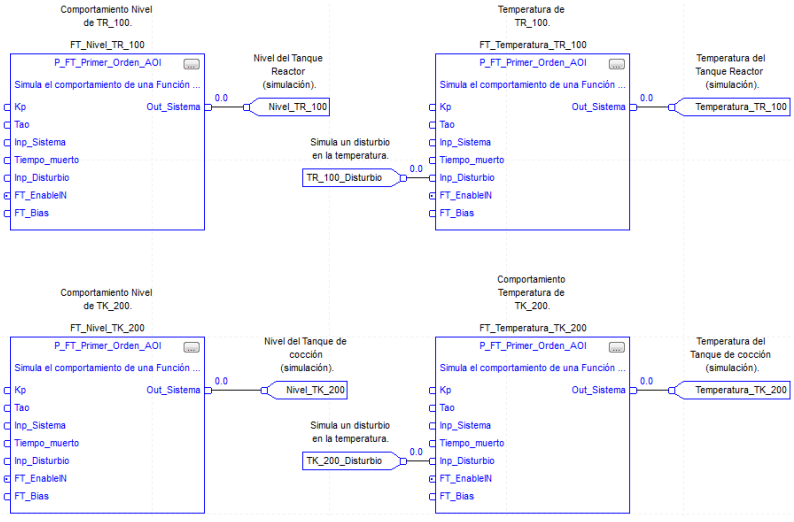


Figura 1.27: Reutilización de código mediante instrucciones *Add-On*.
Fuente propia.

Con las instancias creadas ahora es posible utilizar todos los parámetros de la instancia. A continuación se muestra un ejemplo con las instancias creadas para simular los cambios

en el nivel y la temperatura de un tanque.

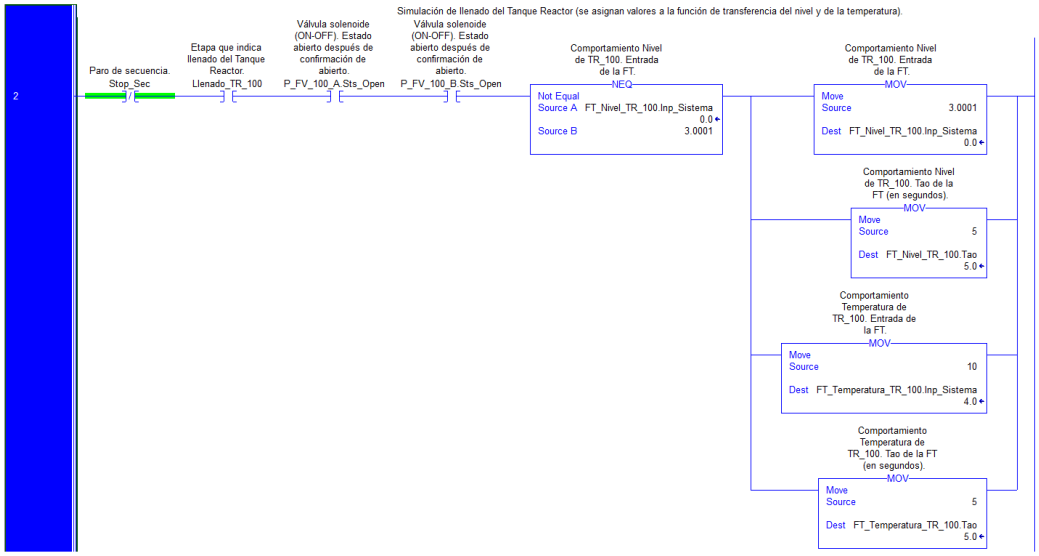


Figura 1.28: Uso de las instancias de las instrucciones *Add-On* en un algoritmo.
Fuente propia.

En la figura 1.28 se muestra una porción de código programado en lenguaje *Ladder* en el cual por medio de las instrucciones *MOV* se asignan valores a los parámetros de las instancias de acuerdo a condiciones previas. Para tener acceso a un parámetro especifico de la instancia se debe escribir la *tag* de la instancia seguida por el operador punto ".", con esto tenemos acceso a todos los parámetros de la instancia que estén habilitados para lectura o escritura (*Read/Write*).

En la porción de código de la figura 1.28 se asignan valores a los parámetros *Inp_Sistema* y *Tao* de las instancias *FT_Nivel_TR_100* y *FT_Temperatura_TR_100*, estos parámetros son los necesarios para generar cambios en la salida de la instrucción.

1.4. Optimización de objetos gráficos reutilizables mediante el uso de objetos globales

1.4.1. Conceptos básicos

Los objetos globales son útiles para brindar modularización y repetibilidad eficiente de elementos comunes en un proyecto, utilizar de manera efectiva los objetos globales permite reducir tiempos de desarrollo ya que estos permiten vincular la apariencia y comportamiento de un objeto gráfico a varias copias de dicho objeto en una misma aplicación, cada copia del objeto se llamará instancia. Cuando se realizan cambios en el objeto original o base se efectúan los cambios en las instancias realizadas [46, 47].

Cuando se está realizando la creación de un objeto global se debe asignar un parámetro o *tag placeholders*, estos permiten asignar valores a los objetos instanciados del objeto base. Un *tag placeholder* se representa con el carácter "#" seguido de un número del 1 al 500. Durante el tiempo de ejecución o visualización de prueba los *tag placeholders* son reemplazados con los nombres de etiquetas reales o *tags* para cada objeto del proceso, para hacer esto, se debe especificar los *tags* en los archivos de parámetros o agregando un valor a cada objeto por separado [46, 47].

1.4.2. Creación de objetos globales

Para la creación de objetos globales se debe ubicar el *Explorador* de *Factory Talk View* en donde se encuentran diferentes funciones de la aplicación como el servidor HMI. El *Explorador* contiene los componentes y editores del proyecto HMI, entre ellos se encuentra la carpeta *Graphics* la cual contiene, entre otras, la sección de *Global Objects*, en esta sección se lleva a cabo la creación y edición de los objetos globales de la aplicación.

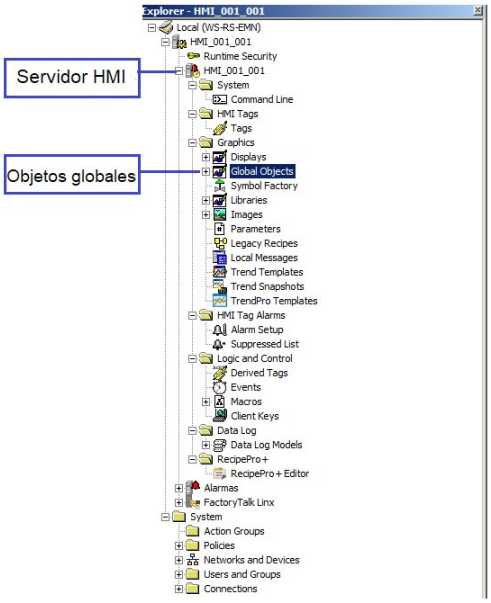


Figura 1.29: Árbol del proyecto.
Fuente propia.

Los objetos globales son considerados como plantillas globales que son creados en un *Display* dentro de *Global Objects*. Para la creación de un objeto global se debe hacer clic derecho en esta sección seleccionar *New* para abrir un nuevo *Display global* en el cual se deben arrastrar los objetos que se crearan, en la figura 1.30 se realiza la creación de la representación de una válvula solenoide como objeto global y se asigna la animación deseada.

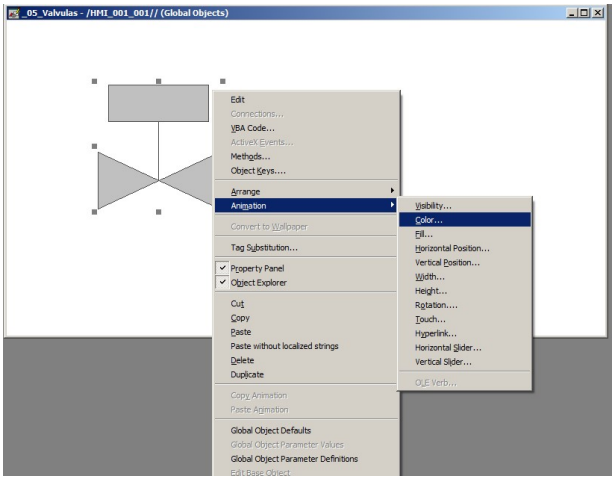


Figura 1.30: Configuración de animación de objeto global válvula solenoide.
Fuente propia.

Para asignar un *tag* a la válvula solenoide que active a la animación, se abre el explorador de *tags* y se selecciona la *tag* correspondiente (para el ejemplo, *[PLC]P_FV_100_A.Sts_Open*) correspondiente a la válvula *P_FV_100_A*, esta *tag* corresponde a la instancia de la instrucción *Add-On* creada en el proyecto de *Logix Designer*, *Sts_Open* corresponde al parámetro de salida del *Add-On* que indica que la válvula está abierta cuando su valor es 1. Este estado hace que el cuerpo de la válvula solenoide cambie de color cuando la válvula se encuentre abierta. Los objetos globales son particularmente adecuados para los tipos de datos de PLC como *UDT* (datos definidos por el usuario) o *AOI* (Instrucciones *Add-On*) ya que estos permiten una estructura de etiquetas coherente y modular.

Siguiendo las buenas prácticas para el diseño de HMI de alto rendimiento descritas en la sección 1.2, se define el color gris oscuro para indicar que la válvula se encuentra cerrada y un color gris claro para indicar que está abierta la válvula.

Los parámetros o *tag placeholders* permitirán asignar valores a los objetos instanciados del objeto base, al parametrizar una *tag* esta pasará a tener una estructura global y una estructura fija o invariable gracias a su estructura lógica modular. Continuando con el ejemplo se definirá el parámetro *#1* que reemplazará la *tag* de la instrucción *Add-On* instanciada pasando a tener una estructura genérica y el parámetro del *Add-On* quedará fijo conservando una estructura invariable, en la expresión 1.1 se muestra como debe ser la estructura del parámetro y en la expresión 1.2 se ejemplifica como debe realizarse la parametrización de la *tag* asociada.

$$\#1 = tag \tag{1.1}$$

$$\underbrace{\{[PLC]P\_FV\_100\_A.Sts\_Open\}}_{tag} \rightarrow \overbrace{\{\#1.Sts\_Open\}}^{tag\ parametrizado} \tag{1.2}$$

La importancia de los objetos globales y su parametrización está en la modularizacion y repetibilidad eficiente de elementos comunes en el proyecto, esto significa qué si se tienen 10 válvulas solenoides con la misma configuración en el proyecto, pero cuentan con diferente *tag*, basta con instanciar el objeto y asignar la estructura de la *tag* propia de cada válvula. En la figura 1.31 se muestra cómo debe ser parametrizada la *tag*.

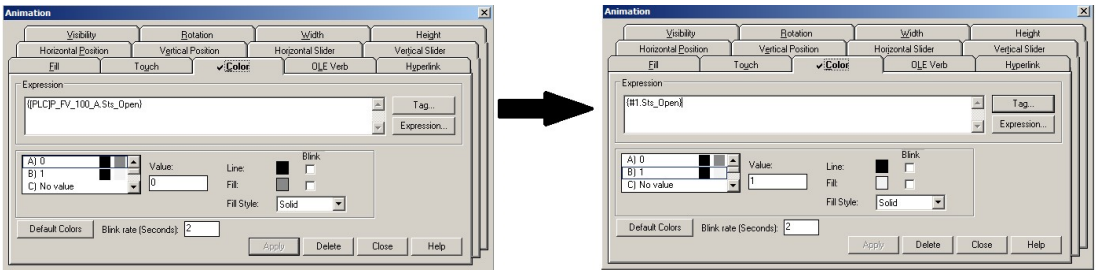


Figura 1.31: Parametrización de la *tag* asociada.
Fuente propia.

Los parámetros son usados para especificar las *tags* a configurar para un objeto que está en la aplicación. Para definir un parámetro a un objeto (ejemplo para una válvula), se debe realizar clic derecho en el objeto que se encuentra en la sección *Global Objects* y seleccionar la opción de *Global Object Parameter Definitions*, después asignar un nombre y una descripción al parámetro como se visualiza en la figura 1.32, con lo anterior el objeto queda listo y configurado para ser usado en el proyecto.

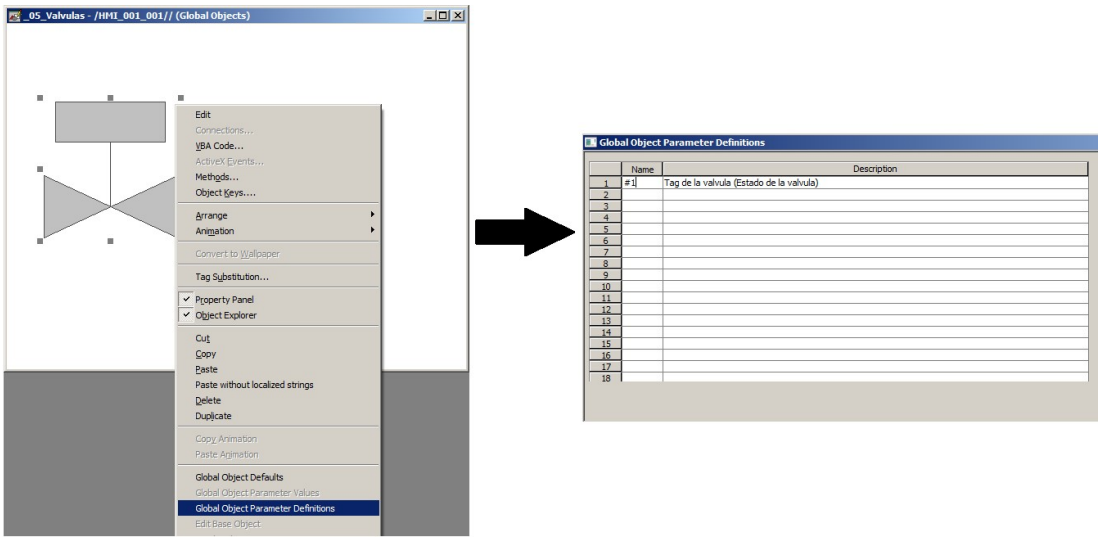


Figura 1.32: Definición de parámetros del objeto global.
Fuente propia.

1.4.3. Uso de objetos globales

Para utilizar un objeto global basta con crear una instancia del objeto base copiándolo y pegándolo en el *display* donde será usado. Desde el momento que se realiza una instancia del objeto se mantiene un vínculo con el objeto global.

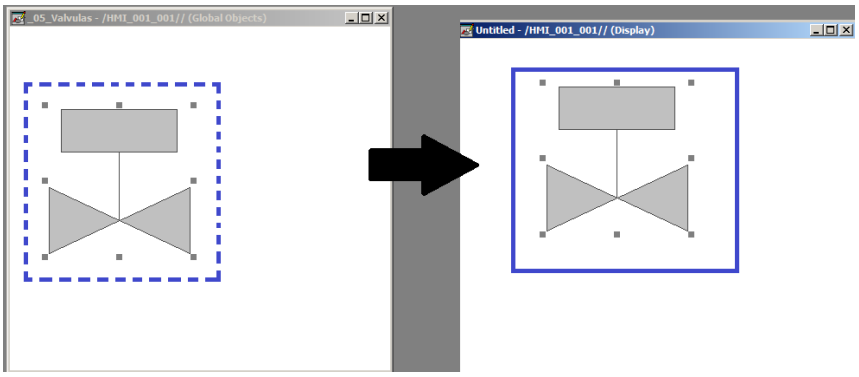


Figura 1.33: Creación de la instancia de un objeto global.
Fuente propia.

Los objetos globales son genéricos y es necesario proporcionarle un valor al parámetro durante su ejecución, una vez instanciado los objetos se debe asignar el *tag* al parámetro definido. Seleccione el objeto y haciendo clic derecho en el objeto seleccione *Global Object Parameter Value* para abrir el panel en donde aparecen los parámetros definidos anteriormente en la creación del objeto global. Se asignará el *tag* (para el ejemplo: *[PLC]P_FV_100_A*). Recuerde que éste había sido reemplazado por el *parámetro #1* anteriormente en la sección 1.4.2, en el objeto instanciado el parámetro adquiere este valor. A continuación se muestra como debe ser la asignación del valor en el panel de valores del parámetro (recuerde la expresión 1.2), la figura 1.34 muestra lo mencionado.

Parámetro: #1
Value: [PLC]P_FV_100_A

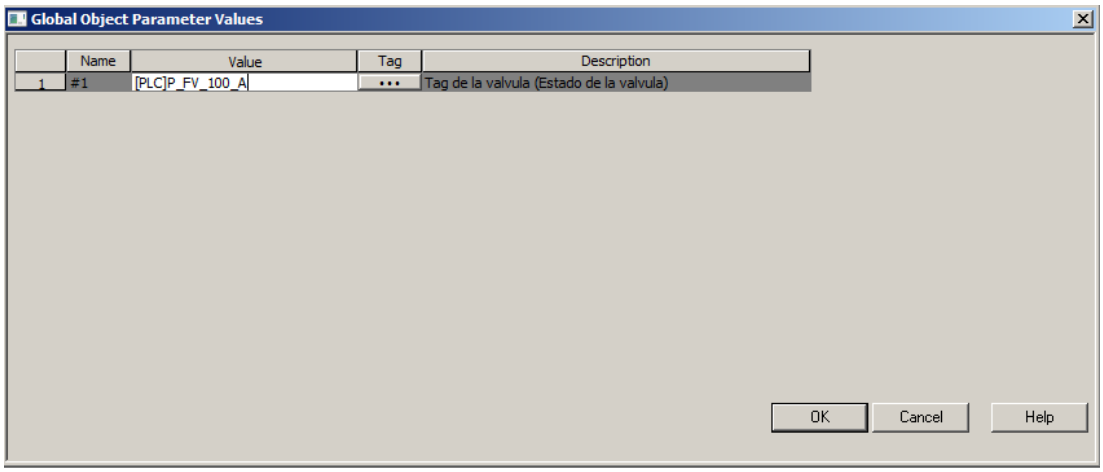


Figura 1.34: Asignación del valor o *tag* al parámetro.
Fuente propia.

1.5. Optimización de aplicaciones mediante el uso de las herramientas de PlantPAx

1.5.1. Generalidades de PlantPAx

El sistema *PlantPAx* es una propuesta de un enfoque moderno del control distribuido, cuenta con tecnología común con otras disciplinas de la automatización lo que permite un flujo continuo de información en toda la planta, brindando la posibilidad de optimización en una empresa [51].

PlantPAx utiliza estándares de *Rockwell Automation* para proponer un sistema de control distribuido (DCS) moderno, flexible y escalable con el fin de brindar confiabilidad, funcionalidad y el rendimiento esperado de un DCS [52].

Un *PlantPAx DCS* cuenta con múltiples elementos que pueden ser implementados de acuerdo a las necesidades del proceso, estos elementos son [52]:

- **Process Automation System Server (PASS):** es el elemento del sistema que contiene los componentes software esenciales para la correcta ejecución del sistema. Los componentes de software esenciales incluyen el servidor de datos, el servidor HMI y el servidor de alarmas.
- **Engineering Workstation (EW):** encargado de soportar la configuración del sistema, el desarrollo de aplicaciones y funciones de mantenimiento. Es la estación central de trabajo donde se monitorea y se mantiene la operación del sistema.
- **Operator Workstation (OWS):** proporciona la vista gráfica y la interfaz en el proceso. El *OWS* admite la interacción del operador y no está destinado a apoyar actividades de desarrollo o mantenimiento, aunque estas actividades son posibles si se desea.
- **Process controller:** hace referencia a los componentes hardware del sistema y sus tipos de conexiones.
- **Application servers:** hace referencia a todas las aplicaciones que pueden ser configuradas para ser utilizadas como servidores en el sistema.

1.5.2. Biblioteca de objetos de proceso

La biblioteca de objetos de proceso de *Rockwell Automation* es un grupo de objetos codificados predefinidos que ofrecen estrategias, funcionalidad y rendimiento a un sistema de control. La biblioteca ha sido creada considerando normas internacionales sobre aspectos como color, funcionalidad y símbolos, lo que hace que los objetos de la biblioteca sean una excelente opción para proyectos de muchos sectores industriales [22, 53].

La biblioteca de objetos de proceso cuenta con instrucciones para motores, válvulas, drivers, interlocks, permisivos y otros objetos usuales en un sistema de control que pueden ser usados con el sistema *PlantPAx*. Sin embargo, hacer uso de los objetos de la biblioteca no es equivalente a diseñar un sistema *PlantPAx*, ya que este requiere de un conjunto de elementos más complejo [22, 53].

La biblioteca de objetos incluye código para el controlador (instrucciones *Add-On*), elementos para *displays* (objetos globales) y *faceplates* los cuales proporcionan herramientas de visualización y operación para el personal [22, 53]. Un ejemplo de lo anterior puede ser observado en la figura 1.35.

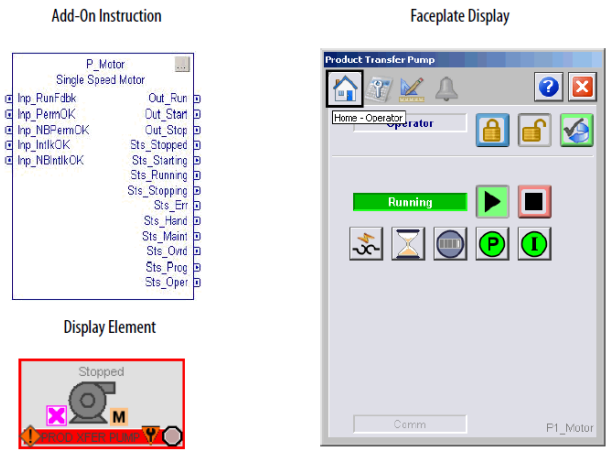


Figura 1.35: Componentes de la biblioteca de objetos de *PlantPAx*.
Tomado de [52].

El uso de los objetos de la biblioteca ofrece beneficios como [22, 53]:

- Simulación y anulación de modos de operación, lo que favorece las operaciones de mantenimiento.

- Objetos gráficos basados en normas que facilitan la identificación de situaciones anormales en el proceso.

- Diseño reutilizable: las instrucciones predefinidas permiten de manera no muy compleja, controlar, monitorear y solucionar problemas en su proceso, ya que estas cuentan con funcionalidad predefinida para utilizar los dispositivos. El uso de objetos de la biblioteca convierte el desarrollo de su algoritmo y/o HMI en una configuración de objetos predefinidos, ahorrando todo el tiempo necesario para diseñar cada función de cada dispositivo en particular.

- Desarrollo simple: cada instancia de cada instrucción puede ser configurada de manera particular sin alterar la lógica original de la instrucción. Cada instrucción cuenta con la lógica necesaria para satisfacer una amplia gama de necesidades. Los objetos gráficos permiten configurar los dispositivos desde un entorno gráfico por medio de los *faceplates*, lo que hace más sencillo e intuitivo la configuración de estos.

- Datos en tiempo real: los *faceplates* ofrecen a los usuarios la posibilidad de conocer en tiempo real las condiciones de los dispositivos, esto por medio de alarmas e información de diagnóstico que permiten a los usuarios monitorear condiciones específicas y tomar mejores decisiones.

La figura 1.36 muestra de manera sencilla el proceso para configurar un objeto de la biblioteca. El utilizar estos elementos permite obtener mejoras en su sistema en cuanto a operación, mantenimiento y eficiencia [53].

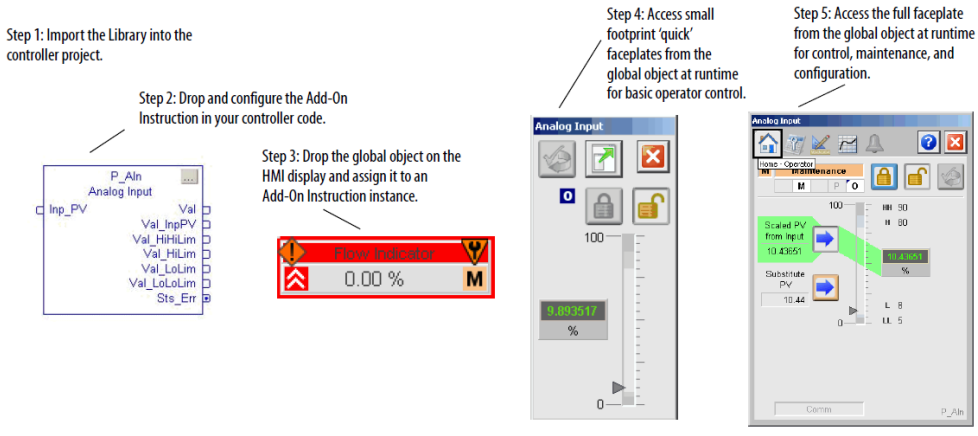


Figura 1.36: Pasos para configurar un objeto de la biblioteca de *PlantPAx*.
Tomado de [52].

La biblioteca además cuenta con herramientas denominadas "*Process Strategies*" las cuales proveen una serie de instrucciones ya conectadas que buscan reducir los tiempos de implementación y mejorar los objetivos de control para los dispositivos de proceso. La biblioteca cuenta con *Process Strategies* dirigidas a configurar el procesamiento de entradas y salidas, control regulatorio, control se secuencias, motores, válvulas, procesos de diagnóstico, entre otros [22, 53].

1.5.2.1. Uso de Instrucciones Add-On

A continuación se describen los pasos para usar las instrucciones *Add-On* que hacen parte de la biblioteca de objetos de proceso de *PlantPAx*.

1. Lo primero es seleccionar correctamente la instrucción que permitirá controlar el dispositivo con el que cuenta el proceso. Para ello tenga en cuenta la funcionalidad y características del dispositivo para utilizar la instrucción adecuada que permita aprovechar por completo el dispositivo con el que cuenta.
   La biblioteca cuenta con documentación muy completa para conocer todas las opciones de instrucciones disponibles antes de elegir una. Dentro de cada manual de cada instrucción se puede encontrar secciones que le ayudarán a elegir. La sección *Guidelines* ofrece pautas para saber cuándo debe o no utilizar dicha instrucción y para qué tipo de dispositivo en específico está diseñada.
   La sección *Functional Description* provee la lista de características con las que cuenta la instrucción y permite conocer para qué puede ser utilizada dentro del proceso.

2. El siguiente paso es importar dentro del proyecto las instrucciones seleccionadas. Una instrucción *Add-On* se define una vez en cada proyecto y se puede instanciar varias veces en el código de su aplicación. Para importar las instrucciones realice los siguientes pasos:

   a) En el entorno de *Logix Designer* haga clic derecho sobre la carpeta de *Add-On Instructions* y seleccione *Import Add-On Instruction*, entonces la ventana para importar instrucciones se desplegará.

   b) Seleccione el archivo de la instrucción que quiere agregar a su proyecto, haga clic en *Import* y espere hasta que aparezca la ventana *Import Configuration*.

   c) Revise los detalles de la configuración y haga clic en *OK*.

Una vez importada correctamente la instrucción esta puede observarse en la carpeta *Add-On Instructions* de su proyecto.

3. Lo siguiente es crear una instancia de la instrucción *Add-On* para que un dispositivo del proceso pueda ser utilizado en el algoritmo. Las instrucciones *Add-On* pueden ser instanciadas en cualquiera de los lenguajes de programación: *Ladder Diagram*, *Function Block Diagram* o *Structured Text*. Para esto realice lo mencionado en la sección 1.3.3. A manera de resumen se enumeran los pasos para crear una instancia.

    a) En la rutina seleccionada, desde la pestaña *Add-On* de la barra de instrucciones, arrastre la instrucción seleccionada hasta el espacio de la lógica de la rutina.

    b) Asigne un nombre, alcance, descripción y otros detalles adecuados para la *tag* de la instancia.

4. Con lo anterior la instancia ya puede ser utilizada dentro del algoritmo, lo siguiente es configurar los parámetros de la instancia y realizar las conexiones para que el dispositivo funcione de acuerdo a la necesidad del proceso, para ello siga los siguientes pasos:

    a) Haga doble clic sobre la instancia para abrir los parámetros de esta, por medio del *check box* de la columna *Vis* active o desactive la visibilidad de los parámetros y deje solo los que sean necesarios para realizar las conexiones y/o para conocer rápidamente alguna entrada o salida con la que cuenta.

    b) Configure cada parámetro de acuerdo a las características del dispositivo y a lo requerido para la lógica de su proceso.

    c) Realice las conexiones necesarias para las entradas y salidas de la instancia creada. Estas conexiones pueden hacerse a parámetros de otras instrucciones *Add-On*, a *tags* o a direcciones de salida o entrada del módulo con el que cuente el controlador.

Con lo anterior la instancia ya está configurada y lista para ser utilizada en la lógica del algoritmo y/o para ser asignada a algún objeto del HMI del proceso. En la sección 1.3.3 se muestra un ejemplo del uso de una instrucción *Add-On* en un algoritmo y en la sección 1.5.2.2 como se usa en un objeto de un HMI.

1.5.2.2. Uso de Objetos gráficos

A continuación se describen los pasos para usar los objetos gráficos *Add-On* que hacen parte de la biblioteca de objetos de proceso de *PlantPAx*.

1. Como primer paso se debe importar correctamente los archivos de visualización mediante una serie de pasos ordenados:

   a) Antes de importar cualquier objeto se debe importar la carpeta de *imagenes*. En el árbol del proyecto se ubica la carpeta *Graphics*, dentro de esta carpeta se encuentra todo lo referente a los objetos gráficos. En la sección *Images* haga clic derecho y seleccione *Add Component Into Application*, se abrirá una ventana donde se deben seleccionar las imágenes utilizadas por los objetos de *PlantPAx* las cuales se encuentran en la carpeta *Graphics* de la biblioteca *Process Objects Library* de *Rockwell Automation*.

   b) Para agregar un objeto global al proyecto debe ubicarse dentro de la carpeta *Graphics* ubicado en el árbol del proyecto, ubique la sección de *Global Objects*, haga clic derecho sobre esta sección y seleccione *Add Component Into Application*, se abrirá una ventana donde se debe seleccionar el objeto global según la funcionalidad y características del dispositivo que desea visualizar, ver figura 1.37, los objetos globales son archivos tipo "*.ggfx*".

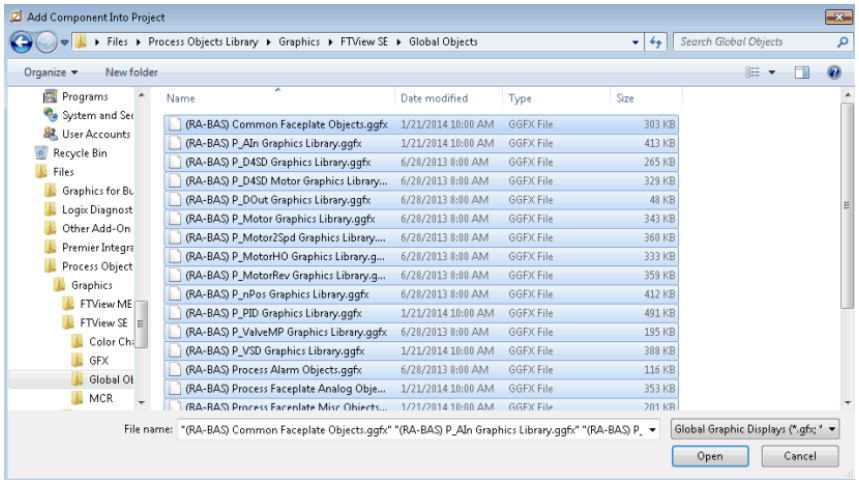


Figura 1.37: Selección de objetos globales a importar.
Tomado de [52].

   c) Los instrumentos o dispositivos que se visualizan en el HMI cuentan con *faceplates* para su manipulación e interacción por parte del operario. Para agregar

un *faceplate*, como una funcionalidad más de un objeto, se debe ubicar en la carpeta *Graphics* como se ha realizado en pasos anteriores y ubicar la sección *Display*, haga clic derecho sobre esta sección y seleccione *Add Component Into Application*, se abrirá una ventana donde debe ser seleccionado únicamente las pantallas correspondiente a los *faceplates* que se utilizaran, los *displays* son archivos tipo "*.gfx*".

2. Una vez importados los objetos que se utilizaran en el proyecto, lo siguiente será crear una instancia del objeto para visualizar el estado de un dispositivo o bien manipularlo desde la interfaz gráfica. Para esto realice lo mencionado en la sección 1.4.3. A manera de resumen se enumeran los pasos para crear una instancia de un objeto:

   a) En el árbol del proyecto ubique la sección de *Global Objects* y abra el archivo de objetos globales (*.ggfx*) que contiene los objetos gráficos de la biblioteca.

   b) Seleccione el objeto y arrástrelo (o copie y pegue) al *display* donde desea será instanciado.

   c) Una vez el objeto haya sido instanciado haga clic derecho en el objeto global y asigne valores a los parámetros del objeto, los objetos de *PlantPAx* vienen con sus parámetros definidos por defecto y en su descripción dan una breve explicación del valor que se le debe dar a cada parámetro, lo anterior se visualiza en la figura 1.38.

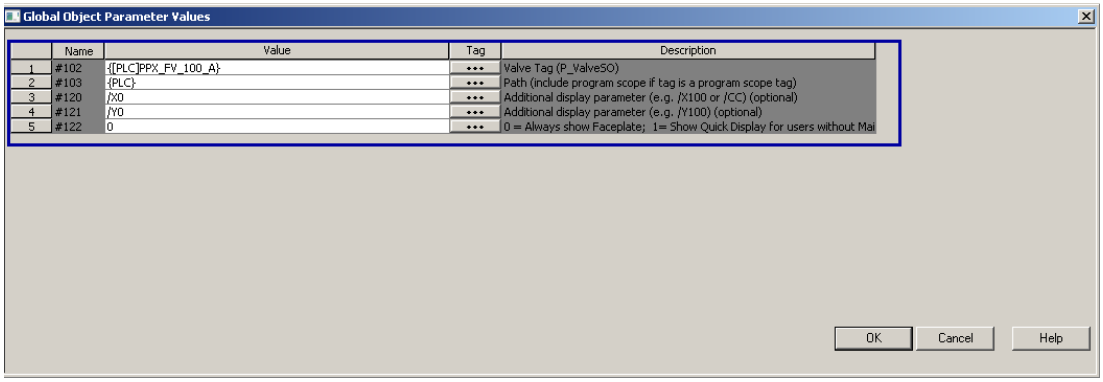


Figura 1.38: Asignación de valores a los parámetros del objeto global.
Fuente propia.

3. Para acceder al *faceplate* de un objeto que ha sido instanciado previamente solo se debe dar clic sobre el objeto cuando la aplicación se encuentre en ejecución, en

la figura 1.39 se visualiza como se despliega el *faceplate* al dar clic sobre el motor, posteriormente desde este se puede encender o apagar el motor.

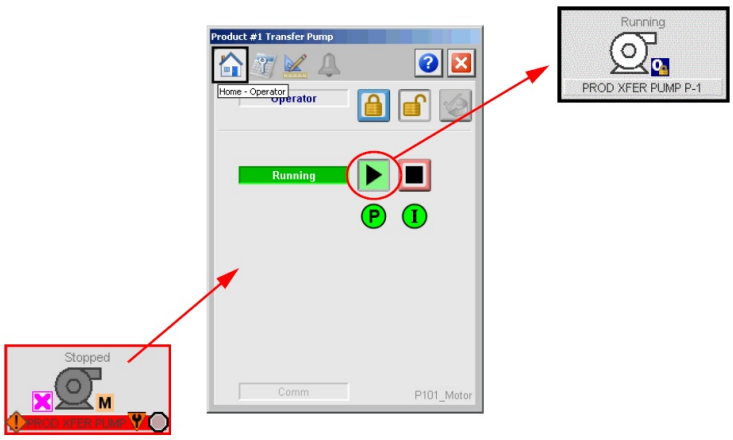


Figura 1.39: Encendido de un motor desde su *faceplate*.
Tomado de [54].

4. El cambio de idioma de *FactoryTalk View* proporciona la capacidad de configurar varios idiomas para una aplicación y ser cambiados durante la ejecución de la aplicación. *FactoryTalk View* cuenta con una pestaña de herramientas donde se ubica la sección *Language*, donde se agregan los diferentes idiomas que se desean configurar para la aplicación. Una vez agregados los idiomas que se configuraran en la aplicación, se debe elegir en la misma sección la opción de *Import* donde aparecerá una ventana con las opciones de importar un archivo de excel (*.xls*) o un archivo de texto (*.txt*), seleccione el archivo del idioma deseado que contiene las cadenas de texto traducidas para los objetos globales cargados en pasos anteriores, en la figura 1.40 se muestra los archivos de idioma que ofrece la biblioteca de *PlantPAx*.

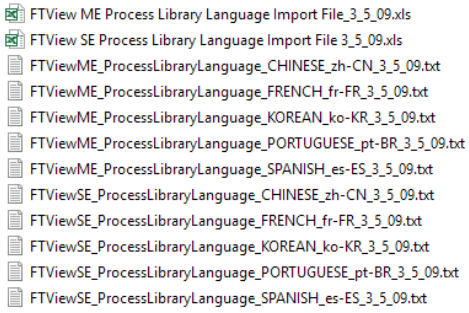


Figura 1.40: Archivos de idioma de la biblioteca de *PlantPAx*.
Fuente propia.

Capítulo 2

Implementación del método

En este capítulo se abarca la implementación de los conceptos del método definido anteriormente, se plantea un caso de estudio basado en el proceso de producción de cerveza específicamente en la etapa de macerado y cocción, se eligió este proceso ya que cuenta con las características básicas de un proceso como una secuencia lógica, control de una variable de proceso, instrumentación básica como válvulas, sensores y motores, además de que la automatización de procesos en la industria cervecera ha sido uno de los casos de éxito de *Rockwell Automation* en donde se han implementado las herramientas de *PlantPAx*.

Se propone para el caso de estudio dos soluciones, una de programación y supervisión mediante el método convencional y otra solución utilizando las herramientas de *PlantPAx*, lo anterior con el fin de crear una noción sobre el tiempo de desarrollo para realizar una propuesta con y sin las herramientas de *PlantPAx*.

2.1. Desarrollo de la solución del caso de estudio mediante el método convencional

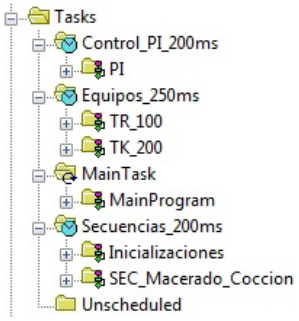
2.1.1. Propuesta de Programación

A continuación se realiza la descripción de la estructura del proyecto y lógica del algoritmo propuesto como solución de programación para el caso de estudio y que servirá de guía

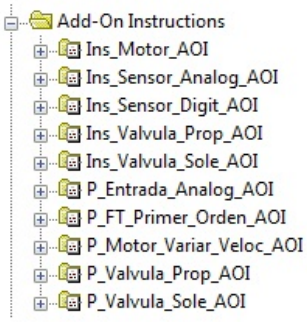
para continuar con la fase experimental de la tesis. La aplicación fue creada en *Studio 5000 Logix Designer*.

2.1.1.1. Estructura del proyecto

El proyecto se encuentra programado mediante tareas y rutinas. La lógica se encuentra estructurada mediante el uso de instrucciones *Add-On*, estas permiten realizar un modelo sencillo de los instrumentos del proceso y crear la lógica encapsulada para manipularlos y usarlos dentro del algoritmo. En la figura 2.1 se muestran las tareas e instrucciones *Add-On* programadas para dar la solución al caso de estudio.



(a) Tareas utilizadas en la solución.



(b) Instrucciones *Add-On* utilizadas en la solución.

Figura 2.1: Estructura del proyecto.
Fuente propia.

2.1.1.2. Estructura del código

En el programa *SEC_ Macerado_ Coccion* se encuentra toda la lógica para simular el proceso de acuerdo a la prosa lógica descrita en la sección A.3.2.

Para simular los procesos de llenado y cambios de temperatura del proceso se utilizaron varias funciones de transferencia de primer orden más tiempo muerto, lo anterior para los niveles y temperaturas de *TR_ 100* y *TK_ 200*, esto permite simular condiciones un poco más reales y además sintonizar los controladores de una manera más sencilla. También es posible simular disturbios en las temperaturas para probar los controladores. Se seleccionaron controladores *PI* porque su configuración es mucho más sencilla e

intuitiva que los *PID* o *PIDE* y esto puede favorecer el desarrollo de estas tareas en la etapa experimental, además sus respuestas fueron muy buenas durante las pruebas realizadas.

Cada secuencia del proceso se encuentra separada en una rutina, dentro de cada rutina está la lógica para que el proceso funcione de manera automática, de manera manual y también si se oprime el botón de *Stop*.

Los cambios en las variables de proceso se ven reflejados al asignar un valor en la entrada (*FT_Nivel_TR _100.In_Sistema*, para el caso del nivel de *TR_100*) y un valor al *tao* (*FT_Nivel_TR_100.Tao*, para el caso del nivel de *TR_100*) de la función de transferencia, lo anterior funciona de igual manera para las temperaturas del proceso.

2.1.1.3. Instrucciones Add-On

Las instrucciones *Add-On* cuyo nombre inicia con "Ins" son usadas para modelar de manera sencilla el comportamiento del instrumento que representa. Las instrucciones cuyo nombre inicia con "P" son usadas para manipular los instrumentos y obtener información de estos, estas instrucciones fueron creadas considerando algunas de las herramientas que proporcionan las instrucciones *Add-On* de *PlantPAx*.

- **Ins_Motor_AOI**: simula un Motor al que se le puede variar la velocidad.
- **Ins_Sensor_Analog_AOI:** simula un Sensor que entrega una Señal Analógica.
- **Ins_Sensor_Digit_AOI:** simula un Sensor que entrega una Señal Discreta.
- **Ins_Valvula_Prop_AOI:** simula una Válvula Proporcional.
- **Ins_Valvula_Sole_AOI:** simula una Válvula Solenoide.
- **P_Entrada_Analog_AOI:** permite manipular Entrada Analógica.
- **P_FT_Primer_Orden_AOI:** simula el comportamiento de una Función de Transferencia de primer orden más tiempo muerto ($G(s) = \frac{Kp}{Tao*s+1}e^{-t*s}$).
- **P_Motor_Variar_Veloc_AOI:** permite manipular Motor al que puede variarse su velocidad.
- **P_Valvula_Prop_AOI:** permite manipular Válvula Proporcional.

- **P_Valvula_Sole_AOI:** permite manipular Válvula Solenoide.

2.1.1.4. Tareas y rutinas

Las tareas y rutinas utilizadas en la solución son:

1. **Control_PI_200ms:** tarea que ejecuta el control de temperatura en *TR_100* y *TK_200*. Esta tarea cuenta con 3 rutinas:

   a) *PI_Main_Routine*: rutina principal donde se realiza el llamado de las subrutinas del programa.

   b) *_00_PI_TR_100*: subrutina que contiene la estructura necesaria para controlar la temperatura en *TR_100*

   c) *_01_PI_TK_200*: subrutina que contiene la estructura necesaria para controlar la temperatura en *TK_200*.

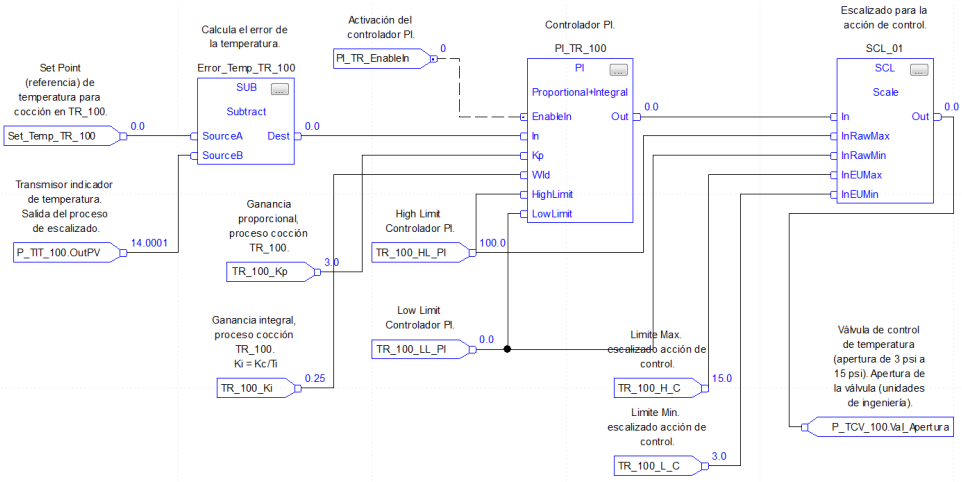
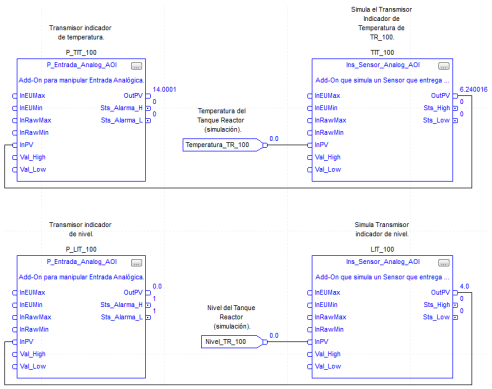
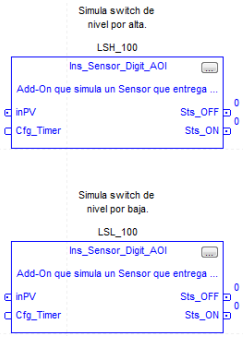


Figura 2.2: Lógica de la rutina *_00_PI_TR_100*.
Fuente propia.

2. **Equipos_250ms:** contiene todas las instancias de los equipos que maneja el PLC. Esta tarea contiene 2 programas, uno con lo referente a *TR_100* y otro a *TK_200*, ambos contienen las mismas rutinas. Las rutinas contienen las instancias de los instrumentos, conexiones y configuración de cada uno.

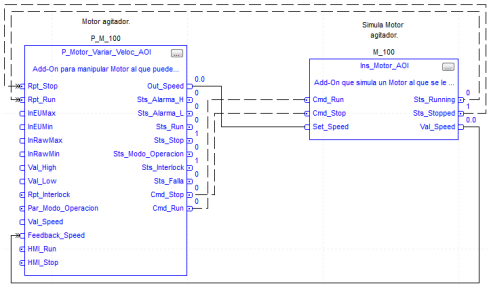


(a) Entradas analógicas de *TR_ 100*.

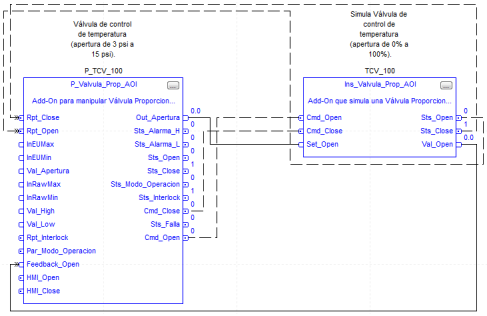


(b) Entradas digitales de *TR_ 100*.

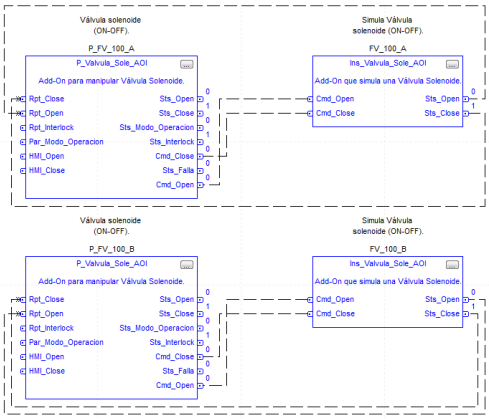
Figura 2.3: Instancias de los transmisores y sensores de *TR_ 100*.
Fuente propia.



(a) Motor de *TR_ 100*.



(b) Válvula de control de *TR_ 100*.



(c) Válvulas solenoides de *TR_ 100*.

Figura 2.4: Instancias de los actuadores de *TR_ 100*.
Fuente propia.

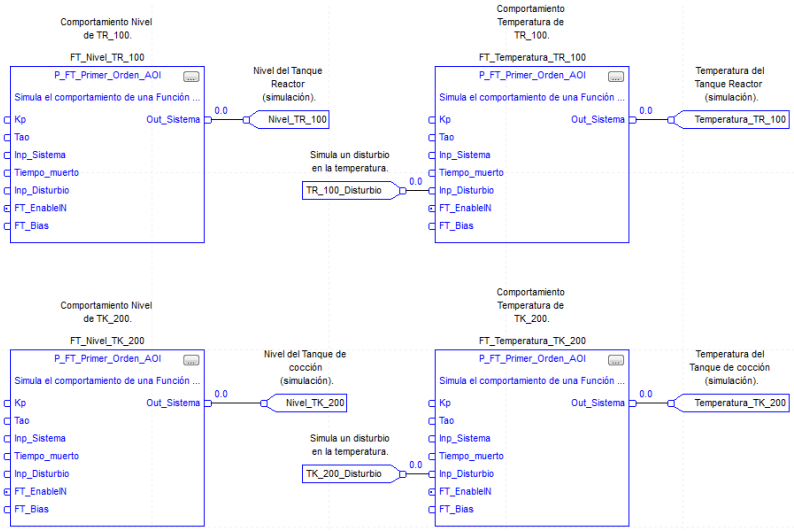
3. **Secuencias_200ms:** contiene las secuencias para llevar a cabo los procesos de macerado y cocción del mosto. Esta tarea contiene 2 programas:

a) *Inicializaciones*: contiene la rutina donde se instancian las funciones de transferencia que se usarán para simular las variables de proceso, figura 2.5a.

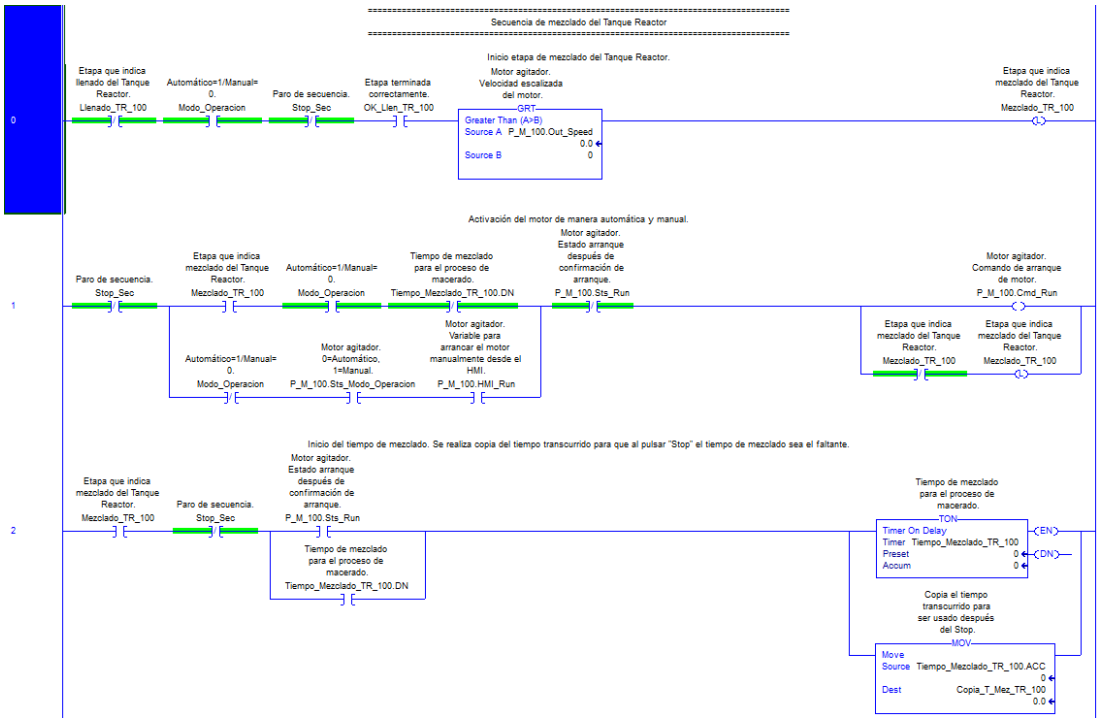
b) *SEC_Macerado_Coccion*: contiene todas las subrutinas que cuentan con la lógica para llevar a cabo el proceso de macerado y cocción, además de los permisivos. Esta tarea cuenta con 9 subrutinas:

1) *SEC_Main_Routine*: rutina principal donde se realiza el llamado de las subrutinas del programa.
2) *SEC_Permisivos_TK_200*: subrutina donde se encuentran las condiciones para iniciar las secuencias del tanque de cocción.
3) *SEC_Permisivos_TR_100*: subrutina donde se encuentran las condiciones para iniciar las secuencias del tanque reactor.
4) *_00_SEC_Llenado_TR_100*: subrutina donde se abren las válvulas de ingreso de materia prima hasta que *LSH_100* o *P_LIT_100* indican un nivel alto, con lo anterior se cierran las válvulas y termina la secuencia de llenado.
5) *_01_SEC_Mezclado_TR_100*: subrutina donde se enciende el motor y se mantiene en este estado hasta alcanzar el tiempo definido según la receta de producción, con lo anterior se apaga el motor y termina la secuencia de mezclado.
6) *_02_SEC_Coccion_TR_100*: subrutina donde se abre la válvula de control y se calienta la mezcla a la temperatura y durante el tiempo definido según la receta de producción, con lo anterior se cierra la válvula y termina la secuencia de cocción.
7) *_03_SEC_Llenado_TK_200*: subrutina donde se abre la válvula de paso y se enciende la motobomba para llevar el líquido de *TR_100* a *TK_200* hasta que *P_LIT_200* indica un nivel de 3 m, entonces se apaga la bomba y se cierra la válvula de paso. Con lo anterior se abre la válvula de ingreso de lúpulo hasta que *P_LIT_200* indica un nivel de 4 m, con lo anterior se cierra la válvula de ingreso de materia prima y termina la etapa de llenado.
8) *_04_SEC_Coccion_TK_200*: subrutina donde se abre la válvula de control y se calienta la mezcla a la temperatura y durante el tiempo definido según la receta de producción, con lo anterior se cierra la válvula y termina la secuencia de cocción.
9) *_05_SEC_Vaciado_TK_200*: subrutina donde se abre la válvula de va-

ciado hasta que *P_LIT_200* indica un nivel bajo, con lo anterior se cierra la válvula y termina la secuencia de vaciado.



(a) Instancias de las funciones de transferencia del proceso.



(b) Porción de la lógica de la rutina de mezclado de *TR_100*.

Figura 2.5: Parte de las rutinas que contiene *Secuencias_200ms*.
Fuente propia.

2.1.2. Propuesta de Supervisión

A continuación se realiza la descripción de los componentes más importantes del HMI propuesto como solución para el caso de estudio y que servirá de guía para continuar con la fase experimental de la tesis. La aplicación fue creada en *FactoryTalk View SE*.

2.1.2.1. Organización de pantallas

La pantalla inicial del sistema de supervisión se compone de 3 secciones que son llamadas al ejecutar la aplicación por medio de una macro de programación. La primera es la barra de navegación, la segunda es el área del despliegue de proceso y la última es el área de alarmas y eventos, lo descrito puede verse la figura 2.6.

Figura 2.6: Áreas del sistema de supervisión
Fuente propia.

2.1.2.2. Barra de herramientas

Como puede observarse en la figura 2.7 la barra de herramientas está dividida en 4 secciones que serán detalladas a continuación:

- **A:** Inicio de sesión.
- **B:** Lenguaje de la aplicación.
- **C:** Pantallas principales de navegación.

- **D:** Fecha.

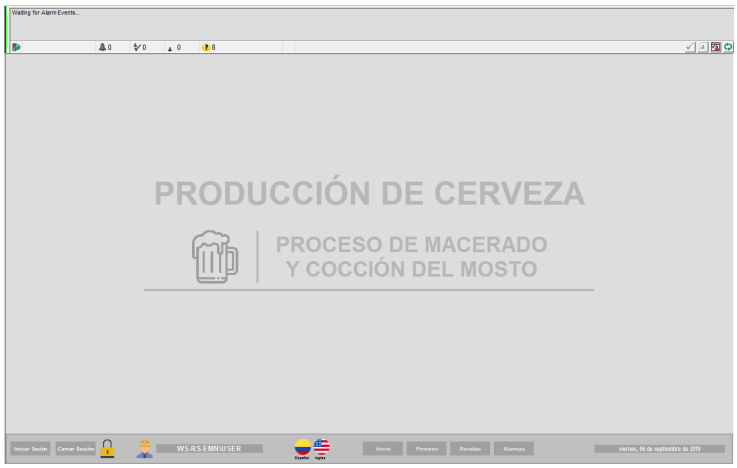
Figura 2.7: Barra de herramientas.
Fuente propia.

2.1.2.3. Inicio de sesión

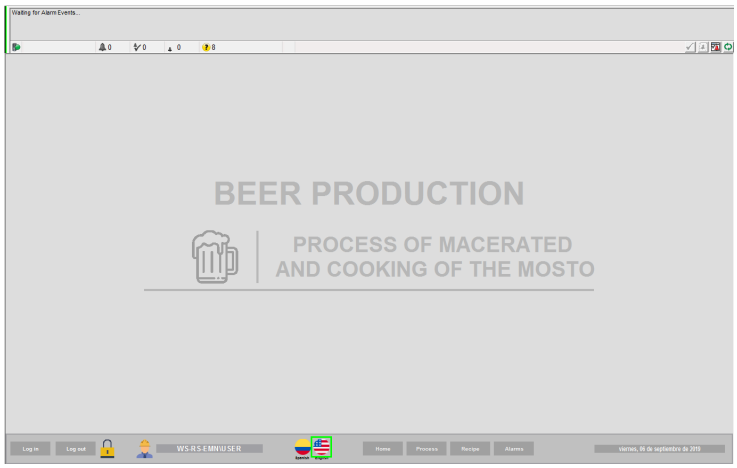
Para el acceso a las diferentes pantallas de la aplicación se debe iniciar sesión previamente, se definen perfiles con respectivos permisos que permiten supervisar el proceso o realizar cambios sobre este, como modificación en los parámetros de la receta o la manipulación de instrumentos, los perfiles tienen definidos un nombre de usuario y una contraseña. Se cuenta con un icono de candado que muestra si alguien ha accedido a la aplicación, además de un pequeño *display* junto a él, que indica el nombre de usuario de quien se encuentre actualmente en la aplicación.

2.1.2.4. Lenguaje de la aplicación

La barra de herramientas cuenta con 2 botones en forma de banderas que permiten realizar el cambio de idioma, los idiomas configurados para la aplicación son: inglés (Estados Unidos) y español (Colombia). Para efectuar el cambio de idioma basta con dar clic sobre la bandera del idioma a seleccionar, en la figura 2.8 se visualiza el cambio de idioma en la pantalla de inicio.



(a) Pantalla de inicio en español.



(b) Pantalla de inicio en inglés.

Figura 2.8: Cambio de idioma en la pantalla de inicio.
Fuente propia.

2.1.2.5. Pantallas de navegación

En la barra de herramientas se tiene diferentes botones que permiten navegar hacia cada uno de los despliegues de proceso configurados para la operación de la aplicación. En esta barra se tienen configurados los siguientes accesos:

1. Inicio: despliegue de la pantalla de inicio de la aplicación.
2. Proceso: despliegue de los procesos de macerado y cocción del proceso de producción de cerveza.

3. Recetas: despliegue de la configuración de parámetros de la receta.

4. Alarmas: despliegue del *banner* de alarmas del proceso.

2.1.2.6. Pantalla de proceso

La pantalla de proceso ofrece una vista general de los equipos que hacen parte de la etapa de macerado y cocción del mosto. Es un diagrama de proceso compuesto por: un reactor, tanque de cocción, válvulas de ingreso de materia prima, válvula de paso del tanque reactor hacia el tanque de cocción, válvula de descarga de materia prima del tanque de cocción, válvulas de control de temperatura para las dos etapas, motor encargado de la mezcla de materia prima y motobomba para el transporte de materia prima. En esta pantalla se visualiza el comportamiento dinámico de las variables de proceso, por medio de tendencias e indicadores y se lleva a cabo la manipulación de los instrumentos, también permite cambiar el proceso a manual o automático según sea el caso. La pantalla cuenta con información del estado y tiempos de las secuencias, el despliegue de proceso se presenta en la figura 2.9.

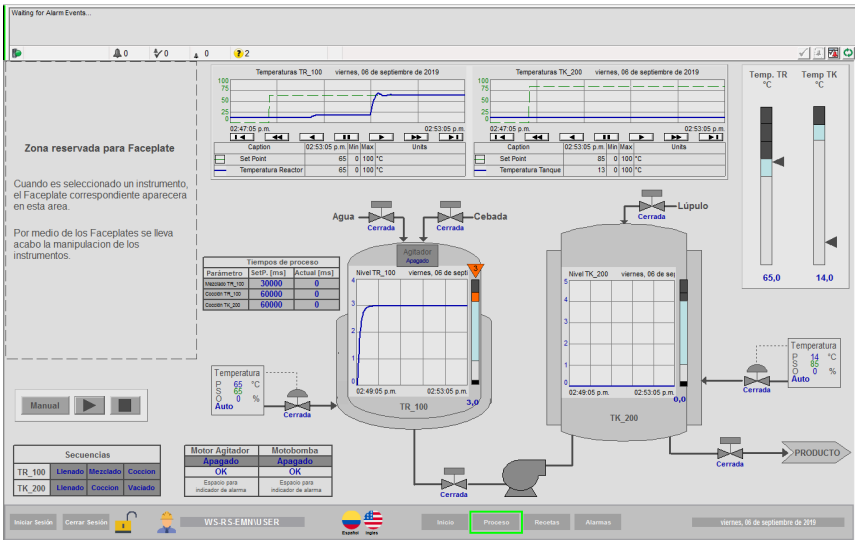


Figura 2.9: Despliegue del proceso.
Fuente propia.

A continuación se muestra la representación del funcionamiento de los instrumentos o dispositivos del proceso.

2.1.2.7. Válvulas

Una válvula que se encuentra cerrada se representa mediante el color gris y una etiqueta de "Cerrada". Una válvula que se encuentra abierta cambia el color del cuerpo de ésta a blanco y la etiqueta cambia a "Abierta".

(a) Válvulas cerradas. (b) Válvulas abiertas.

Figura 2.10: Representación del estado de las válvulas.
Fuente propia.

2.1.2.8. Motores

Los motores al igual que las válvulas cambian de color al cambiar de estado, un motor que está apagado se representa de color gris oscuro y una etiqueta de "Apagado". Un motor que se encuentre en funcionamiento se representa de color blanco y la etiqueta cambia a "Encendido".

(a) Motor apagado. (b) Motor encendido.

Figura 2.11: Representación del estado de los motores.
Fuente propia.

Los motores cuentan con una tabla que confirma el estado en el que se encuentra el motor, esta tabla cuenta con 3 filas, la primera fila confirma si el motor se encuentra encendido o apagado, la segunda fila permite saber el correcto funcionamiento del motor (OK) o si se encuentra en falla ($Falla$), la tercera fila permite visualizar el indicador de alarma si el motor entra en falla.

Figura 2.12: Tabla de resumen del funcionamiento de los motores.
Fuente propia.

2.1.2.9. Controladores

Los controladores son representados por un pequeño cuadro que muestra la siguiente información:

- **Variable del proceso:** se representa con la letra P.
- **Set point:** se representa con la letra S.
- **Esfuerzo de control:** se representa con la letra O.
- **Modo de control:** Automático (AUTO) y manual (MAN)

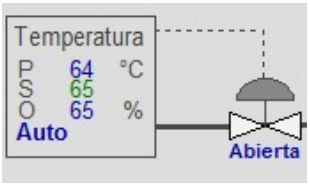


Figura 2.13: Representación de controlador de temperatura.
Fuente propia.

2.1.2.10. Representación de Secuencias

El cuadro de secuencias permite saber en qué secuencia se encuentra el proceso, esto mediante el cambio del color del recuadro de cada etapa, estos cambios de color son los mismos utilizados para representar la apertura o encendido de los instrumentos. En el tanque reactor (*TR_100*) se llevan a cabo 3 secuencias: llenado, mezclado y cocción. En el tanque de cocción (*TK_200*) se llevan a cabo 3 secuencias; llenado, cocción y vaciado.

La figura 2.14 muestra el momento en el que la secuencia de "Cocción" de *TR_100* está activa:

Figura 2.14: Representación de las secuencias del proceso.
Fuente propia.

2.1.2.11. Tiempos de proceso

La tabla de *Tiempos de proceso* permite conocer los tiempos definidos para las etapas de mezclado y cocción, en esta se visualiza el tiempo actual transcurrido hasta llegar al deseado. Cuando el tiempo de una de estas etapas inicia, la fila completa cambia a color a blanco hasta completar el tiempo deseado.

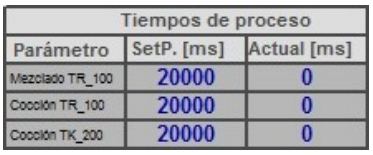


Figura 2.15: Representación de los tiempos de proceso.
Fuente propia.

2.1.2.12. Indicadores

Los indicadores permiten monitorear de una manera más sencilla las variables del proceso mostrando el valor actual por medio de un indicador (flecha) que se desplaza a lo largo de una escala, los rangos definidos para alarmas se encuentran representados con color gris oscuro, si se activa una alarma, éstas cambian de gris a amarillo, rojo o naranja según la prioridad de la alarma, se estableció un color azul claro para el rango de operación deseado de la variable de proceso y un gris claro que indica un rango de operación normal donde no se ha alcanzado el valor deseado pero que tampoco representa una alarma.

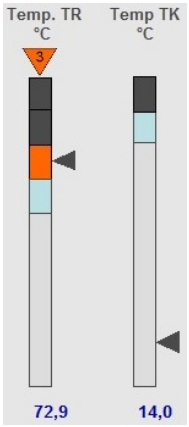


Figura 2.16: Representación de indicadores analógicos de temperatura.
Fuente propia.

2.1.2.13. Faceplates de instrumentos

Los *faceplates* son ventanas *pop up* que son desplegados al hacer clic en instrumentos como válvulas o motores, estos permiten la manipulación directa de los instrumentos. En estas ventanas se pueden realizar cambios en los parámetros que recibe cada instrumento, estos *faceplates* son para uso en modo manual y configuración, por lo tanto, se puede encender, apagar, abrir, cerrar o modificar los valores de configuración del instrumento.

En la pantalla de proceso se reservó una zona para desplegar los *faceplates* sin que oculten información relevante del proceso, esto se muestra en la figura 2.17a.

En la figura 2.17b se muestra un ejemplo de *faceplate* para una válvula de control donde se puede abrir, cerrar o realizar cambios sobre el porcentaje de apertura, límites de apertura y límites de entrada (de acción del instrumento).

(a) Zona reservada para desplegar *faceplates*.

(b) Faceplate válvula de control.

Figura 2.17: Despliegue de *faceplates*.
Fuente propia.

2.1.2.14. Pantalla de Recetas

La pantalla de *Recetas* es una ventana *pop up* que se despliega en la zona reservada para *faceplates*. En esta ventana se pueden realizar cambios sobre los parámetros para la etapa de macerado y cocción, los parámetros que se modifican desde esta ventana son: velocidad y tiempo de mezclado, tiempos y temperaturas de cocción.

Se cuenta con un botón que abre el *RecipePro+* donde se configuran todos los parámetros de los ingredientes de la receta.

El *pop up* cuenta con un botón de *Cargar Receta* el cual carga los cambios realizados en los parámetros de esta ventana a la receta original. El botón *Descargar Receta* sirve para asignar los valores de los parámetros o para volver a los valores predeterminados de la receta original si previamente no se ha "cargado" un nuevo valor.

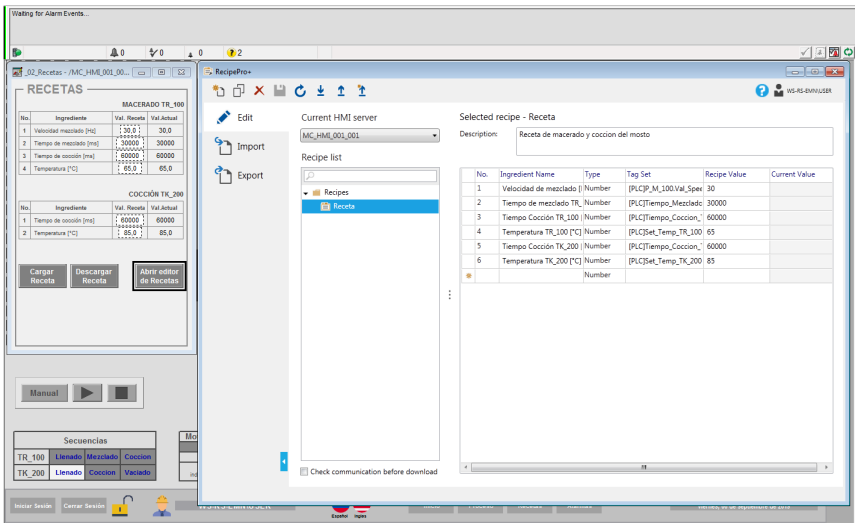


Figura 2.18: Pantalla de recetas.
Fuente propia.

2.1.2.15. Pantalla de Alarmas

En la pantalla de *Alarmas* se despliega el resumen de alarmas y eventos, este es un listado completo con el historial de alarmas o eventos ocurridos, allí se puede ver información detallada, por ejemplo, se especifica un color diferente para cada prioridad de alarma, se pueden realizar filtros, suprimir y reconocer alarmas, entre otras acciones.

Por medio de un *macro* asociado a la aplicación se visualiza un *banner* de alarmas y eventos, este es un indicador ubicado en la parte superior de la aplicación (esta posición se mantiene siempre, no la afecta la navegación entre pantallas) que muestra la alarma más reciente que se presentó, esto se usa para supervisar y reconocer las alarmas críticas del sistema.

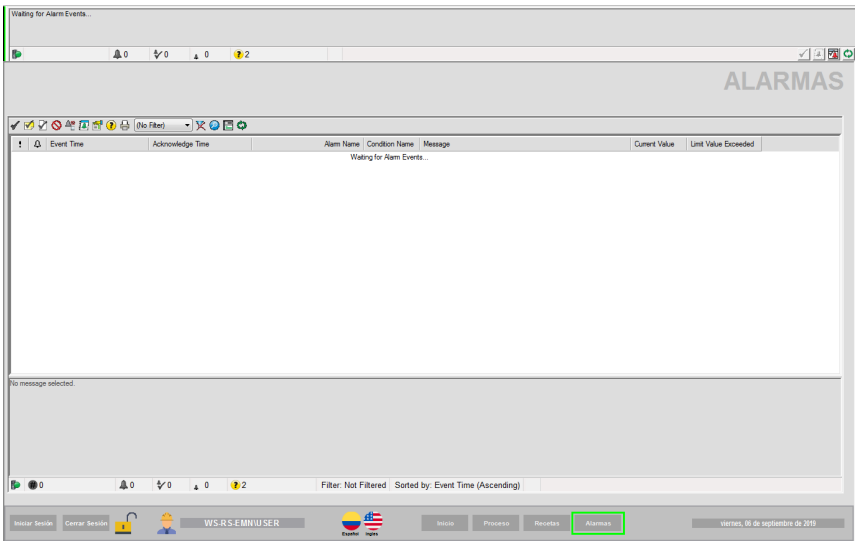


Figura 2.19: Pantalla de resumen de alarmas.
Fuente propia.

2.2. Desarrollo de la solución del caso de estudio mediante las herramientas de PlantPAx

2.2.1. Propuesta de programación

A continuación se realiza la descripción de la estructura del proyecto y lógica del algoritmo propuesto como solución de programación para el caso de estudio utilizando las herramientas de *PlantPAx*.

2.2.1.1. Estructura del proyecto

El proyecto cuenta con la misma estructura descrita en la sección 2.1.1.1, la diferencia radica en el uso de las instrucciones *Add-On* proporcionadas por la biblioteca de objetos de proceso de *PlantPAx* versión *3.50.09*. En la figura 2.20 se observan las instrucciones utilizadas.

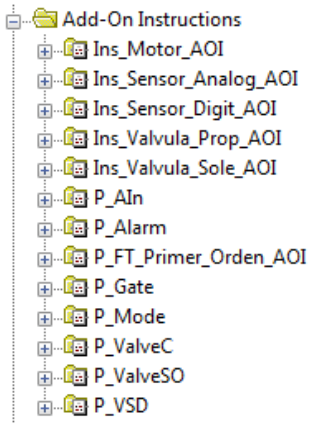


Figura 2.20: Instrucciones *Add-On* utilizadas en la solución.
Fuente propia.

2.2.1.2. Estructura del código

La solución utilizando las herramientas de *PlantPAx* cuenta con la misma estructura definida en la sección 2.1.1.1. Los cambios se ven reflejados en la lógica de las rutinas ya que en estas se instancian y utilizan las nuevas instrucciones *Add-On*.

En la programación del algoritmo con las instrucciones *Add-On* de la biblioteca de *PlantPAx* es posible observar una reducción significativa en el código necesario, además de un número mucho más amplio de parámetros y características que pueden ser utilizadas en al algoritmo, esto debido a que estas instrucciones cuentan con modos de operación, alarmas, estados de falla, entre otras variables que podrían tenerse en cuenta para programar rutinas de falla, solucionar problemas de mantenimiento, entrenar personal, entre otras posibilidades.

2.2.1.3. Instrucciones Add-On

En esta propuesta de programación se cuenta con las mismas instrucciones que permiten modelar el comportamiento de los instrumentos, las nuevas instrucciones son las utilizadas para manipular los instrumentos y obtener información de estos. A continuación se describen las instrucciones utilizadas:

- **P_AIn:** permite monitorear un valor analógico, escalizar el valor de entrada a unidades de ingeniería, además proporciona alarmas cuando el valor analógico excede

límites especificados por el usuario [55]. Esta instrucción requiere las instrucciones *P_Alarm*, *P_Gate*, *P_Mode*, para funcionar, por ello las instrucciones mencionadas son cargadas en el momento de importar la instrucción *P_AIn*, de igual manera las otras instrucciones de *PlantPAx* usadas en el proyecto las requieren.

- **P_ValveC:** permite manipular una válvula de control haciendo uso de una señal analógica o señales discretas, además permite monitorear la válvula mediante la retroalimentación de su posición [56].
- **P_ValveSO:** permite operar (abrir y cerrar) una válvula solenoide en varios modos, además permite monitorear condiciones de falla [57].
- **P_VSD:** permite operar, en varios modos, un motor cuya velocidad es variable, además permite monitorear condiciones de falla [58].

2.2.1.4. Tareas y rutinas

Las tareas y rutinas utilizadas en la solución son las descritas en la sección 2.1.1.4 pero con cambios en cuanto a las instancias y conexiones realizadas. A continuación se muestran los cambios realizados en las rutinas de las tareas debido al uso de las instrucciones *Add-On* de *PlantPAx*.

1. **Control_PI_200ms:**

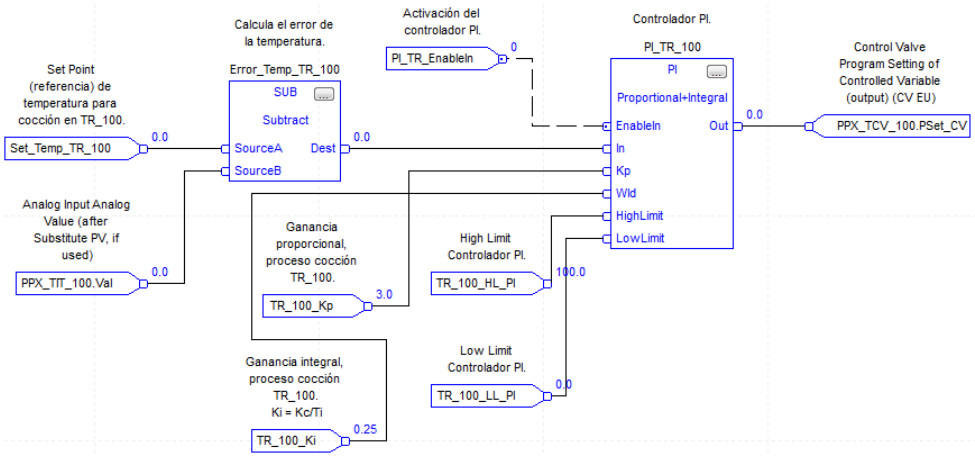
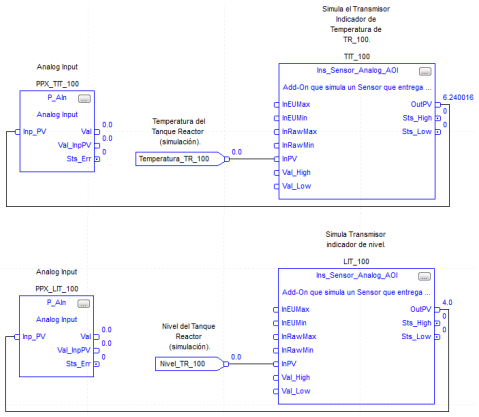
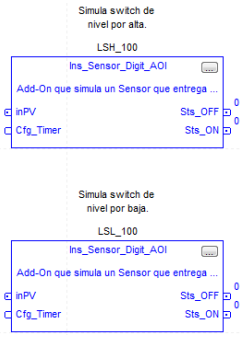


Figura 2.21: Lógica de la rutina *_00_PI_TR_100*.
Fuente propia.

2. **Equipos_250ms:**

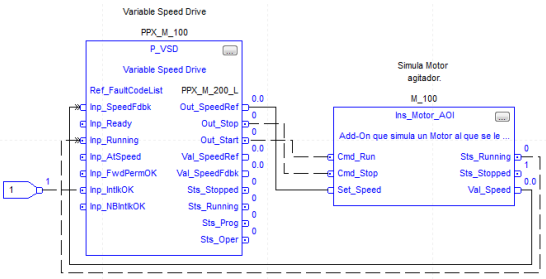


(a) Entradas analógicas de *TR_100*.

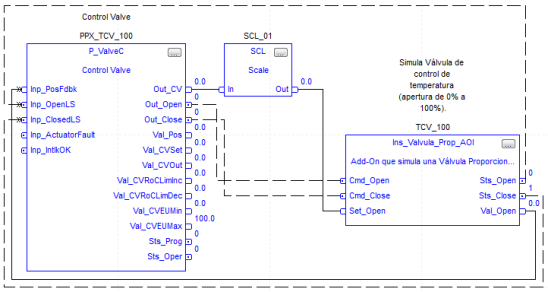


(b) Entradas digitales de *TR_100*.

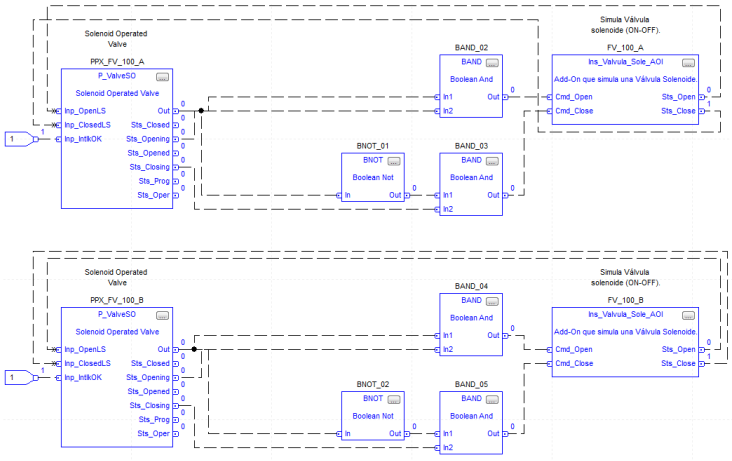
Figura 2.22: Instancias de los transmisores y sensores de *TR_100*.
Fuente propia.



(a) Motor de *TR_100*.



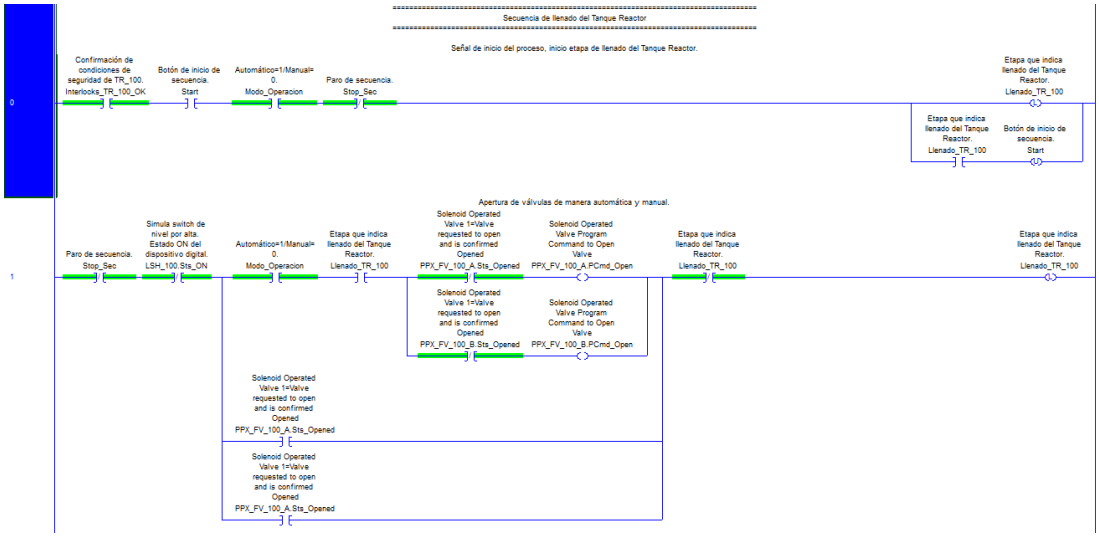
(b) Válvula de control de *TR_100*.



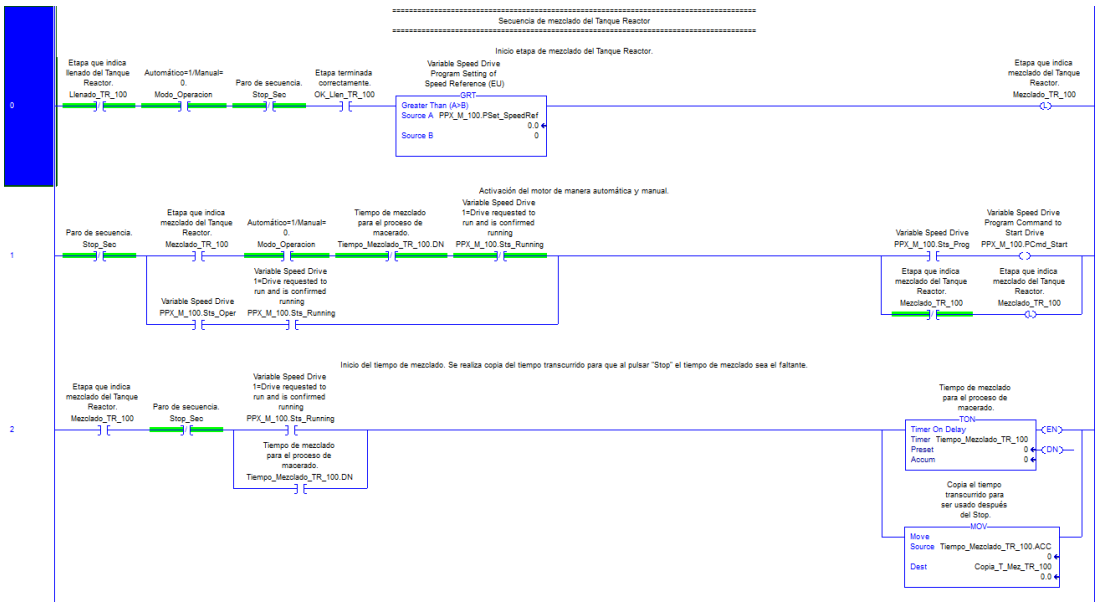
(c) Válvulas solenoides de *TR_100*.

Figura 2.23: Instancias de los actuadores de *TR_100*.
Fuente propia.

3. **Secuencias_200ms:**



(a) Porción de la lógica de la rutina de llenado de *TR_100*.



(b) Porción de la lógica de la rutina de mezclado de *TR_100*.

Figura 2.24: Parte de las rutinas que contiene *Secuencias_200ms*.
Fuente propia.

2.2.2. Propuesta de supervisión

A continuación se realiza la descripción de la pantalla de proceso que contiene los objetos globales y *faceplates* de la biblioteca de *PlantPAx* versión *3.50.09* que fueron utilizados en la aplicación, esta cuenta con la misma estructura descrita en la sección 2.1.2, por lo tanto en esta sección solo se describirá la pantalla de proceso.

2.2.2.1. Pantalla de proceso

En la pantalla de proceso ofrece una vista general de los equipos que hacen parte del proceso como se menciona en la sección 2.1.2.6, los cambios al implementar la biblioteca de objetos de *PlantPAx* se ven reflejados en la disminución de trabajo y tiempo que lleva diseñar los objetos globales y sus respectivos *faceplates*, ya que a diferencia del método convencional descrito en la sección 2.1.2 estos objetos ya se encuentran diseñados y cuentan con funcionalidades más avanzadas, además de estar previamente parametrizados, cada parámetro del objeto gráfico de la biblioteca de *PlantPAx* se encuentra enlazado a los parámetros definidos en las instrucciones *Add-On* contando con funcionalidades previamente diseñadas como los modos de operación, alarmas, estados de falla entre otras variables que pueden ser manipuladas y supervisadas desde la interfaz. El despliegue de proceso se presenta en la figura 2.25.

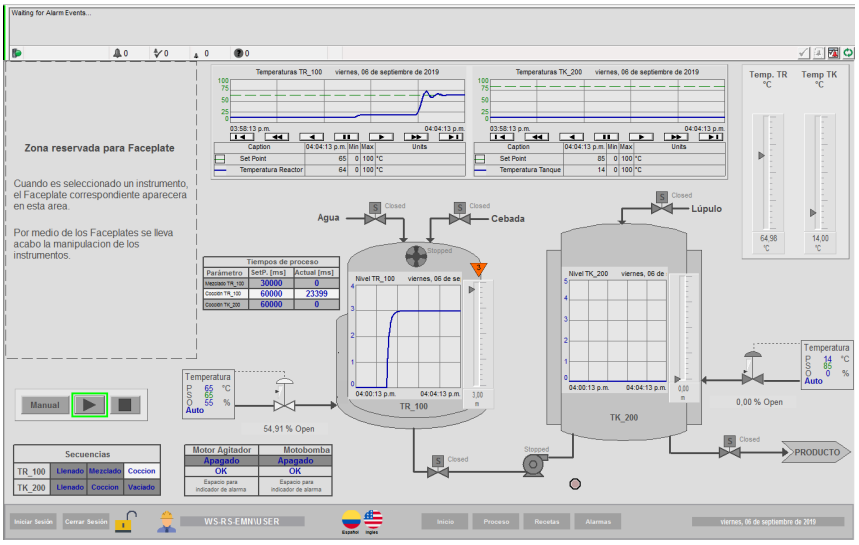
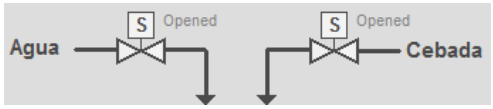


Figura 2.25: Despliegue del proceso.
Fuente propia.

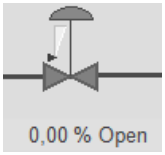
2.2.2.2. Válvulas

Las válvulas de la biblioteca de objetos ya se encuentran configuradas previamente con animaciones que representan el estado de la válvula, de tal forma que si el objeto se encuentra en una tonalidad de color gris oscura se encuentra cerrada y una válvula que se encuentre abierta cambia a color blanco, los objetos vienen configurados con un *display* de texto que permite dar una realimentación del estado y en el caso de las válvulas de control cuentan con un *display* numérico que indica el porcentaje de apertura actual de la válvula. En la figura 2.26 se ilustra la representación del estado de las válvulas de control y solenoides.

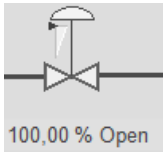
(a) Válvulas solenoides cerradas.



(b) Válvulas solenoides abiertas.



(c) Válvula de control cerrada.



(d) Válvula de control abierta.

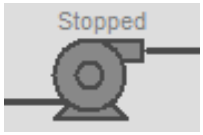
Figura 2.26: Representación del estado de las válvulas.
Fuente propia.

2.2.2.3. Motores

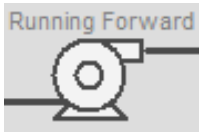
La biblioteca de objetos de *PlantPAx* cuenta con una variedad de motores, para la aplicación se eligió el objeto *P_ VSD* el cual dentro de su biblioteca cuenta con varios objetos gráficos de motores, de la biblioteca de *P_ VSD* se eligió el objeto *mixer* para representar el motor encargado de la mezcla del tanque reactor y una *motobomba* para el transporte de material de un tanque a otro. Los motores al igual que las válvulas cambian de color al cambiar de estado, un motor que este apagado se representa de color gris oscuro y un motor que se encuentre encendido se representa de color blanco, cuentan con una realimentación de estado mediante un *display* de texto.

(a) Motor mezclador apagado.

(b) Motor mezclador encendido.



(c) Motobomba apagada.



(d) Motobomba encendida.

Figura 2.27: Representación del estado de los motores.
Fuente propia.

2.2.2.4. Indicadores

Los indicadores permiten monitorear de una forma más sencilla las variables del proceso, mostrando el valor actual por medio de un indicador que se desplaza a lo largo de una escala. Para los indicadores se elige el objeto *P_Ain* de la biblioteca de objetos, este objeto representa una entrada analógica. El objeto viene con el diseño de la escala, *display* numérico y de texto para la mostrar el valor actual de la variable de proceso.

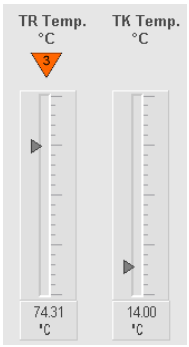


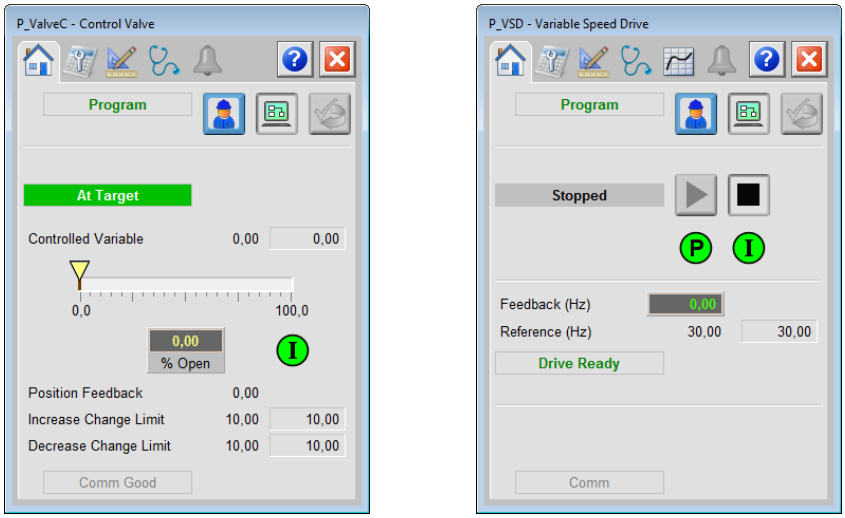
Figura 2.28: Representación de indicadores analógicos de temperatura.
Fuente propia.

2.2.2.5. Faceplates de instrumentos

Los *faceplates* de *PlantPAx* vienen diseñados con pestañas para navegar a través de las distintas opciones, dentro de las pestañas se pueden encontrar funciones para cambiar el

modo de operación del instrumento, información sobre el valor actual de la variable del instrumento, realizar mantenimiento, modificar el escalizado, información sobre diagnósticos, alarmas, tendencias, entre otras opciones.

Al igual que en la sección 2.1.2.13 en la figura 2.29 se muestra un ejemplo de los *faceplates* de la biblioteca de objetos referentes a una válvula de control y un motor.



(a) *Faceplate* válvula de control. (b) *Faceplate* motor.

Figura 2.29: Despliegue de *faceplates*.
Fuente propia.

Capítulo 3

Experimentación y resultados

En este capítulo se presenta el protocolo del experimento realizado, el cual consistió en desarrollar sesiones previas para la conceptualización del método definido en el capítulo 1. Los participantes fueron divididos en dos grupos, denominados: *Grupo I: Método Convencional (MC)* y *Grupo II: PlantPAx (PPX)*, los cuales en la evaluación del caso de estudio implementaron conceptos del método definido, obteniendo así dos soluciones, una mediante el método convencional y la otra haciendo uso de las herramientas de *PlantPAx*. En las soluciones fue tomado como indicador de desempeño el tiempo de desarrollo, posteriormente se realizó un análisis de resultados con los tiempos obtenidos de ambos grupos para la programación y para el diseño del HMI, el análisis se realizó con técnicas estadísticas ampliamente utilizadas.

3.1. Experimentación

3.1.1. Selección de participantes

Para obtener los participantes del experimento se realizó una convocatoria a estudiantes de la *Universidad del Cauca* del programa de *Ingeniería en Automática Industrial*, los requisitos fueron contar con conocimientos en programación de PLCs y conceptos básicos para el diseño de HMI. Con los estudiantes que atendieron la convocatoria se crearon dos grupos, el primer grupo fue denominado *Grupo I: Método Convencional (MC)* y el segundo *Grupo II: PlantPAx (PPX)*. Los estudiantes fueron asignados a cada grupo de

manera aleatoria, para esto se seleccionó el participante y se arrojó una moneda, de acuerdo a la cara resultante el participante se asignó al *grupo I* o *grupo II*, lo anterior se repitió 22 veces hasta conseguir conformar dos grupos cada uno con 11 participantes.

3.1.2. Implementación del método

Como fue mencionado, el caso de estudio a desarrollar se encuentra basado en el proceso de producción de cerveza, específicamente en 2 etapas: macerado y cocción. Los participantes del primer grupo desarrollaron una propuesta de solución de programación y supervisión para el caso de estudio mediante el método convencional y el otro lo desarrolló haciendo uso de las herramientas de *PlantPAx*. Los tiempos para desarrollar cada una de las propuestas (programación y supervisión con los 2 métodos) fueron registrados para realizar un análisis estadístico.

Para obtener las distintas soluciones al caso de estudio los estudiantes debían conocer previamente algunas aplicaciones y conceptos, para esto se propuso una serie de sesiones y ejercicios aplicados para familiarizar a los participantes con las aplicaciones y los conceptos necesarios para proponer una solución utilizando el método convencional y las herramientas de *PlantPAx*. Las sesiones fueron llevadas a cabo mediante presentaciones con diapositivas, trabajando simultáneamente con las herramientas de las aplicaciones y realizando ejercicios prácticos. A continuación se describen brevemente las sesiones.

3.1.2.1. Sesión 1: Introducción a las aplicaciones necesarias

La sesión inicial buscó familiarizar a los participantes con las aplicaciones y funciones básicas que serían usadas durante las sesiones y en el desarrollo de la prueba.

La introducción inició con generalidades sobre *Studio 5000*, esto fue necesario para llegar a la aplicación que se utilizaría para programar durante la prueba, dicha aplicación fue *Studio 5000 Logix Designer* y también para conocer la aplicación que permitiría simular el PLC, la cual fue *Logix Emulate*.

En la introducción a *Logix Designer* se abordaron generalidades sobre componentes fundamentales para programar un algoritmo, tales como: tareas, rutinas e instrucciones *Add-On*.

Para el diseño del HMI se utilizó la aplicación *FactoryTalk View SE*, por ello se realizó una breve exposición de las herramientas de las aplicaciones para posteriormente abordar conceptos fundamentales como: objetos globales, *pop-up* y *faceplates*.

Taller de habilidades y asignación en grupos de control

Se realizó un taller para evaluar la capacidad o el desempeño de los participantes en la programación de un algoritmo y el diseño de un HMI, el taller requirió que los participantes elaboraran una solución por el método convencional, sin ningún tipo de condición respecto al diseño del algoritmo o del HMI, como criterios de evaluación se tuvieron:

1. Destreza en el desarrollo del algoritmo de la propuesta de programación.
2. Destreza en el diseño de una interfaz gráfica (HMI).
3. Funcionalidad de la solución obtenida.

De acuerdo a los resultados obtenidos se analizó si los grupos creados anteriormente se encontraban equilibrados, el resultado fue que no, en el *grupo II* se encontraba un número mayor de participantes con mejores habilidades según la prueba, por consiguiente, utilizando los resultados de la evaluación se seleccionaran los participantes con mejores resultados y de manera aleatoria se asignaron nuevamente a un grupo con el objetivo de equilibrar los grupos en cuanto a participantes con conocimientos en las áreas descritas.

3.1.2.2. Sesión 2: Profundización de conceptos de programación

La sesión 2 tuvo como propósito profundizar en la creación y uso de tareas y rutinas, reutilización de código por medio de instrucciones *Add-On*, todo lo anterior mediante un ejercicio práctico. A través de un ejercicio que contenía características típicas de un proceso industrial se buscó integrar los conceptos mencionados, además, la solución del ejercicio se desarrolló utilizando conceptos de buenas prácticas de programación.

3.1.2.3. Sesión 3: Profundización de conceptos de diseño de HMI

La sesión 3 tuvo como propósito profundizar en la creación y uso de *displays*, *pop-ups*, *faceplates*, objetos globales, todo lo anterior mediante un ejercicio práctico. A través de un ejercicio que contenía características típicas de un proceso industrial se buscó integrar los conceptos mencionados, además, la solución del ejercicio se desarrolló utilizando conceptos de buenas prácticas de diseño de HMI de alto rendimiento.

3.1.2.4. Sesión 4: Introducción a PlantPAx (biblioteca de objetos de proceso - Add-On)

La sesión 4 tuvo como propósito exponer las generalidades de *PlantPAx* y enfatizar en las instrucciones *Add-On* de la biblioteca de objetos de proceso. Aquí se expuso a los participantes las instrucciones disponibles, sus componentes y la configuración básica de las más comunes, todo lo anterior mediante un ejercicio práctico donde se buscó utilizar los objetos de la biblioteca de proceso en un algoritmo desarrollado considerando conceptos de buenas prácticas de programación.

3.1.2.5. Sesión 5: Introducción a PlantPAx (biblioteca de objetos de proceso - Gráficos)

La sesión 5 tuvo como propósito estudiar los objetos gráficos de la biblioteca de objetos de proceso. Aquí se expuso a los participantes los objetos gráficos disponibles, sus componentes y la configuración básica de los más comunes, todo lo anterior mediante un ejercicio práctico donde se buscó utilizar los objetos gráficos en un HMI sencillo desarrollado considerando conceptos de diseño de HMI de alto rendimiento.

3.1.3. Evaluación del caso de estudio

La evaluación del caso de estudio tuvo como objetivo medir los tiempos de desarrollo de una solución de programación y supervisión por el método convencional y con las herramientas de *PlantPAx*. Para llevar a cabo esta evaluación se requirió la integración de conocimientos sobre contenidos específicos estudiados en las sesiones de profundización de conceptos vistas previamente, así como las destrezas y capacidades de los participantes

para lograr ofrecer una solución correcta al caso de estudio.

Todos los participantes trabajaron sobre el mismo caso de estudio, sin embargo, los participantes del *grupo I (MC)* desarrollaron una solución de programación y supervisión mediante el método convencional y los del *grupo II (PPX)* desarrollaron una solución utilizando las herramientas de *PlantPAx*. Ambos grupos diseñaron las soluciones a partir de *templates* que fueron creados para el experimento, lo anterior para evaluar específicamente el uso de los conceptos vistos en las sesiones estudiadas y no el desarrollo del algoritmo o configuración del proyecto o de la interfaz. Como criterios de evaluación se tuvieron:

1. Aplicación de conceptos estudiados en las sesiones previas.
2. Funcionalidad de la solución obtenida.
3. Tiempo de desarrollo de la solución de programación y supervisión

3.2. Análisis de resultados

A continuación se desarrollará el análisis estadístico para los datos de tiempo recolectados para la programación y diseño de HMI utilizando el método convencional (MC) y haciendo uso de las herramientas de *PlantPAx* (PPX) resultado del experimento. Para esto se realizará un análisis descriptivo univariado con el fin de conocer a fondo la naturaleza de los datos y poder obtener conclusiones que permitirán su fácil interpretación [59].

Cabe destacar que estos resultados solo describirán el comportamiento para la muestra de los participantes del experimento, sin embargo, es necesario rectificar que estos resultados sean válidos para cualquier usuario; por esto se aplicarán diferentes técnicas inferenciales para hacer una comparación más profunda y obtener resultados más contundentes respecto a los tiempos de programación, diseño de HMI y tiempo total.

A continuación se muestran los tiempos obtenidos en el experimento.

	Participante	Grupo	Tiempo de programación [min]	Tiempo de diseño de HMI [min]	Total [min]
1	Participante 1	MC	146	94	240
2	Participante 2	MC	160	110	270
3	Participante 3	MC	162	105	267
4	Participante 4	MC	150	83	233
5	Participante 5	MC	173	112	285
6	Participante 6	MC	170	80	250
7	Participante 7	MC	176	92	268
8	Participante 8	MC	131	112	243
9	Participante 9	MC	142	103	245
10	Participante 10	MC	177	79	256
11	Participante 11	MC	135	109	244
12	Participante 12	PPX	130	68	198
13	Participante 13	PPX	127	66	193
14	Participante 14	PPX	134	70	204
15	Participante 15	PPX	155	64	219
16	Participante 16	PPX	140	86	226
17	Participante 17	PPX	192	73	265
18	Participante 18	PPX	165	68	233
19	Participante 19	PPX	129	68	197
20	Participante 20	PPX	122	62	184
21	Participante 21	PPX	132	83	215
22	Participante 22	PPX	156	68	224

Tabla 3.1: Tiempos obtenidos mediante el experimento desarrollado.

3.2.1. Análisis descriptivo

Para el análisis estadístico se tendrán en cuenta únicamente las variables de tiempo calculadas en minutos para poder trabajar con la misma unidad de medida. Se ha decidido renombrar estas variables con el fin de facilitar su uso en el análisis, la tabla 3.2 muestra cómo serán nombradas las variables:

Variable	Etiqueta
Tiempo de programación MC [min]	progMC
Tiempo de diseño de HMI [min]	hmiMC
Tiempo total MC [min]	totMC
Tiempo de programación PPX [min]	progPPX
Tiempo de diseño de HMI PPX [min]	hmiPPX
Tiempo total PPX [min]	totPPX

Tabla 3.2: Etiquetas de variables.

3.2.1.1. Análisis de medidas de tendencia central y de dispersión

Las medidas de tendencia central y de dispersión permiten resumir el comportamiento de los datos para facilitar su interpretación [60, 61]. Calculando y analizando la desviación

estándar de cada variable se podrá concluir respecto a la dispersión de los datos, la cual es una característica fundamental a la hora de acotar el posible comportamiento de la variable. Para este caso se considera pertinente calcular el valor mínimo, el valor máximo, la media y la desviación estándar de cada variable, esto es presentado en la tabla 3.3.

Variable	**Mínimo [min]**	**Media [min]**	**Máximo [min]**	**Desviación estándar [min]**
progMC	131	156,5	177	16,6515
hmiMC	79	98,09	112	13,0112
totMC	233	254,6	285	15,9265
progPPX	122	143,8	192	21,1273
hmiPPX	62	70,55	86	7,5015
totPPX	184	214,4	265	22,8398

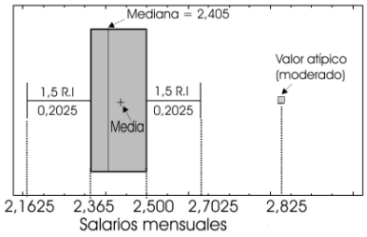
Tabla 3.3: Medidas de tendencia central y de dispersión.

Analizando las medidas de *mínimo*, *media* y *máximo* se puede observar que los tiempos obtenidos con *PPX* son más bajos que los de *MC*.

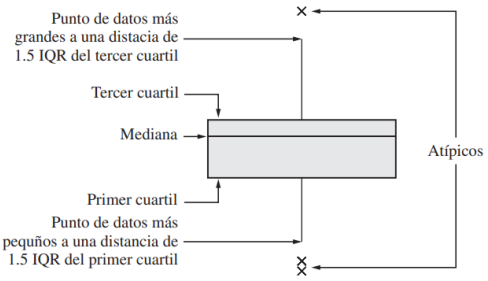
Analizando la *desviación estándar* de la variable *tiempo total* se puede afirmar que *MC* tiene una menor variabilidad que *PPX* para la muestra dada, lo que significa una ventaja estadística pues permite suponer más certeza en el posible comportamiento de la variable, es decir, existe mayor probabilidad de que *totMC* sea cercano a su media a que ocurra lo mismo en la variable *totPPX*.

3.2.1.2. Análisis del diagrama de caja y bigotes

El diagrama de caja y bigotes es una herramienta de análisis de datos que permite estudiar mediante un gráfico la simetría y dispersión de los datos [59, 62]. Al ser una representación gráfica permite que la interpretación de los datos sea más fácil y práctica que la lectura de valores numéricos [60]. *"Este gráfico divide los datos en 4 áreas de igual frecuencia, una caja central dividida en 2 áreas por una línea vertical y otras dos áreas representadas por 2 segmentos horizontales (bigotes) que parten desde el centro de cada lado de la caja"* [59]. La caja central concentra el 50 % de los datos, la línea que divide la caja central representa la mediana, si esta se encuentra en el centro de la caja significa que no existe asimetría en los datos [59, 60]. Los bigotes de los extremos de la caja encierran el 95 % de los datos [60]. La figura 3.1 muestra los componentes de un diagrama de caja y bigotes.



(a) Representación I.



(b) Representación II.

Figura 3.1: Anatomía de un diagrama de caja y bigotes.
Tomado de [59] y [62].

Algunos conceptos necesarios para interpretar de manera correcta los resultados mostrados de esta sección son:

1. **Cuartiles**: son medidas de posición relativa (Q_1, Q_2 *y* Q_3) que dividen en 4 grupos de igual tamaño un conjunto ordenado de datos. Los cuartiles se definen así [59, 60]:

   - Q_1 *(primer cuartil):* está dado por el valor en el cual el 25 % de los datos sea igual o menor a este.
   - Q_2 *(segundo cuartil):* es la mediana, separa el 50 % inferior de los valores ordenados del 50 % superior.
   - Q_3 *(tercer cuartil):* separa el 75 % inferior del 25 % superior.

2. **Rango intercuartílico (RI):** es la diferencia entre el tercer y primer cuartil. El rango intercuartílico contiene el 50 % de los datos, dejando a la izquierda el 25 % inferior y a la derecha el 25 % superior de los datos [59, 62]. Este valor será la altura de la caja del diagrama de caja y bigotes, además es el valor que permitirá brindar una idea sobre la dispersión de los datos alrededor de la mediana, por lo tanto, entre más grande sea nuestra caja más dispersos serán los valores en nuestra variable de estudio. Por otro lado, el RI es la medida empleada en la elaboración de los bigotes cuando existen datos atípicos, los cuales están dados por $3/2$ del valor total del RI, en caso contrario el primer bigote irá desde el valor mínimo y el segundo irá hasta el valor máximo. En caso de que existan datos atípicos estos irán por fuera de los bigotes [62].

3. **Mediana:** es el dato que ocupa la posición central después de haber ordenado todo el conjunto de datos. Para hallar su valor es necesario organizar los datos en orden ascendente y seleccionar el valor que permita dividir el conjunto de datos en dos grupos iguales; de lo que se puede concluir que el valor de la mediana coincide con el valor del segundo cuartil [59, 60, 62].

4. **Diagrama de caja comparativos:** es un diagrama que permite de manera eficaz realizar una comparación visual de las características de dos o más conjuntos de datos [59, 62].

En el caso de este experimento es pertinente hacer tres diagramas de caja comparativos, en el primero se mostrarán los dos diagramas de cajas y bigotes correspondientes al tiempo de programación de los 2 métodos (*MC* y *PPX*), el segundo corresponde al tiempo de diseño de la interfaz gráfica (HMI) y el tercero corresponde al tiempo total; de esta forma se podrán comparar directamente el comportamiento de los métodos estudiados.

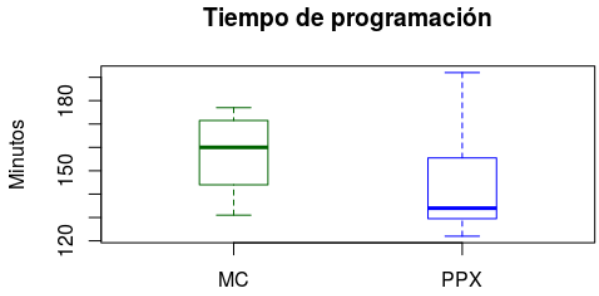


Figura 3.2: Diagrama de caja comparativos: tiempos de programación.
Fuente propia.

De la figura 3.2 se observa una mayor dispersión en los tiempos de programación de *PPX*, también se muestra una asimetría en los datos dado que el 50 % de los valores se encuentran en la zona inferior del diagrama de caja y bigotes. Sin embargo, cabe resaltar que los valores de *PPX* son menores, lo que permite suponer la existencia de una ventaja en términos de menor tiempo requerido respecto a *MC*.

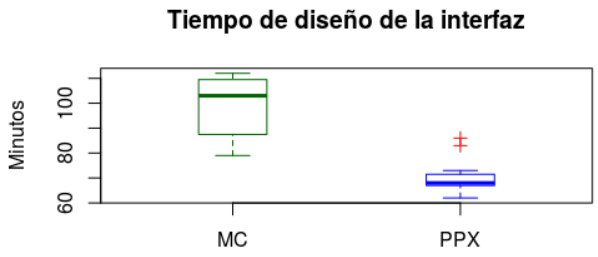


Figura 3.3: Diagrama de caja comparativos: tiempos de diseño de HMI.
Fuente propia.

De la figura 3.3 se observa una mayor dispersión en los tiempos de diseño de HMI en *MC*, sin embargo, cabe resaltar que en *PPX* se presentan dos datos atípicos, lo que significa la posibilidad de obtener tiempos anormales en el diseño de la interfaz.

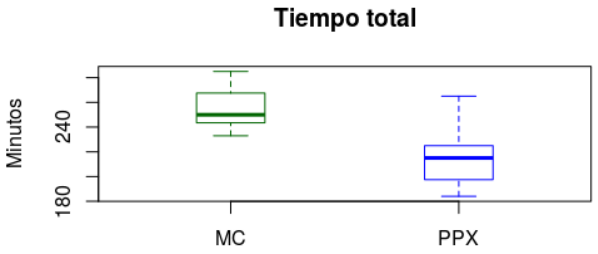


Figura 3.4: Diagrama de caja comparativos: tiempo total.
Fuente propia.

Se puede inferir que la dispersión de los datos obtenidos para el tiempo de programación y diseño de HMI por medio de *PPX* se debe a que los participantes que realizaron la evaluación del caso de estudio en el experimento tuvieron una curva de aprendizaje más lenta en el manejo de estas herramientas, además de no tener conocimientos previos más que los dictados en las sesiones realizadas antes de la evaluación. Lo anterior da como consecuencia una diferencia entre los datos obtenidos.

En conclusión, respecto a los tiempos totales registrados por cada método se puede observar que, aunque los tiempos de *PPX* son menores a los de *MC*, estos presentan una gran dispersión, lo que genera incertidumbre en el momento de imaginar un posible comportamiento general.

Por lo anterior se cree pertinente recurrir a las técnicas inferenciales con el fin de saber si efectivamente es correcto afirmar que el método *PPX* implica un menor tiempo de

desarrollo que *MC*.

3.2.2. Análisis inferencial

La *estadística inferencial* es la parte de la estadística utilizada para concluir o realizar inferencias acerca de las características de una población a partir del estudio de una muestra de la misma [60, 63, 64]. Comprende los métodos y conjuntos de técnicas que se utilizan para concluir sobre las leyes de comportamiento de una población [59, 60]. Los objetivos principales de la estadística inferencial son la estimación y el contraste de hipótesis [60, 64].

El análisis inferencial es de vital importancia en cualquier experimento de comparación de poblaciones, pues si bien el análisis descriptivo proporciona ciertas características relevantes, este análisis impide tener certeza estadística para obtener conclusiones generales sobre una población.

A continuación se ha decidido comparar los dos métodos realizando una prueba de hipótesis comparativa por cada variable de estudio, es decir, una prueba para la variable *tiempo de programación*, una para el *tiempo de diseño de HMI* y, si es necesario, otra para el *tiempo total*. Los resultados que arrojen las tres pruebas de hipótesis permitirán decidir si estadísticamente un método es significativamente más rápido que el otro en términos de tiempo.

Contrastes o pruebas de hipótesis

Hacen referencia a una prueba que mediante el uso de diferentes herramientas estadísticas permiten probar si una afirmación es estadísticamente correcta [60, 62, 63].

3.2.2.1. Prueba de hipótesis para el tiempo de programación

La primera prueba de hipótesis que se desarrollará será una prueba que permita confirmar los resultados obtenidos en el análisis descriptivo respecto al tiempo de programación, es decir, se verificará si efectivamente el uso de las herramientas de *PlantPAx* permite realizar una programación más rápida que el método convencional.

Para lo anterior se propone realizar una comparación de medias de cada variable, de este modo, si se prueba que el tiempo promedio de programación de *PPX* es menor al de *MC*, se podrá concluir que esta es, en general, más rápida.

Cuando se busca realizar una comparación de medias normalmente se recurre a la prueba *t-Student* [65, 66], esta prueba cuenta con diferentes variaciones como la prueba *t-Student* para muestras relacionadas o con diferentes varianzas [66], sin embargo, para este estudio se utilizará la prueba prueba *t-Student* clásica la cual requiere que los datos a analizar cumplan con ciertas condiciones para que los resultados que arroje sean válidos. Estas condiciones son [67]:

- Las muestras que se comparan son independientes entre sí.
- Las muestras provienen de una distribución normal.
- Las varianzas de las muestras son iguales.

Planteamiento de la prueba

Se procede a verificar los supuestos necesarios para aplicar la prueba *t-Student* en la comparación de la media de los tiempos de programación:

1. **Independencia de muestras:** en el experimento se realizó un muestreo aleatorio en el que 11 participantes desarrollaron el caso de estudio mediante el método convencional y otros 11 participantes utilizaron las herramientas de *PlantPAx*. Dado que los 22 participantes fueron todos diferentes, es decir, ninguna persona programó por ambos métodos para la recolección de datos, se concluye que las muestras del estudio son independientes entre sí [66, 68], lo que confirma la primera condición de la prueba *t-Student*.
2. **Normalidad de datos:** para saber si las muestras provienen de una distribución normal se pueden utilizar diferentes pruebas de hipótesis ya definidas como la *Kolmogorov-Smirnov*, *Anderson-Darling* y *Shapiro-Wilk* [68, 69]. Por ser considerada una de las más robustas [69, 70], en este caso se recurrirá a la prueba de *Shapiro-Wilk* con un 95 % de confianza (significancia de $\alpha = 0.05$) definiendo

como hipótesis nula H_0*: "Los datos provienen de una distribución normal"* y como hipótesis alternativa H_1*: "Los datos no provienen de una distribución normal"*. Los resultados de las dos pruebas (se realiza una prueba por cada método) se adjuntan a continuación:

Método	**Resultados**	
	W	**p-value**
progMC	0.92469	0.3597
progPPX	0.86233	0.06173

Tabla 3.4: Resultados prueba de *Shapiro-Wilk* para programación.
Fuente propia.

Para interpretar los resultados de una prueba de hipótesis se puede analizar el valor del *estadístico de la prueba (W)* o comparar con el nivel de significancia fijado y el *p-value* que arroja la prueba de hipótesis, se seleccionará la segunda opción por facilidad. En la teoría se tiene que si el *p-value* es menor o igual al nivel de significancia, entonces se rechaza la hipótesis nula a favor de la hipótesis alternativa, mientras que si el *p-value* es mayor al nivel de significancia entonces no se rechaza la hipótesis nula. Sin embargo, en la práctica se debe tener mayor cuidado a la hora de no rechazar la hipótesis nula, por lo que si el *p-value* es mayor pero muy cercano al nivel de significancia se debe recurrir a analizar el comportamiento de los datos para tomar la decisión final.

En este caso se observa que, aunque los datos de la variable de tiempo de programación de *MC* provienen de una distribución normal, se rechaza la hipótesis nula para el caso de *PPX* pues el *p-value* es muy cercano al nivel de significancia y en el análisis descriptivo presenta una fuerte asimetría, lo que representa una característica típica de poblaciones no normales. De lo anterior se concluye que los datos de la variable de tiempo de programación de *PPX* no provienen de una distribución normal, es decir, no se cumple el criterio de normalidad para los datos de estudio.

Debido a que una muestra no cumple con el criterio de normalidad, se debe recurrir a pruebas estadísticas no paramétricas [65, 68, 71]. En el caso de comparación de muestras independientes (el caso del presente estudio) se recurre a la prueba de hipótesis no paramétrica de *Wilcoxon-Mann-Whitney*, la cual compara las distribuciones de los datos considerando los rangos en su estadística [72, 73].

Para este caso de estudio planteamos la hipótesis nula H_0*: "FprogMC = FprogPPX"*, es decir, las distribuciones del tiempo de programación de *MC* y de *PPX* son iguales; contra la hipótesis alternativa H_1*: "FprogMC > FprogPPX"*, es decir, la distribución del tiempo de programación de *MC* es mayor a la distribución del tiempo de programación de *PPX*. Se fijará un nivel de confianza del 95 % (significancia de $\alpha = 0.05$). El resultado puede observarse a continuación.

Resultados	
W	**p-value**
89	0.03258

Tabla 3.5: Resultado de la prueba *Wilcoxon-Mann-Whitney* para programación.

Como se observa, el *p-value* obtenido en la prueba (0.03258) es menor al nivel de significancia fijado (0.05) lo que permite rechazar la hipótesis nula a favor de la hipótesis alternativa. Dado que el resultado anterior arroja que la distribución de la variable tiempo de programación de *MC* es mayor a la distribución de la variable tiempo de programación de *PPX*, se puede concluir que, en general, el tiempo de programación de *MC* es mayor al tiempo de programación de *PPX*.

3.2.2.2. Prueba de hipótesis para tiempo de diseño de HMI

Análogamente a la prueba del tiempo de programación se verificará si se cumplen las condiciones de la prueba paramétrica *t-Student* o si se debe recurrir a la prueba *Wilcoxon-Mann-Whitney*.

1. **Independencia de muestras:** dado que los datos del tiempo de diseño de HMI se obtuvieron de la misma forma que los del tiempo de programación, se concluye que estos también cumplen el supuesto de independencia de muestras.
2. **Normalidad:** se realizan las mismas pruebas de *Shapiro-Wilk* para los tiempos de diseño de HMI de los dos métodos.

Método	Resultados	
	W	**p-value**
hmiMC	0.87077	0.07931
hmiPPX	0.8297	0.02313

Tabla 3.6: Resultados prueba de *Shapiro-Wilk* para diseño de HMI.
Fuente propia.

Dado que para el caso de *PPX* el *p-value* (0.0213) es menor al nivel de significancia (0.05) se rechaza con un 95 % de confianza la hipótesis nula de normalidad, por lo tanto, el supuesto de normalidad no se cumple.

Por lo anterior se recurre a la prueba *Wilcoxon-Mann-Whitney* con las hipótesis: H_0*: "FhmiMC = FhmiPPX"* y H_1*: "FhmiMC > FhmiPPX"*, fijando un nivel de confianza del 95 % (significancia de $\alpha = 0.05$). A continuación se adjuntan los resultados de la prueba:

Resultados	
W	**p-value**
115.5	0.0001648

Tabla 3.7: Resultado de la prueba *Wilcoxon-Mann-Whitney* para diseño de HMI.

Como se observa, el *p-value* obtenido en la prueba (0.0001648) es menor al nivel de significancia fijado (0.05) lo que permite rechazar la hipótesis nula a favor de la hipótesis alternativa. El resultado anterior permite concluir que, en general, el tiempo de diseño de HMI de *MC* es mayor al tiempo de *PPX*.

Dado que el tiempo total se define como la suma del tiempo de programación y el de diseño de HMI no es necesario realizar una prueba de hipótesis para las distribuciones de esta variable, puesto que en las dos variables fue mayor el tiempo para *MC*.

Conclusiones

A continuación se exponen las conclusiones generadas a partir del presente trabajo de investigación, presentando entre otras, las correspondientes al objetivo general y los objetivos específicos planteados.

- El conocimiento de estándares y buenas prácticas para diseño de HMI o programación de un algoritmo para PLC permite generar soluciones prácticas con un enfoque profesional que proporcionan seguridad al proceso en el que se encuentran, ya que muchos de los conceptos considerados en estándares y buenas prácticas parten de una larga experiencia de profesionales en el desarrollo de soluciones de automatización. El uso de los conceptos mencionados favorece el diseño, flexibilidad, alerta temprana de fallas, mantenimiento y soporte, entre otros factores relevantes para una empresa que cuente con un proceso gobernado por un PLC y un HMI como método de supervisión.

- De manera general, el tiempo requerido para relacionarse con las herramientas de *PlantPAx* y utilizarlas en un proyecto de programación y/o supervisión, es relativamente corto, esto debido a que se cuenta con documentación bastante completa, además el tiempo requerido puede reducirse en gran medida si se conocen algunos conceptos del método convencional.

- Se definió un método el cual sirve como una guía base de conceptos técnicos relevantes para ofrecer una solución con un enfoque profesional, esto debido a que se consideran estándares, buenas prácticas y un enfoque de reutilización de código.

- El trabajo desarrollado en el capítulo 2 permitió evidenciar desde la experiencia, que el uso de las herramientas de *PlantPAx* favoreció la realización de las tareas comunes en un proyecto de programación y supervisión, aumentando la flexibilidad y reduciendo los tiempos de ingeniería de un proyecto respecto al desarrollo por el método convencional.

- En el desarrollo del experimento fue posible evidenciar que los participantes que utilizaron las herramientas de *PlantPAx* requirieron un menor tiempo para desarrollar la solución del caso de estudio planteado, posteriormente por medio del análisis estadístico se logró comprobar que la implementación de estas herramientas disminuye los tiempos de programación y diseño de HMI.

- El caso de estudio planteado permitió utilizar las herramientas de *PlantPAx* en un proceso que es típicamente automatizado y que ha sido uno de los casos de éxito de la implementación de un sistema *PlantPAx*. Las ventajas pudieron evidenciarse gracias a que el proceso del caso de estudio contaba con característica típicas de cualquier proceso como lo son: secuencias lógicas, control de una variable de proceso e instrumentación común, como válvulas, sensores y motores.

Bibliografía

[1] R. Bayindir and Y. Cetinceviz, "A water pumping control system with a programmable logic controller (plc) and industrial wireless modules for industrial plants—an experimental setup," *ISA transactions*, vol. 50, no. 2, pp. 321–328, 2011.

[2] Q. Chen, B. G. de Soto, and B. T. Adey, "Construction automation: Research areas, industry concerns and suggestions for advancement," *Automation in Construction*, vol. 94, pp. 22–38, 2018.

[3] J. G. C. Lugo, J. J. P. Ybarra, and E. Romero, "Metodología para realizar una automatización utilizando plc," *Impulso*, p. 18, 2005.

[4] V. Hajarnavis and K. Young, "An investigation into programmable logic controller software design techniques in the automotive industry," *Assembly Automation*, vol. 28, no. 1, pp. 43–54, 2008.

[5] O. Ljungkrantz, K. Akesson, M. Fabian, and C. Yuan, "Formal specification and verification of industrial control logic components," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 538–548, 2010.

[6] D. Patel, J. Bhatt, and S. Trivedi, "Programmable logic controller performance enhancement by field programmable gate array based design," *ISA transactions*, vol. 54, pp. 156–168, 2015.

[7] J. L. Paredes and D. Díaz, "Improving immunization of programmable logic controllers using weighted median filters." *ISA transactions*, vol. 44, no. 2, pp. 225–241, 2005.

[8] T. Marshall and Y. Brady, *Process Control and Automation Solution.* Zenith Technologies, Dublín, Irlanda: Biopharmaceutical Processing Development, Design, and Implementation of Manufacturing Processes, 2018.

[9] L. Urbas, M. Obst, and M. Stöss, "Formal models for high performance hmi engineering," *IFAC Proceedings Volumes*, vol. 45, no. 2, pp. 854–859, 2012.

[10] Q.-Q. J. Carlos, F.-G. Ernesto, Q.-A. Víctor, and B.-L. Jorge, "Diseño e implementación de un sistema de control y monitoreo basado en hmi-plc para un pozo de agua potable," *Ingeniería, investigación y tecnología*, vol. 15, no. 1, pp. 41–50, 2014.

[11] N. Electro Industria, (2017, "Sistemas de interfaz hombre-máquina, hmi una ventana abierta a los procesos," http://www.emb.cl/electroindustria/articulo.mvc?xid=837.

[12] C. PARKINSON, "Increase efficiencies with standardized hmi," http://polytron.com/blog/increase-efficiencies-standardized-hmi.

[13] D. GALARA and D. PIRUS, "Finding the way up to the standardization of human machine interface," in *2007 IEEE 8th Human Factors and Power Plants and HPRCT 13th Annual Meeting, Monterey, CA, USA, USA*, August. 2007, pp. 133–139.

[14] Z. Lucena and M. Indriago, "Optimización en el desarrollo de un programa para plc," *Revista Ingeniería UC*, vol. 11, no. 3, pp. 70–78, 2004.

[15] S. C. Park, C. M. Park, and G.-N. Wang, "A plc programming environment based on a virtual plant," *The international journal of advanced manufacturing technology*, vol. 39, no. 11-12, pp. 1262–1270, 2008.

[16] E. Walters and E. Bryla, "Software architecture and framework for programmable logic controllers: A case study and suggestions for research," *Machines*, vol. 4, no. 2, p. 13, 2016.

[17] C. E. STAFF, "Process automation roadmap from rockwell includes plantpax," https://www.controleng.com/articles/process-automation-roadmap-from-rockwell-includes-plantpax/.

[18] M. Fernández, "Plantpax como alternativa a dcs o plc / scada," http://www.automatizar.org/2011/09/plantpax-como-alternativa-dcs-o-plc.html.

[19] R. Automation, "Ec: Plantpax modern distributed control system," https://www.controleng.com/articles/ec-plantpax-modern-distributed-control-system/.

[20] M. Automation, "Case study: Modern dcs process automation," https://www.mceneryautomation.com/cmss_files/attachmentlibrary/McEnery_CaseStudy_LubricantsPetroleum_V3.pdf.

[21] S. HILL, "An evolutionary process: Rockwell automation unveils a broad-based solution platform," https://www.plantengineering.com/articles/an-evolutionary-process-rockwell-automation-unveils-a-broad-based-solution-platform/, 2008.

[22] R. Automation, "Biblioteca de objetos de proceso de rockwell automation," vol. PROCES-PP008D-ES-E, pp. 1–7, 2016.

[23] *Rockwell Automation Library of Process Objects: Configuration and Usage*, Process-rm002i-en-p ed., Rockwell Automation, May 2019.

[24] Y. Li, K. Yu, B. Zhu, R. Han, Y. Li, and J. Wang, "Process automation system development of lyocell staple fiber plant based on plantpax," in *5th International Conference on Advanced Design and Manufacturing Engineering*. Atlantis Press, 2015.

[25] *Guía de inicio rápido para controladores Logix5000*, 1756th ed., Rockwell Automation, Mar. 2004.

[26] *Datos de tags y E/S en los controladores Logix5000*, 1756th ed., Rockwell Automation, Oct. 2009.

[27] *Logix 5000 Controllers I/O and Tag Data*, 1756th ed., Rockwell Automation, Feb. 2018.

[28] P. Aguilera Martínez, "Programación de plc´ s," Ph.D. dissertation, Universidad Autónoma de Nuevo León, 2002.

[29] *Sistema ControlLogix*, 1756th ed., Rockwell Automation, Oct. 2014.

[30] *Logix 5000 Controllers Tasks, Programs, and Routines*, 1756th ed., Rockwell Automation, Feb. 2018.

[31] *Logix5000 Controllers Design Considerations Reference Manual*, 1756th ed., Rockwell Automation, Sep. 2018.

[32] P. T. ORGANIZATION, "Plc training best practices," https://plc-training.org/plc-programming-training-BP6.html.

[33] B. I. Network, "Plc training tips," https://aboutoem.com/tips.html.

[34] I. Forum, "Interlocks in plc," https://instrumentationforum.com/t/interlocks-in-plc/4488.

[35] J. Jayaprakash, D. N. J. Eliot, A. ShakilaBanu, and S. K. Viswanath, "Protection and interlocks of critical equipment in power stations using plc," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*. IEEE, 2016, pp. 1–8.

[36] C. C. for Chemical Process Safety), *Guidelines for Safe Process Operations and Maintenance*, . John Wiley & Sons, Ed. Wiley, 2010. [Online]. Available: https://books.google.com.co/books?id=rKoiVaUn3YoC

[37] S. D. ISA, *Human Machine Interfaces for Process Automation Systems*, ISA Std. ANSI/ISA-101.01-2015, 2015.

[38] R. A. Kris Dornan, "How to improve plant operations through better hmi graphics," http://xybernetics.com/techtalk/ASM/docs/CT535_How_to_Improve_Plant_Operations_Through_Better_HMI_Graphics.pdf.

[39] B. Hollifield, E. Habibi, and D. Oliver, *The high performance HMI handbook*. Plant Automation Services, Incorporated, 2013.

[40] ISA, "Applying isa101 concepts to existing hmi applications," http://wilmingtonisa.org/files/Download/ISA-Applying-ISA101-to-Existing-HMIs_MikeHawrylo.pdf.

[41] S. Deodhar, P. Agrawal, and A. Helekar, "Effective use of colors in hmi design," *International Journal of Engineering Research and Applications*, vol. 4, pp. 384–387, 2014.

[42] J. E. R. D. Avila, "Buenas prácticas para diseño de HMI de alto rendimiento," Febrero 2012.

[43] B. D. Rivera Tene, "Desarrollo de una interfaz humano máquina de alto desempeño (hphmi) para procesos de producción de crudo y gas en proyectos integrales del ecuador pil sa," 2018.

[44] B. Hollifield, "The high performance hmi process graphics to maximize operator effectiveness," *The Instrumentation, Systems and Automation Society Intech Journal*, vol. 59, pp. 30–37, 2012.

[45] A. Hossain and T. Zaman, "Hmi design: An analysis of a good display for seamless integration between user understanding and automatic controls," in *American Society for Engineering Education*. American Society for Engineering Education, 2012.

[46] *FactoryTalk View Site Edition User's Guide*, Viewse-um006n-en-e ed., Rockwell Automation, Feb. 2019.

[47] *FactoryTalk View Machine Edition User's Guide*, Viewme-um004o-en-e ed., Rockwell Automation, Jan. 2019.

[48] *Logix para programadores - Sesión de instrucciones Add-On*, Rockwell Automation.

[49] *Sistema ControlLogix - Manual de usuario*, 1756th ed., Rockwell Automation, Oct. 2014.

[50] *Logix 5000 Controllers Add On Instructions*, 1756th ed., Rockwell Automation, Feb. 2018.

[51] *PlantPAx Distributed Control System, Selection Guide*, Proces-sg001i-en-p ed., Rockwell Automation, Apr. 2016.

[52] *PlantPAx Distributed Control System, Reference Manual*, Proces-rm001j-en-p ed., Rockwell Automation, Mar. 2016.

[53] *PlantPAx Distributed Control System, Process Library Reference Manual*, Proces-rm002f-en-p ed., Rockwell Automation, Feb. 2017.

[54] *PlantPAx Distributed Control System, Rockwell Automation Library of Process Objects*, Proces-rm002g-en-p ed., Rockwell Automation, Jul. 2018.

[55] *Library of Process Objects: Basic Analog Input*, Syslib-rm001f-en-e ed., Rockwell Automation, Jan. 2016.

[56] *Library of Process Objects: Analog/Pulsed Control Valve*, Syslib-rm034e-en-p ed., Rockwell Automation, Feb. 2017.

[57] *Library of Process Objects: Solenoid-operated Valve*, Syslib-rm015h-en-e ed., Rockwell Automation, Feb. 2017.

[58] *Library of Process Objects: Variable Speed Drive*, Syslib-rm016g-en-e ed., Rockwell Automation, Feb. 2017.

[59] H. L. Solano and C. R. Álvarez, *Estadística descriptiva y distribuciones de probabilidad*. Universidad del Norte, 2005.

[60] T. Seoane, J. L. R. Martín, E. Martín-Sánchez, S. Lurueña-Segovia, and F. A. Moreno, "Capítulo 7: estadística: estadística descriptiva y estadística inferencial," *SEMERGEN-Medicina de familia*, vol. 33, no. 9, pp. 466–471, 2007.

[61] F. Quevedo, "Medidas de tendencia central y dispersión," *Medwave*, vol. 11, no. 03, 2011.

[62] W. Navidi, *Estadística para ingenieros*. McGraw Hill Interamericana, 2006, no. 519.5 N325.

[63] A. V. Sabadías, *Estadística descriptiva e inferencial*. Univ de Castilla La Mancha, 1995, vol. 8.

[64] E. R. Arias, "Estadística: Medición, descripción e inferencia," *Perspectivas Psicológicas, Santo Domingo (Rep. Dom.), Volúmenes*, vol. 6, pp. 126–172, 2010.

[65] M. Gómez-Gómez, C. Danglot-Banck, and L. Vega-Franco, "Sinopsis de pruebas estadísticas no paramétricas. cuándo usarlas," *Revista Mexicana de Pediatría*, vol. 70, no. 2, pp. 91–99, 2003.

[66] S. P. Fernández and S. P. Díaz, "Métodos paramétricos para la comparación de dos medias. t de student," *Metodología de la Investigación cualitativas*, vol. 8, pp. 37–41, 2001.

[67] R. A. Sánchez Turcios, "t-student: Usos y abusos," *Revista mexicana de cardiología*, vol. 26, no. 1, pp. 59–61, 2015.

[68] E. Flores-Ruiz, M. G. Miranda-Novales, and M. Á. Villasís-Keever, "El protocolo de investigación vi: cómo elegir la prueba estadística adecuada. estadística inferencial," *Revista alergia México*, vol. 64, no. 3, pp. 364–370, 2017.

[69] N. Mohd Razali and B. Yap, "Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests," *J. Stat. Model. Analytics*, vol. 2, 01 2011.

[70] I. Pedrosa, J. Juarros-Basterretxea, A. Robles-Fernández, J. Basteiro, and E. García-Cueto, "Pruebas de bondad de ajuste en distribuciones simétricas,¿ qué estadístico utilizar?" *Universitas Psychologica*, vol. 14, no. 1, pp. 245–254, 2015.

[71] V. Berlanga and M. J. Rubio Hurtado, "Clasificación de pruebas no paramétricas. cómo aplicarlas en spss," *REIRE. Revista d'Innovació i Recerca en Educació, 2012, vol. 5, num. 2, p. 101-113*, 2012.

[72] N. Feltovich, "Nonparametric tests of differences in medians: comparison of the wilcoxon–mann–whitney and robust rank-order tests," *Experimental Economics*, vol. 6, no. 3, pp. 273–297, 2003.

[73] M. W. Fagerland and L. Sandvik, "The wilcoxon–mann–whitney test under scrutiny," *Statistics in medicine*, vol. 28, no. 10, pp. 1487–1497, 2009.

[74] J. Aroni Mocada, J. Bellina Morán, H. Díaz Delgado, C. Escurra Farro, and S. Pérez Asalde, "Diseño de una línea de producción para la elaboración de cerveza artesanal de algarroba," 2016.

[75] M. C. Terán Terán, "Diseño e implementación de un sistema de automatización para una línea de producción de cerveza artesanal," B.S. thesis, Quito, 2018., 2018.

[76] M. F. Colignon and G. E. Roldán, "Automatización de proceso para elaboración de cerveza artesanal," 2018.

[77] I. D. Yubero, "Cerveza," *Distribución y consumo*, vol. 3, p. 45, 2015.

[78] S. Pilla and G. Vinci, *Cervezas de todo el mundo*. Parkstone International, 2013.

Anexo A

Caso de estudio

INCIDENCIA DEL USO DE LAS HERRAMIENTAS DE *PLANTPAX* PARA UNA PROPUESTA DE PROGRAMACIÓN Y SUPERVISIÓN EN UN CASO DE ESTUDIO - EXPERIMENTACIÓN

A.1. Introducción

A continuación se describe el caso de estudio que servirá como experimento para desarrollar los objetivos de la tesis de investigación titulada "*Incidencia del uso de las herramientas de PlantPAx para una propuesta de programación y supervisión en un caso de estudio*".

Los participantes de la etapa experimental son estudiantes de la *Universidad del Cauca* con conocimientos en el uso de aplicaciones del fabricante *Rockwell Automation* tales como: *RSLogix 500*, *RSLogix 5000*, *RSLinx Classic*, *SoftLogix 5800 (Chassis Monitor)* y *Factory Talk View*.

El caso de estudio se encuentra basado en el proceso de producción de cerveza, específicamente en 2 etapas: macerado y cocción, los cuales pueden ser desarrollados en 2 equipos con la instrumentación y conexiones que se describen en la sección A.3.4.

A continuación se hará una breve descripción del proceso de producción de cerveza y sus materias primas. La descripción no busca ser muy detallada puesto que el desarrollo

del caso de estudio estará enfocado en seguir la descripción del proceso visto en la sección A.3.1 y la prosa lógica definida en la sección A.3.2.

A.2. Proceso de producción de cerveza

La cerveza puede clasificarse de acuerdo a su producción en cerveza artesanal y cerveza industrial. La primera hace referencia al proceso en el cual las cantidades de producción no son muy grandes y en dónde generalmente los productores se enfocan en el sabor y en la técnica de fermentación. Por otra parte, la cerveza industrial es aquella que se caracteriza por contar con poca mano de obra ya que la mayoría de procesos se encuentran automatizados, además, las cantidades y demás parámetros de la receta son controlados a detalle [74].

A.2.1. Materia prima

Las materias primas utilizadas para la elaboración de la cerveza son [74, 75, 76, 77, 78]:

- **Cebada:** Es una planta gramínea anual, que cuenta con un gran contenido de almidón, es originaria de Asia Occidental y África nororiental.
- **Lúpulo:** Es una planta trepadora, es la encargada de dar el particular amargor, aroma y sabor a la cerveza, esto gracias a que contiene resinas y aceites esenciales.
- **Levadura:** Microorganismos encargados de fermentar el azúcar del mosto para obtener alcohol y CO_2.
- **Agua:** Materia prima utilizada en mayor cantidad, es utilizada para el malteado de los cereales, para su limpieza, para la maceración, entre otros procesos.

A.2.2. Etapas del proceso

Las etapas que comprenden el proceso de producción de cerveza son [74, 75, 76, 77, 78]:

- **Preparación del agua:** La calidad de muchas cervezas se basa en el agua, por esto el agua debe ser tratada para eliminar gérmenes y bacterias propias del agua y para alcanzar un pH por debajo de 7.
- **Molienda:** En esta etapa se reducen a trozos los granos de cebada, esta etapa es muy importante porque de esta depende el sabor y el rendimiento que se obtenga en el macerado.
- **Macerado:** Esta etapa consiste en mezclar los granos de cebada molidos con agua a una temperatura determinada con el fin de disolver los almidones y demás componentes de la cebada.
- **Cocción:** El mosto se debe cocer para buscar obtener componentes de amargor y obtener el aroma del lúpulo, también para destruir enzimas y evitar que continúen con su efecto biológico, para esterilizar el mosto, entre otros motivos.
- **Enfriado:** En esta etapa se enfría el mosto para obtener una temperatura adecuada para la levadura seleccionada y para el tipo de cerveza que se desea conseguir.
- **Fermentación y maduración:** En esta etapa se agrega la levadura dentro del tanque de fermentación, es aquí donde se produce el alcohol y CO_2.
- **Envasado:** La cerveza que ha estado en los tanques de almacenamiento y ha madurado según las características del maestro cervecero es embotellada considerando variables como la presión y la temperatura con el fin de conservar las características adquiridas en el proceso productivo.

Es necesario aclarar que para el caso de estudio solo se tendrán en cuenta las etapas de macerado y cocción, las cuales comprenden la elaboración y la primera etapa de tratamiento del mosto.

A.3. Definición del caso de estudio

A.3.1. Descripción del proceso

En una planta de producción de cerveza se desea llevar acabo la elaboración y tratamiento del mosto en una unidad de producción compuesta de etapas y operaciones necesarias

para su elaboración. La materia prima para el proceso está compuesta de: agua, trozos de cebada y lúpulo. La velocidad y tiempo de mezclado al igual que las temperaturas y tiempos de cocción dependen de la receta de producción definida. El proceso de producción de cerveza inicia con la etapa de macerado, en el cual los granos de cebada molidos y el agua son depositados en un reactor donde luego se mezclan y cuecen a temperatura controlada durante un tiempo determinado. Finalizada la etapa de macerado se transporta la mezcla a través de una tubería hacia un intercambiador de calor para iniciar la etapa de cocción del mosto, en esta etapa se adiciona el lúpulo a la mezcla y se cuece a temperatura controlada durante un tiempo determinado. El establecimiento de las condiciones del proceso y parámetros de la receta son ajustables.

A.3.2. Prosa lógica o lógica de control

El nombrado de los instrumentos se realizó siguiendo el estándar *ISA 5.1* como se muestra a continuación: ***TCV*** (Temperature Control Valve), ***FV*** (Flow Valve), ***TIT*** (Temperature Indicator Transmitter), ***LIT*** (Level Indicator Transmitter), ***M*** (Motor), ***LSH*** (High Level Swtich), ***LSL*** (Low Level Switch) y ***FSL*** (Low Flow Switch). En el proceso existen 2 lazos, uno para el macerado y otro para la cocción, el primero tiene asignado el número *100* y el segundo el *200*.

Las etapas de macerado y cocción del mosto para la elaboración de cerveza se llevarán a cabo de la siguiente manera en la unidad de producción:

La elaboración del mosto inicia con el llenado del tanque reactor ***TR_ 100*** con las materias primas, aquí se debe tener en cuenta que el tanque esté vacío, esto se logra por medio de la medición de ***LIT_ 100***, ***LSL_ 100*** activado y ***LSH_ 100*** desactivado, esto indica que no hay materia prima y se debe empezar el llenado, lo anterior funciona como condiciones de seguridad. Con las condiciones de seguridad cumplidas se abre la válvula ***FV_ 100A*** (válvula solenoide - ON/OFF), esta es la encargada de proporcionar el agua al proceso y ***FV_ 100B*** (válvula solenoide - ON/OFF), encargada de proporcionar los granos molidos de cebada. El tanque reactor cuenta con una capacidad máxima de 4000 litros y una altura de 3.5 m. Cuando el transmisor de nivel indique un nivel alto (3 m) en el tanque reactor se deben cerrar las válvulas ***FV_ 100A*** y ***FV_ 100B***, si por algún motivo el transmisor falla, ***LSH_ 100*** activado indicará que deben cerrarse las válvulas que proporcionan la materia prima. Con la cantidad completa de materia prima se procede

a iniciar el mezclado, el reactor consta de un mecanismo de agitación impulsado por un motor ***M_ 100*** que garantiza un mezclado uniforme, la velocidad de agitación debe ser dada de acuerdo a las unidades de ingeniería (0-60 Hz). Una vez terminado el mezclado se debe elevar la temperatura de la mezcla y mantenerla constante durante el tiempo que dure la etapa de macerado, por lo que es necesario controlar la temperatura, para esto se cuenta con ***TIT_ 100*** y ***TCV_ 100***, el primero permite conocer la temperatura de ***TR_ 100*** y el segundo instrumento controla el flujo de vapor (mediante la apertura de la válvula) hacia el reactor, el cual es el encargado de modificar la temperatura de la mezcla.

Una vez finalizada la etapa de macerado inicia la etapa de cocción del mosto teniendo en cuenta que el tanque de cocción ***TK_ 200*** debe estar vacío, esto se logra conociendo la medición de ***LIT_ 200***, lo anterior funciona como condición de seguridad. Con la condición de seguridad cumplida se procede a transportar la materia prima de ***TR_ 100*** a ***TK_ 200***, para esto se abre la válvula de paso ***FV_ 200A*** (válvula solenoide – ON/OFF) y se enciende la motobomba ***M_ 200***, la velocidad con la que bombea el líquido debe ser dada de acuerdo a las unidades de ingeniería (0-60 Hz). ***M_ 200*** cuenta con una condición de seguridad que funciona con ***FSL_ 200***, cuando el interruptor está desactivado indica que hay presencia de flujo, entonces ***M_ 200*** podrá funcionar de manera correcta, en caso contrario, ***FSL_ 200*** activado indica que no hay presencia de flujo, entonces ***M_ 200*** permanecerá apagada con el fin de que esta no funcione al vacío y así evitar posibles daños en el instrumento. El tanque de cocción cuenta con una capacidad máxima de 5000 litros y una altura de 4.5 m. El llenado de ***TK_ 200*** se realizará hasta que ***LIT_ 200*** indique una altura de 3 m en el nivel del tanque, en este momento se apaga ***M_ 200*** y se cierra la válvula ***FV_ 200A***. Con lo anterior se abre la válvula ***FV_ 200B***, encargada de suministrar el lúpulo al mosto, hasta que ***LIT_ 200*** indique un nivel alto (4 m) en el tanque. Cumplida esta etapa se debe elevar la temperatura del mosto y mantenerla constante durante el tiempo que dure la etapa de cocción, por lo que es necesario controlar la temperatura, para esto se cuenta con ***TIT_ 200*** y ***TCV_ 200***, instrumentos que funcionan de igual manera como los empleados en la etapa de macerado. Finalizada la etapa de cocción, se vacía el ***TK_ 200***, para esto se abre la válvula ***FV_ 200C*** para transportar el líquido a la siguiente etapa.

La velocidad y tiempo de mezclado en el macerado, el tiempo y la temperatura de cocción a controlar en ambas etapas son parámetros ajustables y se encuentran establecidos según las especificaciones definidas en la receta de producción, esto debido a que estas variables

influyen en las características con las que contará la cerveza.

MACERADO	
Parámetro	**Valor**
Velocidad de mezclado	30 Hz
Tiempo de mezclado	30 s
Temperatura de cocción	65 °C
Tiempo de cocción	60 s
COCCIÓN	
Parámetro	**Valor**
Temperatura de cocción	85 °C
Tiempo de cocción	60 s

Tabla A.1: Receta de producción para la elaboración y cocción del mosto en el proceso de producción de cerveza.

A.3.3. Tabla de instrumentos

	INSTRUMENTO	**TAG**	**IN**	**OUT**	**DESCRIPCIÓN**
1	Válvula de control de temperatura	TCV_100	3-15 psi	0-100 % (apertura)	Válvula encargada de controlar el flujo de vapor hacia el reactor donde se realiza el macerado.
2	Válvula de flujo	FV_100A	0-110 v	0-1 (ON-OFF)	Válvula dosificadora de agua para la etapa de macerado.
3	Válvula de flujo	FV_100B	0-110 v	0-1 (ON-OFF)	Válvula dosificadora de cebada para la etapa de macerado.
4	Transmisor indicador de temperatura	TIT_100	0-100 °C	4-20 mA	Transmisor indicador que mide la temperatura dentro del reactor donde se realiza la maceración.
5	Transmisor indicador de nivel	LIT_100	0-3.5 m	4-20 mA	Transmisor indicador mide la altura dentro del reactor donde se realiza la maceración.
6	Switch de nivel alto	LSH_100	3 m	0-1 (ON-OFF)	Switch que indica el nivel alto en el reactor donde se realiza la maceración.
7	Switch de nivel bajo	LSL_100	0 m	0-1 (ON-OFF)	Switch que indica el nivel bajo en el reactor donde se realiza la maceración.
8	Motor agitador	M_100	0-60 hz	0-100 % (velocidad)	Motor encargado de agitar la mezcla en la maceración.
9	Convertidor de señal de corriente a presión	TY_100	4-20 mA	3-15 psi	Convertidor encargado de transformar la señal de corriente (señal de control) a una señal de presión (señal del actuador).
10	Válvula de control de temperatura	TCV_200	3-15 psi	0-100 % (apertura)	Válvula encargada de controlar el flujo de vapor hacia el tanque de cocción del mosto.
11	Válvula de flujo	FV_200A	0-110 v	0-1 (ON-OFF)	Válvula encargada del paso del mosto hacia el tanque de cocción.
12	Válvula de flujo	FV_200B	0-110 v	0-1 (ON-OFF)	Válvula dosificadora de lúpulo para la etapa de cocción.
13	Válvula de flujo	FV_200C	0-110 v	0-1 (ON-OFF)	Válvula encargada del paso del mosto cocido hacia la siguiente etapa.
14	Transmisor indicador de temperatura	TIT_200	0-100 °C	4-20 mA	Transmisor indicador que mide la temperatura dentro del tanque donde se realiza la cocción.
15	Transmisor indicador de nivel	LIT_200	0-4.5 m	4-20 mA	Transmisor indicador mide la altura dentro del tanque donde se realiza la cocción.
16	Switch de flujo bajo	FSL_200	1 L	0-1 (ON-OFF)	Switch encargado de detectar un flujo mínimo.
17	Motor bomba	M_200	0-60 hz	0-100 % (velocidad)	Motor encargado de transportar el mosto hacia el tanque de cocción.
18	Convertidor de señal de corriente a presión	TY_200	4-20 mA	3-15 psi	Convertidor encargado de transformar la señal de corriente (señal de control) a una señal de presión (señal del actuador).

Tabla A.2: Instrumentación para las etapas de macerado y cocción del mosto en el proceso de producción de cerveza.

A.3.4. Diagrama P&ID

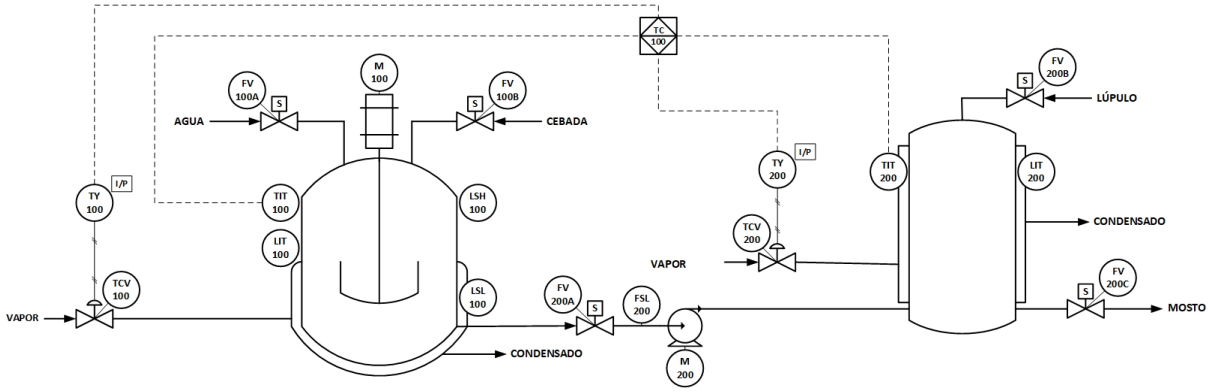


Figura A.1: Diagrama P&ID para las etapas de macerado y cocción del mosto en el proceso de producción de cerveza.