Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Sistema automatizado de alerta en factoría por megafonía

Autor:

Truchuelo Mariscal, Guillermo

Tutor:

Moya De La Torre, Eduardo Julio

Departamento de Ingeniería de Sistemas y Automática

Valladolid, mayo de 2025

RESUMEN

En este documento se describe el proceso de implantación de un nuevo sistema de avisos a operarios dentro de una factoría desarrollado durante las prácticas en empresa en Michelin Valladolid.

Dicho proyecto surge de la necesidad de mejorar la comunicación y eficiencia en la gestión de alertas en tiempo real, reemplazando el sistema antiguo (SAF) por una solución basada en PI System (Osisoft) y Microsoft Power Automate. El sistema integra datos de autómatas industriales, visualiza estados de máquinas (como prensas y cabeceros de línea) mediante paneles interactivos en PI Vision, y envía notificaciones automatizadas a equipos de mantenimiento y producción a través de Microsoft Teams.

PALABRAS CLAVE

Automatización industrial, PI System, Osisoft, Autómatas programables (PLC), Alertas en tiempo real.

ABSTRACT

This document describes the implementation process of a new operator alert system within a factory, developed during an internship at Michelin Valladolid.

The project arose from the need to improve communication and efficiency in real-time alert management by replacing the legacy system (SAF) with a solution based on PI System (Osisoft) and Microsoft Power Automate. The system integrates data from industrial programmable logic controllers (PLCs), visualizes machine statuses (such as presses and line headers) through interactive dashboards in PI Vision, and sends automated notifications to maintenance and production teams via Microsoft Teams.

KEYWORDS

Industrial Automation, PI System, Osisoft, Programmable Logic Controllers (PLC), Real time alerts.

SIGLAS Y ACRÓNIMOS

- IIoT – Industrial Internet of Things (Internet Industrial de las Cosas)
- IA – Inteligencia Artificial
- ML – Machine Learning (Aprendizaje Automático)
- MAG – Magasin (Almacén)
- INT – Entero
- DINT – Doble Entero
- MSG – Mensaje
- MRA – Mantenimiento de Red y Accesos
- SAF – Sistema de Alertas de Fabricación
- PAM – Pantalla de Alertas Múltiples
- CdL – Cabecero de Línea
- XIO – Examine If Open
- XIC – Examine If Closed
- OTE – Output Energize
- OTL – Output Latch
- OTU – Output Unlatch

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN Y OBJETIVOS

1.1. INTRODUCCIÓN

A diario pueden surgir distintos inconvenientes durante el proceso de fabricación de los neumáticos (atasco de ruedas, prensa no abre, falta de una determinada calidad de neumático en almacén, etc.). Con el objetivo de resolver cualquiera de estos problemas de manera eficiente, se pretende diseñar un sistema de alerta a los trabajadores a través de mensajes automatizados que recibirán en sus dispositivos móviles u ordenadores.

Los elementos empleados para el desarrollo del proyecto son los siguientes:

- **Controladores lógicos programables (PLC)**: el software utilizado para la programación de los autómatas es el proporcionado por Rockwell Automation (RSLogix 500 y Logix 5000). Para poder cargar las modificaciones realizadas en los programas de los autómatas es necesario disponer de un servidor de comunicaciones que permita la conectividad de los dispositivos con estas aplicaciones, para ello se dispone del software RSLinx.
- **PI System**: es una plataforma de gestión de datos en tiempo real desarrollada por Osisoft (ahora parte de AVEVA). Se utiliza para recopilar, almacenar, analizar y visualizar datos operacionales procedentes de múltiples fuentes industriales, como los PLC. En este proyecto, PI System permite centralizar la información de los procesos de fabricación, detectar eventos anómalos y generar condiciones de alerta basadas en datos en tiempo real.
- **PI Vision**: es la herramienta de visualización web del PI System que permite crear paneles gráficos e interfaces personalizadas para monitorizar los datos del sistema de forma intuitiva. Los operadores y responsables pueden acceder a estos paneles desde cualquier dispositivo conectado a la red, lo que facilita el diagnóstico y seguimiento de incidencias.
- **Microsoft Power Automate**: es una herramienta de automatización de flujos de trabajo desarrollada por Microsoft. Permite conectar distintas aplicaciones y servicios para ejecutar tareas automáticas. En este caso, Power Automate se encarga de detectar las condiciones de alerta provenientes del PI System y enviar notificaciones automáticas a los usuarios correspondientes.

- **Microsoft Teams**: es una plataforma de colaboración y comunicación en equipo de Microsoft. Dentro del sistema de alertas, Teams actúa como el canal por el cual los trabajadores reciben los mensajes automáticos generados por Power Automate. Esto garantiza una respuesta rápida ante cualquier incidente en la planta de fabricación.

1.2. JUSTIFICACIÓN DEL PROYECTO

Este proyecto surge de la necesidad de renovación de un sistema de alertas a operarios en la fábrica de neumáticos de Michelin en Valladolid, concretamente en la sección de renovado, en la que se realiza el recauchutado de neumáticos de camión usados.

Para informar de cualquiera de los problemas que puedan surgir durante el proceso de fabricación se creó un sistema de alertas de fabricación (SAF), el cual, a través de un sistema de megafonía, alertaba del problema. Dicho sistema fue desarrollado en Visual Basic. NET 2008, su arquitectura monolítica y dependencia de tecnologías obsoletas limitaban su escalabilidad, por lo que era necesario implantar un nuevo sistema.

Debido a sus altas prestaciones se optó por PI System como solución tecnológica para la gestión de alertas. SAF no sólo se encargaba de alertar, sino que también gestionaba la visualización de los estados de las prensas de la Línea C de cocción en tiempo real y el sorteo de cepillado, por lo que el nuevo sistema elegido era perfecto para llevar a cabo el proyecto, ya que también cuenta con un sistema de visualización (PI Vision), el cual ya se utilizaba previamente en otras zonas dentro de la fábrica.

1.3. OBJETIVOS

Los objetivos planteados para el desarrollo del proyecto son los siguientes:

- **Adquirir y aplicar conocimientos** relacionados con la programación de autómatas (PLC).
- **Comprender** el funcionamiento del **proceso industrial**.
- **Conocer** los **protocolos de comunicación** entre los distintos dispositivos que forman parte del proceso.
- **Modificar** el **programa de los autómatas** creando las rutinas y variables oportunas para la gestión de las alertas.
- **Programar** las rutinas necesarias para la visualización de los datos recogidos por los autómatas en pantallas.

- **Gestionar** los **datos** proporcionados por los autómatas desde PI System y crear las pantallas informativas con PI Vision.
- **Automatizar** mensajes a Microsoft Teams a través de **flujos de trabajo** con Microsoft Power Automate.

1.4. ESTRUCTURA DE LA MEMORIA

Este documento recoge los fundamentos teóricos acerca de la automatización industrial utilizados durante el proyecto, el funcionamiento del sistema previo y el proceso de desarrollo del nuevo sistema. La memoria se va a estructurar de la siguiente manera:

- **Introducción y objetivos**: consta de una breve introducción acerca del proyecto, su justificación y los objetivos que se pretende alcanzar durante su elaboración.
- **Fundamentos de automática industrial**: breve introducción a la automatización industrial, se tratarán las principales características de los controladores lógicos programables y su programación.
- **Descripción del sistema previo**: detalla las principales características del sistema de alertas previo y su funcionamiento.
- **Descripción del software (sistema nuevo)**: se describen las características y funcionamiento de los programas que conforman el nuevo sistema que se pretende implantar.
- **Modificación de los programas de los autómatas**: explica el funcionamiento de los programas de los distintos autómatas y las modificaciones realizadas durante el proceso.
- **PI System**: explica el proceso de gestión de los datos obtenidos de los autómatas, la creación de las pantallas y la automatización de alertas.
- **Conclusiones y líneas futuras**: contiene las consideraciones alcanzadas en el proyecto y posibles mejoras a implementar en el futuro.
- **Bibliografía**: material empleado como referencia para la elaboración del proyecto.

2. FUNDAMENTOS DE AUTOMÁTICA INDUSTRIAL

En este apartado se tratará de hacer una introducción a los conceptos básicos de automática industrial.

2.1. CONCEPTOS BÁSICOS

La automatización industrial es el conjunto de herramientas y técnicas utilizadas para solucionar tareas repetitivas de forma automática. [1]

Para explicar el funcionamiento de un sistema automatizado, éste se va a dividir en 3 niveles:

- **Nivel Supervisor**: cualquier ordenador o computadora industrial con el que, a través de un software especial, se puede llevar a cabo la visualización y parametrización del proceso industrial (en este caso el software será PI System). Para la comunicación se utiliza un protocolo Ethernet Industrial.
- **Nivel de Control**: nivel donde se ejecutan todos los programas relacionados con la automatización del proceso. Este nivel está compuesto por controladores lógicos programables o PLC.

  Los PLCs son los autómatas encargados de ejecutar la lógica de control. Pueden estar interconectados con varios dispositivos de entradas y salidas y pueden comunicarse a través de varios protocolos de comunicación industrial.

- **Nivel de Campo**: lo conforman los equipos terminales de datos como sensores (captan variables físicas) y actuadores (ejecutan acciones). [2]

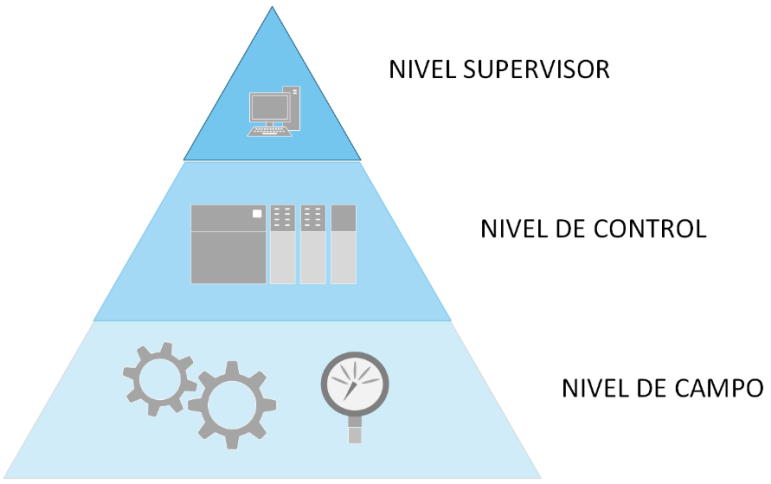


Figura 1. Representación de los niveles que conforman un sistema de automatización industrial

2.2. CONTROLADOR LÓGICO PROGRAMABLE (PLC)

Un PLC (Programmable Logic Controller) es un dispositivo utilizado para controlar maquinaria y procesos industriales de forma automática. Entre sus características podemos destacar:

- **Programación flexible**: los PLC se pueden programar para realizar una amplia variedad de funciones. Los lenguajes de alto nivel más utilizados son lenguaje Ladder, diagramas de funciones secuenciales o diagramas de bloques.
- **Entradas y Salidas digitales y analógicas (E/S)**: las entradas reciben señales de sensores o interruptores, mientras que las salidas controlan actuadores como motores o válvulas.
- **Fiabilidad y Durabilidad**: diseñados para entornos industriales, los PLC son robustos y capaces de operar en condiciones extremas de temperatura, vibración y ruido.
- **Comunicación Industrial**: permiten la comunicación con otros dispositivos industriales y sistemas de control a través de redes industriales como Ethernet o Profibus.
- **Modularidad**: muchos PLC son modulares, lo que permite ampliar su funcionalidad añadiendo más módulos de E/S, módulos de comunicación, etc.

La estructura de un PLC se puede describir en varios componentes esenciales:

- **Unidad Central de Procesamiento (CPU)**: es el cerebro del PLC. Se encarga de ejecutar las instrucciones del programa de control, realizar cálculos, gestionar datos y tomar decisiones. La CPU también maneja la comunicación con otros dispositivos.
- **Memoria**: se divide en varios tipos:
  - Memoria de programa: donde se almacena el programa de usuario que define las operaciones del PLC.
  - Memoria de datos: usada para almacenar datos temporales y valores de procesos como contadores, temporizadores, etc.
  - Memoria no volátil: para almacenar información que debe permanecer después de apagar el dispositivo.

- **Módulos de E/S**: permiten que el PLC interactúe con el proceso que controla. Los módulos de entrada reciben señales de diversos sensores o interruptores, mientras que los módulos de salida envían señales a actuadores como motores, válvulas, etc.
- **Fuente de Alimentación**: suministra la energía necesaria para el funcionamiento de los componentes internos del PLC.
- **Interfaz de Comunicación**: proporciona conectividad con otros dispositivos de control y sistemas de monitorización.
- **Interfaz Hombre-Máquina (HMI)**: aunque no siempre se encuentra integrada en el PLC, es una parte importante del sistema. Permite a los operadores interactuar con el PLC, proporcionando una interfaz para monitorizar y ajustar procesos.
- **Software de Programación**: utilizado para crear, modificar y depurar el programa que el PLC ejecutará. El software utilizado durante este proyecto será el proporcionado por Rockwell Automation, utilizando los programas RSLogix 500 y Logix 5000, según su compatibilidad con las versiones más antiguas y modernas con los controladores Allen-Bradley. [3]

Figura 2. PLC Allen-Bradley [10]

2.3. PROGRAMACIÓN DE AUTÓMATAS

Los 3 principales lenguajes de alto nivel que más se utilizan en la programación de PLCs son diagramas de bloques, diagramas de funciones secuenciales y Ladder.

- **Diagrama de Bloques**: La relación entre las entradas y salidas se establece mediante el uso de bloques de función, donde cada boque tiene un propósito o funcionalidad específica. Dichas entradas y salidas de los bloques están conectadas mediante enlaces los cuales pueden usarse para conectar dos puntos lógicos del diagrama. Ya sea una variable de entrada con una entrada del bloque, una salida de un bloque con una entrada de otro bloque o una salida de un bloque con una variable de salida. [11]

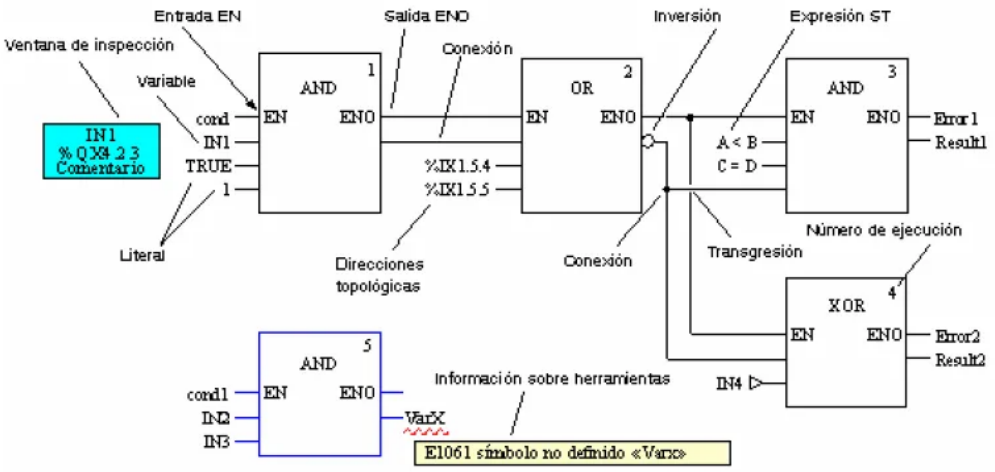


Figura 3. Ejemplo programación Diagrama de Bloques[11]

- **Diagramas de Funciones Secuenciales o GRAFCET**: Este lenguaje es bastante útil para controlar procesos que se basan en etapas secuenciales. Estas etapas pueden ser acciones por ejecutar o transiciones a través de condiciones lógicas. Cada etapa permanece inactiva mientras no se hayan cumplido y activado toda una serie de etapas anteriores que conlleven a la activación de ella, o bien que haya sido activadas directamente por el programador en la configuración inicial. [11]

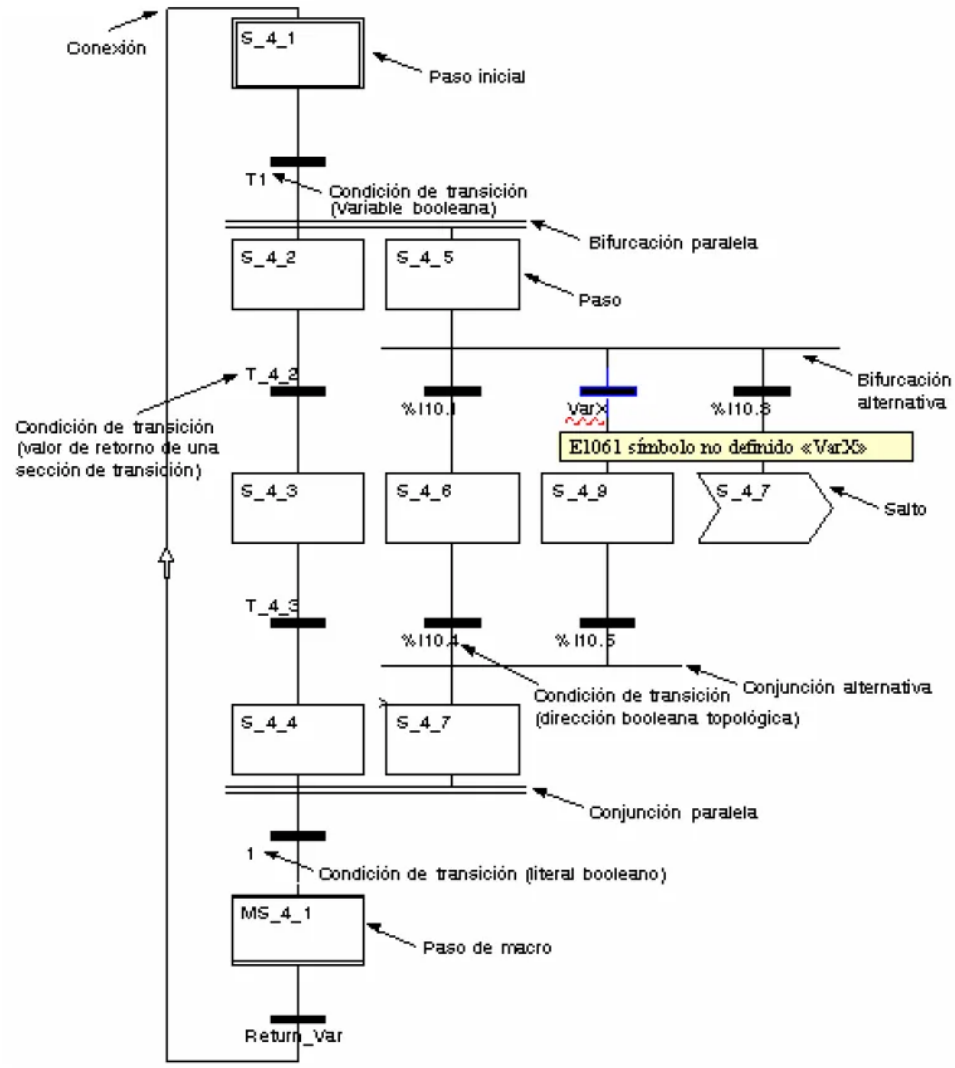


Figura 4. Ejemplo programación Diagramas de Funciones Secuenciales o GRAFCET[11]

- Ladder: el diagrama Ladder o diagrama de escalera es el lenguaje de interfaz gráfica más utilizado sin duda como lenguaje de programación de PLC, su nombre se debe a su forma estructural semejante a una escalera por donde corren dos rieles verticales, entre los cuales existen varios rieles horizontales que contienen la lógica.
  El riel izquierdo es el que recibe el flujo de energía (entrada) que representa el voltaje y deja pasar la energía al riel derecho que representa la tierra (salida). Su parecido con los antiguos controladores de relés es innegable y su lectura obedece siempre la misma secuencia; de izquierda a derecha y de arriba hacia abajo.

  Este lenguaje se basa en el uso de contactos y bobinas, ambos pueden ser normalmente abiertos (NA) o normalmente cerrados (NC), y

en dependencia de esto será su valor para activarse o desactivarse. [11]

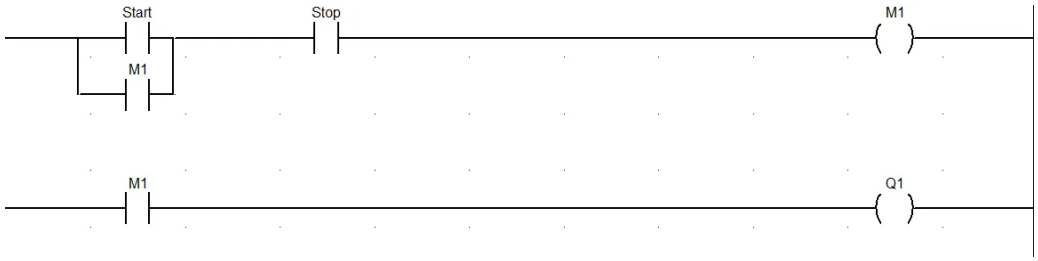


Figura 5. Ejemplo programación Ladder[11]

Los controladores utilizados en este proyecto están programados en lenguaje Ladder. A continuación, se explicará más detalladamente algunas de las instrucciones básicas de este lenguaje y que serán utilizadas a lo largo del proyecto.

INSTRUCCIONES TIPO RELÉ

XIC 

Cuando un dispositivo cierra su circuito, el módulo cuyo terminal está cableado al dispositivo detecta el circuito cerrado. El procesador refleja este estado activado en la tabla de datos. Cuando el procesador encuentra una instrucción XIC que direcciona el bit correspondiente al terminal de entrada, el procesador determina si el dispositivo está activado (cerrado). Si el procesador encuentra un estado activado, establece la lógica de escalera como verdadera para esta instrucción. Si el procesador encuentra un estado desactivado, establece la lógica de escalera como no verdadera para dicha instrucción.

XIO 

Cuando un dispositivo abre su circuito, el módulo cuyo terminal de entrada está cableado al dispositivo detecta un circuito abierto. El procesador refleja este estado desactivado en la tabla de datos. Cuando el procesador encuentra una instrucción XIO que direcciona el bit correspondiente al terminal de entrada, el procesador determina si el dispositivo está desactivado (abierto). Si el procesador encuentra un estado desactivado, establece la lógica de escalera como verdadera para esta instrucción. Si el

procesador encuentra un estado activado, establece la instrucción XIO como falsa.

	Si el bit está:	La instrucción es:	Estado lógico del bit:
XIC ⊢⊣	Activado	Verdadera	1
	Desactivado	Falsa	0
XIO ⊢⊣/⊢	Desactivado	Verdadera	0
	Activado	Falsa	1

Tabla 1. Lógica de las instrucciones XIC y XIO

OTE ——( )⊢

La instrucción OTE se usa para controlar un bit en la memoria. Si el bit corresponde a un terminal del módulo de salida, el dispositivo cableado a este terminal se activa cuando la instrucción se habilita y se desactiva cuando la instrucción se inhabilita. Si las condiciones de entrada que preceden la instrucción OTE son verdaderas, el procesador habilita la instrucción OTE. Si las condiciones de entrada que preceden la instrucción OTE son falsas, el procesador inhabilita la instrucción OTE. Cuando las condiciones de renglón se hacen falsas, el dispositivo correspondiente se desactiva.

OTL ——(L)⊢

Cuando se asigna una dirección a una instrucción OTL que corresponde a un terminal de un módulo de salida, el dispositivo de salida conectado a dicho terminal se activa cuando el procesador establece (habilita) el bit en la memoria del procesador. Si las condiciones de entrada que preceden la instrucción son verdaderas, el procesador habilita la instrucción OTL. Cuando las condiciones del renglón se hacen falsas (después de ser verdaderas), el bit permanece establecido y el dispositivo de salida correspondiente permanece activado. Use la instrucción OTU para desactivar el bit que se enclavó con la instrucción OTL.

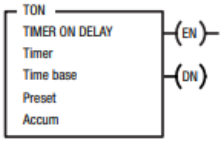
OTU ——(U)⊢

La instrucción OTU es una instrucción de salida retentiva que solamente desactiva un bit (no puede activar un bit). Esta instrucción normalmente se usa en parejas con una instrucción OTL (enclavamiento de salida) donde ambas instrucciones direccionan el mismo bit. La instrucción desactiva el bit que la instrucción OTL activó.

Si el renglón es:	El procesador:	Estado lógico del bit:
Verdadero	Activa el bit	1
Falso	Desactiva el bit	0

Tabla 2. Lógica de las instrucciones OTE, OTL y OTU

INSTRUCCIONES DE TEMPORIZADOR

TON 

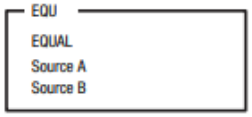
La instrucción TON se usa para activar y desactivar una salida después que el temporizador ha funcionado durante un intervalo de tiempo preseleccionado. La instrucción TON comienza a acumular el tiempo cuando el renglón se hace verdadero y continúa hasta que ocurre cualquiera de los siguientes eventos:

- el valor acumulado es igual al valor preseleccionado
- el renglón se hace falso
- una instrucción de restablecimiento restablece el temporizador
- el procesador restablece el valor acumulado cuando las condiciones del renglón se hacen falsas independientemente de que el temporizador haya sobrepasado o no el tiempo de espera

RES 

La instrucción RES es una instrucción de salida que restablece un temporizador o contador. La instrucción RES se ejecuta cuando su renglón es verdadero.

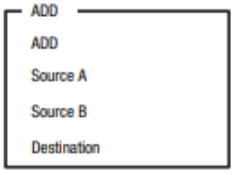
COMPARADORES

EQU 

La instrucción EQU se usa para probar si dos valores son iguales. La fuente A y la fuente B pueden ser valores o direcciones que contienen valores.

También existen otros comparadores para probar si dos valores no son iguales, si uno es mayor que otro, menor que otro, etc.

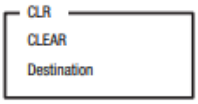
INSTRUCCIONES DE CÁCULO

ADD 

La instrucción ADD se usa para sumar un valor (origen A) y otro valor (origen B) y colocar el resultado en el destino. El origen A y el origen B pueden ser valores o direcciones que contienen valores.

También existen otras muchas operaciones de cálculo como la resta, división, multiplicación, etc.

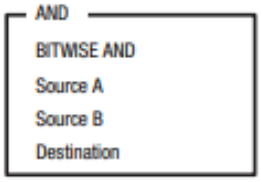
CLR



La instrucción CLR se para poner a cero todos los bits de una palabra. El destino debe ser una dirección de palabra.

INSTRUCCIONES LÓGICAS

AND

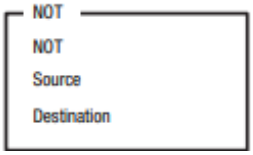


Se usa esta instrucción para realizar una operación AND usando los bits en las dos direcciones de origen.

Origen A	Origen B	Resultado
0	0	0
1	0	0
0	1	0
1	1	1

Tabla 3. Tabla de la verdad para la operación AND

NOT

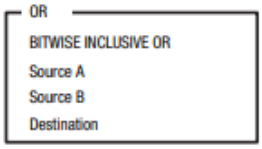


Se usa esta instrucción para realizar una inversión de bit

Origen	Resultado
0	1
1	0

Tabla 4. Tabla de la verdad para la operación NOT

OR

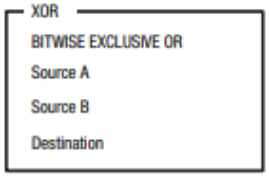


La instrucción OR se usa para realizar una operación OR usando los bits en los dos orígenes (constantes o direcciones).

Origen A	Origen B	Resultado
0	0	0
1	0	1
0	1	1
1	1	1

Tabla 5. Tabla de la verdad para la operación OR

XOR



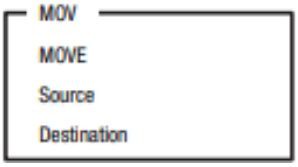
La instrucción XOR se usa para realizar una operación O exclusivo con el uso de los bits en los dos orígenes (constantes o direcciones).

Origen A	Origen B	Resultado
0	0	0
1	0	1
0	1	1
1	1	0

Tabla 6. Tabla de la verdad para la operación XOR

INSTRUCCIONES PARA MOVER O MODIFICAR BITS

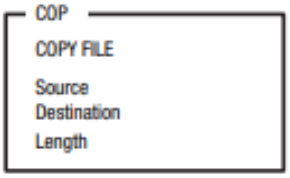
MOV



La instrucción MOV es una instrucción de salida que copia la dirección de origen a un destino. La instrucción mueve los datos durante cada escaneo siempre que el renglón permanezca verdadero.

INSTRUCCIONES DE ARCHIVO

COP



La instrucción COP es una instrucción de salida que copia los valores en el archivo de origen al archivo de destino. No se cambia el origen. La instrucción COP no usa los bits de estado.

INSTRUCCIÓN DE MENSAJE

MSG

Esta instrucción de mensaje (MSG) se usa para leer o escribir un bloque de datos a otra estación en la red DH+, a un coprocesador de control conectado, al VMEbus (Versa Module Europa, arquitectura de bus usada para interconectar módulos) que usa un procesador PLC-5 VME o a otro nodo en una red Ethernet. [4]

3. DESCRIPCIÓN DEL SISTEMA PREVIO

A continuación, se describirá tanto el funcionamiento como la organización del sistema de alertas de fabricación (SAF).

3.1. ANÁLISIS ORGÁNICO

3.1.1. ORGANIZACIÓN DE SAF (SISTEMA DE ALERTAS DE FABRICACIÓN)

La primera versión de SAF fue desarrollada en Visual Basic .NET 2003 con Programación Orientada a Objetos, pero a partir de la versión 2 (SAF2) está en Visual Basic.NET 2008.

Dentro de SAF2 existen las siguientes clases de objetos:

- **Buzón**: es un buffer de recepción de mensajes de autómata que es dado de alta en RSLinx.
- **ColeccionDeBuzones**: es la colección de todos los objetos de Buzón.
- **Campo**: es una entidad que asocia un nombre simbólico con un valor tipo cadena de texto.
- **Variable**: cuyo valor procede de un Buzón y que será referenciado en los mapas de pantalla para que su valor sea mostrado.
- **Constante**: cuyo valor es definido al iniciarse el programa.
- **ColeccionDeCampos**: es la colección de todos los objetos de Campo.
- **Mapa** (de pantalla): es la definición de las informaciones que deben aparecer en una pantalla; se compone de variables, constantes y literales.
- **ColeccionDeMapas**: es la colección de todos los objetos Mapa.
- **Dispositivo**: es un sistema que recibe mensajes de SAF y de él cuelgan una o varias pantallas u otros sistemas de presentación.
- **Pantalla**: es una referencia a la visualización de una pantalla de visualización física.
- **ColeccionDePantallas**: es la colección de todos los objetos pantalla.
- **Funciones script**: son funciones desarrolladas en VBScript o JavaScript que son invocadas para obtener los contenidos de las variables a partir del contenido de los buzones.
- **ColeccionDeOpciones**: es una clase de uso generalizado que contiene las opciones leídas en cada fichero de organización.

3.1.2. Ficheros de configuración

Todos los objetos de las clases citadas son creados al iniciarse el programa según las definiciones contenidas en diversos ficheros de configuración en formato texto ASCII:

- **Fichero de cabecera** o raíz del que cuelgan los demás.
- **Fichero de buzones**: es una lista de elementos formados por un nombre de variable, una función script de conversión y un tag RSLinx. Cada línea bien formada creará un campo en la colección de campos y un buzón en la colección de buzones.
- **Fichero de constantes**: es una lista de elementos formados por un nombre de campo y un valor asignado de tipo texto ASCII. Un campo variable, cuyo valor procederá rutinariamente de la conversión de contenido de un buzón, podrá recibir un valor inicial desde el fichero de constantes.
- **Fichero de mapas**: es una lista de elementos formados por un nombre de mapa, un tiempo de latencia y un nombre de fichero con la definición del mapa.
- **Fichero de definición de un mapa**: es un fichero de texto ASCII en el que aparecen literales, constantes y variables y que, una vez procesado para que las constantes y variables sean sustituidas por sus valores, es enviado a una o varias pantallas; puede haber varios de estos ficheros.
- **Fichero de dispositivos**: es una lista de elementos formados por un nombre de dispositivo, tipo de protocolo, tipo de empaquetado de datos, dirección puerto-IP local y dirección puerto-IP remoto.
- **Fichero de pantallas**: es una lista de elementos formados por un nombre de dispositivo, número de pantalla y el nombre del mapa que se les envía.
- **Fichero script**: es una fuente VBScript o JavaScript donde se programan las funciones para obtener los valores de las variables a partir del contenido de los buzones más otras funciones para otros usos.

3.1.3. Funcionamiento

SAF2 se basa en cuatro hilos de ejecución diferentes:

- El hilo del diálogo principal:
  - Al arrancar:
    - Lee el fichero de cabecera o raíz.
    - Genera los objetos con los que trabaja en función de la configuración y pone operativo el programa.
    - En el momento de cargar en memoria el fichero de funciones script, se ejecutará el cuerpo principal de tal fichero, lo cual permite personalizar SAF.
  - Reinicia el programa cuando se pulsa el botón del dialogo principal.
- El evento DataChange de RSLinx:
  - Efectúa un evento cada vez que se recibe un mensaje de autómata sobre un buzón.
  - Invoca una función script por cada elemento definido en el fichero de buzones que afecte al buzón que recibió un mensaje.
  - Actualiza una variable a partir de cada buzón mediante la ejecución de cada función script.
- Un temporizador cíclico para refrescar los mapas:
  - Cada vez que cambia una variable que afecta a uno o más mapas, se lanza una cuenta atrás de la latencia de cada mapa afectado y, tras agotarse la cuenta atrás, se actualiza el mapa.
  - Cada mapa actualizado es enviado inmediatamente a las pantallas suscritas al mismo (es distribuido).
- Un temporizador cíclico de vigilancia de RSLinx:
  - Verifica periódicamente que RSLinx está operativo y conectado y, si no, lo conecta e incluso lo inicia si está apagado.

3.2. MANUAL DE USUARIO

3.2.1. BREVE DESCRIPCIÓN DE SAF

SAF2 funciona en entorno Windows, recibe mensaje de autómatas Allen-Bradley a través de RSLinx y envía mensajes IP bajo protocolo UDP o TCP, en formato para pantallas luminosas de LEDs MPE (MP Electronics) o como texto ASCII terminado por <CR> (carriage return).

Así, el origen de los mensajes serán los autómatas de la red PLC Allen-Bradley y el destino serán dispositivos de presentación como pantallas de LEDs,

reproductores de mensajes vocales u otros, conectados a una red IP con protocolo UDP o TCP y formato específico MPE o texto ASCII delimitado por <CR>.

SAF2 trabaja con los siguientes elementos:

- Buzones: son los buffers de recepción RSLinx donde los autómatas dejan sus mensajes para SAF.
- Variables: reciben informaciones de los contenidos de los buzones una vez tratados.
- Constantes: son variables que reciben un valor inicial al arrancar el programa y cuyo contenido no cambia posteriormente.
- Scripts: son funciones script estándar de Windows, en VBScript o JavaScript, que son usadas para actualizar el contenido de las variables a partir de los mensajes recibidos en los buzones (todos deben estar en un único fichero).
- Mapas de pantalla: son modelos o formatos de la información que será mostrada en las pantallas y se componen de literales, variables y constantes.
- Dispositivos: son sistemas conectados a la red de los cuales pueden colgar en una red local aneja varias pantallas o elementos funcionalmente equivalentes.
- Pantallas: son los elementos físicos que reciben los mapas procesados a fin de ser visualizados en ellas.

El proceso normal de SAF2 es el siguiente:

- Un autómata hace llegar un mensaje a SAF.
- SAF lanza una o varias funciones scripts y cada una de ellas pone al día una variable interna de SAF a la cual se le asigna el resultado de la función script que se le ha asociado.
- Cada uno de los mapas que usa alguna de las variables que resulta actualizada es puesta al día.
- Cada una de las pantallas que está suscrita a los mapas afectados recibe un mensaje UDP o TCP, de formato MPE o ASCII, dependiendo del tipo de dispositivo del que cuelga.
- Para el caso de las pantallas MPE, se usa un convertidor de protocolo IP Ethernet a red RS485 que, según están configurados actualmente, permite enviarle un mensaje UDP con destino a una de las varias pantallas que pueden estar en la red RS485 (normalmente una).

De lo anterior se deduce que:

- Un mensaje puede poner al día más de una variable gracias a que pueden crearse varios buzones con el mismo PLC Allen-Bradley y lanzar en cada uno de ellos una función script diferente y cada una de ellas depositará su resultado en una variable diferente; una posibilidad es que un script genere un resultado en formato HTML y otro script para pantalla MPE.
- Una variable puede poner al día más de un mapa pues puede ser referenciada en más de uno de ellos.
- Un mapa puede ser enviado a más de una pantalla pues puede haber varias pantallas que hacen uso de un mismo mapa.
- Cada variable es actualizada por una función script, pero una misma función script puede ser invocada por varias variables con el fin de que todas esas variables sean actualizadas siguiendo un mismo proceso, aunque probablemente a partir de un tag PLC Allen-Bradley diferente.

Además:

- Puede programarse una función script que realice un proceso interno a Windows, pero sin utilizar los datos recibidos en el buzón (ejemplo: obtener la fecha del sistema).
- Una función script puede utilizar toda la potencialidad de un módulo script estándar de Windows: acceso a otras aplicaciones, acceso a bases de datos, utilización de diccionarios y listas, etc.

3.2.2. PUESTA EN MARCHA DE SAF2

Para poner en marcha SAF2, hay que ejecutar SAF.exe seguido de un nombre de un fichero de configuración raíz y éste, a su vez, hará referencia a otros ficheros que definen los elementos citados en el capítulo previo.

No obstante, en la versión 2 de SAF se ha empezado a poner todos los ficheros de configuración en un subdirectorio que cuelga de donde está el ejecutable SAF.exe y a pasar como parámetro ese subdirectorio seguido de "\" y el nombre de un fichero raíz de la configuración. De este modo, puede disponerse de varias configuraciones, cada una de ellas separada en un subdirectorio aislado.

Cuando SAF2 arranca, leerá el fichero raíz, cargará los demás ficheros de configuración, creará los objetos con los que trabaja y activará sus tareas internas.

Al iniciarse, aparece el siguiente diálogo:

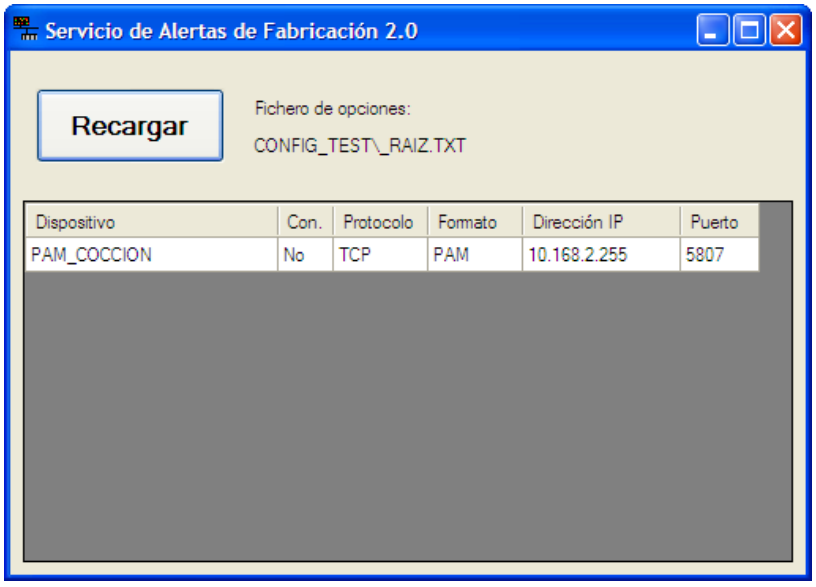


Figura 6. Diálogo 1 SAF

Como se puede ver, aparece la opción recargar opciones, es decir, volver a cargar la configuración y empezar de cero el programa.

Cuando estemos probando una nueva configuración, modificaremos los ficheros implicados y no hará falta detener SAF, lo que nos permite dar marcha atrás y dejar que el programa siga en marcha.

3.2.3. CONFIGURACIÓN DE SAF2

Para configurar SAF2, habrá que crear y rellenar convenientemente los siguientes ficheros de texto:

- Un fichero raíz.
- Un fichero con la lista de constantes.
- Un fichero con la lista de buzones.
- Un fichero de funciones script.
- Un fichero con la lista de mapas de pantalla.
- Un fichero con la lista de dispositivos (de los cuales cuelgan sus pantallas).

- Un fichero con la lista de pantallas de nuestra red.
- Puede haber otros ficheros utilizados por las funciones script (listas de datos u otros).

Todos estos ficheros cumplen unas reglas generales comunes:

- Deben ser editados con un editor plano, como el bloc de notas (notepad.exe).
- Cuando una línea comienza por una comilla simple es considerada un comentario y, si no, un mandato de configuración.
- Pueden escribirse espacios extra donde se desee.
- Un mandato es una línea de texto formada por varios fragmentos separados por uno o varios espacios.
- Cada fragmento, si se compone de subpartes, están separadas por una coma.

Fichero raíz

Veamos el contenido del fichero raíz de la primera puesta en marcha:

```
' Configuracion RAIZ_SAF1
CONSTANTES   CONFIG_SAF1\LISTA_CONSTANTES.TXT
FUNCIONES    CONFIG_SAF1\FUNCIONES.VBS
BUZONES      CONFIG_SAF1\LISTA_BUZONES.TXT
MAPAS        CONFIG_SAF1\LISTA_MAPAS.TXT
PANTALLAS    CONFIG_SAF1\LISTA_PANTALLAS.TXT
DISPOSITIVOS CONFIG_SAF1\LISTA_DISPOSITIVOS.TXT
```

Aparece un comentario en la primera línea (que empieza con una comilla simple).

Cada nombre de fichero de configuración aparece precedido por una palabra de tres letras que indica su función. Estas funciones son:

- CONSTANTES: fichero de lista de constantes.
- FUNCIONES: fichero de funciones script.
- BUZONES: fichero de lista de buzones.
- MAPAS: fichero de lista de mapas.
- PANTALLAS: fichero de lista de pantallas.
- DISPOSITIVOS: fichero de lista de dispositivos.

Por cada línea de las anteriores habrá un fichero con una lista en su interior y además, para cada elemento de la lista MAPAS habrá que definir cada uno de esos mapas.

No importa en que se escriban las líneas de este fichero pues SAF2 las lee todas en primer lugar y después las procesa en el orden lógico que le permite crear su estructura de datos interna.

Fichero de constantes

Veamos el fichero de constantes de la primera puesta en marcha:

```
' Literales
'
' Sintaxis: <simbolico> <literal>
'
' escribir un unico literal sin espacios
'
' para codigos especiales, usar % y dos cifras hexadecimales
'
' para incluir un espacio, escribir %20
'
' el simbolico puede ser usado en los mapas, precedido por @
'

SPACE    %20
SPACE2   %20%20
SPACE4   %20%20%20%20
SPACE8   %20%20%20%20%20%20%20%20
SPACE16  %20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20

PERCENT  %25

SYNC     %C9
NOSYNC   %CA
LINEA    %C7
COLOR    %A1
NEGRO    %A10
ROJO     %A11
VERDE    %A12
AMBAR    %A13
CORRER   %E0
CENTRO   %E1
INMED    %F0
BLINK    %A0
TIPO     %C1
ESPACIO  %20
VELOC    %C4
ESPERA   %B3
```

En primer lugar, decir que no se trata realmente de constantes sino de valores iniciales, es decir, que cada una de estas “llamadas” constantes

podría ser actualizada posteriormente a través de una función script pero tendrá el valor inicial especificado en este fichero.

Para definir una constante, basta con escribir un texto que figurará como identificador o nombre, dejar uno o más espacios de separación y escribir un único fragmento de texto que será su valor.

Cuando se desea especificar un carácter especial, puede hacerse a través del valor hexadecimal de su código ASCII que debe ser expresado con un carácter tanto por ciento (%) seguido por las dos cifras hexadecimales. Este es el caso de los caracteres especiales de control de las pantallas, el propio carácter espacio, que es usado como separador o el carácter "%" que es usado como prefijo de los códigos hexadecimales.

Fichero de buzones

Veamos el fichero de buzones de la primera puesta en marcha de la versión 2.0 de SAF:

```
' Lista de buzones DH
'
' Sintaxis: <campo> <conversor> <tag_rockwell>
'
' el campo puede ser usado en los mapas, precedido por @
'
' el conversor debe ser definido como funcion en el modulo VBS
'
' el topic del tag rockwell debe ser declarado en RSLinx
'
' si un campo tiene el mismo nombre que una constante, esta sera sobrescrita por el buzon
' (en realidad, una constante no es mas que una variable con valor inicial)
'

' calidades de coccion que escasean
CALIDADES      CALIDADES      [SAF1ADCG_UP]N250:0,L51
HHMM_CALIDADES HHMM_CALIDADES [SAF1ADCG_UP]N250:0,L51

' Tres números de operario con destino a cada linea de recogida de cepillado
STATUS_LINEA_1 STATUS_LINEA [SAF1EXCP]N251:0,L1
STATUS_LINEA_2 STATUS_LINEA [SAF1EXCP]N251:1,L1
STATUS_LINEA_3 STATUS_LINEA [SAF1EXCP]N251:2,L1

' Dos canales SAM de megafonia automatica
ALOCUCION_NORMAL MEGAFONIA [SAF1EXCP]N252:0,L2
ALOCUCION_LUDICA MEGAFONIA [SAF1EXCP]N253:0,L2

' Pantalla de Alertas Multiples de Coccion
STATUS_CABECERA_C STATUS_CABECERA    [SAF1CBLC]N100:0,L1
STATUS_PRENSA_C1  STATUS_PRENSA_C1   [SAF1CBLC]N100:1,L1
STATUS_PRENSA_C2  STATUS_PRENSA_C2   [SAF1CBLC]N100:2,L1
STATUS_PRENSA_C3  STATUS_PRENSA_C3   [SAF1CBLC]N100:3,L1
STATUS_PRENSA_C4  STATUS_PRENSA_C4   [SAF1CBLC]N100:4,L1
STATUS_PRENSA_C5  STATUS_PRENSA_C5   [SAF1CBLC]N100:5,L1
STATUS_PRENSA_C6  STATUS_PRENSA_C6   [SAF1CBLC]N100:6,L1
```

```
STATUS_PRENSA_C7  STATUS_PRENSA_C7   [SAF1CBLC]N100:7,L1
STATUS_PRENSA_C8  STATUS_PRENSA_C8   [SAF1CBLC]N100:8,L1
STATUS_GLOBAL_C   STATUS_GLOBAL      [SAF1CBLC]N100:0,L9
CALIDADES_WEB     CALIDADES_WEB      [SAF1ADCG_UP]N250:0,L51
```

Como se puede leer, hay que especificar para cada buzón:

- Un nombre de campo, es decir, una variable.
- Un conversor, es decir, el nombre de una función script.
- Un tag RSLinx.

El nombre de campo, o variable, es un simbólico que podrá ser especificado en los mapas de pantalla precedido por un carácter arroba de modo que, en su lugar, aparecerá el valor que le haya asignado la función script.

El tag RSLinx indica un mapa de memoria similar a un fragmento de fichero de autómata y ese es precisamente el buzón.

Cuando un buzón definido en este fichero recibe un mensaje de un autómata, SAF invoca a la función script especificada pasándole el contenido del buzón. Cuando la función script es ejecutada y SAF retoma el control, el valor retornado por la función script es depositado por SAF como valor del campo. Pasado un cierto instante, llamado latencia, todos los mapas que usan la variable que ha sido modificada son puestos al día y transmitidos a todas las pantallas físicas suscritas al mismo.

En el ejemplo, hay una variable CALIDADES y la función script que actualiza dicha variable también ha sido llamada CALIDADES. Se verá posteriormente que el mapa que usa la variable CALIDADES presentará el texto @CALIDADES donde se desea que aparezca su contenido. Por otro lado, en el fichero de funciones script también habrá una función llamada CALIDADES que es la que recibe los valores del buzón, los procesa y genera el texto que será depositado en la variable CALIDADES. Evidentemente, no hay por qué poner el mismo nombre a la variable y a la función.

Hay otra línea con HHMM_CALIDADES que presenta una particularidad ya citada previamente. La función script HHMM_CALIDADES es invocada cuando se actualiza el mismo buzón usado para CALIDADES, es decir, sobre el mismo instante. Sin embargo, el script HHMM_CALIDADES no usa los datos del buzón, sino que le pide la hora y los minutos al sistema Windows y los retorna a SAF.

Fichero de script

Se trata de un fichero VBScript o JavaScript estándar de Windows. Al ser un único fichero, hay que optar por uno de los dos lenguajes.

Tal fichero, como cualquier otro script, dispondrá de:

- Instrucciones que se ejecutan en el momento de su carga en memoria.
- Funciones invocadas desde tales instrucciones.
- Otras funciones que serán invocadas desde el interior de SAF cuando éste reciba un mensaje sobre algún buzón; estos son los conversores.

Veamos el código de la función CALIDADES ya citada anteriormente:

```
'-----------------------------------------
'
' Conversor CALIDADES
'
' Genera el fragmento de mensaje para visualizar las calidades que escasean
' en el almacen de coccion para pantalla MP Electronics

' Entrada: un parametro array de x enteros donde el primer entero es el
' numero de calidades recibidas y cada entero sucesivo tiene en el byte
' alto el numero de calidad y en el bajo la cantidad media

' Salida: cadena de texto con una linea según sintaxis MP Electronics (ASEL)

' Criterio: visualizar cantidad 3 en ambar y 2 o menos en rojo y, si 0,
' que ademas parpadee
'

function CALIDADES(buffer)
  dim x
  dim calidad
  dim cantidad
  dim prefijo
  dim sufijo
  dim final

  ' calcula elemento final a escrutar
  final= ubound(buffer)
  if buffer(0)<final then final= buffer(0)

  ' inicializa variables
  prefijo= ""
  sufijo= ""
  CALIDADES= "" ' hace que esta variable sea tipo string

  ' escruta los elementos desde el 1 hasta
  ' la cantidad indicada en el elemento 0
  for x=1 to final
    ' toma el byte alto
    calidad= (buffer(x) and &hFF00) / 256
    ' toma el byte bajo
    cantidad= buffer(x) and 255
    ' elige color segun la cantidad que queda
    select case cantidad
```

```
    case 0: ' color rojo y parpadeo
      prefijo= chr(&hA1) & "1" & chr(&hA0)
      sufijo= chr(&hA0)
    case 1: ' color rojo
      prefijo= chr(&hA1) & "1"
      sufijo= ""
    case 2: ' color rojo
      prefijo= chr(&hA1) & "1"
      sufijo= ""
    case 3: ' color ambar
      prefijo= chr(&hA1) & "3"
      sufijo= ""
    case else: ' color normal (verde)
      prefijo= CHR(&HA1) & "2"
      sufijo= ""
  end select
  ' concatenar...
  CALIDADES= CALIDADES & prefijo & calidad & ":" & cantidad & sufijo & " "
 next
 ' quita el espacio de mas del final
 CALIDADES= rtrim(CALIDADES)

end function
```

- El contenido del buzón es pasado a la función script como un objeto del tipo array, eso implica que, para N enteros recibidos de un autómata, dispondremos del elemento 0 al N-1.
- Para el caso particular del mensaje procesado por la función script CALIDADES por acuerdo entre N1 y N2, el primer elemento (el elemento 0) contendrá el número de elementos válidos que van a continuación de modo que:
  - El elemento 0 indica cuántos elementos de información hay a continuación.
  - El primer elemento de información es el 1 y el último elemento de información es aquel cuyo valor aparece en el elemento 0.
  - Puede haber más elementos nulos, no usados, hasta completar la longitud del array.
  - La longitud total del array se obtiene con la función ubound() y ha de ser 37 en nuestro ejemplo, ya que el tag usado previamente en la definición del buzón especifica ‘L37’.

- Observamos que esta función script retorna un valor de cadena, pero vemos también que se usan unos códigos hexadecimales: son códigos de control específicos para la pantalla física utilizada y, los usados aquí, son particularmente para cambiar el color del texto y para generar parpadeo en las pantallas.
  - Para cambiar el color se escribe &HA1 (hexadecimal A1) seguido de 0, 1, 2 o 3 (negro/apagado, rojo, verde, ámbar).

- Para que una sección de texto parpadee, hay que anteponerle y posponerle un carácter &HA1.

Cuando SAF recibe un mensaje sobre un buzón y llama a una de estas funciones, asigna el texto generado a una variable. Pero, cuando la función termina con error, SAF no asignará a la variable su valor retornado, sino el mensaje de error generado por el intérprete del lenguaje script. Esto hará que ese mensaje aparezca en la pantalla de visualización y el error sea fácilmente detectable.

Inicialización de un script

Cuando SAF carga en memoria el fichero script, éste es compilado y ejecutado de tal modo que, si hay funciones en su cuerpo principal, éstas serán ejecutadas al momento.

Esto es particularmente útil cuando necesitamos, por ejemplo, cargar diccionarios o conectar con servicios externos (bases de datos, por ejemplo). También será útil para testear los conversores, ya que podemos invocar un conversor en proceso de prueba desde el cuerpo principal del fichero script sin que sea necesario invocarlo desde SAF, lo cual requeriría preparar una simulación (autómata con programa, buzón, etc.).

Veamos un ejemplo de carga de una lista de números y nombres:

```
'*********************************************
' Inicialización del módulo
'*********************************************
dim nombres
Set nombres= CargarTablaNT("NOMBRES.TXT")
```

Este módulo invoca la función CargarTablaNT() pasándole como argumento el nombre de un fichero formado por líneas de texto compuestas por dos fragmentos: un número en cifras y una o varias palabras sustituyendo un nombre y apellidos (u otra funcionalidad similar).

Una vez ejecutada la función, si en alguna instrucción del fichero script colocamos “nombre(i)” siendo “i” una variable entera, se obtendrá el nombre del empleado número “i” tal cual haya sido definido en el fichero origen. Se trata pues de un diccionario cuyo origen es un fichero de texto.

El código de la función citada es el que sigue:

```
' ----------------------------------------
' Cargar tabla con campos Numérico y Texto
'
' Ejemplo típico: Número de empleado <espacio> Nombre
'

function CargarTablaNT(fichero)
  dim objFSO
  dim objFile
  dim texto
  dim espacio
  dim numero
  dim nombre

  ' crea un diccionario
  set dic= CreateObject("Scripting.Dictionary")

  ' crea un manipulador de ficheros
  Set objFSO = CreateObject("Scripting.FileSystemObject")
  ' crea un fichero abriéndolo como texto
  Set objFile = objFSO.OpenTextFile(fichero,1)
  ' lee linea a linea hasta el final
  While Not objFile.AtEndOfStream
    texto = CStr(objFile.ReadLine)        ' lee una linea y convierte a texto
    espacio= instr(texto," ")             ' busca la posición del primer espacio (separador)
    numero= cint(left(texto,espacio-1))   ' convierte a numérico lo anterior al espacio (clave)
    nombre= right(texto,len(texto)-espacio) ' extrae el texto a partir del espacio (atributo)
    dic.Item(numero)= nombre              ' da de alta el elemento en el diccionario
  Wend
  ' cierra el fichero
  objFile.Close
  ' retorna el objeto diccionario
  Set CargarTablaNT= dic

end function
```

Un posible fichero de datos es el siguiente (NOMBERS.txt):

```
1  Lopez
2  Martinez
3  Najera
4  Perez
5  Menendez
6  Arias
7  Velasquez
8  Navarro
9  Arranz
10 Olea
11 Pereda
12 Mejias
```

Pueden darse otros usos como, por ejemplo, una lista que asocie un código de mensaje con un texto de aviso.

Es posible ejecutar en el cuerpo principal del módulo script esta y otras funciones similares como conectar con una base de datos remota o local, u otras como inicializar datos, enviar mensajes, etc. No obstante, la idea principal es programar las funciones que convierten los datos recibidos por los autómatas en mensajes legibles por el receptor que los necesita (humano o máquina).

Fichero de lista de mapas

Veamos el fichero de mapas de la primera puesta en marcha de la versión 2.0 de SAF:

```
' Lista de mapas de pantalla
'
' Sintaxis: <identificador> <latencia> <fichero>
'
' latencia es el tiempo que se retrasa el refresco del mapa desde que cambia el ultimo
' campo contenido en el mismo (decimas de segundo de 2 a 600)
' la latencia debe ser superior al tiempo de ejecución del script más lento que afecte al mapa
'

COCCION          1 CONFIG_SAF1\MAPA_COCCION.TXT
CEPILLADO        1 CONFIG_SAF1\MAPA_CEPILLADO.TXT

MEGA_NORMAL      1 CONFIG_SAF1\MAPA_MEGAFONIA_NORMAL.TXT
MEGA_LUDICO      1 CONFIG_SAF1\MAPA_MEGAFONIA_LUDICA.TXT

STATUS_CABECERA_C 1 CONFIG_SAF1\MAPA_STATUS_CABECERA_C.TXT
STATUS_PRENSA_C1  1 CONFIG_SAF1\MAPA_STATUS_PRENSA_C1.TXT
STATUS_PRENSA_C2  1 CONFIG_SAF1\MAPA_STATUS_PRENSA_C2.TXT
STATUS_PRENSA_C3  1 CONFIG_SAF1\MAPA_STATUS_PRENSA_C3.TXT
STATUS_PRENSA_C4  1 CONFIG_SAF1\MAPA_STATUS_PRENSA_C4.TXT
STATUS_PRENSA_C5  1 CONFIG_SAF1\MAPA_STATUS_PRENSA_C5.TXT
STATUS_PRENSA_C6  1 CONFIG_SAF1\MAPA_STATUS_PRENSA_C6.TXT
STATUS_PRENSA_C7  1 CONFIG_SAF1\MAPA_STATUS_PRENSA_C7.TXT
STATUS_PRENSA_C8  1 CONFIG_SAF1\MAPA_STATUS_PRENSA_C8.TXT
STATUS_GLOBAL_C   1 CONFIG_SAF1\MAPA_STATUS_GLOBAL.TXT
CALIDADES_ESCASAS 3 CONFIG_SAF1\MAPA_CALIDADES_ESCASAS.TXT
```

En el mismo hay que especificar una línea por cada mapa con:

- El nombre del mapa
- La latencia
- El nombre del fichero donde está la definición del mapa

El nombre del mapa es necesario para cuando definamos las pantallas poder decir a qué mapa está suscrita cada una.

Cuando un buzón recibe un mensaje, se lanza una función script que actualiza el contenido de una variable y, al cambiar la variable, se actualizan los mapas que la usan, al cambiar un mapa, es reenviado a las pantallas que están suscritas al mismo.

La latencia es el tiempo que hacemos transcurrir desde que un mapa se ve afectado por una variable cambiante hasta que el mapa es recalculado y reenviado a las pantallas afectadas. La existencia de la latencia se justifica con el hecho de que, a veces, hay que dejar un pequeño tiempo al sistema operativo para que libere un fichero.

Por otro lado, en ocasiones, un buzón afecta a varias variables por lo que desencadenará la ejecución de varias funciones script que afectarán a su vez a varios mapas y será necesario dar un pequeño tiempo para que todas las funciones scripts tengan tiempo de finalizar su ejecución. De ese modo, se ejecutarán todas las funciones script, posteriormente finalizará el tiempo de latencia que cuenta a partir de la recepción del tag desde un autómata y se producirá un único envío a las pantallas afectadas. Si no hubiera un tiempo de latencia, cada fin de ejecución de una función script provocaría la actualización de su variable asociada con la consecuente actualización de todos los mapas que la usan y los envíos a todas las pantallas suscritas a cada mapa recién afectado.

También, pueden llegar mensajes a diferentes buzones en instantes cercanos del tiempo y la latencia retrasará la actualización del mapa para procurar que no haya 'guiños', es decir, que, si pueden llegar por ejemplo dos mensajes contra las variables de un mismo mapa con una separación de 5 segundos, una latencia de 6 segundos o más hará que sólo se produzca una actualización y, por tanto, un único envío a las pantallas afectadas. Si la información cambia con cierta frecuencia podría interesar establecer una latencia que baje la frecuencia de actualización.

El tercer parámetro es el nombre del fichero dónde aparece la definición del mapa. Cada mapa requiere en general un fichero diferente.

Fichero de definición de un mapa de pantalla

Veamos el mapa de pantalla de la primera puesta en marcha para la información de calidades a punto de agotarse en el almacén de cocción:

```
' Mapa de visualizacion para Coccion / Prensas
'
' Marca: MP Electronics
'
' Modelo: MPC35-16 de 3 lineas independientes de 16 caracteres
'

@SYNC
  @LINEA 1
    @TIPO 8 @VELOC 20
```

```
    @INMED @VERDE ALMACEN @SPACE COCCION
  @LINEA 2
   @TIPO 8 @VELOC 20
    @CORRER
     @CALIDADES @SPACE8
  @LINEA 3
   @TIPO 8 @VELOC 20
    @INMED @VERDE A @SPACE LAS @SPACE @HHMM_CALIDADES
@NOSYNC
```

Todos los elementos que van precedidos de un carácter arroba son variables o constantes y, los que no, son literales; es decir, que los elementos que empiezan por '@' serán sustituidos por su valor y el resto aparecerán tal cual.

Recordemos que una constante y una variable son elementos de SAF que pertenecen a un mismo diccionario interno, pero:

- una constante recibe su valor al arrancar SAF y no será modificada posteriormente, es decir, ningún buzón la referenciará.
- una variable recibe su valor desde un mensaje de autómata, pero nada impide que también reciba un valor inicial en el fichero de constantes.

En el mapa del ejemplo, @CALIDADES y @HHMM_CALIDADES son las únicas variables y todos los demás elementos que comienzan por '@' son constantes.

Las constantes pueden ser divididas en dos grupos:

- @SPACE y @SPACE8 que sirven para presentar espacios en las pantallas.
- las demás corresponden a mandatos hexadecimales específicos para las pantallas MP Electronics.

De lo último se deduce que cada mapa de pantalla es específico para un determinado hardware y requerirá una serie de constantes de mandato en el fichero de constantes.

Como las variables forman parte integrante de los mapas, también las funciones scripts que las generan tendrán que ser específicas para un determinado hardware, ya que usan códigos hexadecimales específicos en sus instrucciones.

Fichero de lista de dispositivos

Un dispositivo es un aparato que se conecta a la red, recibe mensajes de SAF2 y los hace llegar a una o más pantallas (o similares) que cuelgan del mismo a través de una red local (física o virtual).

Actualmente se tienen en el taller 3 tipos de dispositivos:

- Convertidor serie Ethernet Digi Connect. Estos dispositivos incorporan una red RS485 de la que está colgando una sola pantalla.
- Programa SAM (Sistema de Megafonía Automática). Tiene una red virtual de la que pueden colgar varios canales de audio. A cada canal se le puede cambiar la música de inicio, la velocidad de dicción y la música de final.
- PC con programa PAM (Pantalla de Alertas Múltiples). Tiene una red virtual de manera que cada nodo de esta tiene como destino una ventana de internet Explorer que ocupa una determinada área de la pantalla.

Veamos el fichero de lista de dispositivos para la primera puesta en marcha de SAF2:

```
' Lista de pantallas
'
' Lista de dispositivos fisicos de remoto de alertas
'
' Sintaxis: <identificador> <protocolo> <tipo_empaquetado> <direccion_ip_local> <puerto_ip_local> <direccion_ip_remota> <puerto_ip_remoto>
'
' identificador: simbolico del nodo remoto (pantalla, megafonia, etc.)
' protocolo: TCP o UDP
' tipo_empaquetado: PAM, texto plano para Pantalla de Alertas Multiples; MPE, protocolo de pantallas MPE (ASEL)
' direccion_ip_local: direccion ip local (este ordenador)
' puerto_ip_local: puerto ip local (en este ordenador; 0=comodin)
' direccion_ip_remoto: direccion ip remoto (pantalla, etc.)
' puerto_ip_remoto: puerto ip remoto (pantalla, etc.)
'
PAM_COCCION   TCP PAM 10.168.10.242 0 10.168.0.36 5807
MPE_DIS2      UDP MPE 10.168.10.242 58002 10.168.1.252 2101
MPE_MEDALLADO UDP MPE 10.168.10.242 58003 10.168.0.37  2101
MPE_CEPILLADO UDP MPE 10.168.10.242 58004 10.168.0.33  2101
SAM_MEGAFONIA UDP MPE 10.168.10.242 58005 10.168.2.11  2101
```

Para cada dispositivo hay que especificar:

- El nombre simbólico del dispositivo (necesario para después definir sus respectivas pantallas)
- El protocolo IP: TCP o UDP
- El formato de los datos: pantalla MPE o texto ascii para PAM
- La dirección IP local en el PC donde se ejecuta SAF2
- El puerto local (ayuda a mejorar la seguridad)
- La dirección IP remota del dispositivo de presentación de la información
- El puerto IP remoto

Fichero de lista de pantallas

Éste especifica la lista de pantallas de nuestra red. Veamos el fichero para la primera puesta en marcha de SAF2:

```
' Lista de pantallas
'
' Lista de pantallas fisicas de destino de alertas
'
' Sintaxis  <punto_remoto>:<nodo_o_puerto_rs485> <mapa>
'
' punto_remoto: simbolico que referencia el dispositivo remoto que recibe la alerta
' nodo_o_puerto_rs485: nodo dentro de la red local de destino
' mapa: mapa cuyo contenido sera volcado sobre el dispositivo cuando alguna de sus variables cambie de estado
'

PAM_COCCION:100 STATUS_CABECERA_C
PAM_COCCION:101 STATUS_PRENSA_C1
PAM_COCCION:102 STATUS_PRENSA_C2
PAM_COCCION:103 STATUS_PRENSA_C3
PAM_COCCION:104 STATUS_PRENSA_C4
PAM_COCCION:105 STATUS_PRENSA_C5
PAM_COCCION:106 STATUS_PRENSA_C6
PAM_COCCION:107 STATUS_PRENSA_C7
PAM_COCCION:108 STATUS_PRENSA_C8
PAM_COCCION:120 STATUS_GLOBAL_C
PAM_COCCION:121 CALIDADES_ESCASAS

MPE_DIS2:1     COCCION
MPE_MEDALLADO:3 COCCION
MPE_CEPILLADO:1 CEPILLADO

SAM_MEGAFONIA:1 MEGA_NORMAL
SAM_MEGAFONIA:2 MEGA_LUDICO
```

Para cada pantalla hay que especificar:

- un nombre de dispositivo seguido por ":" y un número de pantalla entre 1 y 255
- el nombre del mapa al que está suscrita la pantalla (el mapa que, tras actualizarse, será enviado a la pantalla)

Como puede apreciarse en el ejemplo, el mapa COCCION será enviado a dos pantallas diferentes. [14]

4. DESCRIPCIÓN DEL SOFTWARE (SISTEMA NUEVO)

4.1. PI SYSTEM

4.1.1. INTRODUCCIÓN

PI System es una cartera integrada de soluciones que permite a las operaciones industriales recopilar, limpiar, almacenar, enriquecer y visualizar datos operativos en tiempo real. Facilita a los operadores, gerentes de planta, analistas de datos y ejecutivos optimizar la eficiencia, garantizar la resiliencia e impulsar la sostenibilidad.

Aunque fue creado por OSISOFT, un fabricante de software de aplicación, en 2021 fue comprado por la empresa de producción AVEVA.

A continuación, se tratará de explicar tanto el funcionamiento como las distintas soluciones que ofrece PI System de cara al tratamiento y visualización de datos. [5]

4.1.2. FUNCIONAMIENTO

1- Recopilación de datos

Se capturan datos a través de sensores, gateways IIoT o activos remotos.

2- Almacenamiento de datos

Los datos son almacenados y gestionados por AVEVA Edge Data Store y AVEVA PI Server:

- AVEVA Edge Data Store: Recopila, almacena y accede a datos de operaciones de activos remotos en entornos hostiles. [15]
- AVEVA PI Server: motor de almacenamiento, contextualización, análisis y notificación de datos, de gran volumen y en tiempo real.[8]

3- Servicios en la nube

Dichos datos también pueden ser procesados por unos servicios de datos basados en la nube (AVEVA Connect).[16]

4- Análisis y predicción

Se utilizan plataformas de inteligencia artificial y machine learning para realizar análisis predictivos y generar insights sobre el comportamiento futuro.

5- Visualización

Con AVEVA PI Vision se pueden convertir datos brutos en visualizaciones detalladas y compartir información con toda la empresa.

6- Integración de datos

Se integran otros sistemas empresariales y herramientas para mejorar la integración y análisis a nivel corporativo [6] a través de aplicaciones como AVEVA PI DataLink (complemento de Microsoft Excel que permite recuperar información de PI System directamente en una hoja de cálculo) [17] y AVEVA PI Integrator for Business Analytics (transforma los datos de PI System a un formato listo para la toma de decisiones, compatible con herramientas de inteligencia empresarial, como Microsoft Power BI, Tableau, Tibco Spotfire y QlikView). [18]

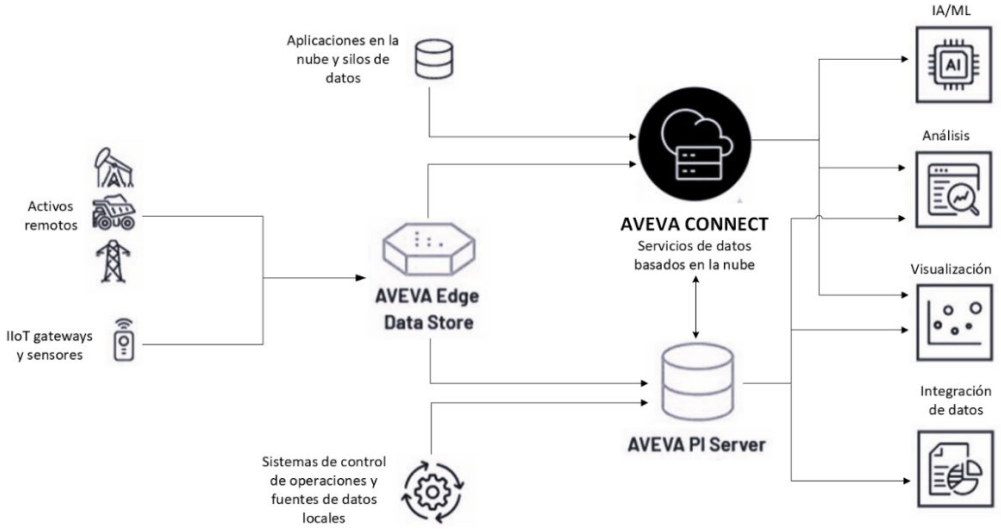


Figura 7. Esquema funcionamiento PI System [12]

4.1.3. PI SERVER

PI Server es la solución estándar de la industria para almacenar, agregar, contextualizar y utilizar datos de operaciones en tiempo real.

Entre sus principales características se pueden distinguir:

- Recuperación y almacenamiento de millones de etiquetas de datos y cientos de miles de valores por segundo a lo largo de varias décadas.

Dichos valores se pueden entregar en tiempo real a usuarios y sistemas en múltiples ubicaciones.

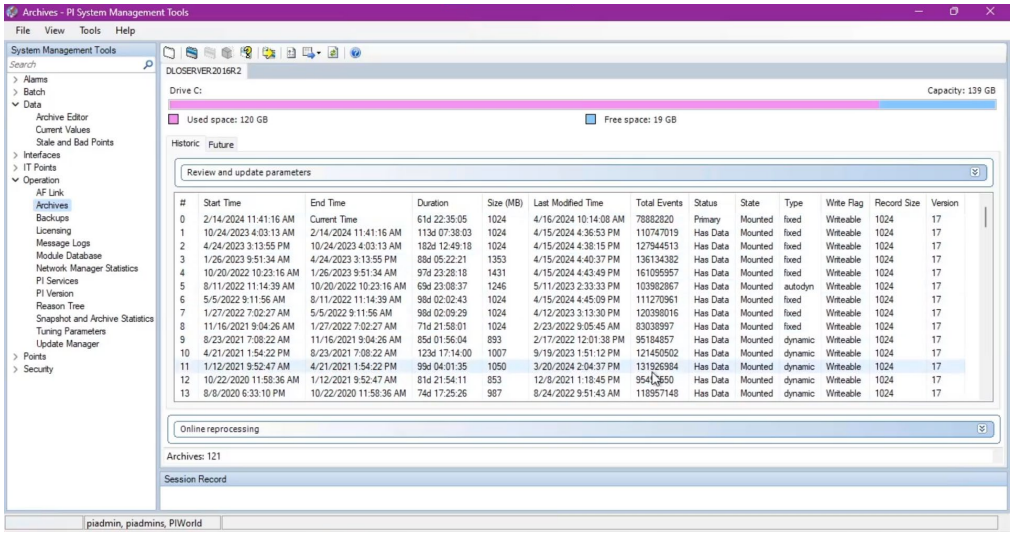


Figura 8. Ejemplo del histórico de los datos almacenados en PI Server[8]

- Estructuración de los datos en función de los equipos, activos y otros objetos conocidos de los procesos de producción, incluido el flujo de procesos entre ellos.

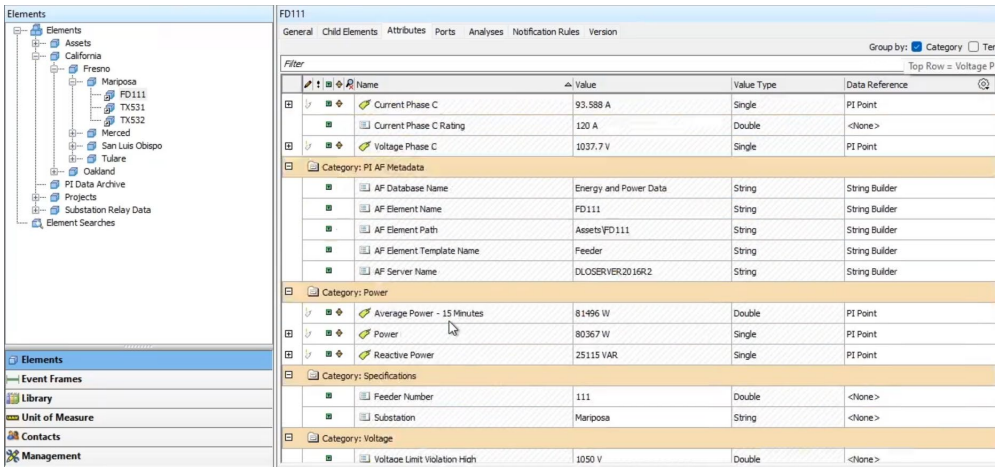


Figura 9. Ejemplo de estructuración de los datos en PI Server[8]

- Creación de cálculos, funciones, modelos predictivos de equipos, análisis hipotéticos y lógica multifacética.

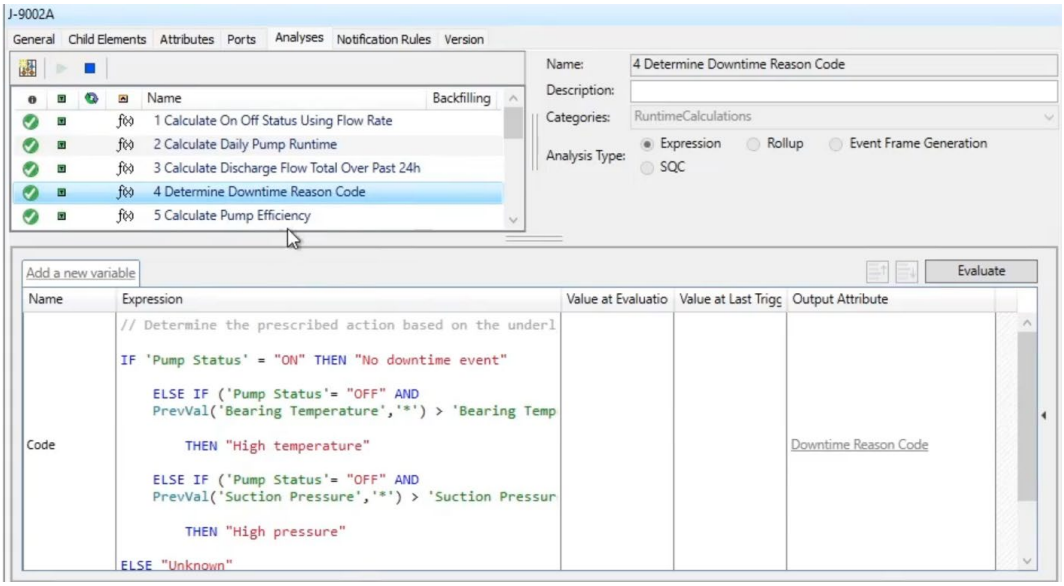


Figura 10. Ejemplo de creación de una función sencilla en PI Server[8]

- Definición de reglas para detectar y registrar eventos.

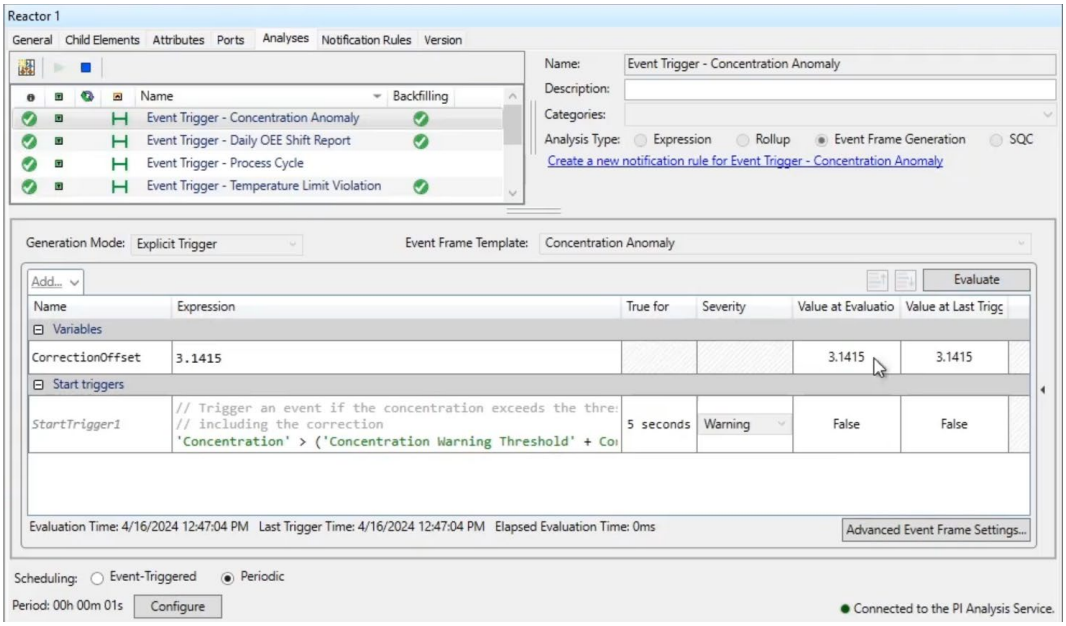


Figura 11. Ejemplo de definición de reglas para detección de anomalías en PI Server[8]

- Envío de alertas en tiempo real cuando se detecte un determinado evento definido por el usuario. [8]

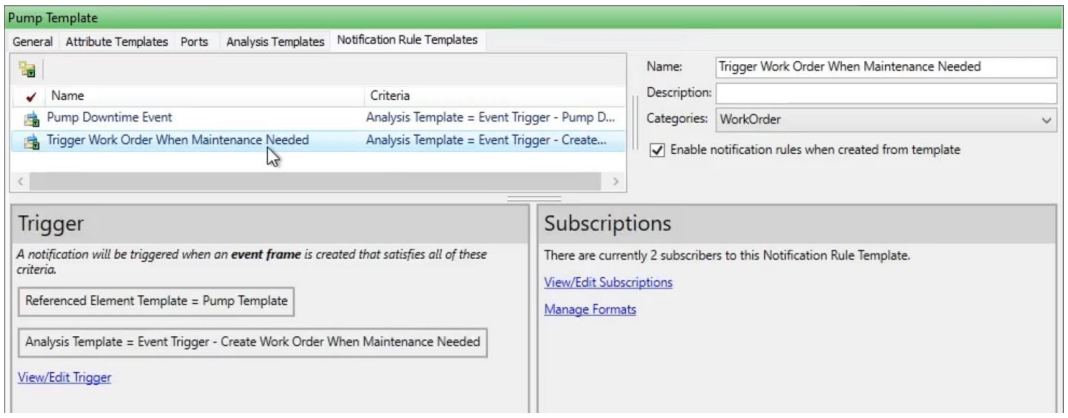


Figura 12. Ejemplo de alerta en PI Server (Trigger, a la izquierda, es la definición del evento; Subscriptions, a la derecha, es el destino de la notificación) [8]

4.1.4. PI VISION

PI Vision es una herramienta basada en la web para crear pantallas y paneles de control. Los usuarios pueden crear fácilmente visualizaciones configurables de sus datos de operaciones en tiempo real.

Se puede acceder a visualizaciones de datos críticos y paneles de control en cualquier momento y lugar, desde cualquier dispositivo o navegador. [7]

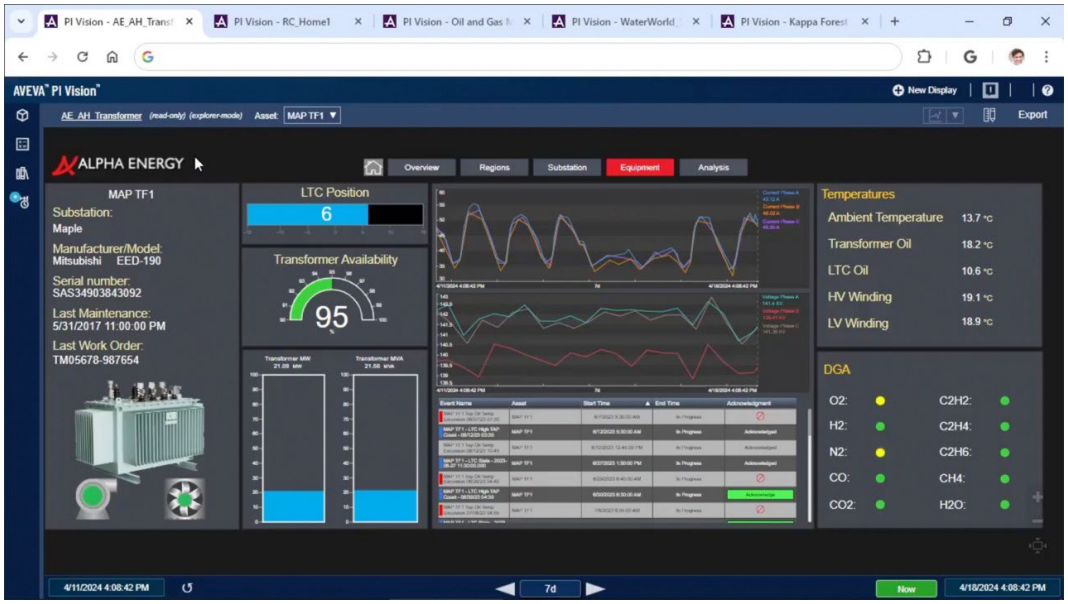


Figura 13. Ejemplo de visualización de datos con PI Vision desde navegador [7]

4.2. MICROSOFT POWER AUTOMATE

Power Automate es una aplicación de Microsoft para la automatización de flujos de trabajo. Permite crear y personalizar tareas automáticas en la nube,

integrando otras aplicaciones y servicios para recibir notificaciones, sincronizar archivos, recopilar datos y mucho más.

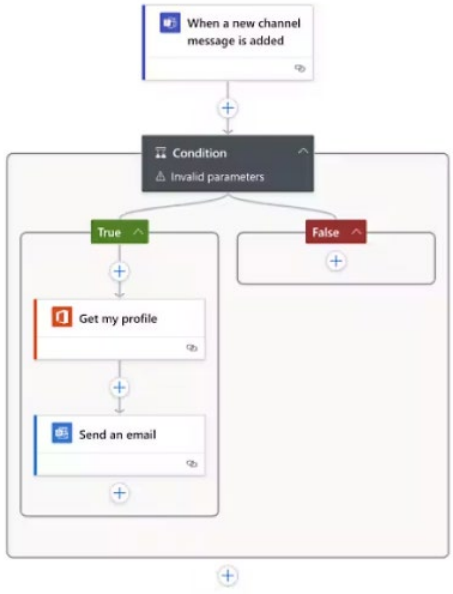


Figura 14. Ejemplo de flujo de trabajo con Power Automate [9]

En la imagen anterior (Figura 14) tenemos un ejemplo muy sencillo de un flujo creado con Power Automate. Cuando se cumplen unas determinadas condiciones se envía un email. [9]

5. MODIFICACIÓN DE LOS PROGRMAS DE LOS AUTÓMATAS

Las zonas de la fábrica de las que nos interesa recibir información son:

- **Cocción**: estado de las prensas y cabecero de la línea C, control de las calidades en almacén y gestión de defectos.
- **Almacén del sótano**: gestión de defectos.
- **Almacén de cepillado**: gestión de defectos.
- **Almacén del sótano**: sorteo, estado de las líneas y gestión de defectos.

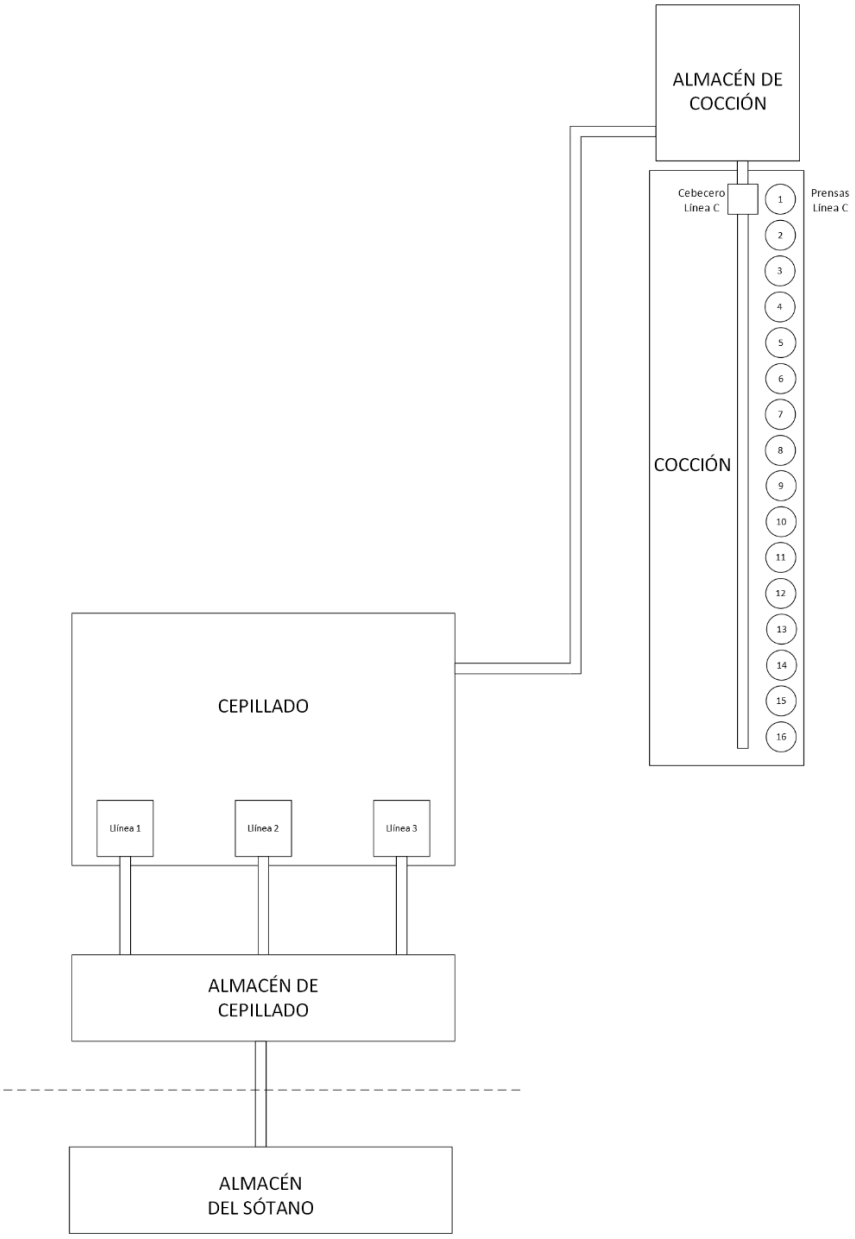


Figura 15. Esquema de distribución de las zonas de interés dentro de la fábrica

5.1. COCCIÓN

Los avisos a megafonía y la pantalla informativa del estado de las prensas están gestionados por el almacén de cocción, el cual se divide en dos programas: el de animación (PROC_ANI) y el de gestión (PROC_GES).

5.1.1. ESTADO DE LAS PRENSAS

En la pantalla se informará mediante un código de colores del estado de las 16 prensas de la línea C y de la cabecera de línea. Además, se irán mostrando cada 3 segundos de dos en dos el tipo de defecto que tiene cada prensa.

Se mostrará en pantalla una plantilla dividida en 17 cuadrados correspondientes a cada una de las 16 prensas y otro para la cabecera, cuando cambie el estado de la prensa su casilla correspondiente cambiará de color:

Figura 16. Ejemplo de visualización del estado de las prensas y cabecero

El significado de cada uno de los colores de los posibles estados y el proceso de creación de la pantalla se explicarán en apartados siguientes.

Cada prensa envía su estado a la Cabeza de Línea a través de los bits S:99. Lo encontraremos en el programa de las prensas desde #42 en líneas 242 en adelante. Los estados definidos para cada bit son los siguientes:

- S:99/1 → Prensa en autonomía
- S:99/2 → Prensa en parada
- S:99/3 → Prensa en producción
- S:99/4 → Prensa recalentando
- S:99/5 → Prensa en defecto
- S:99/6 → Sanción K
- S:99/7 → Aire no rearmado
- S:99/8 → Falta calidad
- S:99/9 → Validar Sanción K
- S:99/10 → Cambio membrana próximo
- S:99/11 → Prensa no abre
- S:99/12 → Prensa no cierra

Tomamos como ejemplo la prensa C9. Observamos cuáles son los desencadenantes para los distintos bits de estado:

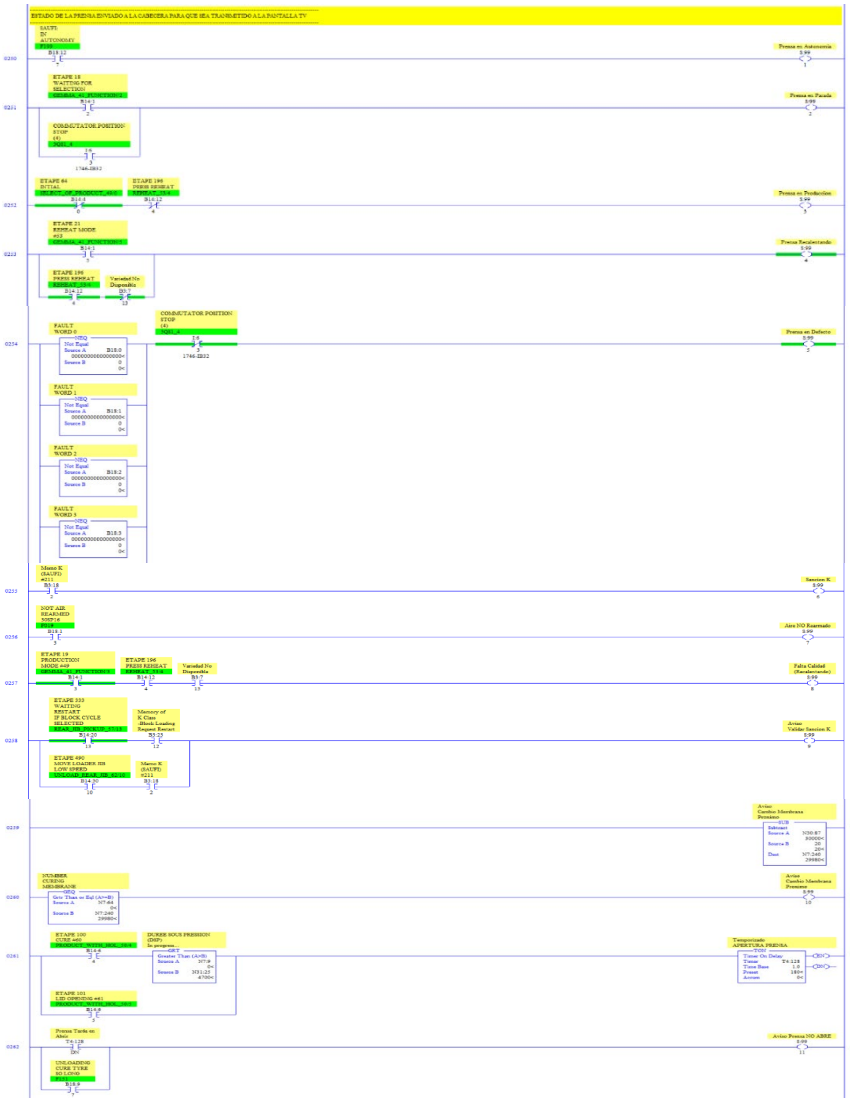


Figura 17. Desencadenantes de defectos en prensa (I)

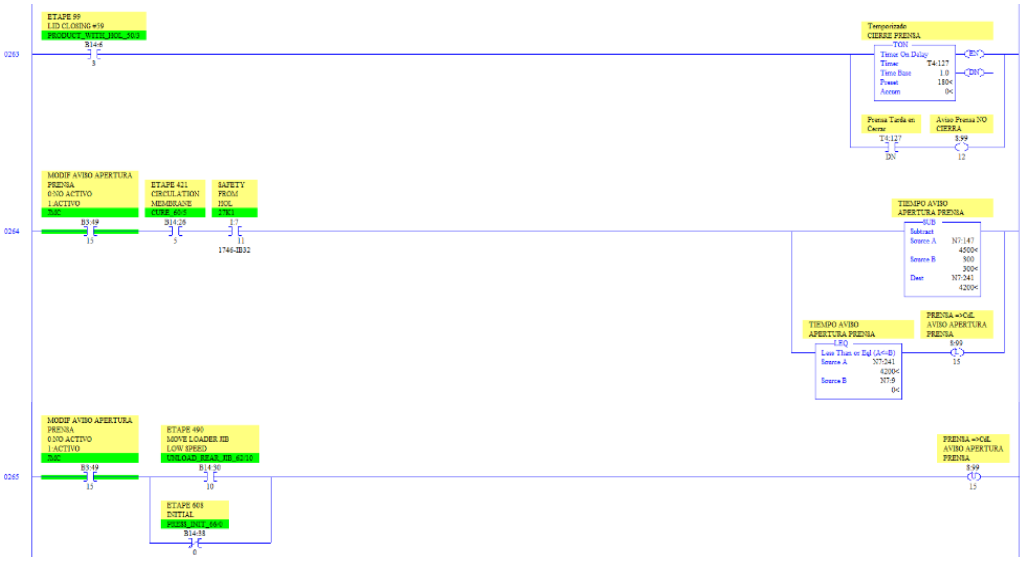


Figura 18. Desencadenantes de defectos en prensa (II)

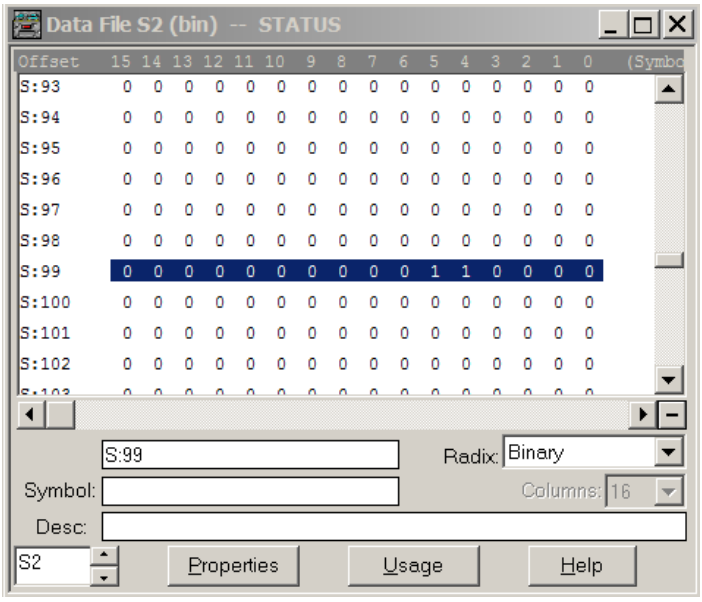


Figura 19. Bits de estado S:99

En este caso los bits activos son el S:99/4 (prensa recalentando) y el S:99/5 (prensa en defecto).

La cabeza de línea envía su estado y el de las prensas al Gateway de MRA para que el PC de MRA tenga acceso a estos cambios mediante el programa SAF2.

A continuación, vemos que, en el programa del autómata del cabecero, el estado de este se almacena en N50:0 y, a partir de N50:1, el estado de las 16 prensas:

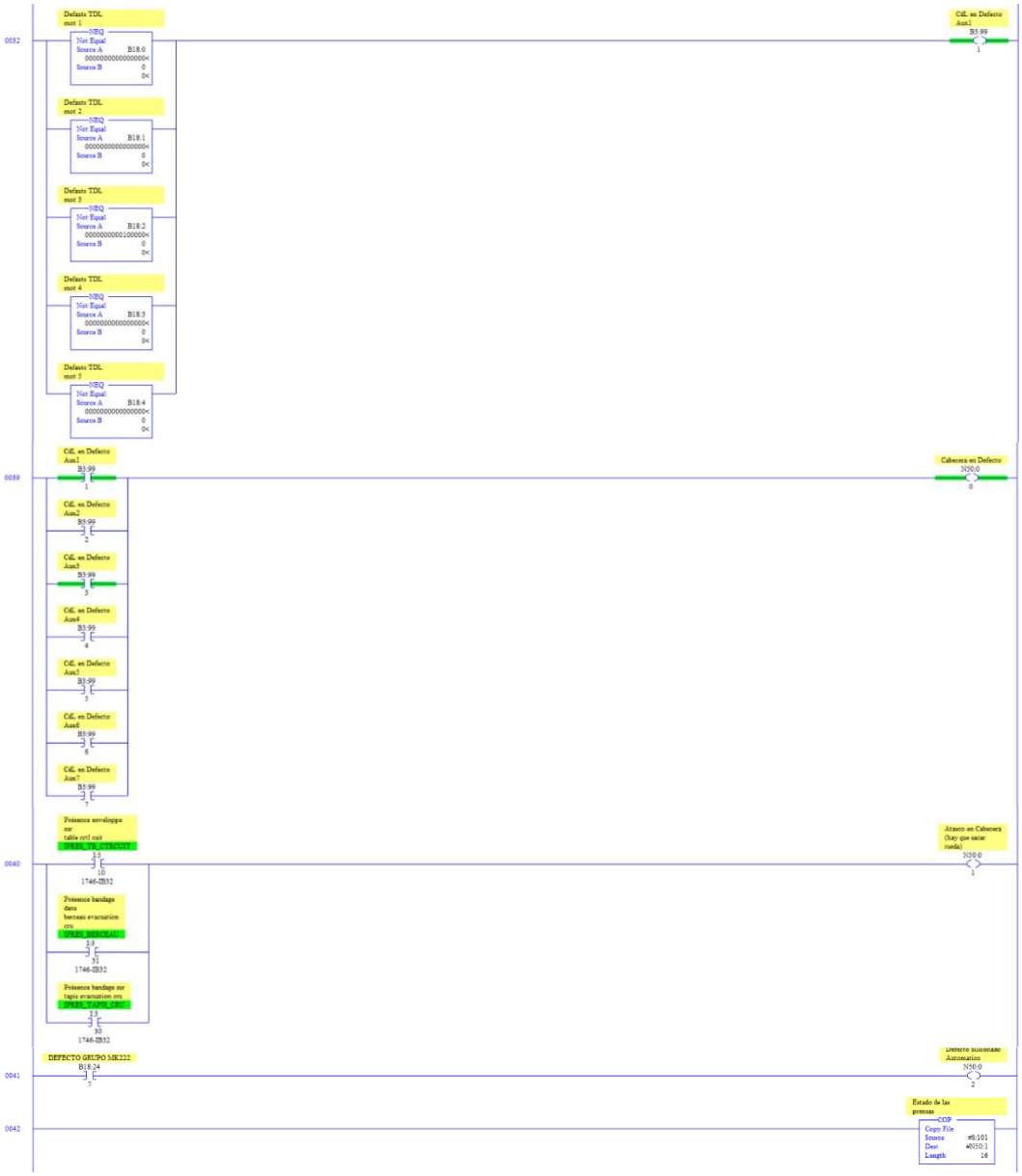


Figura 20. Condiciones necesarias para defecto y atasco en CdL

En la Figura 15, además de ver cuáles son las condiciones que desencadenan la activación de los bits de defecto y atasco en el cabecero de línea (defecto siliconado automático no se usa), tenemos la instrucción COP que nos permite copiar los 16 estados de las prensas (los cuales por medio de una instrucción MSG de tipo lectura hemos guardado en S:101) en N50:1 hasta N50:16.

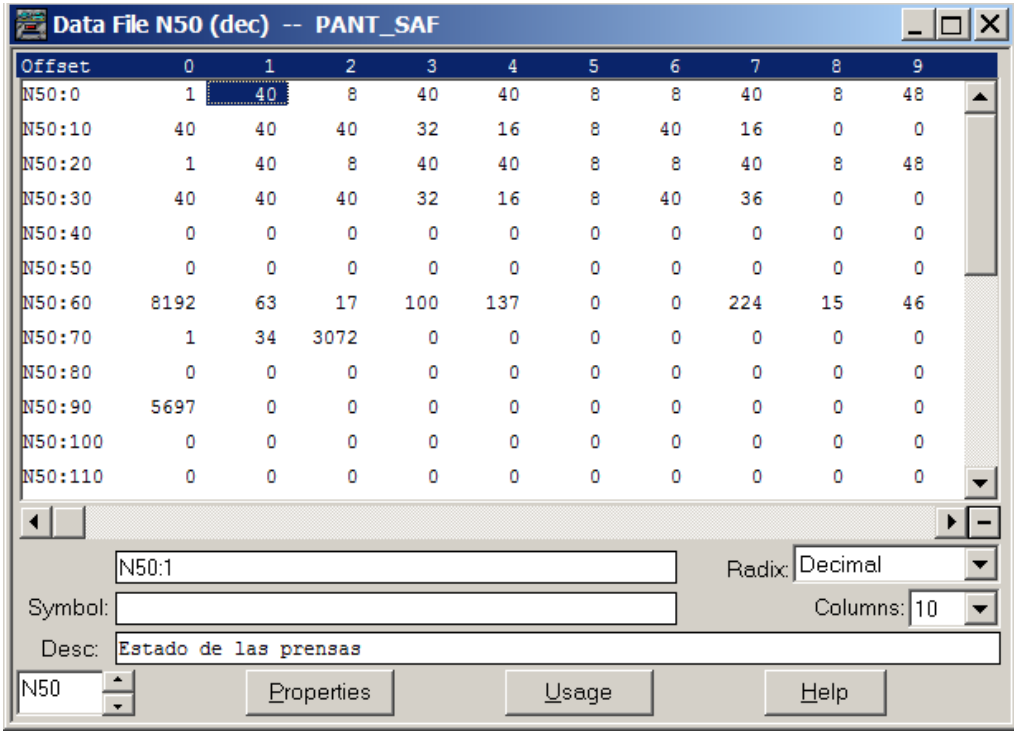


Figura 21. Variables enteras (N:50) donde se almacena el estado del cabecero y de las prensas dentro del programa del cabecero

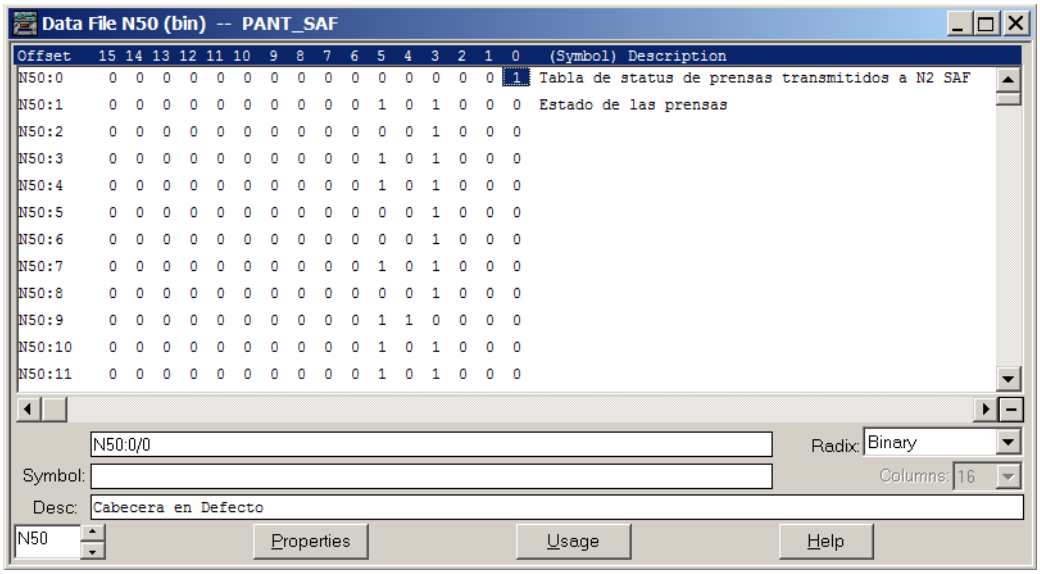


Figura 22. Descomposición en bits 16 de los estados del cabecero y las prensas

Comprobamos que el estado de la prensa 9 es el correcto (bits 4 y 5 activos):

$$2^4 + 2^5 = 48$$

SAF2 recoge los cambios en los estados y se lo envía al ordenador de la pantalla de cocción. En concreto se lo envía a un programa que se llama PAM que es el encargado de mostrarlo.

Como anteriormente el envío del estado de las prensas a SAF se hacía desde el almacén de animación (PROC_ANI), el envío a OSISOFT se hará en este mismo programa.

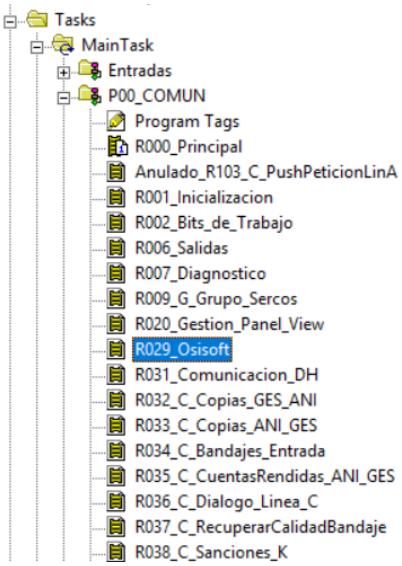


Figura 23. Ubicación de la rutina R029_Osisoft dentro del programa PROC_ANI

Se utiliza la rutina R029_OSISOFT.

Con la modificación de este programa se va a tratar de lograr dos objetivos:

1) Lectura constante de los estados tanto del cabecero como de las prensas, información que leeremos del programa del cabecero.

En primer lugar, hay que crear las variables donde queremos almacenar toda la información que más tarde se va a extraer desde PI Server (OSISOFT), en este caso se va a tratar de un array de 100 enteros:

Nombre Variable	Tipo Dato	
Exch_OSISOFT_INT[0]	INT	Estado Cabecera Línea C
Exch_OSISOFT_INT[1]	INT	Estado Prensa C1
Exch_OSISOFT_INT[2]	INT	Estado Prensa C2
Exch_OSISOFT_INT[3]	INT	Estado Prensa C3
Exch_OSISOFT_INT[4]	INT	Estado Prensa C4
Exch_OSISOFT_INT[5]	INT	Estado Prensa C5
Exch_OSISOFT_INT[6]	INT	Estado Prensa C6
Exch_OSISOFT_INT[7]	INT	Estado Prensa C7

Exch_OSISOFT_INT[8]	INT	Estado Prensa C8
Exch_OSISOFT_INT[9]	INT	Estado Prensa C9
Exch_OSISOFT_INT[10]	INT	Estado Prensa C10
Exch_OSISOFT_INT[11]	INT	Estado Prensa C11
Exch_OSISOFT_INT[12]	INT	Estado Prensa C12
Exch_OSISOFT_INT[13]	INT	Estado Prensa C13
Exch_OSISOFT_INT[14]	INT	Estado Prensa C14
Exch_OSISOFT_INT[15]	INT	Estado Prensa C15
Exch_OSISOFT_INT[16]	INT	Estado Prensa C16
Exch_OSISOFT_INT[20]	INT	Defecto 1 prensa (mostrar por pantalla)
Exch_OSISOFT_INT[21]	INT	Defecto 2 prensa (mostrar por pantalla)
Exch_OSISOFT_INT[99]	INT	Error lectura Mensaje Estado

Tabla 7. Variables OSISOFT para la línea C de cocción

Veremos cómo se realiza la lectura del mensaje proveniente del programa del cabecero.

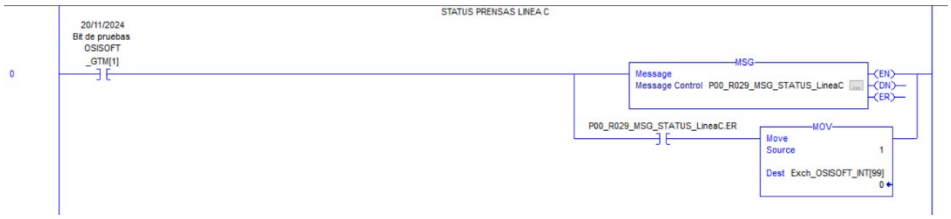


Figura 24. Lectura del estado en programa del almacén de animación

La instrucción es muy sencilla, el programa se va a encontrar constantemente ejecutando la instrucción MSG (siempre que el bit de pruebas esté activo, aunque en un futuro no hará falta y se eliminará). En caso de ocurrir un error de comunicación se escribirá el valor entero 1 en la posición 99 del array de enteros que hemos creado “Exch_OSISOFT_INT”.

Debemos configurar la instrucción MSG como un mensaje de tipo lectura:

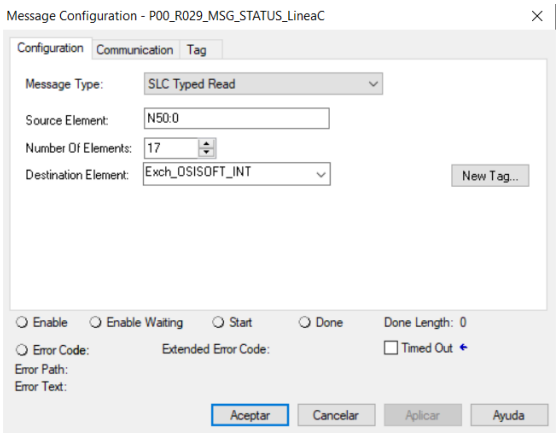


Figura 25. Configuración instrucción MSG para lectura de estados

Se leerán 17 elementos (16 para cada una de las prensas y otro para el cabecero) almacenados en el elemento N50:0 del autómata del que se realiza la lectura y se almacenan en el array de enteros Exch_OSISOFT_INT, empezando por la posición cero.

Ya sólo queda configurar la comunicación con el autómata del cabecero:

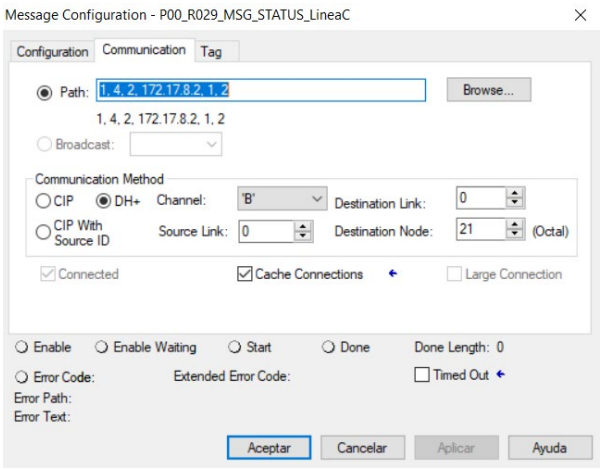


Figura 26. Configuración de la comunicación en la instrucción MSG de lectura de los estados

La comunicación se realiza a través del protocolo de red industrial DH+ (Data Highway), en el canal B de esta red y el nodo del dispositivo de destino en la red es 21 (en octal). También debemos especificar la ruta de comunicación (Path). Toda esta información la podemos obtener conectándonos al autómata del cabecero.

2) Se almacenará el número de dos prensas que en el momento contengan algún tipo de defecto. Cada 3 segundos esos valores cambiarán ordenadamente mostrando las dos siguientes prensas con defecto. Esto servirá más adelante a la hora de visualizar el tipo de defecto de las prensas de dos en dos en la pantalla:

Figura 27. Ejemplo de representación de dos prensas con defecto

Los tipos de defectos que se desean visualizar son defecto (valor en decimal $2^5 = 32$), sanción K ($2^6 = 64$), aire no rearmado ($2^7 = 128$), validar sanción K ($2^9 = 512$), prensa no abre ($2^{11} = 2048$) y prensa no cierra ($2^{12} = 4096$).

Como vemos, sólo nos van a interesar aquellas prensas cuyo valor de estado en decimal sea igual o superior a 32.

Se creará una pila de datos de tipo entero (Defectos_LinC) donde se almacenen aquellas prensas cuyo valor decimar supere 32 (si el cabecero se encuentra en defecto también se informará de ello):

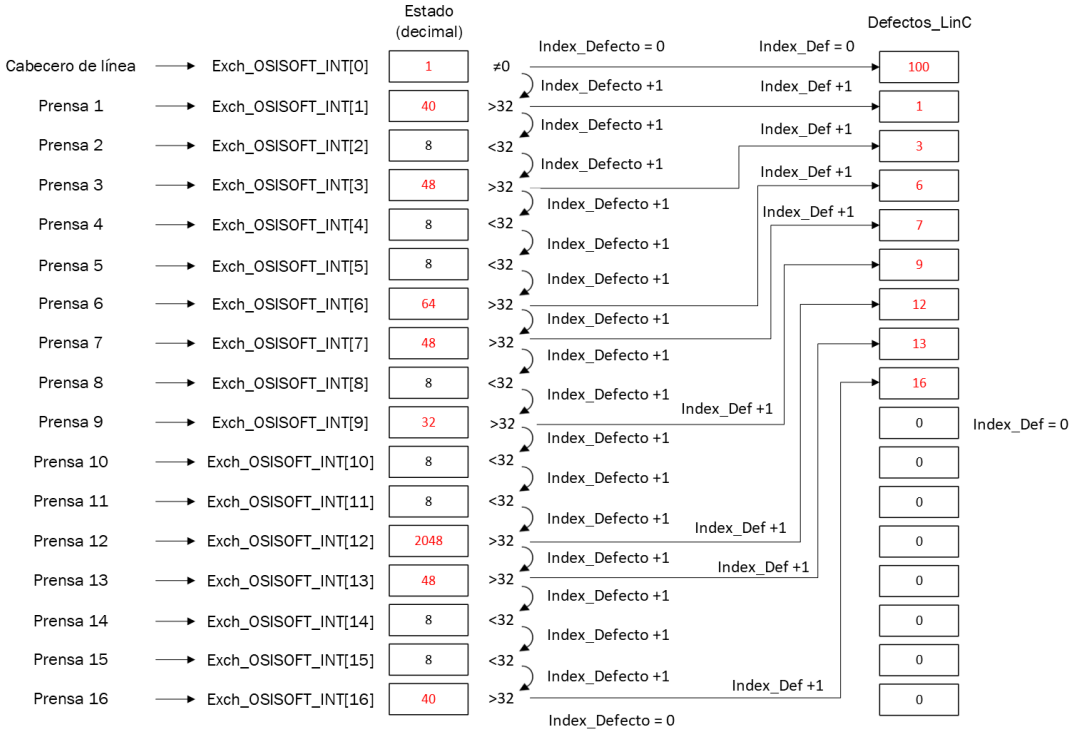


Figura 28. Esquema ejemplo del funcionamiento de la pila de defectos

Para recorrer los estados del cabecero y de las prensas de uno en uno se hará uso de un iterador “Index_Defecto”. Cuando el estado de la prensa sea superior a 32 se almacenará en la pila el valor del iterador (coincide con el número de la prensa en la que nos encontramos). Para el caso del cabecero de línea, cuando se encuentre en defecto (≠0) se añadirá el valor 100, esto servirá para diferenciarlo del valor 0, el cual indica que se ha llegado al final de la pila. Se utiliza otro iterador “Index_Def” que indica la posición dentro de la pila, esto nos servirá a la hora de recorrer la pila ordenadamente de dos en dos y poder almacenar esos dos valores en Exch_OSISOFT_INT[20] y Exch_OSISOFT_INT[21].

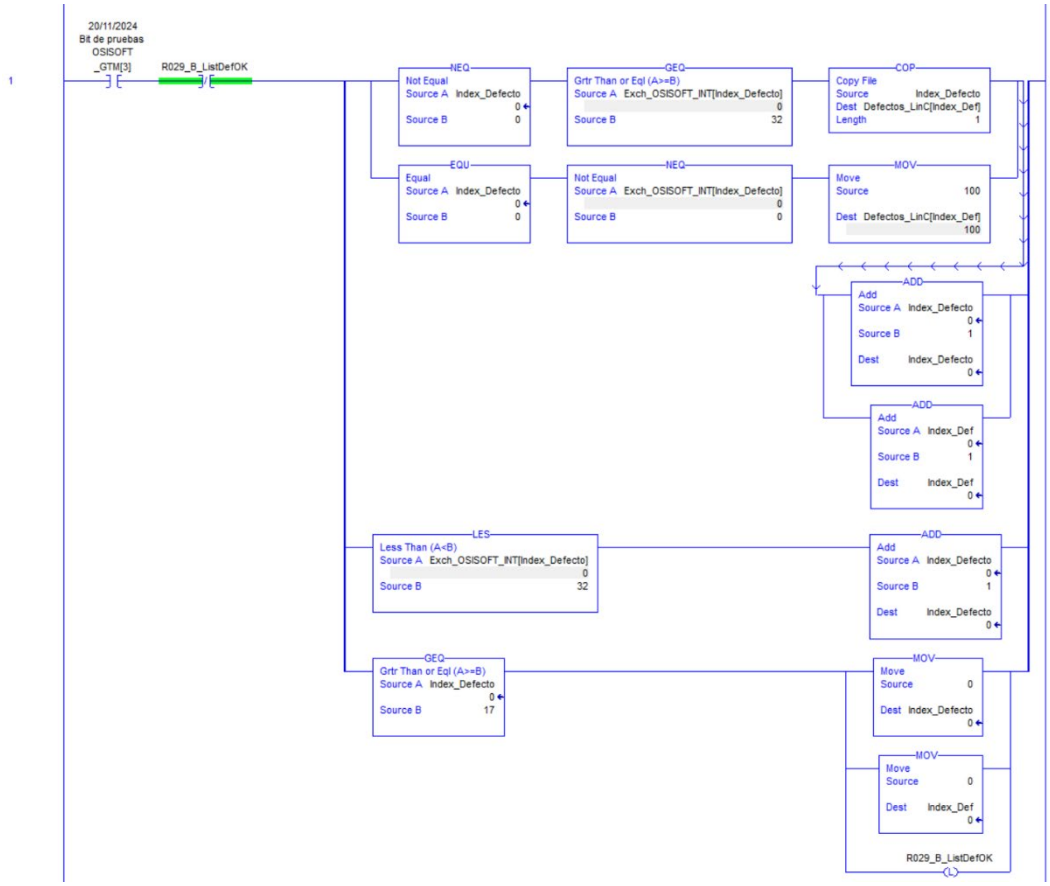


Figura 29. Programación pila de defectos

Cuando se haya analizado el estado del cabecero y las 16 prensas (Index_Defecto ≥ 17) se reiniciarán los iteradores y se activará el bit R029_B_ListDefOK, el cual indica que la lista de defectos ha sido completada. Para evitar que este proceso de búsqueda de defectos se realice de forma constante se añadirá una instrucción XIO de este bit, cuando esté desactivado se desencadena la búsqueda.

Finalmente, se va a tratar de recorrer la pila de defectos de dos en dos, mostrando cada 3 segundos y de manera ordenada las prensas con algún tipo de defecto:

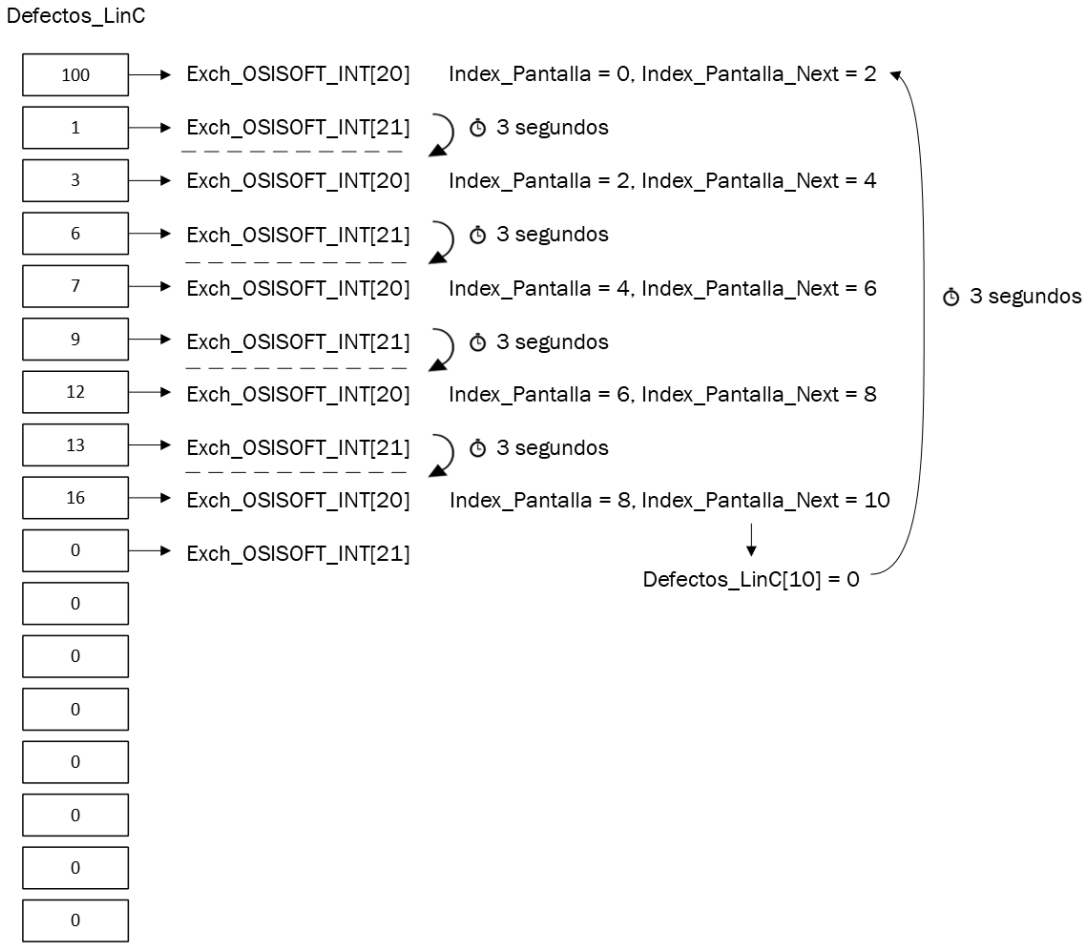


Figura 30. Esquema ejemplo del funcionamiento del muestreo de dos en dos de los defectos

Cuando se ha completado la lista de defectos se inicializa el temporizador (TON) P00_R0029_T_Pantalla_OSISOFT. Cada 3 segundos se copia en Exch_OSISOFT_INT[20] y Exch_OSISOFT_INT[21] dos valores de la lista.

Se utilizan dos iteradores:

- Index_Pantalla: recorre de dos en dos la pila Defectos_LinC.
- Index_Pantalla_Next: indica el valor siguiente del iterador Index_Pantalla, de esta forma se volverá al principio de la pila sin llegar a mostrar dos valores vacíos en el siguiente muestreo.

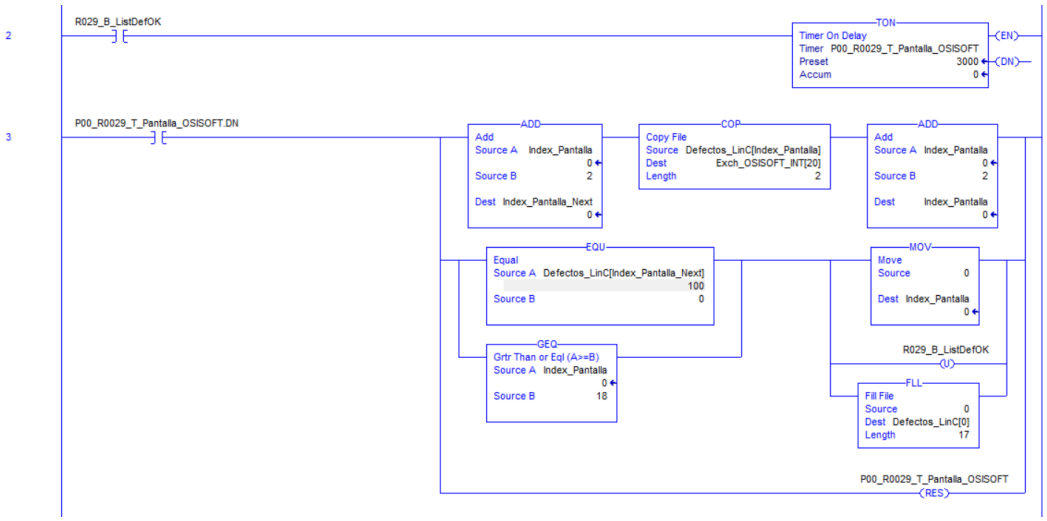


Figura 31. Programación muestreo de defectos de dos en dos

Una vez se ha llegado al final de la pila, se vacía la pila de defectos y se restablece el bit R029_B_ListDefOK que desencadenaba la búsqueda de los defectos.

Con esto sólo informamos de las prensas que se encuentran con alguno de los defectos mencionados, será desde PI System donde se informará del tipo de defecto que contiene la prensa en ese momento (se verá más adelante).

5.1.2. CALIDADES

Además del estado de las prensas, en la pantalla se informa de las calidades cuyo almacenaje está por debajo de un umbral (3 en este caso). Hasta ahora, la salida por pantalla de esta información estaba gestionada por SAF (Sistema de Alertas de Fabricación), para sustituirla por OSISOFT será necesario realizar las debidas modificaciones en el programa del autómata.

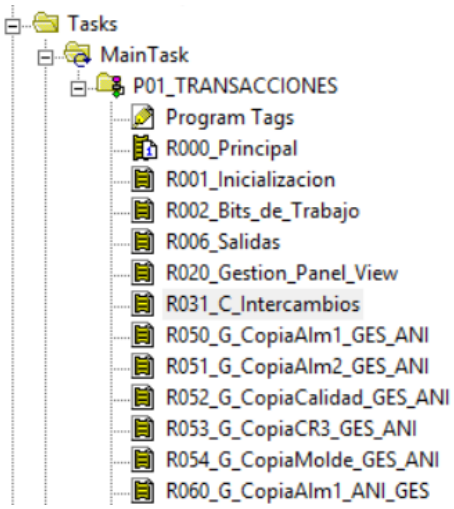


Figura 32. Ubicación de la rutina R031_C_Intercambios dentro del programa PROC_GES

El envío del mensaje a SAF lo encontramos en la rutina R031_C_Intercambios, es ahí donde se implementarán las modificaciones necesarias para el envío de la información a OSISOFT.

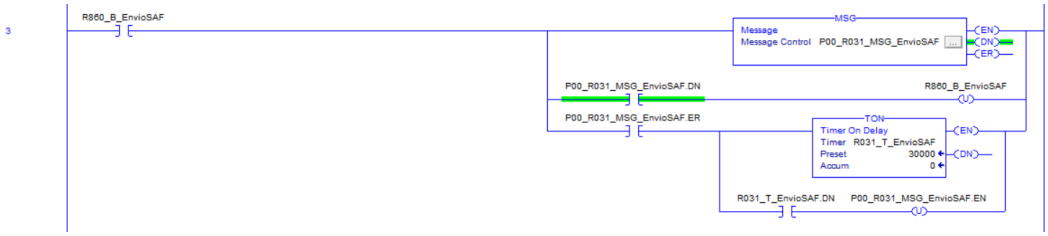


Figura 33. Programación envío a SAF

Cuando se activa el bit de envío a SAF (R860_B_EnvioSAF) se ejecuta la instrucción de escritura del mensaje.

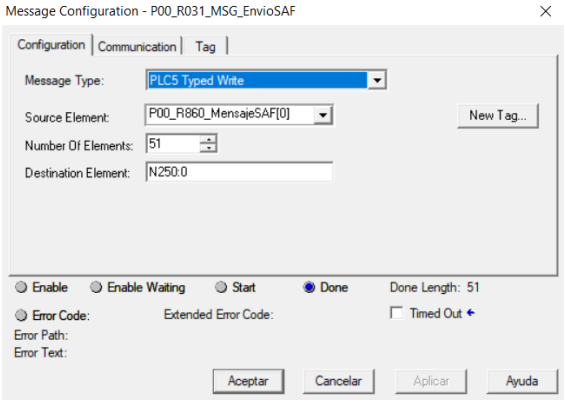


Figura 34.Configuración instrucción MSG tipo escritura para envío a SAF

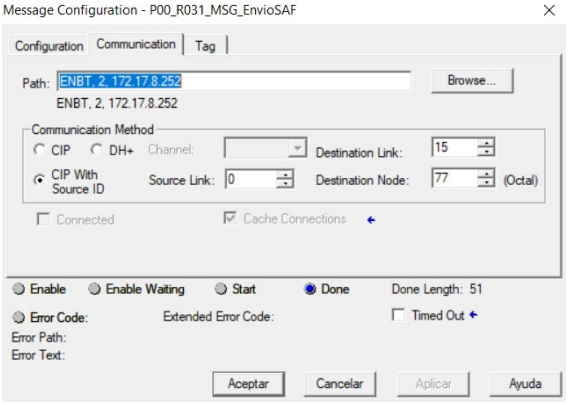


Figura 35. Configuración de la comunicación en la instrucción MSG de escritura

Los 51 datos almacenados en P00_R860_MensajeSAF se escriben en N250:0 en la ruta de comunicación especificada.

Cuando finaliza la escritura del mensaje se reestablece el bit de envío del mensaje. En caso de existir un error en la escritura del mensaje, se inicializará un temporizador de 30 segundos que al finalizar reiniciará la instrucción mensaje.

Anteriormente la visualización por pantalla era gestionada por SAF, ahora será el autómata del almacén de gestión. El objetivo es mostrar aquellas calidades y el número de ruedas para dicha calidad en almacén, siempre y cuando este valor sea igual o inferior a 3. Se irán mostrando una a una cada 3 segundos.

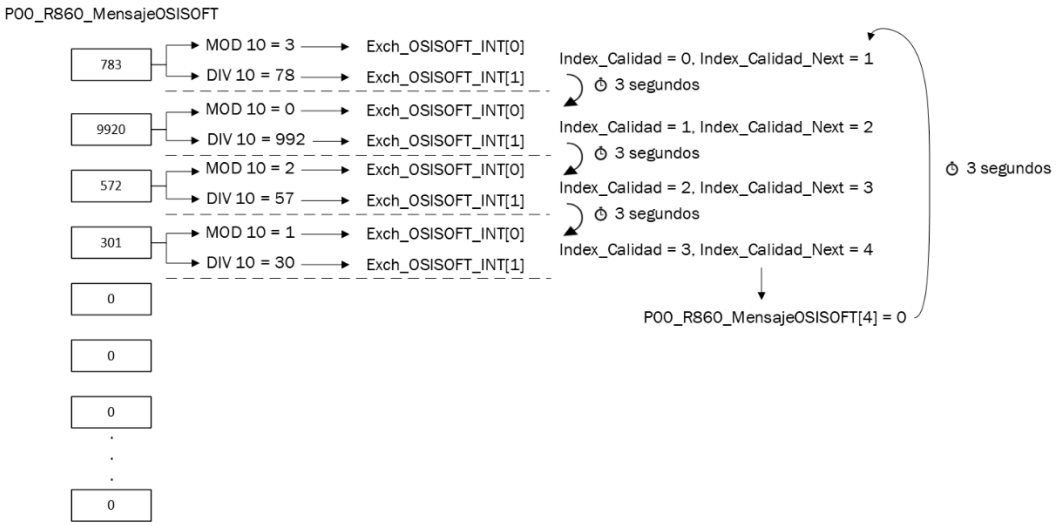


Figura 36. Esquema ejemplo del funcionamiento del programa para muestreo de las calidades y unidades en almacén

Cada 3 segundos se incrementa una posición del array de enteros (P00_R860_MensajeOSISOFT), el valor almacenado en dicha posición se descompone en dos partes:

- Número de ruedas en almacén: por medio de la función MOD se obtiene el resto de la división entre 10, obteniendo como resultado las unidades. Dicho valor se escribe en la variable de enteros Exch_OSISOFT_INT[0].
- Calidad: se divide el valor correspondiente entre 10, de forma que eliminamos las unidades. El resultado se escribe en Exch_OSISOFT_INT[1].

Cuando se llega al final del array se vuelve a comenzar el muestreo desde el primer valor. Para recorrer el array se utiliza el iterador Index_Calidad, además, también contamos con otro iterador Index_Calidad_Next que nos permite saber si el siguiente valor es un cero y así poder volver al inicio sin tener que recorrer el resto de valores nulos del mismo.

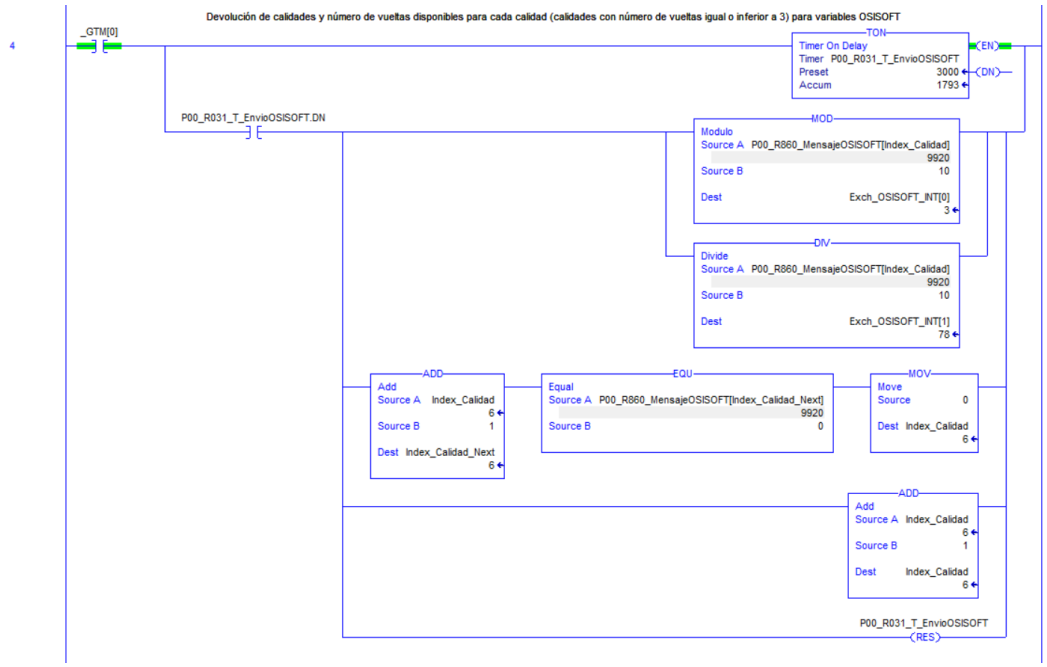


Figura 37. Programación muestreo de las calidades y unidades en almacén

A continuación, se va a explicar detalladamente cómo se obtienen los valores de las calidades. La preparación del mensaje la vamos a encontrar en la rutina R860_B1_MensajeSAF.

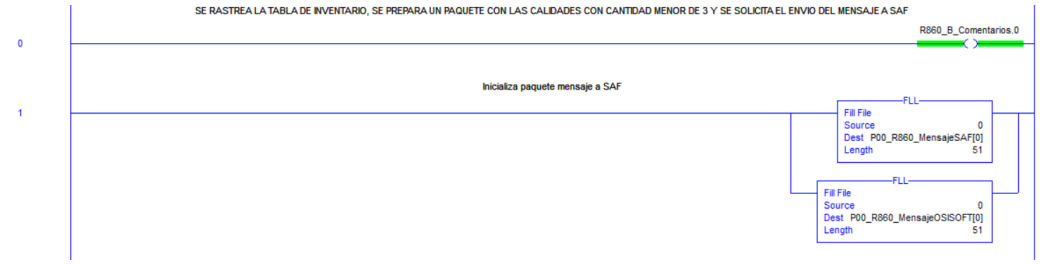


Figura 38. Inicialización paquete mensaje a SAF

En primer lugar, se va a inicializar el paquete de mensaje tanto a SAF como a OSISOFT poniendo todos sus valores a 0 (instrucción FILL).

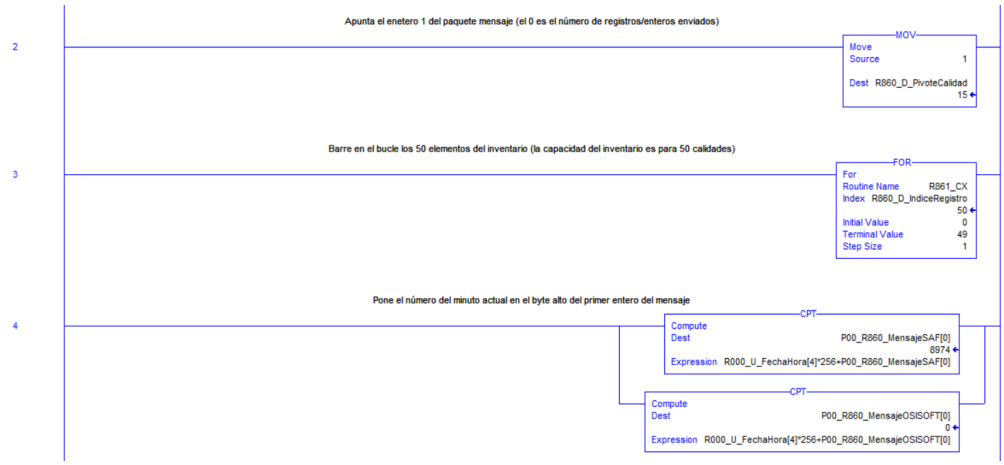


Figura 39. Barride de los elementos del inventario

Se crea un bucle FOR que va a servir para recorrer los 50 elementos de la tabla de inventario en la rutina R861_CX, que es donde obtendremos las calidades con almacenaje igual o inferior a 3. En la primera posición del mensaje (P00_R860_MensajeOSISOFT[0]) se va a escribir la hora actual (servirá para saber la hora de la última actualización sobre el estado del almacén).

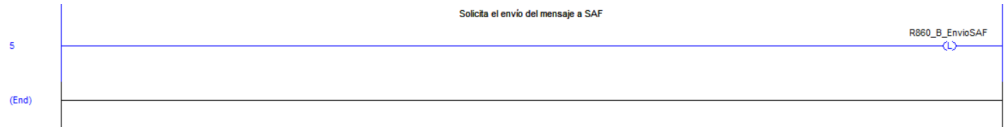


Figura 40. Solicitud envío de mensaje a SAF

Una vez preparado el paquete de mensaje, se activa el bit R860_B_EnvioSAF. No va a ser necesario un bit de envío de mensaje a OSISOFT porque va a estar recogiendo datos constantemente, cuando se llegue al final del paquete de mensaje, se vuelve a mostrar este mismo empezando por el principio mostrando los nuevos valores.

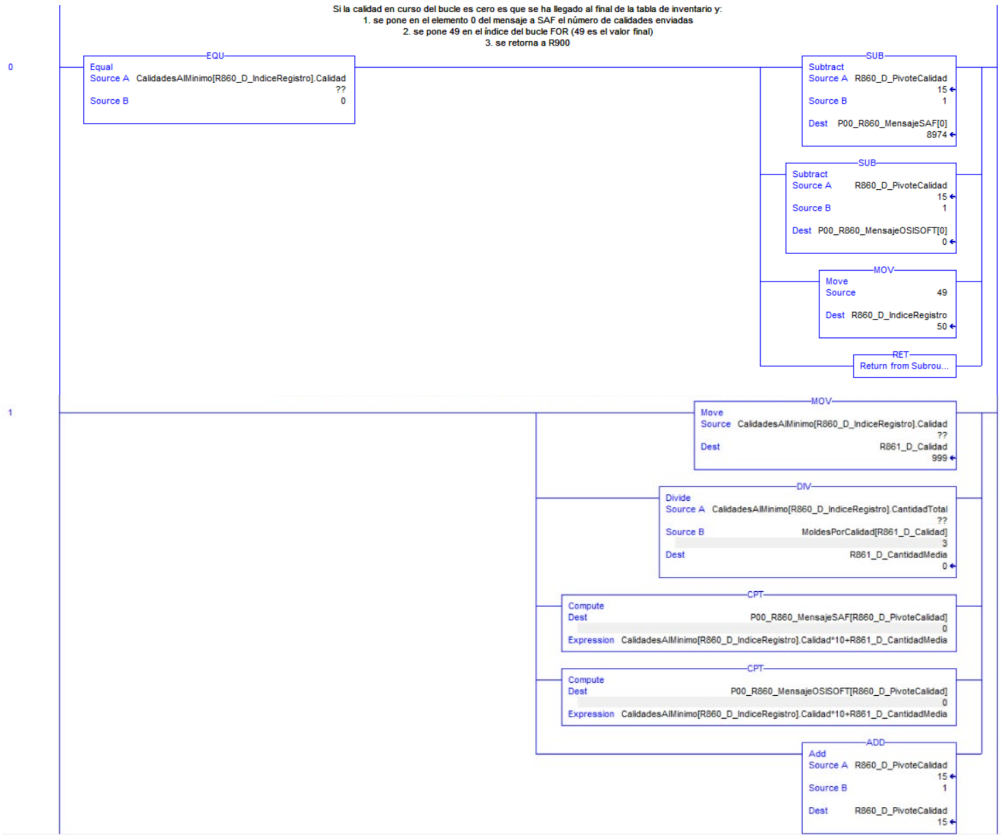


Figura 41. Preparación paquete de mensaje a OSISOFT

Dentro de la rutina R861_CX se recorre el bucle FOR y se escribe en el array de mensaje aquellas cuya cantidad sea menor o igual a 3, cuando se llega al final del bucle se retorna a la rutina 860.

5.2. ALMACÉN SÓTANO

En este apartado se va a tratar de explicar el funcionamiento de la megafonía automática del almacén del sótano y modificaciones implementadas para poder comunicarse con OSISOFT.

En el programa del autómata del almacén del sótano disponemos de una rutina cuya función es avisar por megafonía cuando ocurre un defecto.

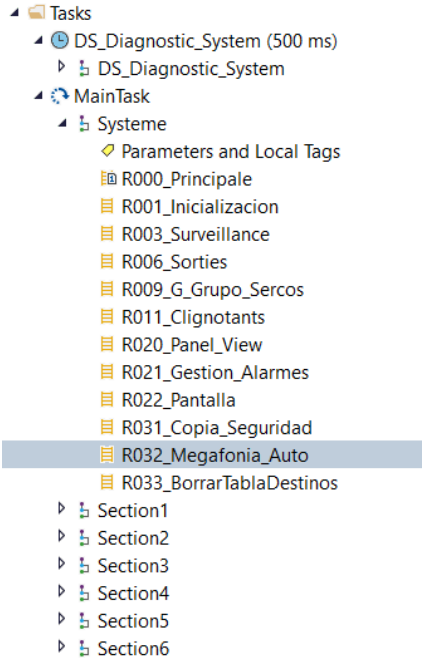


Figura 42. Ubicación de la rutina R032_Megafonia_Auto dentro del programa del Almacén del Sótano

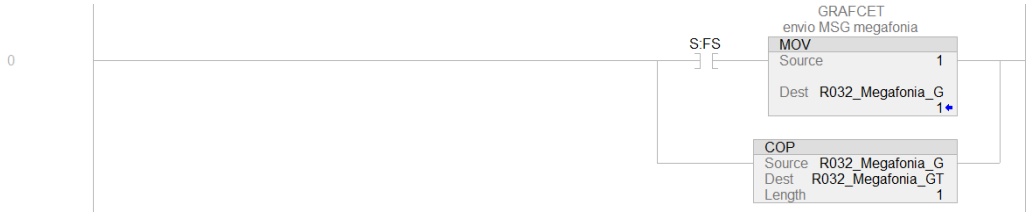


Figura 43. Activación de los avisos a megafonía

Al inicializarse el proyecto (FS, first scan) se escribe un 1 en la variable doble entera (DINT) R032_Megafonia_G para activar los avisos a megafonía. Este valor se copia en R032_Megafonia_GT para inicializar el GRAFCET.

- GRAFCET

Cuando estamos en la etapa 0 del GRAFCET, está activo el bit de enviar mensaje a megafonía (R032_B_MSG_Megafonia) y el bit de anulación de megafonía (1 con megafonía, 0 sin megafonía) o el bit de pruebas para la

eliminación de megafonía _GTM (bit de pruebas "Guillermo Truchuelo Mariscal" creado para probar las modificaciones implementadas), se reseteará el bit 0 de aviso a megafonía (R032_Megafonia_G.0) y se activará el bit R032_Megafonia_G.1, es decir, se pasa a la etapa 1 del GRAFCET.

Figura 44. Etapa 0 del GRAFCET

Cuando estamos en la etapa 1 del GRAFCET, si se ha enviado el mensaje a megafonía o ha transcurrido el tiempo entre envíos de mensajes a OSISOFT (10 segundos), se reiniciará R032_Megafonia_G.1 y se activará R032_Megafonia_G.0 (se vuelve a la etapa 0). En caso de que el mensaje no se haya enviado, se reiniciará R032_Megafonia_G.1 y se activará R032_Megafonia_G.4, se pasa a la siguiente etapa del GRAFCET (los bits 2 y 3 no se utilizan).

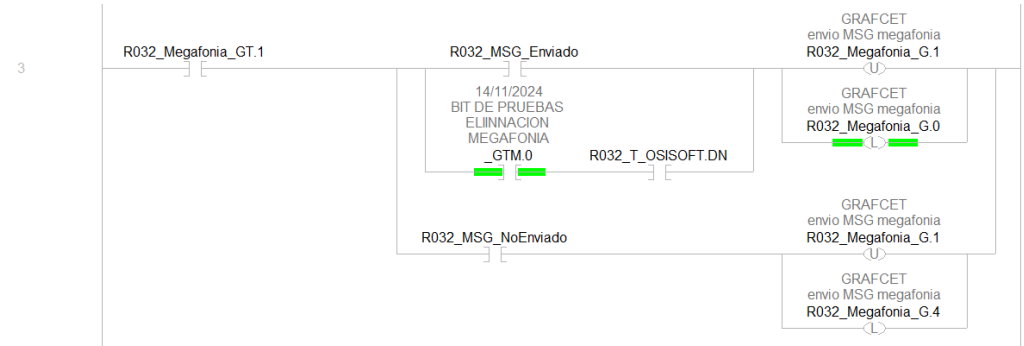


Figura 45. Etapa 1 del GRAFCET

Cuando nos encontramos en la etapa 2 del GRAFCET, se reinicia el bit R032_Megafonia_G.4 y se activa R032_Megafonia_G.1.

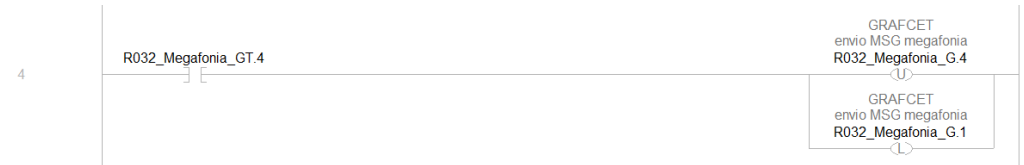


Figura 46. Etapa 2 del GRAFCET

- ACCIONES

Si se activa cualquiera de los defectos o ha trascurrido el tiempo determinado para insistir en el aviso a mantenimiento o a producción, se activará el bit de enviar mensaje.

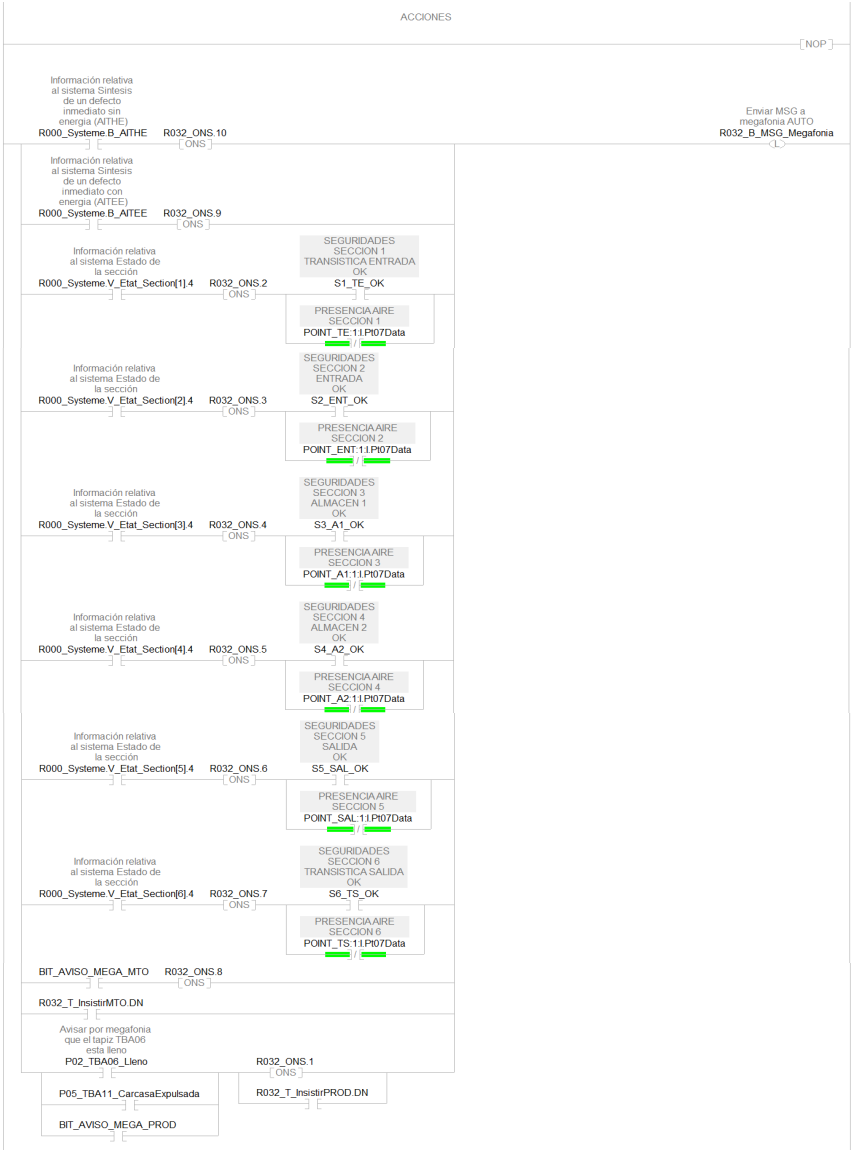


Figura 47. Acciones o desencadenantes de envío de mensaje del almacén del sótano a megafonía automática

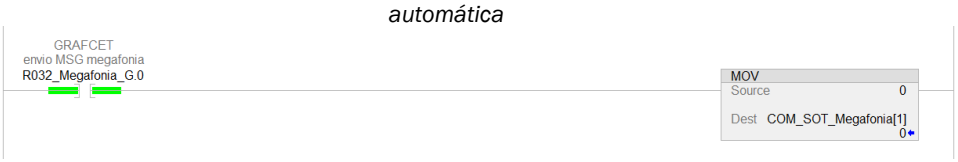


Figura 48. Inicialización de variable entera de para aviso a megafonía

El bit R032_Megafonia_G.0 se utiliza para para poner a 0 el valor de la posición 1 de la variable de enteros COM_SOT_Megafonia. El valor de esta variable va a ser el que se envíe a megafonía.

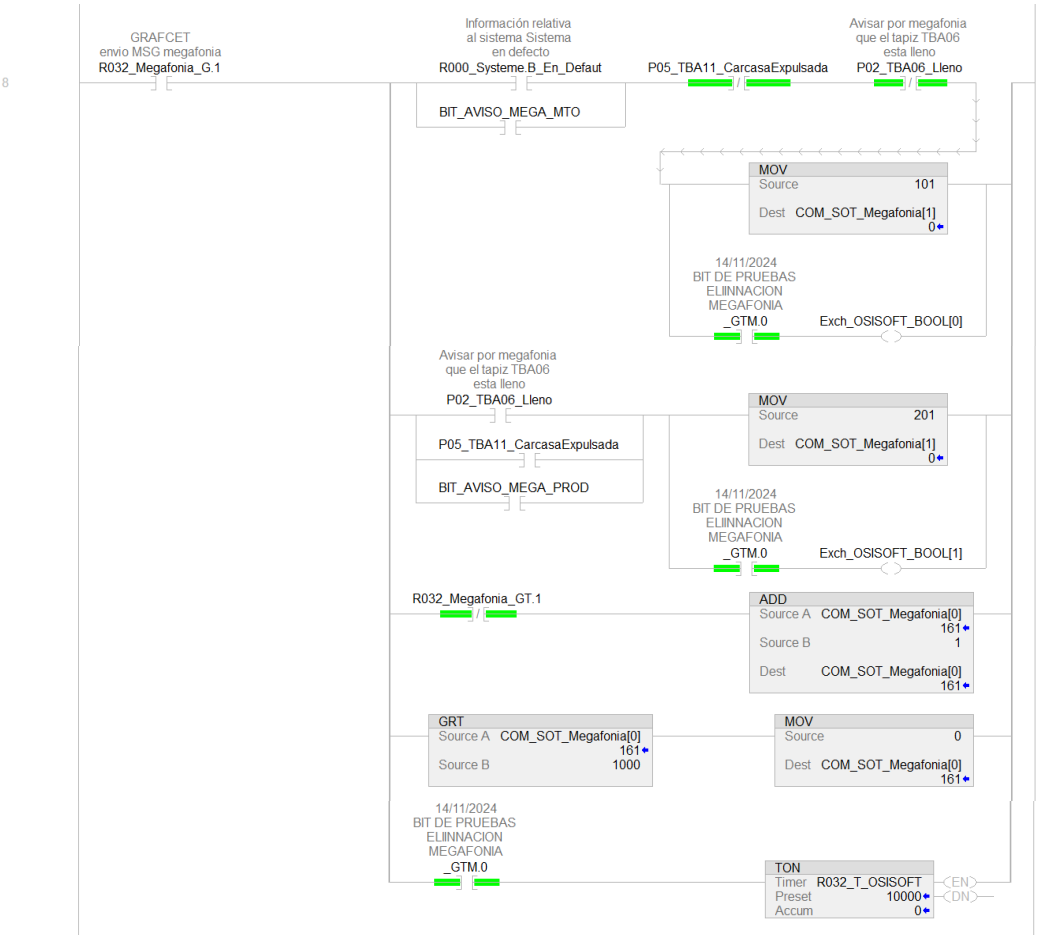


Figura 49. Activación de bits de aviso a mantenimiento y producción

La activación del bit R032_Megafonia_G.1 desencadenará el envío del mensaje. Dependiendo del tipo de defecto detectado se escribirá en COM_SOT_Megafonia[1] el valor entero correspondiente al aviso a mantenimiento (101) o a producción (201). Se ha creado un array de 128 booleanos para avisar a OSISFOT llamado Exch_OSISOFT_BOOL, el bit 0 corresponde a mantenimiento y el 1 a producción. Se declara un tiempo entre mensajes a OSISOFT de 10 segundos.

Cuando se finalice el envío del mensaje saldremos de la etapa 1 del GRAFCET, por lo que el bit R032_Megafonia_GT.1 se desactivará y el valor de COM_SOT_Megafonia[0] se incrementará (se utiliza como un contador de defectos). Si se alcanza el valor 1000 el contador se reseteará.

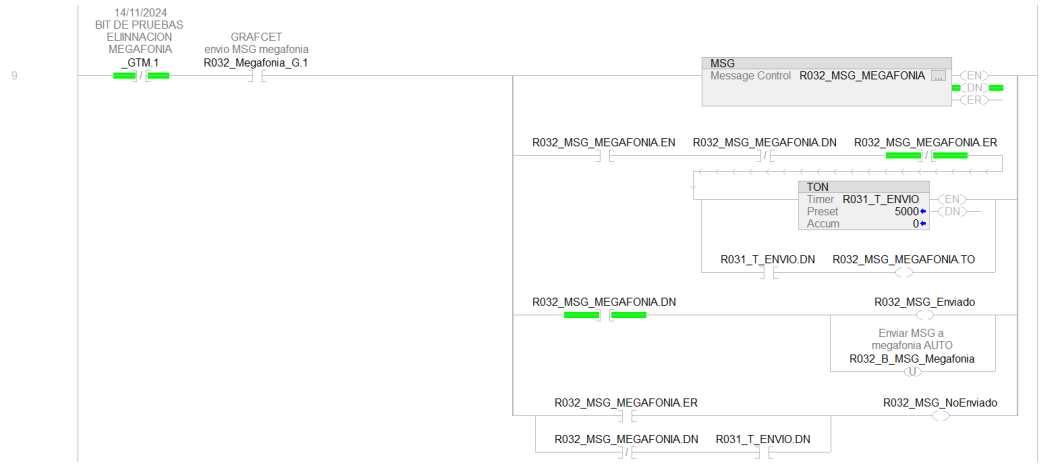


Figura 50. Envío MSG para megafonía

Utilizaremos un bit de pruebas como condición inicial en esta línea ya que tras la eliminación de la megafonía no servirá. En esta línea se realiza la escritura del mensaje de aviso en la dirección correspondiente a megafonía.

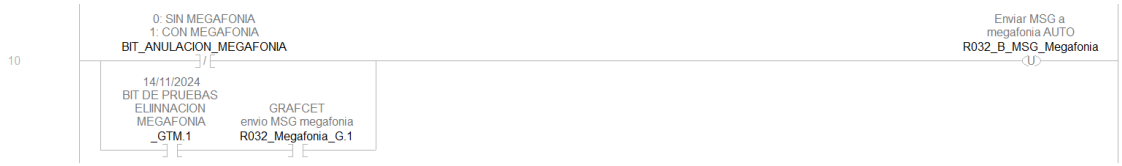


Figura 51. Reinicio del bit enviar MSG a megafonía

Se reinicia el bit de enviar mensaje a megafonía.

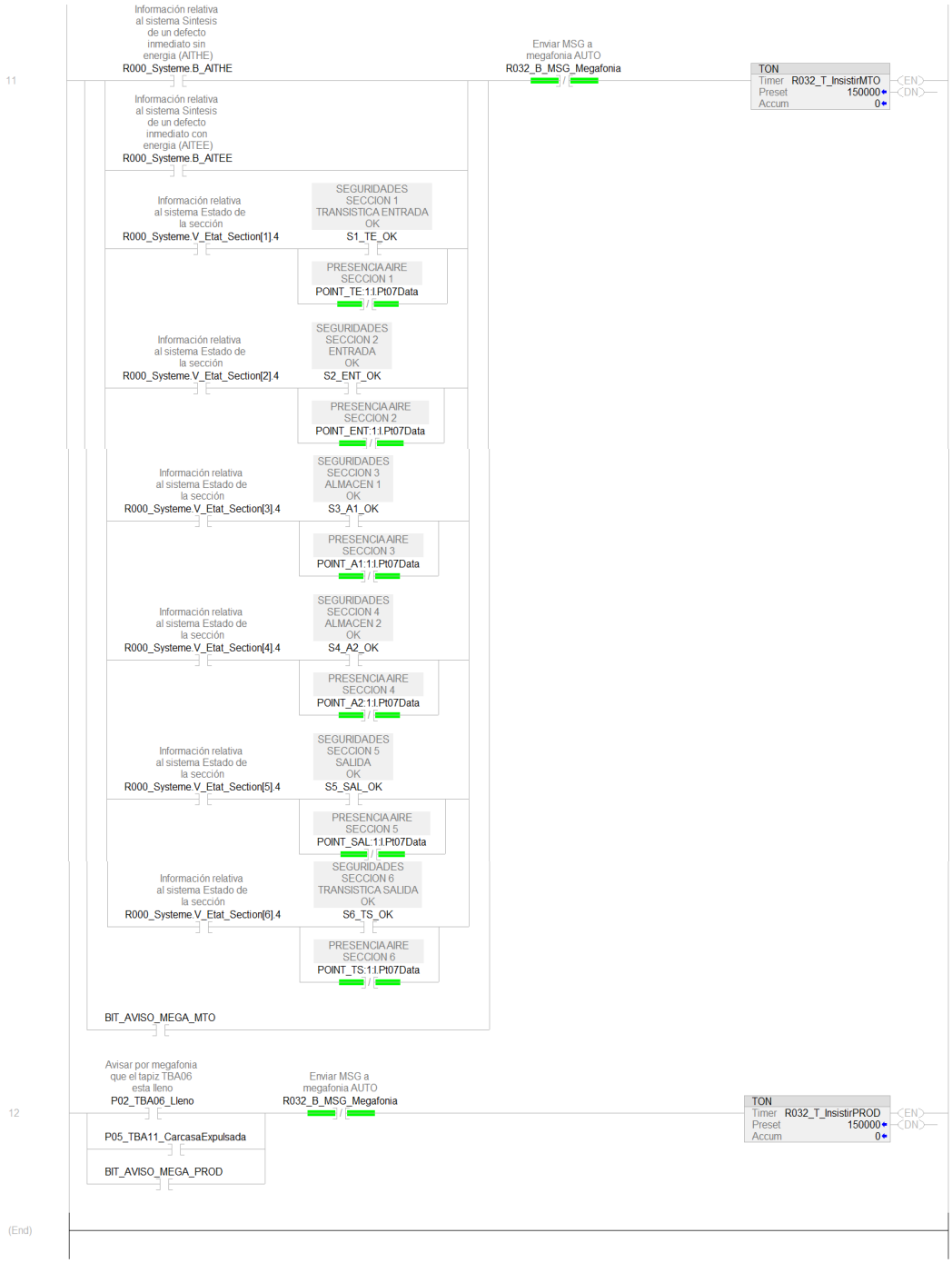


Figura 52. Temporizadores para reenvío de avisos

Si persisten los defectos se volverá a enviar el mensaje de aviso cada 150 segundos tanto para mantenimiento como para producción.

5.3. ALMACÉN DE CEPILLADO

Al igual que en el programa del almacén del sótano, se creará una variable de tipo booleana que se encargará de alertar de los defectos cuando estos se encuentren activos. A continuación, veremos el funcionamiento anterior de la megafonía y las modificaciones realizadas para comunicarnos con OSISOFT.

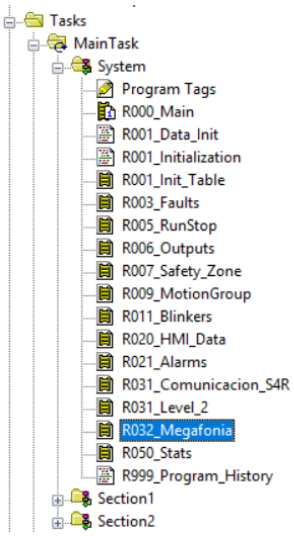


Figura 53. Ubicación de la rutina R032_Megafonia dentro del programa del Almacén de Cepillado

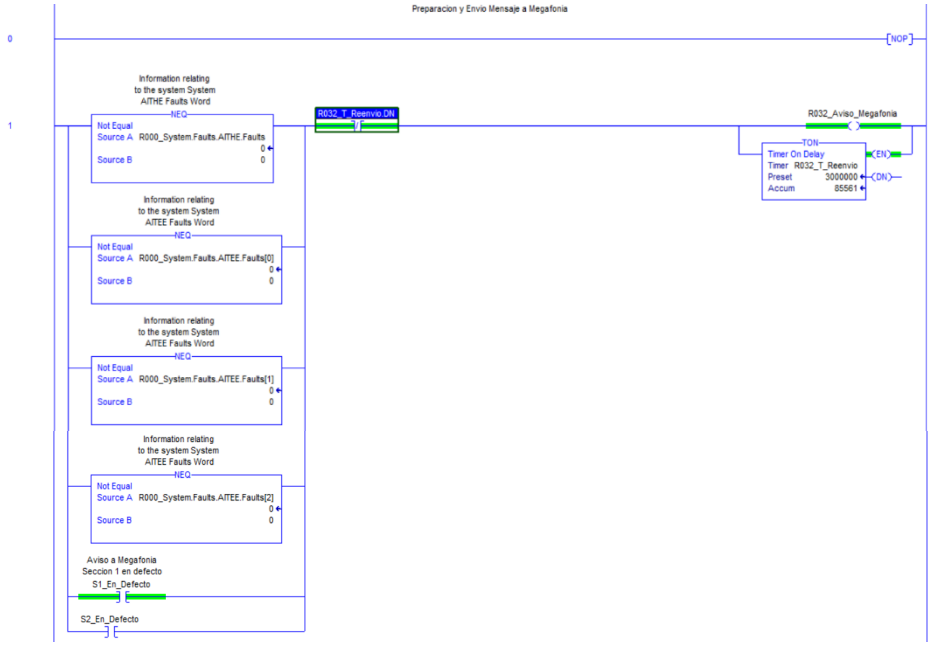


Figura 54. Acciones o desencadenantes de envío de mensaje del almacén de cepillado a megafonía automática

Si se produce un defecto se activa el bit de aviso a megafonía R032_Aviso_Megafonia. Si transcurrido un tiempo continúa activo dicho defecto, se reenvía el aviso.

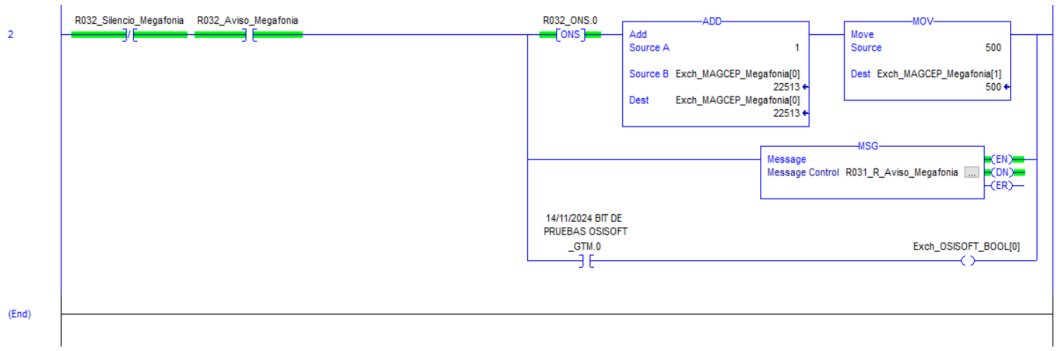


Figura 55. Envío de mensaje y activación del bit de aviso para el almacén de cepillado

Si el silencio de la megafonía está desactivado y se activa el bit de aviso, se envía el mensaje a megafonía a través de la función MSG.

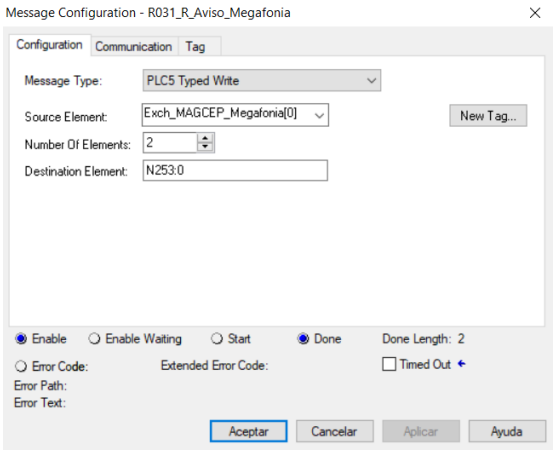


Figura 56. Instrucción MSG tipo escritura del almacén de cepillado

En este caso se va a tratar de una instrucción de tipo escritura (Typed Write), se van a tomar 2 elementos situados en Exch_MAGCEP_Megafonia (en la posición 0 se encuentra el valor del contador de mensaje y en la posición 1 el mensaje).

Para avisar a OSISOFT hemos creado una variable booleana Exch_OSISOFT_BOOL.

5.4. CEPILLADO

Este apartado se dividirá en dos partes: aviso de defectos y sorteo de cepillado.

5.4.1. AVISO DE DEFECTOS CEPILLADO

Para la zona de cepillado se dispone de 2 líneas (B15 y B16) por las que llegan las ruedas a cepillar (existe una tercera línea que se encuentra en desuso).

Los defectos de la CDCN son gestionados por la B15, a continuación, veremos el funcionamiento del programa.

Figura 57. Defectos en Cepillado

Si se produce un defecto (bits a 1) se activará su correspondiente bit de estado de defecto de complejo CDCN enviado a SAF/SAM (N27 SLC a Panel View).

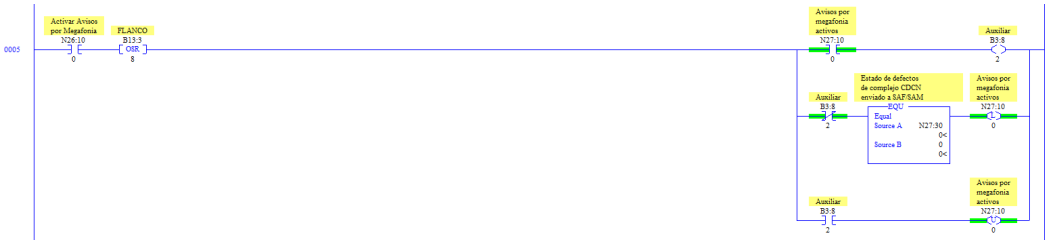


Figura 58. Bit avisos por megafonía activos

Si se produce la activación de los avisos por megafonía (bit 0 de la dirección N26:10 a 1) y el flanco de subida está activo (OSR) se producirán las siguientes acciones:

- Si los avisos por megafonía están activos (bit 0 de la dirección N27:10 a 1) se activa un bit que se va a utilizar como auxiliar.

- Si el bit auxiliar no está activo (avisos por megafonía no están activos) y el estado de defectos de complejo CDCN enviado a SAF/SAM es igual a 0, se activará el set para avisos por megafonía activos.
- Si el bit auxiliar está activo (avisos por megafonía activos) se producirá un reset para avisos por megafonía activos.

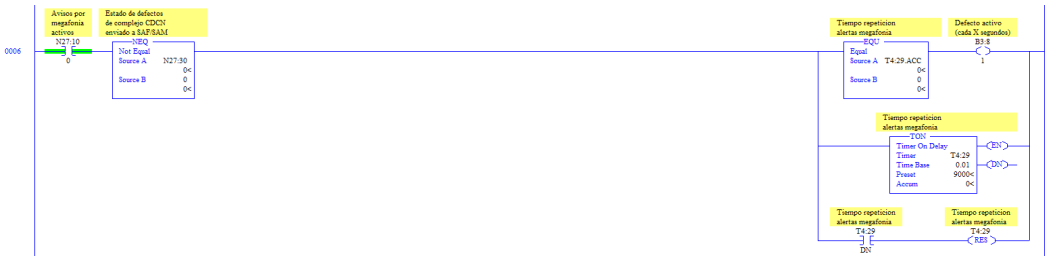


Figura 59. Temporizador para repetición de avisos por megafonía

Si los avisos por megafonía están activos y el estado de defectos de complejo CDCN enviado a SAF/SAM no es igual a 0 se producirán las siguientes acciones:

- Si el valor acumulado (ACC, número de intervalos de base de tiempo que la instrucción ha contado) del temporizador (TON) de alertas megafonía es igual a 0, se activa un bit de defecto (este bit se activará cada X segundos, siendo X el tiempo preseleccionado en el temporizador que veremos a continuación).
- Se ejecuta el temporizador (TON), cuando se alcanza el valor preseleccionado (90 segundos) se activa el bit de efectuado del temporizador (DN).
- Si el bit de efectuado del temporizador (DN) está activo se restablecerá el temporizador (RES).

Figura 60. Activación bit enviar mensaje a megafonía y código de mensaje

Si los avisos por megafonía están activos, hay un defecto activo y se activa el flanco de subida, se enviará un mensaje a megafonía:

- Si el número de envío de mensaje a megafonía es menor que 0, la dirección donde está guardado este valor (entero) se pondrá a 0.
- Si el número de envío de mensaje a megafonía es menor que 32767, se incrementará dicho valor en una unidad.
- Si el número de envío de mensaje a megafonía es igual a 32767, se reiniciará el valor de la dirección, cambiando su valor a 1.
- El código de mensaje de megafonía es 201, se escribirá este valor en la dirección N62:225.
- Se activa el bit para enviar mensaje a megafonía.

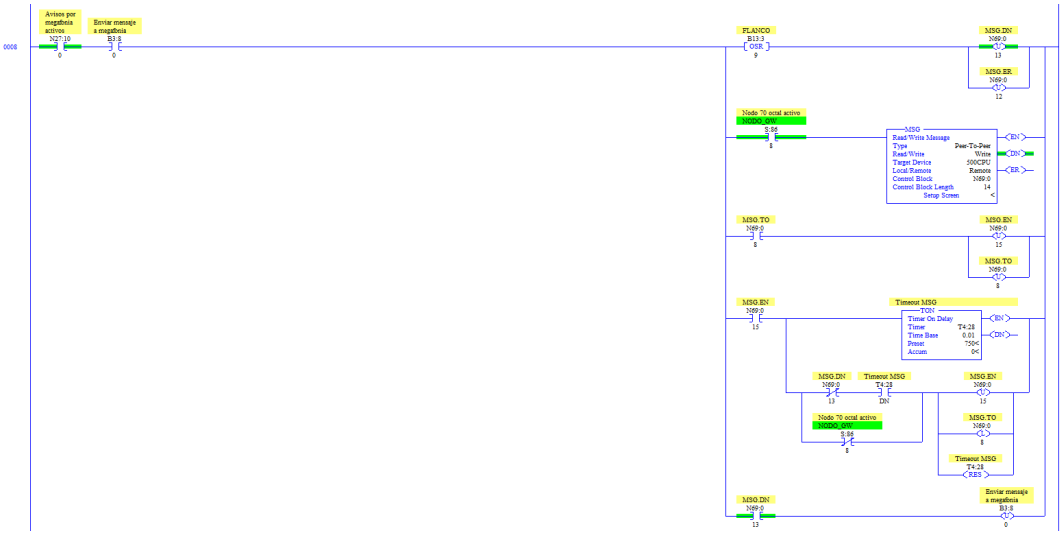


Figura 61. Envío de mensaje a megafonía

Finalmente, se va a leer el mensaje que se va a enviar por megafonía. Si los avisos por megafonía y el bit de enviar mensaje están activos, se producirán las siguientes acciones:

- Si el flanco de subida está activo, se reseteará el bit de efectuado (DN, se establece cuando el mensaje es transmitido correctamente. El bit se restablece la siguiente vez que el renglón asociado va de falso a verdadero) o se reseteará el bit de error (ER, se establece cuando ha fallado la transmisión del mensaje, se restablece la siguiente vez que el renglón asociado va de falso a verdadero).
- Si el nodo 70 octal está activo se ejecutará la instrucción mensaje (MSG). En este caso, se va a tratar de una acción de escritura (Write). En el bloque de control (Control Block) seleccionamos la dirección donde vamos a realizar la escritura (N69:0), se trata de una dirección de archivo entero de 14 palabras que contiene bits de estado,

dirección del archivo receptor y otros datos asociados con la instrucción de mensaje.

Cuando la instrucción de mensaje se está siendo ejecutada se activa el bit de habilitación EN (permanece establecido hasta que la transmisión del mensaje se ha terminado y el renglón se hace falso), cuando el mensaje se ha transmitido correctamente se activa el bit DN, y si la transmisión ha fallado se activará el bit de error, ER.

- Si el bit de tiempo límite está activo (TO*) se restablecerán éste y el bit de habilitación (EN).
- Si el bit de habilitación (EN) está activo se ejecutará un temporizador para establecer un tiempo máximo de escritura del mensaje (Timeout MSG).

  Si el bit de efectuado (DN) no está activo y el temporizador ha alcanzado el tiempo preseleccionado (7.5 segundos, bit DN del temporizador activo), o si el nodo 70 octal no está activo, se restablecerá el bit de habilitación (EN) de la instrucción de mensaje (se detendrá la escritura del mensaje), se establecerá el bit de tiempo límite (TO) y se reiniciará el temporizador (RES).
- Si el bit de efectuado del mensaje está activo (DN), se restablecerá el bit de enviar mensaje (el mensaje ya ha sido enviado correctamente).

5.4.2. SORTEO DE CEPILLADO

La zona de cepillado, como se ha explicado anteriormente, dispone de 2 líneas por las que llegan las ruedas, las cuales son asignadas de manera aleatoria a los operarios según son dispuestas por las líneas disponibles. Se dispone de una pantalla informativa con las líneas y el número del operario asignado.

Al igual que los defectos, la asignación aleatoria del operario es gestionada por el programa de la B15. A continuación se explicará el funcionamiento del programa.

- COMBINATORIA

En primer lugar, se va a tratar de calcular la combinatoria de forma que cada operario sea asignado de forma aleatoria.

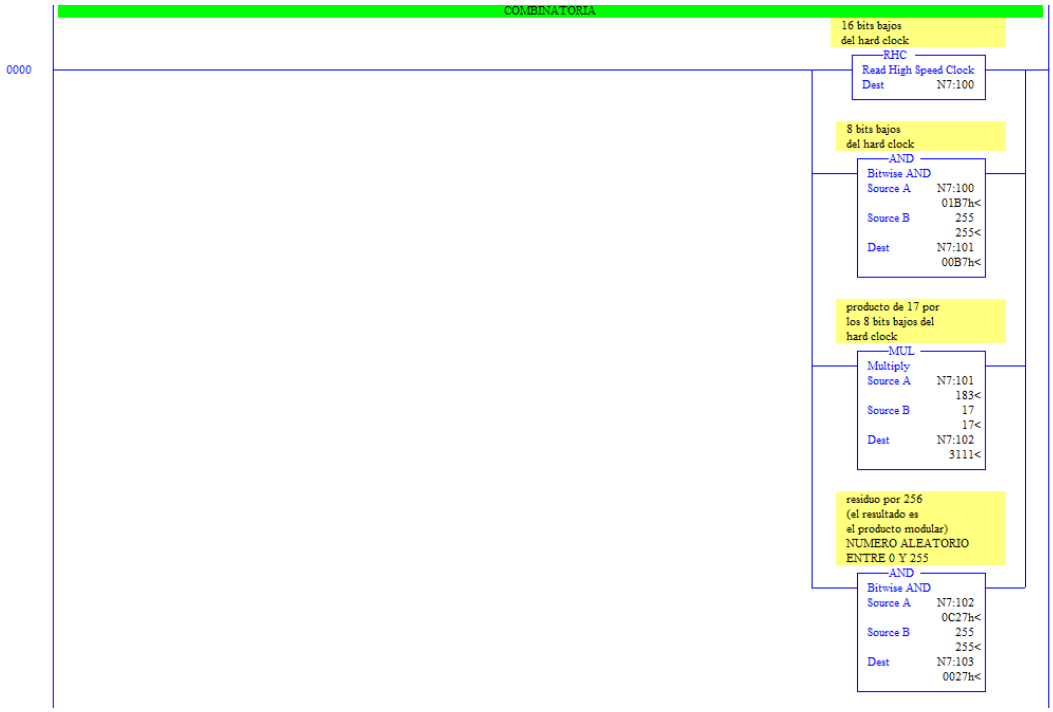


Figura 62. Combinatoria para asignación de operarios por sorteo

Se toman los 16 bits bajos del RHC y se almacenan en la dirección N7:100, se utiliza la instrucción AND para almacenar los 8 bits bajos del hard clock en N7:101 (partiendo de los 16 bits almacenados en N7:100). Estos 8 bits los vamos a multiplicar por 17 (instrucción MUL) y el resultado se va a almacenar en la dirección N7:102. Finalmente se va a obtener el producto modular de este valor utilizando la instrucción AND, obteniendo un número aleatorio entre 0 y 255, dicho valor entero se almacena en N7:103.

RHC (Read High Speed Clock): usado para grabar el tiempo de inicio y final de un evento. SLC 500 dispone de un reloj con valores enteros de 20 bits que se incrementa cada 10 µs. Cuando se activa esta instrucción y la dirección de destino es una dirección de enteros, se almacenan los 16 bits bajos.

El siguiente paso será asignar una cuota de probabilidad a cada línea de trabajo:

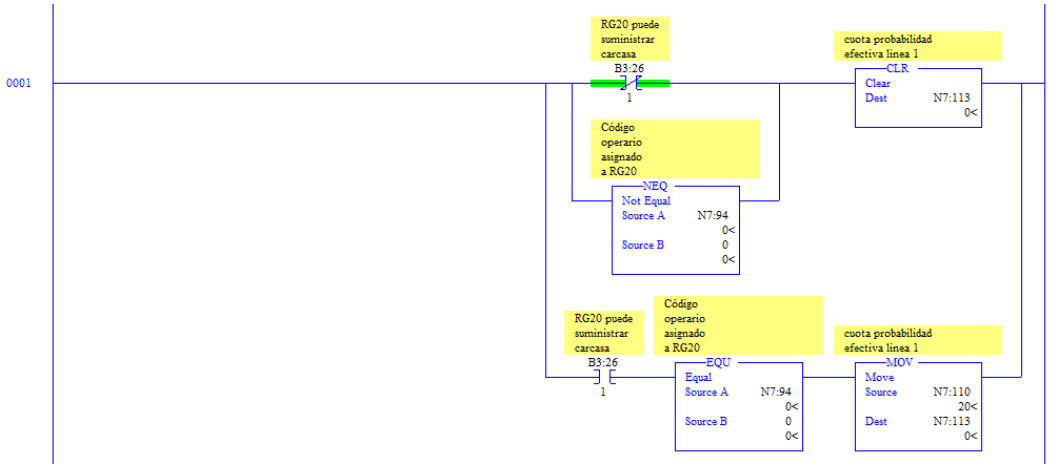


Figura 63. Cuota de probabilidad efectiva para la línea 1

Si la RG20 no puede suministrar carcasa (bit 1 de la dirección B3:26 a 0) o si se ha asignado el código de operario a RG20 (NEQ, código de operario almacenado en N7:94 no es igual a 0), se pone a 0 el valor de la cuota de probabilidad efectiva de la línea 1, almacenado en N7:113.

Si la RG20 puede suministrar carcasa (bit 1 de la dirección B3:26 a 1) y no se ha asignado un código de operario a la RG20 (EQU, código de operario almacenado en N7:94 es igual a 0), se escribe el valor almacenado en N7:110 (cuota de probabilidad asignada a la línea 1) en la dirección N7:113 (cuota de probabilidad efectiva línea 1).

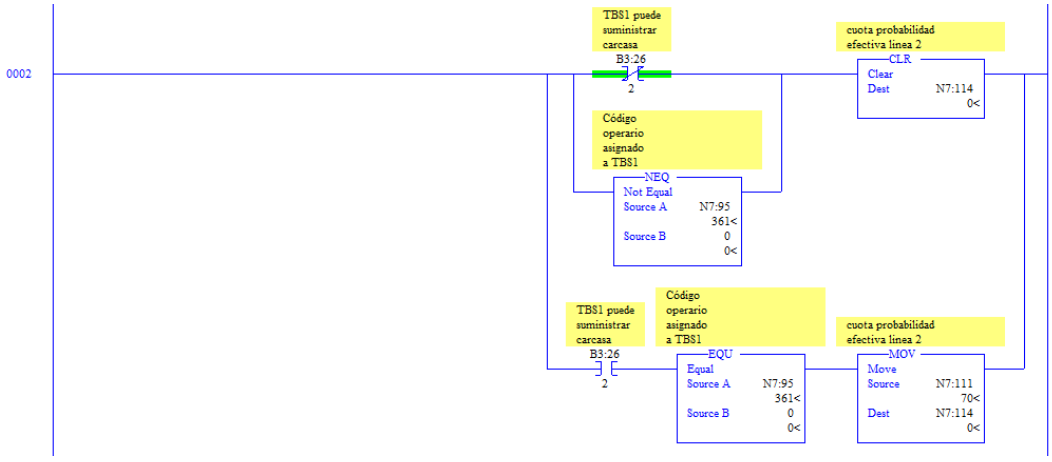


Figura 64. Cuota de probabilidad efectiva para la línea 2

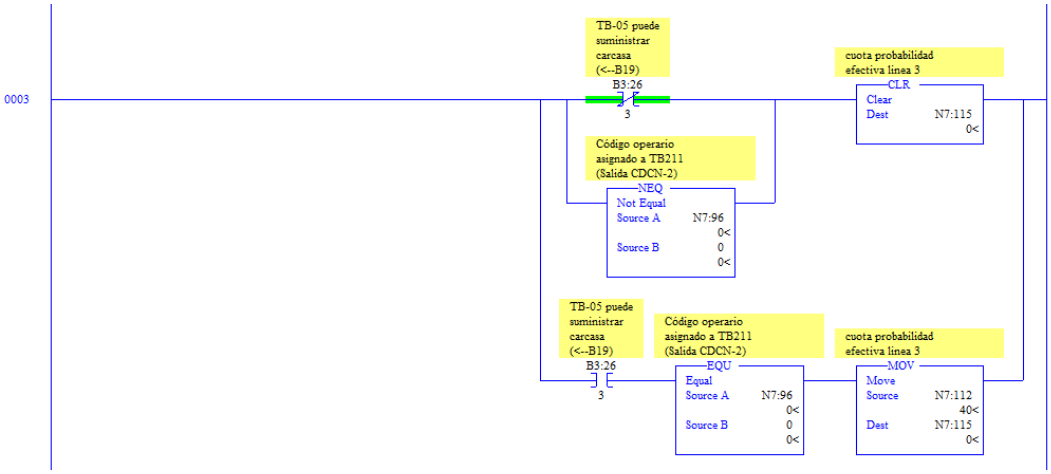


Figura 65. Cuota de probabilidad efectiva para la línea 3

El proceso se repite para las líneas 2 (TBS1) y 3 (TB-05).

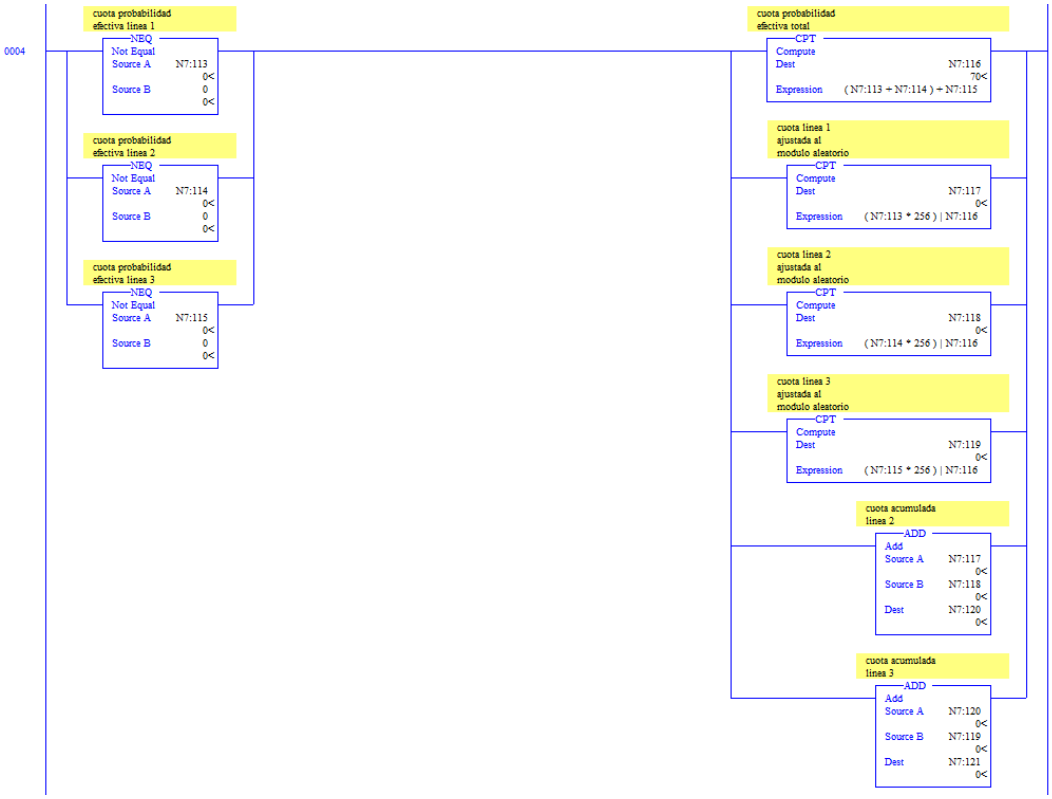


Figura 66. Cuotas de probabilidad acumulada para las 3 líneas

Si se ha asignado una cuota de probabilidad efectiva a cualquiera de las líneas (NEQ, cuota de probabilidad efectiva de la línea no es igual a 0), se pueden calcular las siguientes cuotas de probabilidad: N7:116 = N7:113 + N7:114 + N7:115

- Cuota de probabilidad efectiva total (N7:116): suma de las cuotas de probabilidad efectiva de las 3 líneas (N7:113 + N7:114 + N7:115).

- Cuota de la línea 1 ajustada al módulo aleatorio (N7:117): se multiplica la cuota de probabilidad efectiva de la línea por 256 y se divide entre la total ((N7:114*256)/N7:116).

Mismo procedimiento para las líneas 2 y 3 (N7:118 y N7:119).

- Cuota acumulada línea 2 (N7:120): suma de las cuotas de línea 1 y 2 ajustadas al módulo aleatorio (N7:117 + N7:118).

- Cuota acumulada línea 3 (N7: 121): suma de la cuota de línea 3 ajustada al módulo aleatorio y la acumulada de la línea 2 (N7:119 + N7:120).

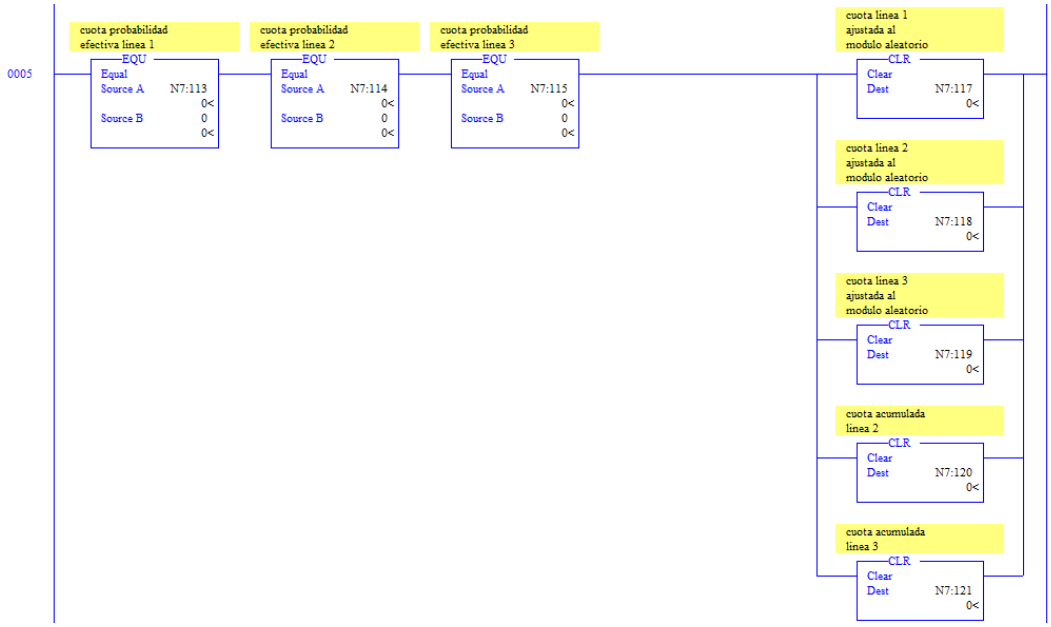


Figura 67. Restablecimiento de las cuotas de probabilidad

Si no se ha asignado una cuota de probabilidad efectiva a todas las líneas (EQ, cuota de probabilidad efectiva de la línea es igual a 0), las cuotas calculadas en el renglón anterior se restablecen al valor 0.

- TRANSICIONES

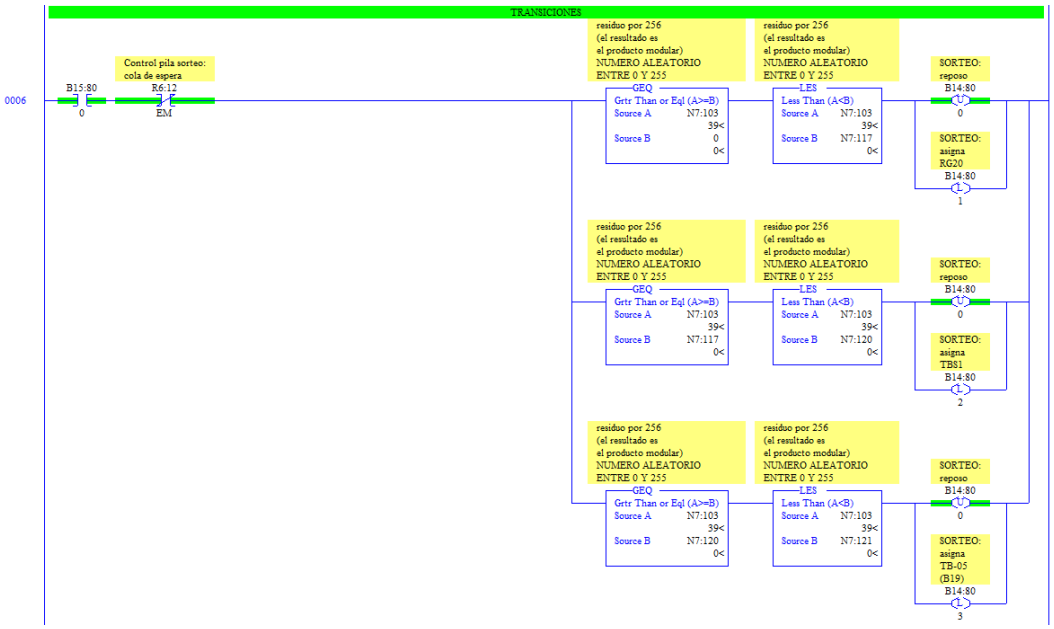


Figura 68. Activación de sorteo de cada una de las líneas

Si el bit 0 de la dirección B15:80 está activo (1) y la cola de espera de la pila de sorteo no está vacía (bit de vacío EM* de la dirección R6:12 está a 0) se realizan las siguientes operaciones:

- Si el número aleatorio almacenado en N7:103 (calculado en el primer renglón del diagrama) es mayor o igual que 0 (GEQ) y menor que (LES) la cuota de línea 1 ajustada al módulo aleatorio, se reiniciará la situación de reposo del sorteo (bit 0 de la dirección B14:80) y se activará la asignación de RG20 para sorteo (bit 1 de la dirección B14:80).

- Se repetirá el mismo proceso para las otras dos líneas, con las únicas diferencias que para la línea 2 el valor aleatorio debe ser mayor o igual que la cuota de línea 1 ajustada al módulo aleatorio (N7:117) y menor que la cuota acumulada de la línea 2 (N7:120). Para la línea 3 debe ser mayor o igual que la cuota acumulada de la línea 2 (N7:120) y menor que la acumulada de la línea 3 (N7:121).

Bit de vacío EM: establecido por la instrucción FFU (descarga FIFO) para indicar que la pila está vacía.

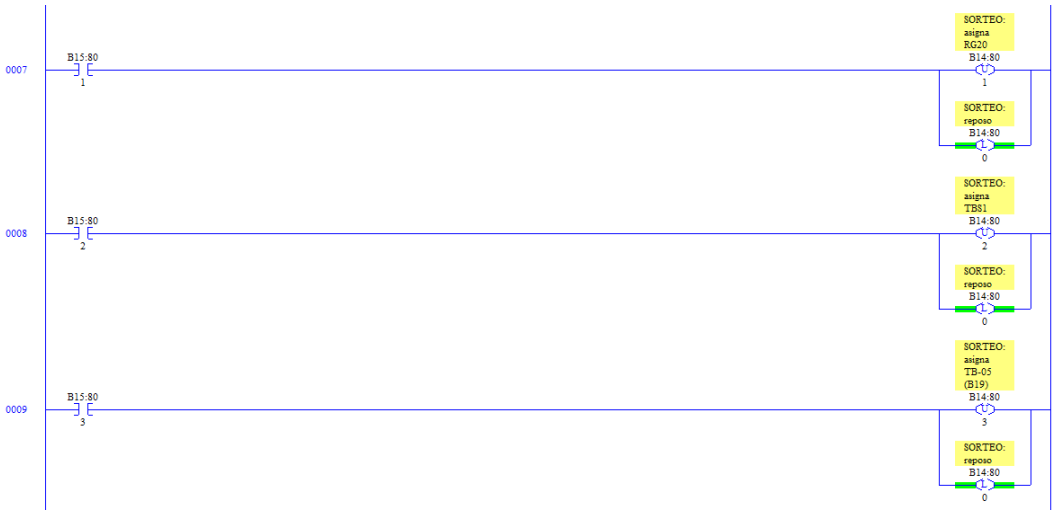


Figura 69. Activación de situación de reposo para el sorteo de cada una de las líneas

Si los bits 1, 2 y 3 de la dirección 15:80 están activos se reseteará la asignación para el sorteo y se activará la situación de reposo para la RG20, TBS1 y TB-05, respectivamente.

- ACCIONES

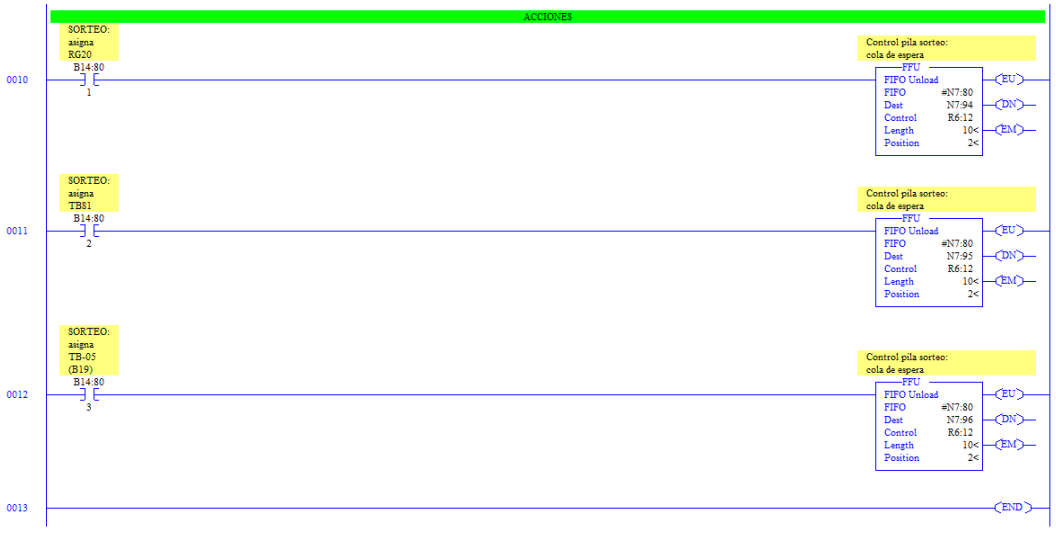


Figura 70. Control pila de sorteo

Si se ha cumplido lo anterior y se ha activado el bit de asignación para el sorteo de la RG20 (bits 1 de B14:80) se realizará la descarga de la FIFO (FFU), de forma que el elemento en la posición 0 de la pila (N7:80) se almacenará como el código de operario asignado a la RG20 (N7:94).

Mismo procedimiento para la TBS1 y TB-05.

FFU (descarga FIFO): cuando las condiciones del renglón cambian de falsas a verdaderas, se establece el bit de habilitación FFU (EU). Esto descarga el contenido del elemento en la posición 0 de la pila en el destino (Dest). Todos los datos en la pila son desplazados un elemento hacia la posición cero, y el elemento con el número más alto queda en cero. Luego decrementa el valor de posición. Esta instrucción descarga un elemento en cada transición de falsa a verdadera del renglón, hasta que la pila esté vacía. Luego se establece el bit de vacío (EM). El bit de efectuado (DN) es establecido por la instrucción FFL (carga FIFO) para indicar que la pila está llena, esto inhibe la carga de la pila. En la dirección de control se almacenan los bits de estado, la longitud de la pila y el valor de la posición.

Una vez ya tenemos un operario asignado a cada línea, desde el programa de la salida de la CDCN2 leemos las variables donde están almacenadas los números de los operarios en la B15 mediante una instrucción mensaje de tipo lectura y los almacenamos en las variables que hemos creado para trabajar con ellas más adelante desde OSISOFT.

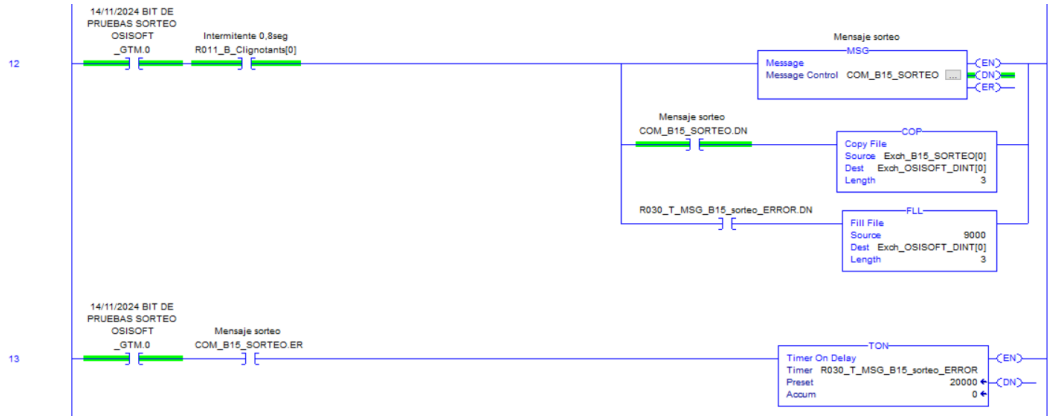


Figura 71. Lectura de mensaje de sorteo y copia en variables OSISOFT

6. PI SYSTEM

Una vez se han modificado los programas de los autómatas y se han creado las variables oportunas, se va a utilizar PI System para apuntar a dichas variables y así crear los avisos por Teams y elaborar mediante PI Vision las pantallas para la línea C de cocción y sorteo de cepillado.

6.1. COCCIÓN

En PI Server se va a trabajar con los datos proporcionados de cada una de las 16 prensas y, además, del cabecero de línea, por lo que trataremos ambos casos por separado.

6.1.1. PRENSAS

El trabajo realizado para cada una de las prensas se puede dividir en 4 pasos:

1) Descomposición del valor del estado en 16 bits
2) Creación de eventos
3) Creación de Avisos (mensajes en Teams)
4) Visualización a través de PI Vision

A continuación, se detallará todo el proceso:

1) Descomposición del valor del estado en 16 bits

Dentro de PI Server localizamos los datos de la Prensa 1 de la línea C de cocción:

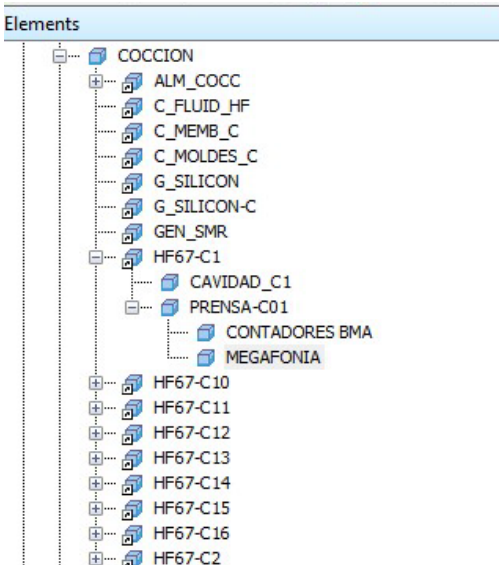


Figura 72. Localización de los datos correspondientes a la megafonía para la prensa 1 dentro de la base de datos de PI Server

El valor enviado por el autómata se descompone, como vimos en apartados anteriores, en 16 bits. Para aquellos casos en los que un determinado bit esté activo su valor (Value) será True, de lo contrario será False.

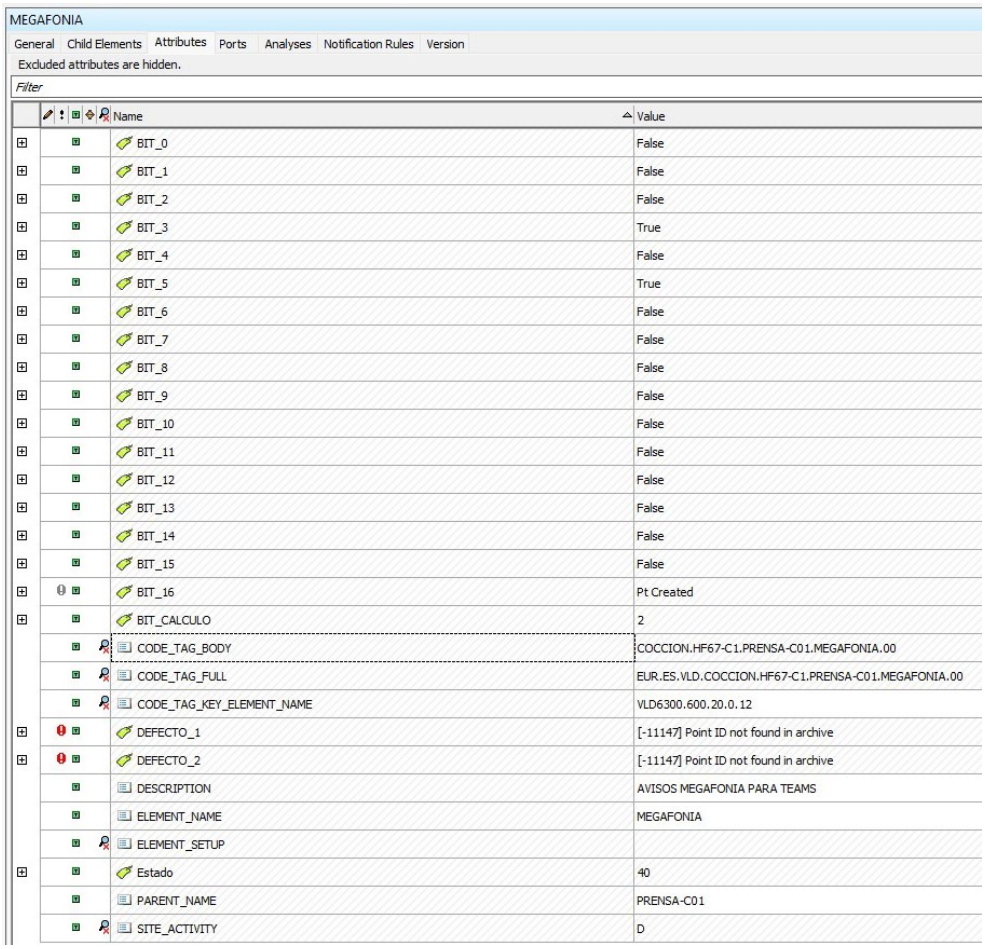


Figura 73. Descomposición de los datos proporcionados por la prensa C1

En este caso están activos los bits 3 y 5, por lo que el valor de estado debería ser: $2^3 + 2^5 = 8 + 32 = 40$

Dicho valor coincide con el valor del estado recibido del autómata de la prensa (40, se puede ver al final de la figura anterior).

Como ya se indicó anteriormente, PI Server permite realizar cálculos y crear funciones con los datos. A continuación, veremos cómo se realiza la descomposición en bits del valor del estado de las prensas.

Se pueden realizar varios tipos de análisis de datos:

- Expression (Expresión): permite realizar cálculos mediante funciones matemáticas y lógicas.

- Rollup: útil para cálculo de métricas agregadas como promedios, sumas, valores máximos y mínimos, etc.
- Event Frame Generation (Generación de Marcos de Eventos): detección y registro de eventos.
- SQC (Control Estadístico de calidad): basado en técnicas de control para el análisis de la variabilidad en los procesos.

En este caso, como necesitamos realizar cálculos para la descomposición en bits, realizaremos un análisis de tipo Expression:

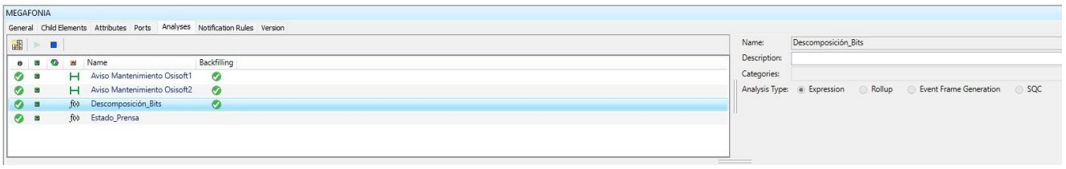


Figura 74. Creación de expresión "Descomposición_Bits"

En primer lugar, necesitamos obtener el valor del estado, el cual se encuentra almacenado en el tag (etiqueta) llamado “Estado”, como vimos en la Figura 72. Para trabajar de una manera más eficiente y no sobrecargar el sistema, se va a crear una variable de tipo entero a la que vamos a llamar “Tabla”, a dicha variable se le va a entregar el valor del último estado registrado:

Name	Expression
Tabla	TagVal(‘Estado’,’*’)

Tabla 8. Expresión para entregar el último estado registrado

Donde:

- TagVal: es una función utilizada para recuperar el valor de una variable.
- ‘Estado’: nombre del tag o variable de la que se desea recuperar el valor.
- ‘*’: representa el momento de la consulta (Timestamp), se utiliza para obtener el valor más reciente.

Una vez obtenido el valor del estado, procedemos a la descomposición en bits, para ello vamos a realizar un análisis de dicho valor bit a bit.

Supongamos que el valor de nuestro estado es 40. Como hemos visto antes, para que se dé este caso tienen que estar activos los bits 3 y 5, por lo que 40 en binario se escribiría de la siguiente manera: 0000000000101000

Como vamos a realizar un análisis bit a bit empezaremos por el de la posición 0 (situado más a la derecha). Dicho bit se corresponde al valor 2^0 igual a 1, el cual en binario se escribe de la siguiente manera: 0000000000000001

Para hacer una comparación de ambos valores en binario utilizaremos la expresión lógica AND, la cual devolverá un 1 para aquellos casos en los que exista un 1 en la misma posición dentro de los dos valores:

0000000000101000
AND
0000000000000001
0000000000000000 → 0

Como vemos ningún valor coincide y el resultado es cero. Si dividimos el valor obtenido expresado en forma decimal entre el valor de dicho bit (2^0) sabremos si ese bit se encuentra activo o no (1 si está activo, 0 si no lo está):

$$\frac{0}{2^0} = \frac{0}{1} = 0$$

Lo mismo ocurrirá para los bits 1 y 2.

Veremos qué ocurre para el bit 3 (2^3 igual a 8):

0000000000101000
AND
0000000000001000
0000000000001000 → 8

Realizamos el mismo cálculo:

$$\frac{8}{2^3} = \frac{8}{8} = 1$$

El análisis del estado del bit 3 ha dado como resultado un 1, con lo que comprobamos que ese bit está activo. Realizaremos el mismo análisis en los 16 bits en los que queremos descomponer el estado de las prensas.

Para realizar estos cálculos dentro del programa se crea la siguiente expresión:

Name	Expression	Output Attribute
VariableN	Abs(Tabla and 2^N)/2^N	BIT_N|TAG VALUE

Tabla 9. Expresión para la descomposición del estado en bits

Donde:

- N: es el número del bit a analizar.
- Tabla: variable de tipo entero donde se encuentra el valor del estado.
- and: operación lógica utilizada para comparar los dos valores.

- Abs: valor absoluto. Se utiliza para asegurarnos que el valor sea positivo.

Add a new variable

Name	Expression
Tabla	TagVal('Estado','*')
Variable1	Abs(Tabla and 2^0)/2^0
Variable2	Abs(Tabla and 2^1)/2^1
Variable3	Abs(Tabla and 2^2)/2^2
Variable4	Abs(Tabla and 2^3)/2^3
Variable5	Abs(Tabla and 2^4)/2^4
Variable6	Abs(Tabla and 2^5)/2^5
Variable7	Abs(Tabla and 2^6)/2^6
Variable8	Abs(Tabla and 2^7)/2^7
Variable9	Abs(Tabla and 2^8)/2^8
Variable10	Abs(Tabla and 2^9)/2^9
Variable11	Abs(Tabla and 2^10)/2^10
Variable12	Abs(Tabla and 2^11)/2^11
Variable13	Abs(Tabla and 2^12)/2^12
Variable14	Abs(Tabla and 2^13)/2^13
Variable15	Abs(Tabla and 2^14)/2^14
Variable16	Abs(Tabla and 2^15)/2^15

Figura 75. Análisis de estado para cada bit

2) Creación de eventos

Una vez descompuesto el estado de las prensas en 16 bits vamos a crear diferentes eventos, en este caso nos interesa saber cuándo la prensa no abre (bit 11) y cuando no cierra (bit 12).

Creamos un análisis del tipo Event Frame Generation (Generación de Marco de Eventos) al que llamamos “Aviso Mantenimiento Osisoft1”.

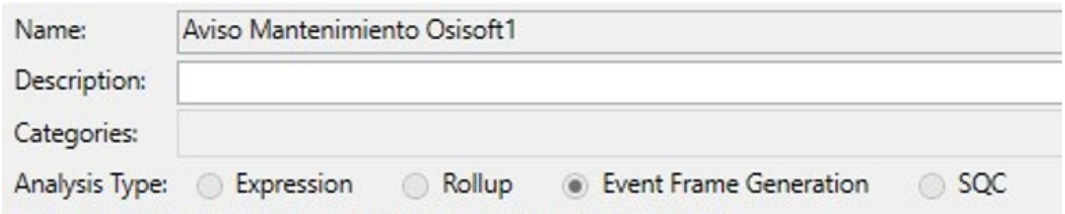


Figura 76. Creación de evento de Aviso a Mantenimiento

El desencadenante del evento (Start Trigger) es muy sencillo, cuando el valor del bit 11 del estado de la prensa se encuentre activo, es decir, su valor sea “True”. El evento finalizará (End Trigger) cuando se desactive el bit (“False”).

Name	Expression
⊟ Start triggers	
StartTrigger1	TagVal('BIT_11','*')="TRUE"
⊟ End trigger	
EndTrigger	TagVal('BIT_11','*')="FALSE"

Figura 77. Definición del evento de aviso a mantenimiento para prensa no abre

Se creará otro evento igual para el bit 12 (prensa no cierra).

3) Creación de Avisos

Se genera una notificación cuando ocurre un evento, en este caso el evento "Aviso Mantenimiento Osisoft1", el cual hemos creado dentro de la sección de megafonía en cada una de las prensas.

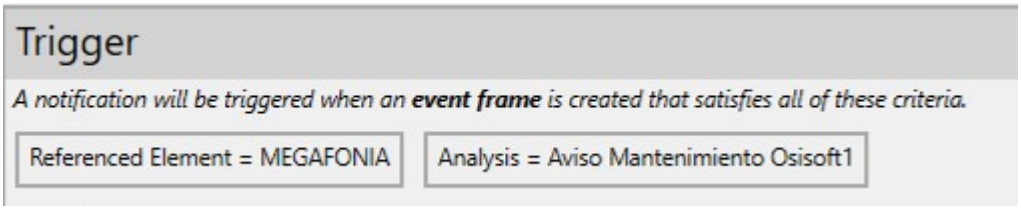


Figura 78. Definición de criterios desencadenantes de aviso

Diseño del mensaje:

Figura 79. Diseño del mensaje de aviso

Donde PARENT_NAME representa una cadena de texto que indica la prensa en la que ha ocurrido el evento (por ejemplo, "PRENSA-C01").

Este mensaje se enviará a una dirección de correo electrónico con el asunto OSISOFT AVISO MANTENIMIENTO. La lista de direcciones a las que se desea llegue los mensajes se puede editar en la pestaña de suscripciones.

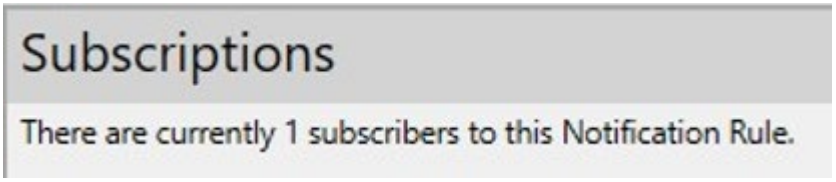


Figura 80. Suscripciones a los avisos

Para poder redireccionar los mensajes enviados al correo electrónico y visualizarlos en Microsoft Teams será necesario crear un flujo de trabajo a través de la herramienta Microsoft Power Automate.

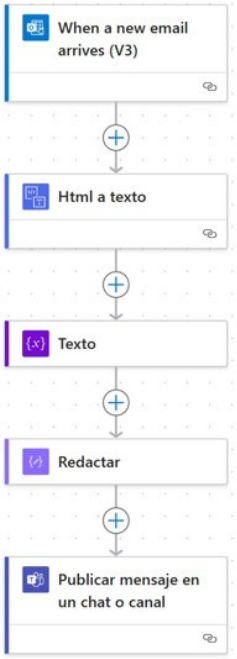


Figura 81. Flujo diseñado con Power Automate para publicar mensaje en Teams

Paso 1- Llegada de un nuevo correo a Outlook

El flujo se inicia cuando llega un correo nuevo a Outlook, para ello disponemos de un trigger específico en Power Automate (When a new email arrives).

Debemos configurar los parámetros del trigger para establecer unos determinados criterios y así concretar qué correos recibidos nos interesan:

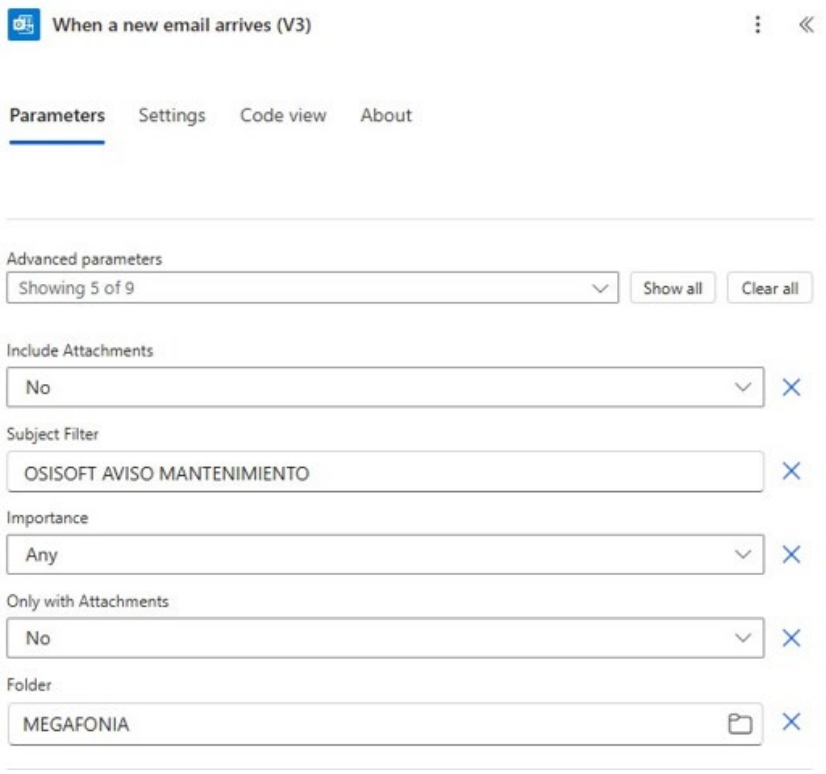


Figura 82. Parámetros del trigger del flujo en Power Automate

- No se incluyen archivos adjuntos (include attachments) en caso de que los tenga.
- Se establece un filtro del asunto (subject filter), de esta forma el trigger solo se activará cuando lleguen correos con el asunto "OSISOFT AVISO MANTENIMIENTO".
- El trigger se activa independientemente de la importancia del correo.
- Aunque el correo no contenga archivos adjuntos el trigger se activará.
- Los correos deben llegar a la carpeta "MEGAFONIA" en la bandeja de entrada.

Paso 2- HTML a texto

Los correos recibidos están en formato HTML (HyperText Markup Language), por lo que para extraer el contenido del correo haremos uso de esta función.

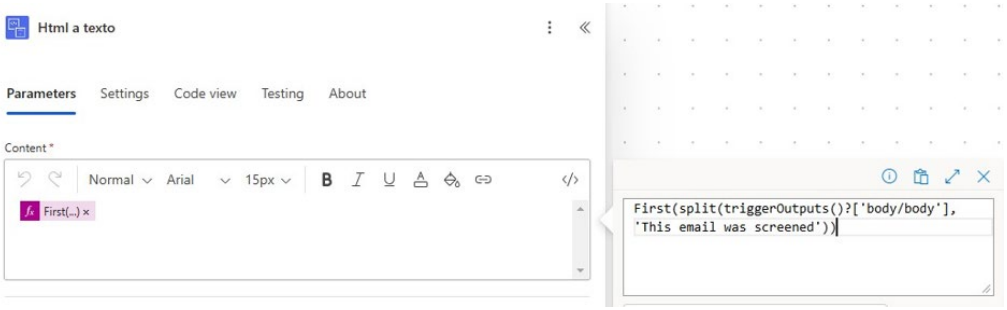


Figura 83. Parámetros de la función Html a texto en Power Automate

Dentro de la sección de contenido (content), mediante una función seleccionamos la parte del correo que nos interesa:

First(split(triggerOutputs()?['body/body'], 'This email was screened'))

- triggerOutputs()?['body/body']: recupera el cuerpo del correo que activó el trigger.
- split(..., 'This email was screened'): La función split divide el contenido del cuerpo del correo electrónico en una matriz de subcadenas, utilizando la frase “This email was screened” como delimitador. Esto significa que el contenido del correo electrónico se separará en dos partes: una antes de la frase y otra después.
- First(...): La función First toma la primera subcadena de la matriz resultante de la división. En otras palabras, se está extrayendo la parte del correo electrónico que viene antes de la frase “This email was screened”.

Paso 3- Texto

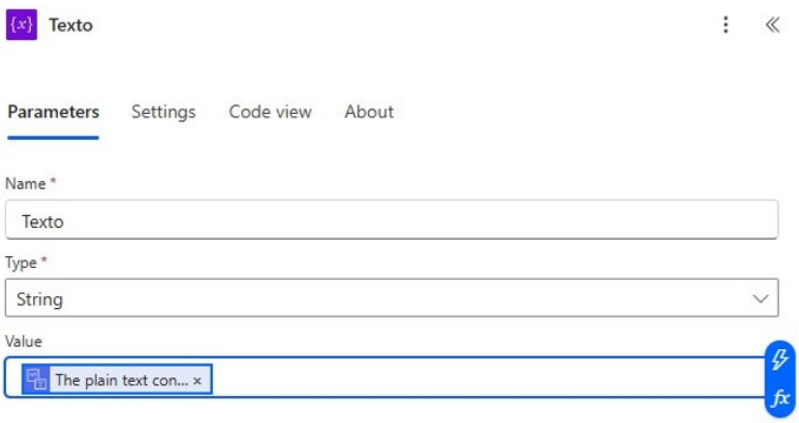


Figura 84. Creación de la variable Texto en Power Automate

Creamos una variable de tipo string llamada “Texto” en la que almacenamos el texto plano extraído en la acción anterior.

Paso 4- Redactar

Para identificar las partes clave del texto se va a hacer uso de una función a la que llamamos “Redactar”, se dividirá el texto en 3 partes, con los doble saltos de línea como separador:

Figura 85. Partes en las que se divide el texto

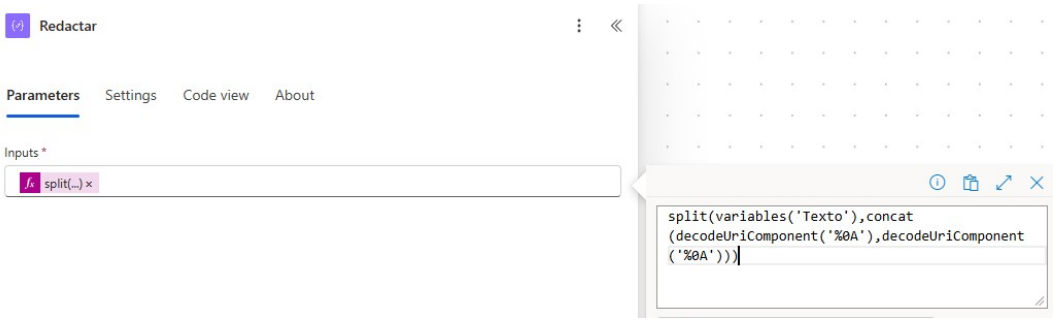


Figura 86. Función Redactar en Power Automate

Donde:

- split(variables('Texto'), ...): división del texto almacenado en la variable “Texto”. El siguiente elemento dentro de la función será el separador.
- concat(decodeUriComponent('%0A'),decodeUriComponent('%0A')): la función concat() se utiliza para unir o concatenar dos o más cadenas de texto en una sola. decodeUriComponent('%0A’) decodifica el carácter ‘%0A’ que representa un salto de línea.

Paso 5- Publicar mensaje en un chat o canal

Finalmente, sólo queda seleccionar el destinatario en Teams y agregar un formato al mensaje.

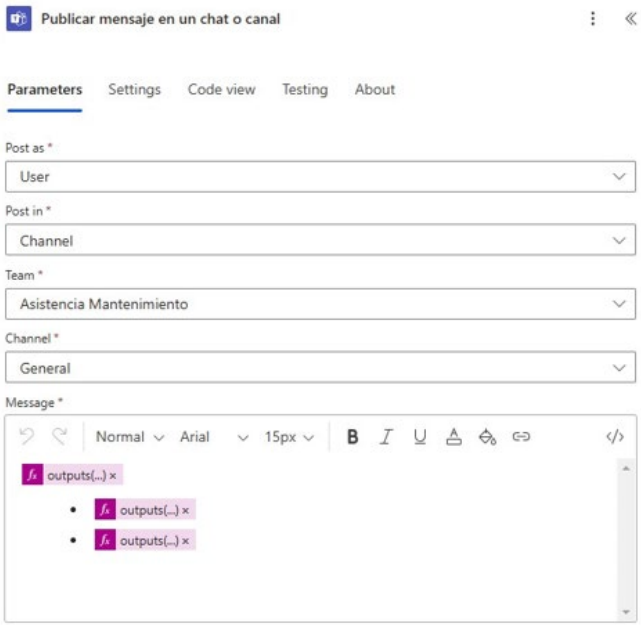


Figura 87. Acción en Power Automate para publicar un mensaje en Teams

Los mensajes se publicarán en el grupo “Asistencia Mantenimiento” en el canal general.

Damos formato al mensaje de acuerdo con la división del texto realizada en la acción anterior.

Una vez configurado todo lo anterior, se recibirán los avisos a través de Microsoft Teams:

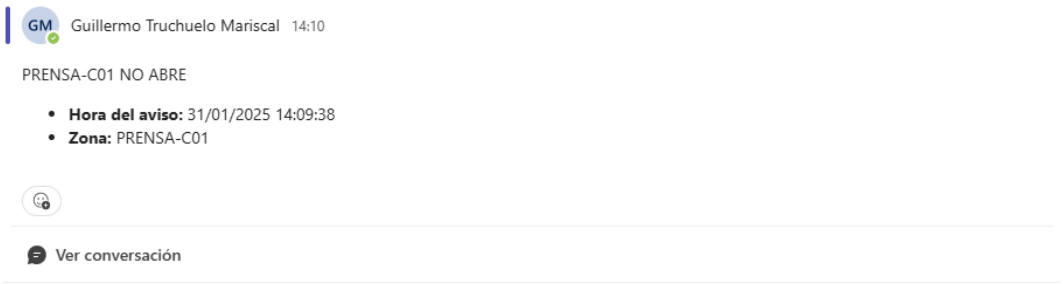


Figura 88. Ejemplo de aviso a través de Teams

4) Visualización a través de PI Vision

Finalmente, vamos a crear otro análisis de tipo Expresión que nos va a servir para la visualización de los estados de cada una de las prensas con PI Vision.

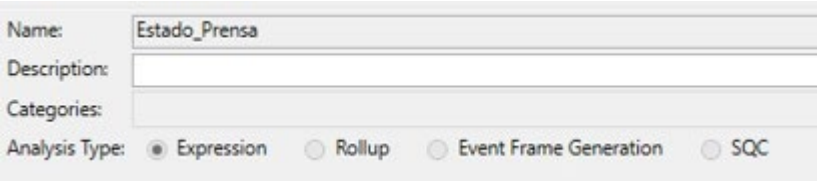


Figura 89. Creación de expresión para visualización del estado de las prensas

Para ello hay que tener en cuenta cómo se va a representar cada estado en la pantalla:

BIT	ESTADO PRENSAS LÍNEA C	
1	Autonomía	
2	Parada	
3	Producción (cociendo)	
4	Recalentando	
5	Defecto	(color en parpadeo)
6	Sanción K	(color fijo)
7	Aire no rearmado	
8	Falta calidad	(no implementado)
9	Validar Sanción K	(color fijo)
10	Cambio membrana próximo	(no implementado)
11	Prensa NO ABRE	(color en parpadeo)
12	Prensa NO CIERRA	(color en parpadeo)

Tabla 10. Asignación de estados y colores para cada bit

Como hay colores, o forma de representación de los colores, que se repiten para varios bits, a cada uno de ellos les vamos a asignar un valor entero:

Valor	Color
1	Rojo fijo
2	Rojo con parpadeo
3	Naranja fijo
4	Verde fijo
5	Blanco fijo
6	Gris fijo
7	Amarillo fijo
8	Rosa fijo

Tabla 11. Asignación de valores enteros a cada color

Crearemos una variable que tomará dicho valor en función del bit activo:

- Si BIT_6 = 1 o BIT_9 = 1, entonces Variable1 = 1
- Si BIT_5 = 1 o BIT_11 = 1 o BIT_12 = 1, entonces Variable1 = 2
- Si BIT_4 = 1, entonces Variable1 = 3
- Si BIT_3 = 1, entonces Variable1 = 4
- Si BIT_2 = 1, entonces Variable1 = 5
- Si BIT_1 = 1, entonces Variable1 = 6
- Si BIT_7 = 1, entonces Variable1 = 7
- En caso de error (ningún bit activo), Variable1 = 8.

El valor de Variable1 se almacena en el atributo de salida BIT_CALCULO, esta es la etiqueta a la que se apuntará para la visualización desde PI Vision.

Name	Expression	Output Attribute
Variable1	IF 'BIT_6'=1 or 'BIT_9'=1 THEN 1 ELSE IF 'BIT_5'=1 OR 'BIT_11'=1 OR 'BIT_12'=1 THEN 2 ELSE IF 'BIT_4'=1 THEN 3 ELSE IF 'BIT_3'=1 THEN 4 ELSE IF 'BIT_2'=1 THEN 5 ELSE IF 'BIT_1'=1 THEN 6 ELSE IF 'BIT_7'=1 THEN 7 ELSE 8	BIT_CALCULO|TAG VALUE

Figura 90. Expresión para definición de colores para cada bit de estado

El objetivo es crear un sistema de visualización igual o simular al anterior y que estaba gestionado por SAF:

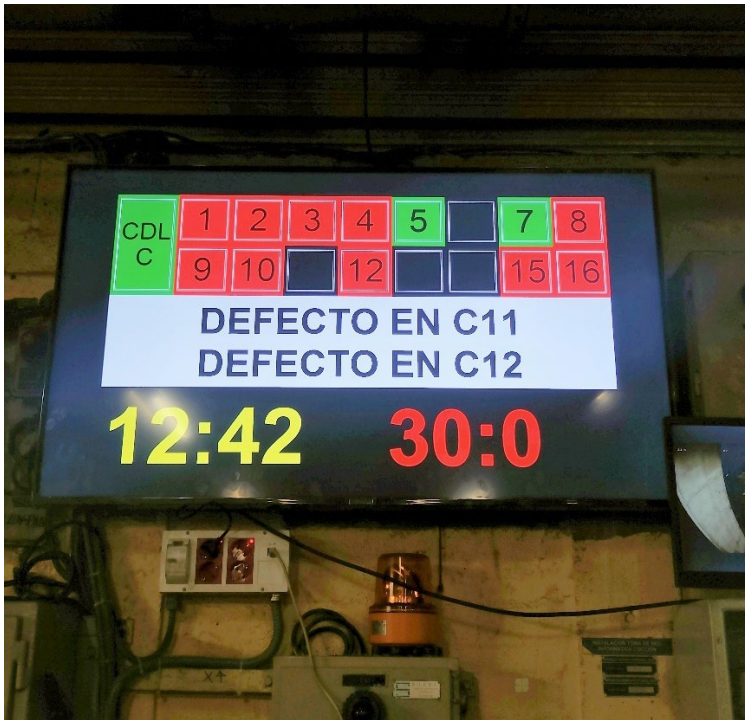


Figura 91. Pantalla Línea C de cocción con sistema previo (SAF)

Basándonos en el sistema anterior (Figura 32), a través de PI Vision creamos la nueva pantalla:

Figura 92. Nuevo sistema de visualización Línea C con PI Vision

En este apartado nos centraremos en el análisis de los estados de las 16 prensas (cuadrados de 1 a 16).

Para la visualización del estado de cada una de las prensas el análisis es muy sencillo, basta con asignar un color a cada posible valor que pueda tomar la variable que apunta a la etiqueta BIT_CALCULO:

Figura 93. Asignación de colores para los distintos estados de las prensas en PI Vision

La visualización de la hora es gestionada por el propio sistema PI Vision, el resto de los elementos son gestionados por el cabecero de la línea C, el cual analizaremos en el apartado siguiente.

6.1.2. CABECERO DE LA LÍNEA C

El análisis de los datos del cabecero será muy similar al realizado con las prensas.

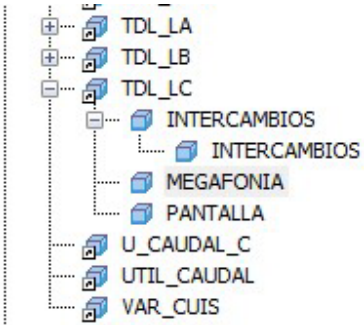


Figura 94. Localización del cabecero de la línea C dentro de la base de datos de PI Server

Dentro de la sección MEGAFONIA descomponemos el valor entero del estado en 16 bits de igual manera y lo almacenamos en las etiquetas (tags) correspondientes.

El procedimiento de cara al envío de los avisos por Teams es el mismo al explicado para las prensas (se avisará cuando el cabecero entre en defecto o atasco).

Con PI Vision se a tratar de visualizar 3 distintos tipos de información: el estado del cabecero, visualización de dos en dos de las prensas que tengan algún tipo de defecto, calidades con almacenaje igual o inferior a 3.

1) Estado del Cabecero

Los posibles estados que puede tomar el cabecero son los siguientes:

BIT	ESTADO CABEZA LÍNEA C
Ningún bit activo	OK
BIT_0=1 y BIT_1=0	Defecto
BIT_1=1	Atasco

Tabla 12. Posibles estados del cabecero y sus colores

Al igual que con las prensas, asignamos un valor a los colores que representan cada estado:

Valor	Color
1	Azul
2	Rojo
3	Verde

Tabla 13. Asignación de valores enteros a cada color

Realizamos un análisis tipo Expression almacenando el resultado en la etiqueta "Calculo_Estado" y configuramos la visualización del estado del cabecero en PI Vision:

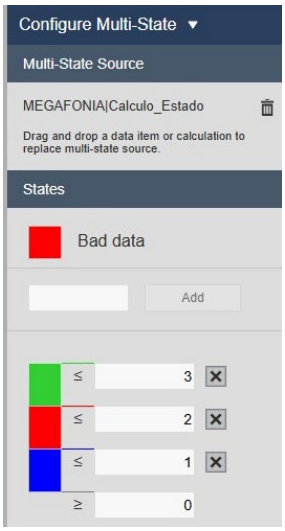


Figura 95. Asignación de colores para los distintos estados del cabecero en PI Vision

2) Visualización de las prensas con algún tipo de defecto

Si recordamos, cuando modificamos el programa del autómata del almacén de animación (PROC_ANI), uno de los objetivos fue que se almacenaran en las variables Exch_OSISOFT_INT[20] y Exch_OSISOFT_INT[21] las prensas que en ese momento tuvieran algún tipo de defecto (defecto, prensa no abre, prensa no cierra, sanción K o validar sanción K), de esta forma podríamos mostrar cada 5 segundos dos prensas distintas y sus estados.

Crearemos dos etiquetas que apunten a dichas variables enteras:

Name	Value
Aviso_Prensa_Fila1	13
Aviso_Prensa_Fila1.0	Pt Created
Aviso_Prensa_Fila1.1	Pt Created
Aviso_Prensa_Fila2	14
Aviso_Prensa_Fila2.0	Pt Created
Aviso_Prensa_Fila2.1	Pt Created

Figura 96. Etiquetas para la visualización de dos defectos

Además, para diferenciar cada tipo de defecto, se crean otras etiquetas que apunten a los bits de estado de las distintas prensas:

⊞		DA_PRENSA_C01_DEFECTO	True
⊞		DA_PRENSA_C01_NO_ABRE	False
⊞		DA_PRENSA_C01_NO_CIERRA	False
⊞		DA_PRENSA_C01_SANCION_K	False
⊞		DA_PRENSA_C01_VALIDAR_K	False
⊞		DA_PRENSA_C02_DEFECTO	False
⊞		DA_PRENSA_C02_NO_ABRE	False
⊞		DA_PRENSA_C02_NO_CIERRA	False
⊞		DA_PRENSA_C02_SANCION_K	False
⊞		DA_PRENSA_C02_VALIDAR_K	False
⊞		DA_PRENSA_C03_DEFECTO	True
⊞		DA_PRENSA_C03_NO_ABRE	False
⊞		DA_PRENSA_C03_NO_CIERRA	False
⊞		DA_PRENSA_C03_SANCION_K	False
⊞		DA_PRENSA_C03_VALIDAR_K	True

Figura 97. Etiquetas de estado de las prensas

Como los elementos que queremos visualizar son dos cadenas de caracteres, se crean dos etiquetas de tipo string de forma que, a través de otro análisis y haciendo uso de las etiquetas vistas en las figuras 37 y 38, se pueda mostrar de manera más legible la información.

⊞		DEFECTO_1	DEFECTO C11
⊞		DEFECTO_2	DEFECTO C12

Figura 98. Cadenas de caracteres para cada una de las líneas a visualizar

La representación en PI Vision quedará de la siguiente forma:

Figura 99. Visualización de dos prensas en defecto en PI Vision

Donde cada 5 segundos se mostrarán las dos siguientes prensas que se encuentren con algún tipo de defecto (si el tipo de defecto fuese el de sanción K en la prensa 14 y prensa no abre en la 17 se mostraría por pantalla "SANCIÓN K C13 / C17 NO ABRE", por ejemplo).

3) Calidades

Como los datos de las calidades y el número de ruedas (inferiores a 3) para dicha calidad los tomamos del almacén de gestión (PROC_GES), el análisis de estos datos lo realizaremos dentro de la sección "PANTALLA".

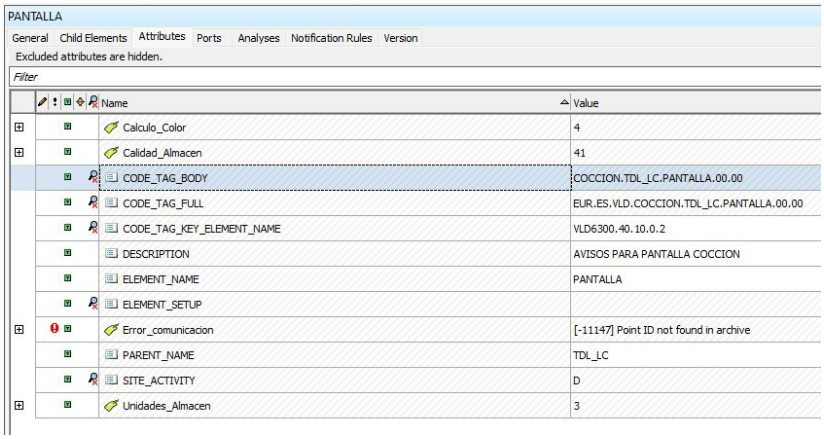


Figura 100. Atributos de la sección PANTALLA del cabecero de línea

Calidad_Almacen y Unidades_Almacen son etiquetas que apuntan a las variables enteras Exch_OSISOFT_INT[0] y Exch_OSISOFT_INT[1] del almacén de gestión (PROC_GES).

El único análisis que se va a realizar es el de asignación de colores al número de unidades en el almacén para su visualización en PI Vision:

Unidades_Almacen	Calculo_Color	Color
0	1	Rojo
1	2	Naranja
2	3	Amarillo
3	4	Verde

Tabla 14. Asignación de valores enteros y colores al número de unidades en almacén

Utilizamos otra variable Calculo_Color y no trabajamos directamente con Unidades_Almacen porque en PI Vision no se puede asignar un valor al cero.

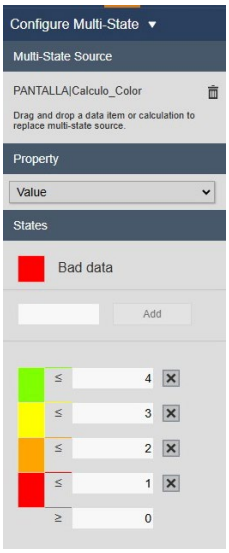


Figura 101. Asignación de colores para las unidades en almacén en PI Vision

La representación en PI Vision quedará de la siguiente manera:

Figura 102. Visualización la calidad y unidades en almacén para dicha calidad con PI Vision

Resultado final de la pantalla de cocción:

Figura 103. Pantalla Línea C de cocción con sistema nuevo (PI Vision)

Para la visualización a través del navegador ha sido necesario la instalación de un ordenador con conexión a Internet.

6.2. CEPILLADO

En la zona de cepillado, para indicar a los operarios a qué línea debían acudir a recoger la siguiente rueda a cepillar, existía un visualizador de 7 segmentos gestionado por SAF. Tras la eliminación de este sistema y con el objetivo de poder representarlo mediante PI Vision fue necesario la instalación de una nueva pantalla y ordenador con acceso a Internet como los instalados en la línea C de cocción.

Como vimos en apartados anteriores, cuando se realizó la modificación del programa del autómata de la salida de la CDCN2 (SAL_CDCN2) se crearon 3 variables de tipo entero para indicar el estado de las líneas por las que llegaban las ruedas a la zona de cepillado.

A continuación, se detallará el proceso de creación de la nueva pantalla.

Figura 104. Sección para los datos de la pantalla de sorteo

La salida de la CDCN2 es gestionada por la transística de neumáticos B16, por lo que en PI Server estos datos se tratarán en la sección TR_B16.

CODE_TAG_BODY	USINADO.TR_B16.SORTEO_PANTALLA.00.00
CODE_TAG_FULL	EUR.ES.VLD.USINADO.TR_B16.SORTEO_PANTALLA.00.00
CODE_TAG_KEY_ELEMENT_NAME	VLD6050.50.0.0.1
DESCRIPTION	PANTALLA SORTEO CARCASAS
ELEMENT_NAME	SORTEO_PANTALLA
ELEMENT_SETUP	
PARENT_NAME	TR_B16
SITE_ACTIVITY	D
Sorteo_Linea 1	3000
Sorteo_Linea 2	5000
Sorteo_Linea 3	2000

Figura 105. Datos y etiquetas para la pantalla de sorteo en PI Server

Las variables Exch_OSISOFT_INT[0], Exch_OSISOFT_INT[1] y Exch_OSISOFT_INT[2] son representadas por las etiquetas Sorteo_Linea 1, 2 y 3, respectivamente.

A diferencia de lo visto en el apartado anterior para cocción, el análisis del estado de las líneas no se hará desde PI Server, sino que se hará directamente con PI Vision, donde también se pueden crear funciones.

Antes de crear nuestra función, debemos conocer qué significan los distintos valores que pueden tomar las líneas:

Sorteo_Linea N	ESTADO
2000	DISPUESTA
3000	OCUPADA
4000	BLOQUEADA
5000	EN ESPERA
9000	ERROR COMUNICACIÓN
otro	número de operario

Tabla 15. Códigos de estado para las líneas de cepillado

La programación de nuestra función es muy sencilla:

- Si Sorteo_Linea N es igual a 2000, entonces escribimos la cadena de caracteres “DISPUESTA”
- Si Sorteo_Linea N es igual a 3000, entonces “OCUPADA”
- Si Sorteo_Linea N es igual a 4000, entonces “BLOQUEADA”
- Si Sorteo_Linea N es igual a 5000, entonces “EN ESPERA”
- Si Sorteo_Linea N es igual a 9000, entonces “ERROR COM”
- Si Sorteo_Linea N es igual a un valor entero distinto a los anteriores, entonces se mostrará el propio valor, el cual representa el número de operario que debe acudir a dicha línea.

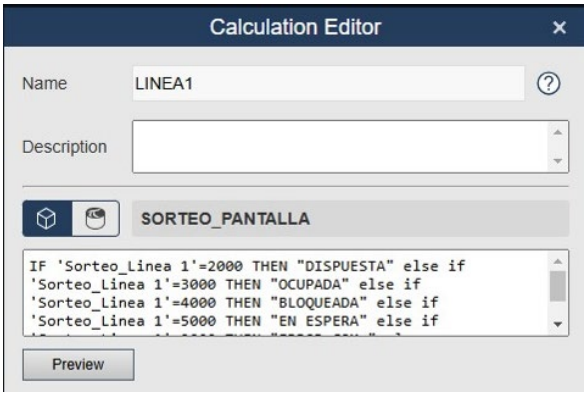


Figura 106. Función estado de la línea 1 en PI Vision

Finalmente, solo queda otorgar un color a cada uno de los posibles estados:

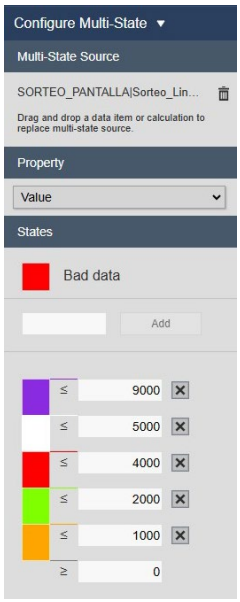


Figura 107. Asignación de colores para el estado de la línea 1

Los estados “OCUPADA” y “BLOQUEADA” se mostrarán en rojo. Como PI Vision sólo permite trabajar con rangos, la representación del número de operario (naranja) será para aquellos valores inferiores a 1000 (no hay operarios con un número superior).

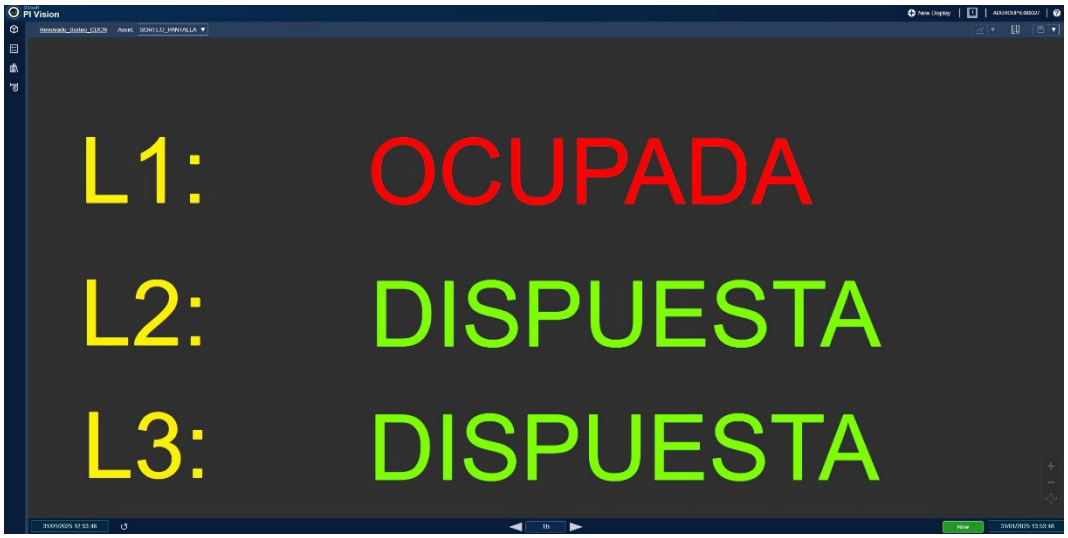


Figura 108. Resultado final de la visualización de la pantalla de sorteo con PI Vision

Resultado final de la nueva pantalla en la zona de cepillado con la visualización del estado de las tres líneas a través de PI Vision:

Figura 109. Pantalla de sorteo de cepillado con nuevo sistema (PI Vision)

6.3. ALMACENES DEL SÓTANO Y CEPILLADO

Para los programas de ambos autómatas se crearon las variables booleanas Exch_OSISOFT_BOOL[0] para avisar a mantenimiento (en el caso del almacén del sótano también se podrá avisar a producción a través de Exch_OSISOFT_BOOL[1]).

Se realizan los ajustes pertinentes en PI Server (visto en el apartado de cocción) y finalmente llegarán avisos a Microsoft Teams como los siguientes:

Figura 110. Aviso a mantenimiento en almacén del sótano a través de Teams

Figura 111. Aviso al equipo AMF2 de producción en almacén del sótano a través de Teams

7. CONCLUSIONES Y LÍNEAS FUTURAS

7.1. CONCLUSIONES

El nuevo sistema de alertas implementado en la factoría Michelin de Valladolid ha resultado ser un éxito en términos de mejora de la comunicación y eficiencia en la gestión de alertas en tiempo real. A continuación, se resumen las principales conclusiones obtenidas:

- **Mejora en la Comunicación**: el nuevo sistema basado en PI System y Microsoft Power Automate ha reemplazado eficazmente el antiguo sistema SAF, eliminando las limitaciones tecnológicas y mejorando la escalabilidad.

  La integración con Microsoft Teams ha permitido enviar notificaciones específicas a los equipos de mantenimiento y producción, reduciendo el ruido producido por la megafonía y asegurando que los avisos lleguen a las personas o grupos indicados, de este modo los trabajadores pertenecientes al grupo de mantenimiento no recibirán avisos correspondientes al de producción, y viceversa.

- **Visualización en Tiempo Real**: la transición desde el sistema anterior (SAF) hacia PI Vision se realizó sin interrupciones significativas en las operaciones, lo que demuestra la robustez del nuevo sistema.

- **Automatización y Eficiencia**: la automatización de los flujos de trabajo con Power Automate ha reducido la necesidad de intervención manual, minimizando errores y acelerando los tiempos de respuesta.

- **Adaptabilidad y Futuro**: el sistema implementado es altamente adaptable y puede extenderse a otras áreas de la fábrica, aprovechando la infraestructura ya existente.

Finalmente, tras la realización del proyecto se han logrado los objetivos marcados en el comienzo de la memoria:

- **Adquirir y aplicar conocimientos** relacionados con la programación de autómatas (PLC).
- **Comprender** el funcionamiento del **proceso industrial**.
- **Conocer** los **protocolos de comunicación** entre los distintos dispositivos que forman parte del proceso.
- **Modificar** el **programa de los autómatas** creando las rutinas y variables oportunas para la gestión de las alertas.

- **Programar** las rutinas necesarias para la visualización de los datos recogidos por los autómatas en pantallas.
- **Gestionar** los **datos** proporcionados por los autómatas desde PI System y crear las pantallas informativas con PI Vision.
- **Automatizar** mensajes a Microsoft Teams a través de **flujos de trabajo** con Microsoft Power Automate.

7.2. LÍNEAS FUTURAS

A continuación, se detallan algunas de las posibles mejoras para este sistema en el futuro:

- **Optimización del Tiempo de Refresco**: por el momento PI Vision presenta ciertas limitaciones, una de ellas es el tiempo de refresco, el cual se realiza cada 5 segundos. Esto es un ligero inconveniente ya que, en ocasiones la visualización del color representado no se corresponde con el que debería (muestra el correspondiente al del dato del refresco anterior).

  La reducción del tiempo de refresco conlleva la modificación para todas aquellas pantallas conectadas a la misma red de fábrica, lo que puede llevar a que el sistema se sobrecargue. Se espera que PI System introduzca mejoras próximamente que permitan una visualización en tiempo real más efectiva.
- **Integración de técnicas de Inteligencia Artificial (IA) y Machine Learning (ML)**: útil para predecir fallos en las máquinas antes de que ocurran, mejorando el mantenimiento preventivo e identificar patrones y optimizar el proceso de producción.
- **Expansión a Otras Áreas de la Fábrica**: extender el sistema de alertas y visualización a otras líneas de producción, aprovechando la escalabilidad de PI System.
- **Mejoras en la Interfaz de Usuario de PI Vision**: desarrollar interfaces más intuitivas y personalizables, permitiendo a los operarios adaptar los paneles a sus necesidades específicas.
- **Sostenibilidad y Eficiencia Energética**: utilizar los datos recopilados para analizar y optimizar el consumo energético de las máquinas.

BIBLIOGRAFÍA

[1] Brolla Factory, «¿Qué es la Automatización Industrial y sus características?,» [En línea]. Disponible en: https://brollafactory.com/es/blog/automatizacion-industrial-que-es-y-sus-caracteristicas/#%C2%BFQue_es_la_Automatizacion_Industrial. [Último acceso: 22 febrero 2025]

[2] Cursos Aula21, «¿Qué es la automatización industrial?,» [En línea]. Disponible en: https://www.cursosaula21.com/que-es-la-automatizacion-industrial/. [Último acceso: 22 febrero 2025]

[3] Cursos Femxa, «Autómatas Programables,» [En línea]. Disponible en: https://www.cursosfemxa.es/blog/automatas-programables. [Último acceso: 26 febrero 2025]

[4] Allen-Bradley, «Controladores Programables PLC-5 — Referencia del Conjunto de Instrucciones,» Manual técnico, 2000.

[5] OSISOFT, «OSISOFT,» [En línea]. Available: https://aplicaciones.campusbigdata.com/aplicacion/osisoft/. [Último acceso: 15 marzo 2025]

[6] AVEVA, «PI SYSTEM,» [En línea]. Available: https://discover.aveva.com/product-pi-system/brochure-aveva-pi-system?utm_source=website-aveva&utm_medium=cta-brochure&utm_campaign=op-pi-new&utm_content=pathfactory-content. [Último acceso: 15 marzo 2025]

[7] AVEVA, «PI VISION,» [En línea]. Available: https://www.aveva.com/content/dam/aveva/documents/datasheets/Datasheet_AVEVA_PIVision_24-01.pdf. [Último acceso: 15 marzo 2025]

[8] AVEVA, «PI SERVER,» [En línea]. Available: https://www.aveva.com/es-es/products/aveva-pi-server/. [Último acceso: 20 marzo 2025]

[9] Microsoft, «Power Automate,» [En línea]. Disponible en: https://www.microsoft.com/es-es/power-platform/products/power-automate. [Último acceso: 21 marzo 2025]

[10] Open Soft Systems, «3-Day Rockwell Allen-Bradley SLC 500 Maintenance & Fault Finding Course,» [En línea]. Disponible en: https://opensoftsystems.co.uk/3-day-rockwell-allen-bradley-slc-500-maintenance-fault-finding-course/. [Último acceso: 8 mayo 2025]

[11] AutomatismosMundo, «Los lenguajes de programación de PLC,» [En línea]. Disponible en: https://automatismosmundo.com/los-lenguajes-de-programacion-de-plc/. [Último acceso: 8 mayo 2025]

[12] TIGA, «Integrating AVEVA PI with Enterprise Systems,» [En línea]. Disponible en: https://www.tiga.us/resources/pi-data-integration. [Último acceso: 8 mayo 2025]

[13] Automation Networks, «RSLinx Classic and Enterprise Software,» [En línea]. Disponible en: https://automation-networks.es/glossary/rslinx-classic-and-enterprise-software. [Último acceso: 9 mayo 2025]

[14] F. Berzosa, *SAF – Sistema de Alertas de Fabricación, Manual de Usuario. Versión 2.0*, Michelin España y Portugal, S.A., 2012.

[15] AVEVA, «AVEVA Edge Data Store,» [En línea]. Disponible en: https://www.aveva.com/es-es/products/aveva-edge-data-store/. [Último acceso: 9 mayo 2025]

[16] AVEVA, «CONNECT,» [En línea]. Disponible en: https://www.aveva.com/es-es/solutions/connect/. [Último acceso: 9 mayo 2025]

[17] AVEVA, «AVEVA PI DataLink,» [En línea]. Disponible en: https://www.aveva.com/en/products/aveva-pi-datalink/. [Último acceso: 9 mayo 2025]

[18] AVEVA, «PI Integrator for Business Analytics 2020 R2 SP1,» [En línea]. Disponible en: https://docs.aveva.com/bundle/pi-integrator-for-business-analytics/page/1023068.html. [Último acceso: 9 mayo 2025]

ANEXOS

ANEXO I. DOCUMENTACIÓN SLC 500

ANEXO II. PANTALLA DE ALERTAS MÚLTIPLES (PAM)