



UNIVERSIDAD DON BOSCO
VICERRECTORÍA DE ESTUDIOS DE POSTGRADO

TRABAJO DE GRADUACIÓN
SISTEMA SCADA PARA LA ESTACIÓN DE ENSAMBLE POR ROBOT DE LA CELDA DE
MANUFACTURA ICIM3000

PARA OPTAR AL GRADO DE
MAESTRA EN MANUFACTURA INTEGRADA POR COMPUTADORA

ASESOR:
MAESTRO EDWIN ARMANDO GUEVARA ALEMÁN

PRESENTADO POR:
TANIA DENISE MARTÍNEZ TORRES

Antiguo Cuscatlán, La Libertad, El Salvador, Centroamérica.

Enero 2016

Agradecimientos

Agradezco primeramente a Dios todopoderoso por estar siempre a mi lado y permitirme alcanzar una más de mis metas, sumando una más a las bendiciones que me ha dado a lo largo de mi vida.

A mi familia, en especial a mis padres Francia y Jorge, por su amor y dedicación y porque nunca han dejado de motivarme y apoyarme.

A la Universidad Don Bosco por todas las facilidades que me brindaron para poder estudiar esta maestría y culminarla con éxito, a todos los profesores del curso por su dedicación en su labor, gracias por poner a nuestra disposición sus conocimientos y experiencias, a mis compañeros por todo el apoyo y la amistad que me brindaron, a mi asesor Mg. Edwin Guevara por compartir su experiencia con los sistemas SCADA y por sus valiosos consejos y recomendaciones en la realización de este proyecto, al encargado del laboratorio iCIM Mg. Hector Carías por su valiosa ayuda con el uso de los equipos de la celda de manufactura y al director de la maestría Mg. Gilberto Carrillo por su disposición e interés en todo lo referente a la maestría.

Índice

| | |
|---|----|
| Resumen..... | 6 |
| 1. Introducción | 7 |
| 2. Objetivos | 8 |
| 2.1 General | 8 |
| 2.2 Específicos | 8 |
| 3. Justificación | 9 |
| 4. Descripción General de la Aplicación | 10 |
| 5. Sistemas SCADA..... | 11 |
| 5.1 Ventajas de los Sistemas SCADA | 11 |
| 5.2 Estructura de un Sistema SCADA | 12 |
| 5.2.1 MTU y RTU..... | 12 |
| 5.2.2 Software SCADA..... | 13 |
| 5.2.2.1 HMI (Human Machine Interface) | 13 |
| 5.2.2.1.1 Normas para el diseño de la HMI | 14 |
| 5.2.2.2 Comunicación entre Aplicaciones | 28 |
| 5.2.2.3 Almacenamiento de Datos | 29 |
| 6. Celda de Manufactura iCIM 3000® | 31 |
| 6.1 Estaciones que la componen | 32 |
| 6.2 Recursos | 33 |
| 6.3 Sistema SCADA del iCIM 3000®..... | 35 |
| 7. Estación de Ensamble por Robot | 38 |
| 7.1 Elementos que la componen | 38 |
| 7.1.1 Robot RV-3SB-S312: | 38 |
| 7.1.1.1 Controlador CR2B-574-S312 | 39 |
| 7.1.1.1.1 Interfaz Ethernet del controlador CR2B-574-S312 | 40 |
| 7.1.1.2 Lenguaje MELFA BASIC IV | 41 |
| 7.1.2 Cámara SBOC-Q- R3C-WB | 42 |
| 7.2 Funcionamiento..... | 42 |
| 8. Software IGNITION de Inductive Automation® | 44 |
| 8.1. Características | 44 |
| 8.2. Requerimientos del sistema | 46 |
| 8.3 Funcionamiento..... | 46 |

| | |
|--|-----|
| 9. Desarrollo de las Aplicaciones SCADA | 48 |
| 9.1 Comunicación de la Estación de ensamble con la computadora | 48 |
| 9.1.1 Programas en el controlador del robot | 51 |
| 9.2 Plan de procesos realizado en CIROS Production® | 56 |
| 9.3 SCADA con el software IGNITION de Inductive Automation®..... | 57 |
| 9.3.1 TCP Driver | 57 |
| 9.3.2 Base de datos..... | 60 |
| 9.3.3 Descripción de la HMI de la modalidad 1: “Supervisión de la estación de ensamble en conjunto con SCADA de FESTO” | 62 |
| 9.3.4 Descripción de la HMI de la Modalidad 2: “Sistema SCADA Para Estación de Ensamble Aislada” | 69 |
| 10. Instrucciones de uso de las aplicaciones | 76 |
| 10.1 Instrucciones uso de SCADA en modalidad 1 | 76 |
| 10.2 Instrucciones uso de SCADA en modalidad 2 | 76 |
| 11. Resultados | 78 |
| 12. Conclusiones | 79 |
| 13. Bibliografía | 80 |
| 14. Anexos | 83 |
| 14.1 Ajuste de parámetros en el controlador del robot | 83 |
| 14.2 Código fuente de los programas y las posiciones almacenados en el controlador del robot. . | 84 |
| 14.2.1 SCADA modalidad 1 | 84 |
| 14.2.2 SCADA modalidad 2 | 111 |
| 14.3 Planes de procesos creados en CIROS Production para el SCADA en modalidad 1 | 121 |
| 14.4 Tags, elementos y scripts utilizados para la elaboración de la HMI del SCADA en la modalidad 1..... | 132 |
| 14.4.1 Tags..... | 132 |
| 14.4.2 Ventanas..... | 134 |
| 14.4.2.1 Elementos y scripts de la ventana principal | 135 |
| 14.4.2.2 Elementos y scripts de la ventana de detalle de fallas | 139 |
| 14.4.2.3 Elementos y scripts de la ventana de estadísticas | 141 |
| 14.4.2.4 Elementos y scripts de la ventana de reset de conteo de alarmas | 142 |
| 14.4.2.5 Elementos y scripts de la ventana de reset de conteo de sets..... | 144 |
| 14.4.2.6 Elementos y scripts de la ventana de historiales | 146 |
| 14.5 Tags, elementos y scripts utilizados para la elaboración de la HMI del SCADA en la modalidad 2..... | 147 |

| | |
|--|-----|
| 14.5.1 Tags..... | 147 |
| 14.5.2 Ventanas..... | 148 |
| 14.5.2.1 Elementos y scripts de la ventana principal | 149 |
| 14.5.2.2 Elementos y scripts de la ventana de aviso | 155 |
| 14.5.2.3 Elementos y scripts de la ventana de detalle de alarma | 155 |
| 14.5.2.4 Elementos y scripts de la ventana de estadísticas | 157 |
| 14.5.2.5 Elementos y scripts de la ventana de reset de conteo de alarmas | 158 |
| 14.5.2.6 Elementos y scripts de la ventana de reset de conteo de sets..... | 160 |
| 14.5.2.7 Elementos y scripts de la ventana de históricos | 162 |
| 14.6 Abreviaturas utilizadas..... | 163 |

Resumen

El proyecto consiste en el desarrollo de un sistema SCADA con fines didácticos exclusivo para la estación de ensamble por robot de la celda de manufactura iCIM 3000® perteneciente a la universidad Don Bosco. Si bien la celda cuenta ya con un sistema SCADA, este no muestra a detalle el comportamiento de las estaciones y no permite cambiar la HMI que trae por defecto para poner otros esquemas o monitorear otras variables de las estaciones, además de que al ser un software didáctico propietario, no es una alternativa que el alumno pueda encontrar en la industria para desarrollar sistemas SCADA. Por esta razón se decidió crear este proyecto como un primer paso en la elaboración de un SCADA alternativo completo para toda la celda en el futuro.

Se eligió iniciar con la estación de ensamble por robot, debido a que esta era comandada por otro tipo de controlador diferente al típico PLC y deseaba evaluarse también como los software SCADA de uso industrial operan con otros controladores, como software se eligió a IGNITION de Inductive Automation® debido a que es un software de uso industrial muy versátil, que puede ser instalado rápidamente en cualquier sistema operativo y ofrece conectividad con una gran variedad de protocolos y de dispositivos de diferentes fabricantes, lo que lo hace ideal para este proyecto y su ampliación en el futuro.

El proyecto consta de dos modalidades, la primera en que el SCADA realizado trabaja en conjunto con el SCADA del fabricante de la celda sin alterar su funcionamiento normal, en este caso solo se llevan a cabo tareas de monitorización como mostrar en pantalla el estado de los sensores de la estación, la posición del robot, nombre del programa que se está ejecutando en la estación y la notificación de alarmas cuando ocurre alguna anomalía en la estación. En la otra modalidad la estación de ensamble trabaja de manera individual y con el SCADA realizado se pueden enviar órdenes para ensamblar tres diferentes productos, además de notificar en pantalla de algún problema ocurrido y de enviar órdenes para detener el proceso o reintentar la tarea que se estaba ejecutando antes del problema. En ambas modalidades se siguieron lineamientos de ergonomía y seguridad para el usuario, además de contar con pantallas que lleven estadísticas de producción y de alarmas y del historial de estas para los últimos 6 días.

1. Introducción

La automatización es un área en constante crecimiento, cada vez son más las empresas que deciden automatizar sus plantas buscando incrementar la producción, mejorar la calidad de los productos, reducir costos, etc., estos sistemas automatizados implican el trabajar con una gran cantidad de sensores y actuadores, por lo que se hace necesario disponer de un adecuado sistema de supervisión y control que garantice su correcto funcionamiento, para ello se cuenta con los sistemas SCADA (sistemas de supervisión, control y adquisición de datos), que de la misma forma han evolucionado mucho con el tiempo, pues pasaron de la supervisión y control manual que el operario tenía que hacer en la planta, a ser aplicaciones que pueden ejecutarse remotamente desde una computadora e incluso un dispositivo móvil, liberando al operario de tener que trasladarse hasta donde se realiza el proceso, el cual puede ser un lugar distante o de difícil acceso y de realizar tareas de manera manual que le serían más trabajosas o incluso peligrosas, ahora puede desde la comodidad de una oficina a través de una interfaz humano-máquina (HMI) conocer como está operando la planta y tener a un clic del ratón el modificar consignas, encender/apagar dispositivos, etc.

La tarea del operario ha cambiado y ahora parte de su trabajo consiste en pasar varias horas frente a la pantalla llevando a cabo tareas de supervisión y control del proceso, lo que implica que la HMI del SCADA deberá de estar diseñada pensando en la ergonomía y seguridad de este, así la interfaz tendrá que ser comprensible, segura (pues estará sujeta a errores humanos) y cuidar aspectos como colores, gráficos, tamaños de texto, etc., cuyo mal empleo pueden generar algún trastorno visual o musculoesquelético en el operario.

Conscientes de la importancia de los sistemas SCADA la universidad Don Bosco como institución formadora de profesionales en las áreas de electrónica, automatización y manufactura busca desarrollar proyectos que abonen conocimiento y experiencia para que luego pueda ser aplicada en la industria por los futuros profesionales, así surge la idea de crear este proyecto que consiste en un sistema SCADA utilizando el software IGNITION® para la estación de ensamble por robot de la celda, siguiendo para la HMI lineamientos de ergonomía y seguridad para el usuario. El proyecto es un primer paso en la elaboración en un futuro de un SCADA alternativo completo para toda la celda.

2. Objetivos

2.1 General

- Desarrollar un sistema de supervisión, control y adquisición de datos (SCADA) con fines didácticos exclusivo para la estación de ensamble por robot de la celda de manufactura iCIM 3000® de la Universidad Don Bosco.

2.2 Específicos

- Utilizar para la elaboración del sistema una alternativa de software SCADA empleado en la industria
- Obtener información de la estación como estado de los sensores, posición del robot y alarmas generadas ante alguna anomalía en el proceso
- Enviar órdenes a la estación para que sean ejecutadas por el robot
- Diseñar la HMI del sistema siguiendo lineamientos que garanticen la ergonomía y seguridad del usuario.

3. Justificación

Se deseaba crear un proyecto con fines didácticos para la universidad Don Bosco en el campo de la manufactura integrada por computadora, específicamente en el área de los sistemas SCADA. La universidad cuenta con la celda de manufactura iCIM 3000® la cual posee un sistema SCADA propietario realizado con CIROS Production® y exclusivo de FESTO Didactic® que controla y supervisa toda la celda, sin embargo no muestra a detalle el comportamiento de cada estación y no permite cambiar la HMI (interfaz humano máquina) que trae por defecto para poner otros esquemas o monitorear otras variables de las estaciones, además de que al ser un software didáctico propietario, no es una alternativa que el alumno pueda encontrar en la industria para desarrollar sistemas SCADA.

Así surge la idea de crear un SCADA con más detalle para cada estación del iCIM usando softwares SCADA empleado en la industria, pero por el tiempo limitado para este proyecto, la aplicación realizada es únicamente para una de las estaciones del iCIM específicamente la de ensamble. Se eligió esta estación debido a que es comandada por otro tipo de controlador distinto a los PLC que son los controladores por excelencia más utilizados en la industria y que por ende los software SCADA han optimizado mucho las herramientas para su comunicación, por lo que se quería experimentar como estos software SCADA trabajan con otro tipo de controladores.

Se utilizó para crear el proyecto al software SCADA IGNITION de Inductive Automation®, debido a que es un software de uso industrial muy versátil, que puede ser instalado rápidamente en cualquier sistema operativo y ofrece conectividad con una gran variedad de protocolos y de dispositivos de diferentes fabricantes, lo que lo hace ideal para integrar después al proyecto las otras estaciones del iCIM, además de ofrecer muchas facilidades para su uso como un sitio con cursos de entrenamiento gratuitos y el acceso a una versión de prueba también gratuita.

4. Descripción General de la Aplicación

Se desarrollaron dos diferentes sistemas SCADA con el software IGNITION de Inductive Automation®, en ambos la interfaces humano-máquina creadas siguen lineamientos para la ergonomía y seguridad del usuario, estas se ejecutan en una computadora laptop que se comunica con el controlador de la estación de ensamble a través de Ethernet utilizando el protocolo TCP/IP.

En la primera modalidad, el sistema SCADA creado trabaja en conjunto con el SCADA del fabricante de la celda realizando únicamente tareas de monitorización de la estación sin afectar su funcionamiento, en esta puede observarse en pantalla el estado de los sensores de la estación, la posición del robot, programa que se está ejecutando y las alarmas que se generan cuando ocurre alguna anomalía.

En la segunda modalidad la estación de ensamble trabaja de manera individual, aquí el usuario puede mandar órdenes a la estación para producir tres diferentes productos, además de mostrar en pantalla las alarmas generadas cuando ocurre algún problema y la posibilidad de reintentar la tarea que se estaba realizando antes de que ocurriera la alarma.

En ambas modalidades también se tendrán pantallas que muestren estadísticas de producción y de alarmas generadas y otra de históricos donde se muestran gráficos de tendencia tanto de producción como de alarmas activadas en los últimos seis días.

5. Sistemas SCADA

Un sistema de Control con Supervisión y Adquisición de Datos mejor conocido como SCADA por sus siglas en inglés (Supervisory Control and Data Acquisition), es una tecnología que permite la adquisición de datos remotos desde un proceso para su monitorización o supervisión y también el envío de instrucciones limitadas para su control. No debe confundirse al sistema SCADA con un sistema de control, pues si bien un SCADA ejecuta tareas de control, estas son menores tales como: activación/desactivación de procesos o dispositivos, ajuste de parámetros como valores de consigna, entre otras; dejando las tareas de control complejas a los dispositivos específicos para ello como son controladores o autómatas programables.

5.1 Ventajas de los Sistemas SCADA

- Los operarios pueden llevar a cabo tareas de supervisión y control desde la comodidad de una oficina en lugar de tener que trasladarse hasta donde se realiza el proceso, el cual puede ser un lugar distante o de difícil acceso.
- Con un SCADA los operarios tienen a un clic del ratón la realización de tareas que de llevarlas a cabo de manera manual serían más trabajosas y les tomarían más tiempo, lo cual es crítico en casos de mal funcionamiento que requieren atención lo antes posible.
- El registro de los datos es mucho más eficiente con un sistema SCADA, pues al ser un proceso computarizado, permite la tabulación y análisis de los mismos de manera rápida, facilitando así el cálculo de factores de rendimiento y por ende la toma de decisiones para mejorar el funcionamiento de la planta, además de llevar un historial de estos datos.
- Los sistemas SCADA actuales permiten el uso de tecnologías Web o celulares permitiendo así que el sistema pueda mantener informado ya sea por correo electrónico, de voz o mensaje de texto al operario sobre situaciones como alarmas o de cómo va el proceso en general cada cierto tiempo.

5.2 Estructura de un Sistema SCADA

5.2.1 MTU y RTU

En la Figura 5.1 se muestra la estructura general de un sistema SCADA: La unidad maestra o MTU por sus siglas en inglés (Master Terminal Unit) es el centro de control del sistema, la cual en los sistemas SCADA modernos estará siempre basada en computadora, esta cuenta con un software especializado que le permite llevar a cabo las tareas de supervisión y control del proceso, mediante la comunicación con las unidades remotas mejor conocidas como RTU (Remote Terminal Unit), las cuales estarán en comunicación con sensores o actuadores del proceso, las RTU suelen estar basadas en ordenadores especiales que controlan directamente el proceso tales como autómatas programables o controladores. En el sistema puede haber una o varias RTU.

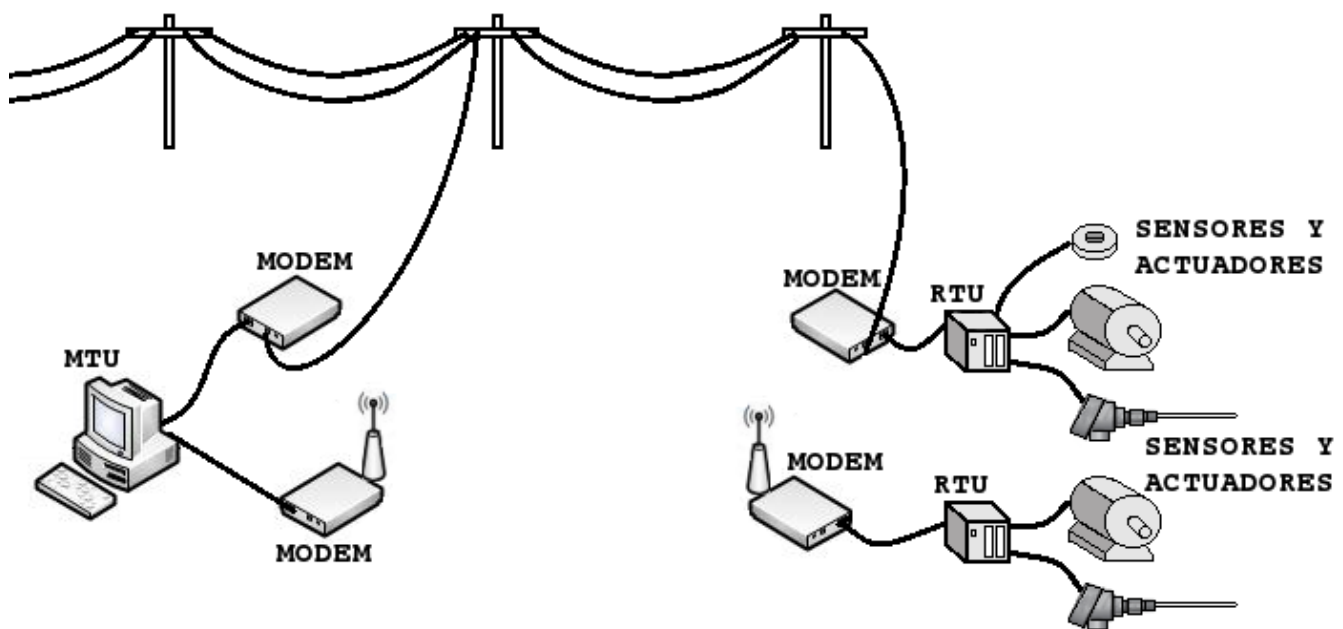


Figura 5.1. Esquema general de un sistema SCADA.

La comunicación entre la MTU y RTU puede ser a través de medios alámbricos o inalámbricos, ya sea propietarios o alquilados a empresas telefónicas y ya que a grandes distancias existe un deterioro de las señales eléctricas, cuando sea el caso será necesario el uso de MODEMs para llevar a cabo la comunicación, en el punto de envío modulando la señal eléctrica en una onda portadora y luego en el destino demodulándola.

La MTU comúnmente está conectada a dispositivos auxiliares como impresoras, memorias de respaldo o alarmas audibles, que se consideran parte del SCADA, pero también mantiene comunicación sobre información de gestión con otras computadoras las cuales no se consideran ya parte del SCADA.

La RTU se comunica con los dispositivos de campo generalmente de manera alámbrica, usualmente brindándoles alimentación a los sensores y a los actuadores de baja potencia, dependiendo de lo crítico de aplicación se acompaña con un UPS (Uninterruptible Power Supply) para evitar fallas por falta de electricidad. Las RTU escanean constantemente a los sensores y actuadores que tiene conectados para coleccionar datos, de igual forma las MTU escanean a las RTU solamente que con una razón mucho más baja que el escaneo que hace la RTU. La cantidad de datos que se mueven en un sistema de comunicación SCADA no tiende a ser muy grande, generalmente una tasa de 300 bits por segundo es suficiente, rara vez se requerirán más de 1200 bits por segundo.

5.2.2 Software SCADA

5.2.2.1 HMI (Human Machine Interface)

Se requiere un software para realizar la interfaz humano máquina, que como su nombre lo indica es la herramienta que permite al operador humano interactuar con el proceso, a través de un entorno gráfico compuesto por una o varias pantallas, en estas, el operario es capaz de monitorear en tiempo real el estado de los dispositivos, el valor de variables físicas, además de poder ejercer acciones de control como activar/desactivar procesos o dispositivos, ajustar variables, etc.

Generalmente la HMI consta de tres pantallas básicas, una con el esquema de la planta o proceso, con objetos gráficos (que pueden incluso ser animados) que representan a los actuadores, sensores, medidores, interruptores y demás elementos presentes en el proceso; así como elementos auxiliares como cajas de texto, cajas de selección, indicadores de colores, barras, etc. En esta pantalla el operador realiza las tareas de control y supervisión para garantizar el buen funcionamiento del sistema, en una segunda pantalla se lleva la gestión de alarmas, las cuales se generan cuando ocurre un suceso no deseado que requiere la atención y/o acción del operario para evitar problemas en el proceso de producción, estas pueden ir desde un aviso de que no hay materia prima para cargar, hasta que la temperatura de un sistema

ha superado el valor normal de operación, se debe llevar un registro de las alarmas que se generaron, hora, razón, etc. Una tercera pantalla será necesaria si se requiere registrar el comportamiento de ciertas variables en el tiempo, para mayor comodidad del usuario es mejor mostrar este comportamiento en gráficos de tendencia.

5.2.2.1.1 Normas para el diseño de la HMI

“La frecuencia de los trastornos musculoesqueléticos y de los problemas relacionados con la fatiga visual y mental suele ser mayor en los trabajadores usuarios de pantallas de visualización que en los que realizan otras actividades tradicionales de oficina”. (Instituto Nacional de Seguridad e Higiene en el Trabajo. Ministerio de Trabajo y Asuntos Sociales, 2005). Dolencias en el cuello, espalda, hombros, brazos y manos en personas que trabajan con pantallas de visualización suelen ser muy comunes y aunque generalmente son provocadas por el mantenimiento de posturas estáticas prolongadas habituales en ese tipo de puestos de trabajo o por posturas incorrectas por el mobiliario o una mala ubicación de la pantalla, también pueden ser originadas por un mal diseño de la HMI, ya que si por ejemplo esta contiene una mala señalización o mensajes confusos, el operario tiene que consultar constantemente documentos para orientarse, lo que provoca continuos giros de cabeza durante la lectura alternativa de la pantalla y los documentos de trabajo, o si la pantalla contiene una inadecuada combinación de colores o de tamaños de texto el operario debe forzar la vista desencadenando visión borrosa e irritación.

Otro trastorno que suele ser bastante frecuente en las actividades realizadas en los puestos de trabajo con pantallas de visualización es la fatiga mental. Este problema tiene diversas causas, desde una organización del trabajo deficiente, como por ejemplo, un ritmo y volumen elevados de trabajo, ejecución de actividades monótonas y repetitivas hasta la inadecuación de los programas informáticos utilizados por el usuario para la realización de su tarea.

Es por ello que es muy importante diseñar una HMI que no solo cumpla con el objetivo de ser la interfaz entre el operador y el proceso sino que también garanticen la ergonomía y seguridad del usuario que pasará frente a la pantalla varias horas al día.

Para tener claro cuáles son los lineamientos que deben de seguirse para un buen diseño de la HMI, la ISO (International Organization for Standardization) que es la federación mundial de organismos nacionales de normalización, creó la norma ISO 9241 que inicialmente se conoció como “Ergonomics

requirements of visual display terminals (VDTs) Used for office tasks” o en español “Requisitos ergonómicos para trabajos de oficina con pantallas de visualización de datos (PVD)” y que actualmente tras posteriores revisiones ha pasado a ser conocida como ISO 9241-210:2010 “Ergonomics of human-system interaction” o traduciendo al español “Ergonomía de la interacción humano-sistema”.

La ISO 9241 es muy amplia y abarca aspectos como mobiliario, entorno o iluminación que están fuera del alcance del diseñador de la HMI, pero también de aspectos que si lo están como son: disposición de los elementos de la interfaz, colores, señalización, tamaños de texto o gráficos, etc. A continuación se describen brevemente estos aspectos que todo diseñador de HMI debe seguir para que su aplicación no represente una fuente de estrés físico o mental para el usuario que trabajará con ella.

Ubicación de los elementos

De manera natural las personas leen la pantalla de forma similar a como lo hacen con un documento impreso, es decir iniciando desde la esquina superior izquierda y luego hacia la derecha y bajando. Este comportamiento determina la mejor forma de colocar los elementos, ya que si lo primero que el usuario ve es la parte superior, ahí deben colocarse la información más importante tal como alarmas o los estados de la operación, siempre teniendo cuidado de no saturar de información esa parte, cuando la información es mucha lo mejor es colocar botones que lleven a ventanas que detallan la información, también se debe tener el cuidado de que las ventanas emergentes (pop-up) que se abran no obstruyan esta parte de la pantalla si los datos son de alta importancia.

Cuando se tienen varias pantallas, se debe de colocar los botones que nos permiten pasar de una página a otra con un nombre claro como “Volver”, “Menú”, “Inicio”, etc. Y colocarlo siempre en el mismo punto de las pantallas, si en algunas está arriba, en otras en medio, a la derecha o a la izquierda será confuso para el operador y perderá tiempo en encontrar el botón.

En Rodríguez (2007) se encuentra la imagen de la Figura 5.2, donde se muestra la disposición recomendada de los elementos en la pantalla.

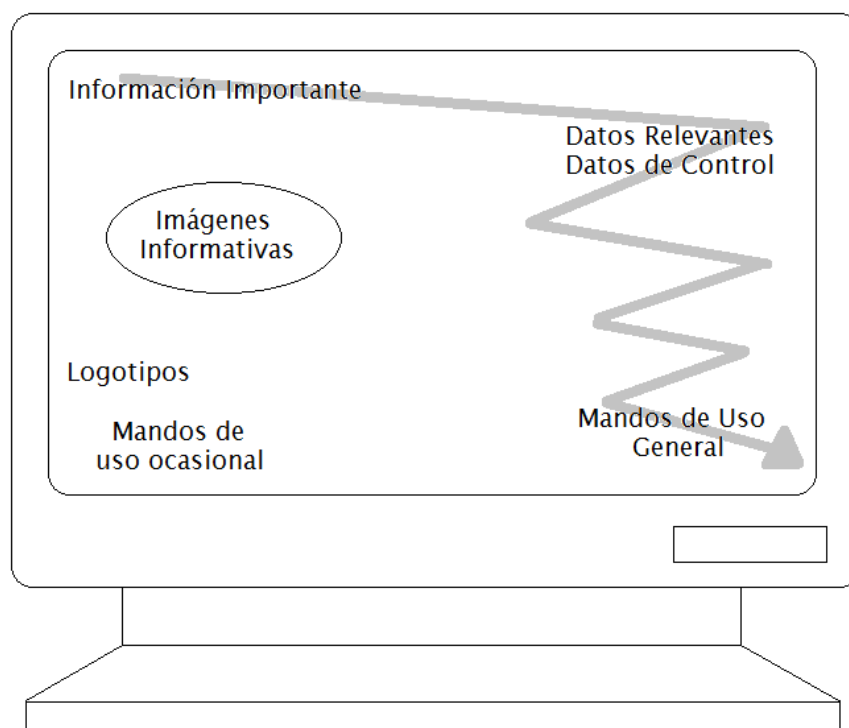


Figura 5.2. Recomendación de colocación de elementos en la pantalla.

Con respecto a la separación que deben tener los elementos, se tiene que tomar en cuenta que se suele asumir que aquellos que se colocan juntos están relacionados, de tal forma que se deben organizar los elementos por categorías y con un espaciado que permita identificar claramente los grupos.

Cuando se coloca algún tipo de indicador este se suele colocar junto con el elemento que lo origina, tal como se muestra en la Figura 5.3(a), sin embargo al estar los elementos en diferentes posiciones y distribuidos por toda la pantalla, se muestra una dispersión de datos a lo largo de esta lo que dificulta su lectura, sobre todo a la hora de querer hacer comparaciones. Para solventar esta problemática se sugiere agrupar los datos en determinadas zonas de pantalla tal como se muestra en la Figura 5.3(b), estas deben estar en la misma posición de la pantalla si hay más pantallas del mismo tipo.

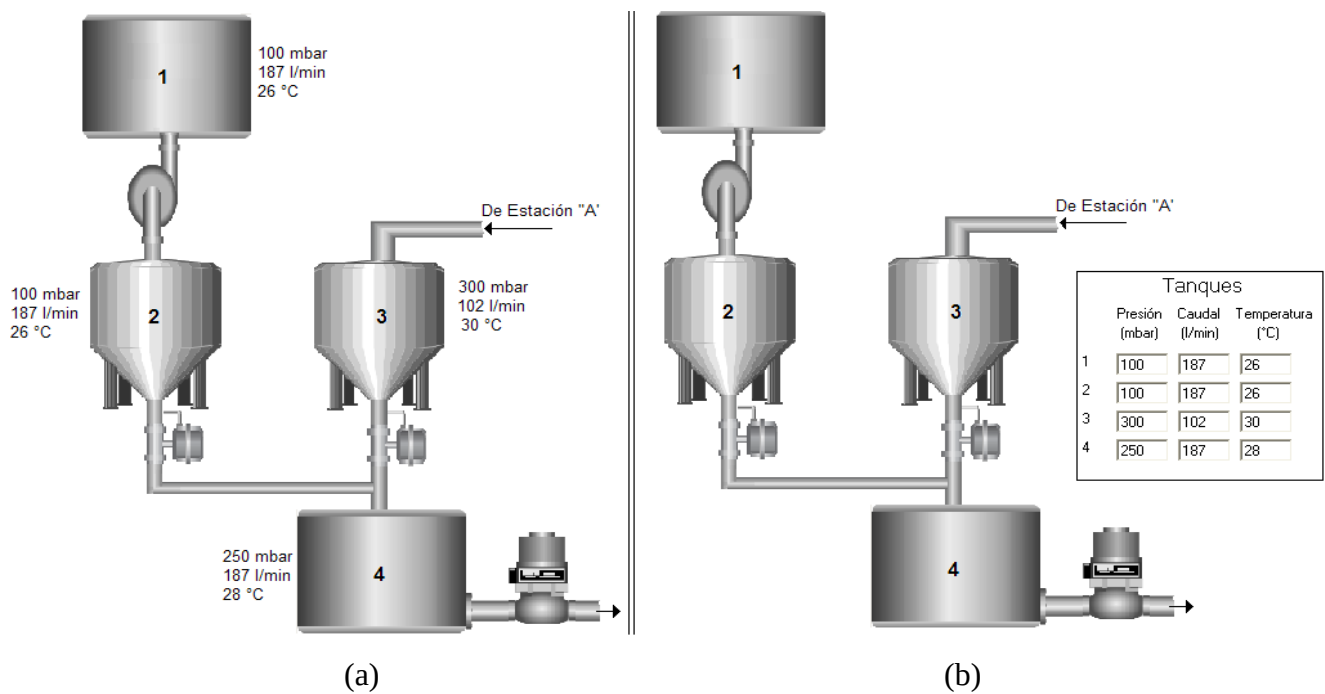


Figura 5.3. Ejemplos de colocación de indicadores.

Representación de los elementos del proceso

Para representar los elementos que componen el proceso a supervisar y controlar pueden utilizarse dibujos, fotografías o elementos gráficos disponibles en librerías dentro de los softwares SCADA, los cuales son muy variados a veces incluso con animación, que representan fielmente casi cualquier instalación real, además de ser configurables para adecuarse al proceso, en la Figura 5.4 se muestran algunos de estos elementos disponibles en las librerías de IGNITION®

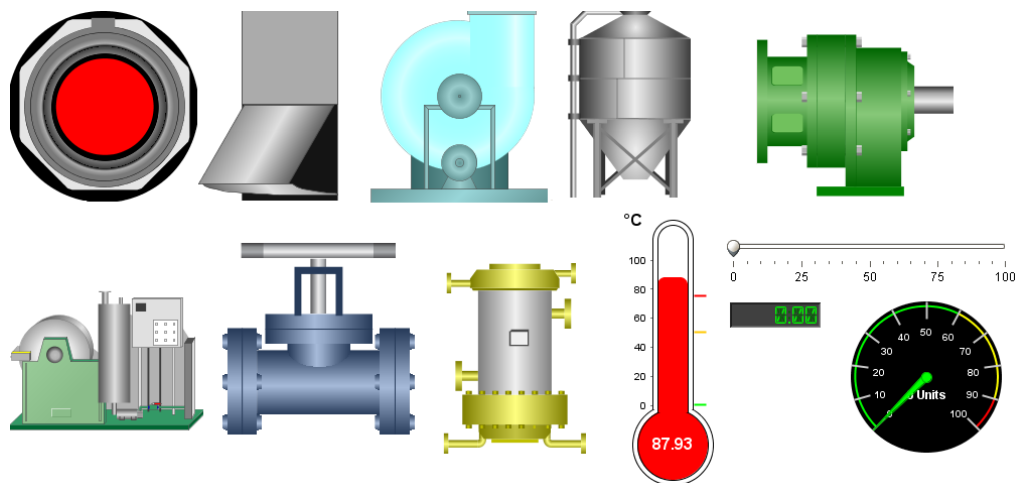


Figura 5.4. Algunos de los elementos gráficos disponibles en IGNITION.

Si bien un gráfico animado tiene mayor impacto visual, usar demasiados elementos así, le restan velocidad a la aplicación, haciendo que existan retrasos tanto a la hora de activar/desactivar algo como a la hora de observarlo, por ejemplo un pulsador, puede ser que ya se haya presionado pero como la animación tarda, se de otro clic desactivándolo inmediatamente después, o puede ser que ya esté en cierto valor un indicador pero la animación retrase su visualización, por lo tanto si se usarán animaciones o que tantas dependerá de la capacidad de procesamiento del sistema en el que se corre la HMI.

Los elementos deben ser representados de manera coherente, es decir por ejemplo la barra deslizante o el medidor en la Figura 5.4, se sabe que como los elementos reales que representan aumentarán su valor desplazándose en el caso de la barra a la derecha y en el caso del medidor con el movimiento de la aguja en el sentido de las agujas del reloj, debiendo estar también las escalas señalizadas para evitar maniobras indebidas por parte del usuario.

Avisos

Con respecto a los avisos que requieran una acción del operario, estos deben de ser coherente, claros, breves y concisos de tal manera que aporten información suficiente para que el operario sepa que hacer de manera rápida sin comprometer el correcto funcionamiento del sistema, en la Figura 5.5 se muestran tres ejemplos de avisos que fallan en algún aspecto, por ejemplo el de la Figura 5.5(a), con esa colocación de botones, el usuario se desconcierta pues como se dijo anteriormente primero se ve la parte superior de la pantalla, a su izquierda en la Figura 5.5(b) se muestra una forma de mostrar el mismo mensaje pero de manera coherente, el de la Figura 5.5(c) no aporta ninguna información de ayuda al usuario para saber qué es lo que debe hacer, el usuario tendrá que ir a buscar de que trata el error para asegurarse de tomar la mejor decisión, a su izquierda en la Figura 5.5(d) se muestra el aviso de una manera más clara para el usuario y que por ende agilizará la toma de decisión, finalmente en la Figura 5.5(e) se muestra un aviso que brinda más información de la necesaria, representando una pérdida de tiempo, pudiendo reducirse como se muestra en la Figura 5.5(f).

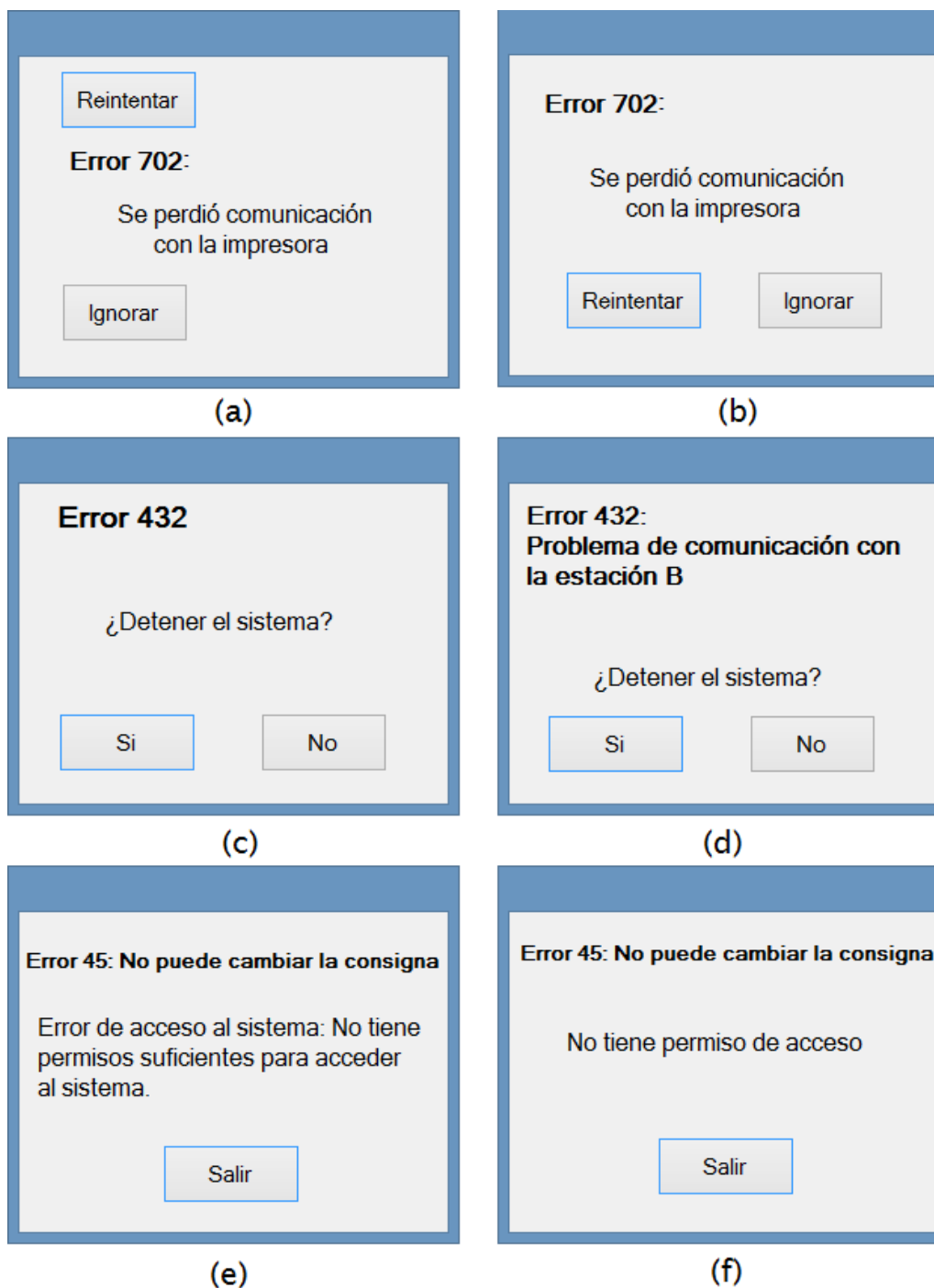


Figura 5.5. Ejemplos de Avisos en la HMI.

Colores

Los colores en la HMI además de hacer un diseño vistoso, nos sirven para aportar información sobre el proceso a supervisar, pero se debe tener cuidado en no abusar de estos y que pasen de ser una herramienta de ayuda a una fuente de confusión o de estrés visual para el operador.

Con respecto al fondo de pantalla o de ventanas secundarias se tienen dos posibilidades: polaridad positiva donde se tiene un primer plano (figuras o caracteres) oscuro sobre un fondo claro y la polaridad negativa que es lo contrario es decir un primer plano claro sobre un fondo oscuro, ambas son igual de válidas a la hora de hacer una HMI, todo dependerá de la posición de la HMI, la cantidad de luz que recibirá, usuario, etc. Así por ejemplo con polaridad positiva al ser la pantalla clara el brillo se aprecia menos y será más fácil equilibrar la iluminación del entorno con ella, en cambio la polaridad negativa al haber menos iluminación, representa una ventaja para las personas con problemas visuales ya que las letras brillan y parecen más grandes al destacar sobre el fondo. Cual sea la opción elegida debe evitarse usar colores de alto contraste, mezclas de tonos azul y verde que hacen la visión de la pantalla incomoda, colores extremos del espectro como rojo y azul que someten a los ojos a esfuerzos excesivos de acomodación y provocan un efecto indeseable de profundidad. Si se están utilizando figuras en color se recomienda que el fondo sea de un color neutro como negro o gris puesto que estos colores maximizan la visibilidad de las figuras en color.

Para mantener la coherencia en la HMI se aconseja que todas las pantallas de una misma categoría tengan el mismo color de fondo, así se genera una respuesta automática por parte del usuario, esto tiene especial importancia en el caso por ejemplo en las alarmas, con solo ver el color del fondo el usuario sabe que hay algo que requiere su atención.

Como fondo de pantalla también pueden colocarse fotografías de la planta pero se corre el riesgo de la pantalla se llene de colores y que los elementos que se coloquen encima no sean tan visibles.

Con respecto al uso de colores en indicadores de estado del proceso, se debe tener especial cuidado en respetar consensos que existen sobre su significado en diversas áreas de trabajo, por ejemplo cuando hablamos de procesos químicos el rojo indica fluidos calientes y el azul fríos, en otras áreas el azul significa aire y el rojo vapor, en estados de maquinaria como motores o bombas el verde significa que el equipo está funcionando y el rojo que están parados, en fin hay tantas variaciones como áreas de trabajo, por lo que se recomienda que según el proceso así se cumpla con lo normalizado o al menos consensuado del área, la dificultad es que no para todas esto está normalizado, para áreas como las tuberías existe la norma DIN 2403 que establece el color de las tuberías para el tipo de fluido y estado en el que se encuentra, en casos como este tenemos lineamientos claros que respetar, pero habrá otras áreas en las

que esto no esté tan definido, por lo que la selección de colores se complica más, además que se debe tener en consideración que existen muchas personas que presentan problemas visuales para distinguir colores, por lo que para evitar confusiones con los colores o simplemente que el usuario no pueda hacer diferencia entre ellos, lo más adecuado es que el color no solo sea la única fuente de información y se acompañe con formas, posiciones o texto como se muestra en la Figura 5.6.

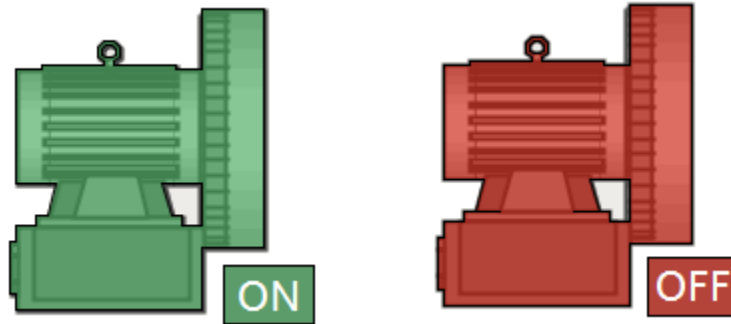


Figura 5.6. Indicador de estado del motor con color y texto.

Se debe evitar juntar indicadores con colores primarios que provocan la aparición de colores complementarios en la imagen retenida en la retina, provocando que se vean colores fantasmas.

Cuando se dibujan algún objeto lo más adecuado es enmarcar su contorno con negro, pues el ojo humano que percibe con precisión los cambios de luminosidad, verá mucho más claro un objeto si su contorno es negro, además de esto se debe evitar en los objetos colores intensos que distraigan la atención del resto de la pantalla, se debe conservar una baja intensidad en todo los objetos de la pantalla, a menos que se quiera destacar algún suceso como una alarma.

Colores en Alarmas

Los colores para alarmas al ser situaciones no deseadas que requieren generalmente atención inmediata para evitar inconvenientes en el proceso, pueden basarse en los colores estandarizados para señales de seguridad tales como Rojo: Peligro, para las alarmas graves que comprometen el proceso y que por lo tanto requieren atención inmediata, Amarillo: Precaución, para alarmas menos graves y Verde: sin problemas para indicar que todo funciona correctamente, de igual forma que con los indicadores de estado del proceso, las alarmas se deben acompañar con texto para asegurarse de que personas con problemas para distinguir colores se den cuenta de la activación de estas, otra herramienta que se utiliza

en el caso de las alarmas para llamar la atención del usuario es la emisión de sonidos.

Con respecto a la combinación de colores para el fondo y primer plano de la alarma, podemos basarnos en la clasificación realizada en su día por el instituto americano de normalización ANSI, (American National Standards Institute) donde definió una clasificación según el grado de apreciación del ojo humano, así un aviso con fondo amarillo y primer plano negro resulta mucho más perceptible que uno en fondo verde y primer plano rojo, en la Tabla 5.1 se muestran los colores de fondo y primer plano ordenados por el nivel de percepción según ANSI:

| Orden | Color | Fondo |
|-------|----------|----------|
| 1 | Negro | Amarillo |
| 2 | Verde | Blanco |
| 3 | Rojo | Blanco |
| 4 | Azul | Blanco |
| 5 | Blanco | Azul |
| 6 | Negro | Blanco |
| 7 | Amarillo | Negro |
| 8 | Blanco | Rojo |
| 9 | Blanco | Verde |
| 10 | Blanco | Negro |
| 11 | Rojo | Amarillo |

Tabla 5.1. Clasificación de niveles de percepción de colores de ANSI.

Para hacer más llamativa una alarma a veces se utilizan mensajes parpadeantes, pero esto no es recomendable ya que el constante cambio de color genera estrés visual.

Otra práctica no recomendada es la aparición de múltiples ventanas cuando se disparan las alarmas, esto puede llegar a bloquear la respuesta del operador, lo mejor es que exista un botón indicador que nos lleve a las alarmas para poder verlas todas e ir atendiendo las más importantes, por lo que otra cuestión importante es categorizar las alarmas y ordenarlas por importancia.

Señales Acústicas en las alarmas

Utilizar colores, textos o ventanas de aviso es de mucha ayuda para alertar al operario de la activación de alguna alarma, pero si este por alguna razón deja de ver la pantalla ya no puede percatarse de si ha ocurrido alguna anomalía, es por ello que las alarmas suelen acompañarse por señales acústicas que aseguran que el operario aunque este distraído o lejos de la pantalla se dé cuenta de que algo anormal ha ocurrido en el proceso.

El sonido que se utiliza debe tener una intensidad y frecuencia adecuados, según el gráfico mostrado en la Figura 5.7 acerca del espectro audible para el ser humano, una conversación normal está en una banda de 400 a 3,000 Hz, por lo que la frecuencia del sonido de la alarma deberá estar por arriba de esta banda para que esta no pase desapercibida.

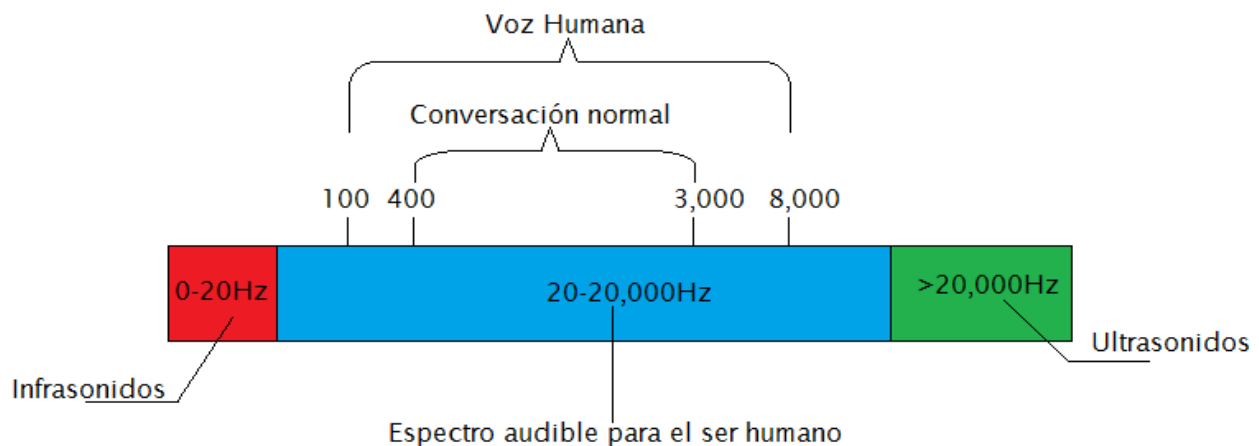


Figura 5.7. Gráfica del espectro audible para el ser humano.

Con respecto a la intensidad, esta debe ser tal que pueda distinguirse claramente del nivel sonoro normal del ambiente de trabajo, recomendándose estar un 10 dB arriba de este, en la Figura 5.8 se muestra un gráfico sobre las fuentes de sonido y su intensidad.

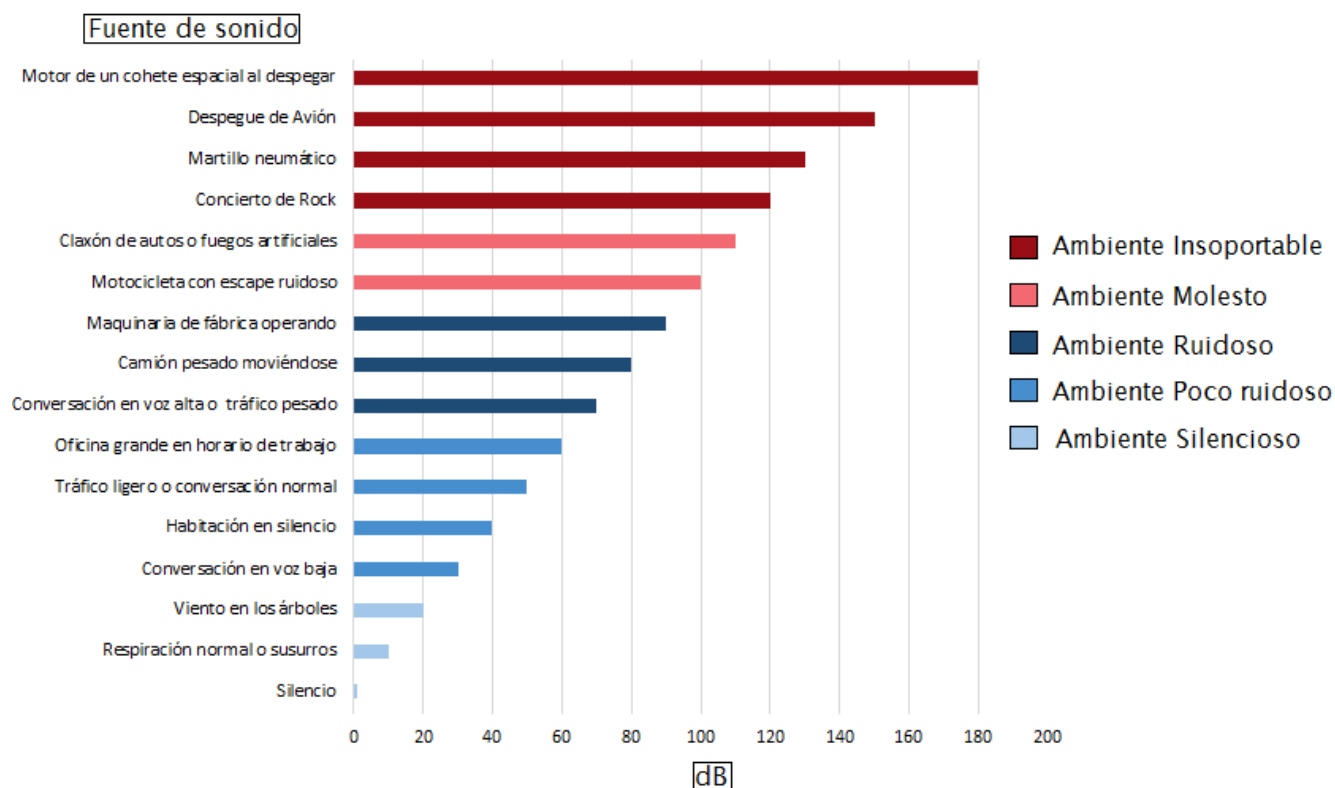


Figura 5.8. Fuentes de sonido y su intensidad en dB.

Para el tipo de señal acústica a utilizar la norma ISO 11429:1996 Ergonomics - System of auditory and visual danger and information signals o en español: Ergonomía - Sistema de señales de peligro y de información auditivas y visuales nos brinda lineamientos como los mostrados en la Tabla 5.2.

| Tipo de señal Acústica | Seguridad | Proceso | Estado |
|-----------------------------|----------------|----------------|----------------|
| Modulante Explosiva | Peligro | emergencia | Fallo |
| Pulsante de tono constante | atención | anormal | anormal |
| Continuo de nivel constante | seguridad | normal | normal |
| Tonos alternos | obligatoriedad | obligatoriedad | obligatoriedad |
| Otros | por acuerdo | por acuerdo | por acuerdo |

Tabla 5.2. Tipos de señales acústicas según la norma ISO 11429:1996.

Así alarmas graves serán de un tono agudo con cadencia rápida y las menos graves con tonos graves y cadencia lenta. Es importante limitar el número de señales acústicas al mínimo, no solo para evitar una mezcla de sonidos inidentificables sino también para no sobrecargar al operador.

Texto: Letras y Números

Tamaño

Con respecto a la configuración y definición de caracteres alfanuméricos, la norma recomienda que la matriz de representación de los caracteres debe estar constituida por un mínimo de 5 x 7 píxeles y cuando se requiera una lectura frecuente de la pantalla, para garantizar la legibilidad del texto, la matriz debe ser de al menos 7 x 9 píxeles (Ver Figura 5.9).

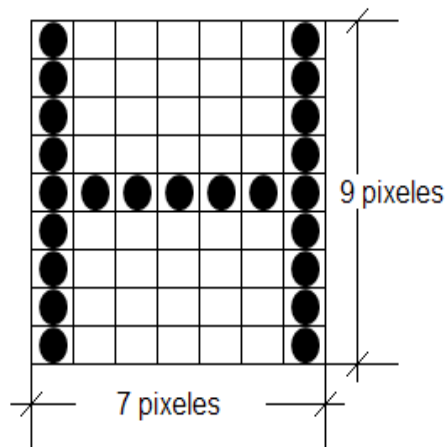


Figura 5.9. Tamaño de matriz de representación de caracteres recomendada.

Este tamaño sería más o menos lo que equivale a un carácter con tamaño de fuente 12, sin embargo la norma también indica que el tamaño de los caracteres dependerá de la distancia de visión, así se tendrá que ajustar el tamaño de los caracteres para que puedan ser leídos de manera cómoda por el operador dependiendo a la distancia en que observará la pantalla.

Otro aspecto importante es el espacio entre los caracteres, si estos se colocan muy separados para algunos parecerá que son independientes y si se ponen muy juntos se tenderán a visualizar como uno solo, por lo que la norma indica que la distancia entre caracteres debe ser equivalente a un píxel y entre palabras a la anchura de un carácter.

En toda la aplicación se recomienda usar dos o tres tamaños de letra de manera coherente en todas las pantallas, es decir títulos siempre en un mismo tamaño, subtítulos en otra y texto común en otra.

Leer texto en mayúsculas genera estrés visual, por lo que lo mejor es utilizar minúsculas y reservar las mayúsculas solo para iniciar el texto y para los títulos, además de evitar subrayar el texto pues vuelve incómoda la lectura.

Tipo de fuente

Con respecto al tipo de fuente de los caracteres se recomienda utilizar tipos conocidos para que no existan inconvenientes a la hora de instalar la aplicación en otras computadoras que puedan no tener instaladas fuentes que no son utilizadas comúnmente.

Dentro de los tipos de fuentes podemos distinguir dos tipos, las que poseen terminales también conocidas como “gracias” o “serifas” y las que no las tienen. En la Figura 5.10 se muestran caracteres con y sin estos terminales. Las fuentes que poseen “serifas” se conocen como tipo “Serif” como por ejemplo “Times New Roman” o “Georgia”, estas son muy utilizadas en libros puesto que ayudan al encaminamiento visual durante la lectura, sin embargo, aquellas que no poseen estos remates conocidas como “sans serif” tales como “Arial” o “Berlin Sans FB”, separan mejor las letras haciendo más claras las palabras, lo cual es muy valioso en una pantalla y por ende se recomienda más el uso de estas.

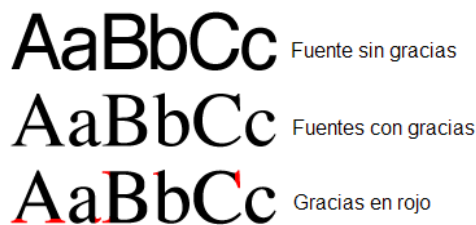


Figura 5.10. Fuentes sin y con “gracias” o “serifas”.

Para toda la aplicación se recomienda usar únicamente un solo tipo de fuente.

Seguridad en los elementos de mando

Como toda actividad humana, el manejo de la HMI está sujeta a errores por parte del operario, por lo que el diseñador debe evitar en la medida de lo posible que estos se produzcan. En el apartado de la

representación de los elementos se indicó que para evitar confusiones los elementos se representen de manera coherente con los dispositivos reales, además de contar con una buena señalización, esto es de mucha ayuda, sin embargo, aún se corre el riesgo de malas maniobras ya sean accidentales o malintencionadas, para evitarlas una buena idea es que aparezca avisos como el mostrado en la Figura 5.11(a), si por ejemplo el operario accidentalmente presionó detener el sistema, el aviso le alerta y pide que confirme o no su acción.

Para aumentar más la seguridad y que a la vez quede registro de quien efectúa las acciones o modificaciones en el proceso lo mejor es utilizar contraseñas de usuario, así al tratar de activar o modificar algo se solicita la identificación como se muestra en la Figura 5.11(b). Si la acción ejecutada no debe realizarse ya sea porque compromete el correcto funcionamiento del proceso o la integridad de las máquinas o personal de planta, entonces se debe denegar en definitiva la acción, como se muestra en la Figura 5.11(c).

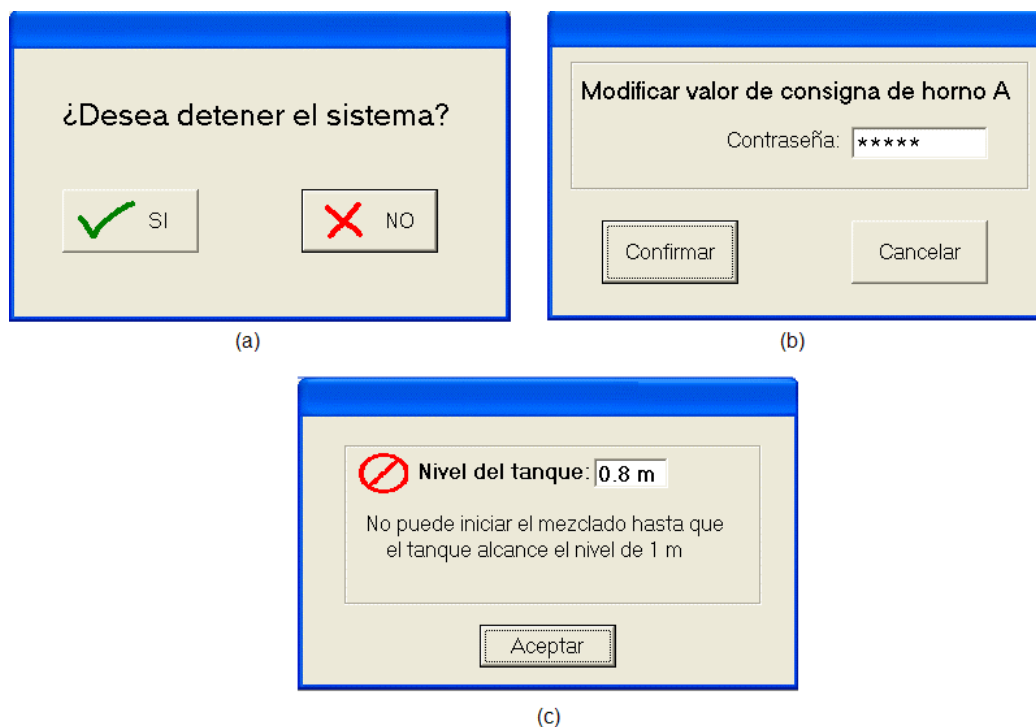


Figura 5.11. Avisos para evitar errores.

5.2.2.2 Comunicación entre Aplicaciones

Entre los métodos más utilizados para comunicar aplicaciones informáticas se tiene a OPC (Object Linking and Embedding for Process Control), el cual es un estándar de interoperabilidad para el intercambio seguro y confiable de datos en el campo de la automatización y otras industrias que consiste en una serie de especificaciones desarrolladas por vendedores, desarrolladores de software y usuarios finales que definen la interfaz entre clientes y servidores y de servidores con servidores, incluyendo el acceso a datos en tiempo real, monitorización de alarmas, eventos y acceso a datos históricos y otras aplicaciones, haciendo que exista independencia de la plataforma utilizada y asegurando un flujo continuo de información entre los dispositivos de múltiples proveedores, para ello basta que los fabricantes de hardware provean un servidor OPC para que sus dispositivos puedan comunicarse con clientes OPC y que los vendedores de software incluyan capacidades de cliente OPC a sus productos para que puedan ser compatibles con cualquier dispositivo OPC, de esta forma el usuario final puede escoger cualquier software de cliente OPC y cualquier hardware OPC y tener la seguridad que se comunicarán perfectamente.

En la Figura 5.12 se muestra el esquema básico del funcionamiento de la tecnología OPC, se tiene una aplicación SCADA corriendo en una computadora que supervisa y controla un proceso gobernado por un controlador lógico programable (PLC). Se necesita un servidor OPC el cual es un programa que convierte el protocolo de comunicación de hardware utilizado por el PLC en el protocolo OPC. La aplicación SCADA que es un software de OPC de cliente usa al servidor OPC para adquirir y enviar información al PLC.

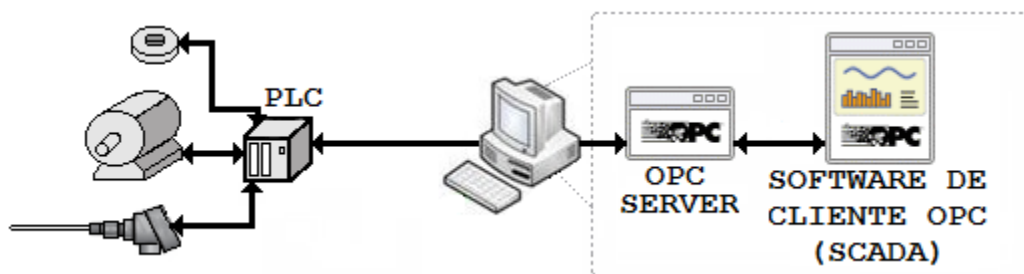


Figura 5.12. Funcionamiento de la tecnología OPC.

Actualmente se tienen dos tipos de tecnologías OPC:

- Clásica:

Este tipo de OPC utiliza la tecnología de Microsoft Windows® de modelo de Objetos de Componentes Distribuidos (COM/DCOM) para el intercambio de datos entre componentes de software. Dentro de las especificaciones más comunes de la norma se tiene:

- OPC DA (Data Access)
- OPC AE (Alarms and events)
- OPC HDA (Historical Data Access)
- OPC DX (Data Exchange)
- OPC XML(Extensive Markup Language)

- Arquitectura Unificada

Las especificaciones OPC clásicas han sido utilizadas satisfactoriamente por la comunidad OPC, pero como toda tecnología debía evolucionar, así en 2008 la fundación OPC lanza OPC UA (Unified Architecture), la cual es una arquitectura orientada a servicios con las siguientes características:

- Integra toda la funcionalidad de las especificaciones del OPC clásico y mantiene compatibilidad con este.
- Brinda independencia de la plataforma, permitiendo a los fabricantes implementar OPC en sistemas que no son de Microsoft®, ahora es posible implementar OPC desde un microcontrolador embebido hasta infraestructura basada en la nube.
- Escalabilidad: Posibilidad de añadir nuevas funciones sin afectar a las aplicaciones existentes.
- Seguridad: brinda encriptación, autenticación y auditoría

5.2.2.3 Almacenamiento de Datos

Para poder llevar a cabo diversos análisis sobre la operación de una planta es necesario disponer de datos almacenados del sistema, para ello se hace uso comúnmente de bases de datos, se tienen los siguientes tipos:

- Base de datos relacional: Es aquella que permiten reflejar estructuras de datos

independientemente del tipo de programas que accede a los datos o de la estructura de estos. Es un conjunto de tablas de datos que contiene campos que sirven de nexos o relaciones y que permiten establecer múltiples combinaciones mediante la utilización de estos nexos. Las combinaciones posibles son prácticamente ilimitadas, solo hay que configurar el método de búsqueda (query) o el tipo de datos que se quiere consultar y aplicarlo a los datos. Este tipo de organización permite la aparición de arquitecturas tipo cliente-servidor, simplificando la gestión de los datos y los programas que trabajan con estos. El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL (Structured Query Language) o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

- Bases de datos industriales: Para sistemas grandes de producción que necesitan almacenar una gran cantidad de variables por segundo, volviéndose por ejemplo al mes una cantidad enorme de datos, resulta inviable utilizar una base de datos relacional común, es por ello que se crearon las base de datos industriales. Desarrollos como IndustrialSQL® de Wonderware® que permite una grabación de datos mayor y reduce espacio en disco. Así *“un servidor dedicado con SQL Server 2000® es capaz de procesar más de 10000 medidas por segundo”* (Rodríguez, 2007).

Para aumentar el rendimiento de las bases de datos se han creado nuevas técnicas tales como:

- OLE DB (Object Linking and Embedding for Databases): conjunto de interfaces basadas en COM que permite hacer accesible los datos a herramientas SQL.
- ADO (ActiveX Data Objects): mecanismo utilizado por programas para interactuar con las bases de datos, ADO es un intermediario entre el programa y la base de datos. El programa no ve la base de datos directamente, sino que hace todo el trabajo a través de ADO.

Gracias a estas tecnologías las bases de datos distribuidas pueden ser accesibles como si formaran una única base de datos local, así desde cualquier panel de operador se puede tener acceso a datos de cualquier lugar de la planta.

6. Celda de Manufactura iCIM 3000®

La celda iCIM 3000® del fabricante FESTO Didactic® con la que cuenta la universidad Don Bosco y cuyo diagrama puede verse en la Figura 6.1, es un sistema de manufactura flexible didáctico capaz de producir una gama de set de escritorios como los mostrados en la Figura 6.2, en los cuales la posición de los elementos no varía sino el material del porta lapicero, algún tipo de maquinado en las piezas y si se coloca o no el lapicero.

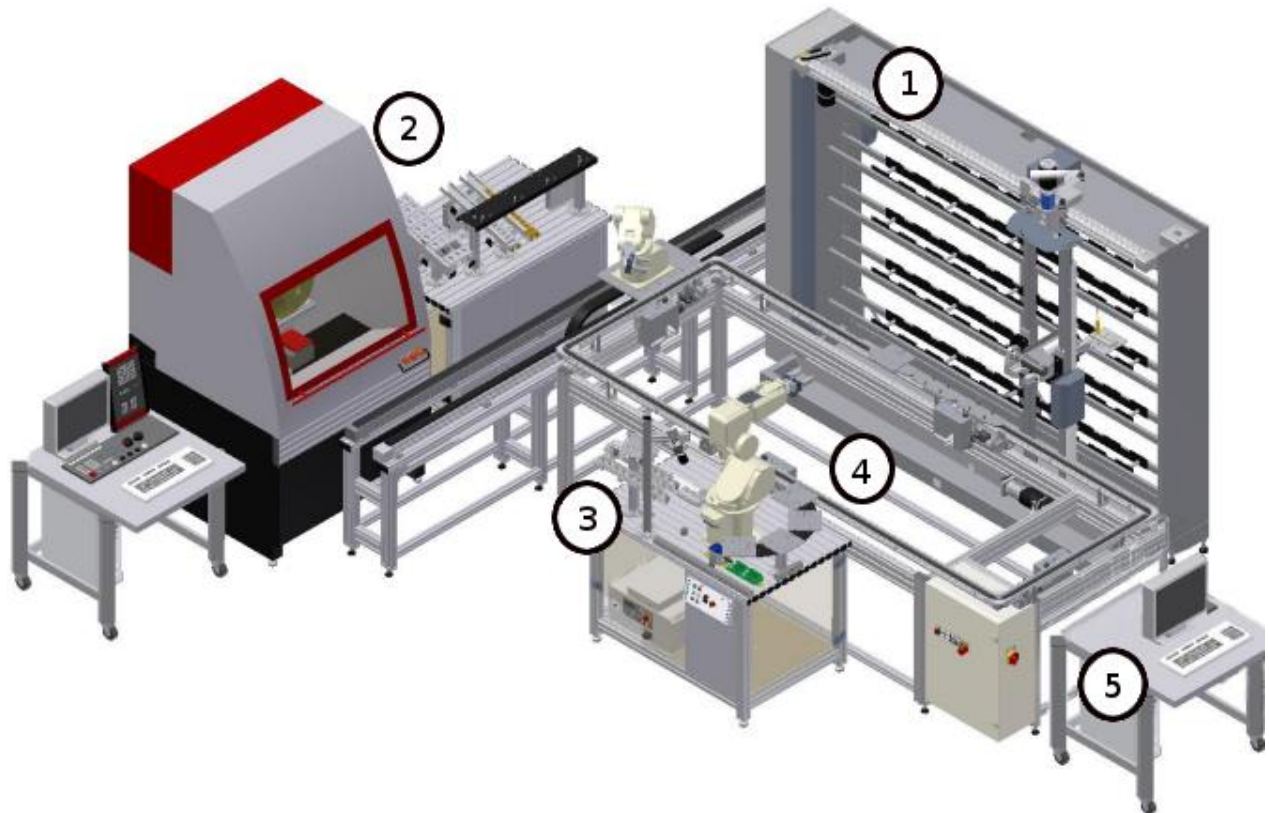


Figura 6.1. Celda de Manufactura iCIM 3000® de la universidad Don Bosco.

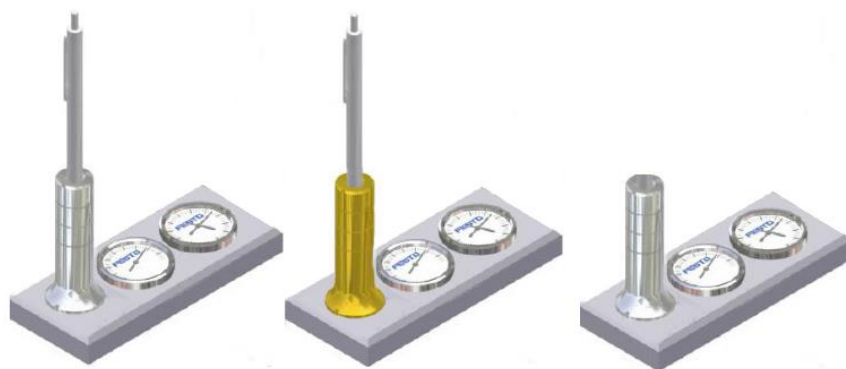


Figura 6.2. Ejemplos de sets de escritorio que se pueden fabricar en el iCIM 3000®.

6.1 Estaciones que la componen

La celda está compuesta de los siguientes elementos:

- Almacén Automático (1): Esta estación AS/RS (Automatic Storage/Retreive System), como su nombre lo indica proporciona y guarda de manera automática los palés con las piezas de trabajo, producto terminado o vacíos por medio de un robot cartesiano. El almacén Posee 40 espacios disponibles.
- Estación de Acoplamiento-CNC (2): Compuesta por un robot Mitsubishi® RV-2AJ que se encarga de tomar material del sistema de transporte o de los compartimentos que posee en su estación e ingresarlos para ser maquinados por la fresadora CNC EMCO® Mill 105 y posteriormente extraerlos y colocarlos en el sistema de transporte, también hay una computadora para operar y configurar a la fresadora.
- Estación de Ensamble por Robot (3): La compone un robot Mitsubishi® RV-3SB que se encarga de tomar las piezas de trabajo del sistema de transporte, ensamblar el set de escritorio y finalmente entregar el producto terminado al sistema de transporte. En la estación también se encuentra una cámara SBOC-Q- R3C-WB de FESTO® que se encarga de captar una imagen de los medidores que la estación SCADA procesa para indicarle al robot como debe girar la pinza para tomar el medidor y que una etiqueta con la palabra FESTO que posee quede alineada con la placa base del set. En el siguiente apartado se describirá con más detalle esta estación, pues es la estación objetivo a la que se le ha realizado su propio sistema SCADA.
- Sistema de transporte (4) : Consiste en una banda transportadora que pasa junto a todas las estaciones de trabajo llevando consigo varios transportes para alojar los palés, en cada estación hay topes que evalúan si se ha requerido un transporte y si es así, lo detienen para que la estación tome o retorne algún palé, cuando la estación esta lista el tope libera el transporte, si se necesita un transporte que porta una pieza específica, los topes verifican de que transporte se trata utilizando un sistema de identificación por radio frecuencia (RFID) de Balluf®, cada transporte posee un identificador con su número.
- Estación SCADA (5): Es la encargada de coordinar el control de palés a las estaciones de trabajo y el arranque de su operación de acuerdo con el plan de procesos ejecutado, en el sistema de ensamble por robot y también se encarga de procesar la imagen para notificar al robot sobre cómo debe orientar los medidores para colocarlos correctamente.

En la Figura 6.3 se muestra el cableado de comunicación de la celda, todas las estaciones están interconectadas a través del protocolo TCP/IP sobre Ethernet con la estación SCADA, excepto el CNC que se comunica a través de puerto serial, además dentro de la banda transportadora el PLC que la gobierna se comunica con los esclavos Wago® que controlan los topes por medio de red Profibus DP y en el almacén hay un gateway que permite interconectar a la red de protocolo CAN Open de los servosistemas de control de los ejes del AS-RS con el maestro Profibus DP presente en el PLC.

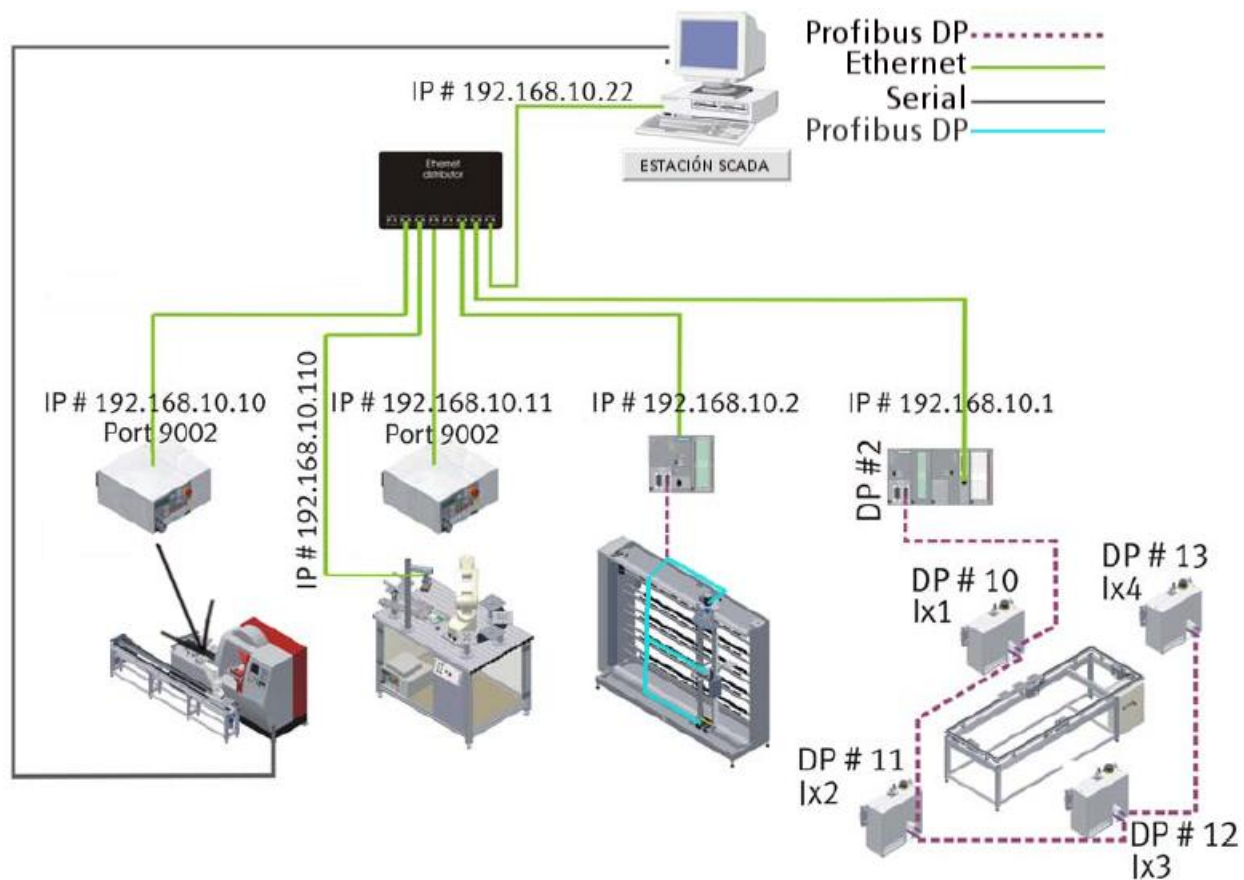


Figura 6.3. Cableado de comunicaciones en la celda iCIM 3000®.

Cada una de estas estaciones posee su propio controlador ya sea un PLC como es la banda transportadora y el almacén, una computadora en el caso del CNC y controladores en el caso de los robots, por lo tanto cada una puede ser usada de manera individual para propósitos de entrenamiento de cada sistema.

6.2 Recursos

Para llevar a cabo la gama de set de escritorios estipulados inicialmente por el fabricante se tienen los siguientes recursos disponibles en el iCIM 3000®:

- Transportes: se cuenta con ocho transportes para movilizar los palés sobre la banda transportadora, cada uno de estos transportes se encuentra identificado como se mencionó anteriormente con una etiqueta RFID.

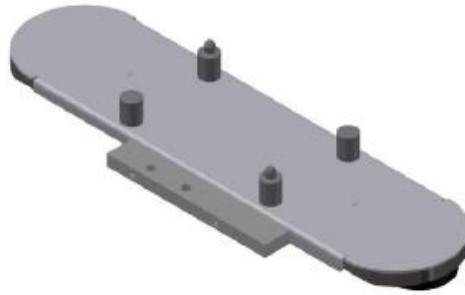


Figura 6.4. Transporte de palés.

- Palés: Se cuenta con dos tipos de palés, los que se utilizan para contener a las placas bases y los otros que contienen a los porta lapiceros (ver Figura 6.5.). Se cuenta con 15 palés para placa base y 25 para porta lapiceros.

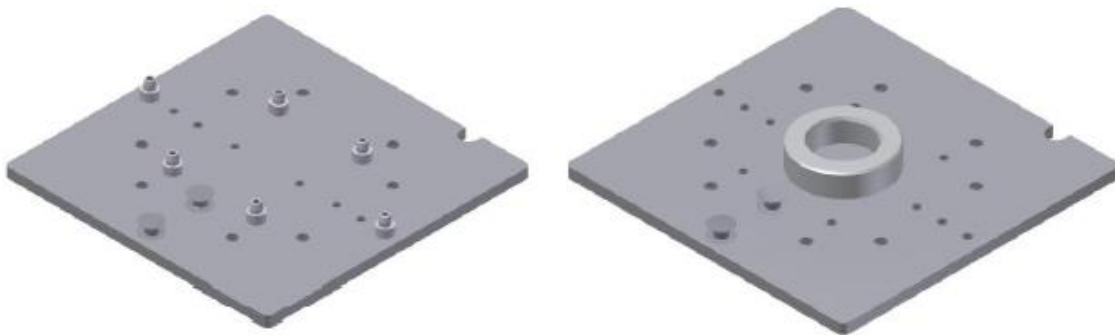


Figura 6.5. Tipos de palés: para placa base y para porta lapiceros.

- Piezas de trabajo:
 - Placas base: Las placas bases existentes son todas de aluminio ya sea sin maquinar o ya maquinadas con diversos tipos de chaflán en sus bordes, cada una de estas está identificada

con un número en la base de datos del almacén.

- Porta lapiceros: hay porta lapiceros tanto de aluminio como de bronce, al no contarse con estación de torno están todos sin maquinar. De igual manera que las placas bases cada tipo de porta lapicero está identificado con un número en la base de datos del almacén.

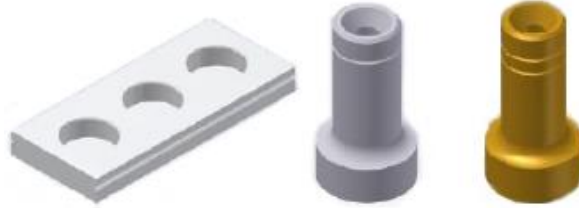


Figura 6.6. Placa base de aluminio sin maquinar, porta lapicero de aluminio y de bronce.

- Medidores: El set de escritorio lleva en la placa base dos tipos de medidores analógicos del mismo tamaño: un termómetro y un higrómetro.



Figura 6.7. Termómetro que se coloca en el set de escritorio.

- Lapiceros: para completar el set de escritorio, se coloca un lapicero estándar en el porta lapicero colocado en la placa base.

El set de escritorio completo estándar se muestra en la primera imagen de la Figura 6.2.

6.3 Sistema SCADA del iCIM 3000®

El Control del iCIM 3000® es llevado a cabo por un sistema SCADA creado con el software CIROS Production®. En la Figura 6.8, se muestra la ventana principal de la HMI, ahí se muestra la vista superior del esquema de la celda, en la parte de arriba se encuentra un botón para acceder a la base de datos del almacén ya sea para consultarla o editarla y a la derecha están las opciones de fabricación: porta lapiceros (función que no puede realizarse para esta celda porque no cuenta con torno), placa base, set de escritorio estándar y productos creados por usuarios, finalmente están las opciones para crear, editar y programar

ordenes de trabajo y para trabajar con la parte de MRP (Manufacturing Resource Planning).

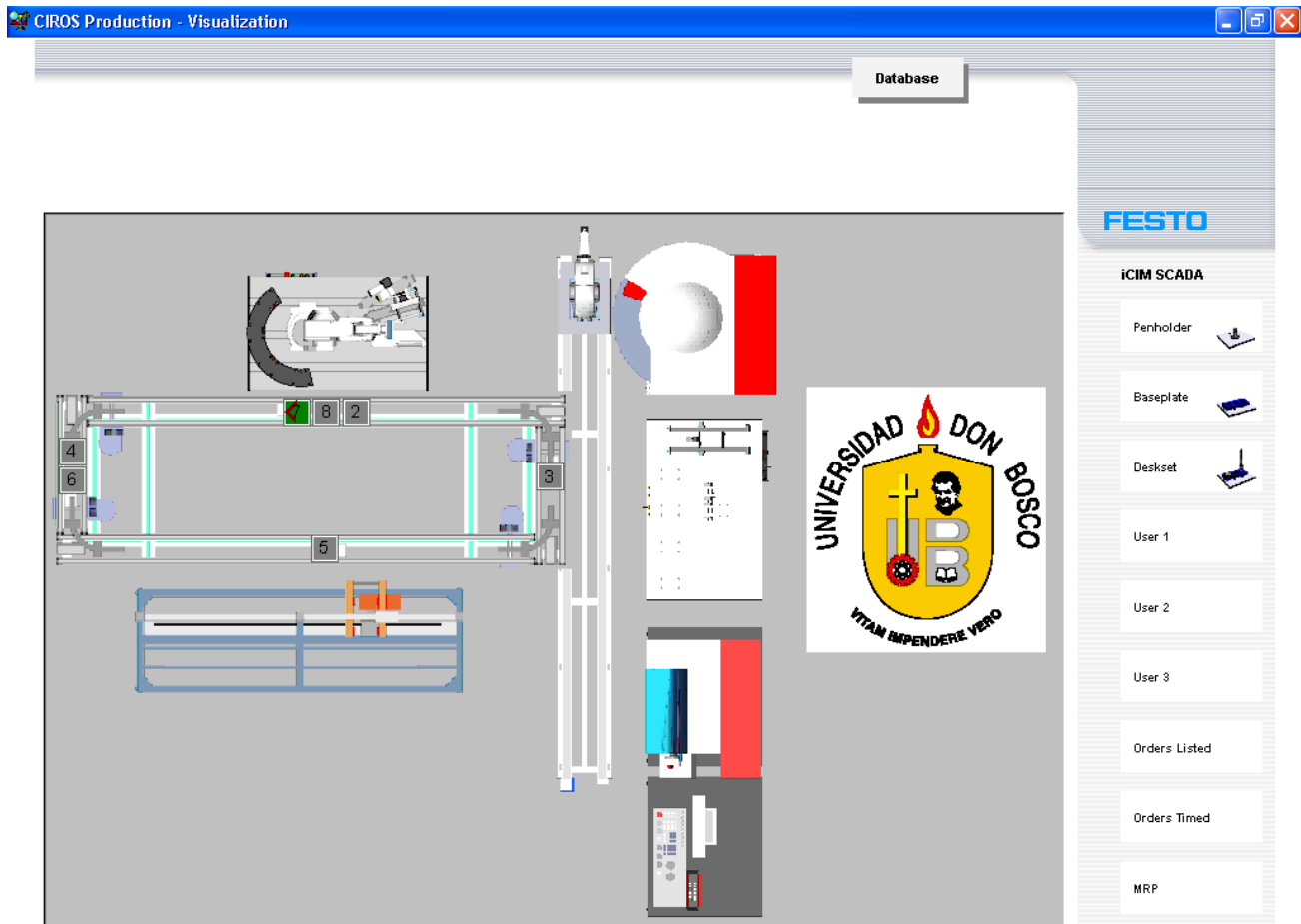


Figura 6.8. Ventana principal del SCADA en CIROS Production.

Cuando se inicia el proceso de manufactura, el SCADA realiza las siguientes operaciones de control de manera automática: activar las estaciones según el proceso, llevar el control de los transportes y palés, comunicación con el software de la cámara para verificar como debe orientar los medidores el robot ensamblador y luego enviarle las indicaciones, actualizar la base de datos del almacén entre otros.

El usuario en pantalla puede observar una animación de cómo se desplazan los transportes en la banda transportadora (ver Figura 6.8), los números de pieza con los que se está trabajando y al llegar a las estaciones se observa en qué lugar son puestos los palés mientras se maquina o se ensambla el producto; también se notifica si se activa alguna alarma en caso de no disponibilidad de materia prima para la fabricación de los productos, problemas con el fresado de las placa base, problemas con la adquisición

de la imagen por parte de la cámara por iluminación inadecuada, que los lugares donde se deben colocar los palés estén ocupados, problemas de comunicación con las estaciones y activación de paros de emergencia en alguna de las estaciones, permitiendo al usuario cancelar el proceso o solventar el problema y reintentar la tarea en la que se había quedado antes de la alarma.

Cuando se extraen o guardan elementos en el almacén la base de datos se actualiza automáticamente.

7. Estación de Ensamble por Robot

7.1 Elementos que la componen

La estación de ensamble por robot se muestra en la Figura 7.1.

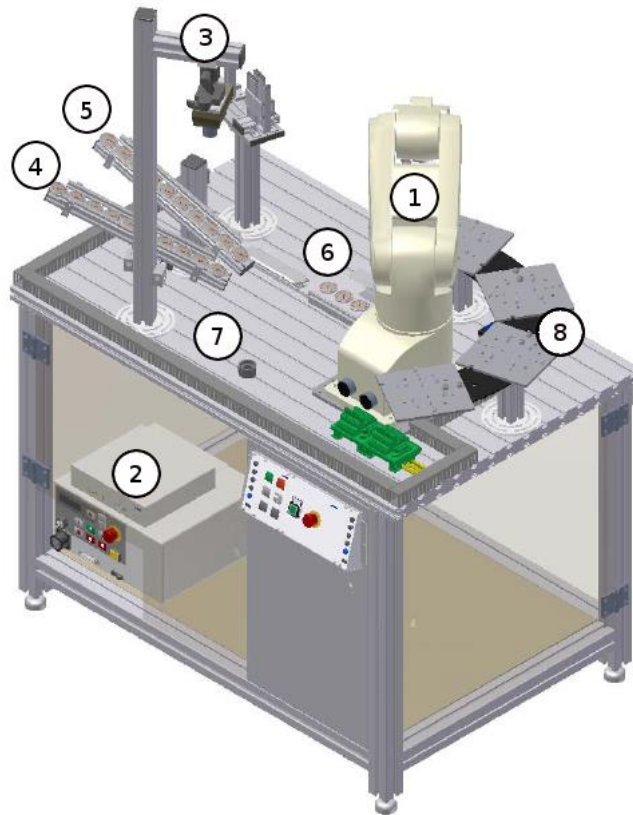


Figura 7.1. Estación por robot de la celda iCIM 3000®.

Está compuesta por un robot industrial Mitsubishi® RV-3SB-S312 (1) con su controlador del mismo fabricante CR2B-574-S312 (2), una cámara FESTO® SBOC-Q- R3C-WB (3), dos recamaras para medidores, uno para higrómetros (4) y otro para termómetros (5); un módulo de ensamble (6), un sostenedor para visión (7) y cuatro receptores para palés (8).

7.1.1 Robot RV-3SB-S312:

El robot RV-3SB-S312 es un robot industrial manipulador de tipo serial, posee 6 grados de libertad, compuesto por articulaciones de un grado de libertad de tipo rotacional como se muestra en la Figura

7.2, con una repetitividad: ± 0.02 mm, velocidad máxima: 5,500 mm/s (limitada a 1,000 mm/s por seguridad para la enseñanza), peso de 37Kg, un alcance de 642 mm (sin pinza) y utiliza como herramienta una pinza de sujeción con funcionamiento electro-neumático.

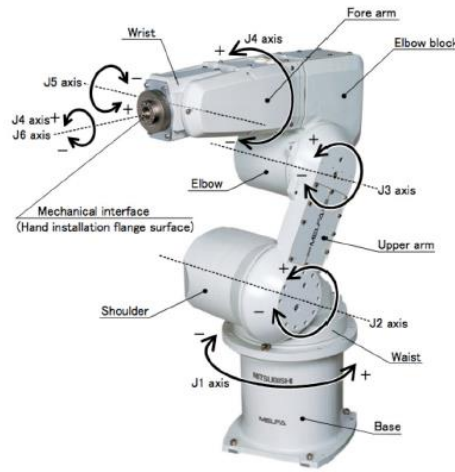


Figura 7.2. Robot Mitsubishi RV-3SB.

El robot RV3SB posee dos sistemas de coordenadas desde los cuales se puede referenciar el movimiento del efector final (pinza), con respecto a la base del robot o con respecto al extremo, en la Figura 7.3 se observa como están dispuesto los ejes para cada caso.

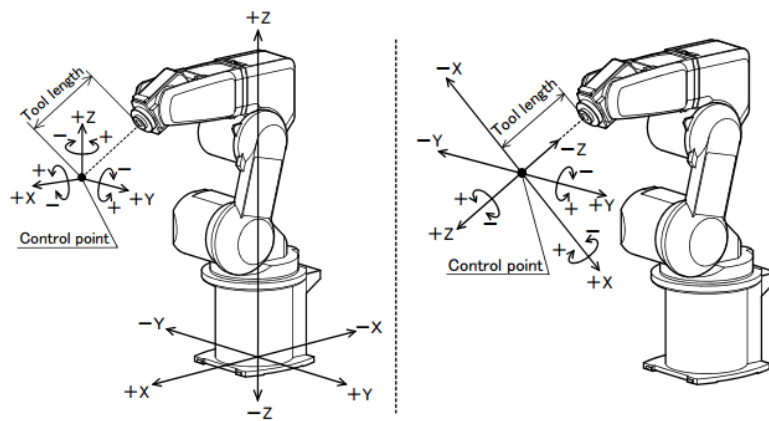


Figura 7.3. Sistemas de coordenadas disponibles en el robot RV-3SB.

7.1.1.1 Controlador CR2B-574-S312

El robot cuenta el controlador CR2B-574-S312 del mismo fabricante, que cumple las funciones de fuente de poder, CPU, control y comunicación del robot.

El controlador puede almacenar hasta 2,500 posiciones y 88 programas, permite programar al robot

utilizando el lenguaje MELFA BASIC IV o MASTER COMMAND, cuenta con consola de aprendizaje o Teaching Pendant (ver Figura 7.4), posee 32 entradas y salidas externas y posibilidad para comunicación Ethernet, RS-232C, RS-422 (exclusivo para el Teaching Pendant), CC-Link y slots para expandir tanto la memoria como las comunicaciones.

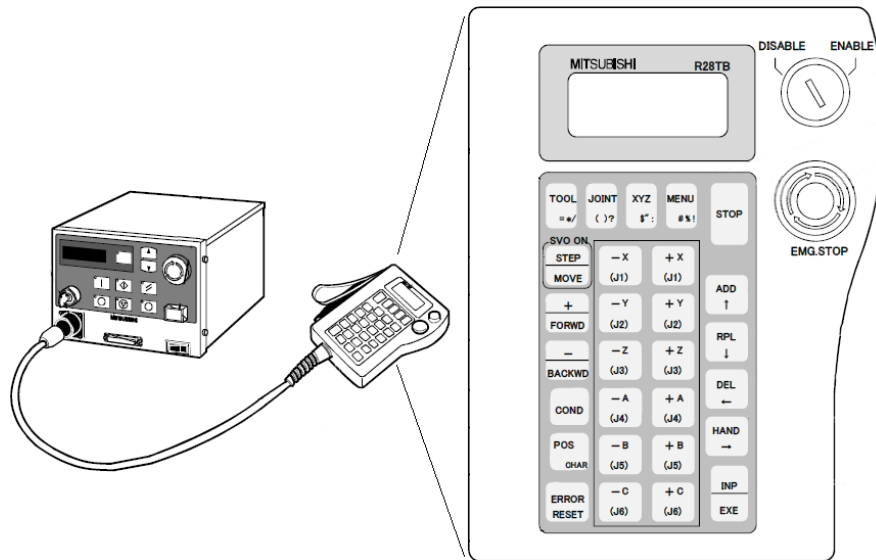


Figura 7.4. Consola de aprendizaje o teaching pendant.

Para funcionar adecuadamente la estación de ensamble con robot RV3SB posee los siguientes programas ya almacenados en la unidad de control:

- MAIN.MB4: Programa principal, inicia automáticamente si el controlador está encendido.
- INIT.MB4: Programa de reinicialización.
- ERR.MB4: Programa para el diagnóstico de errores.
- TOUT.MB4: Programa para diagnóstico de tiempo excedido.
- UBP.MB4: Definición de las variables globales.
- COMM.MB4: Programa de comunicación para CIROS Production®.

7.1.1.1.1 Interfaz Ethernet del controlador CR2B-574-S312

La interfaz Ethernet se puede conectar con cable 10baseT o 10base5, utiliza el protocolo TCP/IP y posee tres modos de comunicación diferentes que se describen brevemente a continuación:

- Función de comunicación con el controlador: con este tipo de comunicación se permite cargar o descargar programas, leer el estado del robot y configurar sus parámetros. En este modo no se puede establecer comunicación con el robot para la ejecución de tareas ya sea que estén estas almacenadas o por medio de envío de comandos. Se puede tener comunicación con hasta 16 clientes.
- Función de enlace de datos: este modo permite el envío y recepción de valores numéricos o datos de posición entre programas del robot y computadoras personales usando el lenguaje MELFA BASIC IV del robot (comandos OPEN/PRINT/INPUT/CLOSE). La comunicación puede ser con hasta 8 aplicaciones especificando un número de COM diferente para cada una.
- Función de control externa en tiempo real: permite escribir y leer la posición del robot en tiempo real, se puede elegir entre escribir/leer la posición de las juntas, posición XYZ o pulso del motor, además de monitorear las señales de entrada/salida. La comunicación debe ser de uno a uno.

7.1.1.2 Lenguaje MELFA BASIC IV

Este lenguaje utiliza un conjunto de instrucciones propias del robot en combinación con sentencias BASIC ya conocidas, su estructura es como casi cualquier otro lenguaje de programación, es decir permite declarar variables y el programa lo forman una secuencia ordenada de instrucciones donde cada línea contiene una sentencia compuesta principalmente por un comando y uno o más parámetros que complementan la acción del comando.

El lenguaje cuenta con un amplio conjunto de instrucciones para realizar tareas como movimientos del robot ya sea por juntas o ejes de coordenadas, cambio de velocidad y aceleración del robot, activación/desactivación de la pinza o comunicación con dispositivos de entrada y salida por medio de alguno de los puertos disponibles en el controlador del robot.

Los programas en MELFA BASIC IV se almacenan con una extensión .MB4 y al ejecutarse lo hacen asociados a un archivo de posiciones previamente almacenado (si el programa no requiere posiciones

este archivo está vacío) que debe tener el mismo nombre que el .MB4 pero con la extensión .POS. En el anexo 14.2 se muestran los códigos del robot que son necesarios para el ensamble del set de escritorio ahí puede observarse que las variables se definen con el comando **DEF** seguido del tipo de variable a definir ya sea de entrada/salida **IO**, entera **INTE**, posición **POS** etc., y luego el nombre de la variable, después aparecen las sentencias del programa compuesto por instrucciones propias del BASIC como **IF ELSE**, **SELECT CASE BREAK** o **GOTO** y de instrucciones propias del robot como **MOV** o **MVS** para moverse a alguna posición, **SPD** para definir la velocidad de interpolación, **ACCEL** para definir la aceleración, etc.

7.1.2 Cámara SBOC-Q- R3C-WB

Es una cámara inteligente con electrónica integrada para procesamiento de imágenes y comunicación, posee un sensor de imagen CMOS a color y un receptáculo con montura CS estandarizado para la lente, que también puede utilizarse como montura C para la lente si se usa el tubo protector de la lente. La cámara es parte del sistema Compacto de Visión SBO...-Q-... y su funcionamiento es llevado a cabo mediante los paquetes de softwares CheckKon®, CheckOpti® y el SBO-DeviceManager®.

7.2 Funcionamiento

Esta estación se encarga de ensamblar los set de escritorio, recibiendo las piezas por medio de la banda transportadora ya sea de la estación CNC o directamente del almacén. En el caso de la fabricación estándar del set de escritorio, el brazo robótico recoge el palé con la placa base y luego el del porta lapicero y los coloca momentáneamente sobre los receptores para palés que posee, cuando ya están ambas piezas, toma la placa base y la coloca en la posición de ensamble, donde es sujeta por un actuador neumático, luego toma de su recámara un termómetro y lo coloca en el sostenedor para visión, la cámara toma una imagen y la envía a la estación SCADA donde el software de la cámara CheckOpti identifica la orientación de la etiqueta FESTO en el medidor, el programa SCADA con esta información calcula como debe moverse el robot para dejar el medidor con la etiqueta de manera recta alineada con el largo de la placa base y le envía la información al controlador del robot para que el robot realice el movimiento necesario al colocar el medidor; este proceso se repite también para el higrómetro, una vez colocados los dos medidores, se procede a tomar un lapicero y colocarlo en el porta lapicero, al finalizar el actuador neumático libera la placa base y el robot la coloca en el palé que había dejado en el receptor y luego

mueve el palé a la banda transportadora para que lo lleve al almacén, finalmente también coloca el palé vacío del porta lapicero en la banda transportadora para ser almacenado.

8. Software IGNITION de Inductive Automation®

IGNITION® es una plataforma de software para crear aplicaciones HMI, SCADA y MES (sistemas de ejecución de manufactura).



Figura 8.1. Logo de plataforma de software IGNITION de Inductive Automation®.

Está diseñado sobre Tecnología Web, tanto la aplicación de diseño como el acceso de los clientes se realiza a través del navegador Web. La tecnología Web Start con la que cuenta permite al usuario acceder al software sin necesidad de complejas instalaciones en cada ordenador.

8.1. Características

- Se vende con licencia de servidor, con una de ellas es posible añadir un número ilimitado de clientes, conexiones, dispositivos y variables (tags).
- Puede ser instalado rápidamente en cualquier sistema operativo (Windows®, Linux® o Mac®), basta con tener instalado un navegador Web y Java.
- Es una plataforma de software modular y escalable, se pueden añadir módulos HMI, SCADA o MES al sistema y trabajarán juntos sin problemas en la misma plataforma.
- La comunicación está protegida por tecnología SSL y se puede integrar Microsoft Active Directory® para definir los perfiles de usuarios.
- Permite el almacenamiento de datos en un formato standard, abierto y fácilmente accesible, ya que el historiador de IGNITION® es compatible con cualquier base de datos SQL. El software incorpora controladores para acceso a MySQL®, Microsoft SQL Server®, Oracle® y PostgreSQL®.
- Conectividad con prácticamente cualquier dispositivo a través de OPC-UA, ofreciendo acceso fácil, robusto y fiable para comunicar con dispositivos industriales.
- Pueden escribirse scripts para cumplir con requisitos individuales y complejos del usuario, para

ello se utiliza el lenguaje Python, el cual es un lenguaje popular, fácil de leer y aprender.

- En general, IGNITION® actúa como un hub efectivo de comunicación en la red.
- Se comunica prácticamente con cualquier servidor OPC, base de datos basada en SQL, PLCs (Allen-Bradley®, Siemens®, Modbus) a través de OPC, soporta cualquier servicio web, se conecta con otros sistemas empresariales (ERP), dispositivos como escáneres de código de barras, báscula y sensores, se ejecuta en cualquier ordenador independientemente del sistema operativo, se conecta a cualquier dispositivo móvil, como teléfonos inteligentes y tabletas inalámbricas además de cualquier pantalla táctil.
- Posee una plataforma en línea llamada Inductive University [20] donde están disponible de manera gratuita cientos de videos de entrenamiento en IGNITION®.
- Puede instalarse una versión de prueba del software con la limitante de solo permitir utilizar el primer elemento gráfico de cada librería y de tener un licenciamiento para dos horas, sin embargo al terminarse ese tiempo el sistema detiene la mayoría de las funciones, pero no se pierde la información almacenada y basta con solamente resetear ese tiempo en la página del Gateway para disponer de dos horas más de trabajo, pudiendo hacer esto cada vez que se acabe el tiempo.

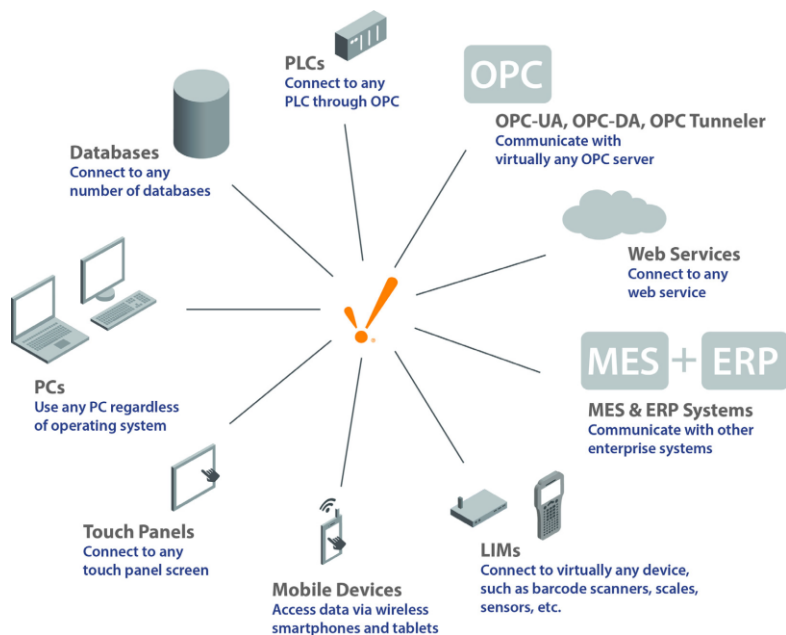


Figura 8.2. IGNITION® como un hub de comunicaciones efectivo en la red.

8.2. Requerimientos del sistema

Sistemas operativos soportados:

- Windows Server 2003/2008/2012
- Windows Vista, 7, 8
- Ubuntu Linux 8.04 o superior
- Otros sistemas operativos que posean Java 8 o superior

Requerimientos (varían según el uso):

- Java edición estándar 8 o superior
- Procesador Dual-Core (32 o 64 bit)
- 4GB RAM
- 10GB libres de disco duro

Bases de dato soportadas

Cualquiera que tenga un driver JDBC, incluyendo:

- Microsoft® SQL Server
- Oracle
- IBM DB2
- MySQL
- PostgreSQL
- Firebird

8.3 Funcionamiento

Como se mencionó anteriormente, IGNITION® no requiere de complejas instalaciones, solamente contar con Java y navegador web, una vez instalado se abre la página web Gateway de la computadora servidora, ya sea desde la misma PC o desde la de un cliente que debe estar en red con la PC servidora, ahí se instalan los módulos necesarios para la aplicación ya sean HMI/SCADA o MES tales como el módulo OPC-UA, drivers para fabricantes de PLC específicos, drivers para protocolos de comunicación específicos, módulo que provee funcionalidades de indicadores de rendimiento como el OEE (Eficiencia

general de los equipos), etc. (Muchos de estos se instalan por defecto al instalar IGNITION®).

Luego se procede a hacer la comunicación con la base de datos, PLCs, sistemas, etc. Si la comunicación es exitosa se muestran como conectados en la página, una vez realizado esto, se ingresa siempre desde la página del Gateway al diseñador donde se crea la HMI, ahí se colocan todos los elementos que se necesiten: displays, indicadores, cajas de texto, tablas, imágenes, etc., se asignan las tags a estos, se configuran los elementos, se realizan scripts si es necesario, etc.

Desde el diseñador y sin terminar aún la aplicación es posible observar en tiempo real el valor de las tags, lo que facilita mucho la creación de la aplicación HMI.

Cuando la aplicación está lista se guarda y ya puede lanzarse para ser vista por los clientes. Al hacer un cambio y guardarlo en el diseñador, los clientes se actualizan automáticamente.

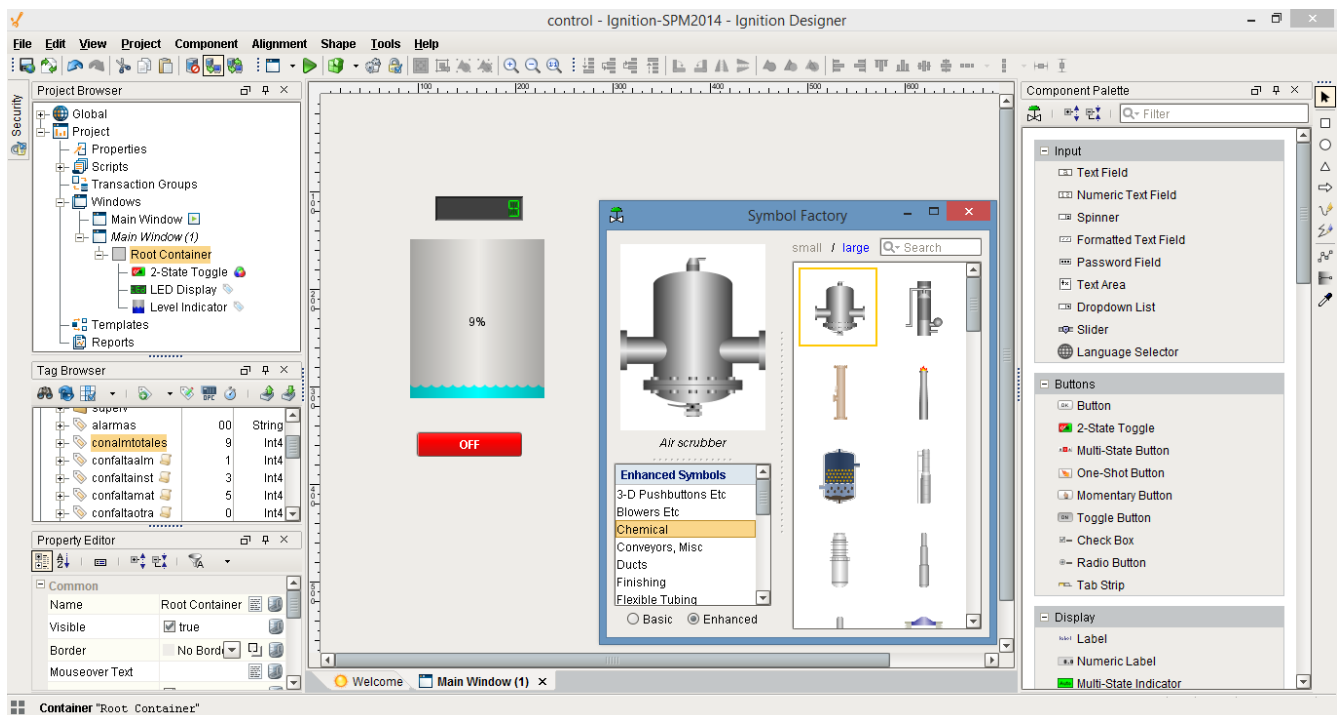


Figura 8.3 Entorno de trabajo de IGNITION®.

9. Desarrollo de las Aplicaciones SCADA

9.1 Comunicación de la Estación de ensamble con la computadora

Tanto el controlador CR2B-574-S312 del robot como la cámara FESTO® SBOC-Q- R3C-WB están en red en la celda a través de Ethernet utilizando el protocolo TCP/IP, por medio de un switch donde se concentran las comunicaciones de todos los elementos, es por esta estructura ya existente que se decidió que la nueva aplicación SCADA se comunicará de la misma forma a través del switch presente en la red.

Con respecto a la cámara debido a que el software de IGNITION® no posee capacidades de procesamiento de imágenes y no se encontró forma de comunicación de este con el software Check Opti® de la cámara, no se estableció comunicación entre la cámara y la computadora, por lo que cuando se trabaje con el SCADA en la modalidad 2, es decir con la celda aislada no se tendrá la función de verificar mediante visión que los medidores queden con la palabra FESTO alineada.

Con respecto al controlador, en el apartado 7.1.1.1.1 se indicó que la interfaz Ethernet del controlador CR2B-574-S312 podía operar de tres formas diferentes, para este proyecto la más conveniente es la función de enlace de datos, pues se requiere una comunicación bidireccional constante entre los equipos, además que en la modalidad 1 la estación se tendrá que comunicar con ambos SCADA.

Para poder trabajar con la función de enlace de datos se deben configurar algunos parámetros en el controlador, pero como la aplicación debe de ser capaz de trabajar en conjunto con el SCADA de CIROS Production® sin afectar su funcionamiento normal, se debía tener especial cuidado en no alterar las configuraciones preestablecidas de esta aplicación, entre ellas la dirección IP (192.168.10.11) o el número de puerto (9002). Como la función de enlace de datos permite trabajar con hasta 8 aplicaciones utilizando un número de puerto diferente, se eligió el número de puerto 10006 para comunicar la computadora con el controlador.

A continuación se describen los parámetros que deben configurarse para lograr una comunicación

Ethernet utilizando la función de enlace de datos:

- NETIP: dirección IP del controlador del robot
- NETMSK: mascara de subred
- NETPORT: número de puerto, para los nueve elementos, el primer elemento es usado para control en tiempo real, los elementos del dos al nueve para software de apoyo o enlace de datos, el rango de valores puede ser de 0 a 32,767 correspondiente a OPT 11-19 de COMDEV
- CPRCE: coloca el protocolo (procedimiento) de comunicación, que puede ser:
 - 0: sin procedimiento. El protocolo es aplicado para usar el software de apoyo de la computadora personal.
 - 1: procedimiento. Reservado
 - 2: enlace de datos. Permite el uso de comandos de comunicación OPEN/INPUT/PRINT/CLOSE
- COMDEV: se establece la definición de los dispositivos correspondientes de COM1 a 8. Estos son usados para el comando de apertura del programa del robot, solo deben ajustarse si se ha especificado el protocolo de enlace de datos.

Los valores de configuración de esta opción deben corresponder con el número de puerto de la siguiente forma:

| n: número de elemento | El nombre del dispositivo puesto por COMDEV(n) | Número de puerto |
|------------------------------|---|--|
| 1 | OPT11 | El número de puerto especificado por NETPORT(2) |
| 2 | OPT12 | El número de puerto especificado por NETPORT(3) |
| 3 | OPT13 | El número de puerto especificado por NETPORT(4) |
| 4 | OPT14 | El número de puerto especificado por NETPORT(5) |
| 5 | OPT15 | El número de puerto especificado por NETPORT(6) |
| 6 | OPT16 | El número de puerto especificado por NETPORT(7) |
| 7 | OPT17 | El número de puerto especificado por NETPORT(8) |
| 8 | OPT18 | El número de puerto especificado por NETPORT(9) |
| 9 | OPT19 | El número de puerto especificado por NETPORT(10) |

Tabla 9.1. Opciones de COMDEV.

Así por ejemplo en el caso de este proyecto en que el número de puerto elegido es el 10006 que se ha especificado en NETPORT(6) debe estar como enlace de datos asignado al COM:6, configurando lo siguiente:

COMDEV(6) = OPT16 *OPT16 debe ser asignado como el sexto elemento de COMDEV.

CPRCE16 = 2 *Configurar el puerto como enlace de datos.

- **NETMODE:** configura la comunicación TCP/IP en función de enlace de datos colocando al controlador como servidor (1) o cliente (0), si se coloca al controlador como cliente debe colocarse en el elemento correspondiente del parámetro NETHSTIP la dirección IP del servidor.

En la Tabla 9.2 se muestra el valor que se colocó a estos parámetros para lograr la comunicación de la computadora con el controlador del robot utilizando la función de enlace de datos en el puerto 10006 y el controlador del robot como servidor. Estos parámetros se configuran utilizando el Teaching Pendant, en el anexo 14.1 se muestra la forma de hacerlo.

| Parámetro | Valor configurado solo para el SCADA de CIROS Production® | Valor configurado para la nueva aplicación SCADA |
|------------------|--|---|
| NETIP | 192.168.10.11 | Sin cambio |
| NETMSK | 255.255.255.0 | Sin cambio |
| NETPORT | 9000,9002,9003,9004,9005,10006,10007,10008,10009 | Sin cambio |
| CPRCE16 | 0 | 2 |
| COMDEV | RS232C,OPT11, , , , , , , | RS232C,OPT11, , , , OPT16, , , |
| NETMODE | 0,1,1,1,1,1,1,1,1 | Sin cambio |

Tabla 9.2. Ajuste de parámetros en el robot del controlador.

A la computadora con las nuevas aplicaciones SCADA se le colocó la dirección IP 192.168.10.33 perteneciente a la misma subred del controlador del robot.

9.1.1 Programas en el controlador del robot

Para realizar los ensambles de los set de escritorio indicados por el plan de procesos que se ejecuta desde el SCADA de FESTO Didactic® creado con CIROS Production® el controlador del robot posee los siguientes programas en MELFA BASIC IV:

- “MP.MB4”: utilizado para mover los palés de cualquiera de los receptores para palés a la banda transportadora y viceversa
- “MBP.MB4”: utilizado para mover la placa base colocada en cualquiera de los palés a la posición de ensamblado y viceversa
- “MINST. MB4”: utilizado para colocar el termómetro e higrómetro de los alimentadores a la posición de prueba de visión y luego a la placa base ubicada en la posición de ensamble
- “MPH.MB4”: utilizado para mover el porta lapicero colocado en cualquiera de los palés a la placa base ubicada en la posición de ensamble
- “ASMP.MB4”: utilizado para colocar el lapicero en el porta lapicero colocado en la placa base ubicada en la posición de ensamble

Desde el plan de procesos se le envía el origen y destino de los elementos a mover.

Cada uno de estos programas está acompañado de un archivo con extensión .POS y con el mismo nombre donde se encuentran guardadas las posiciones que necesitan para trabajar, así se debe contar también con los archivos “MP.POS”, “MBP.POS”, “MINST.POS”, “MPH.POS” y “ASMP.POS”.

Para la modalidad 1 donde la aplicación SCADA extra realiza solamente tareas de monitorización trabajando en conjunto con las demás estaciones y comandada por el SCADA de FESTO Didactic®, fue necesario modificar estos 5 programas y un plan de procesos nuevo que los invocara, para ello se crearon respectivamente los programas “TMP.MB4”, “TMBP.MB4”, “TINS.MB4”, “TMPH.MB4”, “TASM.MB4” y sus archivos de posiciones “TMP.POS”, “TMBP.POS”, “TINS.POS”, “TMPH.POS” y “TASM.POS”.

Los .MB4 realizan las mismas funciones que los originales con la diferencia de que estos poseen además instrucciones OPEN/CLOSE para abrir y cerrar el puerto de comunicaciones 10006 que se configuró, y la instrucción PRINT para enviar los datos por el puerto tales como códigos para indicar el estado de la

estación, estado de los sensores, posiciones del robot y la activación de alarmas, los .POS tienen exactamente las mismas posiciones que los originales pero para que los .MB4 operen correctamente debe haber .POS con el mismo nombre; también se creó un programa nuevo llamado “TENC.MPB4”, este simplemente envía desde la estación de ensamble al SCADA extra los códigos que se le indican desde el plan de procesos en dos momentos durante el proceso de producción, al principio para indicar al SCADA extra que el proceso ha iniciado aunque la estación de ensamble aún no comience y cuando finaliza el ensamble. El .POS de este archivo está en blanco pues el robot no se mueve a ninguna posición cuando los ejecuta pero es necesario que exista. El código fuente de estos seis programas puede verse en el anexo 14.2.1.

Para la modalidad 2 donde la estación de ensamble trabaja de manera aislada y la aplicación SCADA extra realiza tanto tareas de control como de supervisión, fue necesario crear otro programa que contuviera a los programas originales a excepción del de mover palé puesto que ya no hay interacción con la banda transportadora, además del programa de posiciones respectivo con todas las posiciones de los programas anteriores más una extra para poder ensamblar el set de diferente manera. En este nuevo programa también se incluyen instrucciones para abrir y cerrar el puerto de comunicaciones, enviar códigos para indicar el estado de la estación o la activación de alarmas y en este caso también la instrucción INPUT para leer ordenes enviadas desde el SCADA extra. El código fuente de este programa llamado “ABC4” se muestra en el anexo 14.2.2.

En el caso de la modalidad 1 los programas envían un paquete de datos cada vez que ocurre un cambio de posición o se ha realizado una tarea, el paquete consiste primeramente en enviar el estado de los 10 sensores de la estación (0: no detecta, 1: si detecta), uno tras otro en el siguiente orden:

- Sensor de portador de palé 1
- Sensor de portador de palé 2
- Sensor de portador de palé 3
- Sensor de portador de palé 4
- Sensor de placa base en la posición de ensamble
- Sensor de termómetros
- Sensor de higrómetros

- Sensor de lapiceros
- Sensor de cilindro de sujeción 1
- Sensor de cilindro de sujeción 2

Inmediatamente después de estos se envía el carácter “P” seguido de la posición del robot la cual puede ser una de las siguientes:

- 1: posición sobre portador de palé 1
- 2: posición sobre portador de palé 2
- 3: posición sobre portador de palé 3
- 4: posición sobre portador de palé 4
- 5: posición de ensamblado
- 6: posición sobre la banda transportadora
- 7: posición para tomar termómetros
- 8: posición para tomar higrómetros
- 9: posición de prueba de visión para medidores
- A: posición para tomar lapiceros
- I: posición inicial

Luego se envía el carácter “A” seguido del código que indica el estado de la estación de ensamble o las alarmas, este consta de dos dígitos, en la Tabla 9.3 se muestran los códigos con su significado, siendo los que están con fondo gris o blanco los enviados en la modalidad 1.

| Código | Significado |
|---------------|--|
| 01 | No hay palé en la posición indicada |
| 02 | No hay palé con placa base en la posición indicada |
| 03 | No hay placa base en la posición de ensamble |
| 04 | No hay palé donde guardar el set |
| 05 | Hay una placa base en la posición de ensamble |
| 06 | No hay higrómetros |
| 07 | No hay termómetros |
| 08 | No hay palé con porta lapicero |

| | |
|------|---|
| 09 | No hay lapiceros |
| 10 | Está ocupada la posición del palé |
| 11 | No se movió palé válido |
| 12 | No se usa. (estaba reservada para problemas en la cámara) |
| 13 | El robot recibió órdenes no válidas, finalizó el programa |
| 14 | Problemas en el SCADA de Ciro's Production |
| 15 | Ensamble terminado correctamente |
| 16 | Arranca el proceso de ensamblado |
| 17 | Programa del robot finalizado |
| 18 | Proceso de producción iniciado |
| 19 | Se está ejecutando el programa de mover palé |
| 20 | Se está ejecutando el programa de mover placa base |
| 21 | Se está ejecutando el programa de mover medidores |
| 22 | Se está ejecutando el programa de mover porta lapicero |
| 23 | Se está ejecutando el programa de mover lapicero |
| Otro | La estación aun no confirma que la producción haya iniciado |
| | Falla de comunicación, no se recibieron datos validos del robot |

| | |
|--|-------------------|
| | Modalidad 1 |
| | Modalidad 2 |
| | Ambas modalidades |

Tabla 9.3. Significado de los códigos enviados de la estación de ensamble a la computadora.

Finalmente se envía el carácter “F” para indicar fin del paquete.

En el caso de la modalidad 2, el programa solamente envía el código de dos dígitos que indica el estado de la estación o las alarmas a la computadora, en la Tabla 9.3, se muestran con fondo blanco o celeste los códigos enviados en la modalidad 2, siempre seguidos del carácter “F” para indicar el fin del paquete.

En esta modalidad también es necesario enviar órdenes de la computadora a la estación, para enviar la

orden para ensamblar algún set de escritorio se envía un código de tres dígitos cuyo significado se muestra en la Tabla 9.4.

| Código enviado | | | Significado |
|-----------------------|----------------------------------|------------------------------------|--|
| <i>orden</i> | <i>Posición de la placa base</i> | <i>Posición del porta lapicero</i> | |
| 1 | 1 | 1 | Ensamblar set de escritorio estándar con lapicero tomando materiales de las posiciones indicadas |
| | 2 | 2 | |
| | 3 | 3 | |
| | 4 | 4 | |
| 2 | 1 | 1 | Ensamblar set de escritorio estándar sin lapicero tomando materiales de las posiciones indicadas |
| | 2 | 2 | |
| | 3 | 3 | |
| | 4 | 4 | |
| 3 | 1 | 1 | Ensamblar variante de set de escritorio estándar sin lapicero tomando materiales de las posiciones indicadas |
| | 2 | 2 | |
| | 3 | 3 | |
| | 4 | 4 | |
| 4 | 0 | 0 | Finalizar el programa en el robot |

Tabla 9.4. Códigos enviados desde la computadora con el SCADA operando en la modalidad 2 a la estación de ensamble.

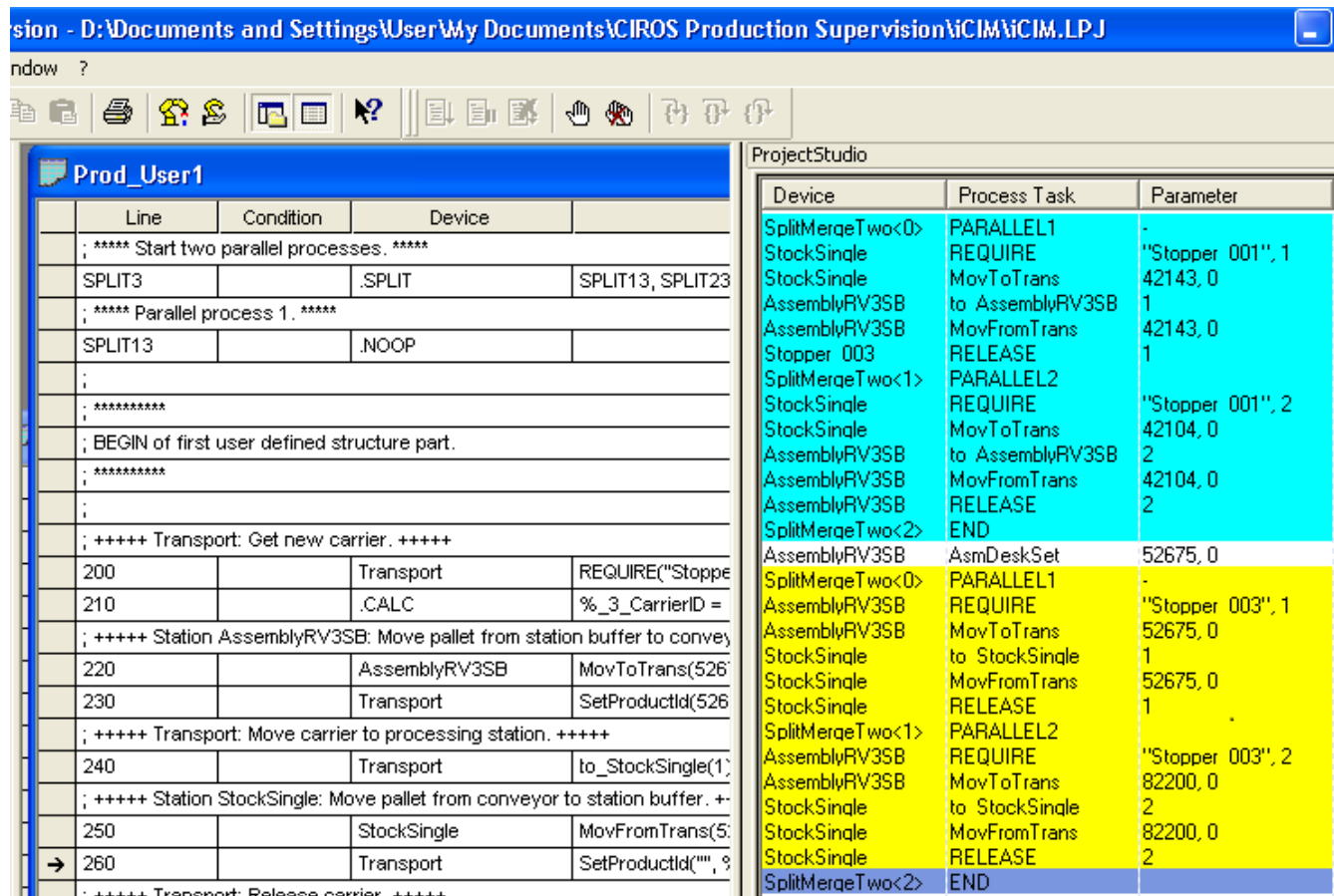
El código se envía seguido del carácter “F” para indicar el fin del paquete.

Cuando ocurre una alarma es posible tanto reintentar la realización de la tarea como finalizar el programa del robot, si se desea finalizar el programa tal como se indica en la Tabla 9.4 la computadora envía a la estación el código “400”, pero si se va a reintentar envía el código “RE”.

Los programas a ser ejecutados por el robot se crean y almacenan en el controlador del robot por medio del programa CIROS Studio® de FESTO Didactic®.

9.2 Plan de procesos realizado en CIROS Production®

Para la modalidad 1 donde el SCADA extra trabaja en conjunto con el SCADA de CIROS Production® es necesario crear en este un plan de procesos utilizando un lenguaje propio de CIROS Production® para el ensamble de un set de escritorio estándar con lapicero como el mostrado en la Figura 9.1.



| Line | Condition | Device | Parameter |
|---|-----------|---------------|--------------------------|
| ; ***** Start two parallel processes. ***** | | | |
| SPLIT3 | | .SPLIT | SPLIT13, SPLIT23 |
| ; ***** Parallel process 1. ***** | | | |
| SPLIT13 | | .NOOP | |
| ; ***** | | | |
| ; BEGIN of first user defined structure part. | | | |
| ; ***** | | | |
| ; ***** Transport: Get new carrier. ***** | | | |
| 200 | | Transport | REQUIRE("Stopper 001", 1 |
| 210 | | .CALC | %_3_CarrierID = |
| ; ***** Station AssemblyRV3SB: Move pallet from station buffer to convey | | | |
| 220 | | AssemblyRV3SB | MovToTrans(526 |
| 230 | | Transport | SetProductId(526 |
| ; ***** Transport: Move carrier to processing station. ***** | | | |
| 240 | | Transport | to_StockSingle(1 |
| ; ***** Station StockSingle: Move pallet from conveyor to station buffer. + | | | |
| 250 | | StockSingle | MovFromTrans(5 |
| 260 | | Transport | SetProductId("", ? |
| ; ***** Transport: Release carrier ***** | | | |

| Device | Process Task | Parameter |
|------------------|------------------|------------------|
| SplitMergeTwo<0> | PARALLEL1 | - |
| StockSingle | REQUIRE | "Stopper 001", 1 |
| StockSingle | MovToTrans | 42143, 0 |
| AssemblyRV3SB | to AssemblyRV3SB | 1 |
| AssemblyRV3SB | MovFromTrans | 42143, 0 |
| Stopper 003 | RELEASE | 1 |
| SplitMergeTwo<1> | PARALLEL2 | - |
| StockSingle | REQUIRE | "Stopper 001", 2 |
| StockSingle | MovToTrans | 42104, 0 |
| AssemblyRV3SB | to AssemblyRV3SB | 2 |
| AssemblyRV3SB | MovFromTrans | 42104, 0 |
| AssemblyRV3SB | RELEASE | 2 |
| SplitMergeTwo<2> | END | - |
| AssemblyRV3SB | AsmDeskSet | 52675, 0 |
| SplitMergeTwo<0> | PARALLEL1 | - |
| AssemblyRV3SB | REQUIRE | "Stopper 003", 1 |
| AssemblyRV3SB | MovToTrans | 52675, 0 |
| StockSingle | to StockSingle | 1 |
| StockSingle | MovFromTrans | 52675, 0 |
| StockSingle | RELEASE | 1 |
| SplitMergeTwo<1> | PARALLEL2 | - |
| AssemblyRV3SB | REQUIRE | "Stopper 003", 2 |
| AssemblyRV3SB | MovToTrans | 82200, 0 |
| StockSingle | to StockSingle | 2 |
| StockSingle | MovFromTrans | 82200, 0 |
| StockSingle | RELEASE | 2 |
| SplitMergeTwo<2> | END | - |

Figura 9.1. Plan de procesos para ensamble de set de escritorio estándar con lapicero.

Las instrucciones simplemente son extraer en paralelo del almacén una palé con porta lapicero y un palé con placa base y enviarlos a la estación de ensamble para ser ensamblados, y luego en paralelo enviar al almacén el set armado y el palé de porta lapicero vacío, sin embargo para que exista comunicación con la aplicación SCADA extra es necesario que las macro tareas de la estación de ensamble (AssemblyRV3SB): “AsmDeskSet”, “MovFromTrans” y “MovToTrans” llamen a los nuevos programas que se crearon en el robot para comunicarse con el otro SCADA, por lo que para no alterar a las originales se crearon otras macro tareas en la librería donde están las originales y se enlazaron al proyecto, estas

nuevas macro tareas llamadas respectivamente “AsmDeskSet1”, “MovFromTrans1” y “MovToTrans1”, en lugar de llamar a los programas originales del robot: “MP”, “MBP”, “MINST”, “MPH” y “ASMP” llama a los modificados “TMP”, “TMBP”, “TINS”, “TMPH” y “TASM”, además para indicar a la celda que el proceso de fabricación ha iniciado, al principio del plan de procesos se manda a ejecutar en la estación de ensamble el programa “TENC” solamente para enviar el código al SCADA extra para que sepa que aunque la estación de ensamble aún no esté en operación ya se inició el proceso de producción, los materiales llegan a la estación, el set se ensambla y se coloca en la banda transportadora, luego se coloca el palé de porta lapicero vacío y aquí se añade que se ejecute de nuevo el programa TENC en la estación de ensamble, pero esta vez envía un código al SCADA extra que le indica a la estación de ensamble que se terminó el ensamble y otro que la estación de ensamble finalizó operaciones, por medio de la banda transportadora los materiales llegan al almacén y son guardados.

El plan de procesos debe guardarse para ser ejecutado desde uno de las opciones disponibles para programas de usuario en el SCADA de CIROS Production®. Este plan de procesos puede ser ejecutado incluso no se tenga conectada la computadora con el SCADA extra, puesto que las tareas de los programas originales no han sido alteradas, simplemente se añadieron instrucciones extra de comunicación de otro puerto. En el anexo 14.3 se muestra el código fuente del plan de procesos y el de las macro tareas modificadas “AsmDeskSet1”, “MovFromTrans1” y “MovToTrans1”.

9.3 SCADA con el software IGNITION de Inductive Automation®

Para la realización de la aplicación se utilizó la versión 7.8.0 (b2015101414) demo de IGNITION®, con la previa instalación de JAVA® 8.0.650.17 y de MySQL® 5.6.25, en una computadora laptop con sistema operativo de 64 bits Windows® 8.1, 8GB de memoria RAM y procesador Intel Core i5® @ 1.70 GHz.

9.3.1 TCP Driver

Para realizar la comunicación de la computadora con el controlador del robot, al no existir a la fecha un OPC para el tipo de controlador del robot de la estación, se recurrió al uso del driver TCP que

IGNITION® proporciona dentro del servidor OPC-UA que provee, este driver es estrictamente un oyente pasivo y está configurado para conectarse a uno o más puertos en una dirección IP dada, pudiendo enviar y recibir datos por este.

Los parámetros que deben configurarse para su funcionamiento se describen a continuación:

- General
 - Name: nombre con el que se quiere llamar al dispositivo a conectar
 - Description: aquí puede colocarse si se desea una descripción del dispositivo
 - Enable: activación del dispositivo
- Connectivity
 - Port(s): el o los puertos a los que se va a conectar
 - Address: dirección IP a conectarse
 - Inactivity Timeout: el número de milisegundos sin recibir datos de la fuente antes de que una desconexión/reconexión se realice. Colocar 0 para deshabilitar la opción.
- Message
 - Message Delimiter Type: establece el método utilizado para determinar cuánto o qué longitud de datos constituye un mensaje completo. Puede elegirse entre las siguientes opciones
 - PacketBased: se supone que todo lo que llega en un paquete, sin importar longitud o el contenido, es el mensaje.
 - CharacterBased: los datos se añaden a un búfer de mensaje hasta que el carácter establecido como delimitador llega, momento en el que el contenido del búfer es considerado el mensaje.
 - FixedSize: los datos se añaden a un búfer de mensaje hasta que recibe el número de caracteres establecido, momento en el que el contenido del búfer es considerado el mensaje.
 - Message Delimiter: este parámetro se deja vacío si el tipo de delimitador de mensaje se

configura como “PacketBased”, pero si es “CharacterBased” o “FixedSize” aquí se coloca el carácter que indica el final del mensaje o el número de caracteres en que consiste el mensaje.

- Field Count: número de campos en un mensaje (default: 0)
- Field Delimiter: el o los caracteres que serán usados como delimitador de campos

Si se requiere que además de leer del dispositivo también se le escriba se debe habilitar la opción de “Show advanced properties”, para poder configurar la escritura:

- Writing
 - Writeback Enabled: activa la capacidad de escritura para el dispositivo
 - Writeback Message Delimiter: aquí se coloca el delimitador esperado por el dispositivo para indicar el final de un mensaje entrante, se deja vacío sino se necesita delimitador.

Para agregar el dispositivo que utiliza el TCP driver en la página web Gateway de IGNITION® en la pestaña de “Configuración” se ingresa al apartado de “OPC-UA”, se selecciona “Devices”, luego “Create New Device”, se elige “TCP Driver” y se presiona el botón “Next” para entrar a la configuración del dispositivo. Al terminar se presiona el botón “Save Changes” y ya se muestra el dispositivo creado con su nombre, descripción, si está activado, y el estado de la conexión, la cual si ya está conectado con la computadora y la configuración es correcta deber aparecer como tal.

La configuración se realizó para leer y escribir datos desde y hacia el controlador del robot con la dirección IP 192.168.10.11 por el puerto 10006, como tipo de delimitador del mensaje se eligió la opción de “CharacterBased” y como carácter delimitador la letra “F”, la configuración de los parámetros se muestra en la Tabla 9.5.

| General | |
|--------------------|----------------------------------|
| <i>Name</i> | ControladorRC2 |
| <i>Description</i> | Controlador para el robot RV-3SB |
| <i>Enabled</i> | True |
| | |

| | |
|------------------------------------|----------------|
| Connectivity | |
| <i>Port(s)</i> | 10006 |
| <i>Address</i> | 192.168.10.11 |
| <i>Inactivity Timeout</i> | 0 |
| | |
| Message | |
| <i>Message Delimiter Type</i> | CharacterBased |
| <i>Message Delimiter</i> | F |
| <i>Field Count</i> | 0 |
| <i>Field Delimiter</i> | |
| | |
| Writing | |
| <i>Writeback Enabled</i> | True |
| <i>Writeback Message Delimiter</i> | |

Tabla 9.5. Parámetros de configuración del TCP Driver para el controlador del robot.

Como este driver no es un OPC exclusivo para el controlador, no se cuenta con tags que brinden valor de sensores o variables para poder ser leídos directamente desde el SCADA, sino que se generan solamente dos tags de tipo string una para enviar datos llamada “Message” y otra para escritura llamada “Writable”, por lo que valores de sensores o variables deben enviarse al SCADA en uno o varios mensajes de tipo string, por lo que es necesario crear tags de expresión en el SCADA para separar e identificar estos valores dentro del mensaje recibido.

9.3.2 Base de datos

Como se mencionó anteriormente para esta aplicación se utiliza una base de datos de MySQL®, para que IGNITION® pueda trabajar con ella esta debe de configurarse en el Gateway, en la pestaña de “Configuración” debe ingresarse al apartado de “Databases”, seleccionar “Connections”, luego “Create new Database Connection...”, elegir “MySQL ConnectorJ” que es el driver JDBC¹ oficial de MySQL y presionar el botón “Next” para entrar a la configuración del dispositivo. Al terminar se presiona el botón

¹ API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, utilizando el dialecto SQL del modelo de base de datos que se utilice.

“Save Changes” y ya se muestra la conexión creada de la base de datos con su nombre, descripción, driver, traductor y estado; el cual si no ha habido ningún inconveniente deberá indicar que la base de datos es válida, de lo contrario hay un problema con la base de datos o en la configuración.

En la configuración de la base de datos solamente se colocó nombre y descripción a la base de datos, además de colocarle el nombre de usuario (root) y el password con que se instaló MySQL® en la computadora, los demás parámetros se dejaron como IGNITION® los coloca por defecto, en la Tabla 9.6 se muestra como quedó la configuración

| | |
|-------------------------------------|--|
| Main Properties | |
| <i>Name</i> | Basededatoscr2 |
| <i>Description</i> | Base de datos del controlador cr2 del robot rv-3sb |
| <i>JDBC Driver</i> | MySQL Connector J |
| <i>Connect URL</i> | jdbc:mysql://localhost:3306/test |
| <i>Username</i> | Root |
| <i>Password</i> | (el mismo password que se colocó cuando se instaló MySQL en la computadora) |
| <i>Extra Connection Properties</i> | zeroDateTimeBehavior=convertToNull;connectTimeout=120000;socketTimeout=120000; |
| <i>Enabled</i> | True |
| <i>Validation Timeout (ms)</i> | 10000 |
| <i>Failover Datasource</i> | None |
| <i>Failover Mode</i> | STANDARD |
| <i>Slow Query Log Threshold</i> | 60000 |
| | |
| SQL Compatibility | |
| <i>Translator</i> | MYSQL |
| <i>Include Schema in Table Name</i> | False |
| | |
| Connection Pooling | |
| <i>Initial Size</i> | 0 |

| | |
|--|----------|
| <i>Max Active</i> | 8 |
| <i>Max Idle</i> | 8 |
| <i>Min Idle</i> | 0 |
| <i>Max wait</i> | 5000 |
| | |
| Connection Testing | |
| <i>Validation Query</i> | SELECT 1 |
| <i>Test on Borrow?</i> | TRUE |
| <i>Test on Return?</i> | FALSE |
| <i>Test While Idle?</i> | FALSE |
| <i>Eviction Rate</i> | -1 |
| <i>Eviction Tests</i> | 3 |
| <i>Eviction Time</i> | 1800000 |
| <i>Connection Initialization Commands</i> | |
| <i>Default Transaction Isolation Level</i> | Default |

Tabla 9.6. Parámetros de configuración de la base de datos.

9.3.3 Descripción de la HMI de la modalidad 1: “Supervisión de la estación de ensamble en conjunto con SCADA de FESTO”

La HMI realizada para la aplicación en esta modalidad consiste de 3 ventanas principales y 3 ventanas emergentes, estas poseen un color de fondo neutro, específicamente un gris de código hexadecimal #EEEECE8 y texto de color negro, utilizando un único tipo de fuente “sans serif” llamada “Dialog”, con respecto al tamaño dependiendo de la importancia del texto varía de 12 a 20, además de resaltarse en negrita aquel de mayor importancia como los títulos, el estado de la estación o las alarmas.

Ventana Principal

En la Figura 9.2 se muestra la pantalla principal, en esta se muestran los siguientes elementos:

- Estado de la estación (1): la estación se puede encontrar en cuatro estados:

- **ON:** indicado con un recuadro con fondo verde y letras negras, la estación se encuentra en este estado cuando se están ejecutando los programas propios de la estación.
 - **OFF:** indicado con un recuadro con fondo gris y letras negras, la estación se encuentra en este estado ya sea porque ya finalizó o porque aún no es su turno, en ambos caso está a la espera de una nueva orden del SCADA de CIROS Production®.
 - **FALLA:** indicado con un recuadro con fondo amarillo y letras negras, la estación se encuentra en este estado cuando se ha detectado una anomalía en la misma, debido a cualquiera de los factores que se muestran con fondo gris o blanco en la Tabla 9.3. Cuando se está en este estado también se activa un sonido corto para alertar al operario sino está viendo la pantalla, más un botón de “Ver Detalle” que abre la ventana emergente como se muestra en la Figura 9.3, que indica textualmente cual ha sido la falla.
 - **INDETERMINADO:** indicado en un recuadro con fondo blanco y letras verdes, la estación se encuentra en este estado cuando no existe comunicación con el SCADA de CIROS Production® ya sea porque este no esté corriendo o porque aún no ha enviado la orden de ensamblado.
- **Posición del robot (2):** cuando el robot real se mueve a alguna posición, en pantalla se actualiza la imagen del robot indicando la posición, en la Figura 9.4 se muestran algunas de las posiciones que adopta el robot para ensamblar el set de escritorio.
 - **Estado de los sensores (3):** en la estación se monitorean 10 sensores: 4 de detección de palés, 1 para detección de termómetros, 1 para detección de higrómetros, 1 para detección de lapiceros, 1 para detección de placa base en la posición de ensamble y 2 de indicación de activación de cilindros de sujeción de placa base en la posición de ensamble.
Para una mejor visualización el estado de los sensores se muestra tanto sobre los elementos en el diagrama de la estación como en forma de listado con los nombres.



Figura 9.2. Ventana Principal del SCADA extra en la modalidad 1, operación normal.



Figura 9.3. Ventana Principal del SCADA extra en la modalidad 1, en falla.

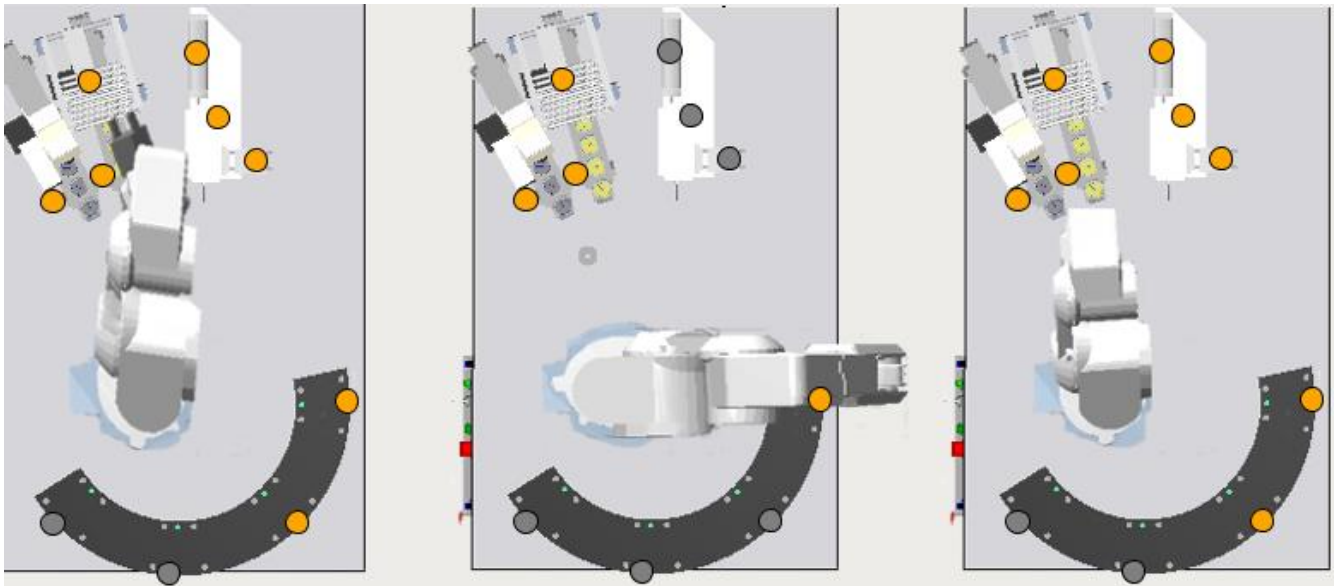


Figura 9.4. Algunas de las posiciones en las que pueda encontrarse el robot.

- Programa en ejecución (4): aquí se indica cual es el programa que se está ejecutando en la estación de ensamble que pueden ser cinco: “mover palé”, “mover placa base”, “mover medidores”, “mover porta lapicero” y “colocar el lapicero”, pero si el SCADA de CIROS Production® no se ha ejecutado o aún no ha mandado a ejecutar el plan de procesos se indica que no se ejecuta ningún programa, y se está a la espera de órdenes del SCADA principal pues el plan de procesos ya inició pero la estación de ensamble no ha comenzado operaciones se indica que aún ningún programa se ejecuta en la estación.
- Códigos de estado y posición (5): en estos recuadros se muestran los códigos de posición y de estado/alarmas de la estación de ensamble, puede que esta información no parezca útil para el operador puesto que la posición la ve en la imagen del robot y el estado/alarma se muestra en el programa de ejecución, en el estado de la estación y como notificación en caso de las alarmas, pero ver esto nos permite saber qué es exactamente lo que está recibiendo la computadora desde la estación, lo cual es útil en caso de que se esté presentando una falla de comunicación.
- Botones para acceder a otras pantallas (6): en la ventana principal se encuentran botones para acceder a las pantallas de estadísticas y de históricos que se explicarán más adelante, más un botón que sirve para cerrar el programa.

Ventana Estadísticas

En esta ventana que se muestra en la Figura 9.5, se encuentran los siguientes elementos:

- Indicador de número de sets ensamblados (1): se muestra un display con el número de sets ensamblados que aumenta su cuenta cuando la estación de ensamble finaliza, junto al display se muestra una imagen estática del tipo de set que se ensambla.

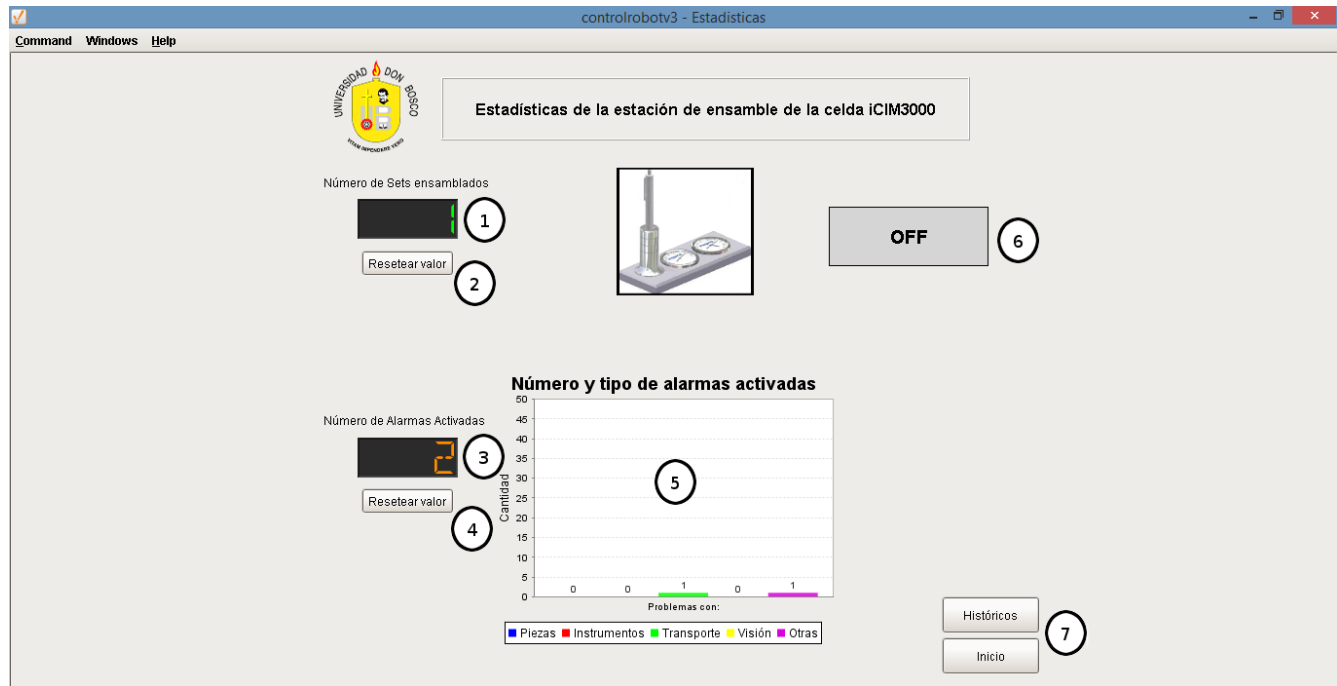


Figura 9.5. Ventana Estadísticas del SCADA extra en la modalidad 1.

- Botón de resetear número de sets ensamblados (2): con este botón se puede resetear la cuenta de sets ensamblados, al presionarlo aparece una ventana emergente para realizarlo, en esta por seguridad se debe ingresar primero una contraseña, tal como se muestra en la Figura 9.6.
- Indicador de número de alarmas activadas (3): se muestra un contador con el número total de alarmas que se han activado.
- Botón de resetear número de alarmas activadas (4): con este botón se puede resetear la cuenta de alarmas activadas, al presionarlo aparece una ventana emergente para realizarlo, en esta por seguridad se debe ingresar primero una contraseña, tal como se muestra en la Figura 9.6.

- Gráfica con número y tipo de alarmas activadas (5): en este gráfico se muestran ordenadas por categoría las alarmas que se han activado.
- Estado de la estación (6): aquí también se muestra el estado de la estación de la misma manera que en la ventana principal, debido a que el operador debe saber siempre cómo está trabajando la estación independientemente de la ventana en la que se encuentre.
- Botones para acceder a otras pantallas (7): desde esta ventana se puede volver a la ventana principal o ingresar a la ventana de históricos que se explicará más adelante.

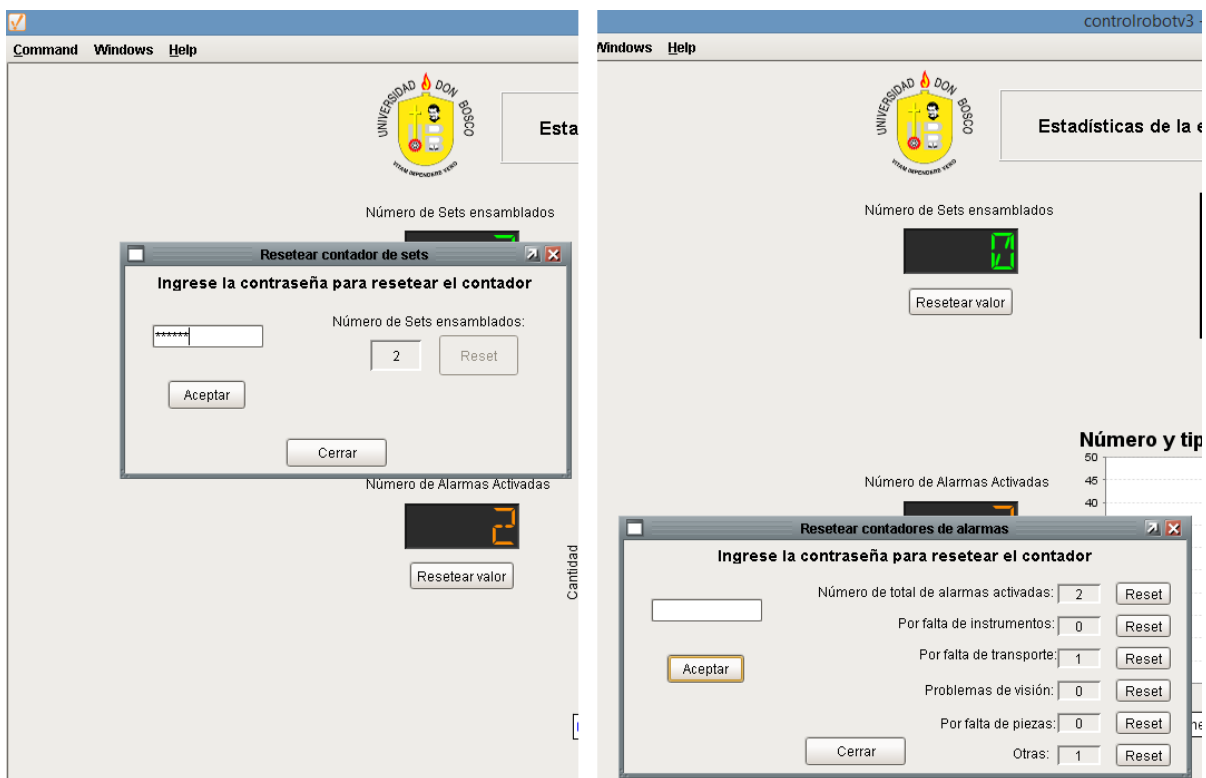


Figura 9.6. Botones para resetear la cuenta de sets ensamblados y alarmas activadas del SCADA extra en la modalidad 1.

Ventana de Históricos

En esta ventana que se muestra en la Figura 9.7, se encuentran los siguientes elementos:

- Historial de sets ensamblados (1): se muestra en un gráfico la tendencia del número de sets

ensamblados en los últimos 6 días, los datos se muestran en rangos de 6 horas.

- Historial de alarmas activadas (2): se muestra en un gráfico la tendencia del número y tipo de alarmas activadas en los últimos 6 días, los datos se muestran en rangos de 6 horas.
- Estado de la estación (3): aquí también se muestra el estado de la estación de la misma manera que en la ventana principal, debido a que el operador debe saber siempre cómo está trabajando la estación independientemente de la ventana en la que se encuentre.
- Botones para acceder a otras pantallas (4): desde esta ventana se puede volver a la principal o ingresar a la ventana de estadísticas.

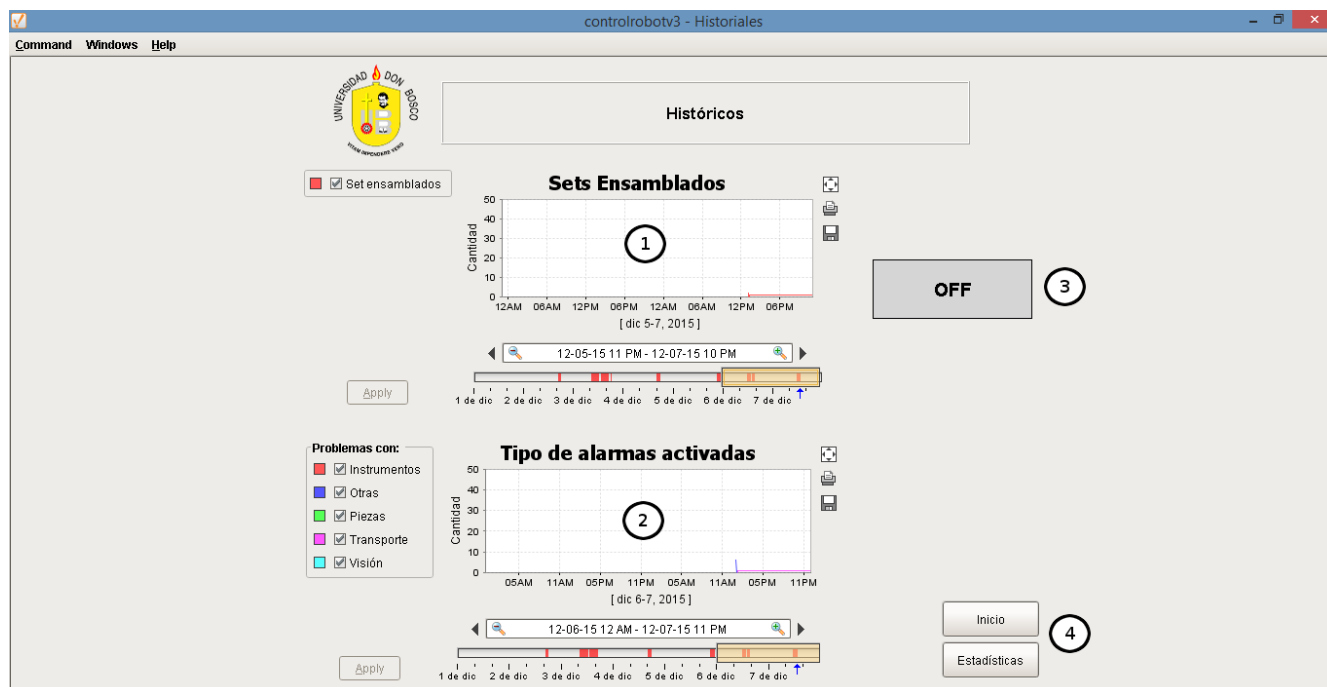


Figura 9.7. Ventana de Históricos del SCADA extra en la modalidad 1.

En el anexo 14.4 se muestran los elementos utilizados junto con las tags y scripts necesarios para llevar a cabo las operaciones descritas en esta modalidad.

9.3.4 Descripción de la HMI de la Modalidad 2: “Sistema SCADA Para Estación de Ensamble Aislada”

La HMI realizada para la aplicación en esta modalidad consiste de 3 ventanas principales y 4 emergentes, las cuales poseen un color de fondo gris de código hexadecimal #EEEECE8 y texto de color negro, se utiliza como único tipo de fuente “Dialog” con un tamaño variable de 12 a 18 dependiendo de la importancia del texto, además de resaltar en negrita aquel con mayor importancia como títulos, estado de la estación o las alarmas.

Ventana Principal

En la Figura 9.8 se muestra la pantalla principal, en esta se muestran los siguientes elementos:

- Menú desplegable para seleccionar el tipo de set a ensamblar (1): puede elegirse una de las siguientes opciones:
 - Set estándar con lapicero: este es el que se muestra en la primera imagen de la Figura 6.2 y que consta de un porta lapicero con lapicero seguido de un higrómetro y un termómetro.
 - Set estándar sin lapicero: este es igual que el anterior solamente que no se coloca el lapicero.
 - Set modificado sin lapicero: en este se coloca el porta lapicero sin lapicero² en medio del higrómetro y del termómetro.

Para saber cómo es el set elegido este aparecerá en la imagen junto al menú.

- Imagen del set que se va a ensamblar (2): aquí se muestra una imagen dinámica que según el tipo de set que se elija cambia para mostrar al usuario como es el set escogido.
- Menú desplegable para seleccionar posición donde se va a tomar la placa base (3): aquí se elige en cuál de las 4 posiciones del receptor de palés se va a tomar la placa base del set, en esta posición también se almacenará el set cuando ya esté ensamblado. En la imagen de junto se aprecia cual ha sido la posición elegida.

² No es posible colocar lapicero cuando el porta lapicero está en esta posición debido a que el brazo robótico colisiona con el palé ubicado en la primera posición del receptor de palés.

- Menú desplegable para seleccionar posición donde se va a tomar el porta lapicero (4): aquí se elige en cuál de las 4 posiciones del receptor de palés se va a tomar el porta lapicero. En la imagen de junto se aprecia cual ha sido la posición elegida.
- Imagen que muestra las posiciones elegidas en el receptor de palés para tomar las piezas (5): aquí aparece en un recuadro con fondo naranja y con las letras **PB** sobre la posición elegida para tomar la placa base, esta también será la posición donde se colocará el set ya ensamblado, y en fondo rosado otro recuadro con las letras **PL** sobre la posición donde se tomará el porta lapicero.



Figura 9.8. Ventana principal del SCADA en modalidad 2.

- Botón Ensamblar y Botón Reintentar (6): ambos ocupan la misma posición en la ventana, con el botón de “Ensamblar” se manda a la estación la orden de ensamblar el tipo de set elegido, tomando las piezas según se indicó en los menús desplegables, sin embargo si el usuario elige la misma posición para tomar la placa base como el porta lapicero, y da clic en el botón, aparece la ventana emergente mostrada en la Figura 9.9 y no permite el envío de la orden hasta que se elijan posiciones distintas para tomar las piezas; si todo es correcto, se manda la orden y el botón se inhabilita hasta que se termina el ensamble u ocurra alguna falla, en este último caso desaparece

y aparece el botón de “Reintentar”, si este se presiona envía la orden al robot de que se realice de nuevo la operación en la que estaba el robot cuando ocurrió la falla. Al finalizar un ensamble este desaparece y vuelve a aparecer el botón de “Ensamblar”.



Figura 9.9. Ventana emergente para indicar que se ha elegido la misma posición para la placa base y el porta lapicero en el SCADA en modalidad 2.

- Botón finalizar (7): cuando se presiona este botón se envía al robot la orden de finalizar la ejecución del programa y en pantalla desaparece tanto él como el botón de Ensamblar/Reintentar y aparece otro para cerrar la interfaz, puesto que en el robot ya no está corriendo el programa. El botón finalizar solo puede ser presionado cuando no se ha iniciado un ensamble, ya se finalizó o cuando ha ocurrido una falla, pues mientras el robot está en operación normal este botón se inhabilita.
- Estado de la estación (8): la estación se puede encontrar en tres estados:
 - ON: indicado con un recuadro con fondo verde y letras negras, la estación se encuentra en este estado cuando se está ejecutando el ensamble con normalidad.
 - OFF: indicado con un recuadro con fondo gris y letras negras, la estación se encuentra en este estado ya sea porque ya finalizó el ensamble o porque aún no se ha iniciado un ensamble.
 - FALLA: indicado con un recuadro con fondo amarillo y letras negras, la estación se

encuentra en este estado cuando se ha detectado una anomalía en la misma, debido a cualquiera de los factores que se muestran con fondo celeste o blanco en la Tabla 9.3. Cuando se está en este estado también se activa un sonido corto para alertar al operario sino está viendo la pantalla, más un botón de “Ver Detalle” que abre la ventana emergente como se muestra en la Figura 9.10, que indica textualmente cual ha sido la falla.

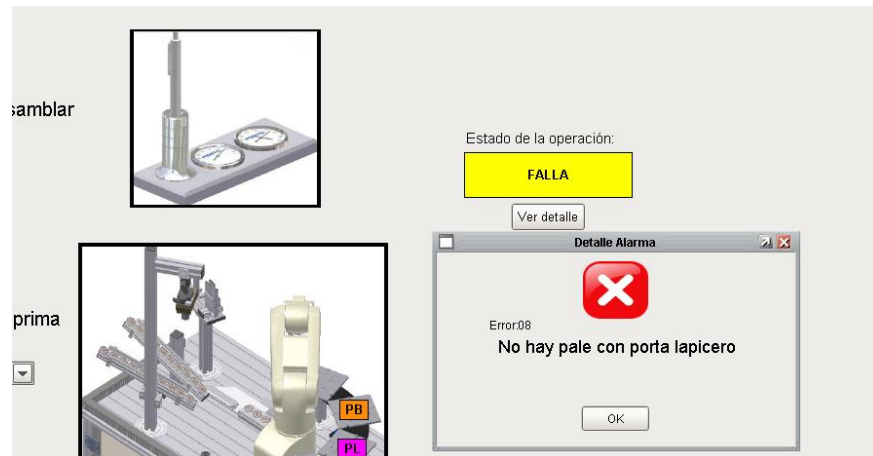


Figura 9.10. Ventana emergente con indicación del tipo de falla en SCADA en modalidad 2.

- Botones para acceder a otras pantallas (9): en la ventana principal se encuentran botones para acceder a las pantallas de estadísticas y de históricos que se explicarán más adelante.
- Valores de datos enviados y recibidos (10): en estos recuadros se muestran los valores de los datos que se están recibiendo y enviando desde y hacia el robot, los datos enviados son los códigos del tipo de set y las posiciones de las piezas y los recibidos los códigos de estado/alarmas de la estación de ensamble, puede que esta información no parezca útil para el operador puesto que el tipo de set y posiciones los ve en los menús y las imágenes y el estado/alarma en el recuadro correspondiente y como notificación en caso de las alarmas, pero ver esto permite saber qué es exactamente la información que está enviando y recibiendo, lo cual es útil en caso de que se esté presentando una falla de comunicación.

Ventana Estadísticas

En esta ventana que se muestra en la Figura 9.11, se encuentran los siguientes elementos:

- Indicador de número total de sets ensamblados (1): se muestra un display con el número de sets

ensamblados que aumenta su cuenta cuando se termina de ensamblar cualquier tipo de set.

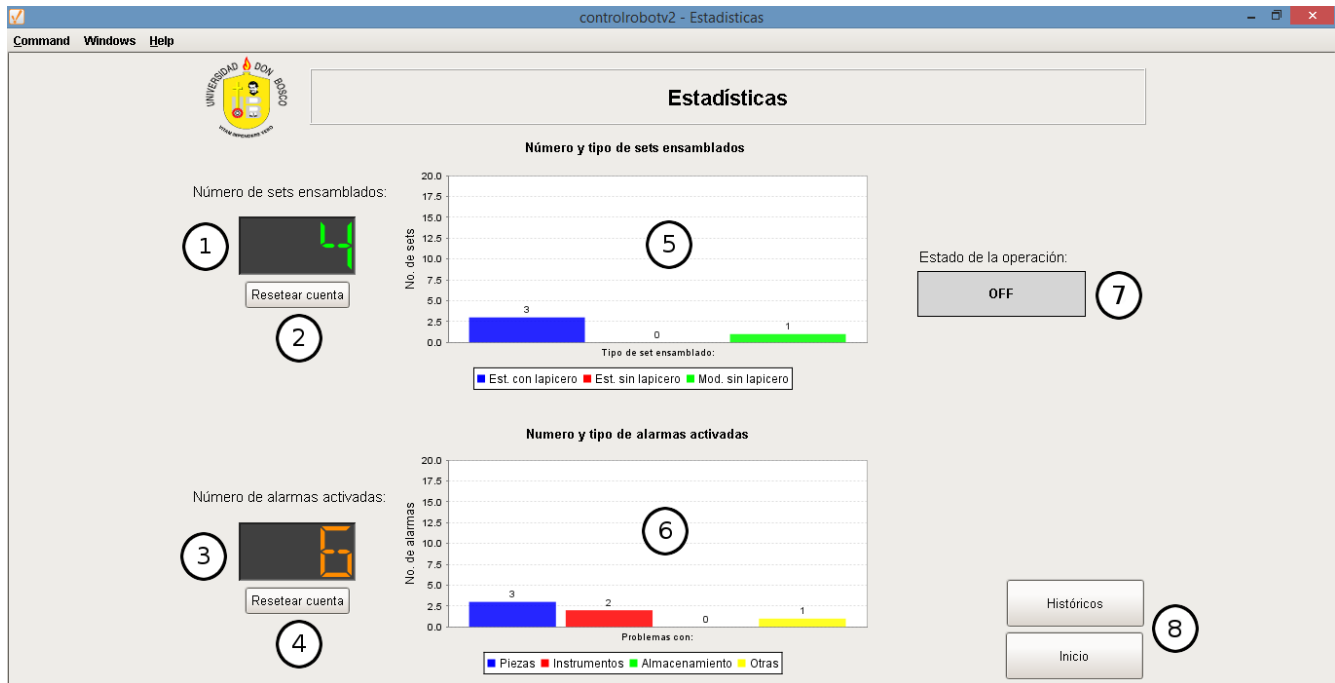


Figura 9.11. Ventana Estadísticas del SCADA en la modalidad 2.

- Botón de resetear número de sets ensamblados (2): con este botón se puede resetear la cuenta de sets ensamblados, ya sea de un tipo en particular o la cuenta total, al presionar el botón aparece una ventana emergente que por seguridad pide ingresar primero una contraseña para permitir el reseteo, tal como se muestra en la Figura 9.12.
- Indicador de número de alarmas activadas (3): se muestra un display con el número total de alarmas que se han activado.
- Botón de resetear número de alarmas activadas (4): con este botón se puede resetear la cuenta de alarmas activadas, ya sea de un tipo en particular o la cuenta total, al presionar el botón aparece una ventana emergente que por seguridad pide ingresar primero una contraseña para permitir el reseteo, tal como se muestra en la Figura 9.12.
- Gráfica con número y tipo de set ensamblado (5): en este gráfico se muestran ordenados por categoría los set que se han ensamblado.

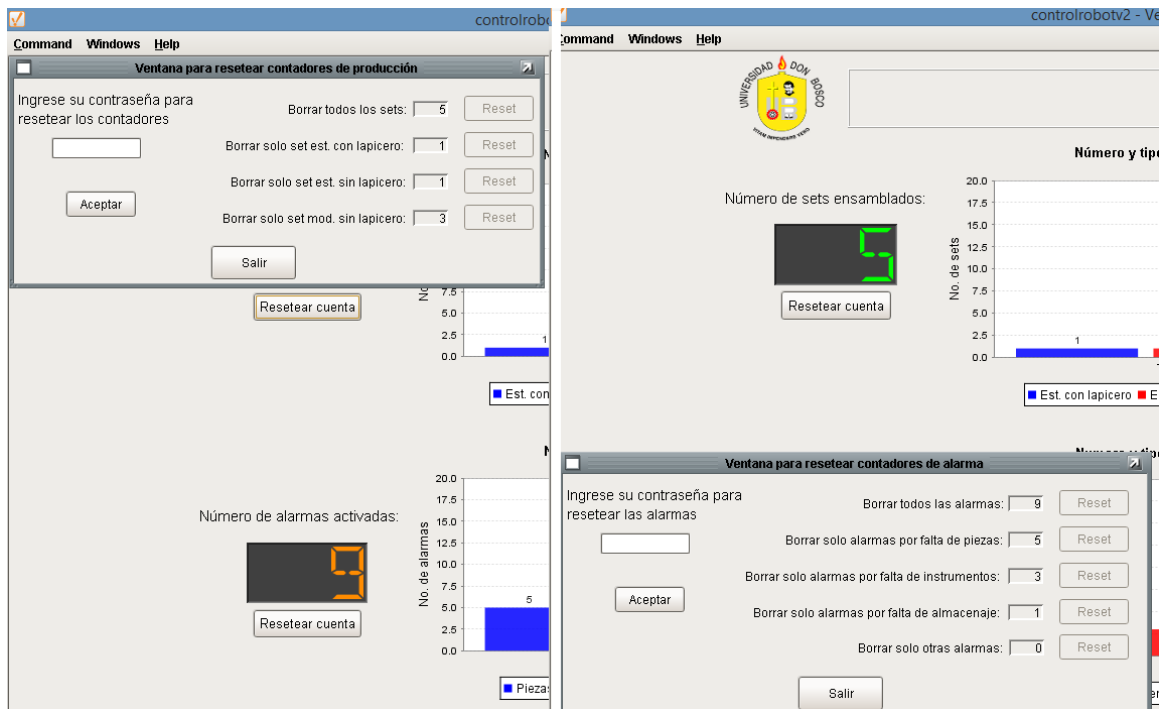


Figura 9.12. Botones para resetear la cuenta de sets ensamblados y alarmas activadas del SCADA extra en la modalidad 2.

- Gráfica con número y tipo de alarma activada (6): en este gráfico se muestran ordenadas por categorías las alarmas que se han activado.
- Estado de la estación (7): aquí también se muestra el estado de la estación de la misma manera que en la ventana principal, debido a que el operador debe saber siempre cómo está trabajando la estación independientemente de la ventana en la que se encuentre.
- Botones para acceder a otras pantallas (8): desde esta ventana se puede volver a la ventana principal o ingresar a la ventana de históricos que se explicará más adelante.

Ventana de Históricos

En esta ventana que se muestra en la Figura 9.13, se encuentran los siguientes elementos:

- Historial de sets ensamblados (1): se muestra en un gráfico la tendencia del número y tipos de sets ensamblados en los últimos 6 días, los datos se muestran en rangos de 4 horas.

- Historial de alarmas activadas (2): se muestra en un gráfico la tendencia del número y tipo de alarmas activadas en los últimos 6 días, los datos se muestran en rangos de 12 horas.
- Estado de la estación (3): aquí también se muestra el estado de la estación de la misma manera que en la ventana principal, debido a que el operador debe saber siempre cómo está trabajando la estación independientemente de la ventana en la que se encuentre.
- Botones para acceder a otras pantallas (4): desde esta ventana se puede volver a la principal o ingresar a la ventana de estadísticas.

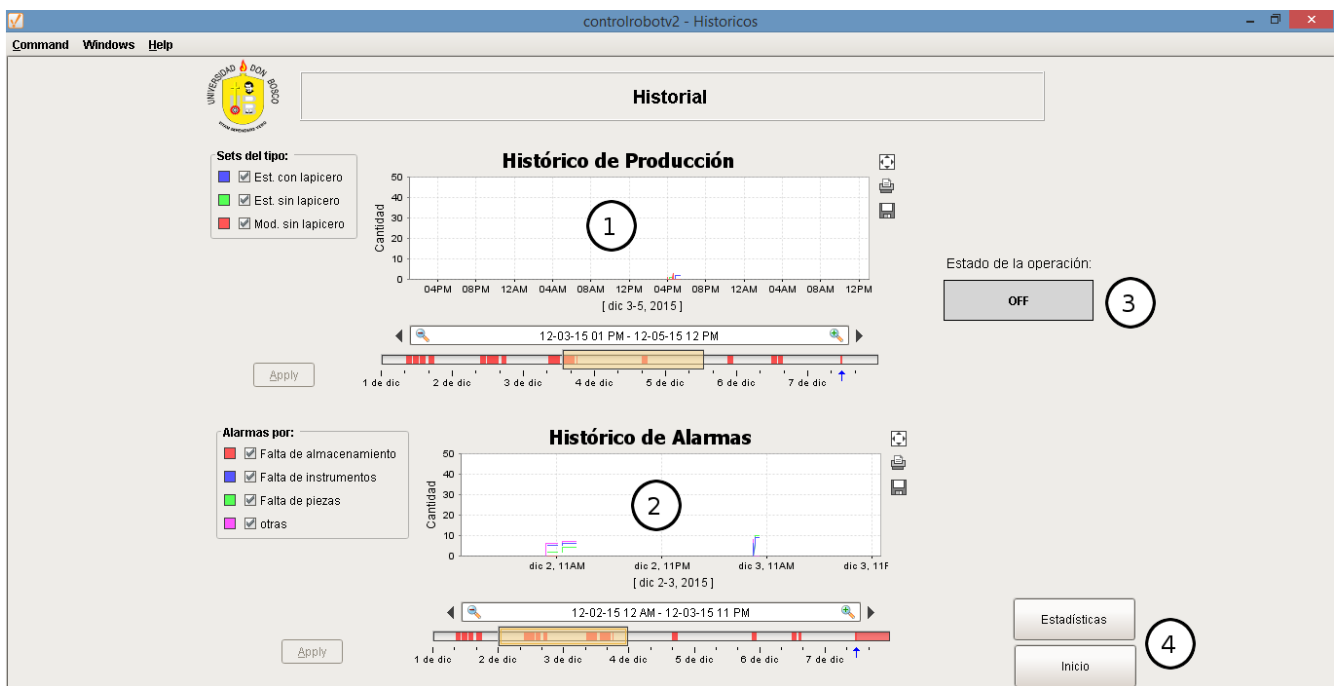


Figura 9.13. Ventana de Históricos del SCADA en la modalidad 2.

En el anexo 14.5 se muestran los elementos utilizados junto con las tags y scripts necesarios para llevar a cabo las operaciones descritas en esta modalidad.

10. Instrucciones de uso de las aplicaciones

10.1 Instrucciones uso de SCADA en modalidad 1

1. Encienda y coloque en automático las estaciones que componen la celda iCIM3000.
2. Encienda la computadora con el SCADA de CIROS Production® y ejecútelo dando doble clic sobre el archivo iCIM.LPJ.
3. Verifique que en la base de datos del almacén y por ende también en el almacén físico se encuentre materia prima para la elaboración de set de escritorio estándar (Placa base de aluminio= artículo 42143 y porta lapicero de aluminio= artículo 42104)
4. Encienda la computadora con el SCADA de IGNITION® y conéctela a través de un cable plano Ethernet al switch de la celda iCIM3000 (si está trabajando con la versión demo ingrese al Gateway en el navegador web y resetee el tiempo de prueba para tener disponible dos horas de trabajo)
5. Ejecute el archivo del SCADA de la modalidad 1 dando clic sobre su acceso directo en el escritorio SCADAm0d1.
6. En iCIM.LPJ entre a modo de producción y de clic en la opción “User 1” para que se ejecute el plan de procesos que se comunica también con el SCADA extra y ya podrá supervisar desde este el comportamiento de la estación de ensamble.

10.2 Instrucciones uso de SCADA en modalidad 2

1. Encienda la estación de ensamble y coloque el selector del controlador en la opción Auto (op).
2. Encienda la computadora con el SCADA de IGNITION® y conéctela a través de un cable plano Ethernet al switch de la celda iCIM3000 (si está trabajando con la versión demo ingrese al Gateway en el navegador web y resetee el tiempo de prueba para tener disponible dos horas de trabajo)
3. Ejecute el archivo del SCADA de la modalidad 2 dando clic sobre su acceso directo en el escritorio SCADAm0d2.
4. Busque en el controlador del robot el archivo ABC4 y presione el botón “START” para ejecutarlo
5. Con esto ya puede mandar a ensamblar el tipo de sets que desee y monitorear el funcionamiento

de la estación de ensamble mientras este se realiza. Recuerde que los palés con placa base y porta lapicero deberá colocarlos manualmente en cualquiera de las posiciones del receptor de palés para que el robot los tome y haga el ensamble y luego retirarlos cuando haya finalizado.

11. Resultados

Se probaron ambas modalidades del sistema SCADA obteniendo resultados exitosos en las pruebas realizadas.

En el caso de la modalidad 1 donde se trabaja en conjunto con el SCADA de CIROS Production®, no hubo problemas de choques al estar ambas aplicaciones SCADA solicitando información por la red Ethernet a la estación de ensamble, se mostraba correctamente el programa que se estaba realizando en la estación, el estado de los sensores y el movimiento del robot se veían casi en tiempo real y cuando ocurría una alarma se notificaban en ambos sistemas SCADA.

En el caso de la modalidad 2 donde la estación trabaja de manera aislada, la estación ensambló correctamente los tres tipos de set posibles, si había alguna anomalía se notificaba en pantalla, si se solventaba el problema y se reintentaba la tarea, se realizaba de nuevo sin problemas.

Para ambas modalidades se llevó correctamente el registro de elementos ensamblados y de alarmas activadas, tanto en las estadísticas actuales como en el historial de los últimos 6 días que se había estipulado.

12. Conclusiones

Se logró una comunicación de manera bidireccional entre la interfaz y un tipo de controlador diferente al tradicional PLC, utilizando uno de los drivers que proporciona IGNITION por defecto, evitando tener que adquirir un OPC del fabricante del controlador, el cual es inexistente o uno general lo cual además de requerir más instalaciones y configuraciones involucra un costo monetario.

Con la aplicación desarrollada con el software SCADA IGNITION® fue posible supervisar correctamente el estado de los sensores y la posición del robot pertenecientes a la estación de ensamble, además de poder enviar órdenes para que el robot las ejecutara y de almacenar valores en la base de datos, demostrando así la eficacia del software IGNITION®.

En cuanto ergonomía y seguridad en la HMI, se cuidaron los colores, se evitó la saturación de información o gráficos, los indicadores a color se apoyaron con texto y las alarmas iban acompañadas de un sonido corto que no fuera muy molesto, el tamaño de letra se colocó de tal manera que a una distancia prudencial se pudiera leer con claridad, con respecto a la facilidad de uso y tolerancia a fallos, se inhabilitaron u ocultaron botones para que no fueran presionados cuando no se debía, y aunque no es crítico para evitar manipulación de las estadísticas de producción y alarmas se protegió con contraseña el reseteo de estas.

Aunque con la limitante de no haber podido obtener información de la cámara debido a que no pudo establecerse comunicación entre el software de análisis de esta e IGNITION®, se logró el objetivo perseguido con este proyecto, teniendo como resultado dos herramientas sencillas y de fácil uso que permitirán a los alumnos interactuar con la estación de ensamble por medio de dos aplicaciones realizadas con software que podrán encontrar y aplicar en la industria en el control y supervisión de procesos.

13. Bibliografía

- [1] Boyer, S. (2010). *SCADA: Supervisory Control and Data Acquisition*, Carolina del Norte, Estados Unidos de América: International Society of Automation (ISA).
- [2] Rodríguez, A. (2007). *Sistemas SCADA*, Barcelona, España: Marcombo, S.A.
- [3] OPC Foundation. (s.f). *What is OPC?*. Recuperado el 19 de septiembre de 2015 de <https://opcfoundation.org/about/what-is-opc/>
- [4] OPC Foundation. (s.f). *OPC Technologies: Classic*. Recuperado el 19 de septiembre de 2015 de <https://opcfoundation.org/about/opc-technologies/opc-classic/>
- [5] OPC Foundation. (s.f). *OPC Technologies: Unified Architecture*. Recuperado el 19 de septiembre de 2015 de <https://opcfoundation.org/about/opc-technologies/opc-ua/>
- [6] Cogent Real-Time Systems Inc. (s.f). *What is OPC?*. Recuperado el 19 de septiembre de 2015 de <http://www.opcdatahub.com/WhatIsOPC.html>
- [7] ISO. (2010). ISO 9241-210:2010(en) *Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*.
- [8] ISO. (1996). ISO 11429:1996 *Ergonomics -- System of auditory and visual danger and information signals*.
- [9] Instituto Nacional de Seguridad e Higiene en el Trabajo. Ministerio de Trabajo y Asuntos Sociales, (2005). *Manual de Normas Técnicas para el diseño Ergonómico de Puestos con pantallas de visualización (2ª Edición)*. España. Recuperado el 04 de octubre de 2015 de http://www.insht.es/InshtWeb/Contenidos/Documentacion/TextosOnline/Guias_Ev_Riesgos/normastecnicaspyd.pdf

- [10] Ruzarovsky, R., Holubek, R. y Delgado D. (2013). Integration Methods and Processes of Product Design and Flexible Production for Direct Production within the iCIM 3000 System. *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, 7(8), 653-658. Recuperado el 21 de octubre de 2015 de <http://waset.org/publications/16236/integration-methods-and-processes-of-product-design-and-flexible-production-for-direct-production-within-the-icim-3000-system>
- [11] Iglesias, A. (2010). *Sistema Automatizado de Ensamble con Visión Artificial para Celdas de Manufactura Flexible* (tesis de maestría). Universidad Nacional Autónoma de México, México.
- [12] Quintana, D. (2011). *Control Supervisorio de un Robot Manipulador Industrial Mitsubishi Melfa Utilizando la Plataforma Labview* (tesis de grado). Pontificia Universidad Javeriana, Bogotá DC, Colombia
- [13] Mitsubishi. (2000). *Mitsubishi Industrial Robot CRn-500 series Instruction Manual Ethernet Interface*. Melfa BFP-A8108-B.
- [14] Mitsubishi. (2007). *Mitsubishi Industrial Robot CR1/CR2/CR3/CR4/CR7/CR8/CR9 Controller Instruction Manual Detailed explanations of functions and operations*. Melfa BFP-A5992-M.
- [15] FESTO Didactic. (2010). *Manual CIROS Production Instrucciones de utilización*.
- [16] FESTO Didactic. (2010). *Manual iCIM*.
- [17] Ignition by Inductive Automation. (S.F). *Ignition User Manual 7.8*.
- [18] Mitsubishi. (2007). *RV-3S/3SJ/3SB/3SJB Series Instruction Manual Robot Arm Setup & Maintenance*. Melfa BFP-A8388-B.
- [19] Mitsubishi. (2007). *RV-3SB/3SJB Series Standard Specifications Manual (CR1B-571/CR2B-574*

Controller). Melfa BFP-A8408-C.

[20] Inductive Automation. (2015). *Courses*. Disponible en <https://inductiveuniversity.com/>

[21] Sanz, J. (2007). *Las normas Técnicas ISO 9241 y en 29241 sobre pantallas de visualización*. Instituto Nacional de Seguridad e Higiene en el Trabajo.

14. Anexos

14.1 Ajuste de parámetros en el controlador del robot

Para ajustar parámetros en el controlador de robot asegúrese de que su selector se encuentre en modo TEACH y que en el Teaching Pendant el selector este puesto en ENABLE

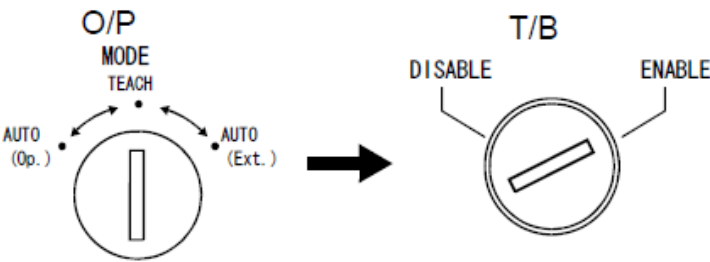
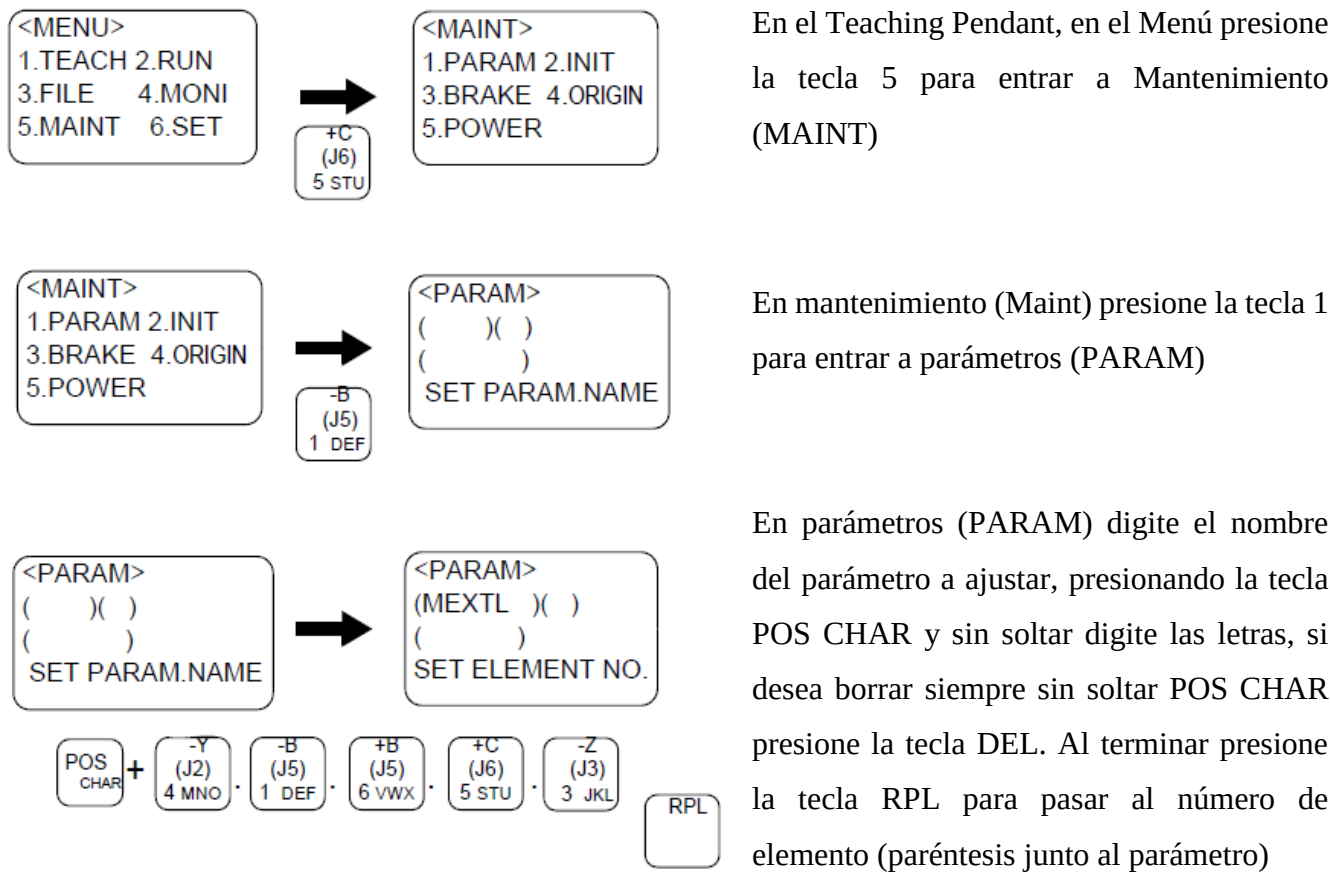
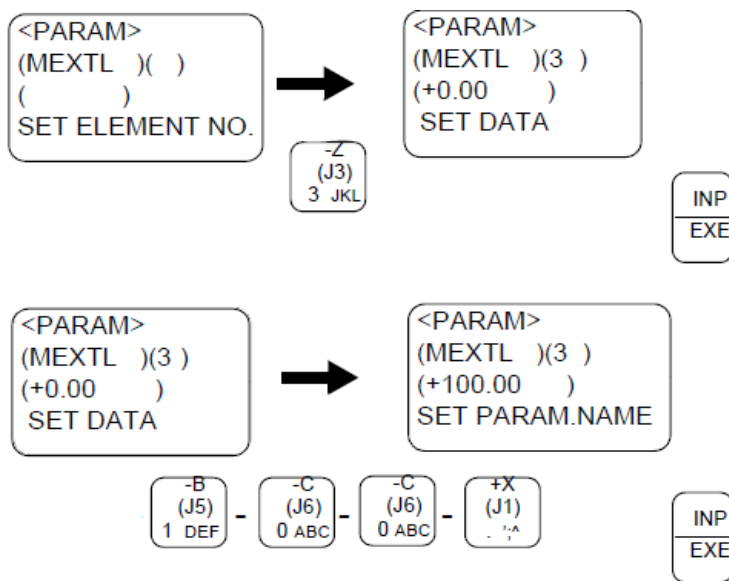


Figura 14.1. Selector del controlador en modo TEACH y el selector en el teaching pendant en ENABLE.





Se digita el número del elemento del parámetro (si es que tiene más de un elemento, sino se deja en blanco) y se presiona la tecla INP EXE para pasar a ajustar el valor

Luego se ingresa el valor del parámetro ya sean números o letras y para que ya quede ajustado se presiona la tecla INP EXE.

Es necesario apagar y volver a encender el controlador para que el cambio del parámetro quede validado.

14.2 Código fuente de los programas y las posiciones almacenados en el controlador del robot.

14.2.1 SCADA modalidad 1

- **TMP.MB4**

```

1000 |-----
1010 | Programa modificado del original MP de Christian Heinisch Festo Didactic
1020 | TMovePallet(Source, Target)VFF
1030 |
1040 | Parameter: MSource/MTarget = 1 --> Retrieve from / Store to : Pallet position 1
1050 | Parameter: MSource/MTarget = 2 --> Retrieve from / Store to : Pallet position 2
1060 | Parameter: MSource/MTarget = 3 --> Retrieve from / Store to : Pallet position 3
1070 | Parameter: MSource/MTarget = 4 --> Retrieve from / Store to : Pallet position 4
1080 | Parameter: MSource/MTarget = 15 --> Retrieve from / Store to : Conveyor position
1090 |-----
1100 | Autor: Tania Martínez 3 de dic de 2015
1110 |
1120 | folgende Positionen müssen geteacht werden / these positions must be taught
1130 |-----

```


[illegible]

[illegible]

```

2820 IF MSOURCE%=15 THEN
2830 RPOS$ = "6" 'envía que es la posición 6 (conveyor)
2840 ELSE
2850 RPOS$ = HEX$(MSOURCE%)
2860 ENDIF
2870 GOSUB *ENVIAR
2880 '*****
2890 CNT 0
2900 MVS PPAL(MSOURCE%) 'move straight to source position
2910 DLY 0.5 'delay 0.5 s
2920 HOPEN 1 'open hand
2930 WAIT GROOPEN = 1 'wait till hand is opened
2940 MVS PPAL(MSOURCE%),-PICK1% 'move straight to a position Pick1 mm above the Source pos.
2950 '*****
2960 GOSUB *ENVIAR 'envía cambio en estado de sensores al tomar el pale
2970 '*****
2980 '-----
2990 ' Place workpiece at target position
3000 '-----
3010 *PLACE
3020 PLACE1% = 200 'write 230 in Place1
3030 IF MTARGET% = 15 THEN
3040 PLACE1% = 230 'if Target position is at the conveyor write 30 in Place1
3050 '*****
3060 RPOS$ = "6" 'envía que es la posición 6 (conveyor)
3070 ELSE
3080 RPOS$ = HEX$(MTARGET%)
3090 '*****
3100 ENDIF
3110 MOV PPAL(MTARGET%),-PLACE1% 'move to a position Place1 mm above the Target position
3120 '*****
3130 GOSUB *ENVIAR
3140 '*****
3150 MVS PPAL(MTARGET%) 'move straight to target position
3160 '*****
3170 GOSUB *ENVIAR
3180 '*****
3190 DLY 0.5 'delay 0.5 s
3200 HCLOSE 1 'close hand
3210 WAIT GRCLOSE = 1 'wait till hand is closed
3220 MVS PPAL(MTARGET%),-PLACE1% 'move straight to a position Place1 mm above the Target pos.
3230 SELECT MTARGET% 'check if a pallette is at target position if there is no pallette,
3240 CASE 1 'the source position at the conveyor was not occupied
3250 IF P1AV = 0 THEN
3260 M_00# = 1
3270 '*****
3280 RALA$="11" 'Alarma 11: No se movió un palé válido
3290 '*****
3300 GOTO *ENDE
3310 ENDIF
3320 BREAK
3330 CASE 2
3340 IF P2AV = 0 THEN
3350 M_00# = 1
3360 '*****
3370 RALA$="11" 'Alarma 11: No se movió un palé válido

```


89

- **TMP.POS**

```

DEF POS PHELP1=(324.42,-145.63,187.08,179.87,-0.07,68.72,0.00,0.00)(7,0)
DEF POS PHELP2=(314.48,25.28,159.64,179.87,-0.06,97.71,0.00,0.00)(7,0)
DEF POS PINIT=(354.18,-72.73,367.67,-176.87,-2.82,59.67)(7,0)
DEF POS PPAL(1)=(262.82,173.40,159.27,179.77,-0.78,28.04,0.00)(7,0)
DEF POS PPAL(2)=(327.75,-39.25,158.54,-179.91,-0.16,-17.11,0.00)(7,0)
DEF POS PPAL(3)=(223.04,-234.68,158.15,179.79,0.03,-61.86,0.00)(7,0)
DEF POS PPAL(4)=(10.44,-299.30,158.27,179.67,-0.48,-107.60,0.00)(7,0)
DEF POS PPAL(5)=(303.17,-350.36,351.79,-179.73,0.01,97.17,0.00)(7,0)
DEF POS PPAL(6)=(99.12,-315.78,198.97,-179.73,0.00,97.17,0.00)(7,0)
DEF POS PPAL(7)=(313.15,13.99,177.08,179.82,-2.88,91.46,0.00)(7,0)
DEF POS PPAL(8)=(259.99,-192.34,177.06,179.82,-2.88,91.46,0.00)(7,0)
DEF POS PPAL(9)=(109.54,-281.21,177.08,-179.73,0.01,122.62,0.00)(7,0)
DEF POS PPAL(10)=(-8.65,-286.38,177.06,-179.73,0.01,97.17,0.00)(7,0)
DEF POS PPAL(11)=(225.61,-350.36,224.72,-179.73,0.01,97.17,0.00)(7,0)
DEF POS PPAL(12)=(143.96,-310.97,170.30,-178.39,-0.26,-38.86,0.00)(7,0)
DEF POS PPAL(13)=(324.69,-146.23,174.48,-178.66,-1.34,1.38,0.00)(7,0)
DEF POS PPAL(14)=(411.25,90.15,453.92,-179.46,2.07,-1.55,0.00)(7,0)
DEF POS PPAL(15)=(472.49,17.35,159.10,179.73,-0.33,89.15,0.00)(7,0)

```

- **TMBP.MB4**

```

1000 '|-----
1010 '| Programa modificado del original MBP de Christian Heinisch Festo Didactic
1020 '| TMoveBasePlate(Source, Target) VFF
1030 '|
1040 '| Parameter: MSource/MTarget = 1 --> Retrieve from / Store to : Pallet position 1
1050 '| Parameter: MSource/MTarget = 2 --> Retrieve from / Store to : Pallet position 2
1060 '| Parameter: MSource/MTarget = 3 --> Retrieve from / Store to : Pallet position 3
1070 '| Parameter: MSource/MTarget = 4 --> Retrieve from / Store to : Pallet position 4
1080 '| Parameter: MSource/MTarget = 5 --> Retrieve from / Store to : assemble position
1090 '|-----
1100 '| Autor: Tania Martínez 3 de dic de 2015
1110 '|
1120 '| folgende Positionen müssen geteacht werden / these positions must be teached
1130 '|-----
1140 '| PBP(1)  pallet place 1 / Palettenposition 1
1150 '|-----
1160 '| PBP(2)  pallet place 2 / Palettenposition 2
1170 '|-----
1180 '| PBP(3)  pallet place 3 / Palettenposition 3
1190 '|-----
1200 '| PBP(4)  pallet place 4 / Palettenposition 4
1210 '|-----
1220 '| PBP(5)  assemble position / Montageplatz
1230 '|-----
1240 '|-----
1250 '| PInit
1260 '|-----
1270 '| Define inputs
1280 DEF IO ASTART  = BIT,1           'start control panel
1290 DEF IO ASTOP   = BIT,2           'stop control panel
1300 DEF IO KSWITCH = BIT,3           'Auto/Man.
1310 DEF IO RRESET  = BIT,4           'reset control panel

```

[illegible]

[illegible]

```

2440 UCLMPWP=0
2450 '-----
2460 GOSUB *ENVIAR 'Envía el nuevo estado de sensores
2470 '-----
2480 BREAK
2490 END SELECT
2500 '-----
2510 ' Test target position
2520 '-----
2530 *TESTT
2540 IF (MTARGET% < 1 OR MTARGET% > 5) THEN 'check if the Paramter is in the right range
2550 M_00# = 40 'write to return parameter 40 for "target does not exist"
2560 '-----
2570 RALA$="14" 'Alarma: 14 problema en SCADA principal.
2580 '-----
2590 GOTO *ENDE 'jump to label *ENDE
2600 ENDIF
2610 SELECT MTARGET% 'check if the Target Position is occupied
2620 CASE 1
2630 IF P1AV = 0 THEN
2640 M_00# = 3
2650 '-----
2660 RALA$="04" 'Alarma 4: No hay palé donde guardar placa base
2670 '-----
2680 GOTO *ENDE
2690 ENDIF
2700 BREAK
2710 CASE 2
2720 IF P2AV = 0 THEN
2730 M_00# = 3
2740 '-----
2750 RALA$="04" 'Alarma 4: No hay palé donde guardar placa base
2760 '-----
2770 GOTO *ENDE
2780 ENDIF
2790 BREAK
2800 CASE 3
2810 IF P3AV = 0 THEN
2820 M_00# = 3
2830 '-----
2840 RALA$="04" 'Alarma 4: No hay palé donde guardar placa base
2850 '-----
2860 GOTO *ENDE
2870 ENDIF
2880 BREAK
2890 CASE 4
2900 IF P4AV = 0 THEN
2910 M_00# = 3
2920 '-----
2930 RALA$="04" 'Alarma 4: No hay palé donde guardar placa base
2940 '-----
2950 GOTO *ENDE
2960 ENDIF
2970 BREAK
2980 CASE 5
2990 IF FWORKP = 1 THEN

```



```

3560 '_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*'
3570 DLY 0.5                                'delay 0.5 s
3580 HOPEN 1                               'open hand
3590 WAIT GROPEN = 1                       'wait till hand opened
3600 DLY 0.5                              'delay 0.5 s
3610 MVS AUXPOS,-PLACE1%                  'move straight to a position which is Place1 mm above the target p
3620 ' COLCHK ON
3630 IF MTARGET% = 5 THEN                 'if target is the assembleplace then clamp base plate
3640     UCLMPWP = 0
3650     CLMPWP = 1
3660     DLY 1
3670     CLMPWP=0
3680 '_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*'
3690     GOSUB *ENVIAR
3700 '_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*'
3710     IF FWORKP = 0 OR FCYL1 = 0 OR FCYL2 = 0 THEN
3720         M_00# = 4
3730 '_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*'
3740         RALA$="03"                     'Alarma 3: No hay placa base
3750 '_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*'
3760         CLMPWP = 0
3770         UCLMPWP = 1
3780         DLY 4 'se altera este tiempo porque 0.5 era muy corto
3790         UCLMPWP=0
3800         MOV PINIT
3810 '_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*'
3820         RPOS$="I"
3830 '_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*'
3840         GOTO *ENDE
3850     ENDIF
3860 ENDIF
3870 GOTO *ENDE
3880 '-----*****-----
3890 ' Subrutina para enviar datos al puerto, el dato a enviar debe estar en RPOS$
3900 '-----
3910 *ENVIAR
3920 PRINT #6, P1AV
3930 PRINT #6, P2AV
3940 PRINT #6, P3AV
3950 PRINT #6, P4AV
3960 PRINT #6, FWORKP
3970 PRINT #6, FED1AV
3980 PRINT #6, FED2AV
3990 PRINT #6, PENAV
4000 PRINT #6, FCYL1
4010 PRINT #6, FCYL2
4020 PRINT #6, "P"
4030 PRINT #6, RPOS$
4040 PRINT #6, "A"
4050 PRINT #6, RALA$
4060 PRINT #6, "F"
4070 RETURN
4080 '-----
4090 *ENDE
4100 CNT 0
4110 '_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*'

```



```

360 DEF IO P2AV      = BIT,6           'pallet 2 available
370 DEF IO P3AV      = BIT,7           'pallet 3 available
380 DEF IO P4AV      = BIT,8           'pallet 4 available
390 DEF IO FCYL1     = BIT,9           'part clamped mit cylinder 1
400 DEF IO FCYL2     = BIT,10          'part clamped mit cylinder 2
410 DEF IO FWORKP    = BIT,11          'base plate available in assembling place
420 DEF IO FED1AV    = BIT,12          'part in feeder 1 available
430 DEF IO FED2AV    = BIT,13          'part in feeder 2 available
440 DEF IO PENAV     = BIT,14          'pen available
450 DEF IO GROPEN    = BIT,900         'gripper opened
460 DEF IO GRCLOSE   = BIT,901         'gripper closed
470 'Define outputs
480 DEF IO HSTART    = BIT,0           'lamp start control panel
490 DEF IO HRESET    = BIT,1           'lamp reset control panel
500 DEF IO LEDQ1     = BIT,2           'lamp Q1 control panel
510 DEF IO LEDQ2     = BIT,3           'lamp Q2 control panel
520 DEF IO CLMPWP    = BIT,5           'clamp base plate
530 DEF IO UCLMPWP   = BIT,4           'unclamp base plate
540 'Define arrays
550 DIM PFD(2)        'feeder positions
560 DIM PASM(2)       'assemble positions
570 'Define local variables
580 DEF INTE PICK1    'pick offset
590 DEF INTE PLACE1   'place offset
600 DEF INTE MSOURCE  'source parameter
610 DEF INTE MTARGET  'target parameter
620 DEF INTE MANGLE   'vision angle return parameter
630 '*****
640 DEF INTE VALOR    'variable auxiliar para realizar operaciones
650 DEF CHAR RPOS     'variables donde se envía información al puerto
660 DEF CHAR RALA     '
670 '*****
680 'Define positions
690 DEF POS PHELP1    'help position 1
700 DEF POS PTEST    'vision test position
710 DEF POS PINIT    'initial position
720 DEF POS PANGLE    'angle offset position
730 DEF POS AUXPOS    'helpposition
740 PHELP1 = (+100.00,+0.00,+0.00,+0.00,+0.00,+0.00)
750 PANGLE = (+0.00,+0.00,+0.00,+0.00,+0.00,+0.00)
760 PHELPA = (+0.00,+0.00,+0.00,+90.00,+0.00,+0.00)
770 MANGLE% = 0
780 ACCEL 10 , 10    'define acceleration
790 JOVRD 50         'define joint override
800 SPD 300          'define interpolation speed
810 MSOURCE% = M_00#  'write the submit parameter 1 into the Variable MSource
820 MTARGET% = M_01#  'write the submit parameter 2 into the Variable MTarget
830 MANGLE% = M_02#   'write the submit parameter 3 into the Variable MAngle
840 M_00# = 0         'clear submit parameter 1
850 M_01# = 0         'clear submit parameter 2
860 M_02# = 0         'clear submit parameter 3
870 '-----
880 ' Test source position
890 '-----
900 '*****-Abre el puerto para comunicarse con la PC *****
910 OPEN "COM6:" AS #6

```


[illegible]

```

2040      HCLOSE 1                                'close hand
2050      WAIT GRCLOSE = 1                        'wait till hand is closed
2060      DLY 0.5                                'delay 0.5 s
2070      MVS PFD(MSOURCE%),-PICK1%              'move straight to a position which is Pick1 mm above the Source
2080      '-----
2090      GOSUB *ENVIAR
2100      '-----
2110  ENDIF
2120  IF MSOURCE%= 5 THEN
2130      PANGLE.C = RAD(MANGLE%)                 'write received turning angle into the position variable PTEST
2140      PAUX=PTEST+PANGLE
2150      PAUX1=PAUX
2160      PAUX.Z=PAUX.Z+50
2170      MOV PAUX TYPE 0,0                       'move to a position Pick1 mm above the Test position
2180      '-----
2190      RPOS$ = "9"
2200      GOSUB *ENVIAR
2210      '-----
2220      HOPEN 1                                'open hand
2230      WAIT GROPEN = 1                        'wait till hand is opened
2240      MVS PAUX1 TYPE 0,0                     'move straight to test position
2250      DLY 0.1
2260      HCLOSE 1                                'close hand
2270      WAIT GRCLOSE = 1                        'wait till hand is closed
2280      DLY 0.5                                'delay 0.5
2290      MVS PAUX TYPE 0,0                     'move straight to a position which is Pick1 mm above the Test pos.
2300  ENDIF
2310  '-----
2320  ' Place workpiece at target position
2330  '-----
2340  *PLACE
2350  PLACE1% = 50
2360  IF (MTARGET% = 3 OR MTARGET% =4) THEN
2370      AUXPOS = PASM(MTARGET%-2)
2380      PLACE1% = 120
2390      '-----
2400      RPOS$ = "5"
2410      '-----
2420  ENDIF
2430  IF MTARGET% = 5 THEN
2440      AUXPOS = PTEST
2450      PLACE1%=50
2460      RPOS$ = "9"
2470  ENDIF
2480  MOV AUXPOS,-PLACE1%                       'move to 140 mm above position
2490      '-----
2500  GOSUB *ENVIAR
2510      '-----
2520  MVS AUXPOS                                'move straight to position
2530  DLY 0.5                                'delay 0.5 s
2540  HOPEN 1                                'open hand
2550  WAIT GROPEN = 1
2560  MVS AUXPOS,-PLACE1%                       'move straight to 140 mm above position
2570  IF MTARGET% = 5 THEN
2580      MOV PTEST+PHELP1,-50
2590  ENDIF

```


[illegible]


```

2180 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*
2190     GOTO *ENDE
2200     ENDIF
2210 BREAK
2220 CASE 4
2230     IF P4AV = 0 THEN
2240         M_00# = 1
2250 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2260         RALA$="08"                                'Alarma 8: No hay palé con porta lapicero
2270 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2280         GOTO *ENDE
2290     ENDIF
2300 BREAK
2310 END SELECT
2320 '-----
2330 ' Test target position
2340 '-----
2350 *TESTT
2360 IF (MTARGET% < 1 OR MTARGET% > 5) THEN 'check if the Paramter Msource is in the right range
2370     M_00# = 30                                     'write to return parameter 30 for "source does not exist"
2380 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2390     RALA$="14"                                'Alarma 14: problema en SCADA principal.
2400 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2410     GOTO *ENDE                                'jump to label *ENDE
2420 ENDIF
2430 SELECT MTARGET%                                'check if pallet is available on source
2440 CASE 1
2450     IF P1AV = 0 THEN
2460         M_00# = 3
2470 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2480         RALA$="08"                                'Alarma 8: No hay palé con porta lapicero
2490 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2500         GOTO *ENDE
2510     ENDIF
2520 BREAK
2530 CASE 2
2540 IF P2AV = 0 THEN
2550     M_00# = 3
2560 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2570     RALA$="08"                                'Alarma 8: No hay palé con porta lapicero
2580 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2590     GOTO *ENDE
2600 ENDIF
2610 BREAK
2620 CASE 3
2630     IF P3AV = 0 THEN
2640         M_00# = 3
2650 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2660         RALA$="08"                                'Alarma 8: No hay palé con porta lapicero
2670 '._*._*._*._*._*._*._*._*._*._*._*._*._*._*._*._
2680         GOTO *ENDE
2690     ENDIF
2700 BREAK
2710 CASE 4
2720     IF P4AV = 0 THEN
2730         M_00# = 3

```



```

2740 '*****_
2750     RALA$="08"                                'Alarma 8: No hay palé con porta lapicero
2760 '*****_
2770     GOTO *ENDE
2780 ENDIF
2790 BREAK
2800 CASE 5
2810     IF FWORKP = 0 OR FCYL1 = 0 OR FCYL2 = 0 THEN      'check if base plate is clamped
2820         M_00# = 8
2830 '*****_
2840         RALA$="03"                                'Alarma 3: No hay placa base
2850 '*****_
2860         GOTO *ENDE
2870     ENDIF
2880 BREAK
2890 END SELECT
2900 '-----
2910 ' Pick workpiece at source position
2920 '-----
2930 *PICK
2940 COLLVL 50,120,120,100,120,100,100,100
2950 ' COLCHK ON
2960 CNT 1
2970 ' MOV PINIT                                'move to initial position
2980 HOPEN 1                                'open hand
2990 WAIT GROPEN = 1                        'wait till hand is opened
3000 MOV PINIT
3010 PICK1% = 200                        'write 80 in Pick1
3020 IF MSOURCE% = 5 THEN PICK1% = 200
3030 MOV PPH(MSOURCE%),-PICK1%      'move to a position which is Pick1 mm above the Source position
3040 '*****_
3050 RPOS$ = HEX$(MSOURCE%)
3060 GOSUB *ENVIAR
3070 '*****_
3080 MVS PPH(MSOURCE%)                'move straight to source position
3090 DLY 0.5                        'delay 0.5 s
3100 HCLOSE 1                        'close hand
3110 WAIT GRCLOSE = 1                'wait till hand closed
3120 DLY 0.5                        'delay 0.5 s
3130 MVS PPH(MSOURCE%),-PICK1%      'move to a position which is Pick1 mm above the Source position
3140 IF MSOURCE% <> 5 GOTO *PLACE
3150 MOV PINIT
3160 '-----
3170 ' Place workpiece at target position
3180 '-----
3190 *PLACE
3200 PLACE1% = 200                        'write 80 in Place1
3210 IF MTARGET% = 5 THEN
3220     PLACE1% = 200
3230 ENDIF
3240 MOV PPH(MTARGET%),-PLACE1%      'move to a position which is Place1 mm above the target pos.
3250 '*****_
3260 RPOS$ = HEX$(MTARGET%)
3270 GOSUB *ENVIAR
3280 '*****_
3290 MVS PPH(MTARGET%)                'move straight to target position

```


[illegible]

[illegible]

- **TASM.POS**

```
DEF POS PHELP1=(234.97,566.42,289.87,-78.74,4.06,-13.20,0.00)(6,1048576)
DEF POS PHELP2=(0.00,-13.00,40.00,0.00,0.00,0.00,0.00,0.00)
DEF POS PHELP3=(0.00,0.00,120.00,0.00,0.00,0.00,0.00,0.00)
DEF POS PPF1=(-30.04,549.65,282.18,-82.62,-1.24,21.52,0.00)(6,1048576)
DEF POS PPAS=(171.17,501.36,75.31,177.94,-88.43,136.38,0.00)(6,0)
DEF POS PINIT=(287.05,156.28,334.16,179.89,-0.29,91.46)(7,0)
```

- **TENC.MB4**

```
10 '|-----
20 '| ICIM3000
30 '| TENC: Programa para SCADA en otra PC
40 '| Utilizado para enviar la información indicada en el plan de proceso, además
50 '| envía estado de sensores y posición
60 '| Realizado por: Tania Martinez 08 dic de 2015
70 '|-----
80 DEF IO P1AV = BIT,5 'pallet 1 available
90 DEF IO P2AV = BIT,6 'pallet 2 available
100 DEF IO P3AV = BIT,7 'pallet 3 available
110 DEF IO P4AV = BIT,8 'pallet 4 available
120 DEF IO FCYL1 = BIT,9 'part clamped mit cylinder 1
130 DEF IO FCYL2 = BIT,10 'part clamped mit cylinder 2
140 DEF IO FWORKP = BIT,11 'base plate available in assembling place
150 DEF IO FED1AV = BIT,12 'part in feeder 1 available
160 DEF IO FED2AV = BIT,13 'part in feeder 2 available
170 DEF IO PENAV = BIT,14 'pen available
180 DEF INTE NCOD1
190 DEF INTE NCOD2
200 DEF CHAR RESP
210 NCOD1% = M_00# 'guarda el parámetro 1 (código) en la variable NCOD1
220 NCOD2% = M_01# 'guarda el parámetro 2 (código) en la variable NCOD2
230 M_00# = 0 'limpia el parámetro 1
240 M_01# = 0 'limpia el parámetro 2
250 M_02# = 0 'limpia el parámetro 3
260 OPEN "COM6:" AS #6 'abre el puerto de comunicaciones
270 RESP$ = HEX$(NCOD1%)
280 GOSUB *ENVCOD
290 DLY 1 'Hace una espera de 1 s para enviar el otro código
300 RESP$ = HEX$(NCOD2%)
310 GOSUB *ENVCOD
320 GOTO *FIND
330 *ENVCOD
340 PRINT #6, P1AV 'Envía estado de sensores
350 PRINT #6, P2AV
360 PRINT #6, P3AV
370 PRINT #6, P4AV
380 PRINT #6, FWORKP
390 PRINT #6, FED1AV
400 PRINT #6, FED2AV
410 PRINT #6, PENAV
420 PRINT #6, FCYL1
430 PRINT #6, FCYL2
440 PRINT #6, "P"
```

```

450 PRINT #6, "I"           'Envía posicion
460 PRINT #6, "A"
470 PRINT #6, RESP$         'Envía codigo
480 PRINT #6, "F"
490 RETURN
500 *FIND
510 CLOSE #6
520 END

```

- **TENC.POS**

(Vacío, este programa no requiere posiciones)

14.2.2 SCADA modalidad 2

- **ABC4.MB4**

```

1000 |-----
1010 | ICIM3000
1020 | Programa para ensamblar 3 diferentes set de escritorio
1030 | Se debe indicar el tipo: Estándar con lapicero (1), Estándar sin lapicero (2) y modificado sin lapicero (3)
1040 | Indicar en que receptor para palé esta la placa base (1,2,3 o 4) y el porta lapicero
1050 |-----Deben estar grabadas estas posiciones en memoria-----
1060 | PBP(1)  Receptor de palé de placa base 1
1070 |-----
1080 | PBP(2)  Receptor de palé de placa base 2
1090 |-----
1100 | PBP(3)  Receptor de palé de placa base 3
1110 |-----
1120 | PBP(4)  Receptor de palé de placa base 4
1130 |-----
1140 | PBP(5)  Posición de ensamblado para placa base
1150 |-----
1160 | PFD(1)  Alimentador de higrómetros
1170 |-----
1180 | PFD(2)  Alimentador de termómetros
1190 |-----
1200 | PASM(1) Posición de ensamblado (agujero 3 de placa base de izquierda a derecha)
1210 |-----
1220 | PASM(2) Posición de ensamblado (agujero 2 de placa base de izquierda a derecha)
1230 |-----
1240 | PASM(3) Posición de ensamblado (agujero 1 de placa base de izquierda a derecha)
1250 |-----
1260 | PPH(1)  Receptor de palé de porta lapicero 1
1270 |-----
1280 | PPH(2)  Receptor de palé de porta lapicero 2
1290 |-----
1300 | PPH(3)  Receptor de palé de porta lapicero 3
1310 |-----
1320 | PPH(4)  Receptor de palé de porta lapicero 4
1330 |-----
1340 | PPH(5)  posición de ensamblado (agujero 3 de placa base de izquierda a derecha)
1350 |-----

```

```

1360 '| PPH(6) posición de ensamblado (agujero 2 de placa base de izquierda a derecha)
1370 '|-----
1380 '| PPFDF alimentador de lapiceros
1390 '|-----
1400 '| PPAS posición de ensamblado de lapiceros
1410 '|-----
1420 '| PHELP1 posición de ayuda para colocar lapicero
1430 '|-----
1440 '| Autora: Tania Martínez (modificación de los programas de Christian Heinisch de Festo
1450 '| Fecha: 3 de diciembre de 2015.
1460 '|-----
1470 'Definir entradas
1480 DEF IO ASTART = BIT,1 'start del panel de control
1490 DEF IO ASTOP = BIT,2 'stop del panel de control
1500 DEF IO KSWITCH = BIT,3 'Auto/Man.
1510 DEF IO RRESET = BIT,4 'reset del panel de control
1520 DEF IO P1AV = BIT,5 'Sensor 1 de receptor de palés
1530 DEF IO P2AV = BIT,6 'Sensor 2 de receptor de palés
1540 DEF IO P3AV = BIT,7 'Sensor 3 de receptor de palés
1550 DEF IO P4AV = BIT,8 'Sensor 4 de receptor de palés
1560 DEF IO FCYL1 = BIT,9 'cilindro uno para sujetar pieza
1570 DEF IO FCYL2 = BIT,10 'cilindro dos para sujetar pieza
1580 DEF IO FWORKP = BIT,11 'sensor de placa base en la posición de ensamble
1590 DEF IO FED1AV = BIT,12 'sensor en alimentador 1
1600 DEF IO FED2AV = BIT,13 'sensor en alimentador 2
1610 DEF IO PENAV = BIT,14 'sensor de lapiceros
1620 DEF IO GROPEN = BIT,900 'pinza abierta
1630 DEF IO GRCLOSE = BIT,901 'pinza cerrada
1640 'Definir salidas
1650 DEF IO HSTART = BIT,0 'Lámpara de start del panel de control
1660 DEF IO HRESET = BIT,1 'lámpara de reset del panel de control
1670 DEF IO LEDQ1 = BIT,2 'lámpara Q1 del panel de control
1680 DEF IO LEDQ2 = BIT,3 'lámpara Q2 del panel de control
1690 DEF IO CLMPWP = BIT,5 'actuador para sujetar placa base
1700 DEF IO UCLMPWP = BIT,4 'actuador para liberar placa base
1710 'Definir arreglos
1720 DIM PFD(2) 'Dos alimentadores
1730 DIM PASM(3) 'Tres posiciones en placa base para ensamblar
1740 DIM PBP(5) 'cinco posiciones donde colocar placa base
1750 DIM PPH(6) 'seis posiciones donde colocar porta lapiceros
1760 'Definir variables
1770 DEF INTE PICK1 'variable donde se almacena una distancia de seguridad para acercarse a tomar algo
1780 DEF INTE PLACE1 'variable donde se almacena una distancia de seguridad para acercarse a colocar algo
1790 DEF INTE ORDER
1800 DEF INTE PBSOURCE 'origen de la placa base
1810 DEF INTE PBTARGET 'destino de la placa base
1820 DEF INTE PLSOURCE 'origen del porta lapicero
1830 DEF INTE PLTARGET 'destino del porta lapicero
1840 DEF INTE INSOURCE 'origen del instrumento
1850 DEF INTE INTARGET 'destino del instrumento
1860 DEF INTE VARI 'variable auxiliar
1870 DEF INTE VARI2 'variable auxiliar 2
1880 DEF CHAR SORDER
1890 DEF CHAR SPBSOU
1900 DEF CHAR SPLSOU
1910 DEF CHAR DATA 'donde se guardan los datos leídos del puerto

```



```

1920 DEF CHAR RESP      'donde se envía información al puerto
1930 'Definir posiciones
1940 DEF POS PINIT
1950 DEF POS PCLAMP
1960 DEF POS AUXPOS
1970 DEF POS PHELP1
1980 DEF POS PHELP2
1990 DEF POS PHELP3
2000 DEF POS PPF
2010 DEF POS PPAS
2020 PCLAMP = (+1.00,+1.00,+0.00,+0.00,+0.00,+0.00)
2030 PHELP2 = (+0.00,-13.00,+40.00,+0.00,+0.00,+0.00)
2040 PHELP3 = (+0.00,+0.00,+120.00,+0.00,+0.00,+0.00)
2050 PLACE1% = 170
2060 PICK1% = 170
2070 ACCEL 10 , 10      'define aceleración
2080 JOVRD 50           'define joint override
2090 SPD 300            'define velocidad de interpolación
2100 M_00# = 0          'clear submit parameter 1
2110 M_01# = 0          'clear submit parameter 2
2120 M_02# = 0          'clear submit parameter 3
2130 M_19# = 0          'clear handshake variable
2140 *****Programa*****
2150 OPEN "COM6:" AS #6  'Abre el puerto para comunicarse con la computadora
2160 SERVO ON
2170 RESP$="18" 'Envía el código 18 para indicar que ya inició el programa
2180 GOSUB *ENVIAR
2190 DLY 0.1 'Hace una espera pequeña para enviar de nuevo el código
2200 RESP$="18" 'debido a que el TCP drive de Ignition cuando se enciende el robot la primera
2210 GOSUB *ENVIAR 'vez y se escribe se come el primer caracter, después ya no ocurre esto
2220 *INICIO
2230 GOSUB *RECIB
2240 RESP$="16" 'envía el código 16 para indicar que arranca el ensamblado
2250 GOSUB *ENVIAR
2260 SORDER$=LEFT$(DATA$,1) 'extrae tipo de set
2270 SPBSOU$=MID$(DATA$,2,1) 'extrae donde está la placa base
2280 SPLSOU$=RIGHT$(DATA$,1) 'extrae donde está el porta lapicero
2290 ORDER%=VAL(SORDER$) 'Convierte el texto a valor numérico
2300 PBSOURCE%=VAL(SPBSOU$)
2310 PLSOURCE%=VAL(SPLSOU$)
2320 PBTARGET%=5 'El destino de la placa base al inicio es la posición de ensamble (5)
2330 IF (ORDER% < 1 OR ORDER% > 4) THEN 'checa si el parámetro de orden está en el rango correcto
2340   RESP$= "13" 'Alarma 13: Fallo en la comunicación
2350   GOTO *FINERR 'Envía Fallo y sale del programa
2360 ENDIF
2370 IF ORDER%=4 THEN GOTO *ENDE 'Va a *Ende donde envía que el programa finalizó correctamente
2380 *DENUEV1
2390 VARI2%=0
2400 GOSUB *PBSAVAIL 'revisar si hay palé con placa base donde se indicó que se iba a tomar
2410 IF VARI2%=55 THEN GOTO *DENUEV1
2420 *DENUEV2
2430 VARI2%=0
2440 GOSUB *PBTAVAIL 'revisar si hay palé donde colocar la placa base o si está vacío la pos. de ensamble
2450 IF VARI2%=55 THEN GOTO *DENUEV2
2460 *DENUEV3
2470 VARI2%=0

```

```

2480 GOSUB *MOVEPB          'si todo está bien manda a mover la placa base
2490 IF VARI2%=55 THEN GOTO *DENUEV3
2500 VARI%=PBSOURCE%       'cuando esté ensamblado el origen será el destino y el destino el origen
2510 PBSOURCE%=PBTARGET%   'se hace ese cambio de valores aquí
2520 PBTARGET%=VARI%
2530 INTARGET%=1 'el destino del primer inst. (higrómetro) será el agujero 1 de la placa base de izq a der
2540 IF (ORDER% =1 OR ORDER% =2) THEN 'si era set de escritorio estándar con lapicero (1) o sin (2)
2550 PLTARGET%=5 'la posición final del portaplapicero es la 5 (agujero 3 de placa base de izq. a der.)
2560 VARI%=2 'el destino del segundo instrumento (termómetro) será el agujero 2
2570 ELSE 'sino si era el set modificado
2580 PLTARGET%=6 'la posición final del lapicero es la 6 (agujero 2 de la placa base de izq. a der.)
2590 VARI%=3 'el destino del segundo instrumento (termómetro) será el agujero 3
2600 ENDIF
2610 INSOURCE%=1 'Instrumento a tomar: higrómetro
2620 *DENUEV4
2630 VARI2%=0
2640 GOSUB *INSAVAIL 'verificar si hay termómetros disponibles
2650 IF VARI2%=55 THEN GOTO *DENUEV4 'si vari2% =55 es que se va a revisar de nuevo sino continua
2660 *DENUEV5
2670 VARI2%=0
2680 GOSUB *MOVEIN 'mover el higrómetro
2690 IF VARI2%=55 THEN GOTO *DENUEV5 'si vari2% = 55 es que se va a revisar de nuevo sino continua
2700 INSOURCE%=2 'Instrumento a tomar: higrómetro
2710 INTARGET%=VARI% 'el destino es lo que indicaba la variable "VARI"
2720 *DENUEV6
2730 VARI2%=0
2740 GOSUB *INSAVAIL 'verificar si hay termómetros disponibles
2750 IF VARI2%=55 THEN GOTO *DENUEV6 'si vari2% =55 es que se va a revisar de nuevo sino continua
2760 *DENUEV7
2770 VARI2%=0
2780 GOSUB *MOVEIN 'mover el termómetro
2790 IF VARI2%=55 THEN GOTO *DENUEV7 'si vari2% = 55 es que se va a revisar de nuevo sino continua
2800 *DENUEV8
2810 VARI2%=0
2820 GOSUB *PLSAVAIL 'revisar si hay palé con porta lapicero donde se indicó que se iba a tomar
2830 IF VARI2%=55 THEN GOTO *DENUEV8 'si vari2%= 55 es que se va a revisar de nuevo sino continua
2840 *DENUEV9
2850 VARI2%=0
2860 GOSUB *MOVEPL 'mover porta lapicero
2870 IF VARI2%=55 THEN GOTO *DENUEV9 'si vari2%= 55 es que se va a revisar de nuevo sino continua
2880 *DENUEV10
2890 VARI2%=0
2900 GOSUB *LAPIC 'Poner lapicero si es necesario
2910 IF VARI2%=55 THEN GOTO *DENUEV10 'si vari2% =55 es que se va a revisar de nuevo sino continua
2920 *DENUEV11
2930 VARI2%=0
2940 GOSUB *PBSAVAIL 'liberar la placa base de la posición de ensamblado
2950 IF VARI2%=55 THEN GOTO *DENUEV11 'si vari2% =55 es que se va a revisar de nuevo sino continua
2960 *DENUEV12
2970 VARI2%=0
2980 GOSUB *PBTAVAIL 'verificar si hay palé de placa base donde colocar el ensamble
2990 IF VARI2%=55 THEN GOTO *DENUEV12 'si vari2%= 55 es que se va a revisar de nuevo sino continua
3000 *DENUEV13
3010 VARI2%=0
3020 GOSUB *MOVEPB
3030 IF VARI2%=55 THEN GOTO *DENUEV13 'si vari2% =55 es que se va a revisar de nuevo sino continua

```

```

3040 RESP$="15" 'envía el código 15 para indicar que finalizó correctamente el ensamblado
3050 PRINT #6,RESP$
3060 PRINT #6, "F"
3070 GOTO *INICIO
3080 '-----***-----
3090 ' Subrutina para enviar datos al puerto, el dato a enviar debe estar en RESP$
3100 '-----
3110 *ENVIAR
3120 PRINT #6,RESP$
3130 PRINT #6, "F"
3140 RETURN
3150 '-----***-----
3160 ' Subrutina para recibir datos de puerto, el dato queda en DATA$
3170 '-----
3180 *RECIB
3190 INPUT #6, DATA$
3200 RETURN
3210 '-----***-----
3220 ' Subrutina para verificar si continuar o finalizar el programa
3230 '-----
3240 *VERIFI
3250 GOSUB *ENVIAR
3260 GOSUB *RECIB
3270 IF DATA$ ="400" THEN GOTO *ENDE 'Si se recibió 400, es que finalice el programa
3280 RESP$="16" 'envía el código 16 para indicar que continua el ensamble
3290 GOSUB *ENVIAR
3300 DLY 1
3310 VARI2%=55
3320 RETURN
3330 '-----
3340 ' Verificar si hay palé con placa base
3350 '-----
3360 *PBSAVAIL
3370 IF (PBSOURCE% < 1 OR PBSOURCE% > 5) THEN 'checa si el parámetro de placa base está en el
3380     RESP$= "13"                      ' rango correcto, sino envía Alarma 13: Fallo en la comunicación
3390     GOTO *FINERR                      'Envía Fallo y sale del programa
3400 ENDIF
3410 SELECT PBSOURCE%                    'chechar si hay palé con placa base
3420 CASE 1
3430     IF P1AV = 0 THEN
3440         RESP$="02"                    'Alarma 2: No hay palé con placa base
3450         GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
3460     ENDIF
3470 BREAK
3480 CASE 2
3490     IF P2AV = 0 THEN
3500         RESP$="02"                    'Alarma 2: No hay palé con placa base
3510         GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
3520     ENDIF
3530 BREAK
3540 CASE 3
3550     IF P3AV = 0 THEN
3560         RESP$="02"                    'Alarma 2: No hay palé con placa base
3570         GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
3580     ENDIF
3590 BREAK

```

```

3600 CASE 4
3610 IF P4AV = 0 THEN
3620     RESP$="02"           'Alarma 2: No hay palé con placa base
3630     GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
3640 ENDIF
3650 BREAK
3660 CASE 5
3670 IF FWORKP = 0 THEN 'si era en la posición de ensamblado verifica que hay algo
3680     RESP$="03"           'sino Alarma 3: No hay placa base
3690     GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
3700 ENDIF
3710 CLMPWP = 0             'desactiva actuadores de sujeción
3720 UCLMPWP = 1
3730 DLY 0.5
3740 UCLMPWP=0
3750 BREAK
3760 END SELECT
3770 RETURN
3780 '-----
3790 ' verificar si hay palé donde colocar la placa base o si está vacío la posición de ensamble
3800 '-----
3810 *PBTAVAIL
3820 SELECT PBTARGET%      'chechar si hay palé donde colocar placa base o posición de ensamble vacía
3830 CASE 1
3840 IF P1AV = 0 THEN
3850     RESP$="04"           'Alarma 4: No hay palé donde guardar la placa base
3860     GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
3870 ENDIF
3880 BREAK
3890 CASE 2
3900 IF P2AV = 0 THEN
3910     RESP$="04"           'Alarma 4: No hay palé donde guardar la placa base
3920     GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
3930 ENDIF
3940 BREAK
3950 CASE 3
3960 IF P3AV = 0 THEN
3970     RESP$="04"           'Alarma 4: No hay palé donde guardar la placa base
3980     GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
3990 ENDIF
4000 BREAK
4010 CASE 4
4020 IF P4AV = 0 THEN
4030     RESP$="04"           'Alarma 4: No hay palé donde guardar la placa base
4040     GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
4050 ENDIF
4060 BREAK
4070 CASE 5
4080 IF FWORKP = 1 THEN      'si el destino es la posición de ensamble verifique que esté vacía
4090     RESP$="05"           'Alarma 5: Hay una placa base en la posición de ensamble
4100     GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
4110 ENDIF
4120 UCLMPWP = 1             'desactiva actuadores de sujeción
4130 DLY 0.5
4140 UCLMPWP = 0
4150 BREAK

```

```

4160 END SELECT
4170 RETURN
4180 '-----
4190 ' Recoge la placa base donde se indicó
4200 '-----
4210 *MOVEPB
4220 COLLVL 50,120,120,100,120,100,100,100 'Especifica el nivel permitible para la deteccion de impactos
4230 CNT 1
4240 HOPEN 1 'abre pinza
4250 WAIT GROPEN = 1 'espera a que esté abierta
4260 MOV PINIT 'se mueve a posición inicial
4270 MOV PBP(PBSOURCE%),-PICK1% 'se mueve a una posición con una distancia de Pick1 mm arriba de
4280 MVS PBP(PBSOURCE%) ' de la posición de origen, luego se mueve directo a la posición de origen
4290 DLY 0.5 'espera 0.5 s
4300 HCLOSE 1 'cierra pinza
4310 WAIT GRCLOSE = 1 'espera que la pinza esté cerrada
4320 DLY 0.5 'espera 0.5 s
4330 MVS PBP(PBSOURCE%),-PICK1% 'se mueve a una distancia Pick1 mm arriba de la posición de origen
4340 IF PBSOURCE% = 5 THEN MOV PINIT 'si era desde la de ensamble se mueve a posición de inicio
4350 '-----
4360 ' Coloca la placa base en la posición final
4370 '-----
4380 AUXPOS = PBP(PBTARGET%) 'guarda la posición final en una variable auxiliar
4390 IF PBTARGET% = 5 THEN
4400 MOV PINIT 'si es la de ensamble se mueve a posición de inicio
4410 AUXPOS = PBP(PBTARGET%)+PCLAMP 'guarda la pos. tomando en cuenta distancia de sujeción
4420 ENDIF
4430 MOV AUXPOS,-PLACE1% 'se mueve a una distancia Place1 mm del destino
4440 ' COLCHK OFF
4450 MVS AUXPOS 'se mueve directo a la posición de destino
4460 DLY 0.5 'espera 0.5 s
4470 HOPEN 1 'abre pinza
4480 WAIT GROPEN = 1 'espera a que la pinza esté abierta
4490 DLY 0.5 'espera 0.5 s
4500 MVS AUXPOS,-PLACE1% 'se mueve directo a una distancia Place1 mm del destino
4510 ' COLCHK ON
4520 IF PBTARGET% = 5 THEN 'Si el destino era la posición de ensamble sujeta la placa base
4530 UCLMPWP = 0
4540 CLMPWP = 1
4550 DLY 1
4560 CLMPWP=0
4570 IF FWORKP = 0 OR FCYL1 = 0 OR FCYL2 = 0 THEN 'verifica si era una placa base sujeta
4580 RESP$="03" 'Alarma 3: No hay placa base en la pos. de ensamblado
4590 CLMPWP = 0
4600 UCLMPWP = 1
4610 DLY 4
4620 UCLMPWP=0
4630 MOV PINIT
4640 GOSUB *VERIFI 'en VERIFI envía alarma y verifica si se revisará de nuevo o si se saldrá del prog.
4650 ENDIF
4660 ENDIF
4670 MOV PINIT
4680 CNT 0
4690 RETURN
4700 '-----
4710 ' Verifica si hay instrumentos disponibles

```

```

4720 '-----
4730 *INSAVAIL
4740 SELECT INSOURCE%           'chechar si hay medidores
4750 CASE 1
4760   IF FED1AV = 0 THEN         'verificar termómetros
4770     RESP$="07"              'Alarma 7: No hay termómetros disponibles
4780     GOSUB *VERIFI           'salta a verificar si se revisará de nuevo o si se saldrá del programa
4790   ENDIF
4800   BREAK
4810 CASE 2
4820   IF FED2AV = 0 THEN
4830     RESP$="06"              'Alarma 6: No hay higrómetros disponibles
4840     GOSUB *VERIFI           'salta a verificar si se revisará de nuevo o si se saldrá del programa
4850   ENDIF
4860   BREAK
4870 END SELECT
4880 RETURN
4890 '-----
4900 ' Toma instrumento
4910 '-----
4920 *MOVEIN
4930 COLLVL 50,180,120,100,200,100,100,100
4940 CNT 1
4950 HOPEN 1                     'abre pinza
4960 WAIT GROOPEN = 1            'espera a que se habra la pinza
4970 PICK1% = 50                 'coloca una distancia de seguridad de 50mm en Pick1
4980 MOV PFD(INSOURCE%),-PICK1%  'se mueve Pick1 mm arriba de donde está el instrumento
4990 MVS PFD(INSOURCE%)          'se mueve directo al instrumento
5000 DLY 0.5                     'espera 0.5s
5010 HCLOSE 1                   'cierra pinza
5020 WAIT GRCLOSE = 1            'espera a que se cierre la pinza
5030 DLY 0.5                     'espera 0.5s
5040 MVS PFD(INSOURCE%),-PICK1%  'se mueve Pick1 mm arriba de donde está el instrumento
5050 '-----
5060 ' Coloca el instrumento en la placa base
5070 '-----
5080 AUXPOS = PASM(INTARGET%)     'guarda la posición de destino
5090 PLACE1% = 120                'ajusta la distancia de seguridad a 120mm
5100 MOV AUXPOS,-PLACE1%          'se mueve Pick1 mm arriba de la posición de destino
5110 MVS AUXPOS                   'se mueve directamente a la posición
5120 DLY 0.5                     'espera 0.5s
5130 HOPEN 1                     'abre pinza
5140 WAIT GROOPEN = 1            'espera a que se abra la pinza
5150 MVS AUXPOS,-PLACE1%          'se mueve Pick1 mm arriba de la posición
5160 CNT 0
5170 RETURN
5180 '-----
5190 ' Verifica si hay palé con porta lapicero
5200 '-----
5210 *PLSAVAIL
5220 IF (PLSOURCE% < 1 OR PLSOURCE% > 4) THEN 'checa si el parámetro de porta lapicero está en el
5230   RESP$= "13"                ' rango correcto sino envía Alarma 13: Fallo en la comunicación
5240   GOTO *FINERR               'Envía Fallo y sale del programa
5250 ENDIF
5260 SELECT PLSOURCE%             'checa si hay palé con porta lapicero
5270 CASE 1

```

```

5280 IF P1AV = 0 THEN
5290     RESP$="08"                                'Alarma 8: No hay palé con porta lapicero
5300     GOSUB *VERIFI                             'salta a verificar si se revisará de nuevo o si se saldrá del programa
5310 ENDIF
5320 BREAK
5330 CASE 2
5340 IF P2AV = 0 THEN
5350     RESP$="08"                                'Alarma 8: No hay palé con porta lapicero
5360     GOSUB *VERIFI                             'salta a verificar si se revisará de nuevo o si se saldrá del programa
5370 ENDIF
5380 BREAK
5390 CASE 3
5400 IF P3AV = 0 THEN
5410     RESP$="08"                                'Alarma 8: No hay palé con porta lapicero
5420     GOSUB *VERIFI                             'salta a verificar si se revisará de nuevo o si se saldrá del programa
5430 ENDIF
5440 BREAK
5450 CASE 4
5460 IF P4AV = 0 THEN
5470     RESP$="08"                                'Alarma 8: No hay palé con porta lapicero
5480     GOSUB *VERIFI                             'salta a verificar si se revisará de nuevo o si se saldrá del programa
5490 ENDIF
5500 BREAK
5510 END SELECT
5520 RETURN
5530 '-----
5540 ' Toma porta lapicero
5550 '-----
5560 *MOVEPL
5570 *PICK
5580 COLLVL 50,120,120,100,120,100,100,100
5590 ' COLCHK ON
5600 CNT 1
5610 HOPEN 1                                     'abre pinza
5620 WAIT GROPEN = 1                             'espera a que esté abierta la pinza
5630 MOV PINIT                                   'mueve a posición inicial
5640 PICK1% = 200                                'coloca una distancia de seguridad de 200mm en Pick1
5650 MOV PPH(PLSOURCE%),-PICK1%                 'se mueve a Pick1mm arriba de la posición de origen
5660 MVS PPH(PLSOURCE%)                         'se mueve directo a la posición
5670 DLY 0.5                                     'espera 0.5 s
5680 HCLOSE 1                                   'cierra pinza
5690 WAIT GRCLOSE = 1                             'espera a que la pinza se cierre
5700 DLY 0.5                                     'espera 0.5s
5710 MVS PPH(PLSOURCE%),-PICK1%                 'se mueve a Pick1 mm arriba de la posición de origen
5720 '-----
5730 'Coloca porta lapicero
5740 '-----
5750 PLACE1% = 200                                'coloca una distancia de seguridad de 200mm en PLACE1
5760 MOV PINIT                                   'se mueve a posición inicial
5770 MOV PPH(PLTARGET%),-PLACE1%                 'se mueve a PLACE1 mm arriba de la posición final
5780 MVS PPH(PLTARGET%)                         'se mueve directo a la posición final
5790 DLY 0.5                                     'espera 0.5 s
5800 HOPEN 1                                     'abre pinza
5810 WAIT GROPEN = 1                             'espera hasta que se abra la pinza
5820 DLY 0.5                                     'espera 0.5 s
5830 MVS PPH(PLTARGET%),-PLACE1%                 'se mueve Place1 mm arriba de la posición final

```

```

5840 MOV PINIT                                'se mueve a la posición de inicio
5850 CNT 0
5860 RETURN
5870 '-----
5880 'Subrutina para verificar si se pondrá o no el lapicero
5890 '-----
5900 *LAPIC
5910 IF (ORDER% = 2 OR ORDER% = 3) THEN GOTO *NOLAP
5920 '-----
5930 ' Verifique que haya lapiceros
5940 '-----
5950 IF PENAV = 0 THEN                        'checa si hay lapiceros
5960     RESP$ = "09"                        'Alarma 9= no hay lapiceros
5970     GOSUB *VERIFI                      'salta a verificar si se revisará de nuevo o si se saldrá del programa
5980 ENDIF
5990 IF VARI2%=55 THEN GOTO *NOLAP
6000 '-----
6010 ' Tomar el lapicero
6020 '-----
6030 COLLVL 50,120,120,100,120,100,100,100
6040 'COLCHK ON
6050 CNT 1
6060 MOV PINIT                                'moverse a posición inicial
6070 HOPEN 1                                'abrir pinza
6080 WAIT GROOPEN = 1                        'esperar que se abra la pinza
6090 MOV PHELP1                                'moverse a posición de ayuda
6100 PICK1% = 25                            'colocar una distancia de seguridad de 25mm
6110 MOV PPFD,-PICK1%                        'moverse Pick1 mm arriba de la posición
6120 MVS PPFD                                'moverse directo a la posición
6130 DLY 0.5                                'esperar 0.5s
6140 HCLOSE 1                                'abrir pinza
6150 WAIT GRCLOSE = 1                        'esperar hasta que se cierre la pinza
6160 DLY 0.5                                'esperar 0.5 s
6170 MVS PPFD+PHELP2                        'alejarse de la posición
6180 '-----
6190 ' colocar lapicero
6200 '-----
6210 PLACE1% = 35                            'distancia de seguridad de 35 mm
6220 MOV PHELP1                                'moverse a posición de ayuda
6230 MOV PPAS+PHELP3                        'moverse a otra posición de ayuda
6240 MVS PPAS                                'moverse directo a posición de ensamble
6250 DLY 0.5                                'esperar 0.5s
6260 HOPEN 1                                'abrir pinza
6270 WAIT GROOPEN = 1                        'esperar hasta que se abra la pinza
6280 DLY 0.5                                'esperar 0.5 s
6290 MVS PPAS+PHELP3                        'alejarse a posición de ayuda
6300 MOV PINIT                                'regresar a posición inicial
6310 CNT 0
6320 *NOLAP
6330 RETURN
6340 *ENDE
6350 RESP$= "17" 'envía el código 17 para indicar que se finalizó el programa en el robot
6360 *FINERR
6370 GOSUB *ENVIAR
6380 SERVO OFF
6390 CLOSE #6

```


6400 END

- **ABC4.POS**

```
DEF POS PINIT=(287.05,156.28,332.87,179.89,-0.29,91.46,0.00,0.00)(7,0)
DEF POS PFD(1)=(-54.27,442.80,83.83,179.52,-22.36,112.71,0.00)(6,0)
DEF POS PFD(2)=(-132.58,401.30,83.40,178.71,-25.33,123.74,0.00)(6,0)
DEF POS PASM(1)=(176.76,405.47,17.28,179.69,0.10,91.34,0.00)(7,0)
DEF POS PASM(2)=(176.97,450.33,17.24,179.69,0.10,91.33,0.00)(7,0)
DEF POS PASM(3)=(176.98,495.09,17.55,179.69,0.10,91.34,0.00)(7,0)
DEF POS PBP(1)=(289.40,123.31,171.40,-180.00,-0.98,27.90,0.00)(7,0)
DEF POS PBP(2)=(309.91,-93.85,171.40,-179.87,-0.30,-16.99,0.00)(7,0)
DEF POS PBP(3)=(172.48,-261.05,171.40,-179.96,-0.69,-62.01,0.00)(7,0)
DEF POS PBP(4)=(-43.65,-281.94,171.40,179.77,-0.48,-106.82,0.00)(7,0)
DEF POS PBP(5)=(177.00,449.43,10.54,179.27,-0.02,89.88,0.00)(7,0)
DEF POS PPH(1)=(290.65,122.50,183.48,179.97,-0.46,28.89)(7,0)
DEF POS PPH(2)=(311.24,-94.66,182.07,179.79,-0.22,-16.25)(7,0)
DEF POS PPH(3)=(173.06,-262.30,181.91,179.80,-0.23,-61.07)(7,0)
DEF POS PPH(4)=(-44.36,-282.99,181.85,179.55,-0.46,-106.50)(7,0)
DEF POS PPH(5)=(176.74,495.47,29.27,179.40,-0.13,90.04)(7,0)
DEF POS PPH(6)=(176.74,451.30,29.27,179.40,-0.13,90.04)(7,0)
DEF POS PCLAMP=(1.00,1.00,0.00,0.00,0.00,0.00)
DEF POS AUXPOS=(172.48,-261.05,171.40,-179.96,-0.69,-62.01,0.00,0.00)(7,0)
DEF POS PHELP1=(234.97,566.42,289.87,-78.74,4.06,-13.20,0.00)(6,1048576)
DEF POS PHELP2=(0.00,-13.00,40.00,0.00,0.00,0.00,0.00,0.00)
DEF POS PHELP3=(0.00,0.00,120.00,0.00,0.00,0.00,0.00,0.00)
DEF POS PPFD=(-30.04,549.65,282.18,-82.62,-1.24,21.52,0.00)(6,1048576)
DEF POS PPAS=(171.17,501.36,75.31,177.94,-88.43,136.38,0.00)(6,0)
```

14.3 Planes de procesos creados en CIROS Production para el SCADA en modalidad 1

- **Plan de procesos principal**

```
; -----*****-----*****-----*****-----
; Plan de procesos para SCADA con otra PC
; Creado por: Tania Martinez
; Date: 09.12.2015
; Time: 09:29:10
; -----
; *-*-*-utiliza macro tareas: asmdeskset1, movtotrans1 y movfromtrans1 para la estación de ensamble*-*-*-
; -----
; BEGIN of automatically generated process plan.
; -----
;
; -----
; BEGIN of user defined process plan part.
; -----
;
```

```

;
; *****
; BEGIN of SplitMergeTwo structure.
; *****
;
; ***** Start two parallel processes. *****
SPLIT1      .SPLIT  SPLIT11, SPLIT21      MERGE1
; ***** Parallel process 1. *****
SPLIT11     .NOOP
;
; *****
; BEGIN of first user defined structure part.
; *****
; +-+ - Antes de iniciar se ejecuta en la estación de ensamble el programa TENC +-+ -
; +-+ - este envía 24d que en el robot se convierte a 18h para indicar inicio del proceso +-+ -
; +++++ Transport: Get new carrier. +++++
10           AssemblyRV3SBRobot   ExecProg("TENC", 24,24,0)
20           Transport            REQUIRE("Stopper_001", 1)
30           .CALC  %_1_CarrierID = 1
; +++++ Station StockSingle: Move pallet from station buffer to conveyor. +++++
40           StockSingle          MovToTrans(42143, 0)
50           Transport            SetProductId(42143, %_1_CarrierID)
; +++++ Transport: Move carrier to processing station. +++++
60           Transport            to_AssemblyRV3SB(1)
; +++En la estación AssemblyRV3SB: Mueve el pale desde el conveyor al receptor de pales utilizando MovFromTrans1 ++
70           .CALL  HS_CLASS_Assembly_MovFromTrans1(42143, 0)
80           Transport            SetProductId("", %_1_CarrierID)
; +++++ Transport: Release carrier. +++++
90           Transport            RELEASE(1)
100          .CALC  %_1_CarrierID = 0
;
; *****
; END of first user defined structure part.
; *****
;
ENDSPLIT11   .NOOP          MERGE1
; ***** Parallel process 2. *****
SPLIT21     .NOOP
;
; *****
; BEGIN of second user defined structure part.
; *****
;
; +++++ Transport: Get new carrier. +++++
110          Transport            REQUIRE("Stopper_001", 2)
120          .CALC  %_2_CarrierID = 2
; +++++ Station StockSingle: Move pallet from station buffer to conveyor. +++++
130          StockSingle          MovToTrans(42104, 0)
140          Transport            SetProductId(42104, %_2_CarrierID)
; +++++ Transport: Move carrier to processing station. +++++
150          Transport            to_AssemblyRV3SB(2)
; +++En la estación AssemblyRV3SB: Mueve el pale desde el conveyor al receptor de pales utilizando MovFromTrans1 ++
160          .CALL  HS_CLASS_Assembly_MovFromTrans1(42104, 0)
170          Transport            SetProductId("", %_2_CarrierID)
; +++++ Transport: Release carrier. +++++
180          Transport            RELEASE(2)

```

```

190          .CALC  %_2_CarrierID = 0
;
; *****
; END of second user defined structure part.
; *****
;
;
ENDSPLIT21          .NOOP
; ***** Parallel processes merged together. *****
MERGE1              .MERGE
;
; *****
; END of SplitMergeTwo structure.
; *****
;
; ++se ejecuta la macro tarea AsmDeskSet1 para iniciar el ensamble del set++++
200          .CALL  HS_CLASS_Assembly_AsmDeskSet1(52675, 0)
;
; *****
; BEGIN of SplitMergeTwo structure.
; *****
;
; ***** Start two parallel processes. *****
SPLIT3              .SPLIT  SPLIT13, SPLIT23      MERGE3
; ***** Parallel process 1. *****
SPLIT13              .NOOP
;
; *****
; BEGIN of first user defined structure part.
; *****
;
; ++++++ Transport: Get new carrier. ++++++
210          Transport      REQUIRE("Stopper_003", 1)
220          .CALC  %_3_CarrierID = 1
; +++En la estación AssemblyRV3SB: Mueve el pale desde el receptor de pales al conveyor utilizando MovToTrans1++
230          .CALL  HS_CLASS_Assembly_MovToTrans1(52675, 0)
240          Transport      SetProductId(52675, %_3_CarrierID)
; ++++++ Transport: Move carrier to processing station. ++++++
250          Transport      to_StockSingle(1)
; ++++++ Station StockSingle: Move pallet from conveyor to station buffer. ++++++
260          StockSingle     MovFromTrans(52675, 0)
270          Transport      SetProductId("", %_3_CarrierID)
; ++++++ Transport: Release carrier. ++++++
280          Transport      RELEASE(1)
290          .CALC  %_3_CarrierID = 0
;
; *****
; END of first user defined structure part.
; *****
;
;
ENDSPLIT13          .NOOP          MERGE3
; ***** Parallel process 2. *****
SPLIT23              .NOOP
;
; *****
; BEGIN of second user defined structure part.
; *****

```

```

;
; +++++ Transport: Get new carrier. +++++
300      Transport      REQUIRE("Stopper_003", 2)
310      .CALC   %_4_CarrierID = 2
; +++En la estación AssemblyRV3SB: Mueve el pale desde el receptor de pales al conveyor utilizando MovToTrans1++
320      .CALL   HS_CLASS_Assembly_MovToTrans1(82200, 0)
330      Transport      SetProductId(82200, %_4_CarrierID)
; +-+-+Al finalizar el ensamble y devolver los materiales a la banda transportadora se ejecuta el programa TENC
; +-+-+este envía 21d que en el robot se convierte a 15h para indicar que terminó el ensamble y también 23d que luego
; +-+-+se convierte a 17d para indicar el final del proceso+-+-+
340      AssemblyRV3SBRobot   ExecProg("TENC",21,23,0)
; +++++ Transport: Move carrier to processing station. +++++
350      Transport      to_StockSingle(2)
; +++++ Station StockSingle: Move pallet from conveyor to station buffer. +++++
360      StockSingle      MovFromTrans(82200, 0)
370      Transport      SetProductId("", %_4_CarrierID)
; +++++ Transport: Release carrier. +++++
380      Transport      RELEASE(2)
390      .CALC   %_4_CarrierID = 0
; *****
; END of second user defined structure part.
; *****
;
ENDSPLIT23      .NOOP
; ***** Parallel processes merged together. *****
MERGE3      .MERGE
;
; *****
;
; END of SplitMergeTwo structure.
; *****
;
;
; -----
; END of user defined process plan part.
; -----
400      .NOOP      END
; -----
; END of automatically generated process plan.
; -----

```

- Proceso de la macro tarea AsmDeskset1

```

; -----*****-----
; +++++*****+++++
; Process: HS_CLASS_Assembly_AsmDeskset1 modificado del original HS_CLASS_Assembly_AsmDeskset
; Class: Station->Assembly
; Modificado por Tania Martinez el 09 de diciembre de 2015
; Se debe llamar con .CALL HS_CLASS_Assembly_AsmDeskset1(Par.1, Par.2)
;   Par.1   part number to assemble
;   Par.2   order number to assemble
;
; Result value:
;   000000:1...n   product assembled successfully
;   else           process terminated with error
;

```

```

; Description:
;   assembles a deskset from parts already available in buffers/feeders inside the station
;-----
;+++ Es necesario indicar la estación en %HS.Device y la tarea en %HS.Task+++
10      .CALC %HS.Device="AssemblyRV3SB"
20      .CALC %HS.Task="AsmDeskset"
CLAIM   .CLAIM #Stat[%HS.Device].SemaProd
START   .CALC %res = "000000:0"
INIT     .CALC %Par.0
60      2    .CALC %partno = %Par.1
          .CALC %res = "010010:" & #Proj.Errors[1]          REL
70      .CALC %orderno = %Par.2
80      .CALC %robot = ..LocalHost.$STATION[%HS.Device].Robot
90      .CALC %vision = ..LocalHost.$STATION[%HS.Device].VisionData
VIS2     .CALC #Stat[%HS.Device].Vis.Task = "AsmDeskset(" & %Par.1 & ", " & %Par.2 & ")"
;-----
; read production data from database
;-----
QUERY1   .CLAIM #Proj.Database.Sema
130      .CALC %SQL = "SELECT  SubPart1, SubPart2, SubPart3, SubPart4, AddPart1, AddPart2, AddPart3,
AddPart4 FROM ProductionData WHERE (([PartNo] = " & %partno & ") AND ([Producer] = \" &
#Stat[%HS.Device].Class & "\") AND ([Macro] = \" & %HS.Task & "\"));"
140      Database      ExecSQL(%SQL)
150      Database      %retval = GetResultRow( "%PD")
160      .RELEASE      #Proj.Database.Sema
170      .CALC %retval
180      ""            .CALC %res = "040301:" & #Proj.Database.Errors[30]          REL
          .NOOP
GETNUM1   .CALC %PD.0
200      8            .NOOP
          .CALC %res = "040311:" & #Proj.Database.Errors[31]          REL
QUERY2   .CLAIM #Proj.Database.Sema
230      .CALC %SQL = "SELECT  OnPallet FROM Products WHERE [PartNo] = " & %PD.2 & " ;"

240      Database      ExecSQL( %SQL )
250      Database      %retval = GetResultRow( "%EmptyPallet")
260      .RELEASE      #Proj.Database.Sema
270      .CALC %retval
280      ""            .CALC %res = "040301:" & #Proj.Database.Errors[30]          REL
          .NOOP
GETNUM2   .CALC %EmptyPallet.0
300      1            .NOOP
          .CALC %res = "040311:" & #Proj.Database.Errors[31]          REL
;-----
; find pallet position of base plate (%PD.1 = part number of base plate (subpart 1) )
;-----
SUBPART1  .CALL %retval = Tools_Buffers_Find (%HS.Device, %PD.1, %orderno, 0)

340      .CALC FIELD(%retval, 0, ":")
350      0            .CALC %bp = FIELD(%retval, 1, ":")
          .CALC %res = "510980:" & #Stat[%HS.Device].Errors[98] & "\n" & FIELD(%retval, 1, ":")          REL

360      0            .CALL %retval = Tools_Buffers_Find (%HS.Device, %PD.1, 0, 0)
          .NOOP      SUBPART2
370      .CALC FIELD(%retval, 0, ":")

```

```

380    0      .CALC %bp = FIELD(%retval, 1, ":")
          .CALC %res = "510980:" & #Stat[%HS.Device].Errors[98] & "\n" & FIELD(%retval, 1, ":")    REL

390    0      .CALC %res = "510810:" & #Stat[%HS.Device].Errors[81]    REL
          .NOOP

; -----
; if PD.2 <> 0 AND PD.2 <> "" Then find pallet position of penholder (%PD.2 = part number of pen holder (subpart 2) )

; -----
SUBPART2      .CALC %PD.2
400    "",0    .NOOP      FEED1
          .CALL %retval = Tools_Buffers_Find (%HS.Device, %PD.2, %orderno, 0)
405          .CALC FIELD(%retval, 0, ":")
410    0      .CALC %ph = FIELD(%retval, 1, ":")
          .CALC %res = "510980:" & #Stat[%HS.Device].Errors[98] & "\n" & FIELD(%retval, 1, ":")    REL

415    0      .CALL %retval = Tools_Buffers_Find (%HS.Device, %PD.2, 0, 0)
          .NOOP      FEED1
420          .CALC FIELD(%retval, 0, ":")
425    0      .CALC %ph = FIELD(%retval, 1, ":")
          .CALC %res = "510980:" & #Stat[%HS.Device].Errors[98] & "\n" & FIELD(%retval, 1, ":")    REL

430    0      .CALC %res = "510810:" & #Stat[%HS.Device].Errors[81]    REL
          .NOOP

; -----
; find responsible feeder for Additional Part 1, if AddPart1 <> 0 and AddPart1 <> ""
; -----
FEED1      .CALC (%PD.5 == 0) OR (%PD.5 == "")
440    1      .CALC %feed1 = 0      ADD2
          .CALL %retval = Tools_Feeder_Find(%HS.Device, %PD.5)
450          .CALC FIELD(%retval, 0, ":")
460    0      .CALC %feed1 = FIELD(%retval, 1, ":")
          .CALC %res = "510970:" & #Stat[%HS.Device].Errors[97] & "\n" & FIELD(%retval, 1, ":")    REL

470    0      .CALC %res = "510820:" & #Stat[%HS.Device].Errors[82]    REL
          .NOOP

; -----
; find responsible feeder for Additional Part 2, if AddPart2 <> 0 and AddPart2 <> ""
; -----
ADD2      .CALC (%PD.6 == 0) OR (%PD.6 == "")
480    1      .CALC %feed2 = 0      STARTASM
          .CALL %retval = Tools_Feeder_Find(%HS.Device, %PD.6)
490          .CALC FIELD(%retval, 0, ":")
500    0      .CALC %feed2 = FIELD(%retval, 1, ":")
          .CALC %res = "510970:" & #Stat[%HS.Device].Errors[97] & "\n" & FIELD(%retval, 1, ":")    REL

510    0      .CALC %res = "510820:" & #Stat[%HS.Device].Errors[82]    REL
          .NOOP

; -----
; Assemble deskset
; -----
STARTASM      .NOOP

; -----
; Mueve la placa base del pale a la posición de ensamble utilizando TMBP
; -----
640          .TASK %retval = %robot, "ExecProg"("TMBP", %bp, 5, 0 )

```

```

660      .CALC FIELD(%retval, 1, ":")
670      0      .NOOP      ASMAD1
      .CALC %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]
; -----
; first error processing
; -----
REL      .CALC FIELD(%res, 0, ":")
ERRREACT 0      .NOOP      ASMAD1
      Dialog %react = RetryAbort(FIELD(%res, 0, ":"), %HS.Device, "AsmDeskset(" & %Par.1 & ", "
& %Par.2 & ")", FIELD(%res, 1, ":"))
ERRDO "RETRY" .NOOP      START
      .NOOP      RELSEMA
; -----
; Mueve higrómetro utilizando el programa TINS de los alimentadores a posición de prueba de vision y luego a la placa
; base -----
ASMAD1      .CALC %feed1
730      0      .NOOP      ASMAD2
      .TASK %retval = %vision, "GetData"("Available")
740      .CALC %part = FIELD(%retval, 1, ":")
750      999    .CALC %res = "511000:" & FIELD(%retval, 2, ":") MOVBP
      1      Dialog %react = RetryAbort(510050, %HS.Device, "AsmDeskset(" & %Par.1 & ", " & %Par.2 & ")",
#Dev[%robot].Errors[5])
      .TASK %retval = %robot, "ExecProg"("TINS", %feed1, 5, 0 )      ASMAD12
760      "RETRY" .NOOP      ASMAD1
      .CALC %res = "510050:" & #Dev[%robot].Errors[5]      MOVBP
ASMAD12      .CALC FIELD(%retval, 1, ":")
770      0      .NOOP      ASMAD13
      .CALC %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]
ASMAD1ERR1    Dialog %react = RetryAbort(FIELD(%res, 0, ":"), %HS.Device, "AsmDeskset(" & %Par.1 & ",
" & %Par.2 & ") - Instrument 1 Vision", FIELD(%res, 1, ":"))
ASMAD1REACT1  "RETRY"      .CALC %res = "000000:0"      ASMAD1
      .NOOP      MOVBP
ASMAD13      .TASK %retval = %vision, "GetData"("PN" & %PD.5)
780      .CALC %orientation = FIELD(%retval, 1, ":")
790      .TASK %retval = %robot, "ExecProg"("TINS", 5, 3, %orientation )
      999    .CALC %res = "510090:" & FIELD(%retval, 2, ":") MOVBP
800      .CALC FIELD(%retval, 1, ":")
810      0      .NOOP      ASMAD2
      .CALC %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]
ASMAD1ERR2    Dialog %react = RetryAbort(FIELD(%res, 0, ":"), %HS.Device, "AsmDeskset(" & %Par.1 & ",
" & %Par.2 & ") - Instrument 1", FIELD(%res, 1, ":"))
ASMAD1REACT2  "RETRY"      .CALC %res = "000000:0"      ASMAD1
      .NOOP      MOVBP
; -----
; Mueve termómetro utilizando el programa TINS de los alimentadores a posición de prueba de vision y luego a la placa
; base -----
ASMAD2      .CALC %feed2
830      0      .NOOP      ASMPH
      .TASK %retval = %vision, "GetData"("Available")
840      .CALC %part = FIELD(%retval, 1, ":")
850      999    .CALC %res = "511000:" & FIELD(%retval, 2, ":") MOVBP
      1      Dialog %react = RetryAbort(510050, %HS.Device, "AsmDeskset(" & %Par.1 & ", " & %Par.2 & ")",
#Dev[%robot].Errors[5])

```

```

.TASK %retval = %robot, "ExecProg"("TINS", %feed2, 5, 0)      ASMAD22
860    "RETRY"      .NOOP      ASMAD2
          .CALC %res = "510050:" & #Dev[%robot].Errors[5]      MOVBP
ASMAD22      .CALC FIELD(%retval, 1, ":")
870    0      .NOOP      ASMAD23
          .CALC %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]
ASMAD2ERR1      Dialog %react = RetryAbort(FIELD(%res, 0, ":"), %HS.Device, "AsmDeskset(" & %Par.1 & ",
" & %Par.2 & ") - Instrument 2 Vision", FIELD(%res, 1, ":"))
ASMAD2REACT1      "RETRY"      .CALC %res = "000000:0"      ASMAD2
          .NOOP      MOVBP
ASMAD23      .TASK %retval = %vision, "GetData"("PN" & %PD.6)
880      .CALC %orientation = FIELD(%retval, 1, ":")
890      .TASK %retval = %robot, "ExecProg"("TINS", 5, 4, %orientation)
999      .CALC %res = "510090:" & FIELD(%retval, 2, ":") MOVBP
900      .CALC FIELD(%retval, 1, ":")
910    0      .NOOP      ASMPH
          .CALC %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]
ASMAD2ERR2      Dialog %react = RetryAbort(FIELD(%res, 0, ":"), %HS.Device, "AsmDeskset(" & %Par.1 & ",
" & %Par.2 & ") - Instrument 2", FIELD(%res, 1, ":"))
ASMAD2REACT2      "RETRY"      .CALC %res = "000000:0"      ASMAD1
          .NOOP      MOVBP
; -----
; Ensambla el porta lapicero del pale a la placa base en la posicion de ensamble utilizando TMPH
; -----
ASMPH      .CALC %PD.2
930    "",0      .NOOP      MOVBP
          .TASK %retval = %robot, "ExecProg"("TMPH", %ph, 5, 0)
960      .CALC FIELD(%retval, 1, ":")
970    0      .NOOP      ASMPHOK
          .CALC %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]
ASMPHERR      Dialog %react = RetryAbort(FIELD(%res, 0, ":"), %HS.Device, "AsmDeskset(" & %Par.1 & ",
" & %Par.2 & ") - Penholder", FIELD(%res, 1, ":"))
ASMPHREACT      "RETRY"      .CALC %res = "000000:0"      ASMPH
          .NOOP      MOVBP
ASMPHOK      .CALC #Stat[%HS.Device].Buffers[%ph].PNo = %EmptyPallet.1
990      .CALC #Stat[%HS.Device].Buffers[%ph].ONo = 0
; -----
; Coloca el lapicero en el porta lapicero utilizando TASM
; -----
ASMPEN      .CALC (%PD.7 == 0) OR (%PD.7 == "")
1040    1      .NOOP      MOVBP
          .TASK %retval = %robot, "ExecProg"("TASM", 0, 0, 0)
1060      .CALC FIELD(%retval, 1, ":")
1070    0      .NOOP      MOVBP
          .CALC %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]
ASMPENERR      Dialog %react = RetryAbort(FIELD(%res, 0, ":"), %HS.Device, "AsmDeskset(" & %Par.1 & ",
" & %Par.2 & ") - Pen", FIELD(%res, 1, ":"))
ASMPENREACT      "RETRY"      .CALC %res = "000000:0"      ASMPEN
          .NOOP
; -----
; Mueve la placa base de la posición de ensamble al pale utilizando TMBP
; -----

```



```

MOVBP      .TASK  %retval = %robot, "ExecProg"("TMBP", 5, %bp, 0 )
1150      .CALC  FIELD(%retval, 1, ":")
1160    0    .NOOP      MOVBP
           .CALC  %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]
MOVBPERR    Dialog  %react = RetryAbort(FIELD(%res, 0, ":"), %HS.Device, "AsmDeskset(" & %Par.1 & ",
" & %Par.2 & ") - Baseplate", FIELD(%res, 1, ":"))
MOVBPREACT  "RETRY"  .CALC  %res = "000000:0"      MOVBP
           .NOOP      RELSEMA
MOVBPOK      .CALC  FIELD(%res, 0, ":")
1180    0    .CALC  #Stat[%HS.Device].Buffers[%bp].PNo = %partno
           .NOOP      REL2
1190      .CALC  #Stat[%HS.Device].Buffers[%bp].ONo = %orderno
1200      .CALC  %res = "000000:0"
; -----
; return result value
; -----
REL2      .CALC  FIELD(%res, 0, ":")
ERRREACT2  0      .NOOP
           Dialog  %react = Abort(FIELD(%res, 0, ":"), %HS.Device, "AsmDeskset(" & %Par.1 & ", " & %Par.2 &
")", FIELD(%res, 1, ":"))
RELSEMA    .RELEASE      #Stat[%HS.Device].SemaProd
QUIT      .CALC  #Stat[%HS.Device].Vis.Task = ""
2030      .CALC  %res  END

```

- Proceso de la macro tarea MovFromTrans1

```

; -----*****-----
;+++++*****+++++
; Process: HS_CLASS_Assembly_MovFromTrans1 modificado del original HS_CLASS_Assembly_MovFromTrans
; Class: Station->Assembly
; Creado por Tania Martinez el 08 de diciembre de 2015
; Se debe llamar con .CALL HS_CLASS_Assembly_MovFromTrans1(Par.1, Par.2)
;   Par.1   part number to store
;   Par.2   order number to store
;
; Result value:
;   000000:1...n   product moved into returned buffer position
;   else           process terminated with error
;
; Description:
;   searches for empty buffer position and moves product from conveyor to buffer position
; -----
; -----
; claim target buffer position
; ----Se debe indicar la estación en %HS.Device ----
10      .CALC  %HS.Device="AssemblyRV3SB"
CLAIM    .CLAIM #Stat[%HS.Device].SemaInOut
START    .CALC  %res = "000000:0"
40      .CALL  %retval = Tools_Buffers_Claim(%HS.Device, %Par.1, %Par.2, 0)
VIS1     .CALC  #Stat[%HS.Device].Vis.Task = "MovFromTrans(" & %Par.1 & ", " & %Par.2 & ")"
; -----
; check if empty buffer could be claimed for product
; -----
GETPOS    .CALC  FIELD(%retval, 0, ":")

```

```

120    0      .CALC %targetpos = FIELD(%retval, 1, ":")
          .CALC %res = "510980:" & #Stat[%HS.Device].Errors[98] & "\n" & FIELD(%retval, 1, ":")    REL

130    0      .CALC %res = "510800:" & #Stat[%HS.Device].Errors[80]    REL
          .CALC %robot = ..LocalHost.$STATION[%HS.Device].Robot
; -----
; Mueve el pale a la estación con TMP
; -----
MOVEIN          .TASK %retval = ..LocalHost.$STATION[%HS.Device].Robot, "ExecProg"("TMP",
15, %targetpos )
210          .CALC FIELD(%retval, 1, ":")
220    0      .CALC %res = "000000:" & %targetpos    REL
          .CALC %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]
230          .CALL Tools_Buffers_Release(%HS.Device, %targetpos)
; -----
; release semaphore for station
; -----
REL          .CALC FIELD(%res, 0, ":")
ERRREACT    0      .NOOP          RELSEMA
          Dialog %react = RetryAbortIgnore(FIELD(%res, 0, ":"), %HS.Device, "MovFromTrans(" & %Par.1 &
", " & %Par.2 & ")", FIELD(%res, 1, ":"))
ERRDO "RETRY"    .NOOP          START
          "ABORT"    .NOOP
          .CALC %res = "000000:0"
RELSEMA      .RELEASE    #Stat[%HS.Device].SemaInOut
VIS2        .CALC #Stat[%HS.Device].Vis.Task = ""
; -----
; return result value
; -----
QUIT        .CALC %res    END

```

- Proceso de la macro tarea MovToTrans1

```

; -----*****-----
;+++++*****+++++
; Process: HS_CLASS_Assembly_MovToTrans1 modificado del original HS_CLASS_Assembly_MovToTrans
; Class: Station->Assembly
; Modificado por Tania Martinez el 08 de diciembre de 2015
; Se debe llamar con .CALL HS_CLASS_Assembly_MovToTrans(Par.1, Par.2)
;   Par.1   part number to retrieve
;   Par.2   order number to retrieve
;
; Result value:
;   000000:1...n   product moved to carrier from buffer position
;   else           process terminated with error
;
; Description:
;   searches for product on buffer position and moves product from buffer position to conveyor
;
; -----
; -----
; claim target buffer position
; ----Se debe indicar la estación en %HS.Device ----
10          .CALC %HS.Device="AssemblyRV3SB"

```

```

CLAIM      .CLAIM #Stat[%HS.Device].SemaInOut
START      .CALC %res = "000000:0"
40         .CALL %retval = Tools_Buffers_Find(%HS.Device, %Par.1, %Par.2, 0)
VIS1       .CALC #Stat[%HS.Device].Vis.Task = "MovToTrans(" & %Par.1 & ", " & %Par.2 & ")"
; -----
; check if product could be found on buffer
; -----
GETPOS      .CALC FIELD(%retval, 0, ":")
120      0  .CALC %sourcepos = FIELD(%retval, 1, ":")
           .CALC %res = "510980:" & #Stat[%HS.Device].Errors[98] & "\n" & FIELD(%retval, 1, ":")    REL

130      0  .CALC %res = "510810:" & #Stat[%HS.Device].Errors[81]    REL
           .CALC %robot = ..LocalHost.$STATION[%HS.Device].Robot
; -----
; Retira el pale de la estación utilizando TMP
; -----
MOVEOUT     .TASK %retval = %robot, "ExecProg"("TMP", %sourcepos, 15 )
210         .CALC FIELD(%retval, 1, ":")
220      0  .CALC %res = "000000:" & %sourcepos
           .CALC %res = "510990:" & #Stat[%HS.Device].Errors[99] & "\n" &
#Dev[%robot].Errors[FIELD(%retval, 1, ":")]    REL
230         .CALL Tools_Buffers_Release(%HS.Device, %sourcepos)
; -----
; release semaphore for station
; -----
REL         .CALC FIELD(%res, 0, ":")
ERRREACT    0      .NOOP      RELSEMA
           Dialog %react = RetryAbortIgnore(FIELD(%res, 0, ":"), %HS.Device, "MovToTrans(" & %Par.1 & ", "
& %Par.2 & ")", FIELD(%res, 1, ":"))
ERRDO "RETRY" .NOOP      START
           "ABORT"      .NOOP
           .CALC %res = "000000:0"
RELSEMA     .RELEASE      #Stat[%HS.Device].SemaInOut
VIS2        .CALC #Stat[%HS.Device].Vis.Task = ""
; -----
; return result value
; -----
QUIT        .CALC %res    END

```

14.4 Tags, elementos y scripts utilizados para la elaboración de la HMI del SCADA en la modalidad 1

14.4.1 Tags

| Nombre | Tipo de Tag | Tipo de dato | Expresión | Comentario |
|----------------|------------------------|--------------|-----------|--|
| Message | OPC Tag | String | | Generada automáticamente por el TCP driver para leer datos por Ethernet |
| alarmas1 | Memory Tag | Integer | | Tag de memoria en que se almacena el número de código de la alarma generada |
| contadoralarma | Memory Tag/History Tag | Integer | | Tag de memoria en la que se almacena la cuenta total de las alarmas generadas, se coloca como tag de historia para almacenarla en la base de datos |
| contadorset | Memory Tag/History Tag | Integer | | Tag de memoria en la que se almacena la cuenta total de los set ensamblados, se coloca como tag de historia para almacenarla en la base de datos |
| contafaltai | Memory Tag/History Tag | Integer | | Tag de memoria en la que se almacena la cuenta de las alarmas generadas por falta de instrumentos, se coloca como tag de historia para almacenarla en la base de datos |
| contafaltao | Memory Tag/History Tag | Integer | | Tag de memoria en la que se almacena la cuenta de las alarmas generadas por otro tipo de faltas, se coloca como tag de historia para almacenarla en la base de datos |
| contafaltap | Memory Tag/History Tag | Integer | | Tag de memoria en la que se almacena la cuenta de las alarmas generadas por falta de piezas, se coloca como tag de historia para almacenarla en la base de datos |
| contafaltat | Memory Tag/History Tag | Integer | | Tag de memoria en la que se almacena la cuenta de las alarmas generadas por falta de transportes, se coloca como tag de historia para almacenarla en la base de datos |
| contafaltav | Memory Tag/History Tag | Integer | | Tag de memoria en la que se almacena la cuenta de las alarmas generadas por problemas de visión, se coloca como tag de historia para almacenarla en la base de datos |
| estado1 | Memory Tag | Integer | | Tag de memoria en que se almacena el estado en que se encuentra la celda: OFF (0), ON(1), Falla (2) o Indeterminado(otro) |

| | | | | |
|-------------|---------------|---------|-------------------------------|--|
| estadoalarm | Expresion Tag | Integer | substring({[~]Message},37,39) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado/alarma |
| posicion | Expresion Tag | Integer | substring({[~]Message},33,34) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente a la posición del robot |
| sencil1 | Expresion Tag | Integer | substring({[~]Message},26,27) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor del cilindro de sujeción 1 |
| sencil2 | Expresion Tag | Integer | substring({[~]Message},29,30) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor del cilindro de sujeción 2 |
| senhigr | Expresion Tag | Integer | substring({[~]Message},20,21) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor de higrómetros |
| senlap | Expresion Tag | Integer | substring({[~]Message},23,24) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor de lapiceros |
| senpale1 | Expresion Tag | Integer | substring({[~]Message},2,3) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor de palé 1 |
| senpale2 | Expresion Tag | Integer | substring({[~]Message},5,6) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor de palé 2 |
| senpale3 | Expresion Tag | Integer | substring({[~]Message},8,9) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor de palé 3 |
| senpale4 | Expresion Tag | Integer | substring({[~]Message},11,12) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor de palé 4 |
| senpbasm | Expresion Tag | Integer | substring({[~]Message},14,15) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor de placa base en la pos. De ensamble |
| senterm | Expresion Tag | Integer | substring({[~]Message},17,18) | Tag de expresión donde se extrae de la Tag OPC "Message" el código equivalente al estado del sensor de termómetros |

Tabla 14.1. Tags utilizadas en el SCADA en la modalidad 1.

14.4.2 Ventanas

Ventanas:

| Nombre | Tipo | Título | Tamaño | Localización | Comentario |
|----------------|-----------|--------------------------------|-----------------------|---------------|---|
| borrarcontalar | Emergente | Resetear contadores de alarmas | Ancho= 505, Alto= 209 | X=100, Y= 408 | Acceso a borrar la cuenta de las alarmas |
| borrarcontaset | Emergente | Resetear contador de sets | Ancho= 396, Alto= 186 | X=100, Y= 160 | Acceso a borrar la cuenta de los set ensamblados |
| detallefalla | Emergente | Detalle Falla | Ancho= 464, Alto= 240 | X=800, Y= 260 | Se muestra a detalle la alarma activada |
| estadistic | Principal | Estadísticas | Ancho= 800, Alto= 678 | X=0, Y=0 | Se muestra el número total de set ensamblados y de alarmas, además de la cuenta según su tipo |
| historiales | Principal | Historiales | Ancho= 800, Alto= 678 | X=0, Y=0 | Se muestra el historial de set ensamblados y alarmas activadas en los últimos 6 días |
| vent principal | Principal | Ventana Principal | Ancho= 800, Alto= 678 | X=0, Y=0 | Se muestra estado de sensores, posición del robot, programas en ejecución |

Tabla 14.2. Ventanas utilizadas en el SCADA en la modalidad 1.

14.4.2.1 Elementos y scripts de la ventana principal

Elementos más importantes de la ventana principal (“vent principal”)





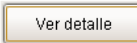



| Elemento | Nombre | Tipo | Property Binding | | | Script asociado | Comentario |
|---|-----------------------|-----------------------|------------------|--|--|-----------------|--|
|  | Circle 0 a Circle 19 | Circulo | Fill Paint | Value >=0 | color gris | No | Se utilizan círculos para cada uno de los 10 sensores de la estación que están asociados a su respectiva tag de expresión, los círculos se duplican para ser mostrados como lista y sobre la imagen de la estación. |
| | | | | Value >=1 | color anaranjado | | |
| Pos: <input type="text" value="2"/> | labelpos | Label | Text | Indicado por la tag de expresión "posición" | | Si | Recuadro donde se muestra el código equivalente a la posición del robot. |
| Estado: <input type="text" value="19"/> | labelestadalar | Label | Text | Indicado por la tag de expresión "estadoalarm" | | Si | Recuadro donde se muestra el código equivalente al estado/alarma de la estación. |
| Mover pale | labelcodprog | Label | | | | No | El texto que aparece lo establece el script de "labelestadalar" |
|  | Multi-State Indicator | Multi-State indicator | State | Value=0 | OFF (Fondo gris letras negras) | Si | El valor se lo da la tag de memoria "estado1" |
| | | | | Value=1 | ON (Fondo verde letras negras) | | |
| | | | | Value=2 | FALLA (Fondo amarillo letras negras) | | |
| | | | | Value=3 | INDETERMINADO (Fondo blanco letras verdes) | | |
|  | Sound Player1 | Sound Player | | | | No | Posee como propiedad de Play mode= On Trigger y como Loop Mode=Play Once, ejecuta el archivo de sonido colocado en la propiedad de Sound data= "WeaponHomming.wav", el disparo o trigger lo hace el script de "labelestadalar" |
|  | Image a Image 10 | Image | | | | No | El script de "labelpos" hace visible la imagen correspondiente a la posición |
|  | Buttonver | Button | | | | Si | El script de "Multi-State Indicator" hace visible o invisible este botón, al presionarlo muestra la ventana emergente para ver el detalle de la alarma |
|  | Buttonestad | Button | | | | Si | Al presionarlo se abre la ventana "estadistic" |
|  | Buttonhistor | Button | | | | Si | Al presionarlo se abre la ventana "historiales" |
|  | Buttoncerr | Button | | | | Si | Al presionarlo se cierran todas las ventanas de la aplicación |

Tabla 14.3. Elementos de la ventana principal (“vent principal”) del SCADA en la modalidad 1.

Scripts

Script de “labelpos”:

#Este script se ejecuta cada vez que cambia la propiedad “text” de esta etiqueta la cual está asociada a la tag de expresión #“posicion”, según el valor que tenga esta tag así se hace visible la imagen correspondiente que muestra al robot en esa #posición.

```
if event.propertyName=="text":
    event.source.parent.getComponent('Image').visible=0
    event.source.parent.getComponent('Image 1').visible=0
    event.source.parent.getComponent('Image 2').visible=0
    event.source.parent.getComponent('Image 3').visible=0
    event.source.parent.getComponent('Image 4').visible=0
    event.source.parent.getComponent('Image 5').visible=0
    event.source.parent.getComponent('Image 6').visible=0
    event.source.parent.getComponent('Image 7').visible=0
    event.source.parent.getComponent('Image 8').visible=0
    event.source.parent.getComponent('Image 9').visible=0
    event.source.parent.getComponent('Image 10').visible=0
    if event.source.text=="1":
        event.source.parent.getComponent('Image 1').visible=1
    elif event.source.text=="2":
        event.source.parent.getComponent('Image 2').visible=1
    elif event.source.text=="3":
        event.source.parent.getComponent('Image 3').visible=1
    elif event.source.text=="4":
        event.source.parent.getComponent('Image 4').visible=1
    elif event.source.text=="5":
        event.source.parent.getComponent('Image 5').visible=1
    elif event.source.text=="6":
        event.source.parent.getComponent('Image 6').visible=1
    elif event.source.text=="7":
        event.source.parent.getComponent('Image 7').visible=1
    elif event.source.text=="8":
        event.source.parent.getComponent('Image 8').visible=1
    elif event.source.text=="9":
        event.source.parent.getComponent('Image 9').visible=1
    elif event.source.text=="A":
        event.source.parent.getComponent('Image 10').visible=1
    else:
        event.source.parent.getComponent('Image').visible=1
```

Script de “labelestadalar”:

#Este script se ejecuta cada vez que cambia la propiedad “text” de esta etiqueta la cual está asociada a la tag de expresión #“estadoalarm”, según el valor recibido sabe si es un estado normal y que programa se está ejecutando o si es una alarma, #siguiendo los códigos con fondo blanco o gris de la tabla 9.3, así modifica el valor de las tags de memoria “estado1”, #“alarmas1”, las que llevan las cuentas del numero de sets, número de alarmas total y según el tipo, activación o no del #sonido de alarma y la modificación de la etiqueta “labelcodprog” que indica si se está ejecutando algún programa y el #nombre de este.

```
#
if event.propertyName == "text":
    event.source.parent.getComponent('Sound Player1').trigger=0
    valor1=event.source.text
```



```

if valor1=="19" or valor1=="20" or valor1=="21" or valor1=="22" or valor1=="23" :
    system.tag.write('superv/estado1', 1)
    system.tag.write('superv/alarmas1', "00")
    if valor1=="19":
        event.source.parent.getComponent('labelcodprog').text="Mover pale"
    elif valor1=="20":
        event.source.parent.getComponent('labelcodprog').text="Mover placa base"
    elif valor1=="21":
        event.source.parent.getComponent('labelcodprog').text="Mover medidor"
    elif valor1=="22":
        event.source.parent.getComponent('labelcodprog').text="Mover porta lapicero"
    elif valor1=="23":
        event.source.parent.getComponent('labelcodprog').text="Colocar lapicero"
elif valor1=="15":
    system.tag.write('superv/estado1',0)
    system.tag.write('superv/alarmas1', "00")
    event.source.parent.getComponent('labelcodprog').text="Ensamble completo"
    contensamb=system.tag.read("superv/contadorset").value
    contensamb=contensamb+1
    system.tag.write('superv/contadorset', contensamb)
elif valor1=="17":
    system.tag.write('superv/estado1',0)
    system.tag.write('superv/alarmas1', "00")
    event.source.parent.getComponent('labelcodprog').text="Proceso finalizado, la estacion queda en espera"
elif valor1=="18":
    system.tag.write('superv/alarmas1', "00")
    system.tag.write('superv/estado1',0)
    event.source.parent.getComponent('labelcodprog').text="Proceso iniciado, aun no entra en operacion la
estacion"
elif valor1=="01" or valor1=="04" or valor1=="11":
    contala=system.tag.read("superv/contadoralarma").value
    contala=contala+1
    system.tag.write('superv/contadoralarma', contala)
    system.tag.write('superv/alarmas1', valor1)
    event.source.parent.getComponent('Sound Player1').trigger=1
    event.source.parent.getComponent('labelcodprog').text="Falla, esperando ordenes del SCADA principal"
    system.tag.write('superv/estado1', 2)
    ctrans=system.tag.read("superv/contafaltat").value
    ctrans=ctrans+1
    system.tag.write('superv/contafaltat',ctrans)
elif valor1=="02" or valor1=="03" or valor1=="08":
    contala=system.tag.read("superv/contadoralarma").value
    contala=contala+1
    system.tag.write('superv/contadoralarma', contala)
    system.tag.write('superv/alarmas1', valor1)
    event.source.parent.getComponent('Sound Player1').trigger=1
    event.source.parent.getComponent('labelcodprog').text="Falla, esperando ordenes del SCADA principal"
    system.tag.write('superv/estado1', 2)
    cpieza=system.tag.read("superv/contafaltap").value
    cpieza=cpieza+1
    system.tag.write('superv/contafaltap',cpieza)
elif valor1=="06" or valor1=="07" or valor1=="09":
    contala=system.tag.read("superv/contadoralarma").value
    contala=contala+1
    system.tag.write('superv/contadoralarma', contala)
    system.tag.write('superv/alarmas1', valor1)

```

```

        event.source.parent.getComponent('Sound Player1').trigger=1
        event.source.parent.getComponent('labelcodprog').text="Falla, esperando ordenes del SCADA principal"
        system.tag.write('superv/estado1', 2)
        cinst=system.tag.read("superv/contafaltai").value
        cinst=cinst+1
        system.tag.write('superv/contafaltai',cinst)
    elif valor1=="12":
        contala=system.tag.read("superv/contadoralarma").value
        contala=contala+1
        system.tag.write('superv/contadoralarma', contala)
        system.tag.write('superv/alarmas1', valor1)
        event.source.parent.getComponent('Sound Player1').trigger=1
        event.source.parent.getComponent('labelcodprog').text="Falla, esperando ordenes del SCADA principal"
        system.tag.write('superv/estado1', 2)
        cvis=system.tag.read("superv/contafaltav").value
        cvis=cvis+1
        system.tag.write('superv/contafaltav',cvis)
    elif valor1=="05" or valor1=="10" or valor1=="14":
        contala=system.tag.read("superv/contadoralarma").value
        contala=contala+1
        system.tag.write('superv/contadoralarma', contala)
        system.tag.write('superv/alarmas1', valor1)
        event.source.parent.getComponent('Sound Player1').trigger=1
        event.source.parent.getComponent('labelcodprog').text="Falla, esperando ordenes del SCADA principal"
        system.tag.write('superv/estado1', 2)
        cotr=system.tag.read("superv/contafaltao").value
        cotr=cotr+1
        system.tag.write('superv/contafaltao',cotr)
    else:
        system.tag.write('superv/alarmas1', "13")
        system.tag.write('superv/estado1',3)
        event.source.parent.getComponent('labelcodprog').text="Ninguno el SCADA principal aun no se
comunica"

```

Script de “Multi-State Indicator”:

```

#Este script se ejecuta cuando cambia la propiedad “state” de este indicador la cual está asociada a la tag de memoria
#“estado1”, si el estado es falla o indeterminado hace visible el botón de “Ver detalle” (Butonver)
#
if event.propertyName== "state":
    if event.source.state==2 or event.source.state==3:
        event.source.parent.getComponent('Buttonver').visible=1
    else:
        event.source.parent.getComponent('Buttonver').visible=0

```

Script de “Buttonver”:

```

# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana donde se muestra el texto con la
#alarma generada.
#
window = system.nav.openWindow('detallefalla')

```

Script de “Buttonestad”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “estadistic”
#
window = system.nav.openWindow('estadistic')
```

Script de “Buttonhistor”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “historiales”
#
window = system.nav.openWindow('historiales')
```

Script de “Buttoncerr”:

```
#Este script se ejecuta cuando se presiona este botón, lo que hace es cerrar todas las ventanas que componen la #aplicación
#
system.nav.closeWindow('detallefalla')
system.nav.closeWindow('borrarcontaaalar')
system.nav.closeWindow('borrarcontaset')
system.nav.closeWindow('estadistic')
system.nav.closeWindow('historiales')
```

14.4.2.2 Elementos y scripts de la ventana de detalle de fallas

Elementos de ventana de detalle de fallas (“detallefalla”)


| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario |
|---|--------------|--------|------------------|---|-----------------|--|
|  | Image | Image | | | No | Imagen que se hace visible solamente si ha ocurrido una alarma, esto lo controla el script que está dentro de "Labelcod" |
| Código: 00 | Labelcod | Label | Text | Indicado por la tag de memoria "alarmas1" | Si | Aquí se muestra el código de alarma |
| La estacion aun no confirma que la | indicaalar | Label | | | No | Texto que indica el tipo de alarma ocurrido, lo gestiona el el script que está dentro de "Labelcod" |
| produccion haya iniciado | indicaalar 1 | Label | | | No | Texto que continúa indicando el tipo de alarma ocurrido, lo gestiona el el script que está dentro de "Labelcod" |
| Aceptar | Buttonok | Button | | | Si | Botón que al presionarlo cierra esta ventana |

Tabla 14.4. Elementos de la ventana de detalle de fallas (“detallefalla”) del SCADA en la modalidad 1.

Scripts

Script de “Labelcod”:

```
#Este script se ejecuta cada vez que cambia la propiedad “text” de esta etiqueta la cual está asociada a la tag de memoria
#“alarmas1”, si es 0 es que no hay alarma y así lo indica en las etiquetas “indicaalar” e “indicaalar1”, sino aparece la
#imagen “Image” y el texto explicando la alarma siguiendo los códigos con fondo blanco o gris de la tabla 9.3
#
if event.propertyName== "text":
    if event.source.text=="00":
```

```

event.source.parent.getComponent('labelindcod').visible=0
event.source.visible=0
event.source.parent.getComponent('Image').visible=0
event.source.parent.getComponent('indicaalar').text= "No hay alarma activada"
event.source.parent.getComponent('indicaalar 1').text= "en este momento"

else:
event.source.parent.getComponent('labelindcod').visible=1
event.source.visible=1
event.source.parent.getComponent('Image').visible=1
if event.source.text == "01":
    event.source.parent.getComponent('indicaalar').text="No hay pale que tomar en la"
    event.source.parent.getComponent('indicaalar 1').text="posicion indicada"
elif event.source.text == "02":
    event.source.parent.getComponent('indicaalar').text="No hay pale con placa base"
    event.source.parent.getComponent('indicaalar 1').text=""
elif event.source.text=="03":
    event.source.parent.getComponent('indicaalar').text="No hay placa base en la"
    event.source.parent.getComponent('indicaalar 1').text="posicion de ensamblado"
elif event.source.text=="04":
    event.source.parent.getComponent('indicaalar').text="No hay pale donde guardar el set"
    event.source.parent.getComponent('indicaalar 1').text=""
elif event.source.text=="05":
    event.source.parent.getComponent('indicaalar').text="Posicion de ensamble ocupada"
    event.source.parent.getComponent('indicaalar 1').text=""
elif event.source.text=="06":
    event.source.parent.getComponent('indicaalar').text="No hay higrometros"
    event.source.parent.getComponent('indicaalar 1').text=""
elif event.source.text=="07":
    event.source.parent.getComponent('indicaalar').text="No hay termometros"
    event.source.parent.getComponent('indicaalar 1').text=""
elif event.source.text=="08":
    event.source.parent.getComponent('indicaalar').text="No hay placa base"
    event.source.parent.getComponent('indicaalar 1').text="con porta lapicero"
elif event.source.text=="09":
    event.source.parent.getComponent('indicaalar').text="No hay lapiceros"
    event.source.parent.getComponent('indicaalar 1').text=""
elif event.source.text=="10":
    event.source.parent.getComponent('indicaalar').text="Esta ocupada la posicion"
    event.source.parent.getComponent('indicaalar 1').text="donde se va a colocar el pale"
elif event.source.text=="11":
    event.source.parent.getComponent('indicaalar').text="No se movio un pale valido"
    event.source.parent.getComponent('indicaalar 1').text=""
elif event.source.text=="12":
    event.source.parent.getComponent('indicaalar').text="La imagen del medidor no se capta bien"
    event.source.parent.getComponent('indicaalar 1').text="el angulo calculado esta fuera de rango"
elif event.source.text=="14":
    event.source.parent.getComponent('indicaalar').text="se enviaron ordenes no validas"
    event.source.parent.getComponent('indicaalar 1').text="desde el SCADA principal"
else:
    event.source.parent.getComponent('indicaalar').text="La estacion aun no confirma que la"
    event.source.parent.getComponent('indicaalar 1').text="produccion haya iniciado"

```

Script de “Buttonok”:

#Este script se ejecuta cuando se presiona este botón, lo que hace es cerrar esta ventana

```
#
system.nav.closeParentWindow(event)
```

14.4.2.3 Elementos y scripts de la ventana de estadísticas

Elementos más importantes de la ventana de estadísticas (“estadistic”)




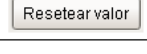





| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario |
|---|-----------------------|-----------------------|--|---|-----------------|--|
|  | LED Display | LED Display | Value | Indicado por la tag de memoria "contadorset" | No | Muestra el número total de set ensamblados |
|  | LED Display 1 | LED Display | Value | Indicado por la tag de memoria "contadoralarma" | No | Muestra el número total de alarmas activadas |
|  | Button | Button | | | Si | Al presionarlo se abre la ventana de "borrarcontaset". |
|  | Button 1 | Button | | | Si | Al presionarlo se abre la ventana de "borrarcontaalar". |
|  | Multi-State Indicator | Multi-State indicator | State | Value=0 OFF (Fondo gris letras negras) | Si | El valor se lo da la tag de memoria "estado1" |
| | | | Value=1 ON (Fondo verde letras negras) | | | |
| | | | Value=2 FALLA (Fondo amarillo letras negras) | | | |
| | | | Value=3 INDETERMINADO (Fondo blanco letras verdes) | | | |
|  | Bar Chart | Bar Chart | | | No | En la propiedad de DATA se especifica que los datos a mostrar serán las tags de memoria/historia que llevan la cuenta del tipo de alarma: "contafaltap", "contafaltai", "contafaltat", "contafaltav" y "contafaltao" |
|  | Buttonver | Button | | | Si | El script de "Multi-State Indicator" hace visible o invisible este botón, al presionarlo muestra la ventana emergente para ver el detalle de la alarma |
|  | Buttonhistor | Button | | | Si | Al presionarlo se abre la ventana "historiales" y se cierra esta |
|  | Buttonini | Button | | | Si | Al presionarlo se abre la ventana "vent principal" y se cierra esta |

Tabla 14.5. Elementos de la ventana de estadísticas (“estadistic”) del SCADA en la modalidad 1.

Scripts

Script de “Button”:

```
#Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “borrarcontaset”
#
window = system.nav.openWindow('borrarcontaset')
```

Script de “Button 1”:

```
#Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “borrarcontaalar”
```

```
#
window = system.nav.openWindow('borrarcontaalar')
```

Script de “Buttonhistor”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “historiales” y cerrar esta
#
window = system.nav.openWindow('historiales')
system.nav.closeParentWindow(event)
```

Script de “Buttonini”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “vent principal” y cerrar esta
#
window = system.nav.openWindow('vent principal')
system.nav.closeParentWindow(event)
```

Nota: El Script de “Multi-State Indicator” y “Buttonver” es idéntico al descrito en los elementos de la ventnana principal (“vent principal”).

14.4.2.4 Elementos y scripts de la ventana de reset de conteo de alarmas

Elementos de la ventana de reset de conteo de alarmas (“borrarcontalar”)







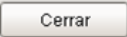
| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario |
|---|---------------------|----------------|------------------|---|-----------------|---|
|  | Password Field | Password Field | | | No | Cuadro de texto que permite al usuario ingresar la contraseña para poder borrar los contadores (los caracteres se muestran como asteriscos) |
|  | Label 1 | Label | Text | Indicado por la tag de memoria "contadoralarma" | No | Muestra el número total de alarmas activadas |
|  | Label 9 a Label 13 | Label | Text | Indicado por la tags de memoria de cada tipo de alarma "contafaltai", "contafaltat", "contafaltav", "contafaltap" y "contafaltao" | No | Muestra el número de alarmas activadas según su tipo |
|  | Button 1 a Button 6 | Button | | | Si | Se habilitan según el script de "Button" al ingresar la contraseña correcta, al presionarlo borra la cuenta respectiva y actualiza la total |
|  | Label 2 | Label | | | No | Se hace visible si la contraseña es incorrecta, esto lo gestiona el script de "Button" |
|  | Button | Button | | | Si | Al presionarlo verifica si es la contraseña correcta, si es así habilita los botones de reset, sino muestra un mensaje de contraseña incorrecta y los botones continúan inhabilitados |
|  | Button 7 | Button | | | Si | Cierra la ventana |

Tabla 14.6. Elementos de la ventana de reset de conteo de alarmas (“borrarcontalar”) del SCADA en la modalidad 1.

Scripts

Script de “Button”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es verificar si la contraseña ingresada en “Password
#Field” es correcta, sino lo es, hace visible el mensaje de “Label 2” e inhabilita los botones de reset de los contadores, si es
#correcta oculta el mensaje de “Label 2” y habilita los botones de reset
#
if event.source.parent.getComponent('Password Field').text=="123456":
    event.source.parent.getComponent('Label 2').visible=0
    event.source.parent.getComponent('Button 1').componentEnabled=1
    event.source.parent.getComponent('Button 2').componentEnabled=1
    event.source.parent.getComponent('Button 3').componentEnabled=1
    event.source.parent.getComponent('Button 4').componentEnabled=1
    event.source.parent.getComponent('Button 5').componentEnabled=1
    event.source.parent.getComponent('Button 6').componentEnabled=1
    event.source.parent.getComponent('Password Field').text=""
else:
    event.source.parent.getComponent('Label 2').visible=1
    event.source.parent.getComponent('Button 1').componentEnabled=0
    event.source.parent.getComponent('Button 2').componentEnabled=0
    event.source.parent.getComponent('Button 3').componentEnabled=0
    event.source.parent.getComponent('Button 4').componentEnabled=0
    event.source.parent.getComponent('Button 5').componentEnabled=0
    event.source.parent.getComponent('Button 6').componentEnabled=0
    event.source.parent.getComponent('Password Field').text=""
```

Script de “Button 1”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es resetear todas las tags de cuenta de alarmas y luego
#inhabilita los botones de reset para que sea necesario reingresar la contraseña si se desea hacer otro reseteo
#
system.tag.write('superv/contadoralarma', 0)
system.tag.write('superv/contafaltai', 0)
system.tag.write('superv/contafaltap', 0)
system.tag.write('superv/contafaltav', 0)
system.tag.write('superv/contafaltat', 0)
system.tag.write('superv/contafaltao', 0)
event.source.componentEnabled=0
event.source.parent.getComponent('Button 2').componentEnabled=0
event.source.parent.getComponent('Button 3').componentEnabled=0
event.source.parent.getComponent('Button 4').componentEnabled=0
event.source.parent.getComponent('Button 5').componentEnabled=0
event.source.parent.getComponent('Button 6').componentEnabled=0
```

Script de “Button 7”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es limpiar el cuadro de texto “Password Field”, hacer
#invisible el mensaje de “Label 2”, inhabilitar los botones de reset y cerrar la ventana
#
event.source.parent.getComponent('Password Field').text=""
event.source.parent.getComponent('Label 2').visible=0
event.source.parent.getComponent('Button 1').componentEnabled=0
event.source.parent.getComponent('Button 2').componentEnabled=0
```

```

event.source.parent.getComponent('Button 3').componentEnabled=0
event.source.parent.getComponent('Button 4').componentEnabled=0
event.source.parent.getComponent('Button 5').componentEnabled=0
event.source.parent.getComponent('Button 6').componentEnabled=0
system.nav.closeParentWindow(event)

```

Script de “Button 2”:

```

# Este script se ejecuta cuando se presiona este botón, lo que hace es resetear las tag de la cuenta de alarmas por falta de
instrumentos (contafaltai), resta el valor a la cuenta total de alarmas (“contadoralarma”) para que tenga el valor correcto y
#luego inhabilita los botones de reset para que sea necesario reingresar la contraseña si se desea hacer otro reseteo
#
mival=system.tag.read("superv/contafaltai").value
totval=system.tag.read("superv/contadoralarma").value
nueval=totval-mival
system.tag.write('superv/contadoralarma', nueval)
system.tag.write('superv/contafaltai', 0)
event.source.componentEnabled=0
event.source.parent.getComponent('Button 1').componentEnabled=0
event.source.parent.getComponent('Button 3').componentEnabled=0
event.source.parent.getComponent('Button 4').componentEnabled=0
event.source.parent.getComponent('Button 5').componentEnabled=0
event.source.parent.getComponent('Button 6').componentEnabled=0

```

Nota: Los Scripts de “Button 3” a “Button 6” son similares al de “Button 2” solo que resetean la tag correspondiente al tipo de alarma "contafaltat","contafaltav","contafaltap" y "contafaltao" respectivamente.

14.4.2.5 Elementos y scripts de la ventana de reset de conteo de sets

Elementos de la ventana de reset de conteo de sets (“borrarcontaset”)






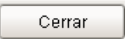
| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario |
|---|----------------|----------------|------------------|--|-----------------|---|
|  | Password Field | Password Field | | | No | Cuadro de texto que permite al usuario ingresar la contraseña para poder borrar el contador (los caracteres se muestran como asteriscos) |
|  | Label 1 | Label | Text | Indicado por la tag de memoria "contadorset" | No | Muestra el número total de set ensamblados |
|  | Button 1 | Button | | | Si | Se habilita según el script de "Button" al ingresar la contraseña correcta, al presionarlo borra la cuenta |
|  | Label 2 | Label | | | No | Se hace visible si la contraseña es incorrecta, esto lo gestiona el script de "Button" |
|  | Button | Button | | | Si | Al presionarlo verifica si es la contraseña correcta, si es así habilita el botón de reset, sino muestra un mensaje de contraseña incorrecta y el botón continúa inhabilitado |
|  | Button 7 | Button | | | Si | Cierra la ventana |

Figura 14.7. Elementos de la ventana de reset de conteo de sets (“borrarcontaset”) del SCADA en la modalidad 1.

Scripts

Script de “Button”:

Este script se ejecuta cuando se presiona este botón, lo que hace es verificar si la contraseña ingresada en “Password #Field” es correcta, sino lo es hace visible el mensaje de “Label 2” e inhabilita el botón de reset de los contadores, si es #correcta oculta el mensaje de “Label 2” y habilita el botón de reset

```
if event.source.parent.getComponent('Password Field').text=="123456":
    event.source.parent.getComponent('Button 1').componentEnabled=1
    event.source.parent.getComponent('Label 2').visible=0
    event.source.parent.getComponent('Password Field').text=""
else:
    event.source.parent.getComponent('Button 1').componentEnabled=0
    event.source.parent.getComponent('Label 2').visible=1
    event.source.parent.getComponent('Password Field').text=""
```

Script de “Button 1”:

Este script se ejecuta cuando se presiona este botón, lo que hace es resetear la cuenta de sets ensamblados y luego #inhabilitar el botón

```
#
value = 0
system.tag.write('superv/contadorset', value)
event.source.componentEnabled=0
```

Script de “Button 2”:

Este script se ejecuta cuando se presiona este botón, lo que hace es limpiar el cuadro de texto “Password Field”, hacer #invisible el mensaje de “Label 2”, inhabilita el botón de reset y cerrar la ventana

```
#
event.source.parent.getComponent('Password Field').text=""
event.source.parent.getComponent('Label 2').visible=0
event.source.parent.getComponent('Button 1').componentEnabled=1
system.nav.closeParentWindow(event)
```

14.4.2.6 Elementos y scripts de la ventana de historiales

Elementos más importantes de la ventana de historiales (“historiales”)


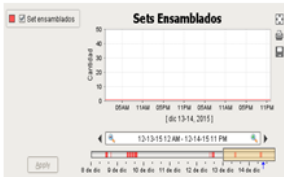
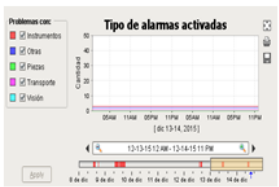



| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario |
|---|-----------------------|-----------------------|------------------|---------|--|--|
|  | Multi-State Indicator | Multi-State indicator | State | Value=0 | OFF (Fondo gris letras negras) | Si El valor se lo da la tag de memoria "estado1" |
| | | | | Value=1 | ON (Fondo verde letras negras) | |
| | | | | Value=2 | FALLA (Fondo amarillo letras negras) | |
| | | | | Value=3 | INDETERMINADO (Fondo blanco letras verdes) | |
|  | Easy Chart | Easy Chart | | | No | En este gráfico se muestra la tendencia del número de set ensamblados para los últimos 6 días. En la propiedad Tag Pens se coloca la tag de memoria/historia asociada que en este caso es "contadorset" y el valor de 6 días en la propiedad "Startup Range" y en "Startup Selection" 1 día. |
|  | Easy Chart 1 | Easy Chart | | | No | En este gráfico se muestra la tendencia del número de alarmas según su tipo generadas en los últimos 6 días. En la propiedad Tag Pens se coloca las tags de memoria/historia asociada que en este caso son "contafaltai", "contafaltao", "contafaltap", "contafaltat" y "contafaltav", el valor de 6 días en la propiedad "Startup Range" y en "Startup Selection" 1 día. |
|  | Buttonver | Button | | | Si | El script de "Multi-State Indicator" hace visible o invisible este botón, al presionarlo muestra la ventana emergente para ver el detalle de la alarma |
|  | Buttonestad 1 | Button | | | Si | Al presionarlo se abre la ventana "Estadísticas" y se cierra esta |
|  | Buttonini | Button | | | Si | Al presionarlo se abre la ventana "vent principal" y se cierra esta |

Tabla 14.8. Elementos de la ventana de historiales (“historiales”) del SCADA en la modalidad 1.

Scripts

Script de “Buttonestad 1”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “estadistic” y cerrar esta
#
window = system.nav.openWindow('estadistic')
system.nav.closeParentWindow(event)
```

Script de “Buttonini”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “vent principal” y cerrar esta
#
window = system.nav.openWindow('vent principal')
system.nav.closeParentWindow(event)
```

Nota: El Script de “Multi-State Indicator” y “Buttonver” es idéntico al descrito en los elementos de la ventana principal (“vent principal”).

14.5 Tags, elementos y scripts utilizados para la elaboración de la HMI del SCADA en la modalidad 2

14.5.1 Tags

| Nombre | Tipo de Tag | Tipo de Dato | Comentario |
|---------------|------------------------|--------------|--|
| Message | OPC Tag | String | Generada automáticamente por el TCP driver para leer datos por Ethernet |
| Writable | OPC Tag | String | Generada automáticamente por el TCP driver para escribir datos por Ethernet |
| alarmas | Memory Tag | String | Tag de memoria en que se almacena el número de código de la alarma generada |
| conalmtotales | Memory Tag | Integer | Tag de memoria en la que se almacena la cuenta total de las alarmas generadas |
| confaltaalm | Memory Tag/History Tag | Integer | Tag de memoria en la que se almacena la cuenta de las alarmas generadas por falta de almacenamiento |
| confaltainst | Memory Tag/History Tag | Integer | Tag de memoria en la que se almacena la cuenta de las alarmas generadas por falta de instrumentos |
| confaltamat | Memory Tag/History Tag | Integer | Tag de memoria en la que se almacena la cuenta de las alarmas generadas por falta de materiales |
| confaltaotra | Memory Tag/History Tag | Integer | Tag de memoria en la que se almacena la cuenta de las alarmas generadas por otras razones |
| consettotales | Memory Tag | Integer | Tag de memoria en la que se almacena la cuenta total de los set ensamblados |
| consmo | Memory Tag/History Tag | Integer | Tag de memoria en la que se almacena la cuenta total de los set ensamblados del tipo variante sin lapicero |
| constand | Memory Tag/History Tag | Integer | Tag de memoria en la que se almacena la cuenta total de los set ensamblados del tipo estándar con lapicero |
| constandsl | Memory Tag/History Tag | Integer | Tag de memoria en la que se almacena la cuenta total de los set ensamblados del tipo estándar sin lapicero |
| estado | Memory Tag | Integer | Tag de memoria en que se almacena el estado en que se encuentra la celda: OFF (0), ON(1), o Falla (2) |

| | | | |
|---------|------------|---------|---|
| tiposet | Memory Tag | Integer | Tag de memoria donde se almacena el tipo de set elegido: estándar con lapicero (1), estándar sin lapicero (2) y variante sin lapicero (3) |
|---------|------------|---------|---|

Tabla 14.9. Tags utilizadas en el SCADA en la modalidad 2.

14.5.2 Ventanas

Ventanas:

| Nombre | Tipo | Título | Tamaño | Localización | Comentario |
|-----------------|-----------|--|---------------------------|---------------|--|
| detallealarma | Emergente | Detalle Alarma | Ancho= 368, Alto= 199 | X=900, Y= 306 | Se muestra a detalle la alarma activada |
| estadisticas | Principal | Estadísticas | Ancho= 905, Alto= 600 | X=0, Y=0 | Se muestra el número total de set ensamblados y de alarmas activadas, además de la cuenta según su tipo |
| historicos | Principal | Históricos | Ancho= 1007, Alto= 674 | X=0, Y=0 | Se muestra el historial de set ensamblados y alarmas activadas en los últimos 6 días |
| mismapos | Emergente | Aviso | Ancho= 307, Alto= 168 | X=50, Y=415 | Indica que se ha elegido la misma posición para tomar la placa base y el porta lapicero, lo cual no es posible |
| principal | Principal | Controlador del Robot | Ancho= 905, Alto= 600 | X=0, Y=0 | Se muestra opciones de sets a ensamblar, posiciones de donde tomar materia prima y estado de la estación |
| resetcontalarma | Emergente | Ventana para resetear contador de alarmas | Ancho= 580, Alto= 239 | X=7, Y= 397 | Acceso a borrar la cuenta de las alarmas |
| resetcontset | Emergente | Ventana para resetear contador de producción | Ancho= 522, Alto= 203 | X=0, Y= 0 | Acceso a borrar la cuenta de los set ensamblados |

Tabla 14.10. Ventanas utilizadas en el SCADA en la modalidad 2.

14.5.2.1 Elementos y scripts de la ventana principal

Elementos más importantes de la ventana (“principal”)

| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario |
|----------|-----------------------|-----------------------|---|---|-----------------|--|
| | listasets | Dropdown List | | | Si | Se utiliza para seleccionar entre tres tipos de sets de escritorio a ensamblar, en la propiedad DATA se define el valor y la etiqueta de los 3 tipos de set |
| | listapospb | Dropdown List | | | Si | Se utiliza para seleccionar una de las 4 posiciones posibles de donde se tomará la placa base para el set |
| | listapospl | Dropdown List | | | Si | Se utiliza para seleccionar una de las 4 posiciones posibles de donde se tomará el porta lapicero para el set |
| | queenvio | Label | | | No | El texto que aparece lo establece el script del "Buttonensam" y es lo que se le está enviando al robot |
| | labelrecib | Label | Text | Indicado por la tag OPC "Message" | Si | Aquí se observa el dato que se recibe por el puerto |
| | Multi-State Indicator | Multi-State indicator | State | Value=0 OFF (Fondo gris letras negras) | Si | El valor se lo da la tag de memoria "estado" |
| | | | Value=1 ON (Fondo verde letras negras) | | | |
| | | | Value=2 FALLA (Fondo amarillo letras negras) | | | |
| | Sound Player | Sound Player | | | No | Posee como propiedad de Play mode= On Trigger y como Loop Mode=Play Once, ejecuta el archivo de sonido colocado en la propiedad de Sound data= "WeaponHomming.wav", el disparo o trigger lo hace el script de "labelrecib" |
| | Image a Image 2 | Image | | | No | Imágenes que se hacen visibles según el tipo de set que se elija, su aparición la gestiona el script de "listasets" |
| | LabelPL 1 a LabelPL 4 | Label | | | No | Etiqueta que aparece sobre la imagen estática de la estación, para que el usuario sepa cual es la posición elegida para tomar el porta lapicero, su aparición la gestiona el script de "listapospl" |
| | LabelPB 1 a LabelPB 4 | Label | | | No | Etiqueta que aparece sobre la imagen estática de la estación, para que el usuario sepa cual es la posición elegida para tomar la placa base, su aparición la gestiona el script de "listapospb" |
| | verdetalle | Button | | | Si | El script de "Multi-State Indicator" hace visible o invisible este botón, al presionarlo muestra la ventana emergente para ver el detalle de la alarma |
| | estadisticas | Button | | | Si | Al presionarlo se abre la ventana "estadisticas" |
| | historicos | Button | | | Si | Al presionarlo se abre la ventana "historicos" |
| | Buttonensam | Button | | | Si | Al presionarlo se envía un código al robot que contiene el tipo de set a ensamblar y las posiciones de donde tomar la placa base y el porta lapicero |
| | Buttonreint | Button | | | Si | Al presionarlo envía la orden al robot de reintentar la tarea donde se generó una alarma |
| | Buttonfinalizar | Button | | | Si | Al presionarlo envía la orden de finalizar el programa en el robot y aparece el botón "Buttonsalirprog" para poder cerrar la aplicación |
| | Buttonsalirprog | Button | | | Si | Al presionarlo se cierran todas las ventanas de la aplicación |

Tabla 14.11. Elementos más importantes de la ventana principal del SCADA en la modalidad 2.

Scripts

Script de “listasets”:

#Este script se ejecuta cada vez que se selecciona una opción de la lista, dependiendo de la elegida así muestra la imagen del #set correspondiente y oculta las demás

```
if event.source.selectedStringValue=="1":
    event.source.parent.getComponent('Image').visible= 1
    event.source.parent.getComponent('Image 1').visible= 0
    event.source.parent.getComponent('Image 2').visible= 0
elif event.source.selectedStringValue=="2":
    event.source.parent.getComponent('Image').visible= 0
    event.source.parent.getComponent('Image 1').visible= 1
    event.source.parent.getComponent('Image 2').visible= 0
elif event.source.selectedStringValue=="3":
    event.source.parent.getComponent('Image').visible= 0
    event.source.parent.getComponent('Image 1').visible= 0
    event.source.parent.getComponent('Image 2').visible= 1
```

Script de “listapospb”:

#Este script se ejecuta cada vez que se selecciona una opción de la lista, dependiendo de la elegida así muestra la etiqueta #de placa base sobre la posición correspondiente en la imagen estática de la estación y se ocultan las demás.

```
#
if event.propertyName== "selectedLabel":
    if event.source.selectedLabel=="1":
        event.source.parent.getComponent('LabelPB 1').visible=1
        event.source.parent.getComponent('LabelPB 2').visible=0
        event.source.parent.getComponent('LabelPB 3').visible=0
        event.source.parent.getComponent('LabelPB 4').visible=0
    elif event.source.selectedLabel=="2":
        event.source.parent.getComponent('LabelPB 1').visible=0
        event.source.parent.getComponent('LabelPB 2').visible=1
        event.source.parent.getComponent('LabelPB 3').visible=0
        event.source.parent.getComponent('LabelPB 4').visible=0
    elif event.source.selectedLabel=="3":
        event.source.parent.getComponent('LabelPB 1').visible=0
        event.source.parent.getComponent('LabelPB 2').visible=0
        event.source.parent.getComponent('LabelPB 3').visible=1
        event.source.parent.getComponent('LabelPB 4').visible=0
    elif event.source.selectedLabel=="4":
        event.source.parent.getComponent('LabelPB 1').visible=0
        event.source.parent.getComponent('LabelPB 2').visible=0
        event.source.parent.getComponent('LabelPB 3').visible=0
        event.source.parent.getComponent('LabelPB 4').visible=1
    else:
        event.source.parent.getComponent('LabelPB 1').visible=0
        event.source.parent.getComponent('LabelPB 2').visible=0
        event.source.parent.getComponent('LabelPB 3').visible=0
        event.source.parent.getComponent('LabelPB 4').visible=0
```

Script de “listapospl”:

#Este script se ejecuta cada vez que se selecciona una opción de la lista, dependiendo de la elegida así muestra la etiqueta #de porta lapicero sobre la posición correspondiente en la imagen estática de la estación y se ocultan las demás.

```
#
if event.propertyName== "selectedLabel":
    if event.source.selectedLabel=="1":
        event.source.parent.getComponent('LabelPL 1').visible=1
        event.source.parent.getComponent('LabelPL 2').visible=0
        event.source.parent.getComponent('LabelPL 3').visible=0
        event.source.parent.getComponent('LabelPL 4').visible=0
    elif event.source.selectedLabel=="2":
        event.source.parent.getComponent('LabelPL 1').visible=0
        event.source.parent.getComponent('LabelPL 2').visible=1
        event.source.parent.getComponent('LabelPL 3').visible=0
        event.source.parent.getComponent('LabelPL 4').visible=0
    elif event.source.selectedLabel=="3":
        event.source.parent.getComponent('LabelPL 1').visible=0
        event.source.parent.getComponent('LabelPL 2').visible=0
        event.source.parent.getComponent('LabelPL 3').visible=1
        event.source.parent.getComponent('LabelPL 4').visible=0
    elif event.source.selectedLabel=="4":
        event.source.parent.getComponent('LabelPL 1').visible=0
        event.source.parent.getComponent('LabelPL 2').visible=0
        event.source.parent.getComponent('LabelPL 3').visible=0
        event.source.parent.getComponent('LabelPL 4').visible=1
    else:
        event.source.parent.getComponent('LabelPL 1').visible=0
        event.source.parent.getComponent('LabelPL 2').visible=0
        event.source.parent.getComponent('LabelPL 3').visible=0
        event.source.parent.getComponent('LabelPL 4').visible=0
```

Script de “labelrecib”:

#Este script se ejecuta cada vez que cambia la propiedad “text” de esta etiqueta la cual está asociada a la tag OPC #“Message”, según el valor recibido sabe si es un estado normal o una alarma siguiendo los códigos con fondo blanco o #celeste de la tabla 9.3, modificando así las tags de memoria “estado”, “alarmas”, las que llevan las cuentas del numero de #alarmas y sets totales y según tipo, activación o no del sonido de alarma, además de gestionar que botón debe mostrarse si #“Ensamblar”, “Reintentar” o “Cerrar programa”

```
#
if event.propertyName== "text":
    event.source.parent.getComponent('Sound Player').trigger=0
    if event.source.text=="15":
        system.tag.write('alarmas', "00")
        event.source.parent.getComponent('Buttonensam').componentEnabled=1
        event.source.parent.getComponent('Buttonreint').visible=0
        event.source.parent.getComponent('Buttonfinalizar').componentEnabled=1
        system.tag.write('estado', 0)
        numset = system.tag.read("consettotales").value
        numset = numset+1
        system.tag.write('consettotales', numset)
        vartiposet = system.tag.read("tiposet").value
        if vartiposet==1:
            numstand = system.tag.read("constand").value
            numstand = numstand+1
```

```

        system.tag.write('constand', numstand)
    elif vartiposet==2:
        numstandsl = system.tag.read("constandsl").value
        numstandsl = numstandsl+1
        system.tag.write('constandsl', numstandsl)
    elif vartiposet==3:
        nummod = system.tag.read("consmod").value
        nummod = nummod+1
        system.tag.write('consmod', nummod)
elif event.source.text=="16":
    system.tag.write('alarmas', "00")
    system.tag.write('estado', 1)
    event.source.parent.getComponent('Sound Player').trigger=0
#-1 es texto vacío por que la tag no tiene nada, ocurre cuando recién se enciende el robot
elif event.source.text=="17" or event.source.text=="-1":
    system.tag.write('alarmas', "00")
    system.tag.write('estado', 0)
    event.source.parent.getComponent('Sound Player').trigger=0
    event.source.parent.getComponent('Buttonreint').visible=0
    event.source.parent.getComponent('Buttonensam').visible=0
    event.source.parent.getComponent('Buttonfinalizar').visible=0
    event.source.parent.getComponent('Buttonsalirprog').visible=1
#Recien se enciende el robot, la primera vez que envía algo se come el primer carácter por eso se prueba si es 18 u 8 para
#saber si ha iniciado el programa, luego esto ya no ocurre.
elif event.source.text== "18" or event.source.text=="8":
    system.tag.write('estado', 0)
    system.tag.write('alarmas', "00")
    event.source.parent.getComponent('Sound Player').trigger=0
    event.source.parent.getComponent('Buttonreint').visible=0
    event.source.parent.getComponent('Buttonensam').visible=1
    event.source.parent.getComponent('Buttonfinalizar').visible=1
    event.source.parent.getComponent('Buttonsalirprog').visible=0
else:
    system.tag.write('alarmas', event.source.text)
    system.tag.write('estado', 2)
    event.source.parent.getComponent('Sound Player').trigger=1
    event.source.parent.getComponent('Buttonreint').componentEnabled=1
    event.source.parent.getComponent('Buttonreint').visible=1
    event.source.parent.getComponent('Buttonensam').componentEnabled=1
    event.source.parent.getComponent('Buttonfinalizar').componentEnabled=1
    numalmtotales = system.tag.read("conalmtotales").value
    numalmtotales= numalmtotales+1
    system.tag.write('conalmtotales', numalmtotales)
    if event.source.text== "02" or event.source.text== "03" or event.source.text== "08":
        numfaltamat = system.tag.read("confaltamat").value
        numfaltamat= numfaltamat+1
        system.tag.write('confaltamat', numfaltamat)
    elif event.source.text== "04":
        numfaltaalm = system.tag.read("confaltaalm").value
        numfaltaalm= numfaltaalm+1
        system.tag.write('confaltaalm', numfaltaalm)
    elif event.source.text== "05":
        numotra = system.tag.read("confaltaotra").value
        numotra= numotra+1
        system.tag.write('confaltaotra', numotra)
    elif event.source.text== "06" or event.source.text== "07" or event.source.text== "09":

```



```

        numfaltainst = system.tag.read("confaltainst").value
        numfaltainst= numfaltainst+1
        system.tag.write('confaltainst', numfaltainst)
    elif event.source.text== "08":
        numfaltainst = system.tag.read("confaltainst").value
        numfaltainst= numfaltainst+1
        system.tag.write('confaltainst', numfaltainst)
    else:
        numotra = system.tag.read("confaltaotra").value
        numotra= numotra+1
        system.tag.write('confaltaotra', numotra)
        event.source.parent.getComponent('Buttonreint').visible=0
        event.source.parent.getComponent('Buttonensam').visible=0
        event.source.parent.getComponent('Buttonfinalizar').visible=0
        event.source.parent.getComponent('Buttonsalirprog').visible=1

```

Script de “Multi-State Indicator”:

#Este script se ejecuta cuando cambia la propiedad “state” de este indicador la cual está asociada a la tag de memoria #“estado”, si el estado es falla hace visible el botón de “Ver detalle”, sino no se muestra.

```

#
if event.propertyName== "state":
    if event.source.state==0 or event.source.state==1:
        event.source.parent.getComponent('verdetalle').visible=0
    elif event.source.state==2:
        event.source.parent.getComponent('verdetalle').visible=1

```

Script de “verdetalle”:

Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana donde se muestra el texto con la #alarma generada.

```

#
window = system.nav.openWindow('detallealarma')

```

Script de “estadísticas”:

Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “estadísticas”

```

#
window = system.nav.openWindow('estadisticas')
system.nav.centerWindow(window)

```

Script de “historicos”:

Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “historicos”

```

#
window = system.nav.openWindow('historicos')
system.nav.centerWindow(window)

```

Script de “Buttonensam”:

Este script se ejecuta cuando se presiona este botón, lo primero que hace es comparar si se ha elegido la misma posición #para tomar la placa base como el porta lapicero, si es así abre la ventana emergente “mismapos” para alertar al usuario, si #las posiciones son diferentes entonces crea un código compuesto por el valor del tipo de set +posición de placa base+

```

#posición de porta lapicero y lo envía al robot a través de la tag OPC "Writable", el valor enviado también lo coloca en la
#etiqueta "queenvio" para que el usuario pueda visualizarlo en pantalla
#
if event.source.parent.getComponent('listapospb').selectedStringValue ==
event.source.parent.getComponent('listapospl').selectedStringValue:
    window = system.nav.openWindow('mismapos')
    system.nav.centerWindow(window)
else:
    event.source.parent.getComponent('queenvio').text=
event.source.parent.getComponent('listasets').selectedStringValue+event.source.parent.getComponent('listapospb').selected
StringValue+event.source.parent.getComponent('listapospl').selectedStringValue
    system.tag.write('tiposet',event.source.parent.getComponent('listasets').selectedStringValue)

system.tag.write('Writable',event.source.parent.getComponent('listasets').selectedStringValue+event.source.parent.getComp
onent('listapospb').selectedStringValue+event.source.parent.getComponent('listapospl').selectedStringValue)
    system.nav.closeWindow('mismapos')
    event.source.componentEnabled=0
    event.source.parent.getComponent('Buttonfinalizar').componentEnabled=0

```

Script de "Buttonreint":

```

# Este script se ejecuta cuando se presiona este botón, lo que hace es enviar el código "RE" al robot a través de la tag OPC
#"Writable" e inhabilitar el botón junto con el de finalizar.
#
valor="RE"
system.tag.write('Writable',valor)
event.source.componentEnabled=0
event.source.parent.getComponent('Buttonfinalizar').componentEnabled=0

```

Script de "Buttonfinalizar":

```

# Este script se ejecuta cuando se presiona este botón, lo que hace es enviar el código "400" al robot a través de la tag OPC
#"Writable" con este código el robot finaliza su programa
#
system.tag.write('Writable',"400")

```

Script de "Buttonsalirprog":

```

#Este script se ejecuta cuando se presiona este botón, lo que hace es cerrar todas las ventanas que componen la aplicación
#
system.nav.closeWindow('mismapos')
system.nav.closeWindow('detallealarma')
system.nav.closeWindow('resetcontalarma')
system.nav.closeWindow('resetcontset')
system.nav.closeParentWindow(event)

```

14.5.2.2 Elementos y scripts de la ventana de aviso

Elementos de la ventana de aviso (“mismapos”)


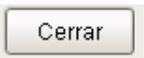
| Elemento | Nombre | Tipo | Property Binding | Script asociado | Comentario |
|---|--------|--------|------------------|-----------------|------------------------------------|
|  | Image | Image | | No | Señal de advertencia |
| La posición de la placa base y el porta lapicero no puede ser la misma | Label | Label | | No | Texto para explicar la advertencia |
|  | Button | Button | | Si | Cierra esta ventana |

Tabla 14.12. Elementos de la ventana de aviso (“mismapos”) del SCADA en la modalidad 2.

Scripts

Script de “Button”:

#Este script se ejecuta cuando se presiona este botón, lo que hace es cerrar la ventana “mismapos”

system.nav.closeParentWindow(event)

14.5.2.3 Elementos y scripts de la ventana de detalle de alarma

Elementos de la ventana de detalle de alarma (“detallealarma”)


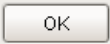
| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario |
|---|--------------|--------|------------------|--|-----------------|---|
|  | Image | Image | | | No | Imagen que se hace visible solamente si ha ocurrido una alarma, esto lo controla el script que está dentro de "codalarma" |
| Error:00 | codalarma | Label | Text | Indicado por la tag de memoria "alarmas" | Si | Aquí se muestra el código de alarma |
| No hay alarma activada | textoalarma | Label | | | No | Texto que indica el tipo de alarma ocurrido, lo gestiona el script que está dentro de "codalarma" |
| en este momento | textoalarma1 | Label | | | No | Texto que continúa indicando el tipo de alarma ocurrido, lo gestiona el script que está dentro de "codalarma" |
|  | Button 1 | Button | | | Si | Botón que al presionarlo cierra esta ventana |

Tabla 14.13. Elementos de la ventana de detalle de alarma (“detallealarma”) del SCADA en la modalidad 2.

Scripts

Script de “codalarma”:

#Este script se ejecuta cada vez que cambia la propiedad “text” de esta etiqueta la cual está asociada a la tag de memoria #“alarmas”, si es 0 es que no hay alarma y así lo indica en las etiquetas “textoalarma” y “textoalarma 1”, sino aparece la #imagen “Image” y el texto explicando la alarma siguiendo los códigos con fondo blanco o celeste de la tabla 9.3

```
#
if event.propertyName== "text":
    if event.source.text=="00":
        event.source.parent.getComponent('indcod').visible=0
        event.source.visible=0
        event.source.parent.getComponent('Image').visible=0
        event.source.parent.getComponent('textoalarma').text= "No hay alarma activada"
        event.source.parent.getComponent('textoalarma 1').text= "en este momento"
    else:
        event.source.parent.getComponent('indcod').visible=1
        event.source.visible=1
        event.source.parent.getComponent('Image').visible=1
        if event.source.text == "02":
            event.source.parent.getComponent('textoalarma').text="No hay pale con placa base"
            event.source.parent.getComponent('textoalarma 1').text=""
        elif event.source.text=="03":
            event.source.parent.getComponent('textoalarma').text="No hay placa base"
            event.source.parent.getComponent('textoalarma 1').text="el pale estaba vacio"
        elif event.source.text=="04":
            event.source.parent.getComponent('textoalarma').text="No hay pale donde guardar el set"
            event.source.parent.getComponent('textoalarma 1').text=""
        elif event.source.text=="05":
            event.source.parent.getComponent('textoalarma').text="Posicion de ensamble ocupada"
            event.source.parent.getComponent('textoalarma 1').text=""
        elif event.source.text=="06":
            event.source.parent.getComponent('textoalarma').text="No hay higrometros"
            event.source.parent.getComponent('textoalarma 1').text=""
        elif event.source.text=="07":
            event.source.parent.getComponent('textoalarma').text="No hay termometros"
            event.source.parent.getComponent('textoalarma 1').text=""
        elif event.source.text=="08":
            event.source.parent.getComponent('textoalarma').text="No hay pale con porta lapicero"
            event.source.parent.getComponent('textoalarma 1').text=""
        elif event.source.text=="09":
            event.source.parent.getComponent('textoalarma').text="No hay lapiceros"
            event.source.parent.getComponent('textoalarma 1').text=""
        elif event.source.text=="13":
            event.source.parent.getComponent('textoalarma').text="El robot recibio ordenes no validas"
            event.source.parent.getComponent('textoalarma 1').text="Finalizo el programa"
        else:
            event.source.parent.getComponent('textoalarma').text="Falla de comunicacion, no"
            event.source.parent.getComponent('textoalarma 1').text="se recibieron datos validos del robot"
```

Script de “Button 1”:

#Este script se ejecuta cuando se presiona este botón, lo que hace es cerrar esta ventana

```
#
system.nav.closeParentWindow(event)
```

14.5.2.4 Elementos y scripts de la ventana de estadísticas

Elementos más importantes de la ventana de estadísticas (“estadísticas”)




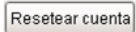


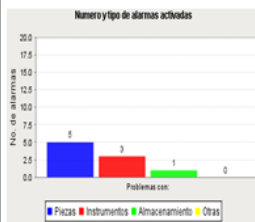



| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario | |
|---|-----------------------|-----------------------|------------------|---|--------------------------------------|--|--|
|  | LED Display | LED Display | Value | Indicado por la tag de memoria "consettotales" | No | Muestra el número total de set ensamblados | |
|  | LED Display 1 | LED Display | Value | Indicado por la tag de memoria "conalmntotales" | No | Muestra el número total de alarmas activadas | |
|  | resetset | Button | | | Si | Al presionarlo se abre la ventana de "resetcontset". | |
|  | resetalarm | Button | | | Si | Al presionarlo se abre la ventana de "resetcontalarma". | |
|  | Multi-State Indicator | Multi-State indicator | State | Value=0 | OFF (Fondo gris letras negras) | Si | El valor se lo da la tag de memoria "estado" |
| | | | | Value=1 | ON (Fondo verde letras negras) | | |
| | | | | Value=2 | FALLA (Fondo amarillo letras negras) | | |
|  | Bar Chart 1 | Bar Chart | | | No | En la propiedad de DATA se especifica que los datos a mostrar serán las tags de memoria/historia que llevan la cuenta del tipo de sets ensamblados: "constand", "constandsi" y "consmod" | |
|  | Bar Chart | Bar Chart | | | No | En la propiedad de DATA se especifica que los datos a mostrar serán las tags de memoria/historia que llevan la cuenta del tipo de alarmas generadas: "confaltamat", "confaltainst", "confaltaalm" y "confaltaotra" | |
|  | verdetalle | Button | | | Si | El script de "Multi-State Indicator" hace visible o invisible este botón, al presionarlo muestra la ventana emergente para ver el detalle de la alarma | |
|  | historicos | Button | | | Si | Al presionarlo se abre la ventana "historicos" y se cierra esta | |
|  | volver | Button | | | Si | Al presionarlo se abre la ventana "principal" y se cierra esta | |

Tabla 14.14. Elementos de la ventana de estadísticas (“estadísticas”) del SCADA en la modalidad 2.

Scripts

Script de “resetset”:

```
#Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “resetcontset”
#
window = system.nav.openWindow('resetcontset')
```

Script de “resetalarm”:

```
#Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “resetcontalarma”
```

```
#
window = system.nav.openWindow('resetcontalarma')
```

Script de “historicos”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “historicos” y cerrar esta
#
window = system.nav.openWindow('historicos')
system.nav.closeParentWindow(event)
```

Script de “volver”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “principal” y cerrar esta
#
window = system.nav.openWindow('principal')
system.nav.closeParentWindow(event)
```

Nota: El Script de “Multi-State Indicator” y “verdetalle” es idéntico al descrito en los elementos de la ventana principal (“principal”).

14.5.2.5 Elementos y scripts de la ventana de reset de conteo de alarmas

Elementos de la ventana de reset de conteo de alarmas (“resetcontalarma”)







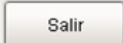
| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario |
|--|--|----------------|------------------|---|-----------------|---|
|  | Password Field | Password Field | | | No | Cuadro de texto que permite al usuario ingresar la contraseña para poder borrar los contadores (los caracteres se muestran como asteriscos) |
| Borrar todos las alarmas:  | labelalar | Label | Text | Indicado por la tag de memoria "conalmtotales" | No | Muestra el número total de alarmas activadas |
| Borrar solo alarmas por falta de piezas:  | labelpiez labelins labelalm labelotr | Label | Text | Indicado por la tags de memoria de cada tipo de alarma "confaltai", "confaltamat", "confaltainst", "confaltaalm" y "confaltaotra" | No | Muestra el número de alarmas activadas según su tipo |
|  | Buttonralar Buttonrmat Buttonrins Buttonralm Buttonrot | Button | | | Si | Se habilitan según el script de "Buttonaceptar" al ingresar la contraseña correcta, al presionarlo borra la cuenta respectiva y actualiza la total |
|  | msgincorrecta | Label | | | No | Se hace visible si la contraseña es incorrecta, esto lo gestiona el script de "Buttonaceptar" |
|  | Buttonaceptar | Button | | | Si | Al presionarlo verifica si es la contraseña correcta, si es así habilita los botones de reset, sino muestra un mensaje de contraseña incorrecta y los botones continúan inhabilitados |
|  | Buttonsalir | Button | | | Si | Cierra la ventana |

Tabla 14.15. Elementos de la ventana de reset de conteo de alarmas (“resetcontalarma”) del SCADA en la modalidad 2.

Scripts

Script de “Buttonaceptar”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es verificar si la contraseña ingresada en “Password
#Field” es correcta, sino lo es hace visible el mensaje de “msgincorrecta” e inhabilita los botones de reset de los contadores,
#si es correcta oculta el mensaje de “msgincorrecta” y habilita los botones de reset
#
if event.source.parent.getComponent('Password Field').text=="123456":
    event.source.parent.getComponent('Buttonralar').componentEnabled=1
    event.source.parent.getComponent('Buttonralm').componentEnabled=1
    event.source.parent.getComponent('Buttonrmat').componentEnabled=1
    event.source.parent.getComponent('Buttonrins').componentEnabled=1
    event.source.parent.getComponent('Buttonrot').componentEnabled=1
    event.source.parent.getComponent('msgincorrecta').visible=0
    event.source.parent.getComponent('Password Field').text=""
else:
    event.source.parent.getComponent('msgincorrecta').visible=1
    event.source.parent.getComponent('Buttonralar').componentEnabled=0
    event.source.parent.getComponent('Buttonralm').componentEnabled=0
    event.source.parent.getComponent('Buttonrmat').componentEnabled=0
    event.source.parent.getComponent('Buttonrins').componentEnabled=0
    event.source.parent.getComponent('Buttonrot').componentEnabled=0
```

Script de “Buttonralar”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es resetear todas las tags de cuenta de alarmas y luego
#inhabilita los botones de reset para que sea necesario reingresar la contraseña si se desea hacer otro reseteo
#
system.tag.write('conalmtotales', 0)
system.tag.write('confaltaalm', 0)
system.tag.write('confaltainst', 0)
system.tag.write('confaltamat', 0)
system.tag.write('confaltaotra', 0)
event.source.componentEnabled=0
event.source.parent.getComponent('Buttonrmat').componentEnabled=0
event.source.parent.getComponent('Buttonrins').componentEnabled=0
event.source.parent.getComponent('Buttonrot').componentEnabled=0
event.source.parent.getComponent('Buttonralm').componentEnabled=0
```

Script de “Buttonsalir”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es inhabilitar los botones de reset, limpiar el cuadro de
#texto “Password Field”, hacer invisible el mensaje de “msgincorrecta” y cerrar la ventana
#
event.source.parent.getComponent('Buttonralm').componentEnabled=0
event.source.parent.getComponent('Buttonrmat').componentEnabled=0
event.source.parent.getComponent('Buttonrins').componentEnabled=0
event.source.parent.getComponent('Buttonrot').componentEnabled=0
event.source.parent.getComponent('Password Field').text=""
event.source.parent.getComponent('msgincorrecta').visible=0
system.nav.closeParentWindow(event)
```

Script de “Buttonrmat”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es resetear las tag de la cuenta de alarmas por falta de
#piezas (confaltamat), resta el valor a la cuenta total de alarmas (“conalmtotales”) para que tenga el valor correcto y luego
#inhabilita los botones de reset para que sea necesario reingresar la contraseña si se desea hacer otro reseteo
#
totalmat=system.tag.read("confaltamat").value
system.tag.write('confaltamat', 0)
totalalar=system.tag.read("conalmtotales").value
totalalar=totalalar-totalmat
system.tag.write('conalmtotales', totalalar)
event.source.componentEnabled=0
event.source.parent.getComponent('Buttonralar').componentEnabled=0
event.source.parent.getComponent('Buttonrarm').componentEnabled=0
event.source.parent.getComponent('Buttonrins').componentEnabled=0
event.source.parent.getComponent('Buttonrot').componentEnabled=0
```

Nota: Los Scripts de “Buttonrins”, “Buttonrarm” y “Buttonrot” son similares al de “Buttonrmat” solo que resetean la tag correspondiente al tipo de alarma “confaltainst”, “confaltaalm” y “confaltaotra” respectivamente.

14.5.2.6 Elementos y scripts de la ventana de reset de conteo de sets

Elementos de ventana de reset de conteo de sets (“resetcontset”)





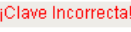


| Elemento | Nombre | Tipo | Property Binding | | Script asociado | Comentario |
|--|---|----------------|------------------|---|-----------------|---|
|  | Password Field | Password Field | | | No | Cuadro de texto que permite al usuario ingresar la contraseña para poder borrar los contadores (los caracteres se muestran como asteriscos) |
| Borrar todos los sets:  | labelset | Label | Text | Indicado por la tag de memoria “consettotales” | No | Muestra el número total de set ensamblados |
| Borrar solo set est. con lapicero:  | labelsc1 labelssl labelsetmod | Label | Text | Indicado por la tags de memoria de cada tipo de alarma “constand”, “constands1” y “consmod” | No | Muestra el número de set ensamblados según su tipo |
|  | Buttonrs Buttonrsc1 Buttonrssl Buttonrsm | Button | | | Si | Se habilitan según el script de “Buttonaceptar” al ingresar la contraseña correcta, al presionarlo borra la cuenta respectiva y actualiza la total |
|  | msgincorrecta | Label | | | No | Se hace visible si la contraseña es incorrecta, esto lo gestiona el script de “Buttonaceptar” |
|  | Buttonaceptar | Button | | | Si | Al presionarlo verifica si es la contraseña correcta, si es así habilita los botones de reset, sino muestra un mensaje de contraseña incorrecta y los botones continúan inhabilitados |
|  | Buttonsalir | Button | | | Si | Cierra la ventana |

Tabla 14.16. Elementos de la ventana de reset de conteo de sets (“resetcontset”) del SCADA en la modalidad 2.

Scripts

Script de “Buttonaceptar”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es verificar si la contraseña ingresada en “Password
#Field” es correcta, sino lo es hace visible el mensaje de “msgincorrecta” e inhabilita los botones de reset de los contadores,
#si es correcta oculta el mensaje de “msgincorrecta” y habilita los botones de reset
#
if event.source.parent.getComponent('Password Field').text=="123456":
    event.source.parent.getComponent('Buttonrs').componentEnabled=1
    event.source.parent.getComponent('Buttonrssl').componentEnabled=1
    event.source.parent.getComponent('Buttonrscl').componentEnabled=1
    event.source.parent.getComponent('Buttonrsm').componentEnabled=1
    event.source.parent.getComponent('msgincorrecta').visible=0
    event.source.parent.getComponent('Password Field').text=""
else:
    event.source.parent.getComponent('msgincorrecta').visible=1
    event.source.parent.getComponent('Buttonrs').componentEnabled=0
    event.source.parent.getComponent('Buttonrssl').componentEnabled=0
    event.source.parent.getComponent('Buttonrscl').componentEnabled=0
    event.source.parent.getComponent('Buttonrsm').componentEnabled=0
```

Script de “Buttonrs”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es resetear todas las tags de cuenta de sets y luego
#inhabilita los botones de reset para que sea necesario reingresar la contraseña si se desea hacer otro reseteo
#
system.tag.write('consettotales', 0)
system.tag.write('consmod', 0)
system.tag.write('constand', 0)
system.tag.write('constandsl', 0)
event.source.componentEnabled=0
event.source.parent.getComponent('Buttonrssl').componentEnabled=0
event.source.parent.getComponent('Buttonrscl').componentEnabled=0
event.source.parent.getComponent('Buttonrsm').componentEnabled=0
```

Script de “Buttonsalir”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es inhabilitar los botones de reset, limpiar el cuadro de
#texto “Password Field”, hacer invisible el mensaje de “msgincorrecta” y cerrar la ventana
#
event.source.parent.getComponent('Buttonrs').componentEnabled=0
event.source.parent.getComponent('Buttonrssl').componentEnabled=0
event.source.parent.getComponent('Buttonrscl').componentEnabled=0
event.source.parent.getComponent('Buttonrsm').componentEnabled=0
event.source.parent.getComponent('Password Field').text=""
event.source.parent.getComponent('msgincorrecta').visible=0
system.nav.closeParentWindow(event)
```

Script de “Buttonrscl”:

```
#Este script se ejecuta cuando se presiona este botón, lo que hace es resetear las tag de la cuenta de sets de escritorio
#estándar con lapicero (constand), resta el valor a la cuenta total de sets (“consettotales”) para que tenga el valor correcto y
#luego inhabilita los botones de reset para que sea necesario reingresar la contraseña si se desea hacer otro reseteo
```

```
#
totalscl=system.tag.read("constand").value
system.tag.write('constand', 0)
totals=system.tag.read("consettotaes").value
totals=totals-totalscl
system.tag.write('consettotaes', totals)
event.source.componentEnabled=0
event.source.parent.getComponent('Buttonrs').componentEnabled=0
event.source.parent.getComponent('Buttonrssl').componentEnabled=0
event.source.parent.getComponent('Buttonrsm').componentEnabled=0
```

Nota: Los Scripts de “Buttonrssl” y “Buttonrsm” son similares al de “Buttonrsl” solo que resetean la tag correspondiente al tipo de set "constandsl" y "consmo" respectivamente.

14.5.2.7 Elementos y scripts de la ventana de históricos

Elementos más importantes de la ventana de históricos (historicos)



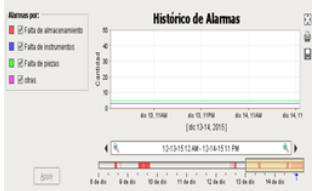
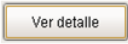


| Elemento | Nombre | Tipo | Property Binding | Script asociado | Comentario |
|---|-----------------------|-----------------------|---|-----------------|--|
|  | Multi-State Indicator | Multi-State indicator | State Value=0 OFF (Fondo gris letras negras) Value=1 ON (Fondo verde letras negras) Value=2 FALLA (Fondo amarillo letras negras) | Si | El valor se lo da la tag de memoria "estado" |
|  | Easy Chart 1 | Easy Chart | | No | En este gráfico se muestra la tendencia del número de set ensamblados según su tipo para los últimos 6 días. En la propiedad Tag Pens se coloca la tag de memoria/historia asociada que en este caso son "constand", "constandsl" y "conmod" el valor de 6 días en la propiedad "Startup Range" y en "Startup Selection" 1 día. |
|  | Easy Chart | Easy Chart | | No | En este gráfico se muestra la tendencia del número de alarmas según su tipo generadas en los últimos 6 días. En la propiedad Tag Pens se coloca las tags de memoria/historia asociada que en este caso son "confaltaalm", "confaltainst", "confaltamat" y "cconfaltaotra", el valor de 6 días en la propiedad "Startup Range" y en "Startup Selection" 1 día. |
|  | verdetalle | Button | | Si | El script de "Multi-State Indicator" hace visible o invisible este botón, al presionarlo muestra la ventana emergente para ver el detalle de la alarma |
|  | estadisticas | Button | | Si | Al presionarlo se abre la ventana "estadisticas" y se cierra esta |
|  | inicio | Button | | Si | Al presionarlo se abre la ventana "principal" y se cierra esta |

Tabla 14.17. Elementos de la ventana de históricos (“historicos”) del SCADA en la modalidad 2.

Scripts

Script de “estadisticas”:

Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “estadistic” y cerrar esta

#

```
window = system.nav.openWindow('estadisticas')
system.nav.closeParentWindow(event)
```

Script de “inicio”:

```
# Este script se ejecuta cuando se presiona este botón, lo que hace es abrir la ventana “principal” y cerrar esta
#
window = system.nav.openWindow('principal')
system.nav.closeParentWindow(event)
```

Nota: El Script de “Multi-State Indicator” y “verdetalle” es idéntico al descrito en los elementos de la ventana “principal”:

14.6 Abreviaturas utilizadas

- ADO: ActiveX Data Objects (objetos de datos ActiveX).
- ANSI: American National Standards Institute (Instituto nacional estadounidense de estándares)
- AS/RS: Automatic Storage/Retrieve System (Sistemas de recuperación y almacenamiento automatizado).
- CAN Open: Controller Area Network Open (Red de controladores de área abierta).
- CIM: Computer Integrated Manufacturing (Manufactura integrada por computadora).
- CNC: Computer Numeric Control (Control numérico por computadora).
- COM: Component Object Model (Modelo de objetos de componentes).
- DCOM: Distributed Component Object Model (Modelo de objetos de componentes distribuidos).
- ERP: Enterprise Resource Planning (Planificación de recursos empresariales).
- HMI: Human Machine Interface (Interfaz humano máquina).
- IP: Internet Protocol (Protocolo de Internet).
- ISO: International Organization for Standardization (Organismo Internacional de Normalización).
- JDBC: Java Database Connectivity (Conectividad de base de datos Java)
- MES: Manufacturing Execution System (Sistema de ejecución de manufactura).
- MODEM: abreviatura de Modulador Demodulador.
- MRP (Manufacturing Resource Planning).
- MTU: Master Terminal Unit (Unidad terminal maestra).
- OEE: Overall Equipment Efficiency (Eficiencia general de los equipos).
- OLE DB: Object Linking and Embedding for Databases (base de datos de enlace e integración)

de objetos).

- OPC: Object Linking and Embedding for Process Control (Objetos de enlace embebidos para control de procesos).
- OPC UA: OPC Unified Architecture (OPC de arquitectura unificada).
- PLC: Programmable Logic Controller (Controlador lógico programable).
- Profibus DP: Process Field bus- Decentralized Periphery (Bus de campo para procesos con periferia descentralizada).
- PVD: Pantallas de visualización de datos.
- RFID: Radio Frequency Identification (identificación por radio frecuencia).
- RTU: Remote Terminal Unit (Unidad terminal remota).
- SCADA: Supervisory Control and Data Acquisition (Control con supervisión y adquisición de datos).
- SED: Surface-conduction Electron-emitter Display (Panel de emisiones de electrones dirigidos).
- SQL: Structured Query Language (Lenguaje Estructurado de Consultas).
- SSL: Secure Sockets Layer (Capa de conexiones seguras).
- TCP: Transmission Control Protocol (Protocolo de control de transmisión).
- TIC: Tecnologías de la información/comunicación.
- UPS: Uninterruptible Power Supply (Fuente alimentación ininterrumpida).
- VDT: Visual Display Terminals (Terminales de visualización de datos).