Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

ÁREA DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

APLICACIÓN DE METODOLOGÍA ISA88 AL SISTEMA DE PLCNEXT PARA CONTROL, SUPERVISIÓN, INTEGRACIÓN Y ANÁLISIS DE DATOS EN LA NUBE

D. QUINTANA SIÑERIZ, Gabriel
TUTOR: D. MATEOS MARTÍN, Felipe

FECHA: JULIO 2023

ÍNDICE DE LA MEMORIA

ÍNDICE DE CONTENIDOS:

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

MEMORIA

Gabriel Quintana Siñeriz

Glosario

Este apartado es destinado a recopilar algunos términos reflejados a lo largo del documento que por su complejidad, reiteración o procedencia técnica conviene tener agrupados para disponer de una referencia rápida de consulta.

- **AXC F 2152:** Sistema de control (generalmente referido directamente como controlador) de la gama de AXC F XX52 asociada a la tecnología PLCnext de Phoenix Contact.
- **Azure:** Plataforma de computación en la nube de Microsoft.
- **Balena-Engine:** Componente encargado de gestionar los contenedores en el dispositivo IoT.
- **CFC:** Lenguaje gráfico de funciones continuas. Es uno de los diferentes lenguajes de programación para los controladores lógicos programables (PLC) recogido bajo el estándar IEC 61131-3.
- **Cliente:** En una situación cliente-servidor es el encargado de demandar la información del servidor.
- **Codesys:** Entorno de desarrollo de código para programación de controladores lógicos programables (PLCs) bajo la normal IEC 61131-3.
- **CPS:** Sistemas ciberfísicos. Sistema que integra capacidades de computación, almacenamiento y comunicación. Estrechamente ligados al apartado IoT. Algunos ejemplos son: drones, vehículos autónomos, domótica…
- **Dashboard:** Término empleado para referirse a una pantalla de interacción con el usuario en el entorno Node-RED.
- **Docker:** Componente encargado de gestionar los contenedores en el dispositivo IoT.
- **Docker Desktop:** Aplicación adaptada en forma de programa a un sistema Windows para ser operado de manera más sencilla.
- **EDU AXC F 2152:** Es un entorno físico de la tecnología PLCnext para familiarizarse con la programación del controlador AXC F 2152, disponiendo este de módulos de entradas y salidas tanto analógicas como digitales integradas y listo para ser operado.
- **ENUM:** Es un tipo de dato en el entorno de la programación.

- **FBD:** Lenguaje de diagrama de bloques de funciones. Es uno de los diferentes lenguajes de programación para los controladores lógicos programables (PLC) recogido bajo el estándar IEC 61131-3.
- **HMI:** Siglas referidas a Human Machine Interface que hacen referencia a un panel gráfico de interacción máquina-usuario.
- **IBM Cloud:** Plataforma de computación en la nube de IBM.
- **Industria 4.0:** Término para referirse a la cuarta revolución industrial.
- **Industria 5.0:** Término aún no muy extendido para referirse a la quinta etapa de la revolución industrial.
- **IoT:** Internet de las Cosas. Se refiere a la interconexión de dispositivos con la finalidad de intercambiar datos entre si a través de internet u otras redes de comunicación.
- **IoT Hub:** Herramienta para realizar la comunicación del dispositivo con la plataforma Cloud Azure.
- **IoT Watson:** Herramienta para realizar la comunicación del dispositivo con la plataforma Cloud IBM.
- **ISA88:** Abreviatura de ANSI/ISA88. Es una filosofía de diseño aplicada en la industria para la programación de plantas.
- **ISA 101:** Estándar para el diseño de HMIs.
- **Javascript:** Se trata de un lenguaje de programación usado por el entorno Node-RED.
- **LD:** Lenguaje de programación Ladder. Es uno de los diferentes lenguajes de programación para los controladores lógicos programables (PLCs) recogido bajo el estándar IEC 61131-3.
- **MODBUS:** Protocolo de comunicación abierto para intercambio de información entre dispositivos.
- **Modelo de actividad o control de procedimiento:** Comúnmente referido como control en el estándar ISA88.
- **Modelo físico o de equipamiento:** Referido de manera común como módulo de equipamiento en el estándar ISA88.
- **Modelo de receta procedural o procedimental:** Conocido como receta en el estándar ISA88.

- **Node-RED:** Entorno de programación gráfico, basado en el lenguaje de programación JavaScript.
- **OPC UA:** Estándar de comunicación segura en el entorno industrial.
- **Pantalla de operador:** Término que se empleará para referirse a un HMI.
- **Phoenix Contact:** Fabricante en cuyo catálogo de productos se pueden encontrar elementos para desarrollo de aplicaciones de automatización industrial.
- **Plataforma Cloud**: Conjunto de servicios de computación en la nube que permiten el desarrollo de aplicaciones, almacenamiento y computación.
- **PLC:** Controlador lógico programable. Sistema de control informático para el desarrollo de proyectos de automatización industrial.
- **PLCnext:** Ecosistema de automatización industrial desarrollado por Phoenix Contact.
- **PLCnext Engineer:** Entorno de programación para los controladores asociados a la tecnología PLCnext.
- **POU:** Término para referirse a unidad de organización de programa que se utiliza para referirse a los elementos de programación que se emplean para crear una aplicación de controlador.
- **Proficloud.io:** Plataforma de computación en la nube para tecnologías PLCnext.
- **Servidor:** En una situación cliente-servidor es el encargado de provisionar la información solicitada por el cliente.
- **SFC:** Siglas correspondientes a Gráfico Funcional Secuencial. Es uno de los diferentes lenguajes de programación para los controladores lógicos programables (PLCs) recogido bajo el estándar IEC 61131-3.
- **SFTP:** Término para referirse a protocolo de transferencia segura de archivos empleado por la herramienta WinSCP.
- **SSH:** Protocolo de red empleado por el cliente Putty para generar una emulación de terminal usada con el controlador PLCnext.
- **ST:** Siglas correspondientes a texto estructurado. Es uno de los diferentes lenguajes de programación para los controladores lógicos programables (PLCs) recogido bajo el estándar IEC 61131-3.
- **Terminal:** Término empleado para referirse a la ventana de introducción de comandos de un dispositivo.

- **Timestamp:** Nodo empleado en Node-RED para la realización de una acción en un momento seleccionado por el usuario.
- **UaExpert:** Cliente OPC UA empleado para la verificación del correcto funcionamiento e interacción con el servidor.
- **UTC:** Tiempo universal coordinado. Estándar principal de tiempo a nivel global.
- **Webserver:** Programa informático que forma parte de un servidor. En este caso se asocia al controlador PLCnext para obtención de información asociada al controlador físico y al HMI desarrollado por el usuario.
- **WinSCP:** Aplicación empleada para intercambio de archivos entre dispositivo local y PLC.

1. Introducción

1.1.- DATOS GENERALES

DATOS GENERALES DEL PROYECTO	
Título del proyecto	Aplicación de metodología ISA88 al sistema PLCnext para control, supervisión, integración y análisis de datos en la nube
Autor del proyecto	Gabriel Quintana Siñeriz
Tutor del proyecto	Felipe Mateos Martín
Área	Ingeniería de Sistemas y Automática
Fecha de presentación	Julio 2023

Tabla 1.1 Datos generales del proyecto

1.2.- ANTECEDENTES

La finalidad por la que ha sido desarrollado este proyecto es con el fin de explorar la creación y desarrollo de un sistema de automatización basado en controlador PLC incluyendo prestaciones avanzadas para cubrir las necesidades y requisitos del nuevo paradigma de la Industria 4.0.

Se han empleado una variada cantidad de recursos que permiten adaptar dicho sistema a este paradigma. Algunos de estos ejemplos es el uso de la filosofía de diseño conocida como ISA88 y la metodología GEMMA. El uso de Codesys ha sido concebido como herramienta inicial de implementación y desarrollo del sistema aplicando estas metodologías, para posteriormente realizar su completo trasvase de programación e implementación en PLCnext Engineer para ser aplicado a controladores AXC F XX52 pertenecientes a la tecnología PLCnext de Phoenix Contact, que es una novedosa plataforma para abarcar estas necesidades requeridas en la Industria 4.0.

Durante la realización de este proyecto se ha aplicado una mentalidad de desarrollo de proyecto a nivel de carácter educativo y aprendizaje en el cual se ha buscado realizar una simulación completa de proceso consiguiendo la mayor independencia posible de una planta física y casi siendo completamente independiente de la necesidad de un controlador real.

Antes de proceder con el siguiente apartado, también es interesante mencionar que el presente trabajo ha servido como base para dos presentaciones (workshops) de carácter internacional:

- "Developing IEC 61131-3 code for Industry 4.0" para el EXP.AT WS'21 (Experiment@International Workshop) in Topic Online Experimentation (OE) & Digital Transformation, en sesion online desarrollada el día 22-Noviembre-2021, y

- "An Approach to the Development of an Automation 4.0 System Using PLCnext Technology" durante la celebración "Workshops about Applications on Industry 4.0" para el proyecto Erasmus+ ETAT (Education & Training for Automation 4.0 in Thailand) celebrada en el Aula Magna de la E.P. Ingeniería de Gijón (22-Junio-2023).

A su vez, también ha servido para la elaboración de dos documentos (uno en castellano y otro en inglés) de ayuda para este mismo proyecto ETAT, ya referenciado, que reciben el nombre de: "OPC-UA, Node-RED and Cloud functionalities for PLCnext Technology" [1] y "OPC-UA, Node-RED y funcionalidades Cloud con tecnología PLCnext" [2]. Estos documentos contienen información de interés para la configuración y puesta en marcha de los dispositivos y las tecnologías involucradas en este trabajo, habiendo siendo apreciado por el consorcio de universidades de proyecto Erasmus+ ETAT y la red de universidades Edunet promovida por Phoenix Contact y a la que pertenece el Área ISA-Uniovi desde hace más de 14 años.

1.3.- VISIÓN GENERAL DE PROYECTO

Como ya se ha indicado en el apartado previo, el conjunto de tecnologías compuesto tanto por software y hardware ha sido variado de cara a realizar una primera aproximación y desarrollo de un proyecto adaptado a las necesidades generadas por la industria 4.0.

Comenzando por la parte de hardware asociada al proyecto desarrollado, indicar que el controlador de PLCnext empleado corresponde a la unidad AXC F 2152 de Phoenix Contact que se dispone en el laboratorio asociado al módulo de entrenamiento EDU AXC F 2152. Este equipo proporciona las prestaciones necesarias y suficientes para la implementación de este trabajo.

La utilización de la tecnología PLCnext exige la necesidad de la herramienta de configuración y programación denominada PLCnext Engineer, proporcionada por Phoenix Contact. Sin embargo, previo al desarrollo completo de la aplicación/programación en dicho software, se ha empleado el software Codesys para los primeros desarrollos, disponiendo de la posibilidad de aproximarse al control y programación del sistema de una manera totalmente independiente de un controlador.

A su vez, para la realización de muchas partes del proyecto ha sido necesario el acceso a software externo como WinSCP y UaExpert. Por supuesto, también ha sido necesario el uso de una plataforma Cloud de cara a afrontar las funcionalidades destinadas a la nube; para ello, se han empleado las plataformas IBM y Azure, en concreto haciendo uso de herramientas como IoT Watson y IoT Hub que se tratarán en mayor detalle a lo largo de este documento.

1.4.- OBJETIVOS

Una vez que ya han sido introducidas unas pequeñas pinceladas del concepto general de proyecto y tecnologías necesarias para la implementación del mismo, se enumeran los objetivos que han sido perseguidos con la elaboración de este trabajo.

1. Diseño a implementación de una primera versión del programa de control en Codesys para un sistema de alimentación y tratamiento térmico de piezas en un horno. Se considera el uso de metodologías ISA88, GEMMA y GRAFCET para el análisis y modelado del sistema.
2. Generación de código para simulación del proceso en Codesys e integración en el programa de control con la finalidad de verificar su funcionamiento sin necesidad del uso ni de una planta ni un controlador reales.
3. Adaptación completa del programa de control y supervisión del sistema desarrollado en Codesys a la herramienta PLCnext Engineer para su operación en equipos con tecnologías PLCnext.
4. Carga y verificación de la correcta configuración y funcionamiento de la aplicación en el controlador PLCnext AXC F 2152 disponible en el módulo entrenador EDU AXC F 2152.
5. Creación de servidor OPC UA con PLCnext Engineer en el controlador AXC F 2152 con la finalidad de visualizar y actuar sobre algunas de las variables de control.
6. Instalación e implementación de programa Node-RED que actúe sobre las ya citadas variables del servidor OPC UA y permitan su envío a la plataforma Cloud seleccionada.
7. Generación de una pantalla de visualización (Dashboard) en Node-RED con la finalidad de observar información del proceso e incluso poder interactuar con la planta.
8. Habilitar el paso de datos desde el controlador a la nube, verificando la llegada y generando una pantalla de visualización de acceso libre externo a la planta.

1.5.- ALCANCE ASOCIADO AL PROYECTO

La finalidad de este proyecto ha sido la adaptación de un sistema ejemplo de un horno industrial a un control que introduzca estándares que amplíen la flexibilidad de la planta como puede ser la ISA 88, una guía para la organización de funcionamiento de planta como puede ser la GEMMA y el uso de herramientas externas para poder interactuar con la planta por medio de elementos diferentes al tradicional que obligan al usuario a interactuar con el sistema de manera local.

Este proyecto como se ha mencionado se presenta y desarrolla sobre un entrenador específico al que ya se ha hecho mención (EDU AXC F 2152), pero con el fin de que el usuario pueda realizar cualquier tipo de prueba aplicada a su situación específica no es necesario disponer de este modelo en específico.

La gran ventaja es que puede ser adaptado a cualquiera de los modelos AXC F XX52 de la gama PLCnext, al igual que empleado con cualquiera de los sistemas externos a los propios controladores AXC, como pueden ser los PC industriales PC Box o los Edge devices EPC, ambos asociados a la tecnología PLCnext de Phoenix Contact ya que estos permiten realizar el desarrollo en el entorno PLCnext Engineer ya citado.

De esta manera, con la inclusión de estos mecanismos y explicación del uso de estos, se pretende proporcionar al usuario diferentes herramientas base y flexibilidad suficiente para adaptarse a lo que el usuario requiera en su implementación. Si bien es cierto que parece que se usa de forma exclusiva tecnología PLCnext, en gran parte del desarrollo seguido se puede considerar muy general salvando las ventajas que aporta PLCnext como novedosa plataforma de integración hacia la digitalización en la industria.

En resumen, lo que se pretende conseguir con este proyecto es una aproximación educativa y detallada de cómo abordar la programación de un proyecto de automatización de cara a abarcar las necesidades solicitadas por el paradigma de la Industria 4.0.

1.6.- CONTENIDO Y DOCUMENTOS DEL PROYECTO

En esta sección final contenida dentro de la introducción del presente documento, se recopilarán los documentos asociados y apartados que componen el trabajo.

- **Memoria**: Es este documento donde se desarrolla de forma detallada los aspectos clave de proyecto, sus objetivos y alcance, los componentes involucrados, el diseño y desarrollo de toda la programación en diferentes lenguajes y plataformas, planificación, presupuesto y conclusiones finales.

Un conjunto de documentos y ficheros a modo de **Anexos** que se citan a continuación:

- **Programa de control desarrollado en Codesys**: Código fuente realizado en Codesys v3.5 del desarrollo de la planta, incluido como anexo bajo el nombre "*Codesys_Gabriel_Programa_HORNO.project*"
- **Programa de control desarrollador en PLCnext Engineer**: Adaptación del código inicialmente desarrollado en Codesys al entorno PLCnext Engineer 2022.6, incluido como anexo con el título:
  "*PLCnext_Gabriel_HORNO_PLCnext_2022.6.0_Proficloud_funcionando.pcwex* ".
- **Sesión de cliente en UaExpert**, para realizar rápida de prueba de verificación de variables en el servidor OPC UA del controlador PLCnext, añadida como fichero anexo de título "*UaExpert_PLCnext_OPC_UA_variables.uap*".
- **Programa desarrollado en Node-RED para controlador PLCnext**: Programa que permite la recepción y envío de datos al servidor OPC UA del controlador, genera una representación gráfica local y envía datos a la plataforma Cloud. Añadido como anexo bajo el nombre:
  "*Node_RED_PLCnext_Gabriel_Horno_wAzure.json*"
- **Programa desarrollado en Node-RED para plataforma Cloud**: Programa que permite la recepción de datos por parte del controlador en la plataforma Cloud. Añadido como anexo bajo el título:
  "*Node_RED_CloudAzure_Gabriel_Horno.json*"

En este documento **Memoria** se incluyen los siguientes capítulos:

- **Introducción**: Apartado inicial en el que se realiza una breve puesta en contexto de la razón de desarrollo del proyecto y un resumen del mismo.
- **Planta automatizada**: Se describe del ejemplo de planta o proceso a automatizar que sirve como base para el desarrollo de los programas de control desde el controlador hasta la nube.
- **Diseño detallado del sistema:** Capítulo en el cual se desarrolla en detalle, todos los elementos y tecnologías que han sido empleados para el desarrollo del proyecto.

  - **IEC 61131-3, ISA88, GEMMA (GRAFCET) e ISA 101**: Incluye las normativas y guías estándar que han sido empleadas para el diseño de la programación asociada al proyecto.
  - **PLCnext**: En este capítulo se hace una descripción de la tecnología PLCnext de Phoenix Contact, haciendo hincapié en sus ventajas y motivos de aplicación al proyecto.
  - **OPC UA**: Se desarrolla en que consiste este protocolo de comunicación industrial, se justifica y explica cómo ha sido empleado en este trabajo.
  - **Node-RED**: Incluye una descripción de este entorno de programación y su función en este trabajo.
  - **Servicios en la nube**: Este capítulo trata de los servicios en la nube y cuáles han sido empleados en este proyecto.
- **Configuración**: Se hace una explicación detallada de todos los pasos de configuración a ejecutar para dejar operativos antes de programar todos los sistemas empleados en el desarrollo.
- **Programación**: Este capítulo desarrolla en profundidad de toda la estructuración de programación seguida en todos los entornos empleados.
- **Manual de usuario**: Describe los pasos a seguir para la utilización del sistema tras el arranque del proceso simulado y las funcionalidades implementadas.
- **Planificación**: Descripción de las tareas realizadas a nivel temporal de manera textual y gráfica.
- **Presupuesto**: Desglose de costes asociados al desarrollo del proyecto.
- **Conclusiones y futuras ampliaciones**: Pensamientos finales extraídos de la implementación realizada y sugerencias de futura implementación.
- **Referencias**: Elementos externos al proyecto que han ayudado al desarrollo del mismo.

2. Descripción de planta a automatizar

Como base para el desarrollo de este trabajo se ha tenido en cuenta un proceso o planta industrial relativamente sencilla y que sirve de hilo conductor para el diseño e implantación de una solución de automatización avanzada. Sobre la idea simplificada de esta instalación se aplican metodologías de diseño, herramientas de desarrollo e integración orientadas a facilitar la implantación de funcionalidades asociadas a la industria 4.0 y que en esencia constituyen los pilares de este trabajo.

2.1.- DESCRIPCIÓN DE LA PLANTA

Como planta ejemplo de proyecto se ha considerado un sistema que permite la preparación de un conjunto de piezas y que se introducen en un horno en el que son sometidas a un proceso de control de temperatura. Dicho horno industrial constaría de tres partes de esencial distinción y reconocimiento puesto que tenerlas presentes como elementos independientes ayudará mucho al lector/usuario a entender el procedimiento de programación y diseño basados en ISA88, en el cual se incidirá con detalle en siguientes apartados. Antes de proceder a la descripción de los mismos, es importante destacar que nos referiremos a los tres elementos diferenciales como módulos o módulos de equipamiento para orientar y entender mejor la terminología empleada durante la programación.

Los tres elementos (módulos de equipamiento) claramente identificados en el sistema son los siguientes:

1) **Alimentador**: Módulo encargado de la introducción/desplazamiento del volumen de materia prima (término al que a partir de ahora se referirá como “piezas”) a calentar deseado, desde la zona de disposición de piezas hasta el siguiente módulo conocido como Cargador.
2) **Cargador**: Módulo de equipamiento del sistema encargado de recibir el volumen deseado de piezas por parte del alimentador para tanto cargar como descargar en el horno (el cual realiza las tareas principales de calentamiento y enfriamiento), y a su vez actuar también como puerta hermética de cierre que impide que se pierda calor

del interior del horno mientras aplica la tarea de regulación de temperatura sobre las piezas.

3) **Horno**: Módulo de equipamiento en el que se produce el calentamiento y enfriamiento de la pieza. Dentro del horno se encuentra la resistencia y ventilador encargados de desencadenar estos procesos y de realizar la correspondiente regulación de temperatura asociada al proceso.

Un esquema general de cómo estaría compuesto todo el sistema automatizado se puede ver en la siguiente Figura 2.1.

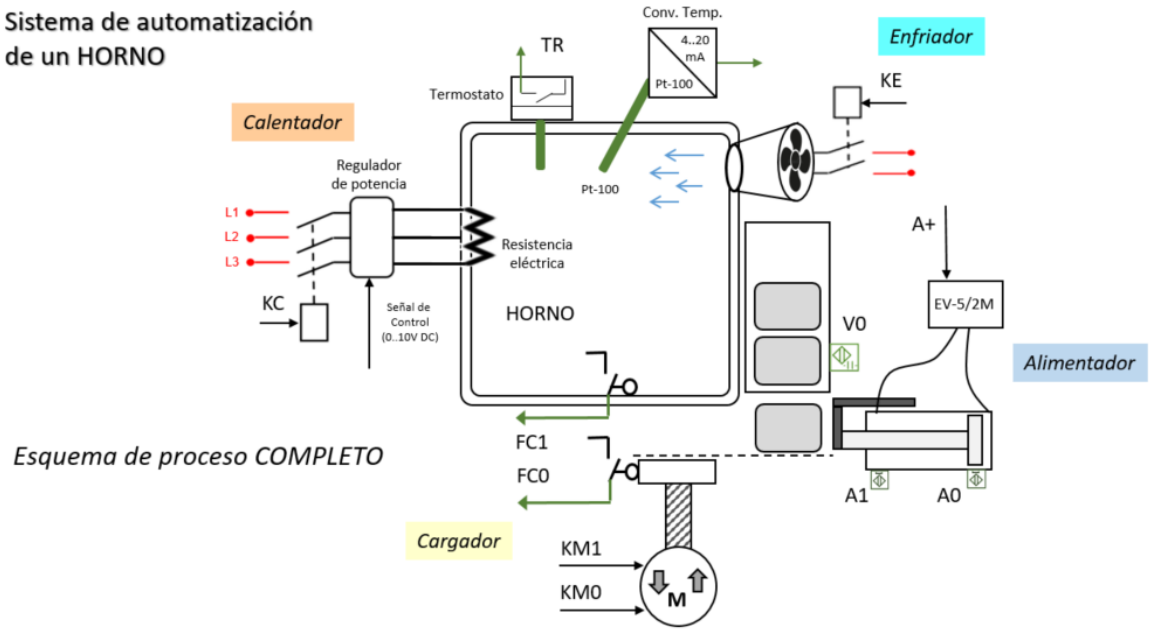

Figura 2.1 Esquema general de la planta desarrollada en el trabajo.

De forma más detallada se describen los componentes principales de la parte operativa y sus relaciones, al objeto de identificar aspectos tecnológicos que son trascendentes para el diseño y desarrollo de la automatización de este sistema. Puesto que no se trata de un sistema real se consideran supuestas las características de los componentes tecnológicos asociados a la instrumentación (sensores, actuadores, interfaces de señal, …) aunque muy habituales en este tipo de desarrollos.

El **horno** consiste en un recinto que puede ser calentado por acción de una resistencia eléctrica que inicialmente se acciona por medio de un contactor (KC). Para su enfriamiento se dispone de un ventilador monofásico que se activa por medio del contactor correspondiente (KE). También será posible instalar, en serie con el contactor, un regulador de potencia gobernado por una señal analógica (0..10 V DC) que puede ser generada como una salida analógica del PLC, para aplicación de potencia calorífica del 0..100%.

La medida de temperatura en el horno se efectúa, por una parte, con el termostato (TR) y por otra, usando una sonda Pt-100 conectada a un convertidor que genera una salida en corriente (4..20mA), que será una entrada analógica en el controlador. Las temperaturas de ajuste de la sonda y el convertidor, en concordancia con el proceso, se considera entre -50ºC y 250ºC.

Las piezas se llevan dentro del horno por medio de un **cargador** que está basado en un actuador eléctrico de tornillo, que ejecuta el movimiento de la plataforma sobre las que se sitúan las piezas. Las señales KM1 y KM0 permiten el avance y retroceso respectivamente, mientras que las posiciones finales están marcadas por FC1 y FC0 que son sendos finales de carrera. A través de unas células de carga conectadas a un sensor de peso, se genera una señal analógica en el rango 4..20mA, para una medida de entre 0 y 400Kg.

Las piezas en el cargador son depositadas por otro elemento: el alimentador. Este consiste en un cilindro neumático de doble efecto que saca las piezas de un contenedor de petaca que las va dejando caer en cada secuencia de alimentación. El cilindro se gobierna por una electroválvula 5/2 monoestable (señal A+) disponiendo de dos detectores magnéticos (A1 y A0) que permiten determinar la posición de avance o retroceso del actuador. A su vez, el contenedor incluye un detector capacitivo (V0) para conocer si se ha quedado sin piezas.

El sistema debe estar dotado de un conjunto de elementos correspondientes a un panel de mando físico y/o a través de una interface gráfica mediante HMI o SCADA.

2.2.- FUNCIONAMIENTO DE LA PLANTA

A partir de los componentes disponibles, sus características tecnológicas y capacidades, así como las relaciones entre ellos es posible plantear el proceso que debe tener lugar para el desarrollo de la producción y que consiste en la elaboración de un producto terminado (Pedido ID) a partir de una receta que tiene las siguientes características:

- **SP_Peso** (en Kg de piezas): Cada pieza tiene un peso determinado, y el alimentador debe proveer al cargador de un número de piezas cuyo peso total supere el especificado en este parámetro (SP_Peso).
- **SP_Temp** (en ºC): Es la referencia de temperatura a la que debe mantenerse el horno durante el proceso en el que se está realizando el tratamiento de piezas.
- **TCT** (en s): Es el tiempo de control de temperatura que debe mantenerse el horno a la temperatura de referencia (SP_Temp) mientras hay piezas en tratamiento.

La Figura 2.2 muestra los datos para un supuesto PEDIDO_1 a realizar con el sistema donde se especifican valores para los parámetros indicados anteriormente; el sistema en producción deberá proveer de 4 piezas que, suponiendo cada pieza pesa 40 Kg superarán SP_Peso (160 Kg) y han de someterse a un control térmico de 125 ºC durante minuto y medio (90 s).

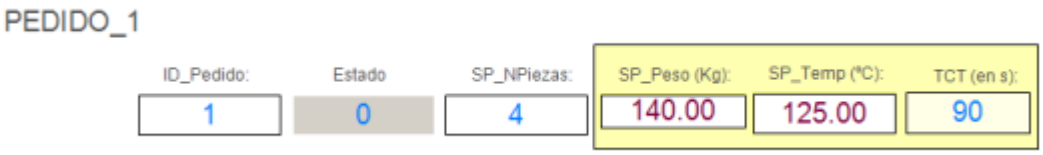

Figura 2.2 Datos asociados a la petición de pedido.

Esta ejecución de la producción se puede configurar para que realice un único pedido o para que trabaje de forma continua con la misma implementación de receta. Y la forma de terminar también puede ser a ciclo terminado o de forma instantánea. El sistema debe proporcionar estos mecanismos para una producción más flexible.

Aparte de este funcionamiento en producción (automático), es habitual disponer de modos de funcionamiento para poder comandar los elementos de forma independiente (manual), bien sea con operaciones muy simples sobre los elementos de control o de manera más sincronizada de varios elementos conformando los servicios de los módulos de equipamiento. En cualquier caso, estas operaciones son muy socorridas en los ajustes del sistema para llevarlo a condiciones iniciales tras un fallo, la prueba y verificación de accionamientos y servicios, calibraciones y testeo de elementos, etc.

También es necesario tener en cuenta posibles circunstancias y anomalías que se pueden producir en la planta debido a necesidad de actuación por parte del operario –por ejemplo, para reponer materia prima- o debido a errores, defectos o emergencia (pulsación de seta). El sistema debe actuar de forma correcta procurando la seguridad de las personas y finalmente de las máquinas haciéndolos evolucionar hasta estados seguros antes de proceder de nuevo a la puesta en servicio.

2.3.- ORGANIZACIÓN DE COMPONENTES DE LA PLANTA

Debido al planteamiento del método ISA 88 empleado para desarrollar este proyecto, el cual se mencionará posteriormente en el presente escrito en el apartado **“3.1 IEC 61131-3, ISA88, GEMMA (GRAFCET) e ISA 101 (HMI)”**, es conveniente detallar la estructuración asociada a los módulos de equipamiento previamente introducidos.

Teniendo en cuenta conceptos básicos de ISA88, el sistema puede considerarse que implementa un modelo físico que como nivel superior consiste en una célula para el tratamiento de piezas en un horno industrial. En la célula se distinguen dos unidades diferentes. En la unidad 1 se encuentra toda la parte encargada de la preparación de piezas, distinguiendo los previamente mencionados, alimentador y cargador de piezas, mientras que en la unidad 2 se reconoce como el sistema de control de temperatura donde se localiza el propio horno. Teniendo en cuenta lo indicado se puede ver una estructura más gráfica en la siguiente Figura 2.3.

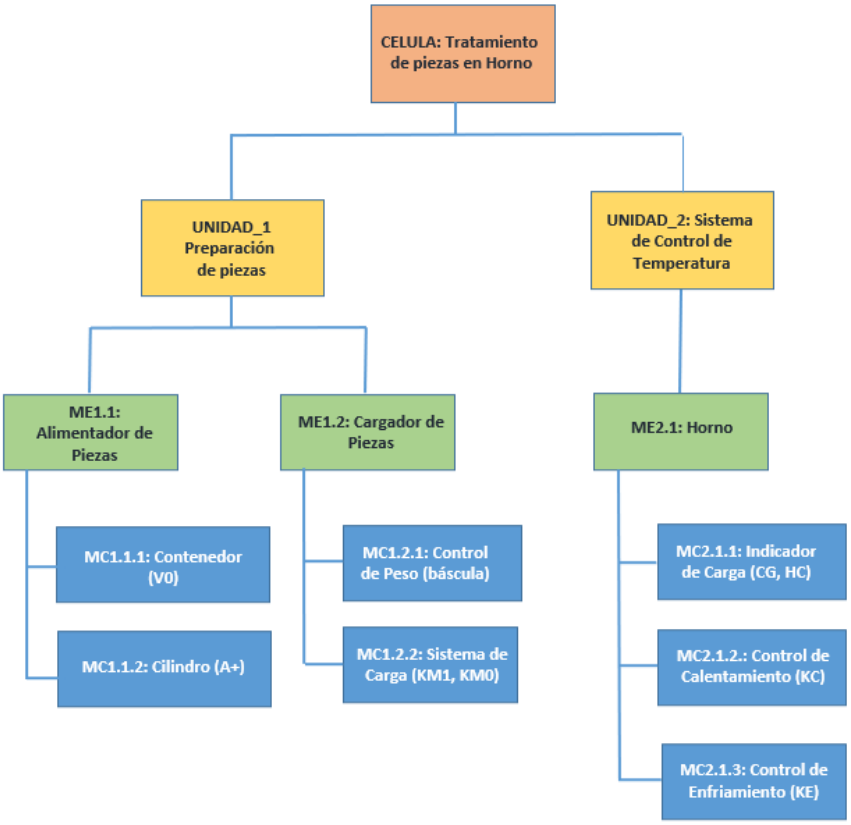

Figura 2.3 Representación esquematizada de los componentes asociados a la planta descrita.

2.4.- DESCRIPCIÓN GENERAL LA PROGRAMACIÓN PARA CONTROL DEL SISTEMA

Se ha realizado una descripción de los elementos que conforman la planta y sus relaciones, así como el funcionamiento resumido que se espera, permitiendo disponer de una organización de elementos aplicando el modelado ISA88. Esto permite una estructuración de la programación del control para describir tanto los módulos de equipamiento, como los servicios asociados, así como los defectos, avisos, alarmas y emergencias que se consideren en cada caso de importancia.

Por tanto, para cada módulo de equipamiento definido:

1) **Alimentador:** Se generará una estructura de programación la cual contenga los servicios de: sacar una pieza, sacar un valor específico de peso (en piezas) y vaciar el alimentador de piezas. Como defecto se tendrá en cuenta que se produzca una incongruencia entre los sensores A0 y A1. El aviso reflejará que el alimentador estará vacío (no hay piezas). Alarma será generada cuando se produzca un fallo en la detección de la extracción del cilindro (es decir, no se active A1) y finalmente emergencia como en los demás módulos de equipamiento se generará cuando se pulse la seta de emergencia.

2) **Cargador:** Los servicios que se asociarán a este módulo de equipamiento consisten en: cargar las piezas (ascender con el cargador hasta detectar FC1) y descargar piezas (descender el cargador detectando FC0). En cuanto al defecto se generará cuando se supere el peso máximo establecido. Aviso se generará cuando el cargador se encuentre vacío. Alarma se producirá cuando se producto un fallo en el movimiento del cargador, ya sea no detectando FC1 cuando se realiza la carga o la no detección de FC0 cuando se produce la descarga. La emergencia, como ya se ha dicho, vendrá condicionada por la seta de emergencia.

3) **Horno:** Los servicios que se implementan para el horno son: calentamiento del horno por medio de activación de KC, enfriamiento del horno, actuando sobre KE, realizar el control de temperatura establecido según la receta (parámetro de temperatura y tiempo de control) dada por el usuario y finalmente realizar la calibración de los valores de temperatura de regulación. Como defecto se detectará que el horno no calienta o no enfría. El aviso será mostrado siempre que el horno se encuentre "frío" (por debajo de 50 (ºC)). La alarma se generará cuando haya algún fallo en los contactores o en la regulación de temperatura. La emergencia como ya se ha descrito.

El presente desarrollo del programa de control en base a estas consideraciones, se ha culminado finalmente con dos versiones de programa asociados al controlador, una implementada en Codesys y otra versión en PLCnext Engineer, según:

- Programa **Codesys**:

  Codesys_Gabriel_Programa_HORNO.project

- Programa **PLCnext Engineer**:

  PLCnext_Gabriel_HORNO_PLCnext_2022.6.0_Proficloud_funcionando.pcwex

La programación en PLCnext Engineer ha servido como implementación definitiva y se ha descargado la aplicación en el controlador físico AXC F 2152.

A su vez, complementando la solución con PLCnext, también se dispone de dos programas independientes desarrollados en Node-RED que complementan las funcionalidades descritas y que permiten tanto el visionado de datos en un Dashboard implementado en Node-RED como permitir el enlace de variables entre el programa localizado en el controlador físico y la plataforma Cloud seleccionada, también serán tratados con mayor detalle en futuros apartados del presente documento.

- **Node-RED** del controlador físico:

  Node_RED_PLCnext_Gabriel_Horno_wAzure.json

- **Node-RED** del entorno Cloud:

  Node_RED_CloudAzure_Gabriel_Horno.json

3. Diseño detallado del sistema

En este capítulo se va a realizar una descripción en mayor detalle de todos los elementos y tecnologías que han sido empleadas para desarrollar el trabajo, justificando su aplicación y la manera de llevarlo a cabo. Aunque con el trascurso de los apartados siguientes se vaya construyendo la conceptualización completa de todo el sistema generado, convendría visualizar la Figura 3.1, para tener una idea general sobre la estructura general del mismo.

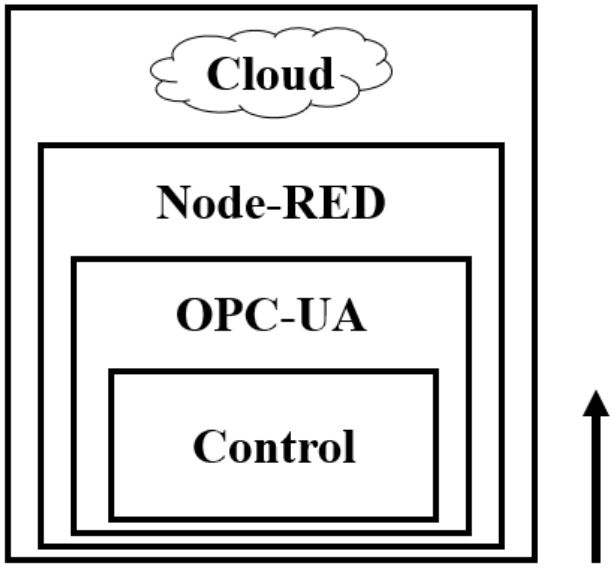

Figura 3.1 Estructura general del sistema.

Se partirá del desarrollo del control de la planta para implementación de las especificaciones funcionales consignadas, y a continuación se incorporarán los datos de control al servidor OPC-UA. Desde el servidor OPC-UA, Node-RED tomará los datos, realizará una representación local de los mismos y establecerá un puente de conexión con la plataforma Cloud Desde la plataforma Cloud se podrán representar de nuevo los datos, permitiendo disponer de un punto de acceso global a la información.

3.1.- IEC 61131-3, ISA88, GEMMA (GRAFCET) E ISA 101 (HMI)

Descrita la planta y la idea que se ha seguido para desarrollar la programación de la misma, en este capítulo se describen brevemente las diferentes normativas, metodologías y guías de apoyo a la programación que se han tenido en cuenta para el diseño y creación de los distintos programas.

Si bien es cierto que podría enumerarse alguno más, el conjunto de elementos esenciales que se han tenido en cuenta podrían ser resumidos en: la normativa IEC 61131-3, la metodología ISA88 (la cual es altamente importante tener siempre en cuenta en el desarrollo del trabajo), la guía GEMMA y la norma de diseño de HMIs (ISA 101).

3.1.1- Introducción a IEC 61131-3

Este apartado está reservado al estándar sobre autómatas programables industriales, y más concretamente a la parte 3 relativa a los lenguajes de programación, IEC 61131-3. Desde hace 30 años esta normativa se está convirtiendo en la base para programar muchos de los PLCs disponibles en el mercado que, aunque no lo cumplan al 100% sí que se ha apreciado un notable incremento en todos los ámbitos y por distintos actores con importantes ventajas: fabricantes, desarrolladores, integradores, formadores, etc.

La tercera parte de la normativa IEC 61131 hay que indicar que se compone por dos partes distintas como se indica en la Figura 3.2:

- **Elemento Comunes**: Donde se encuentran: 1) Tipos de datos (BOOL, INT, REAL, BYTE, WORD, DATE, TIME-OF-DAY y STRING); 2) Variables, las cuales pueden ser definidas como local o globales (VAR_GLOBAL); 3) Configuración, recursos y tareas; 4) Unidades de Organización de Programa y 5) Gráfico Funcional Secuencial (SFC).
- **Lenguajes de Programación**: En los cuales se definen 4 lenguajes de programación normalizados y que se dividen en dos de tipo literal y dos de tipo gráfico. Los literales son: 1) Lista de instrucciones (IL) y 2) Texto estructurado (ST), mientras que en los

gráficos se encuentran: 1) Diagrama de contactos (LD) y 2) Diagrama de bloques funcionales (FBD).

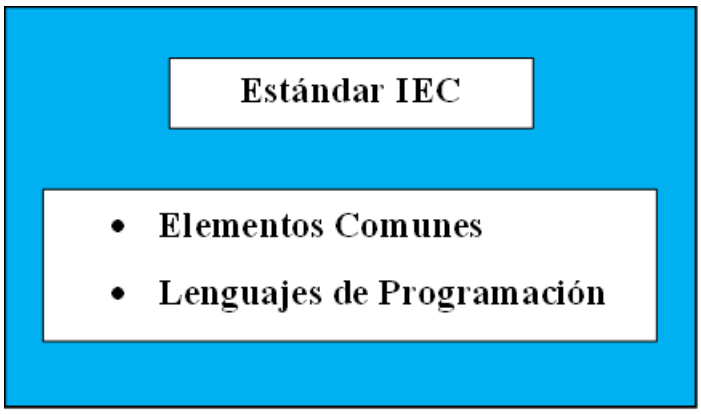

Figura 3.2 Esquema de composición del IEC 61131-3.

Incidiendo más sobre los lenguajes de programación que permiten a los usuarios desarrollar el programa de control asociado a la automatización, ha de indicarse que es posible usar simultáneamente varios de estos lenguajes para el desarrollo de dicho programa como se puede observar en la siguiente Figura 3.3.

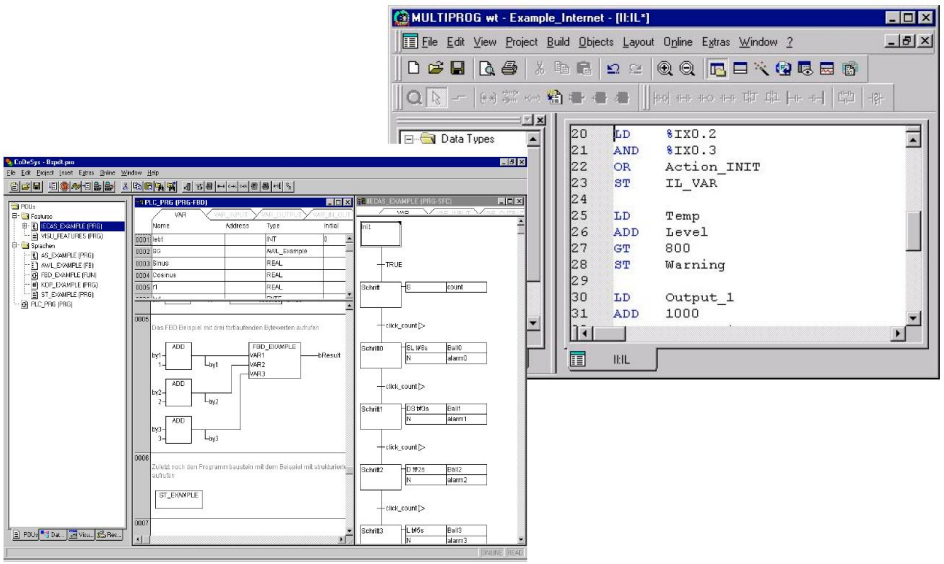

Figura 3.3 Ejemplo de operación simultanea de diferentes lenguajes de programación bajo la normativa IEC 61131-3 [3]

Al igual que se indicará en algunos de los subapartados posteriores a la normativa IEC 61131-3 (ISA88, GEMMA, ISA 101) el cumplimiento del estándar al completo es complicado, por lo que se acepta una adecuación en cuanto a su aplicación.

La edición actual del IEC 61131-3 está publicada desde 2013 lo cual indica que es una normativa sólida que se aplica correcta e intensamente en el desarrollo de procesos automatizados a nivel industrial.

3.1.2- Aplicación de IEC 61131-3 al control de la planta

En el desarrollo del proyecto se han empleado diferentes lenguajes de programación del estándar IEC 61131-3 y éstos se han llevado a cabo en dos herramientas diferentes de programación: Codesys y PLCnext Engineer.

Hay que destacar una pequeña particularidad en la programación de la planta al emplear PLCnext Engineer y es que para poder utilizar SFC (Gráfico Funcional Secuencial) se requiere una licencia específica. Esta licencia, en concreto, cuando se realice la descarga del software PLCnext Engineer desde la web del fabricante, habrá que indicar que se desea además realizar la adición de esa licencia que recibe el nombre de: “Sequential Function Chart Editor”, lo cual supone un coste adicional. En posteriores apartados se incidirá con mayor detalle sobre adición y uso de la citada licencia.

Otro elemento particular que destacar es que uno de los mayores beneficios que aporta el uso del entorno PLCnext Engineer es que permite conjugar la programación del estándar IEC 61131-3 con diversos recursos externos como ha sido en este caso OPC-UA y Node-RED, de manera que el sistema sea totalmente flexible a los requisitos del usuario como se puede observar en la siguiente Figura 3.4.

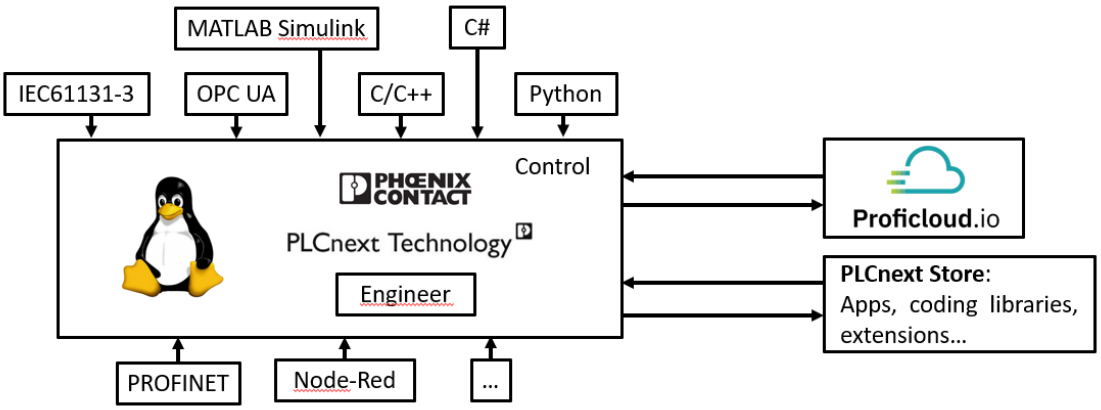

Figura 3.4 Representación de la capacidad de adaptación por parte de la tecnología PLCnext al usuario.

3.1.3- Introducción a la metodología ISA88

Otra de las metodologías aplicadas para este proyecto es la asociada a la norma internacional S88 la cual se asocia a los sistemas de fabricación por lotes con la finalidad de solventar los siguientes problemas: 1) Generar un control de lotes determinado, 2) Terminología específica para explicar el proceso por lotes, 3) Integrar e implementar en un solo programa los sistemas de control.

La normativa ISA88 proporciona conceptos, modelos y terminología suficientes y necesarios para definir los requisitos y necesidades asociados al control de las plantas de fabricación por lotes.

Este se guía por 3 preguntas y 3 respuestas: 1) ¿Cómo hacer el producto? Recetas, 2) ¿Qué herramientas son necesarias para la realización del producto? Equipamiento y 3) ¿Cómo hacer que el equipamiento funcione? Control.

Por entrar un poco más en el detalle técnico, ISA88 describe tres modelos (en su forma más básica) de la información que se debería tener en cuenta:

- Modelo de receta procedural o procedimental: Lo que se ha referido como receta y representa la información adecuada para la fabricación de un producto con unas características determinadas.

- Modelo físico o de equipamiento: Nombrado de manera más común como módulo de equipamiento y representa el equipo necesario a juntar para llevar a cabo la receta.

- Modelo de actividad o de control de procedimiento: Indicado simplemente como control, el cual ya genera una indicación del conjunto de acciones (generalmente se referirán como servicios de los módulos de equipamiento) ordenadas asociadas a los módulos de equipamiento para poder completar la receta solicitada por el usuario

En el fondo se resume en que se ha de buscar independizar cada una de las partes que componen el sistema completo (los diferentes módulos de equipamiento) y orientarlas como elementos diferenciados que sean capaces de operar independientemente unos de otros por medio de llamada a servicios, que con el fin de obtener un producto con unas características concretas (receta) operen de manera coordinada (control).

Es una programación orientada a objetos, pero incrustada dentro del paradigma industrial. La particularización seguida en el presente proyecto se incidirá con mayor detalle en el siguiente apartado.

3.1.4- Aplicación de ISA88 al sistema automatizado

Como ya se venía introduciendo, en el presente proyecto se ha empleado la metodología ISA88, pero hasta el momento solo se ha indicado a nivel teórico, pero no como se ha aplicado sobre la planta.

Sin embargo, sí que es cierto que debido a como se ha tratado la descripción de la planta, ya puede apreciarse como se ha realizado la aplicación de la metodología. Primero, el control busca desarrollar un proceso completo en cadena de toma de piezas, carga de las mismas a un horno, aplicar una regulación de temperatura y su posterior descarga. Para ello,

se requiere que diferentes módulos de equipamiento funcionen de manera sincronizada, teniendo que proporcionar unas recetas adecuadas para obtener los productos finales con las características deseadas.

En este caso se distinguen perfectamente 3 módulos de equipamiento: 1) Alimentador, 2) Cargador y 3) Horno. Los 3 módulos pueden operar perfectamente de manera independiente. El alimentador podría simplemente impulsar piezas por orden del usuario sin requerir la existencia de ninguno de los otros módulos. Una teoría semejante se puede aplicar en los otros dos casos. El cargador podría subir y bajar por sí mismo y el horno podría calentar o enfriar también de manera autónoma.

Una vez diferenciado el equipamiento del que se dispone y lo que se pretende conseguir, aplicando las 3 preguntas al ejemplo. 1) ¿Cómo hacer el producto? Receta: Calentar x (Kg) de piezas, 2) ¿Qué herramientas son necesarias para la realización del producto? Equipamiento: Alimentador, Cargador y Horno y 3) ¿Cómo hacer que el equipamiento funcione? Control: Sincronización de estos 3 elementos independientes para que operen de manera ordenada por medio de llamadas a servicios.

Aplicando esta conceptualización y resultando de gran ayuda siempre antes de realizar la programación. Se pueden llegar a desarrollar unas tablas asociadas a cada módulo de equipamiento que sirvan como guía visual a la hora de generar el programa. En este caso:

1) El **Alimentador** quedaría resumido en el diseño visualizado en la Figura 3.5. Donde se verán reflejadas todas las entradas al módulo de equipamiento, al igual que las salidas y referenciado a la manera de proceder del módulo de equipamiento.

Entradas	Módulo de Equipamiento - ALIMENTADOR	Salidas
Del HMI		**Al HMI**
- Configuración de Modo (Man/Auto)		- Modo de operación
- Acuse / Rearme / Marcha / Paro		- Información Npiezas suministradas
- Tiempo supervisión actuador (Tsup en s)		- Información funcionamiento y estado
- Seta de emergencia		
		A otros módulos
Parámetros de receta		- Peso suministrado de piezas
- Cantidad peso a sacar (SP_Peso en Kg)		
- Peso de la pieza (en Kg)		
De otros módulos		
- Iniciar Pedido		
- Finalizar		
		Señales de campo
Señales de campo		- Detector alimentador vacio (V0)
- Sacar pieza (A+)		- Detector cilindro en retroceso (A0)
		- Detector cilindro en avance (A1)
	Secuencia de operación (en automático)	
	- Comprobar que hay piezas	**Alertas**
	- Sacar pieza / Verificar	- Falta pieza en alimentador
	- Repetir hasta SP_Peso	- Error en cilindro
		- Error de lectura
	Manejo de excepciones	
	- Retroceder cilindro y parar	

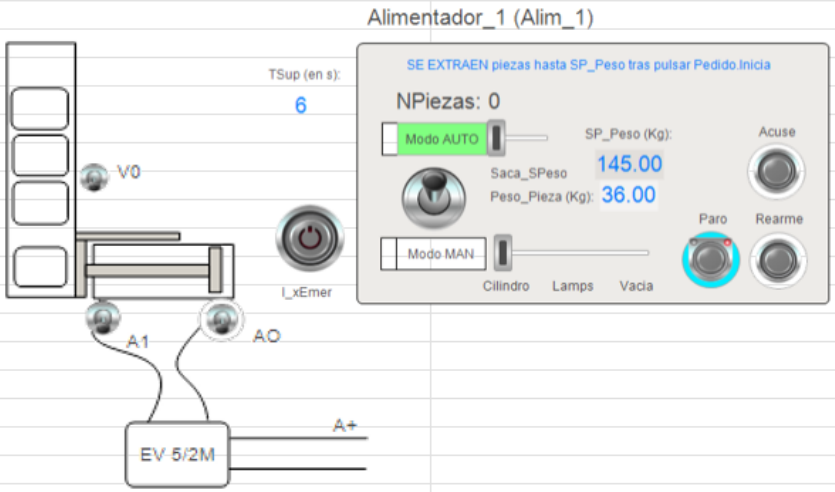

Figura 3.5 Representación gráfica de la fase de diseño en ISA88 del módulo de equipamiento: Alimentador.

2) Lo mismo sucedería para el caso del **Cargador** representador en la Figura 3.6.

Entradas	Módulo de Equipamiento - CARGADOR		Salidas
Del HMI			**Al HMI**
- Configuración de Modo (Man/Auto)			- Modo de operación
- Acuse / Rearme / Marcha / Paro			- Información Npiezas en el cargador
- Tiempo supervisión actuador (Tsup en s)			- Información del Peso en el cargador (en Kg)
- Seta de emergencia			- Información funcionamiento y estado
			- Indicación vaciar cargador
Parámetros de receta			
- Cantidad peso a cargar (SP_Peso en Kg)			**A otros módulos**
			- Peso suministrado de piezas
			- Peso actual del cargador
De otros módulos			
- Iniciar Pedido			
- Finalizar			
			Señales de campo
			- Detector cargador en avance (FC1)
Señales de campo	**Secuencia de operación (en automático)**		- Detector cargador en retroceso (FC0)
- Cargar horno (KM1)	- Comprobar peso OK		- Sensor de peso en el cargador (Peso)
- Descargar horno (KM0)	- Llevar carga al horno		
	- Esperar fin de proceso en horno		
	- Descargar horno		**Alertas**
			- Error durante la carga del horno
	Manejo de excepciones		- Error durante la descarga del horno
	- Parar movimiento del cargador		- Peso máximo excedido
			- Error de lectura

Figura 3.6 Representación gráfica de la fase de diseño en ISA88 del módulo de equipamiento: Cargador.

3) En la misma medida se ha procedido con el **Horno** representado sobre la Figura 3.7.

Entradas	Módulo de Equipamiento - HORNO		Salidas
Del HMI			**Al HMI**
- Configuración de Modo (Man/Auto/Test)			- Modo de operación
- Acuse / Rearme / Marcha / Paro			- Información Npiezas en el cargador
- Tiempo supervisión actuadores (Tsup_Acc en s)			- Información de la temperatura del horno (en ºC)
- Tiempo supervisón calentamiento (Tsup_Cal en s)			- Información del peso de las piezas (en Kg)
- Tiempo supervisón enfriamiento (Tsup_Enf en s)			- Información funcionamiento y estado
- Seta de emergencia			
			A otros módulos
			- Temp actual del horno (Temp en ºC)
Parámetros de receta			
- Temp. Calentamiento (SP_Temp en ºC)			
- Tiempo en calentamiento (TCT en s)			
- Temperatura horno frio (TE en ºC)			
			Señales de campo
			- Verificación de carga en el horno (CG)
De otros módulos	**Secuencia de operación (en automático)**		- Final de carrera de horno cerrado (HC)
- Iniciar Pedido	- Esperar carga en horno cerrado y piezas dentro		- Termostato Temp. Máxima (TR)
- Finalizar	- Calentar hasta SP_Temp (ºC)		- Sensor de temperatura (Temp)
	- Mantener SP_Temp durante TCT (s)		
	- Activar enfriamiento hasta TE (ºC)		
Señales de campo	- Finalizar acción en horno, descargar		**Alertas**
- Activar resistencia de calentamiento (KC)			- Error durante calentamiento
- Activar ventilador de enfriamiento (KE)			- Error durante enfriamiento
	Manejo de excepciones		- Temperatura máxima alcanzada
	- Parar calentamiento		- No hay piezas en el horno
	- Parar enfriamiento		- Horno abierto
			- Error de lectura

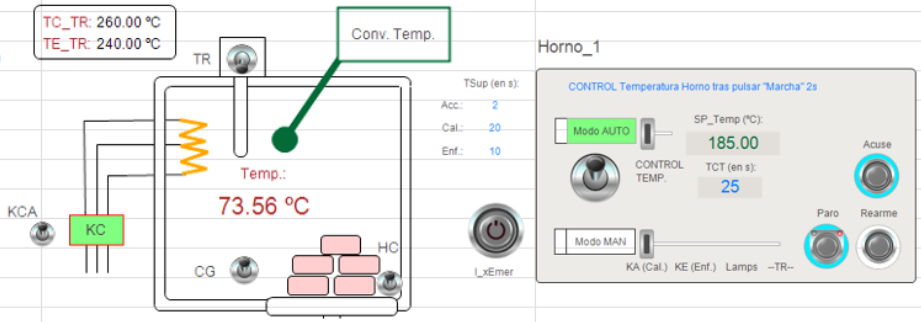

Figura 3.7 Representación gráfica de la fase de diseño en ISA88 del módulo de equipamiento: Horno.

3.1.5- Introducción a metodología GEMMA

Una vez ya se ha introducido la normativa ISA88, es conveniente exponer ligeramente los aspectos más representativos de la guía GEMMA.

Se trata de una representación organizada de todos los modos o estados de Marchas y Paradas que se pueden encontrar en un proceso de producción automatizado y orienta sobre los saltos y transiciones que pueden darse de un estado a otro. Fue desarrollada por la agencia ADEPA (Agence nationale pour le Develomepent de la Production Apliquée a l'industrie).

Para la automatización de una máquina es conveniente prever todos los estados posibles: funcionamiento manual o semiautomático, paradas de emergencia, puesta en marcha…

Básicamente ha de tenerse en cuenta que en el sistema puede encontrarse en tres situaciones: 1) Funcionando (Producción); 2) Parado o 3) En defecto (El sistema se encuentra con algún tipo de fallo o error durante el proceso de producción o de funcionamiento).

El gráfico asociado a GEMMA muestra cuatro posibles situaciones (control sin alimentación (PZ), funcionamiento (F), parada (A) y defecto (D)) como puede verse en la siguiente Figura 3.8.

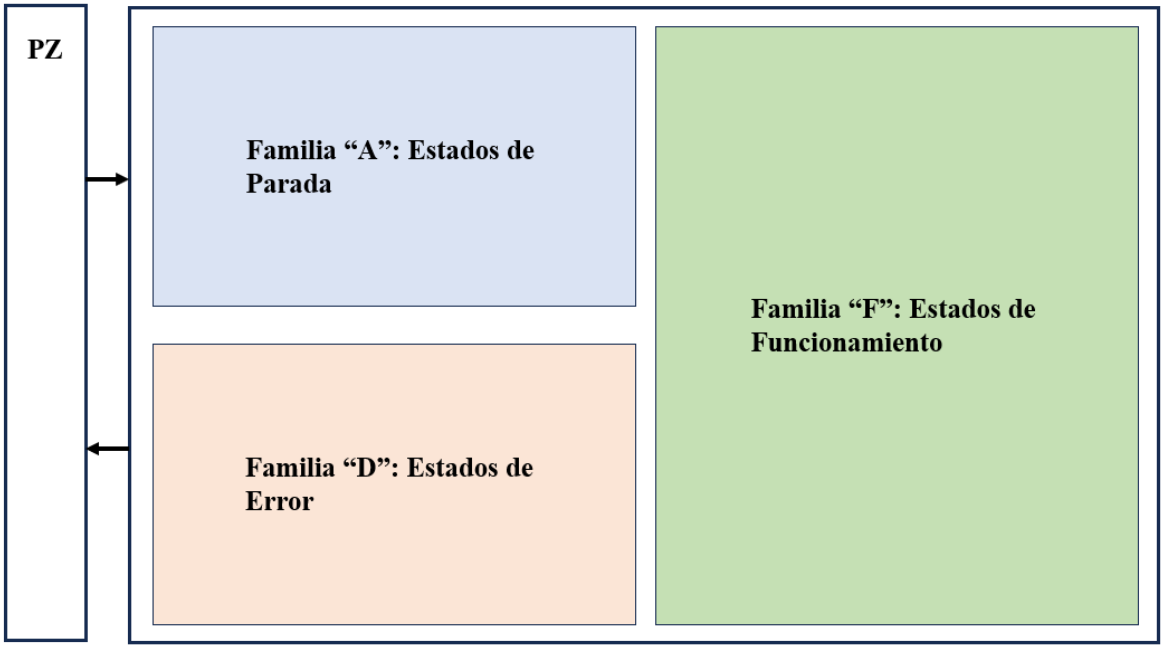

Figura 3.8 Esquema general de un GEMMA.

3.1.6- Aplicación de GEMMA (GDMMA)

El GEMMA no dispone de una aplicación exclusiva para todos los sistemas, sino que tiene cierta flexibilidad cuando se realiza la aplicación a un caso particular y a partir del cual procederá a denominarse GDMMA.

En este proyecto en particular, el GDMMA ha contemplado los siguientes estados de situación de planta:

1) Estados de Parada (A): A1: Paro en estado inicial, A2: Solicitud de Paro a fin de ciclo, A5: Preparación de Arranque después de fallo, A6: Inicialización de la Parte Operativa.
2) Estados de Funcionamiento (F): F1: Producción normal, F2: Marcha de preparación, F3: Marcha de finalización, F4: Verificación en desorden, F5: Verificación en orden, F6: Marcha de test.
3) Estados de Defecto (D): D1: Parada de emergencia, D2: Diagnóstico y/o tratamiento de fallo/defecto, D3: Producción a pesar de fallo/defecto.

Los estados mencionados se pueden observar de manera gráfica en la siguiente Figura 3.9, la cual representa el GDMMA, también implementado en el programa de Codesys desarrollado.

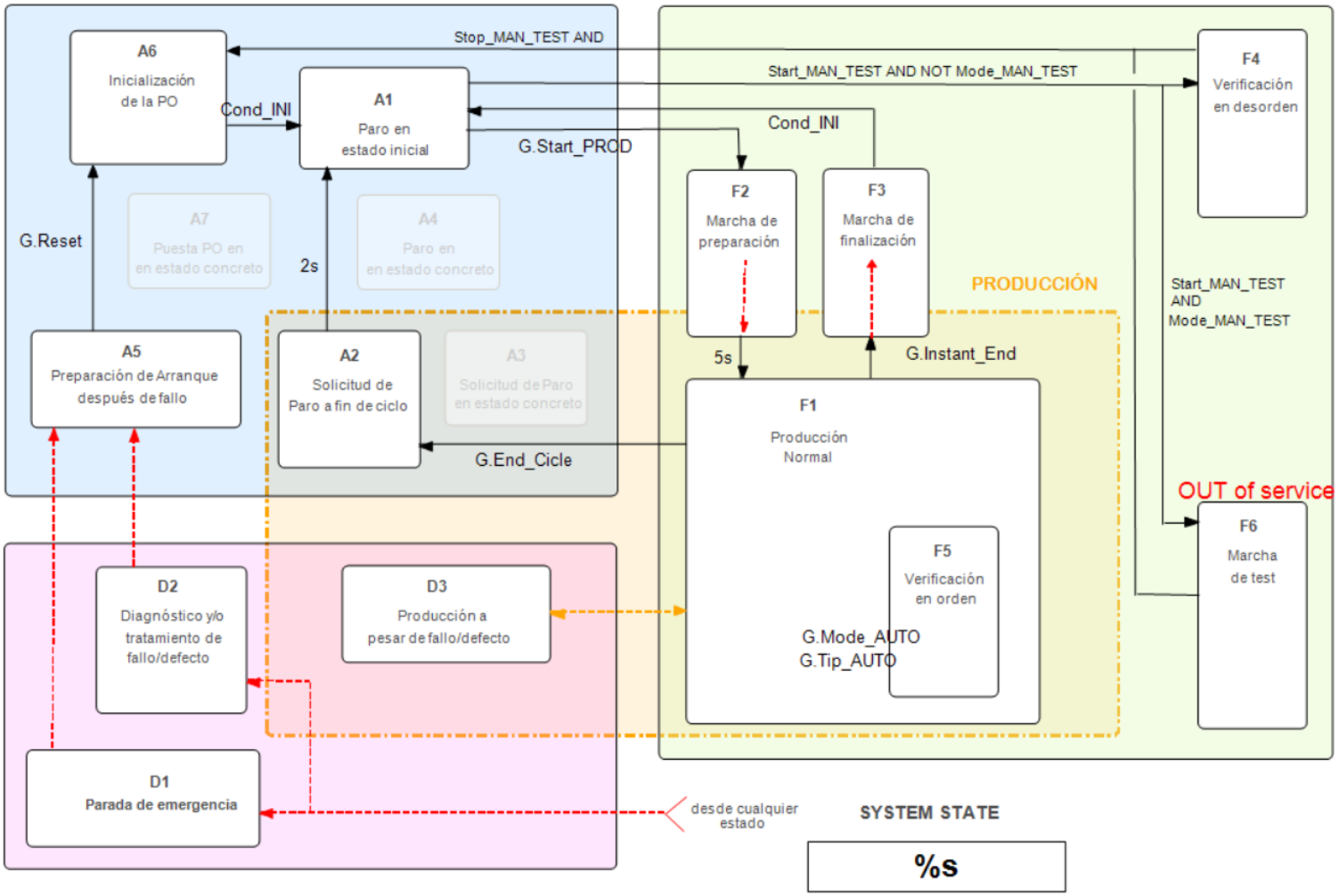

Figura 3.9 GDMMA particular de la aplicación implementada.

Describiendo ligeramente el GDMMA desarrollado, se puede tomar como ejemplo una transición entre estados de parada una vez iniciado el sistema. Inicialmente el sistema, partirá del estado A5 (Preparación del Arranque después de fallo), una vez se haga un reset general (G.Reset), el sistema evolucionará hacía el estado A6 (Inicialización de la Parte Operativa) y una vez se alcancen las condiciones iniciales (Cond_INI) el sistema se quedará en A1 (Paro en estado inicial) a la espera de que el usuario decida que el sistema, o evolucione a F4 (Verificación en desorden) o hacia ejecución de una receta de producción, iniciando el paso hacia F1 (Preparación normal).

3.1.7- Introducción a ISA 101

El estándar ISA 101 establece unos estándares de terminología y normativas para el desarrollo de pantallas de explotación (HMI) para que se desarrollen de la manera más eficiente y con un estilo común.

Esta norma pretende orientar al programador a la hora de plantear, diseñar e implementar las pantallas y objetos de visualización y configuración adecuados para representar de manera fiel la operación de la planta y mostrar los datos de son de interés.

3.1.8- Aplicación de ISA 101 al proyecto

En este caso se han desarrollado diferentes pantallas de explotación en los entornos de programación Codesys y PLCnext Engineer intentando cumplir estos requisitos esenciales:

- **Fondo**: Nunca utilizar un fondo negro. Usar un fondo claro,
- **Color**: Sigue un poco la misma premisa que el fondo. Colores claros que no resulten un de un contraste muy fuerte a la vista. Exceptuando casos de peligro.
- **Secciones claras**: Tender a diferenciar bien las partes que compongan la pantalla.
- **Iconos**: Procurar recurrir al uso de iconos para referenciar ciertas tareas.
- **Dispositivos móviles**: Buscar en la medida de lo posible la adaptación a equipos móviles.

Como resultado se muestran un par de pantallas de ejemplo que se han desarrollado para las aplicaciones reseñadas (ver Figura 3.10 y Figura 3.11), que posteriormente serán objeto de una explicación más detallada a lo largo de esta documentación.

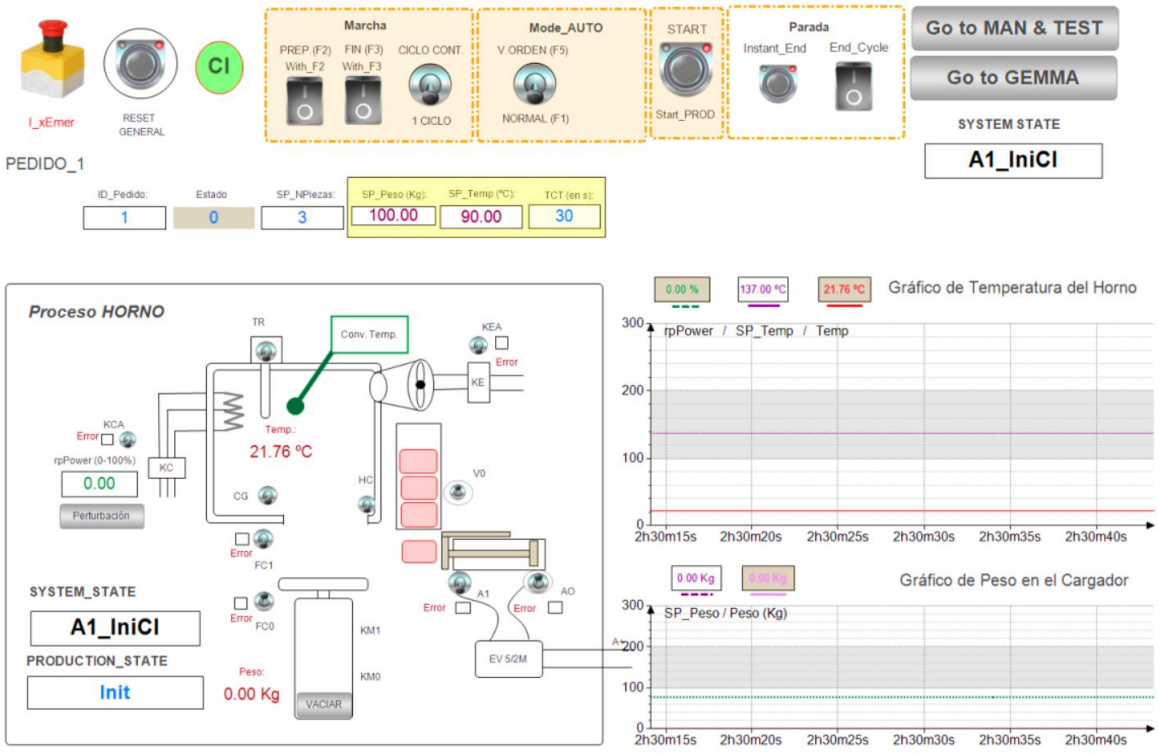

Figura 3.10 Ejemplo de pantalla de operación desarrollada en Codesys.

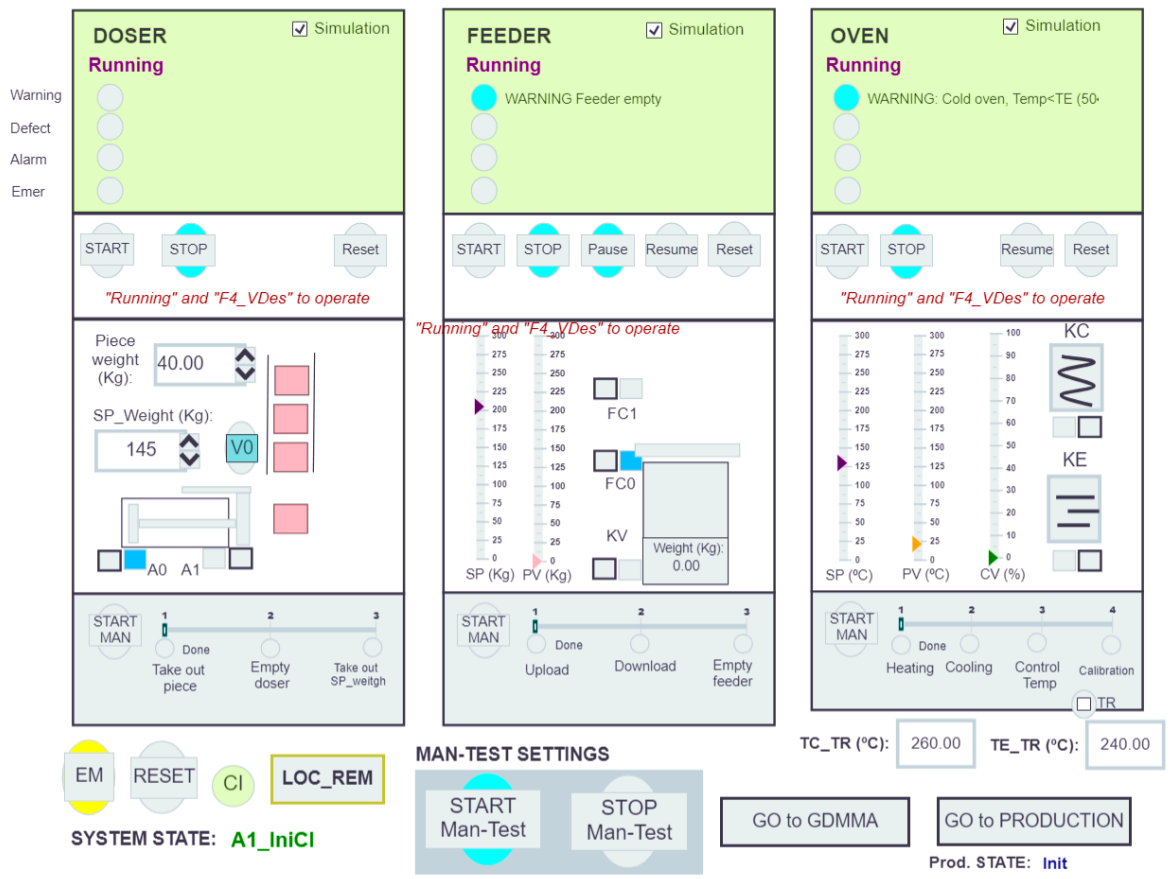

Figura 3.11 Ejemplo de pantalla de operación desarrollada en PLCnext Engineer.

También se han desarrollado algunas pantallas de explotación en el entorno gráfico de programación Node-RED, en este caso se referirá a ellos como Dashboard (debido a que es el término más común empleado para esta interacción planta-usuario en este entorno). Se han implementado tanto para el Node-RED instalado localmente en el controlador AXC F 2152, el cual se puede visualizar en la Figura 3.12, como para el Node-RED accesible a través del entorno Cloud, del cual se muestra un ejemplo en la Figura 3.13.

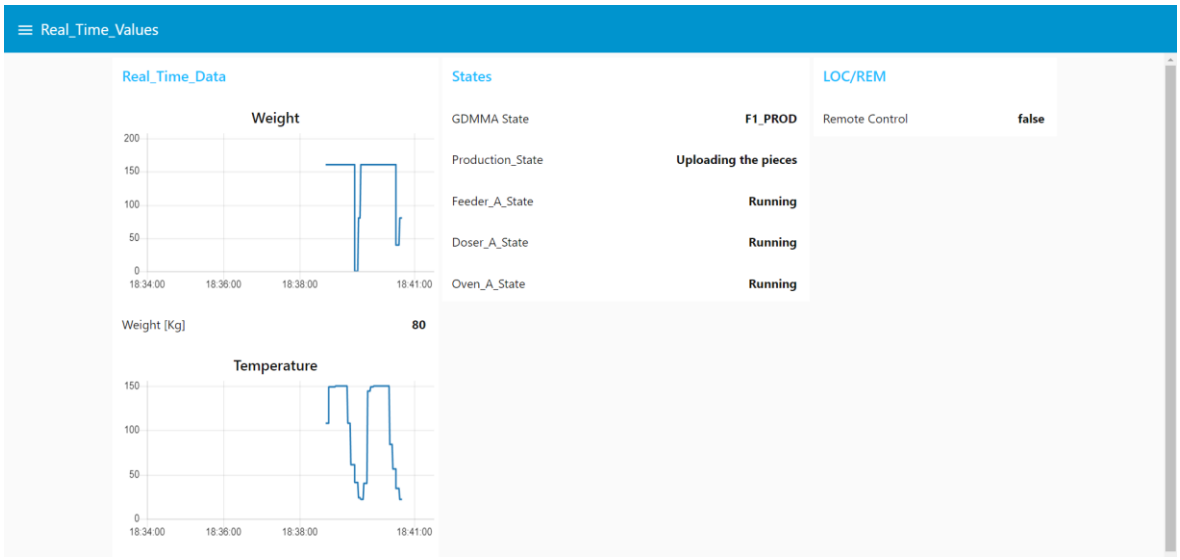

Figura 3.12 Ejemplo de Dashboard desarrollado en el controlador AXC F 2152.

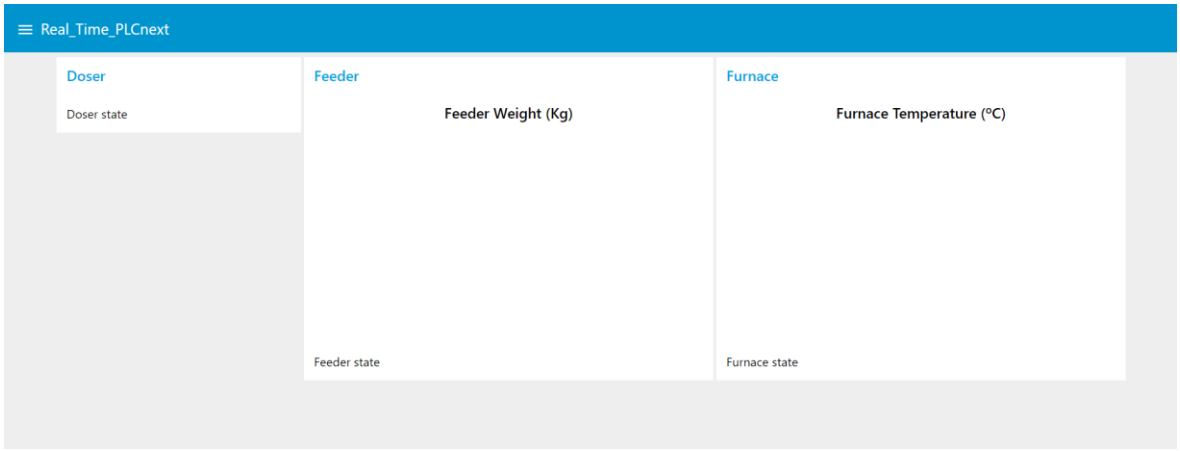

Figura 3.13 Ejemplo de Dashboard desarrollado en la plataforma Cloud.

3.2.- PLCNEXT

3.2.1- Industria 4.0

Como ya se venía introduciendo, la Industria 4.0, como se puede observar en la Figura 3.14 implica la promesa de una nueva revolución que combina técnicas avanzadas de producción y operaciones con tecnologías inteligentes que se integrarán en las empresas y organismos industriales de manera progresiva.

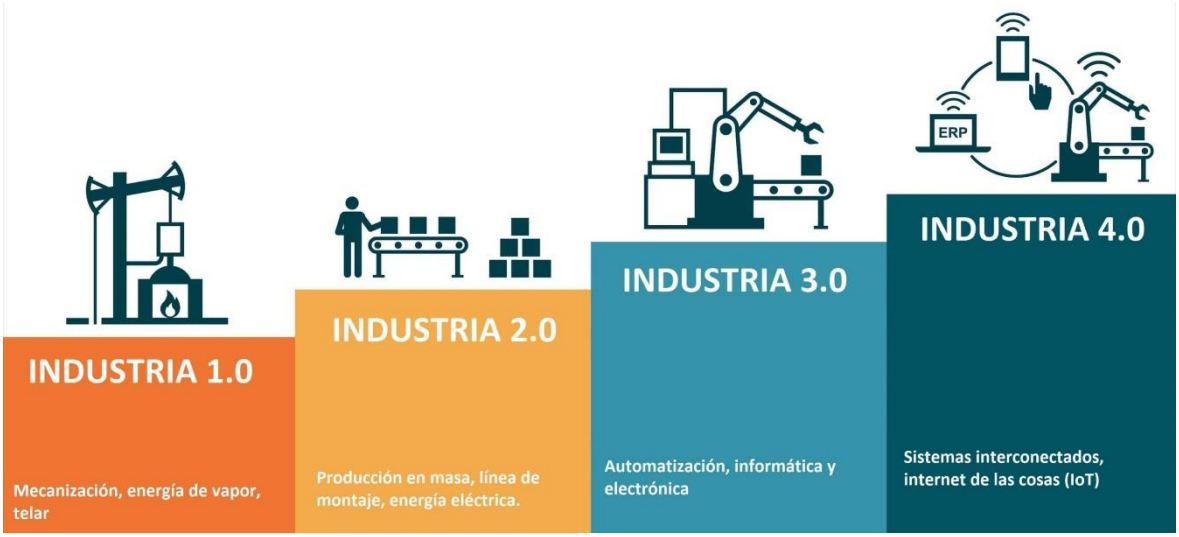

Figura 3.14 Esquema gráfico de las diferentes etapas de la evolución industrial (Tomada de la referencia [15]).

Esta revolución está marcada por la aparición de nuevas tecnologías como la robótica, la analítica, la inteligencia artificial, las tecnologías cognitivas, la nanotecnología y el Internet de las Cosas (Internet of Things [IoT]) entre otras.

Debido a esta aparición de nuevas tecnologías, la industria se ve obligada, en la medida de lo posible, a adaptar las tecnologías ya existentes a estos nuevos paradigmas y a ir introduciendo nuevos recursos tecnológicos que permitan cumplir estos requisitos.

Si bien es cierto que la Industria 4.0 es un término muy reciente y aún por implantarse de forma más extendida en la industria moderna, ya hay personas que comienzan a citar el término de la Industria 5.0 la cual aboga por buscar una industria basada en tres objetivos principales:

- Sostenibilidad: Disminuir el impacto ambiental de la industria.

- Protagonismo de la persona: Trato de los derechos humanos y fundamentales como una prioridad.

- Resiliencia: Adaptando este término como capacidad de la industria de adaptarse a condiciones económicas y sociales complicadas.

Si bien es cierto que ya se empieza a incorporar la etapa de la Industria 5.0 en el esquema general de la industria como se muestra en la Figura 3.15, este aún no es un término comúnmente mencionado en la industria moderna.

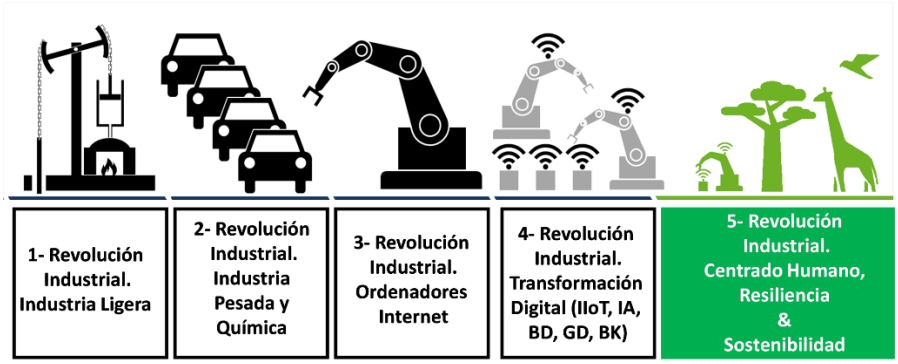

Figura 3.15 Esquema general de las etapas industriales incluyendo la Industria 5.0 [17].

3.2.1.1 Importancia de la Industria 4.0

Es importante entender el potencial de esta cuarta revolución industrial ya que no solo afectará a los procesos de fabricación. Su alcance es mucho más amplio, afectando a todos los sectores industriales e incluso a la sociedad.

Las tecnologías relacionadas con la Industria 4.0 también pueden conducir a productos y servicios completamente nuevos. El uso de sensores y dispositivos portátiles, el análisis y la robótica, entre otros, permitirán mejoras en los productos de diversas maneras, desde la creación de prototipos y pruebas hasta la incorporación de conectividad a productos previamente desconectados. Estos cambios en los productos se traducen, a su vez, en cambios en la cadena de suministro y, por consecuencia directa, en los clientes.

3.2.1.2 Tecnologías en detalle de la Industria 4.0

Los elementos y objetivos a tener en cuenta en la Industria 4.0, se ven reflejadas en los siguientes puntos:

1) Mundo IoT: Internet de las Cosas. Hay que tener en cuenta que el Internet de las Cosas se refiere a máquinas interconectadas vía/dentro de la red.
2) Ciberseguridad: Obviamente en cuanto se habla de máquinas funcionando por ellas mismas, han de protegerse de posibles accesos no deseados.
3) Sistemas basados en la nube: Almacenamiento de datos en cualquiera de los sistemas Cloud disponibles: Google Cloud, IBM, Amazon…
4) Big Data: Cuando se habla de almacenamiento en la nube, se está hablando del Big Data directamente el cual se refiere a grandes volúmenes de datos que han de ser accedidos y analizados.
5) M2M: Machine to machine, comunicación máquina a máquina. Cuando previamente se ha hablado del mundo IoT ya se ha introducido este término el cual se refiere a máquinas trabajando entre ellas por sí mismas.
6) Robótica avanzada.
7) Y por supuesto, muchos otros conceptos que seguramente resulten conocidos al lector como la impresión 3D, tecnologías móviles, simulación…

Estos pilares de la Industria 4.0 pueden verse reflejados en la Figura 3.16.

Figura 3.16 Pilares base de la Industria 4.0 [18].

3.2.2- Ecosistema PLCnext

La tecnología PLCnext desarrollada por Phoenix Contact, consiste en sistema integrado para la automatización industrial que consta de un software abierto, un hardware de ingeniería modular, una comunidad global y un acceso libre a aplicaciones basadas en esta tecnología que también se denomina en su conjunto Ecosistema PLCnext.

Como ya se había mencionado previamente en el apartado "**3.1.2 Aplicación de IEC 61131-3 sobre planta"**, además de la programación estándar de sistemas PLC según la norma IEC 61131-3, con PLCnext también es posible la combinación de esta normativa con el uso de lenguajes de programación como C/C++, C#, MATLAB, Simulink, Node-Red, Python… en sistemas tiempo real. Todas estas interacciones descritas pueden verse en la Figura 3.4.

Gracias a la sencilla integración de elementos en la nube, la posibilidad de utilizar el software Open Source (software abierto) y los conocimientos especializados tanto de usuarios como de empleados asociados directamente a Phoenix Contact, en constante crecimiento, se vuelve una herramienta esencial y extremadamente adecuada para la

implementación de los requisitos solicitados por la Industria 4.0. Pero esto se justificará con más detalle en el siguiente subapartado.

3.2.3- PLCnext en la Industria 4.0

Teniéndose en cuenta que los conceptos previamente descritos de: diferentes máquinas interactuando entre si dentro de la red, los datos constantemente siendo analizados, actualizados y accesibles en la nube o en la base de datos, las capas de seguridad que estos avances implican que sean implementadas… No es posible que sean cubiertos con las actuales tecnologías PLC sin que se empleen dispositivos externos que requieren más control y más esfuerzo para mantener todo organizado y flexible para adaptarse a cualquier futuro requisito de la Industria 4.0.

Es en este punto donde PLCnext entra al juego. Phoenix Contact proporciona al usuario una potente herramienta que permite cubrir todos los objetivos mencionados minimizando el número de dispositivos requeridos y preparando un entorno adecuado para ellos.

La tecnología PLCnext de Phoenix Contant involucra 5 aspectos esenciales que permite al usuario cubrir y estar preparado para todos los requisitos de la Industria 4.0.

1) **PLCnext Control:** Básicamente esto se refiere a la parte asociada al hardware. Todas las diferentes partes de hardware que involucra el uso del PLC: El controlador (AXC F 2152, AXC F 1152…) módulos digitales, módulos analógicos… (Esto no difiere mucho de un PLC típico con la excepción que presenta una mayor customización y una interacción más directa entre el operador y el PLC debido al factor de que se puede añadir, almacenar y modificar muchas de las propiedades del PLC, obviamente con ciertas limitaciones de seguridad).
2) **PLCnext Engineer:** Herramienta de desarrollo software de acceso completamente gratuito que permite al usuario programar y configurar el PLC para obtener la operación deseada.

3) **PLCnext Store:** Es el punto común entre usuarios y desarrolladores el cual busca expansionar las funcionalidades de PLCnext al máximo exponente.
4) **PLCnext Community:** Es donde usuarios, desarrolladores y staff oficial de Phoenix Contact están constantemente en contacto con la finalidad de ayudarse mutuamente para desarrollar el producto y proporcionar soluciones robustas y eficientes a todos los problemas o necesidades que puedan derivarse a partir del uso del equipo físico o software.
5) Proficloud.io: Sistema Cloud directamente desarrollado por Phoenix Contact que permite un fácil acceso a los usuarios de PLCnext a un entorno cloud adaptado y ajustado a los productos de Phoenix Contact.

Un pequeño esquema de todos estos apartados comentados puede verse en la Figura 3.17 y Figura 3.18 respectivamente.

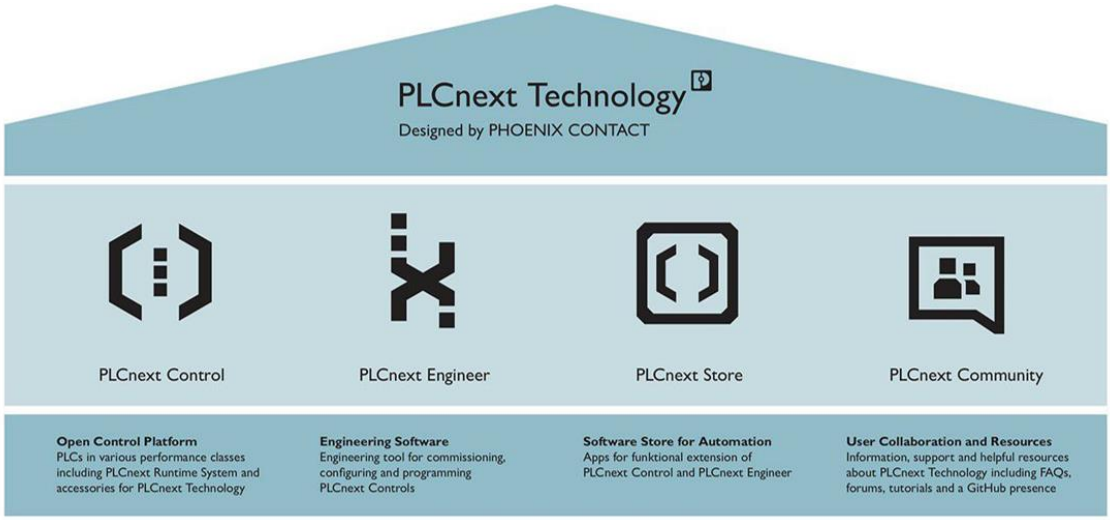

Figura 3.17 Marco de tecnologías comprendidas por la tecnología PLCnext [19].

Figura 3.18 Logo asociado a la plataforma Cloud Proficloud.io [20].

3.2.4- Hardware y software empleado

3.2.4.1 Componentes hardware

Para el desarrollo de este proyecto se ha usado tanto hardware como software específico de la compañía Phoenix Contact. Se ha de comenzar visualizando el hardware asociado a este sistema.

Ha sido empleado un módulo de entrenamiento EDU AXC F 2152 de Phoenix Contact (Figura 3.19) el cual está dotado de un controlador AXC F 2152, dos módulos de entradas y salidas digitales AXL F DI8/ DO8/1, un módulo de entradas y salidas analógicas AXL F AI2/ AO2 y un acoplador del bus AXL F BK PN.

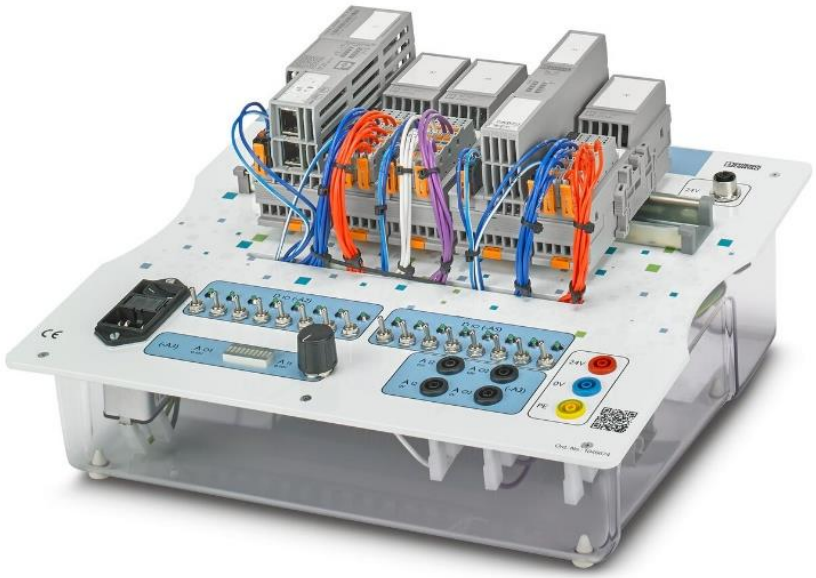

Figura 3.19 Módulo didáctico EDU AXC F 2152 [21].

3.2.4.2 Componentes software. Codesys

Si bien es cierto, Codesys es un software externo a las tecnologías que engloba PLCnext. Sin embargo, ha sido una herramienta fundamental en la concepción del proyecto y la base fundamental sobre la que se ha desarrollado el programa de PLCnext Engineer.

Describiendo ligeramente esta herramienta, Codesys es un término procedente de Sistema de Desarrollo de Controladores (CoDeSys: Controller Development Systems) y se corresponde con el software de automatización basado en el estándar IEC 61131-3 independiente de cualquier fabricante líder para sistemas de control.

Codesys básicamente se trata de un entorno de desarrollo basado en la programación industrial recogida dentro de la normativa ya previamente descrita en este documento, la llamada IEC 61131-3. [14]

La versión que concretamente se ha empleado para el desarrollo de la parte asociada a este software mencionado es la **V3.5 SP16 Patch 3**, esta versión puede ser vista en la Figura 3.20 Versión de Codesys v3.5 empleada en el desarrollo del proyecto..

Figura 3.20 Versión de Codesys v3.5 empleada en el desarrollo del proyecto.

3.2.4.3 Componentes software. PLCnext Engineer

Han de distinguirse dos partes esenciales dentro del software específico de desarrollo:

1) Software propio de Phoenix Contact, en especial PLCnext Engineer.
2) Software externo necesario, o más bien recomendable para el desarrollo, como por ejemplo WinSCP y UaExpert.

Las herramientas software externas se explicará su uso en detalle en apartados posteriores de este documento, concretamente dentro de los apartados **"3.3.2.1. UaExpert"** y **"3.4.2.1. WinSCP"**.

En este apartado, con el fin de mantener la integridad estructural y organización de este documento se centrará en PLCnext Engineer.

PLCnext Engineer es una plataforma de software específico para el desarrollo de programas de control automatizado de plantas desarrollado por Phoenix Contact para su uso directo en productos hardware de la citada compañía.

PLCnext Engineer cumple con el estándar de programación IEC 61131-3 y se dispone de un acceso totalmente gratuito para descarga en la página de Phoenix Contact (Figura 3.21)

Figura 3.21 Página de descarga de entorno de programación PLCnext Engineer [22]

Si bien es cierto que el programa de desarrollo base es totalmente libre, ha de tenerse en cuenta que hay ciertos módulos adicionales en la aplicación que requieren de un cierto pago [23]. Estos módulos son:

- **Sequential Function Chart Editor** (PLCnext ENG SFC): Licencia necesaria para la programación mediante normativa IEC 61131-3 en SFC.
- **Application Control Interface** (PLCnext ENG ACI): Interfaz para el control remoto de software PLCnext Engineer de aplicaciones de lenguaje de alto nivel externas.
- **Viewer for Simulink** (PLCnext ENG MV): Visualizador para la representación de modelos MATLAB Simulink en PLCnext Engineer.
- **Functional Safety Editor** (PLCnext ENG SAFETY): Editor (con certificación de TÜV-Rheinland) para programar aplicaciones de usuario relacionadas con la

seguridad y para la configuración y puesta en servicio de equipos PROFIsafe en sistemas de control orientados a la seguridad con tecnología PLCnext.

- **Complemento SAFE-CFUNC** (PLCnext ENG SAFE-CFUNC): Permite crear bibliotecas de funciones C y dotarse de certificados. Estas bibliotecas pueden enviarse a un sistema de control de seguridad sin necesidad de tener que actualizar el firmware del sistema de control.
- **Complemento generador HMI** (PLCnext ENG HMI): Generador HMI para generar una visualización completa basada en un proyecto de usuario sin esfuerzo manual.
- **ETH TOP VIEW** (PLCnext ENG ETH TOP VIEW): Ofrece la posibilidad de leer y representar la topología Ethernet conectada. En distintas vistas, pueden representarse los equipos con direcciones de red, puertos y tipos de conexión.
- **Complemento SIM** (PLCnext ENG SIM): La simulación permite probar aplicaciones para AXC F 2152 y 3152 sin un sistema de control real conectado. En caso de realizar simulación de AXC F 1152 esta licencia no sería necesaria.
- **Complemento VCS** (PLCnext ENG VCS): La ampliación de funciones permite la conexión de los sistemas de gestión de versiones Git y Subversion. De este modo, podrá sincronizar directamente desde PLCnext Engineer sus proyectos con el sistema de gestión de versiones.

En este caso solo ha sido necesaria para el desarrollo del proyecto, la primera de todas las mencionadas: Sequential Function Chart Editor (PLCnext ENG SFC) referenciada visualmente en la Figura 3.22.

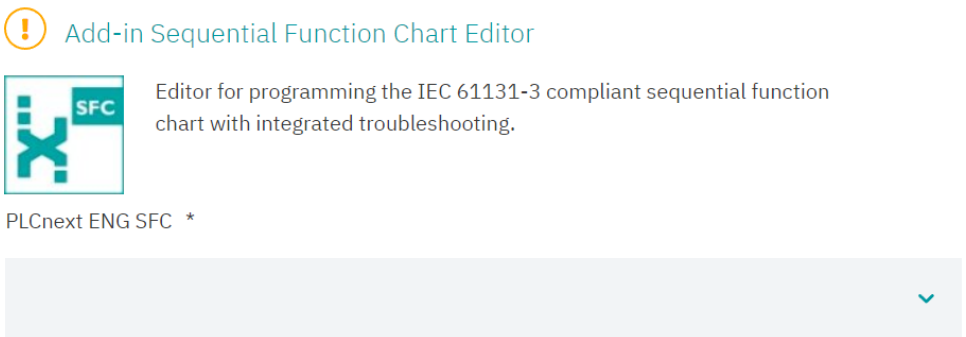

Figura 3.22 Módulo complementario SFC para PLCnext Engineer [23]

Para el desarrollo del programa se ha empleado PLCnext Engineer 2022.6 que se puede obtener en el apartado de Descargas -> Software, como se puede ver en la Figura 3.23 y Figura 3.24.

PLCnext_Engineer_Setup_2026.exe (616 MB) SHA256 Checksum: f8623fe0f5e3ad3f158f2309312e14e5586d0485fb59bbf17f45595e1ee619c0	PLCnext Engineer 2022.6: PLCnext Engineer is the modular software platform for PLCnext Control devices. It covers the technical disciplines needed to configure, develop, and commission an automation application.	Multilingual	2022.6

Figura 3.23 Descarga de instalador de PLCnext Engineer 2022.6 [22]

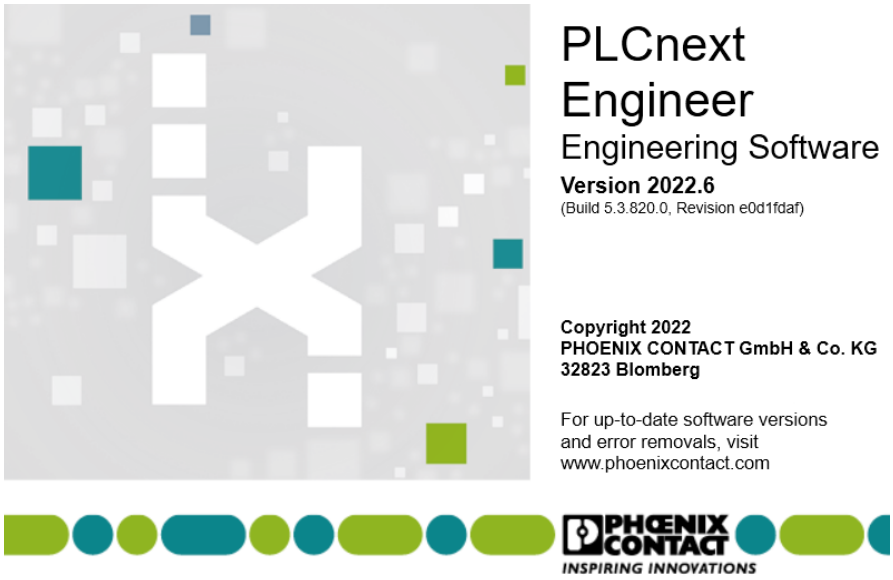

Figura 3.24 Versión de PLCnext Engineer 2022.6 empleada en el proyecto

3.3.- OPC UA

3.3.1- Introducción a OPC UA

OPC UA (del inglés, “OLE for Process Control Unified Access”) es la evolución del protocolo de comunicación OPC clásico que solo funcionaba con dispositivos Windows de Microsoft. OPC UA nace con la idea de aprovechar la tecnología moderna para dar lugar a la creación de una fábrica inteligente, por lo que busca que al usuario se le haga lo más fácil

posible la comunicación máquina a máquina (M2M que ya se había comentado en apartados anteriores de este documento).

El protocolo de comunicación OPC UA además de permitir la compatibilidad con distintos equipos, incorpora nuevos elementos de seguridad, recursos de control de acceso, autenticación y encriptación haciendo posible garantizar la transmisión segura de sus datos.

Como se venía diciendo, OPC UA es compatible con Windows, macOS, Android y Linux. Funciona en PC, infraestructuras basadas en la nube, PLC, microcontroladores y sistemas ciberfísicos (CPS).

En resumen, OPC UA proporciona una serie de ventajas que hace que resulte muy útil su utilización en la implementación de sistemas de automatización modernos. Lo primero es que en este caso en particular la disponibilidad de un servidor directamente implementado en el controlador lo hace muy conveniente para su aplicación debido a la directa incorporación de las variables en el servidor. A su vez, también es importante destacar que en la industria 4.0 destaca la relevancia de la seguridad al trabajar con los datos máquina a máquina. En este caso, el usuario también dispone de la posibilidad de asociar el certificado de seguridad que le convenga de manera directa. Y aunque en este caso no hayan sido explotadas, porque el servidor se ha empleado como mero puente de comunicación de datos, el protocolo OPC UA proporciona otras opciones interesantes como historizar variables o generar alarmas.

3.3.2- Implementación de servidor OPC UA en PLCnext

En la realización de este proyecto se ha empleado OPC UA para realizar la comunicación adecuada entre el controlador físico y Node-RED instalado en el mismo, permitiendo el intercambio de los datos que posteriormente se tratarán en la nube.

Para ello se ha recurrido al uso del servidor OPC UA que incorpora el controlador, debido a que permite una configuración directa en el propio entorno PLCnext Engineer haciendo muy sencillo el intercambio de datos desde el programa generado a Node-RED y

así se puedan tratar ahí para prepararlos como mensaje que el entorno Cloud seleccionado pueda interpretar.

Para comprobar si el servidor está funcionando correctamente, si bien es cierto que se ha implementado un programa específico para Node-RED que permite ver el servidor una vez configurado y las variables que lo componen, se ha empleado una herramienta para interactuar de manera más sencilla con el servidor y que recibe el nombre de UaExpert.

3.3.2.1 UaExpert

UaExpert es un cliente OPC UA el cual está diseñado para uso general que admite funciones como acceso a datos, alarmas y condiciones, acceso a históricos de variables y llamadas de métodos UA.

Para su descarga simplemente será necesario acceder a la página de Unified Automation [24] y proceder, requiriendo la creación de un usuario para ello. En este caso se ha empleado la versión 1.5.1 331 como se puede observar en la siguiente Figura 3.25.

Figura 3.25 Versión de UaExpert empleada por el proyectante en el trabajo.

El software UaExpert permitirá el acceso en modo cliente a un servidor OPC UA al que este tenga acceso y operar en el mismo.

El marco básico de UaExpert incluye funciones generales como el manejo de certificados, el descubrimiento de servidores OPC UA, la conexión con servidores OPC UA,

la exploración del modelo de información, la visualización de atributos y referencias de nodos OPC UA particulares.

Se centrará el foco en este último elemento mencionado y concretando para esta aplicación. El usuario se conectará como cliente al servidor OPC UA habilitado en el controlador físico por medio del entorno PLCnext Engineer y a través del cliente UaExpert, el usuario obtendrá las referencias de los nodos OPC UA asociados a las variables que se hayan habilitado para el servidor OPC UA en el programa de control. Todo esto se tratará con mayor detalle en los apartados asociados a la programación de PLCnext Engineer y Node-RED.

3.4.- NODE-RED

3.4.1- Introducción a Node-RED

Node-RED es una herramienta de desarrollo gráfico muy extendida y que está basada en el lenguaje de programación JavaScript y el cual guarda una gran relación con aplicaciones actuales con servicios asociados al IoT.

Node-RED proporciona un editor basado en navegador que facilita la conexión de flujos mediante la amplia gama de nodos que puede generar tanto el usuario, como incorporar nodos de procedencia externa de manera directa desde el mismo entorno. Node-RED es reconocible por medio del siguiente logo mostrado en la Figura 3.26.

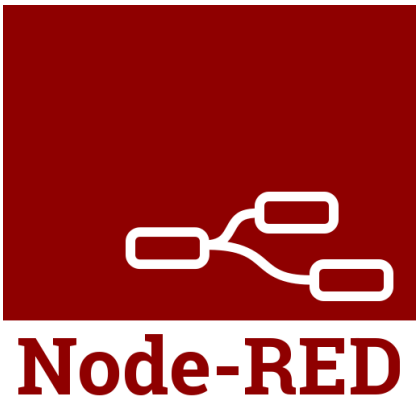

Figura 3.26 Logo de Node-RED [25]

Es un entorno de ejecución ligero que lo hace ideal para ejecutarse en sistemas computacionales de bajo coste como podría ser una Raspberry Pi o dispositivos que tengan compatibilidad con este tipo de entorno como es el controlador AXC F 2152 empleado para el desarrollo de este proyecto.

Los flujos creados en Node-Red por el usuario se almacenan mediante JSON, se puede importar y exportar fácilmente para compartir con otros usuarios. Un ejemplo de cómo quedaría parte del flujo desarrollado en el Node-RED asociado al controlador AXC F 2152 empleado se visualiza en la siguiente Figura 3.27.

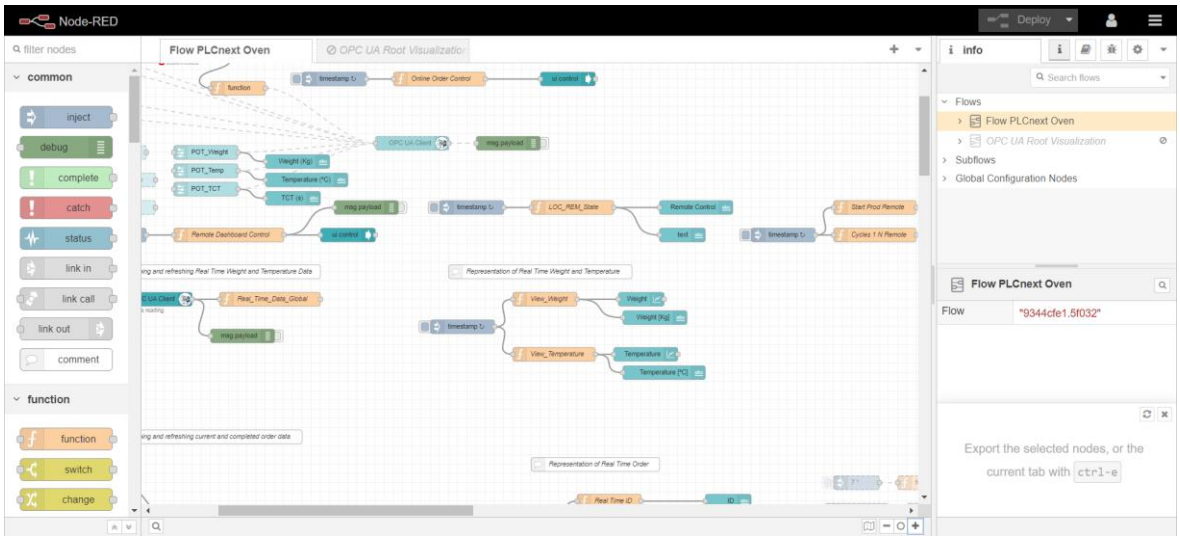

Figura 3.27 Flujo de código asociado al controlador AXC F 2152.

3.4.2- Instalación de Node-RED en PLCnext

Como se venía diciendo, el controlador seleccionado para el desarrollo de este proyecto permite al usuario la instalación del entorno Node-RED en el mismo. Para realizar la instalación de Node-RED en PLCnext se disponen de dos opciones:

a) Instalarlo manualmente mediante el acceso de la carpeta raíz del PLCnext.
b) Instalarlo mediante comandos, proporcionando acceso al controlador PLCnext a una red de internet.

Se ha optado por la segunda opción debido a la sencillez y conveniencia de este método y puesto que al controlador se le ha de proporcionar si o si de acceso a una red para completar el posterior envío de datos desde el controlador a la nube.

Antes de proceder, se debe de disponer de dos herramientas que harán falta para la realización de esta instalación. Estas dos aplicaciones se llaman WinSCP y Putty respectivamente y se analizará en detalle en los siguientes apartados en qué consisten y con qué finalidad han sido empleadas.

3.4.2.1 WinSCP

La primera herramienta que ha de tenerse en cuenta es WinSCP, que permitirá al usuario acceder al almacenamiento del controlador PLCnext.

WinSCP es cliente SFTP gráfico para Windows que emplea SSH. Su función principal es facilitar la transferencia segura de archivos entre dos sistemas, en este caso, el PC del usuario en el que se encuentre instalado WinSCP y el controlador PLCnext respectivamente. Un ejemplo de cómo se podría visualizar una conexión entre PC y PLC se puede observar en la siguiente Figura 3.28.

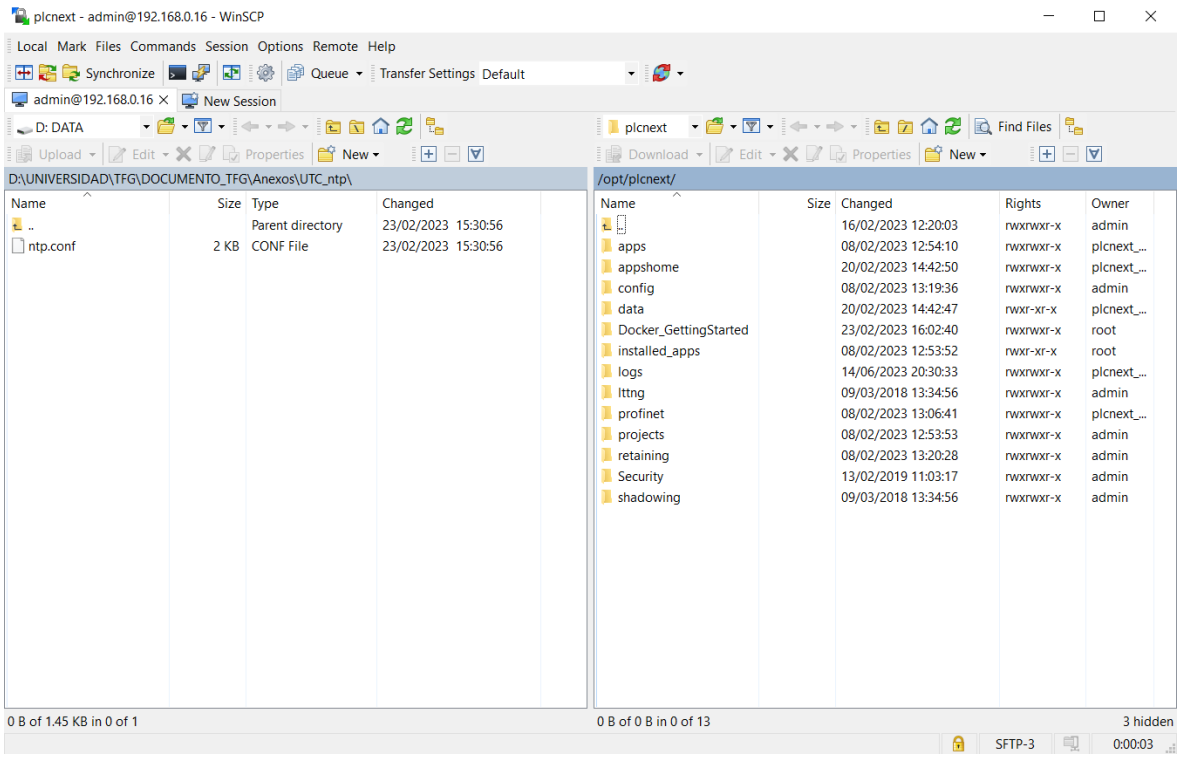

Figura 3.28 Conexión realizada entre PC (izquierda) y PLC (derecha) en WinSCP.

Dicho software podrá se puede descargar por el usuario en la página de WinSCP [26]

La versión del software WinSCP instalado y empleado en el equipo local ha sido la que se puede visualizar en la siguiente Figura 3.29.

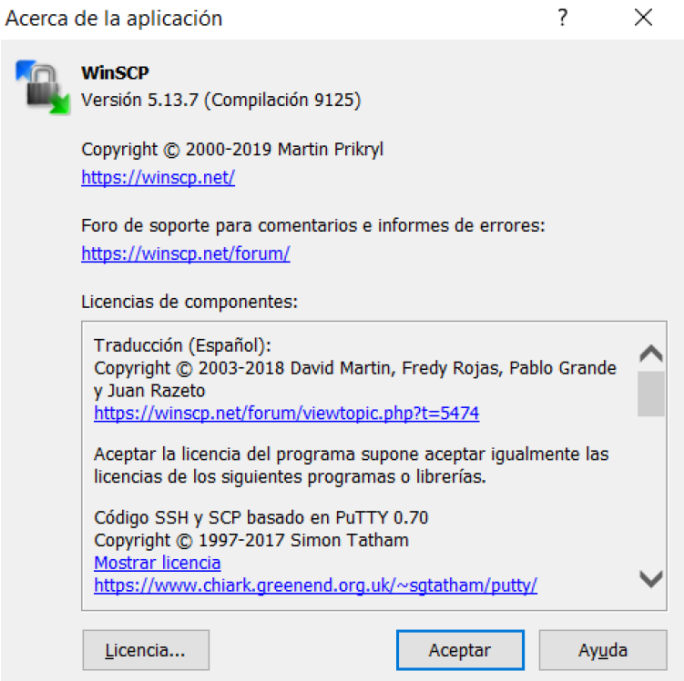

Figura 3.29 Versión 5.13.7 de la herramienta WinSCP instalada por el proyectante.

3.4.2.2 Putty

La segunda herramienta necesaria y que será empleada en conjunción con el ya previamente mencionado WinSCP, es la llamada Putty, la cual permitirá al usuario disponer de una ventana de comandos asociada al controlador PLCnext para crear un usuario raíz en el controlador PLCnext empleado e introducir los comandos necesarios para desarrollar la instalación del entorno Node-RED en el controlador.

Dicho software podrá ser descargado por el usuario en la página de descarga de Putty [27].

La herramienta software Putty instalada y empleado ha sido la versión 0.76 tal y como se puede visualizar en la siguiente Figura 3.30.

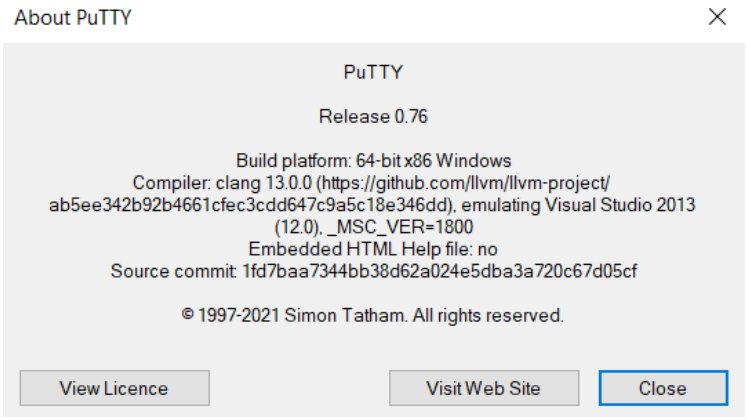

Figura 3.30 Versión de Putty instalada.

3.4.2.3 Procedimiento de instalación de Node-RED

Una vez instaladas las herramientas de software previamente mencionadas, se ha de proceder de la siguiente manera para completar la instalación de Node-Red:

1) Se abre WinSCP. Ha de introducirse la dirección IP asociada al PLCnext (192.168.0.16) y las credenciales (admin y contraseña) que siempre emplean para conectarse al mismo como se muestra en la siguiente Figura 3.31.

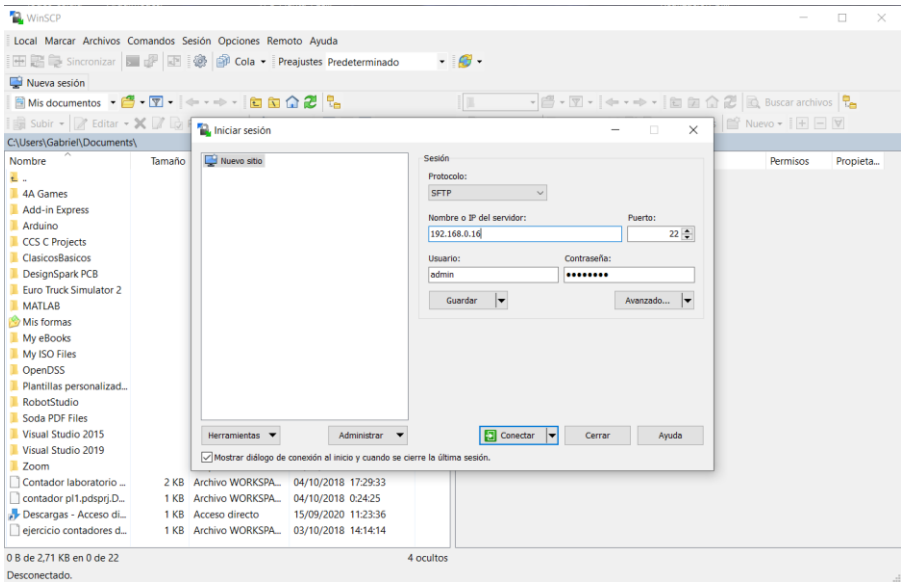

Figura 3.31 Acceso al sistema de PLCnext por medio de protocolo SFTP.

NOTA: Tras pulsar *Conectar*, saldrán dos avisos de verificación de seguridad que se pueden asumir.

2) Como se podrá observar, una vez que se establece la conexión con el controlador PLCnext en WinSCP, el usuario dispondrá a la derecha del directorio /opt/plcnext/ asociado al propio controlador PLCnext empleado como se puede observar en la siguiente Figura 3.32.

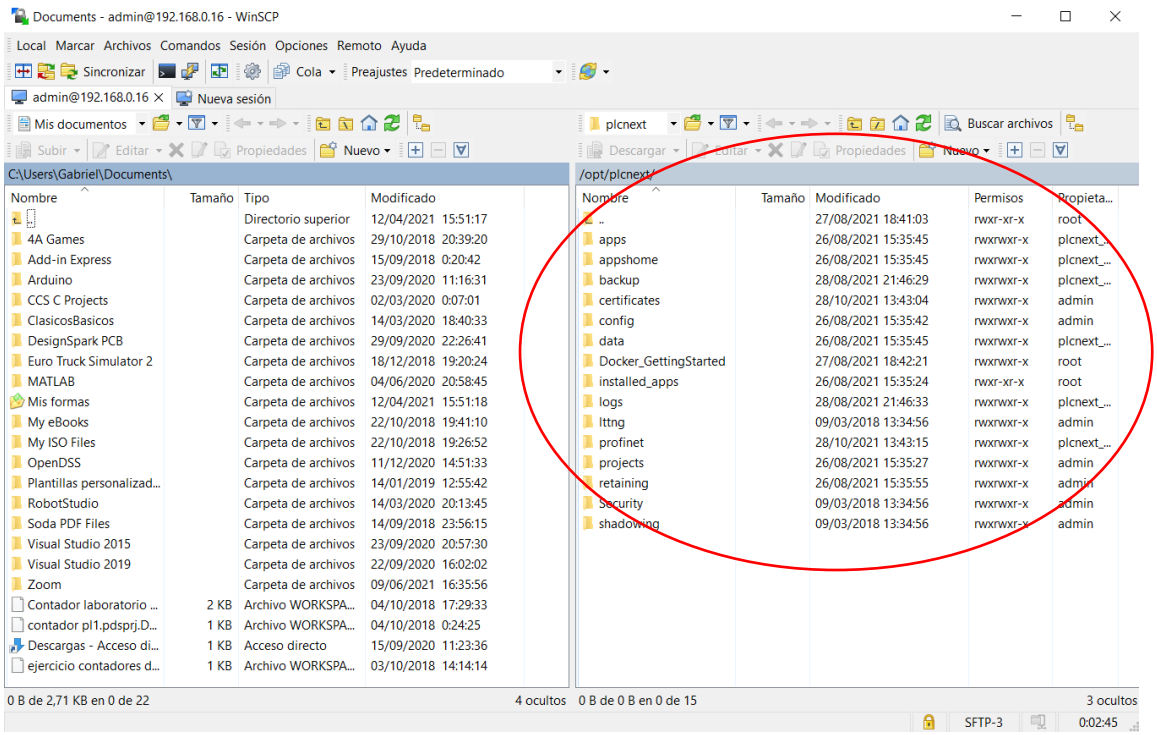

Figura 3.32 Sesión de sistema abierta contra PLCnext.

3) Una vez el usuario ya haya ingresado en el directorio asociado al controlador PLCnext, lo siguiente a realizar será ejecutar Putty, con la finalidad de introducir los comandos necesarios de instalación de Node-RED. Se puede activar esta ejecución con el icono indicado ( ), se puede ver su localización en la siguiente Figura 3.33.

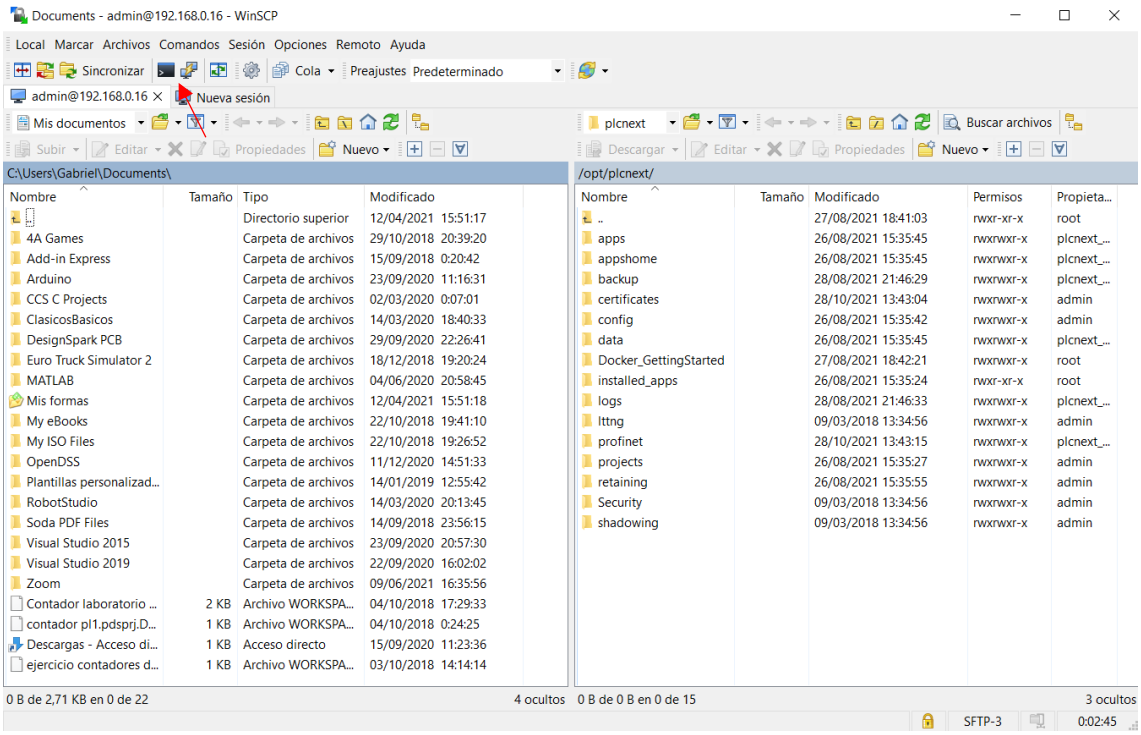

Figura 3.33 Acceso al controlador realizado. Apertura de terminal en Putty.

Se abrirá entonces una sesión en Putty, apareciendo la siguiente ventana visualizada en la Figura 3.34 que pedirá identificarse con el usuario y la contraseña asociadas al controlador PLCnext.

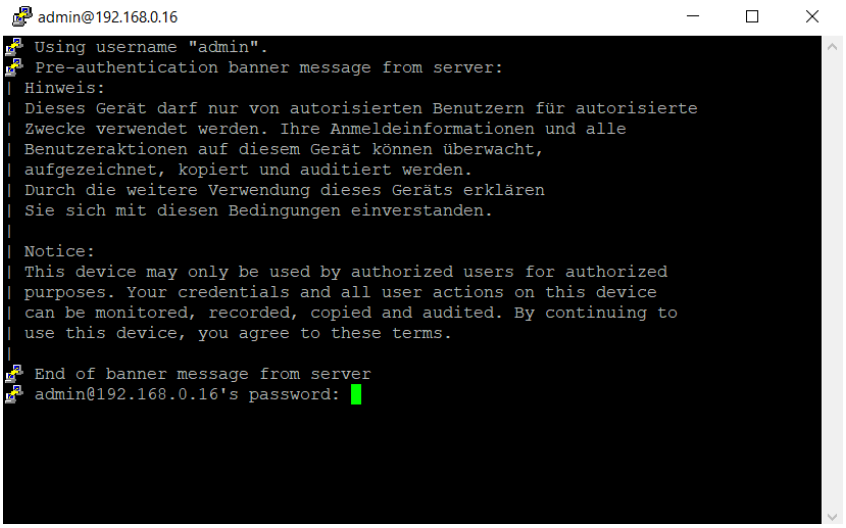

Figura 3.34 Apertura de terminal. Petición de identificación con credenciales del controlador.

4) Tras realizar dicha identificación, la ventana aparecerá de la manera que se muestra en la siguiente Figura 3.35.

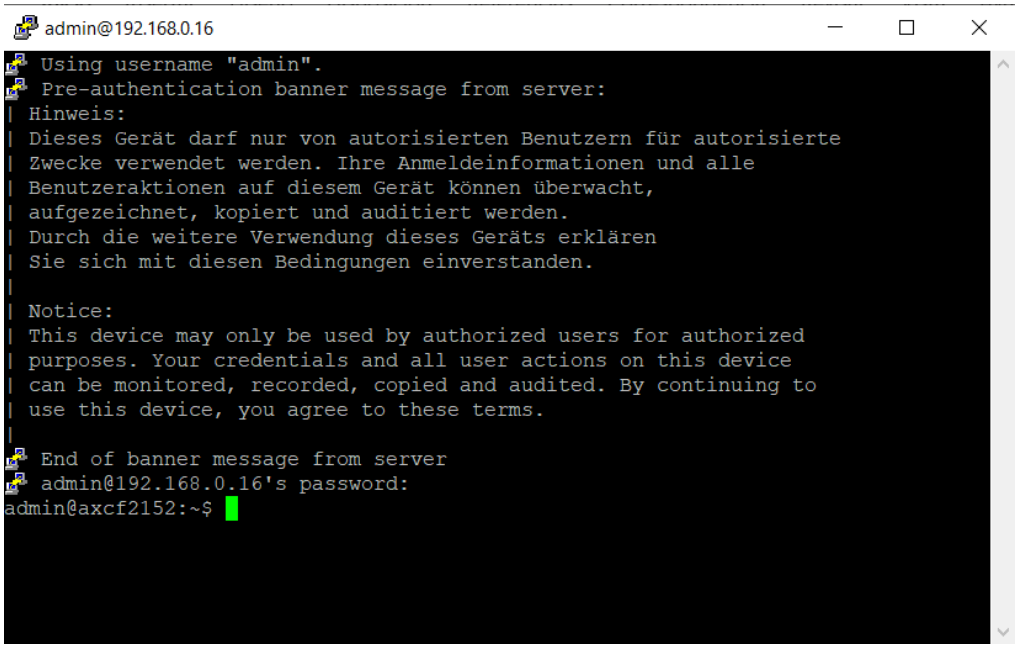

Figura 3.35 Identificación realizada por parte del usuario.

Antes de proceder con los comandos de instalación hay que realizar unos pasos previos. Lo primero es crear un usuario en la raíz ("root"), para ello se teclea el siguiente comando:

sudo passwd root

Pedirá una contraseña a elegir. En este caso se ha decidido establecer la contraseña asociada al controlador PLCnext seguida de 2023. Quedando esta: **7ab546092023**

NOTA: Puede que no se vea el cursor y los caracteres tecleados.

Una vez ya ha sido creado el usuario de acceso a la raíz, el usuario ha de identificarse de la siguiente manera:

su

7ab546092023

Una vez realizada la identificación del usuario raíz creado, la ventana aparecerá de la manera mostrada en la Figura 3.36, viéndose como ya se representa el directorio previamente mencionado /opt/plcnext/ visualizado previamente nada más conectarse con el controlador en WinSCP como se podía observar en Figura 3.32 indicando que ya se dispone de acceso al directorio raíz del PLC ("root") y es posible proseguir con la instalación.

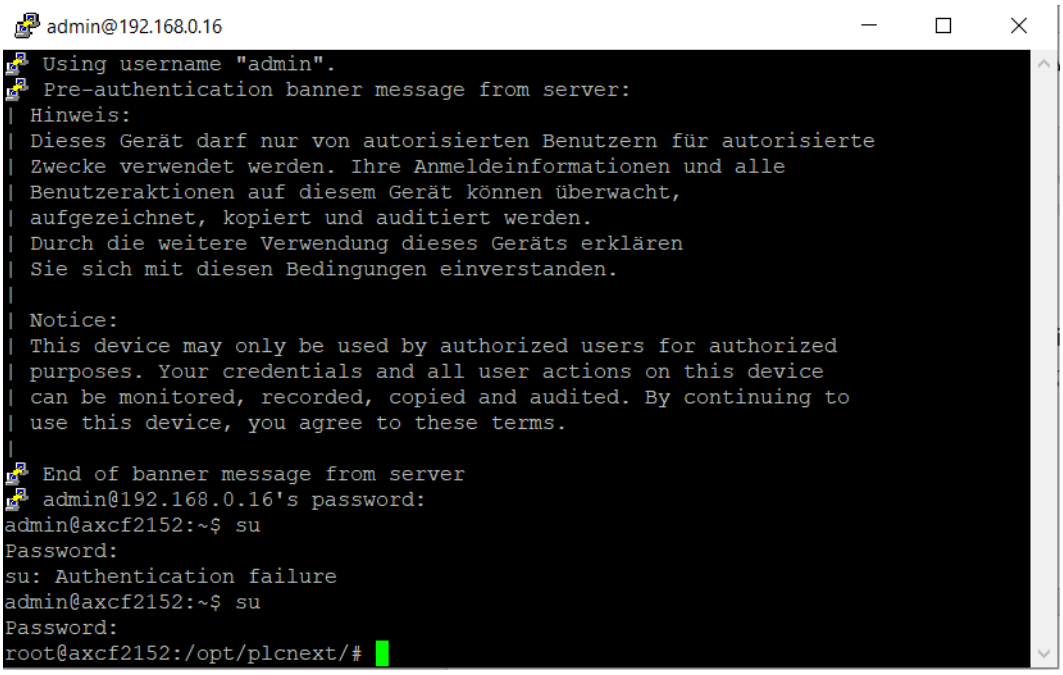

Figura 3.36 Identificación realizada por parte del usuario raíz.

Antes de comenzar con los comandos de instalación de Node-Red, en aras de asegurar el acceso a internet del controlador PLCnext para poder aplicar el método b) de instalación de Node-Red previamente mencionada, se han de introducir los siguientes comandos:

nano /etc/network/interfaces

Hay que verificar que salga la ventana de la manera visible en la Figura 3.37, verificando que la dirección del Gateway sea 192.168.0.99 para que se corresponda con la del Router que da acceso a internet dentro de la red seleccionada y que la dirección asociada al controlador PLCnext se encuentre dentro de la máscara de red correspondiente (255.255.255.0) en este caso se ha seleccionado 192.168.0.16

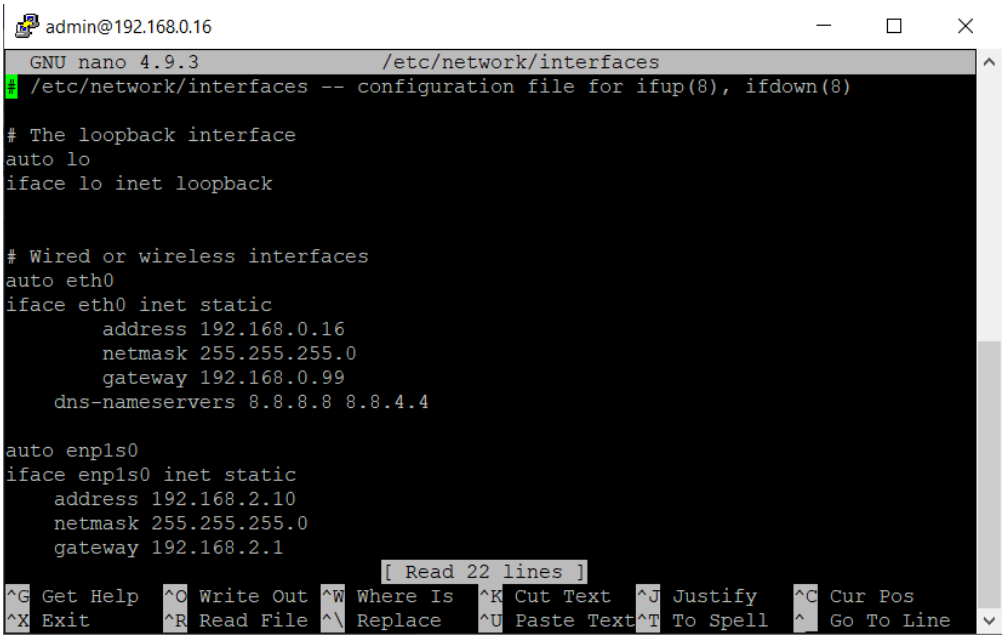

Figura 3.37 Configuración de red del controlador PLCnext.

Si alguno de estos valores no se corresponde, se podrá cambiar manualmente y guardar en el directorio /etc/network/interfaces.

Una vez que el usuario ya esté conforme con los datos establecidos ha de pulsarse se siguiente juego de teclas con la finalidad de volver a la carpeta raíz:

Ctrl + **x**

Si se ha realizado de manera correcta, se debería visualizar la ventana de sesión Putty de la manera visualizada en la Figura 3.38.

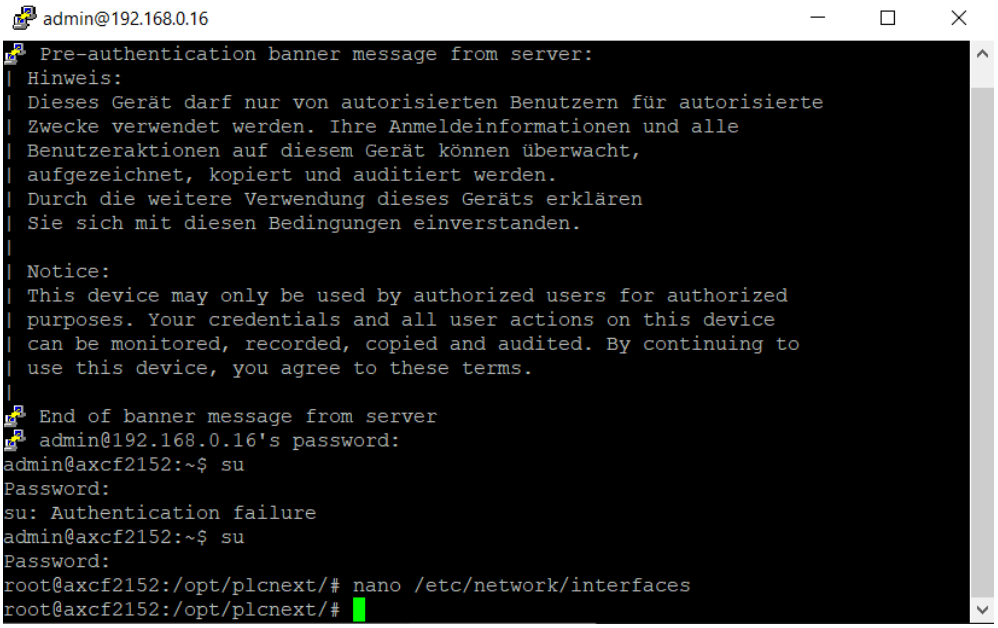

Figura 3.38 Directorio principal con identificación de usuario raíz realizada.

Lo siguiente será verificar que el controlador PLCnext empleado tiene acceso a internet mediante los siguientes comandos y juegos de teclas:

ping 8.8.8.8

Ctrl + **c**

ping google.com

Ctrl + **c**

El juego de teclas **Ctrl** + **c** se hace con la finalidad de parar el comando ping que en otro caso al igual que sucede con Windows, se quedaría a la espera ejecutándose en bucle la ejecución del comando ping.

NOTA: Para poder realizar todos estos pasos previos de configuración de IP del controlador PLCnext será esencial tener el controlador PLCnext mencionado correctamente actualizado para que no pueda presentar ningún problema tras realizar estos cambios necesarios.

Remarcar, que este paso de configuración de IP asociadas al controlador, desde la versión de controlador 2022.0, se puede realizar directamente desde el webserver asociado al mismo. Para ello, se procede de la siguiente manera:

1) Accedemos al webserver del dispositivo simplemente con poner en el navegador de preferencia la IP asociada al controlador seguido de /wbm, quedando en este caso: 192.168.0.16/wbm como se puede observar en la siguiente Figura 3.39.

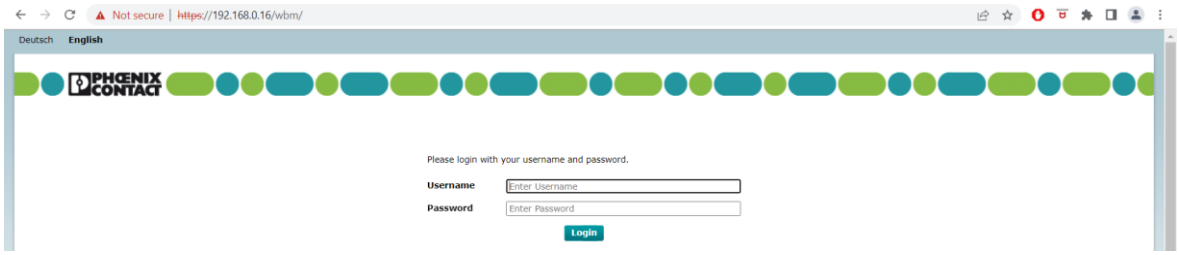

Figura 3.39 Acceso al webserver del controlador PLCnext.

2) Pedirá identificarse con las credenciales asociadas al controlador de admin y contraseña, dando acceso a diferentes opciones a realizar en el controlador y mostrando diferentes datos del mismo como la versión del mismo. Se puede ver en la siguiente Figura 3.40.

Figura 3.40 Página principal una vez garantizado el acceso al webserver del controlador PLCnext.

3) Para acceder a la configuración de IPs tal y como se ha realizado en el Putty bastará con ir a la pestaña: Configuration/Network visualizando la siguiente pantalla mostrada en la Figura 3.41.

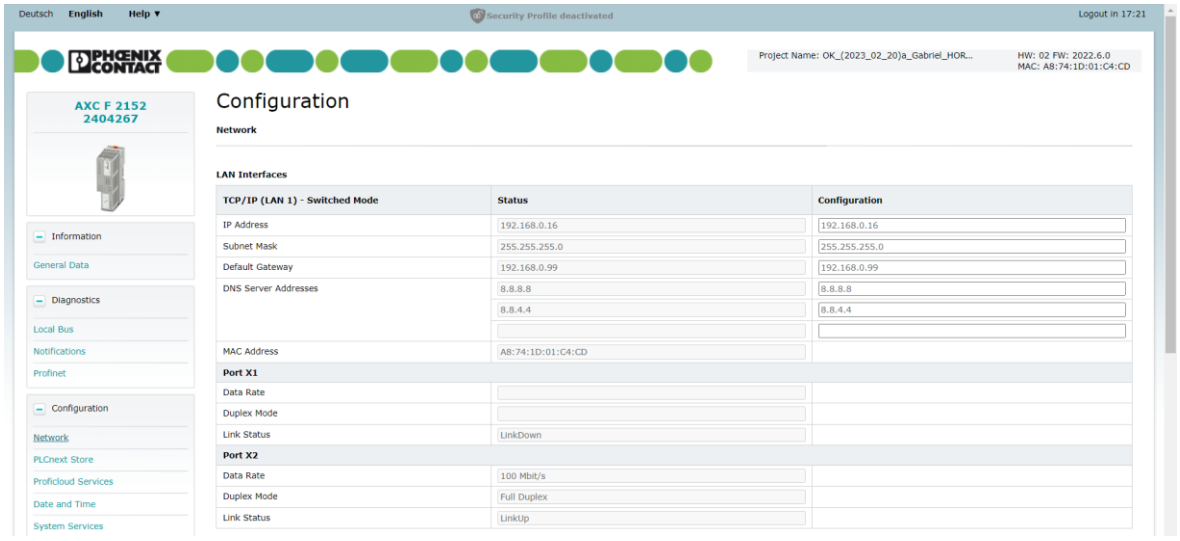

Figura 3.41 Acceso a la pestaña de configuración de red dentro del webserver del controlador PLCnext.

Verificada esta conexión a internet, se procede a introducir los siguientes comandos necesarios para llevar a cabo la instalación de Docker [28], con la finalidad de luego generar un contenedor que vaya a albergar el Node-Red dentro del controlador PLCnext empleado.

Lo primero que se realiza es copiar el repositorio de Docker creado en github y acceder a la nueva carpeta creada, la cual puede verse en la Figura 3.42.

git clone https://github.com/PLCnext/Docker_GettingStarted.git

cd Docker_GettingStarted

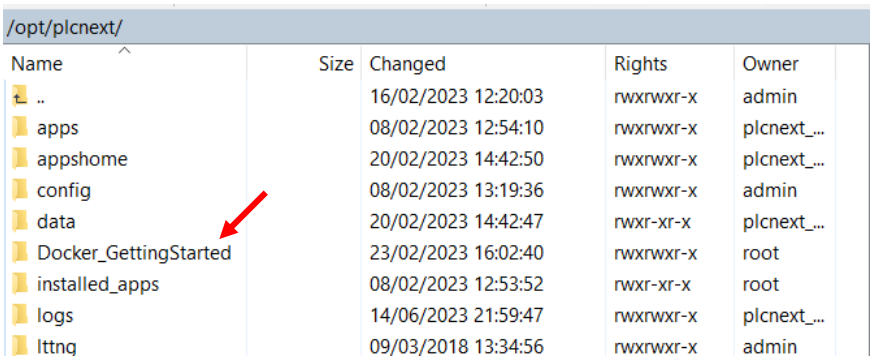

Figura 3.42 Directorio Docker_GettingStarted clonado al directorio /opt/plcnext/ del controlador PLCnext.

A continuación, lo que se hace es convertir en ejecutable el script de setup y posteriormente, ejecutarlo.

chmod -c 777 setup.sh

./setup.sh

Se solicitará el permiso para instalar Balena-engine o Docker. Se ha seleccionado la instalación de Balena-Engine ya que es la recomendada por el propio equipo.

Una vez completada la instalación es posible verificar la versión del motor instalado y que está correctamente instalado en el sistema introduciendo el siguiente comando.

balena-engine version

Debería aparecer la ventana de sesión en el Putty de una manera semejante a la mostrada en la Figura 3.43.

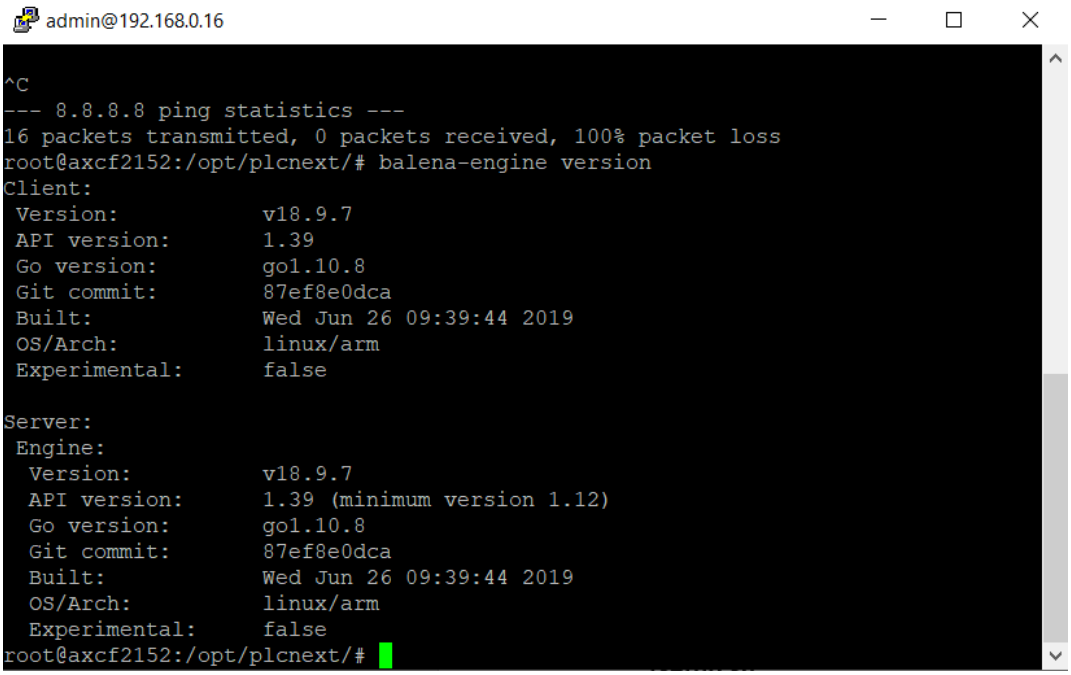

Figura 3.43 Comprobación de versión de Balena-engine.

Ahora se podrá introducir el comando específico de creación del contenedor que contenga Node-RED. Es muy importante remarcar que hay que crear un volumen de datos específico, porque esto permitirá incluirle al Node-RED del controlador una seguridad asociada por medio de una identificación de usuario y contraseña.

balena-engine run -d -p 1880:1880 --name nodered --restart always -v node_red_data_plcnextd nodered/node-red

En este momento se está generando un contenedor con Node-RED en el puerto 1880 con el nombre *nodered*, el cual hará *restart always* cada vez que arranquemos el controlador. A su vez, se ha creado un volumen de datos con el nombre *node_red_data_plcnextd* en el directorio */media/rfs/rw/var/lib/balena/volumes/node_red_data_plcnext/_data* como se puede contemplar en la Figura 3.44.

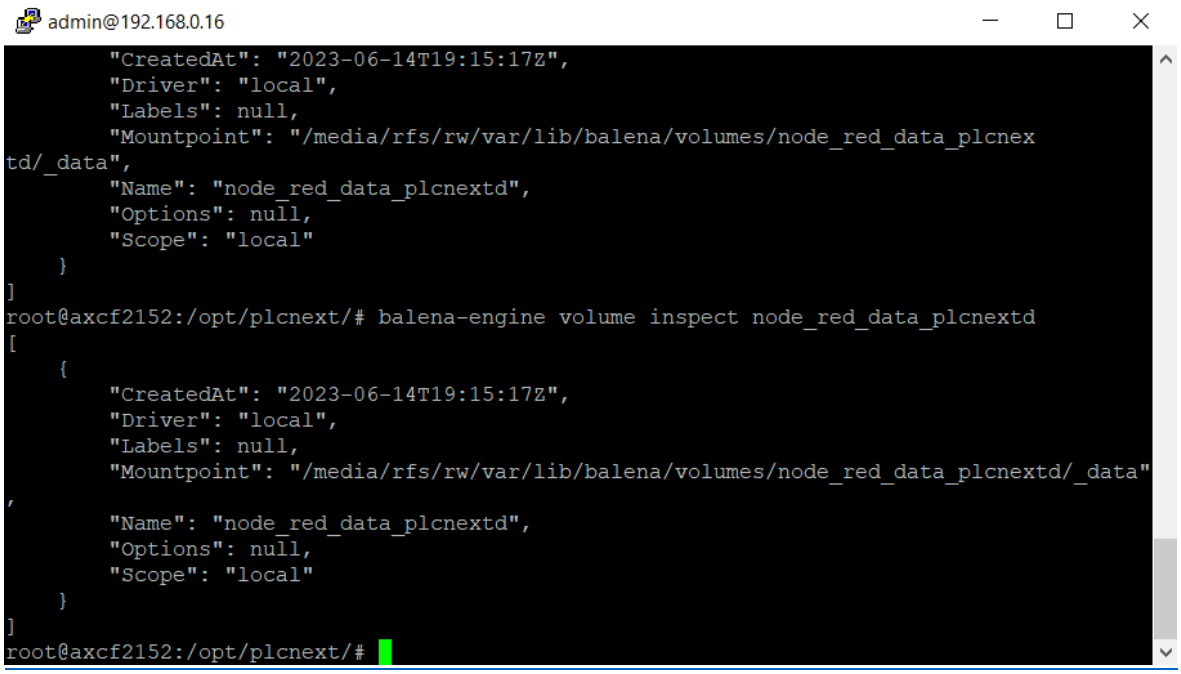

Figura 3.44 Verificación del volumen de datos creado para el contenedor de Node-RED.

El siguiente paso antes de proceder con cualquier elemento de programación asociado a Node-RED y con el fin de dejarlo completamente adaptado al usuario, sería la adición de seguridad por medio de identificación por usuario y contraseña. Para ello hemos de acceder al archivo *settings.js* localizado en el directorio de volumen de datos previamente citado.

Si el usuario intentase acceder al directorio por medio de WinSCP para poder abrir el archivo *settings.js* directamente con el Bloc de notas, se toparía con el siguiente mensaje mostrado en la Figura 3.45, impidiéndole avanzar hacia este.

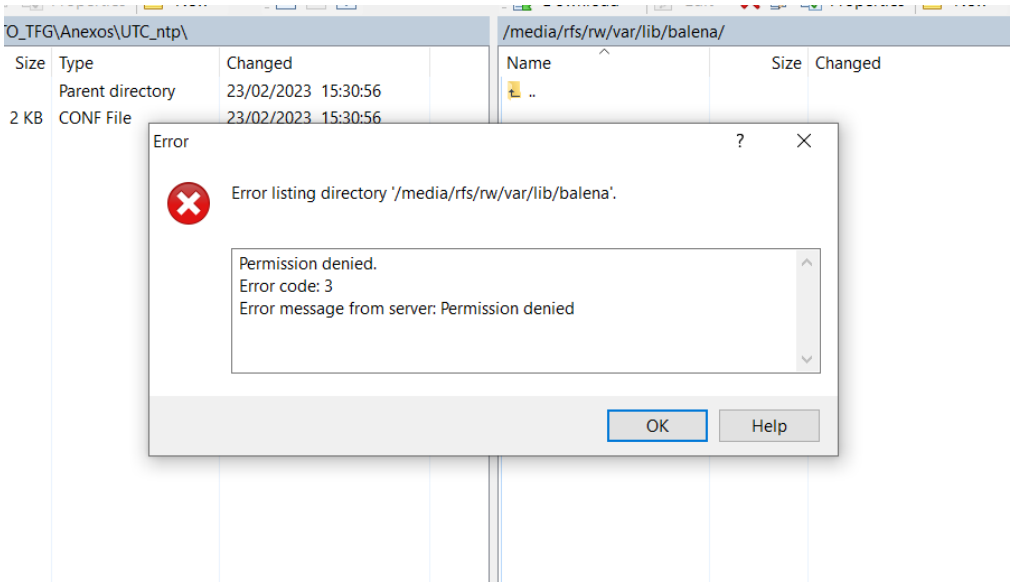

Figura 3.45 Error si se intenta acceder directamente por medio de WinSCP al archivo *settings.js*.

No quedando más remedio que acceder al directorio por medio de la herramienta Putty empleada en la instalación de Node-RED. Se ha de avanzar hasta el directorio donde se encuentra el volumen de datos ya mencionado y se vea el archivo *settings.js* como se observa en la Figura 3.46.

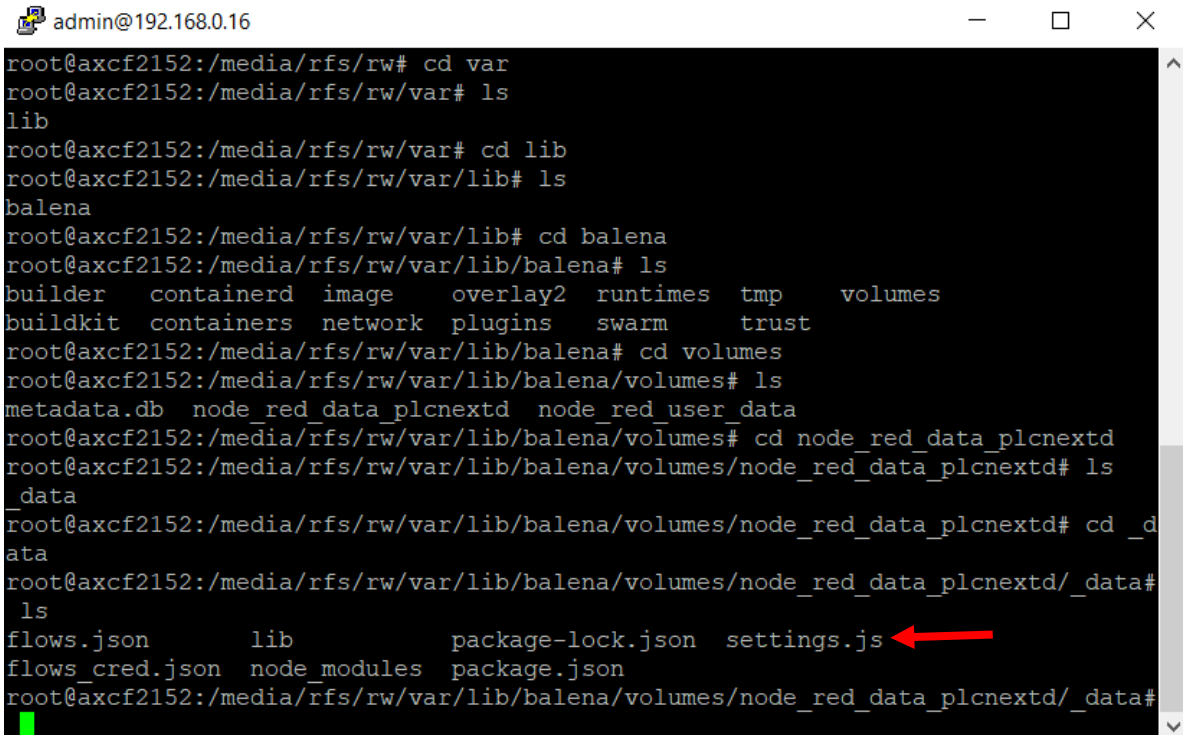

Figura 3.46 Localización del archivo settings.js por medio de la terminal del controlador PLCnext.

Si se realizase su apertura por medio del comando indicado a continuación y se accediese a la parte del código en la que se encuentran los identificadores de usuario y contraseña, se vería lo mostrado en la Figura 3.47.

nano settings.js

```
adminAuth: {
    type: "credentials",
    users: [{
        username: "admin",
        password: "$2b$08$wpyojqpOWd2sY9qckyyNxuGMhxAbFn0eDezxZztDSAXQkEZm0
        permissions: "*"
    }]
},
```

Figura 3.47 Credenciales de seguridaad asociadas al programa de Node-RED.

El usuario no muestra mayor problema, sin embargo, la contraseña se puede observar que tiene una codificación propia. Para hacer esa codificación específica asociada a la contraseña de Node-RED bastaría con aplicar el siguiente comando:

node-red admin hash-pw

Pero este comando no es válido en la línea de comandos de Putty. Para solventar este pequeño inconveniente, se ha recurrido a un Node-RED instalado a nivel local en ordenador del proyectante por medio de Docker Desktop. El procedimiento de instalación de Node-RED por medio de Docker Desktop es similar al ya descrito en PLCnext, así que se omite la explicación.

La versión de Docker Desktop que se ha empleado para el desarrollo de esta tarea ha sido la v4.20.1 como se puede observar en la siguiente Figura 3.48.

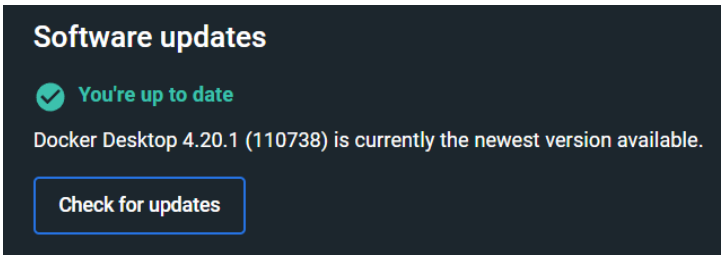

Figura 3.48 Versión de Docker Desktop empleada en el proyecto.

Creado el contenedor con Node-RED al igual que se ha hecho para el controlador PLCnext, el usuario podrá acceder también al archivo *settings.js* asociado al Node-RED del Docker Desktop, accediendo mediante la terminal al directorio /data como se puede visualizar en la Figura 3.49.

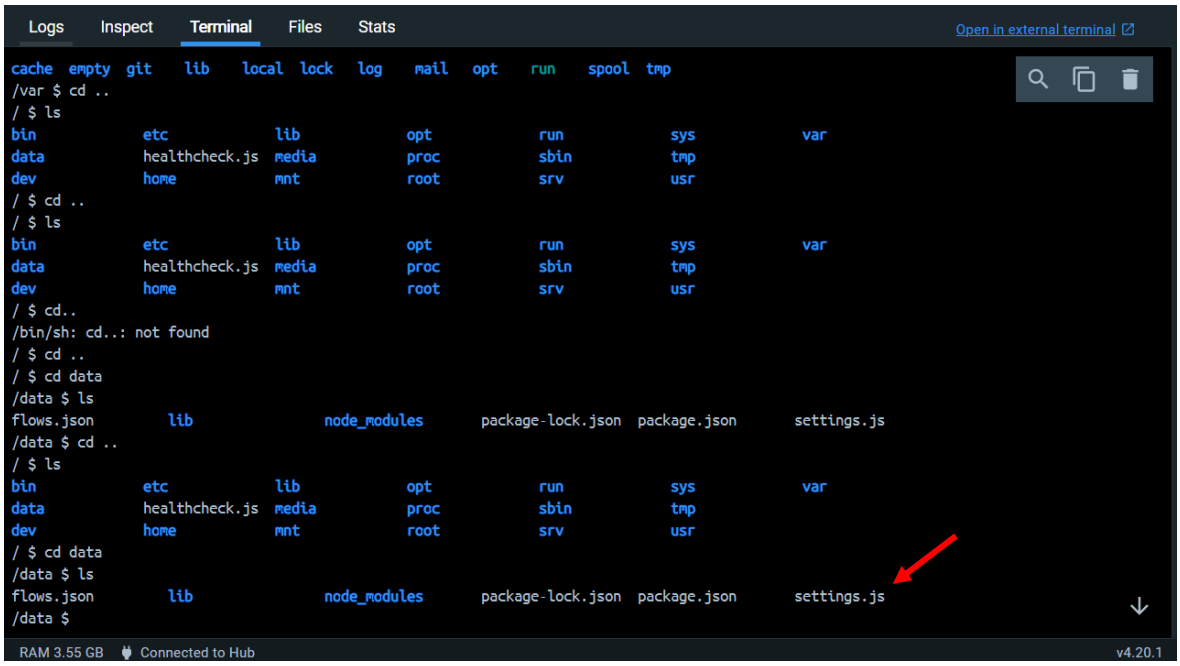

Figura 3.49 Acceso al archivo settings.js del contenedor Node-RED creado en Docker Desktop.

Se podría abrir al igual que el *settings.js* del Node-Red del controlador, pero en este caso el interés no es este archivo, si no que aquí sí que se dispone de la posibilidad de ejecutar el comando previamente citado de **node-red admin hash-pw** y a continuación pedirá que se introduzca la contraseña deseada por el usuario como se indica en la Figura 3.50.

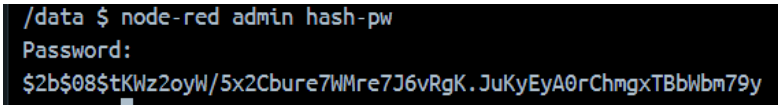

Figura 3.50 Contraseña generada por comando hash-pw.

Una vez obtenida la codificación asociada a la contraseña generada por el usuario, se accedería al *settings.js* del Node-RED asociado al controlador PLCnext como se había realizado previamente y se representará como se había visualizado en la Figura 3.47.

Una vez se haya completado el proceso de asociación de identificación a Node-RED por medio de usuario y contraseña, el acceso al Node-RED del controlador, debería verse como se aprecia en la Figura 3.51.

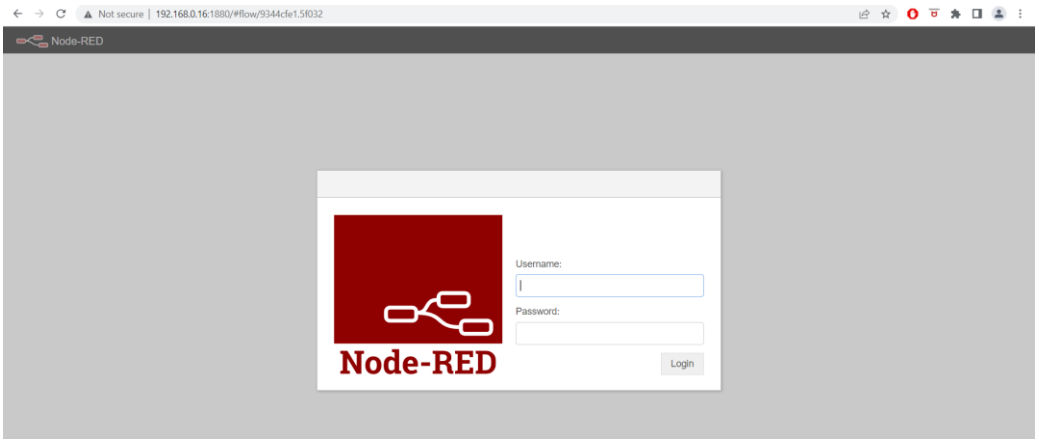

Figura 3.51 Identificación pedida por Node-RED tras concretar las credenciales en el archivo *settings.js*.

3.4.2.4 Cloud

El Node-RED también ha sido empleado en los entornos Cloud que se han probado. En este caso se ha realizado el desarrollo sobre 3 entornos Cloud. El primer entorno Cloud es el propio de PLCnext, ya citado, que se conoce como Proficloud.io, sin embargo, ahí no se ha realizado ningún tipo de desarrollo con Node-RED. Ha sido en los entornos Cloud de IBM y Azure donde se ha recurrido a su uso.

Inicialmente se había hecho el desarrollo con IBM, para emplear IBM Watson como servicio IoT, sin embargo, debido a cese de servicio por parte de IBM, se ha tenido que hacer un traslado a la nube de Azure. En ambas se ha empleado Node-RED. Puesto que la instalación de Node-RED en las plataformas Cloud va más ligada individualmente a cada una de las nubes, la preparación de Node-RED para su uso en estos entornos se tratará más adelante en este documento.

3.4.3- Implementación en Node-RED de PLCnext

La aplicación en este proyecto del entorno Node-RED viene fundamentada como ya se venía citando como un medio que permita realizar un paso de las variables localizadas en el programa de control asociado al controlador PLCnext a el entorno Cloud seleccionado.

Por parte del controlador, las variables de interés a gestionar en el entorno Cloud se encuentran localizadas en el servidor OPC UA ya referenciado y disponible en el propio controlador. En el otro lado se dispone de un entorno Cloud que proporciona una plataforma de comunicación segura con dispositivos para traspaso de datos y generalmente referenciada en la plataforma con IoT o terminología semejante. En este caso, aunque se expondrá más en detalle en el siguiente apartado, el entorno Cloud seleccionado ha sido Microsoft Azure y la plataforma de comunicación que ofrece se conoce con el nombre de IoT Hub.

Entre medias se encuentra por tanto Node-RED que establece de base de unión entre ambos *"hemisferios"*, el controlador y la nube. Se puede apreciar de una manera más visual en el esquema mostrado en la Figura 3.52, que establece una referencia resumida de lo que se acaba de narrar.

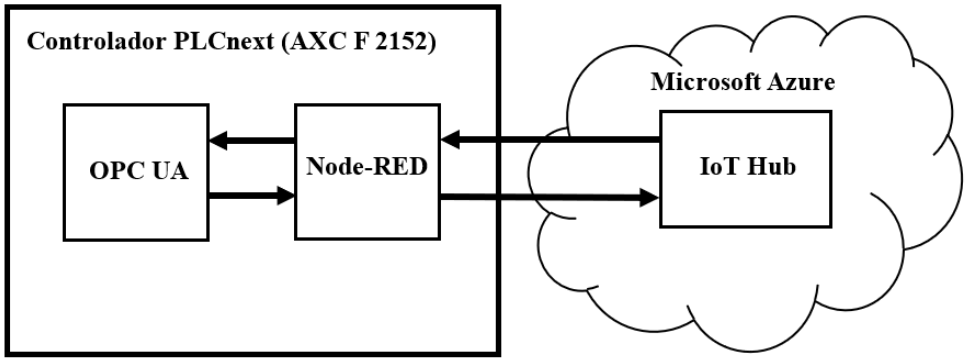

Figura 3.52 Esquema simplificado de interacción de datos entre controlador y plataforma Cloud.

En el entorno Cloud, también habría Node-RED pero el traspaso de datos es similar y se referenciará con más detalle en el siguiente apartado de este documento.

3.5.- SERVICIOS DE LA NUBE

3.5.1- Introducción

Uno de los puntos también importantes a destacar e implementar dentro de cualquier sistema de automatización moderno, es el que se incluyan funcionalidades Cloud complementarias al funcionamiento típico y normal de la planta base.

A día de hoy, la inclusión de todas las funcionalidades y objetivos introducidos por la evolución hacia la Industria 4.0 hacen esencial la inclusión de estos servicios Cloud para análisis y almacenamiento de datos a gran volumen y a su vez generar un acceso a un entorno más global a la evolución y trabajo de la planta sin necesidad de estar ligado al entorno local de la misma como tradicionalmente se venía haciendo.

3.5.2- Implementación de servicios en la nube

Una vez realizada una pequeña introducción de la importancia de los servicios Cloud en cualquier sistema automatizado moderno, ha de visualizarse ligeramente como ha sido el seguimiento de la estructura general de proyecto en la parte asociada a la parte Cloud.

El esquema general implementado visto con más detalle respecto el observado al final del apartado anterior con la inclusión de la parte Cloud al completo y destacando que Node-RED del controlador se encuentra comprendido dentro de un contenedor, se puede visualizar en la siguiente Figura 3.53 que muestra gráficamente la interacción general entre PLCnext y el entorno Cloud.

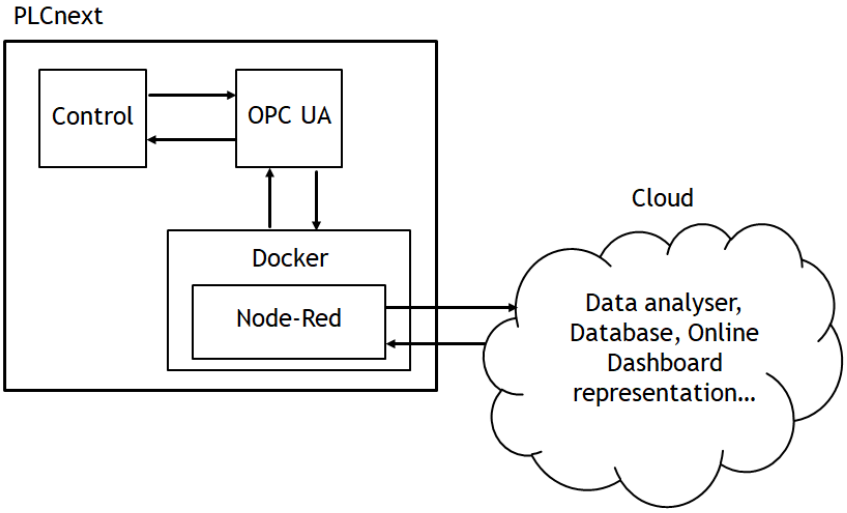

Figura 3.53 Planteamiento general completo de controlador PLCnext conectado a plataforma Cloud.

Si bien es cierto que PLCnext dispone de su propio entorno Cloud el cual se conoce como Proficloud.io, y se introducirá más detalladamente en un apartado posterior. Para este caso, no ha sido empleado en detalle, ya no solo por su implementación directa, si no por sus limitaciones al no ser un entorno que proporcione muchas funcionalidades de manera gratuita (o al menos no las especificadas para este proyecto).

En este caso se hace interesante el ilustrar brevemente cómo implementar una programación sencilla que permita subir datos del programa Node-RED a la nube ya que en todas las plataformas Cloud (Google Cloud, IBM, Azure, AWS…) se implementa de manera semejante.

Como se indicará en los siguientes apartados el sistema Cloud por el cual se ha optado es Microsoft Azure. Este permite una rápida vinculación entre dispositivo y nube mediante el empleo de la herramienta IoT Hub de acceso completamente gratuito. Inicialmente se había realizado el desarrollo en IBM Cloud por medio de la herramienta IoT Watson (se expondrá también en siguientes apartados) pero debido a cese del soporte por parte de la plataforma de IBM de manera gratuita, se ha decidido hacer la migración de funciones a la ya citada nube de Microsoft. En todo caso, el procedimiento de actuación para ambos casos será narrado para toma de referencia y visualizar distintos procedimientos de actuación

dependiendo del entorno Cloud seleccionado, remarcando que ambas se guardan muchas similitudes.

La idea de implementación seguida para el intercambio de datos con la plataforma Cloud seleccionada por el usuario, seguiría una estructura similar a los siguientes pasos narrados:

1) Comenzaría tras la recepción de los datos en el entorno Node-RED instalado en el controlador físico, con el paso de esos datos al entorno Cloud por medio de los nodos disponibles para interacción con la plataforma de comunicación de la nube seleccionada. Un esquema gráfico de este paso se podría ver en la siguiente Figura 3.54.

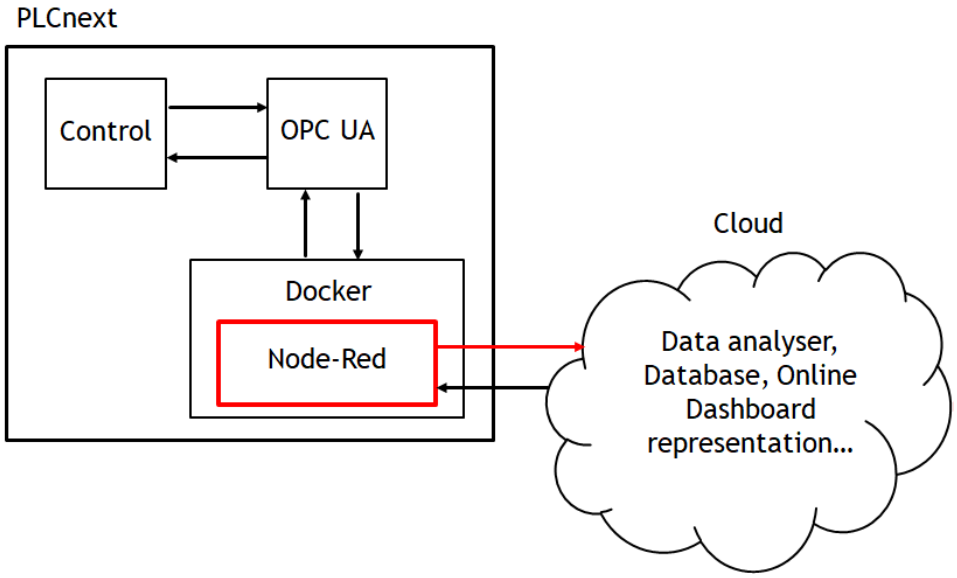

Figura 3.54 Paso de datos del controlador a la plataforma Cloud seleccionada.

2) El siguiente paso sería el tratamiento de los datos recibidos en el entorno Cloud, para ello habría que crear el Dashboard de análisis y realizar una visualización/depuración, para asegurarse de que llegan correctamente los datos a la nube. Una representación de la localización en el gráfico ya

introducido donde se encontraría este paso se puede visualizar en la siguiente Figura 3.55.

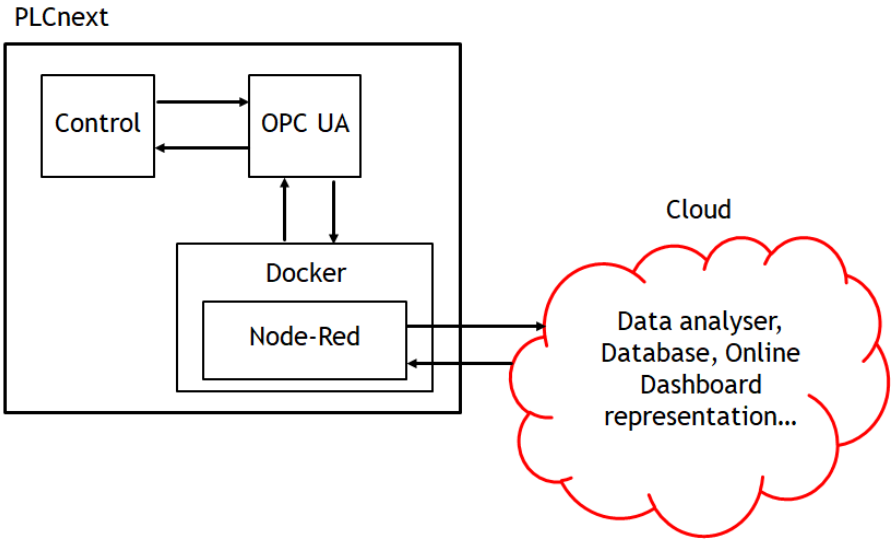

Figura 3.55 Tratamiento de datos en la plataforma Cloud.

3) Una vez que ya se disponen de los datos dentro del entorno de la nube uno de los siguientes pasos a realizar sería el realizar una instalación del entorno Node-RED en el entorno Cloud como se muestra en la siguiente Figura 3.56.

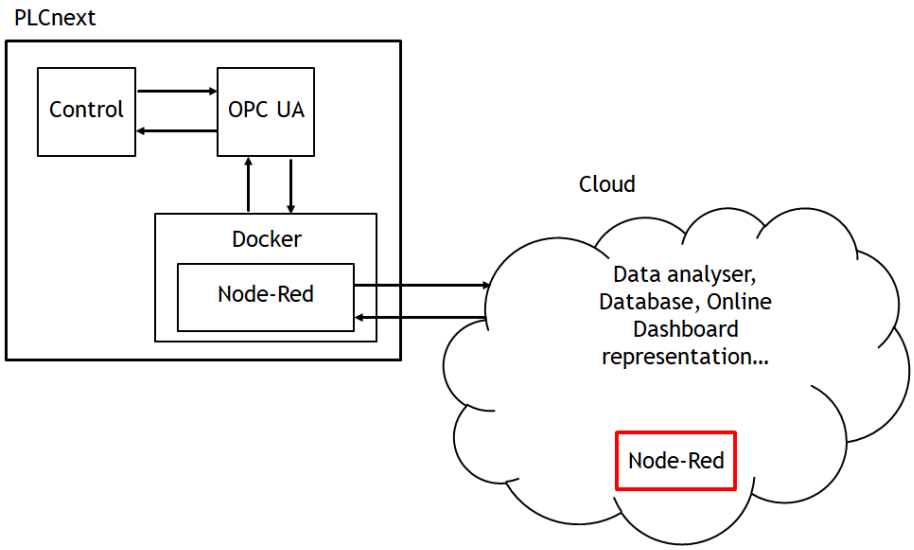

Figura 3.56 Generación de acceso a Node-RED en la plataforma Cloud.

4) Tras la instalación del entorno Node-RED, el siguiente proceso a realizar será el paso de los datos procesados en el entorno Cloud al entorno de programación gráfica Node-RED instalado en este en este entorno como se muestra en la siguiente Figura 3.57.

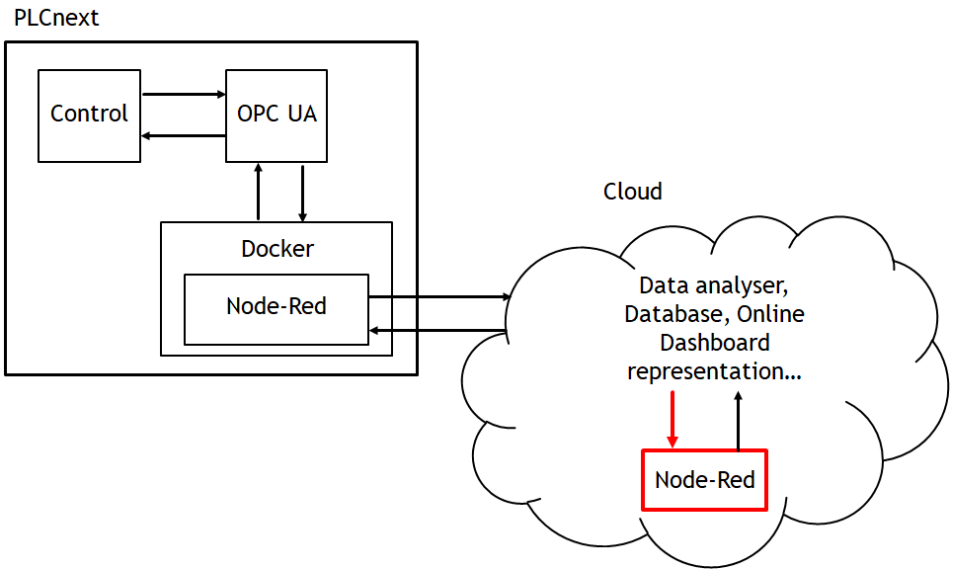

Figura 3.57 Paso de datos de plataforma Cloud a Node-RED localizado en esta.

5) Lo siguiente sería recorrer el camino inverso con los datos generados en Node-RED desde el entorno de programación gráfico Node-RED hacia el entorno Cloud de nuevo como se muestra en la siguiente Figura 3.58.

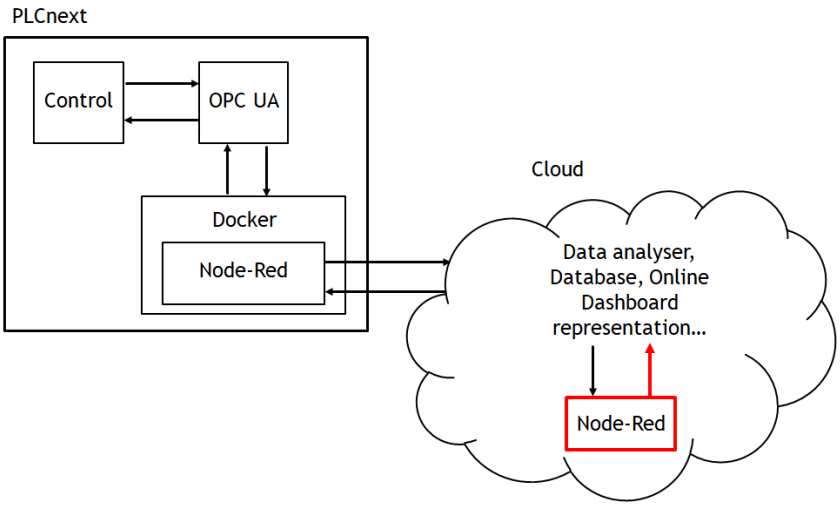

Figura 3.58 Recepción de datos en la plataforma Cloud por parte del Node-RED localizado en esta.

6) Para finalizar el proceso de interacción con el entorno Cloud será el recibir datos por parte del entorno Cloud en el entorno gráfico de Node-RED que se encuentra instalado en el controlador de PLCnext empleado a partir de los nodos que se indicarán en apartados futuros de este documento como se puede visualizar en la Figura 3.59.

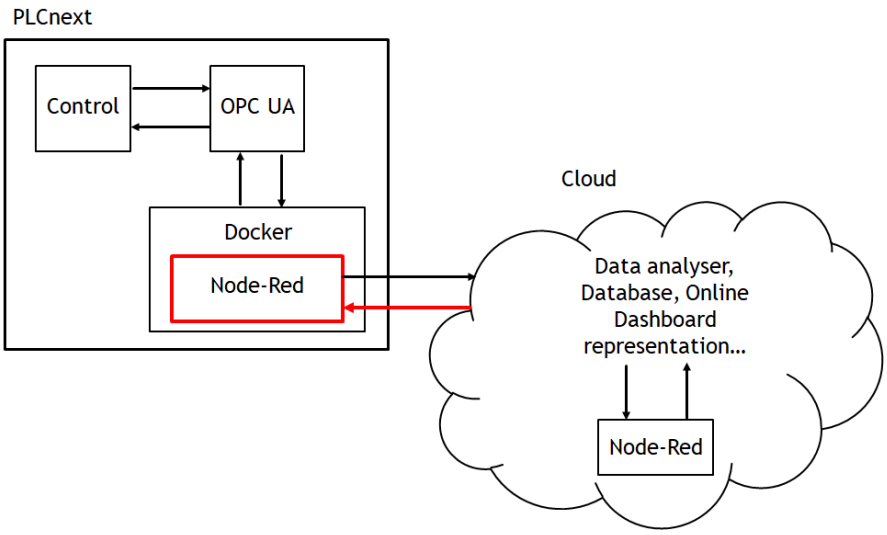

Figura 3.59 Envío de datos por parte de la plataforma Cloud al controlador PLCnext.

3.5.3- Servicio Cloud empleado

Como se ha introducido en los dos apartados previos relacionados con la parte Cloud de este documento, se ha trabajado con varias plataformas Cloud como son Proficloud.io, IBM Cloud (IoT Watson) y Microsoft Azure (IoT Hub).

3.5.3.1 Proficloud.io

Antes de incidir más firmemente sobre la plataforma Cloud seleccionada finalmente, sería recomendable el hacer una especial mención a la plataforma Cloud proporcionada directamente por el fabricante Phoenix Contact de integración directa con el controlador PLCnext. Esta es la conocida como Proficloud.io visualizada en la Figura 3.18 de acceso directo para cualquier controlador PLCnext.

Proficloud.io da la posibilidad (mediante servicio gratuito) al usuario de:

- Monitorizar y rastrear todos los dispositivos PLCnext de Phoenix Contact asociados al entorno Cloud de Proflicloud.io.
- Supervisar y rastrear la localización de estos dispositivos PLCnext de Phoenix Contact asociados al entorno Cloud de Proflicloud.io.
- Ejercer mantenimiento y actualización de los dispositivos PLCnext de Phoenix Contact asociado al entorno Cloud de Proficloud.io.
- Establecer un contacto directo *"plug and play"* entre el controlador de PLCnext y el entorno Cloud mencionado Proficloud.io.

3.5.3.2 IBM Cloud

Para el desarrollo implementado en este proyecto, se había optado inicialmente por el entorno Cloud de IBM. El logo identificativo asociado a la plataforma de IBM Cloud se puede visualizar en la siguiente Figura 3.60.

Figura 3.60 Logo de la plataforma de IBM Cloud [29].

La marca de IBM Cloud ofrece al usuario tres servicios principales: la infraestructura, el software y los servicios de plataforma. Cada uno de estos servicios proporciona al usuario diversas funcionalidades de carácter totalmente público y que permite al usuario seleccionar el plan que mejor se adapte a sus necesidades.

En este caso, se había seleccionado inicialmente esta plataforma en contraposición con otras opciones debido a que este proporciona una serie de herramientas útiles e interesantes facilitando mucho el desarrollo del proyecto, aunque claramente esta selección siempre queda a gusto del usuario y lo que requiera completar en su desarrollo concreto.

La herramienta que en este caso se empleaba como plataforma de comunicación era la llamada IoT Watson, pero IBM ha anunciado que cesa el soporte a esta herramienta por parte de la plataforma, esto sumado a que también eliminaron la posibilidad de desplegar Node-RED de manera sencilla y gratuita en el entorno Cloud, ha hecho que se opte por realizar una migración de lo desarrollado en este ámbito a la plataforma de Microsoft Azure.

Antes de destacar la plataforma Microsoft Azure finalmente seleccionada, se va a proceder a describir ligeramente que era esta herramienta IoT Watson porque servirá también de base para comprender la herramienta IoT Hub que proporciona Microsoft Azure y ver como se mantienen esas similitudes entre los diferentes entornos Cloud.

3.5.3.2.1 IoT Watson

Esta herramienta proporcionada por el entorno de IBM Cloud, cuyo logo puede visualizarse en la Figura 3.61. Consiste en un servicio incorporada en el servicio Cloud que busca simplificar y proporcionar una herramienta totalmente adaptada a las necesidades actuales de uno de los conceptos ya destacados previamente, la Industria 4.0, concretamente como el nombre de IoT Watson propiamente indica, el concepto de internet de las cosas (IoT: Internet of Things).

Figura 3.61 Logo de la plataforma de comunicación IBM Watson IoT [30].

IoT Watson se trata de una plataforma de comunicación con dispositivo que integra un panel de control potente, flexible, escalable y de sencilla utilización. Proporciona al usuario una interfaz de usuario simple y bien definida que permite ejercer un completo control y análisis sobre los dispositivos conectados a la par que los datos que se están recibiendo.

3.5.3.3 Microsoft Azure

Tras el problema que suponía que se cesase el soporte por parte de la plataforma de IBM Cloud a la herramienta IoT Watson, se optó por buscar una solución de semejantes beneficios en la cual se pudiese implementar la misma solución.

La búsqueda se acabó acotando a dos posibles alternativas. La primera podría haber sido Google Cloud, pero esta se descartó por tener que realizar varios cambios en el contenedor que alberga Node-RED en el controlador PLCnext, además de no poder implementar Node-RED en el entorno Cloud de manera sencilla y no proporcionar una herramienta IoT semejante a la que proporciona IBM. La otra alternativa y al final la solución por la que se ha optado es Microsoft Azure, cuyo logo se visualiza en la Figura 3.62, ya que permitía mantener la configuración de Node-RED ya desarrollada, a la par que proporcionar una herramienta IoT de características semejantes a la de IBM y la posibilidad de emplear Node-RED en el entorno Cloud.

Figura 3.62 Logo de la platforma Cloud Azure de Microsoft [31].

3.5.3.3.1 IoT Hub

Como ya se venía mencionando, Microsoft Azure proporciona una herramienta de comunicación con dispositivo muy semejante a la que proporciona/proporcionaba IBM Cloud (IoT Watson) salvando algunas diferencias como la de que la interfaz con el usuario no es tan avanzada, pero en contraposición IoT Hub, de logo referenciado en la Figura 3.63, proporciona más capacidad de configuración en la selección del método de comunicación con el dispositivo y establecimiento de criterios de seguridad para la autentificación del dispositivo/dispositivos.

Figura 3.63 Logo de la plataforma de comunicación de IoT Hub [32].

4. Configuración de componentes hardware y software

Una vez que ya se ha descrito todo lo necesario y empleado para llevar a cabo la implementación del proyecto y previo a entrar en la parte de manual destinado al usuario en el cual se describe la programación y el uso de la aplicación descrita para desarrollar el citado proceso, sería conveniente incluir dentro de esta descripción todas la configuraciones previas, necesarias y seguidas para poder desarrollar la programación correctamente.

4.1.- CONTROLADOR PLCNEXT

Antes de realizar cualquier programación, es interesante asegurarse que se han realizado todos los procedimientos, configuraciones y actualizaciones necesarias aplicadas al controlador.

Es preciso fijarse en la parte física del controlador teniendo en cuenta el apartado **“3.2.4. Hardware y software empleado”**. Al mencionado controlador AXC F 2152 ha de estar conectados a su derecha directamente los módulos AXL F DI8/ DO8/1 (de entradas y salidas digitales) y AXL F AI2/ AO2 (de entradas y salidas analógicas) y luego a través del acoplador de bus AXL F BK PN otro módulo AXL F DI8/ DO8/1 como se puede observar en la siguiente Figura 4.1 representando el módulo de entrenamiento EDU AXC F 2152 empleado añadiendo una indicación del conexionado del módulo acoplador bus al controlador.

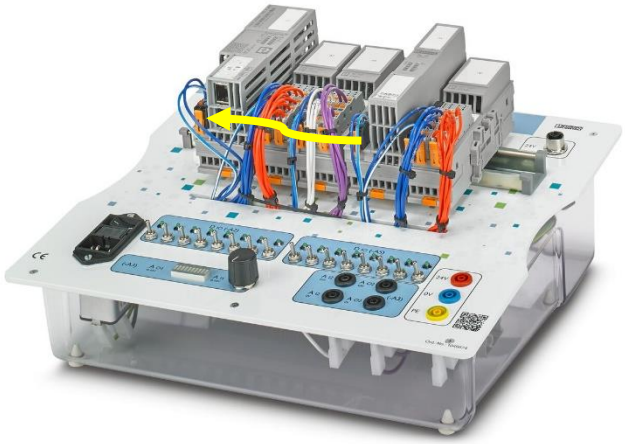

Figura 4.1 Enlazado del acoplador con el controlador PLCnext.

Una vez asegurado el conexionado, el siguiente paso en la puesta a punto del controlador AXC F 2152 es la actualización del software asociado al mismo. El software original integrado en el sistema es la versión 2019.0.4 LTS versión de hardware 02, la cual limita mucho las funcionalidades del sistema, por lo que es necesario actualizar el sistema. La actualización seleccionada para este proyecto ha sido la 2022.6.0.

Para comenzar este procedimiento de actualización lo primero es disponer en el equipo local del usuario de la versión deseada a actualizar. Para ello, se accederá a la página de Phoenix Contact asociada al controlador AXC F 2152 [33] para acceder a la versión de controlador deseada y necesaria, empleando el siguiente procedimiento:

1) En el apartado “Detalles de Producto” se desplegará la pestaña de Firmware como se puede observar en la siguiente Figura 4.2.

Figura 4.2 Acceso a la versión de firmware del controlador.

2) A continuación, se buscará la versión deseada como puede visualizarse en la siguiente Figura 4.3.

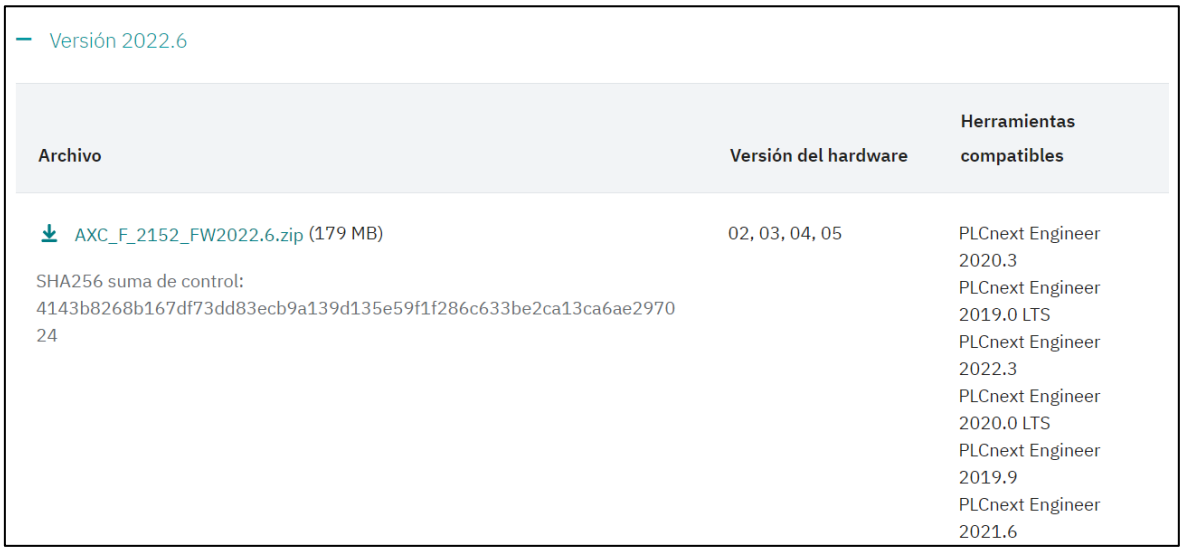

Figura 4.3 Búsqueda de la versión 2022.6 de controlador.

3) Descargado el .zip de versión deseado al sistema local del usuario, lo siguiente será descomprimir este archivo y tener localizado el archivo de extensión *.raucb* como se muestra en la Figura 4.4 ya que se empleará en la actualización automática del controlador.

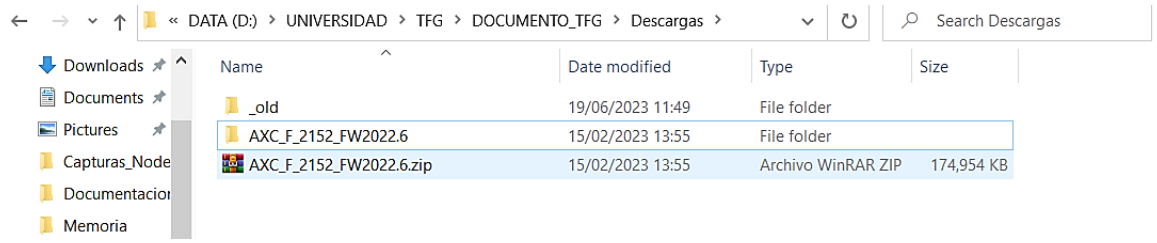

Figura 4.4 Localización del archivo .raucb asociado a la versión 2022.6 dentro del equipo local.

4) Una vez que el usuario tenga localizada la versión, el siguiente paso será acceder al webserver de configuración del controlador. Para ello, se ha de introducir en la barra de direcciones la dirección IP asociada al controlador seguida de */wmi* o */wbm*, en este caso: *192.168.0.16/wmi*, apareciendo una pantalla de bienvenida como se ve en la siguiente Figura 4.5:

Figura 4.5 Página principal tras acceder a la extensión */wmi* en el navegador.

5) Para acceder a la configuración e información el usuario ha de acceder a *Easy configuration*, ahí este dispondrá de la capacidad de verificar la información general del sistema y como es el caso que se requiere, el actualizar la versión del sistema. Una vez que el usuario acceda a *Easy configuration*, aparecerá una pantalla como la

que se puede ver en la Figura 4.6 en la que se ha de introducir los datos de acceso al controlador.

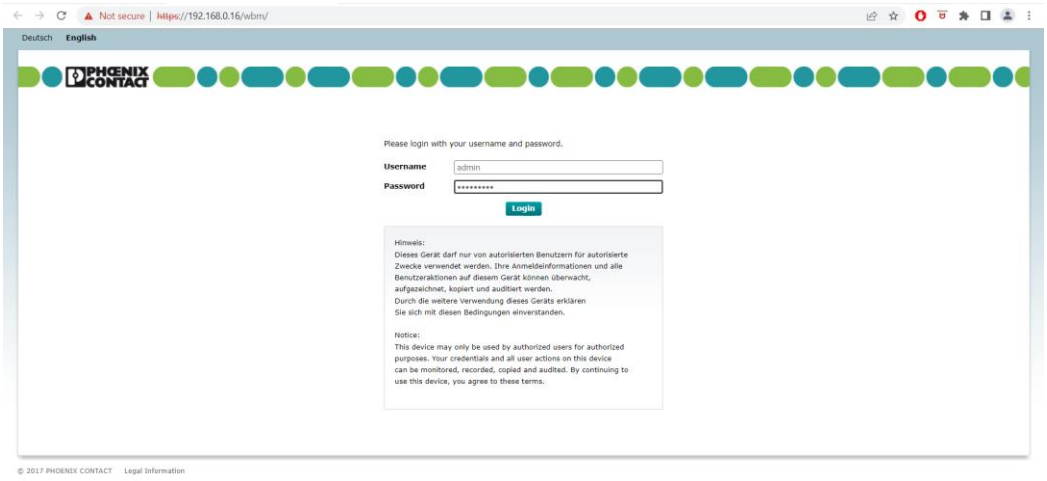

Figura 4.6 Solicitud de credenciales del controlador para acceder a la configuración de este.

6) Cuando el usuario se identifique con los datos de acceso al controlador aparecerá la pantalla visualizada en la siguiente Figura 4.7 en la cual ya se pueden observar datos básicos como el programa cargado y la versión instalada en el controlador.

Figura 4.7 Página principal tras realizar la autentificación por parte del usuario.

7) Si el usuario deseara consultar más información asociada al hardware, tendrá que acceder a la pestaña “Information -> General Data” en la cual se mostrará información del sistema de manera similar a la que se muestra en la Figura 4.8.

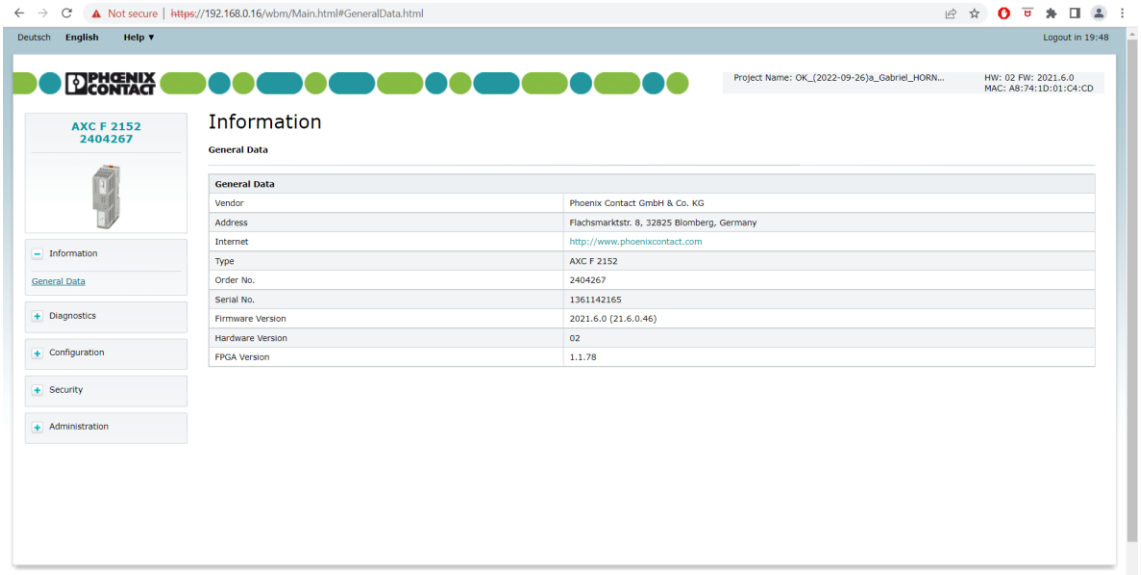

Figura 4.8 Pantalla de información general del controlador PLCnext.

8) En caso de querer actualizar el sistema, se debería proceder de la siguiente manera, “Administration -> Firmware Update” visualizando una pantalla semejante a la de la Figura 4.9.

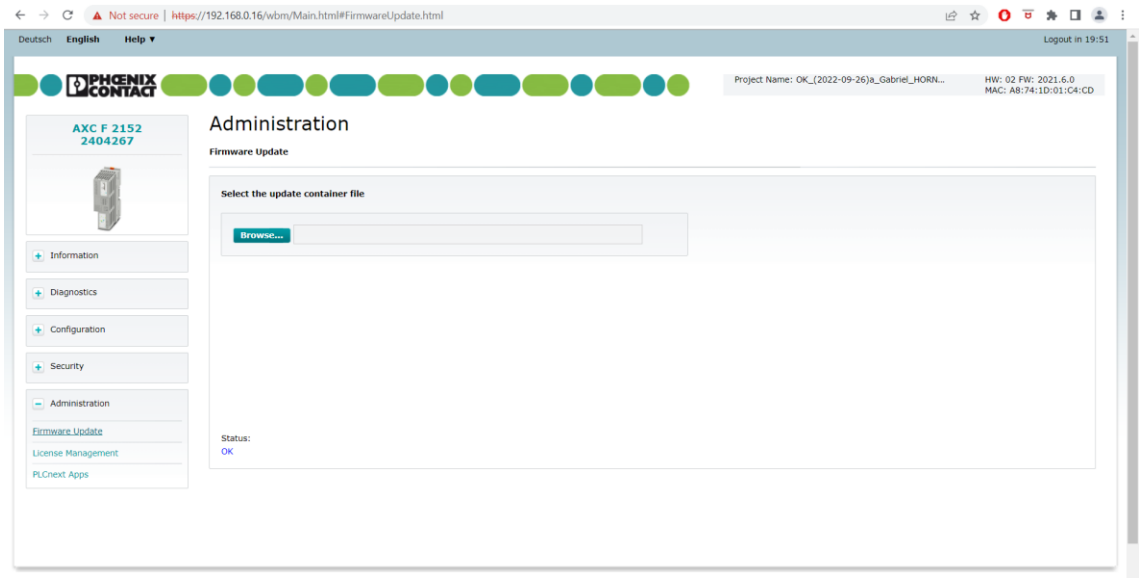

Figura 4.9 Acceso a la pestaña de actualización del firmware del controlador PLCnext.

9) Finalmente el usuario deberá pulsar en “Browse…” y buscar el archivo de versión deseado con la extensión *.raucb* y abrirlo como se puede ver en la siguiente Figura 4.10.

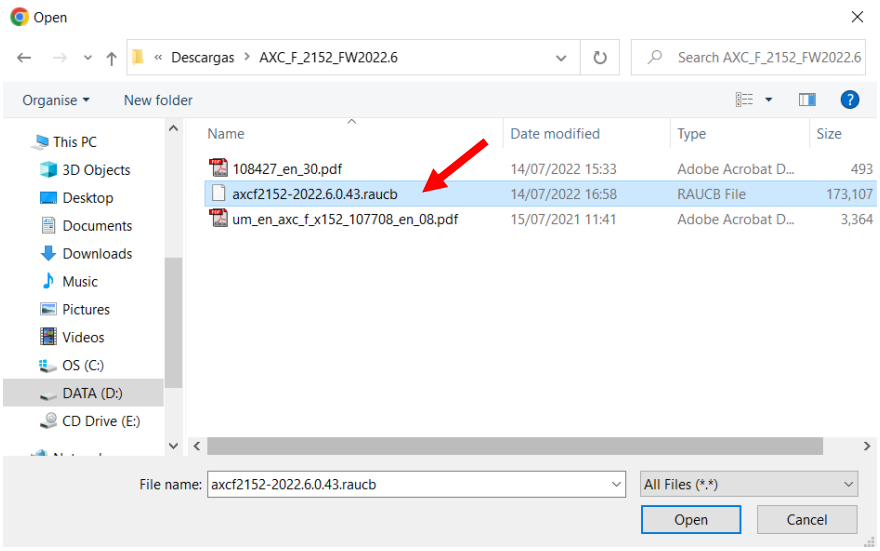

Figura 4.10 Buscar en el equipo local la actualización deseada del controlador PLCnext.

Con esto el usuario ya dispondrá de un controlador con la versión actualizada a la deseada como se muestra en la Figura 4.11 y con un mayor número de funcionalidades habilitadas, necesarias para desarrollar el resto de las tareas requeridas por el proyecto.

Serial No.	1361142165
Firmware Version	2022.6.0 (22.6.0.43)
Hardware Version	02

Figura 4.11 Verificación de firmware ya actualizado en la pestaña de dtos generales del controlador PLCnext.

Quedará por lo tanto una verificación más que hacer al controlador físico para dejarlo completamente preparado para su utilización. Se trata de comprobar que el reloj interno del controlador coincide con el reloj UTC. El UTC es el reloj universal coordinado y es imprescindible que coincidan para que se puedan aplicar las funcionales Cloud dentro del proyecto.

El UTC se puede consultar en cualquier momento en la página de time.is [34]. El siguiente paso será consultarlo en el controlador, para ello el usuario ha de emplear las herramientas ya previamente introducidas que son WinSCP y Putty para realizar la comprobación del reloj interno y en caso de no ser correcto, el poder actualizarla.

Para verificar la fecha en la cual se encuentra el controlador del usuario será necesaria la introducción de los siguientes comandos como se puede ver en la siguiente Figura 4.12 contrastando con la hora y fecha UTC.

Identificación del usuario raíz:

su

7ab546092023

Comando de consulta de fecha y hora del controlador:

date

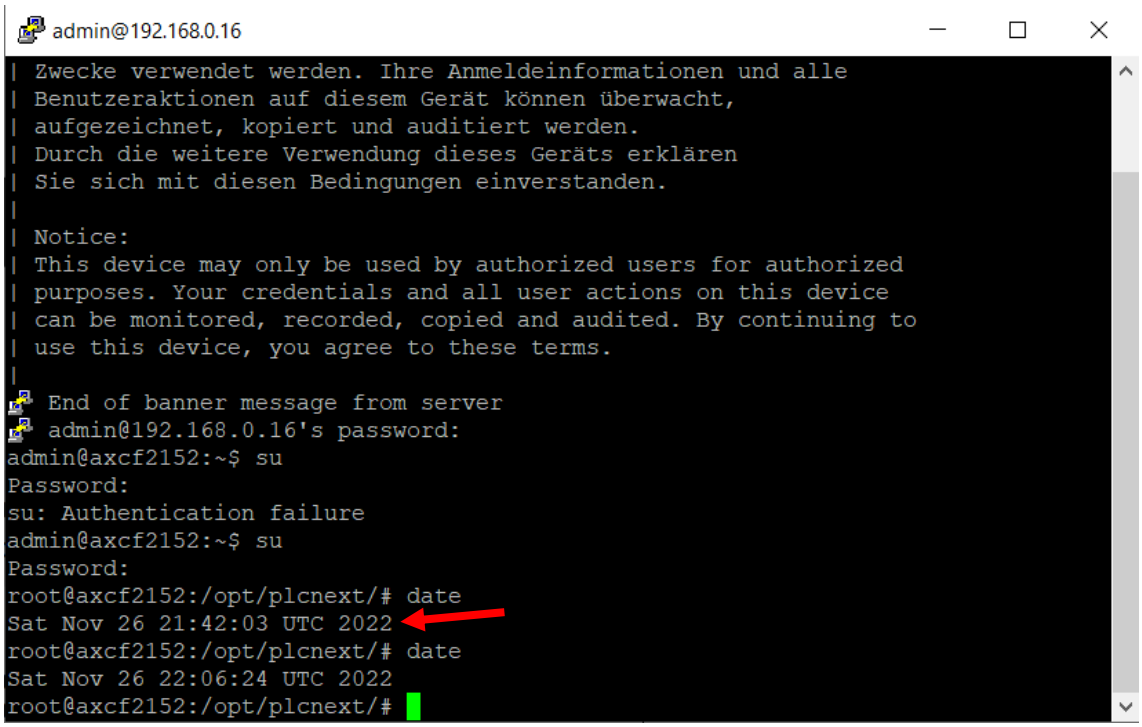

Figura 4.12 Consulta de fecha y hora asociadas al controlador PLCnext.

En caso de que no coincidiese la fecha del controlador con la UTC, ni en día, ni en año, el controlador no podrá acceder a servicios proporcionados por la conexión a internet y habría que proceder de la siguiente manera:

1) Copiar vía WinSCP el fichero ntp.conf al directorio /opt/plcnext/
2) Abrir una sesión Putty en el PLCnext.
3) Identificarse con el usuario root.
4) Una vez hecho el log con el usuario root ejecutar el siguiente comando.

mv /opt/plcnext/ntp.conf/etc/ntp.conf

5) A continuación, ejecutar el siguiente comando para reiniciar el servicio ntp.

/etc/init.d/ntpd restart

6) Ejecutar el comando citado a continuación para ver la lista de servidores ntp con los que se conecta el equipo. En cuanto este sincronizado con un servidor, este aparecerá con un asterisco "*" delante.

ntpq -p

7) De nuevo ejecutar el comando **date** para verificar que es correcta la fecha configurada ahora en el equipo.

Este es el procedimiento habitual para versiones de controlador inferiores a la 2022, el cual es relativamente complejo y requiere completar distintos pasos, en este caso como se ha realizado la actualización a la versión de controlador 2022.6, este proceso se puede simplificar en gran medida ya que se puede realizar desde el mismo webserver del controlador.

El usuario podría acceder directamente como ya se ha mencionado al apartado /wbm o /wmi del controlador accediendo simplemente al apartado: Configuration/Date and Time como se muestra en la Figura 4.13 y apareciendo una pantalla semejante a la vista en la Figura 4.14 donde se podrá ver el servidor de configuración de fecha y hora empleado por el controlador e incluso la posibilidad de que el propio usuario incorpore acciones alternativas.

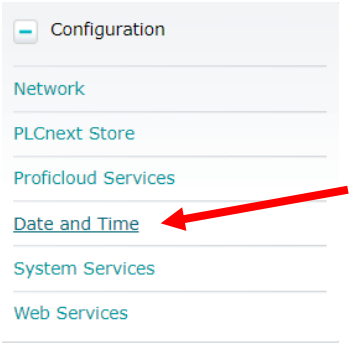

Figura 4.13 Acceso a la pestaña de configuración de fecha y hora sobre el webserver del controlador PLCnext.

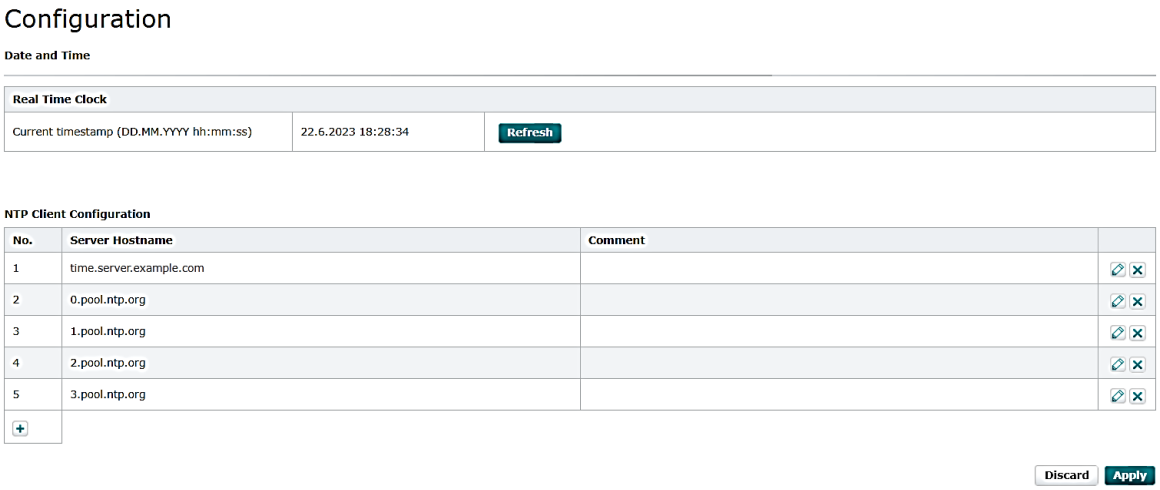

Figura 4.14 Pestaña de configuración de fecha y hora en el webserver del controlador PLCnext.

Una vez que el controlador ya está actualizado a la versión deseada y que la hora interna está en concordancia con la UTC, además de la correcta instalación de Node-RED, el controlador ya estará capacitado a nivel de configuración para poder desarrollar las necesidades del proyecto.

Sin embargo, aún no está todo el apartado de configuración definido ya que aún falta por indicar como ha sido configurada/preparada toda la parte de entorno software empleado para la realización del proyecto. Es por ello que a continuación se describirá toda la

configuración seguida para la primera implementación realizada, la cual fue fuera del propio entorno PLCnext Engineer de Phoenix Contact y es el ya citado previamente, Codesys.

4.2.- CONFIGURACIÓN DE CODESYS 3.5

La primera configuración importante en Codesys es asegurarse de la coincidencia en la selección de la versión, que como se venía diciendo, se ha desarrollado en Codesys V3.5 SP16 Patch 3, por lo tanto, en la configuración de proyecto, para la compilación y visualización hay que asegurarse de marcar esa versión como se muestra en la Figura 4.15 y Figura 4.16.

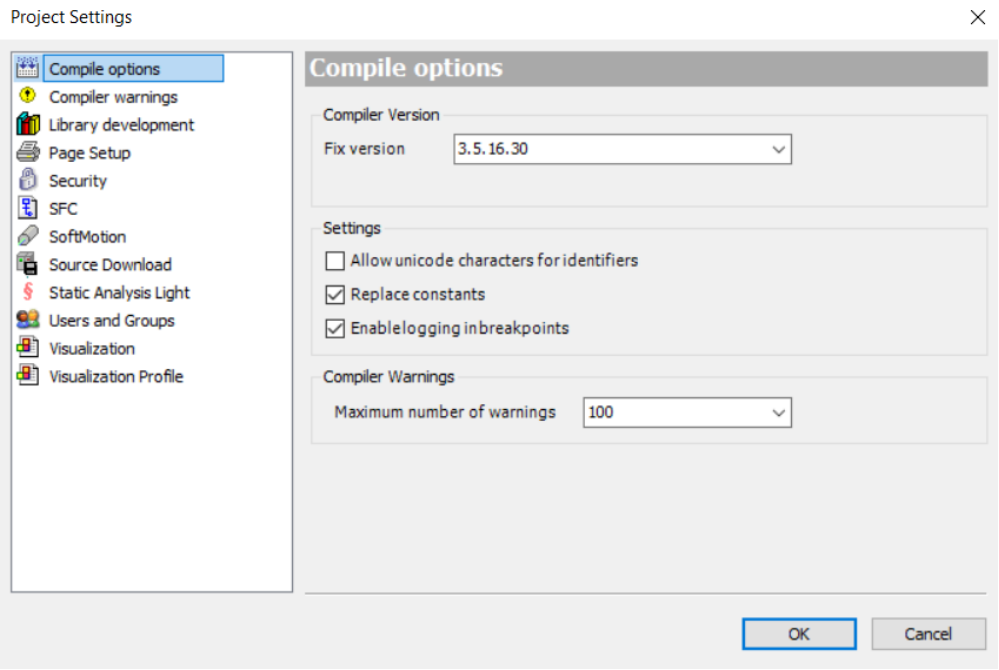

Figura 4.15 Configuración de la versión de compilador en Codesys.

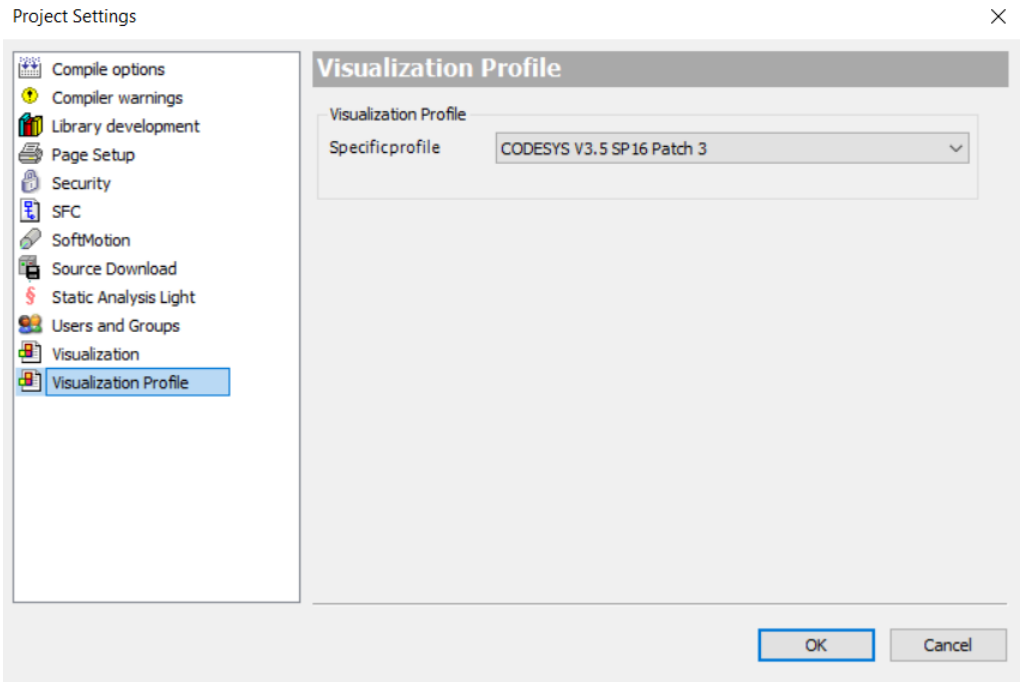

Figura 4.16 Configuración de la versión del perfil de visualización en Codesys.

Tras este paso, hay que asegurarse de poner el dispositivo en modo simulación como se muestra en la siguiente Figura 4.17.

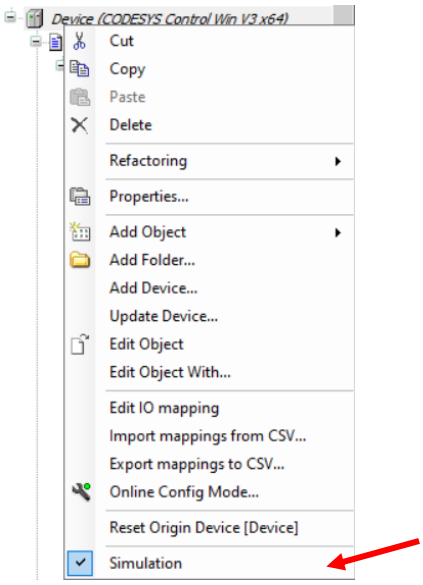

Figura 4.17 Configuración del modo simulación en Codesys.

Visualizándose el modo simulación en la pestaña de dispositivo como se observa en la Figura 4.18.

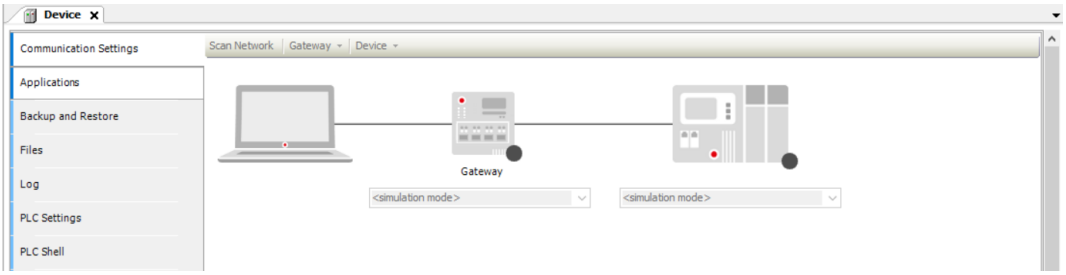

Figura 4.18 Vista del modo configuración ya seleccionado en Codesys.

Una vez ya se ha introducido toda la configuración asociada al programa desarrollado en Codesys, el siguiente paso ya por orden de realización del proyecto es la descripción de la configuración seguida en PLCnext Engineer, ya que tras la implementación del proyecto de manera simulada en Codesys el siguiente proceso fue el trasladar todo el código a PLCnext Engineer para poder operar con el controlador físico AXC F 2152, incluido en el entrenador EDU AXC F 2152.

4.3.- CONFIGURACIÓN DE PLCNEXT ENGINEER

La configuración general de proyecto se distribuye de la siguiente manera visualizada en la Figura 4.19.

Figura 4.19 Distribución general de proyecto en PLCnext Engineer.

Como se puede observar en la Figura 4.19 y si se procede describiendo con el orden visualizado de arriba abajo se irán encontrando los siguientes apartados a entender:

1) axcf2152: AXC F 2152: Inclusión en el proyecto del controlador específico empleado para este proyecto; como se puede observar su inclusión ya refleja determinados condicionantes ligados a este controlador. Por ejemplo, uno de estos condicionantes directamente generados tras la inclusión del controlador es la visualización de los dos núcleos de operación (ESM1 y ESM2) en contraposición por ejemplo a un controlador AXF F 1152 el cual si fuese incorporado se vería que solo tiene un núcleo de operación (ESM1) como se puede visualizar en la Figura 4.20.

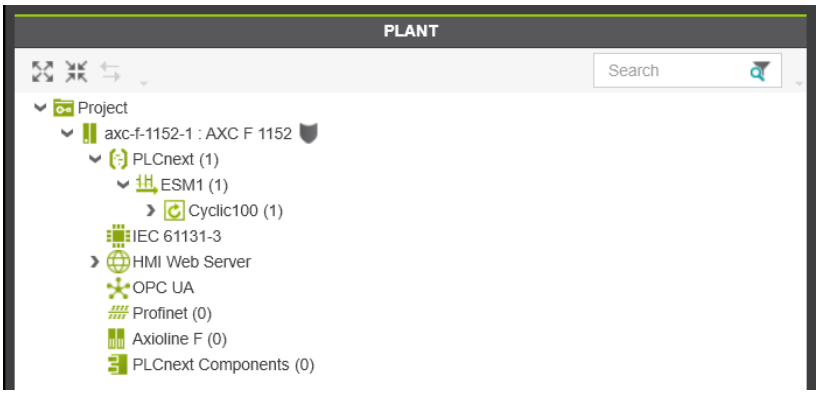

Figura 4.20 Comparación con el controlador AXC F 1152.

En este caso para la realización de la programación de proyecto solo se ha requerido el uso de uno de los núcleos de operación como se puede observar en la previa Figura 4.19.

Ahora que ya se ha visto cómo sería la estructura final de configuración, lo siguiente a comentar en este apartado sería el cómo alcanzar esta estructura de configuración.

Lo primero sería la incorporación del controlador al programa de PLCnext Engineer, para ello habría que proceder de la siguiente manera:

1) En la parte derecha del entorno PLCnext Engineer se encuentra ‘COMPONENTS’

2) A continuación, dentro de ‘COMPONENTS’ ha de entrarse en el apartado de Network.

3) Dentro del apartado de Network se ha de generar la siguiente cadena de accesos: Axiocontrol -> Devices -> Phoenix Contact -> Axiocontrol -> Controller

4) En la carpeta Controller hay que seleccionar el controlador específico que en este caso ha sido: **AXC F 2152 Rev. >= 00/2022.6.0** quedando finalmente este procedimiento general de seguimiento para encontrar el controlador: COMPONENTS -> Network -> Axiocontrol -> Devices -> Phoenix Contact -> Axiocontrol -> Controller -> AXC F 2152

Rev. >= 00/2022.6.0 como se ve en la siguiente Figura tomando ese controlador y arrastrándolo hacia la parte de PLANT en el apartado Project, ya sea haciendo un **'Drag and drop'** o botón derecho copiar y pegar sobre Project.

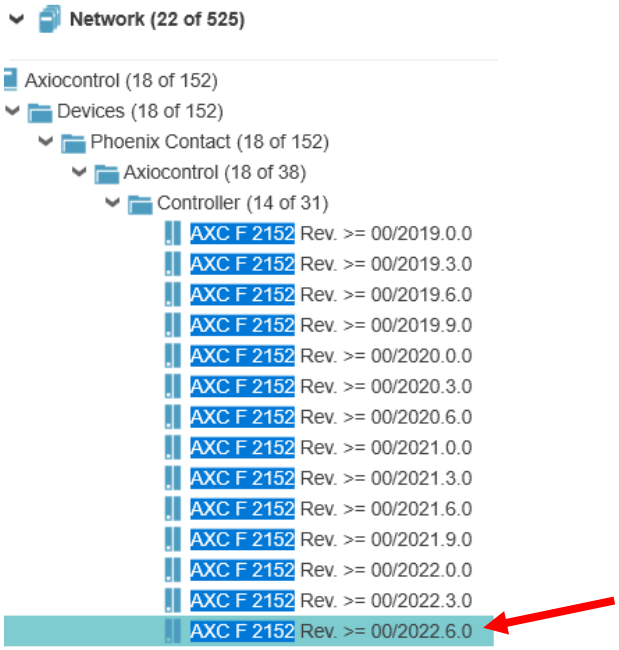

Figura 4.21 Selección del controlador adecuado en el entorno PLCnext Engineer.

Añadido el controlador asociado a la versión instalada el siguiente punto corresponde con el enlace del controlador físico para ello, lo primero y esencial es que la IP del equipo personal del usuario en el que se va a realizar la programación asociada se encuentre dentro de la máscara IP de controlador físico y con una terminación distinta a la de este.

Puesto que la IP asociada al controlador físico es 192.168.0.16 y la máscara empleada es 255.255.255.0 la IP del ordenador personal del usuario ha de ser 192.168.0.X. En este caso, para no colisionar con otros equipos incorporados dentro de la red (el controlador PLC tiene la IP 192.168.0.16 para poder prestarle acceso a internet dentro de la red del laboratorio 2.2.19 del Área de Ingeniería de Sistemas y Automática del Campus de Gijón) se ha asociado el equipo personal la IP: 192.168.0.121 como se puede visualizar en la Figura 4.22.

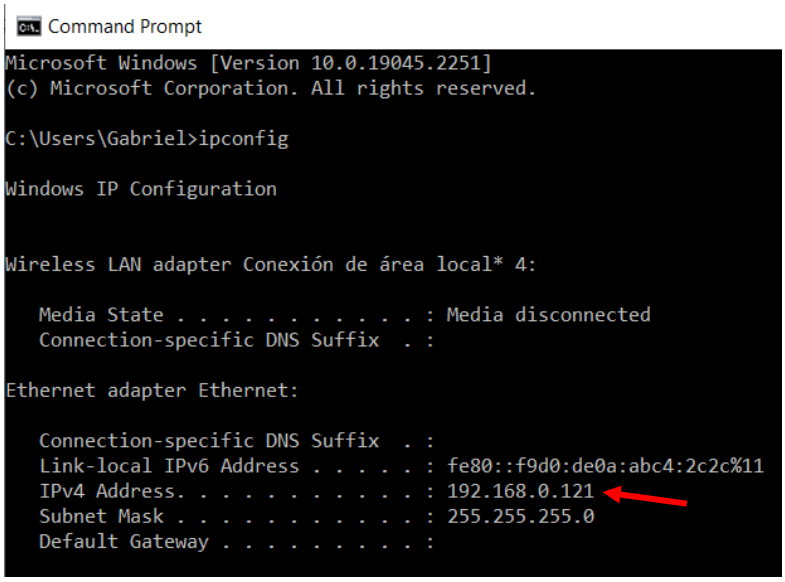

Figura 4.22 Configuración de IP asociada al equipo personal del proyectante.

Una vez configurada la IP, el siguiente paso sería acabar de preparar todas las configuraciones en PLCnext para completar el enlace con el controlador PLCnext AXC F 2152.

Se debe configurar la planta para que concuerden todos los datos con los previamente mencionados y que realice un correcto enlace. En la pestaña Settings dentro de la pestaña de Project (PLANT -> Project -> Settings -> IP subnet) se deben introducir las IP de inicio de escaneo: 192.168.0.1, la de final de escaneo: 192.168.0.253, la máscara empleada: 255.255.255.0 y finalmente el Gateway por defecto, el cual en este caso es la IP asociada al router dentro del laboratorio cuyo valor es: 192.168.0.99 como se puede observar en la Figura 4.23.

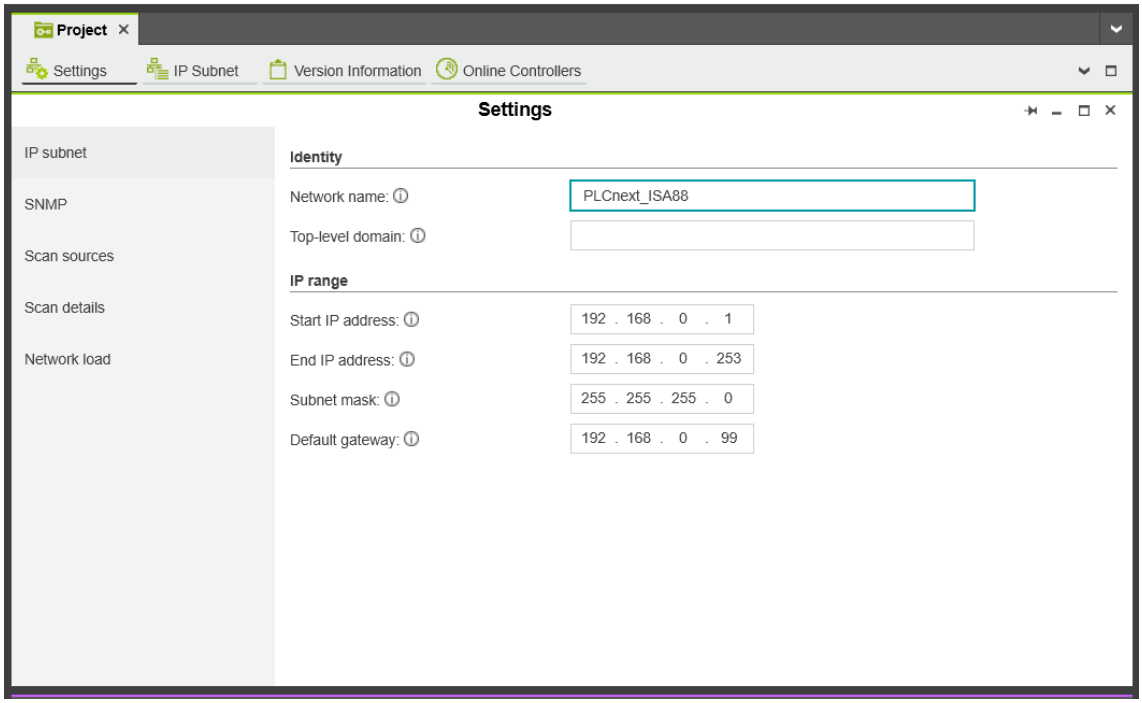

Figura 4.23 Configuración de redes en el entorno PLCnext Engineer (I).

Posteriormente en la pestaña asociada al controlador accediendo a la pestaña axcf2152 : AXC F 2152, localizada en una subdivisión asociada a Project. Dentro de la pestaña axcf2152, lo siguiente sería acceder a la pestaña Settings dentro de esta y en All ya se podrá acabar de completar la información recomendable para completar el proceso de configuración.

Lo más recomendable para asegurar el correcto conexionado con el controlador sería realizar en el apartado TCP/IP [Profinet] introducir la IP asociada al controlador: 192.168.0.16, la máscara asociada: 255.255.255.0 y el Gateway específico de conexión también previamente citado: 192.168.0.99. Para que permita introducir estos valores de manera manual por parte del usuario hay que seleccionar el modo “manual” en el apartado “IP address assigment mode”. Todos estos pasos se pueden visualizar en la siguiente Figura 4.24.

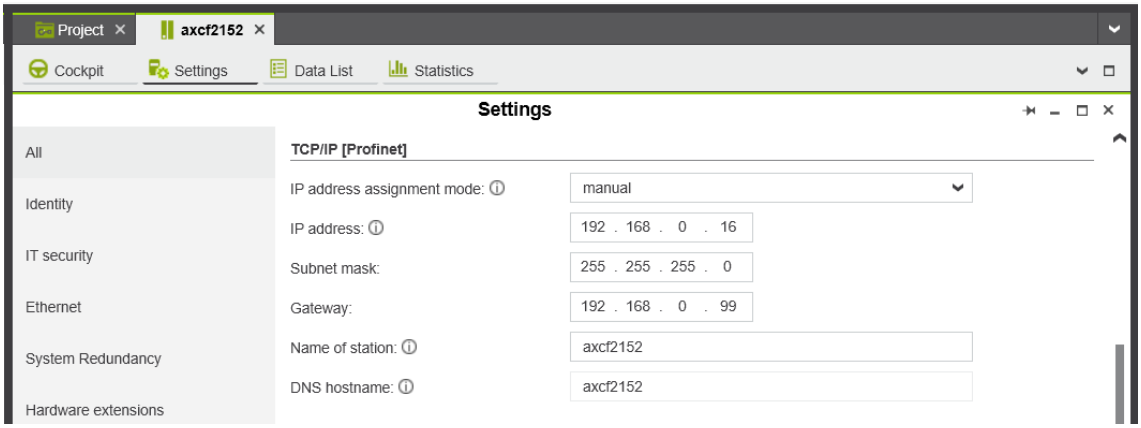

Figura 4.24 Configuración de redes en el entorno PLCnext Engineer (II).

Finalmente, el usuario ha de volver a la pestaña Project ya previamente citada la cual tendrá el siguiente aspecto que se visualiza en la Figura 4.25.

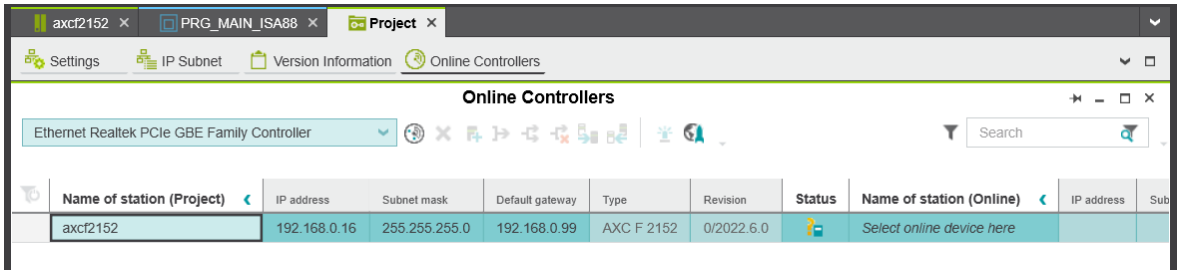

Figura 4.25 Enlace de controlador físico con el entorno de PLCnext Engineer (I).

Como se puede observar aún no hay controlador online, para disponer del controlador físico conectado con el entorno software, habrá que escanear en busca del mismo, asegurándose el usuario de que se encuentre directamente conectado al módulo controlador o en su defecto se encuentra dentro de la misma red de internet. Para realizar el escaneo se pulsará sobre (“Scan the network”) como se visualiza en la Figura 4.26.

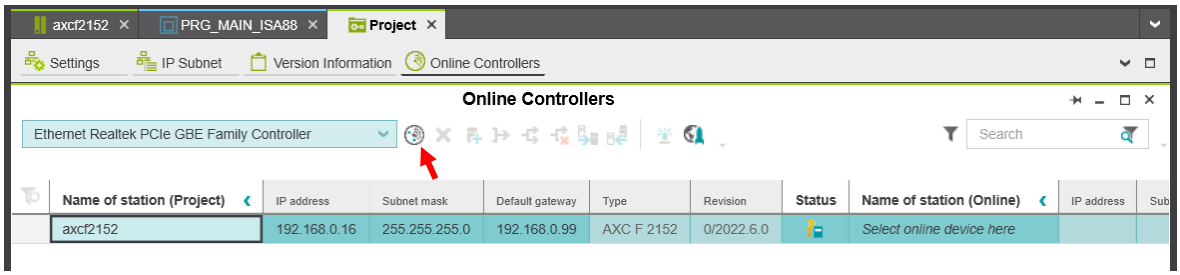

Figura 4.26 Enlace de controlador físico con el entorno de PLCnext Engineer (II).

Si el controlador físico ha sido localizado y enlazado correctamente con el equipo del usuario en la casilla Status se podrá visualizar un tick verde ( ) como se puede observar en la siguiente Figura 4.27.

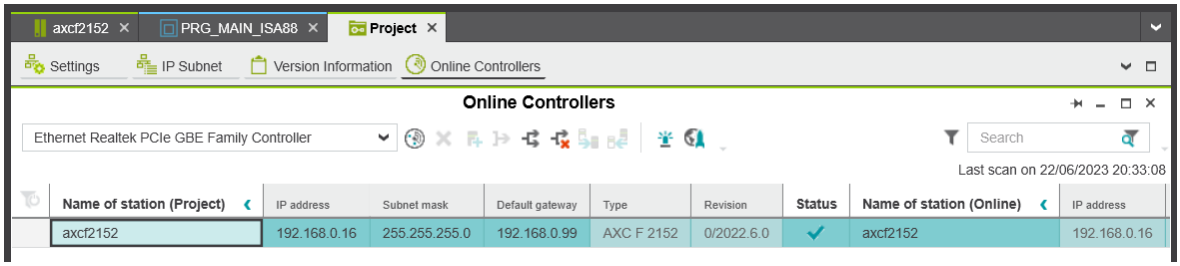

Figura 4.27 Establecido el enlace entre controlador físico y entorno PLCnext Engineer.

Con esto el usuario ya podrá empezar a operar en PLCnext Engineer con el controlador físico para realizar todas las pruebas de programa que sean necesarias. Sin embargo, sería conveniente realizar un último comentario de configuración dentro de este entorno PLCnext Engineer ya que ha sido empleado y esta es la configuración del servidor OPC UA incorporado en el controlador AXC F 2152.

4.3.1- OPC UA Server

Uno de los elementos importantes a configurar como ya se venía comentando en el apartado de configuración del entorno de programación PLCnext Engineer es el servidor OPC UA.

Para realizar la configuración de dicho servidor el usuario deberá acceder a la pestaña OPC UA localizada dentro de Project en el apartado de PLANT. Siguiendo el siguiente orden: PLANT -> Project -> OPC UA como se puede observar en la siguiente Figura 4.28.

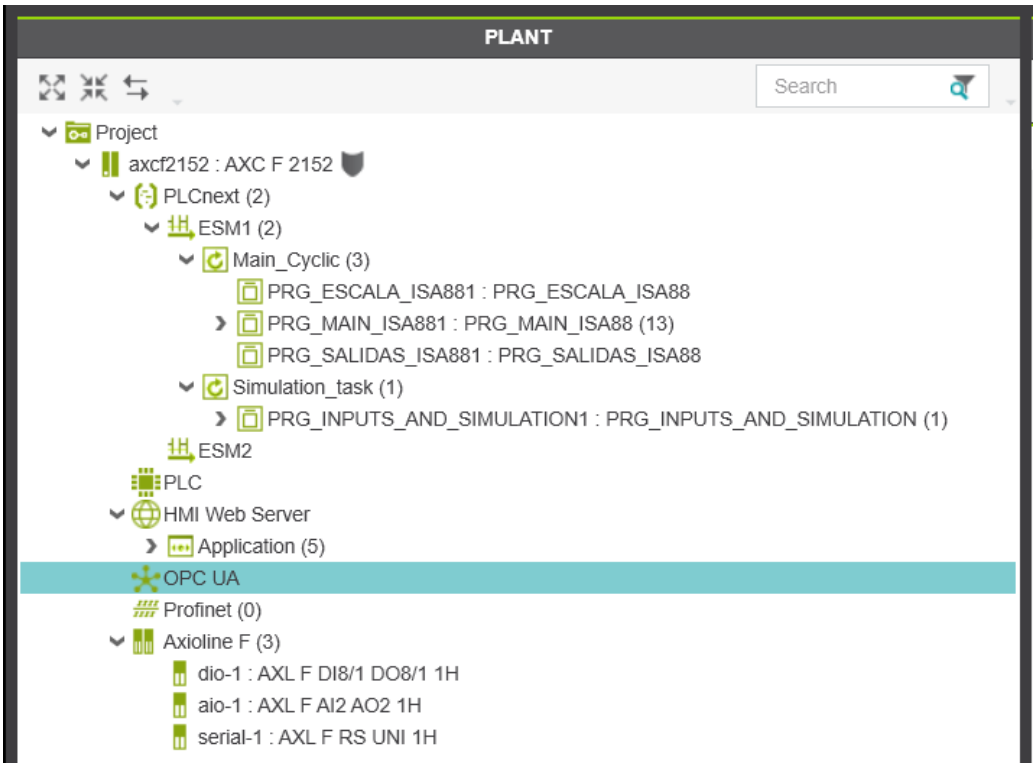

Figura 4.28 Localización del apartado destinado a la configuración del servidor OPC UA en PLCnext Engineer.

En el apartado de ‘Basic Settings’ asegurarse que en: Information model -> Visibility of variables la opción seleccionada sea ‘Marked’ como se puede observar en la Figura 4.29.

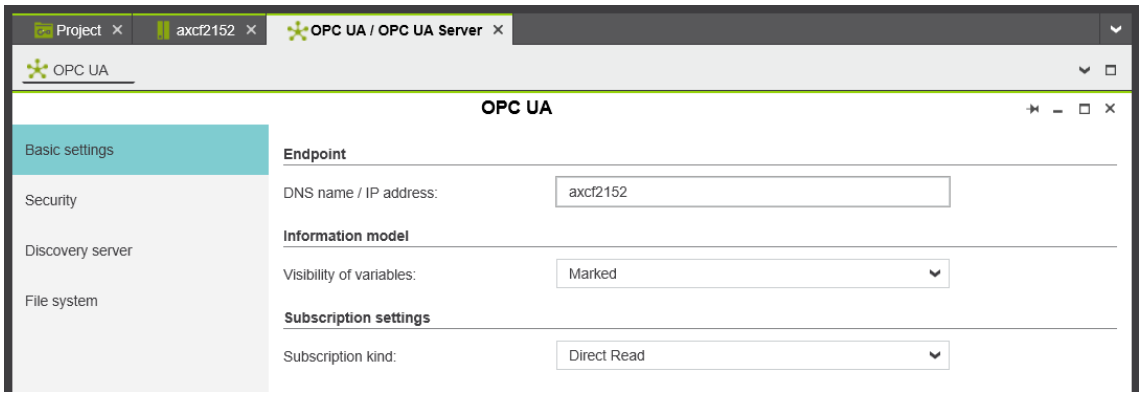

Figura 4.29 Página principal de configuración del servidor OPC UA.

NOTA: Endpoint -> DNS name / IP address se mantiene el nombre asociado al proyecto "axcf2152", sin embargo, para clarificar y asociar de manera directa el controlador físico empleado, se podría indicar directamente la IP asociada al controlador físico, la cual en este caso sería: 192.168.0.16 como en la siguiente Figura 4.30.

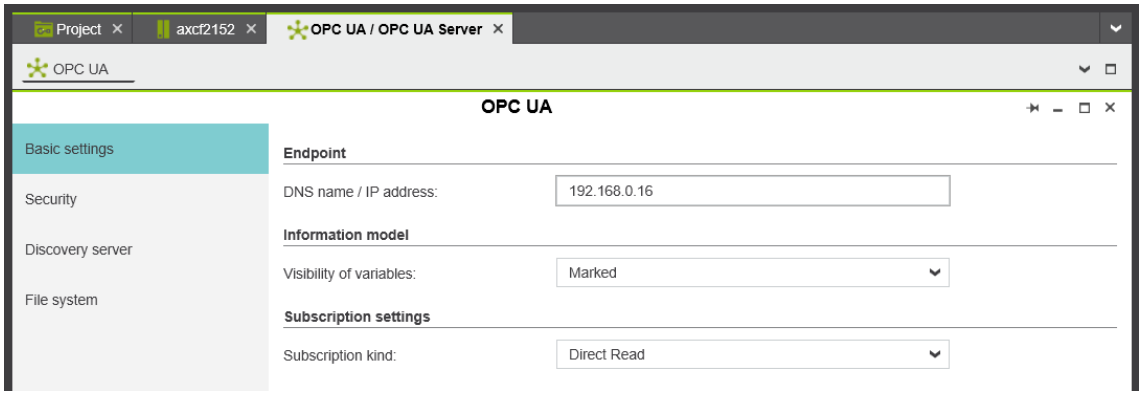

Figura 4.30 Alternativa de nombre para el servidor OPC UA.

Otro de los elementos que sería posible configurar a gusto del usuario es la seguridad asociada al servidor. Para ello simplemente bastaría con acceder al apartado 'Security'. Por ejemplo, para el caso de este Proyecto implementado por ejemplo se ha marcado un algoritmo básico 256 (256 bits de encriptado) de ingreso y encriptado como política de seguridad como se puede visualizar en la Figura 4.31.

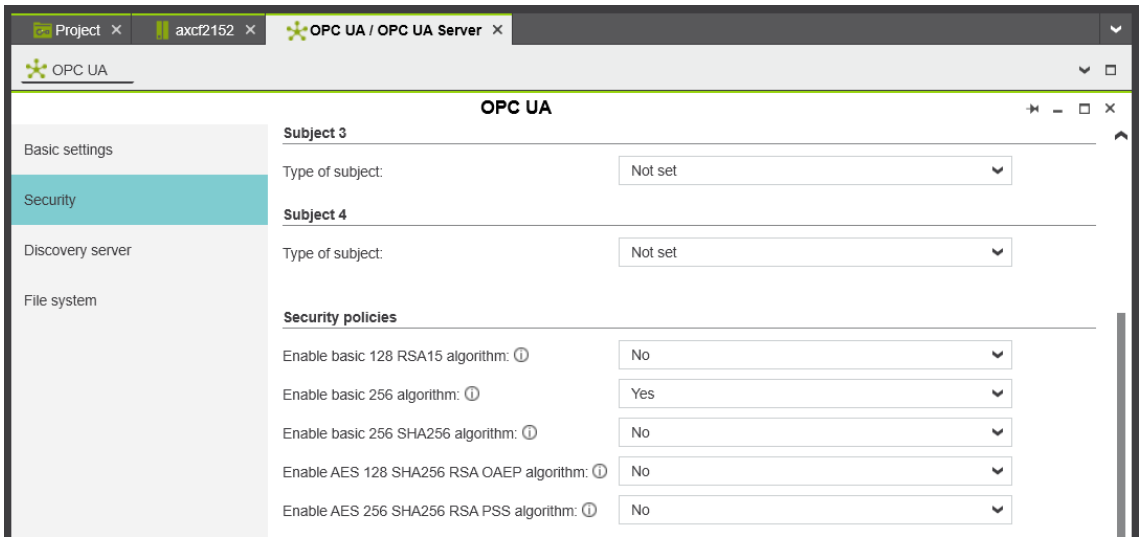

Figura 4.31 Configuración de la seguridad asociada al servidor OPC UA.

Esto lo que hará es solicitar al usuario identificarse antes de acceder al servidor OPC UA por medio de los elementos de identificación del PLC (usuario y contraseña). Esta política de seguridad se podría también eliminar para con ello eliminar cualquier tipo de identificación o encriptación asociada al servidor. Obviamente esta práctica no es lo más recomendable porque dejaría el servidor totalmente desprotegido, sin embargo, para este entorno de carácter educativo y cerrado de operación podría ser también una opción a aplicar. Simplemente para conseguir esta eliminación de cualquier política de seguridad habría que marcar todas las opciones con 'No' de la manera que se puede observar en la Figura 4.32.

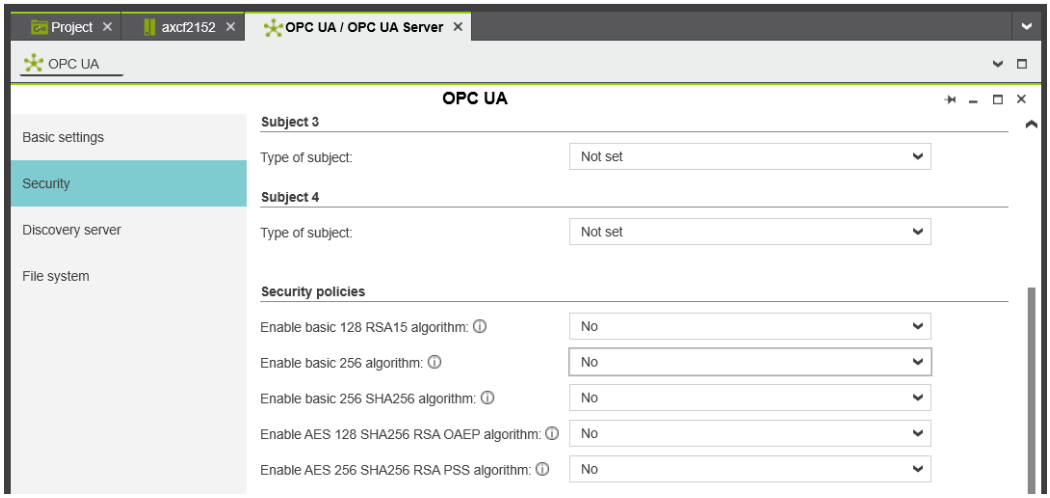

Figura 4.32 Servidor OPC UA configurado sin ningún tipo de seguridad asociada.

Con esto ya se completaría la configuración básica y necesaria del servidor OPC UA, solo restaría seleccionar las variables que el usuario desee incorporar al servidor, pero esto se tratará con más detalle en el apartado **"5. Programación global del sistema"**.

NOTA: Es importante tener en cuenta que la configuración de este servidor OPC UA está limitada al usuario por lo que al menos en esta versión implementada (2021.6, ya previamente citada) se carece por ejemplo de la posibilidad de historizar variables, modificación de la organización de variables y uno de los elementos importantes que se mencionará también en el apartado de programación, es que en este servidor implementado OPC UA no se dispone de la posibilidad de tener más de 5 clientes simultáneos conectados al servidor. Esto será un elemento clave para planificar la programación asociada a Node-RED.

4.4.- CONFIGURACIÓN DE NODE-RED Y NODOS NECESARIOS

Ya en apartados previos de este documento, se ha indicado la instalación de Node-RED en el controlador AXC F 2152 por medio de las herramientas, WinSCP y Putty. Sin embargo, no se ha indicado la configuración asociada al Node-RED empleado.

Una vez que se acceda a Node-RED a través del navegador introduciendo 192.168.0.16:1880 e identificándose con el usuario y contraseña introducido por el usuario, aparecerá una pantalla como la visualizada en la Figura 4.33.

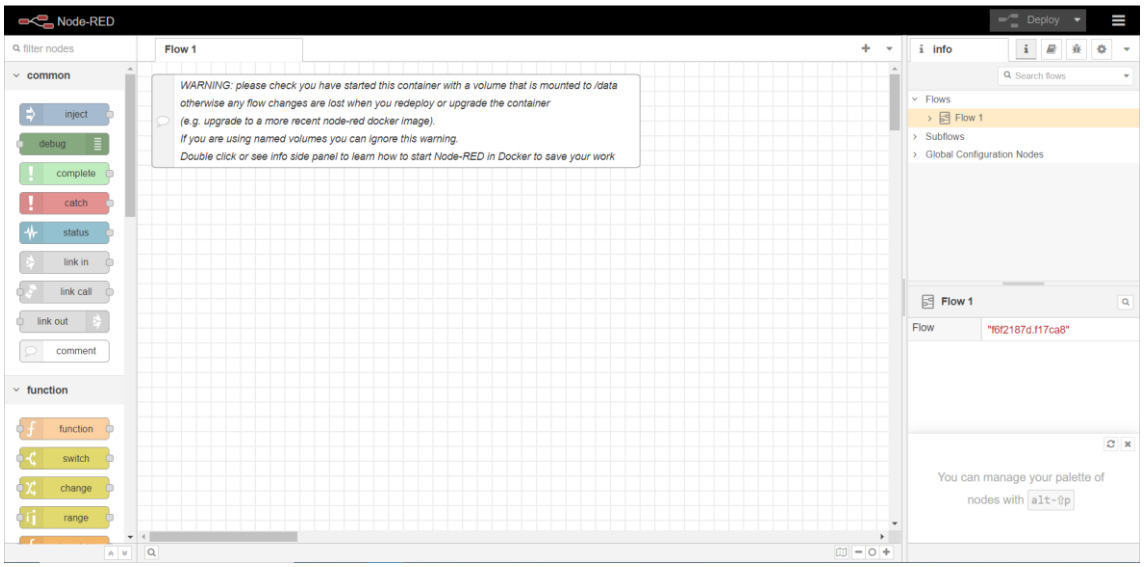

Figura 4.33 Primera visualización de programa en Node-RED tras la identificación realizada.

Se ve que se arranca con un “Flow” vacío en el cual se podrá iniciar a programación asociada a Node-RED, pero cabe destacar que falta un paso vital y es que Node-RED carece de los nodos necesarios para poder implementar esta programación citada.

Los tres nodos que el usuario necesitará añadir para poder proceder con la programación serán los siguientes:

1) **node-red-contrib-opcua** (en este caso v0.2.309) (Figura 4.34): Este paquete de nodos añadirá los nodos necesarios para la conexión con el servidor OPC UA. Permitirá la inclusión de nodos cliente y buscador, de los cuales se especificará posteriormente su funcionamiento.

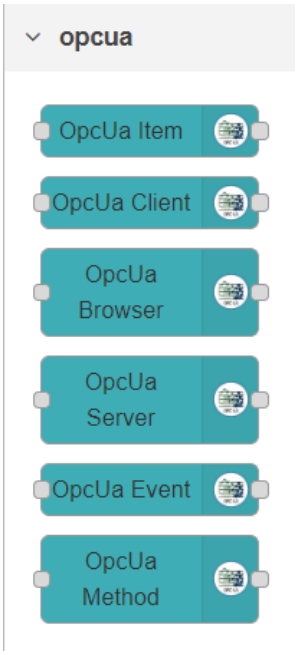

Figura 4.34 Paquete de nodos de OPC UA.

2) **node-red-contrib-ibm-watson-iot** (v0.2.8) (Figura 4.35): Este paquete permitía el uso de nodos que el usuario empleará para enviar y recibir datos de la herramienta de IoT Watson de IBM, se mantienen por realizar pruebas mientras que se mantenga el soporte con la plataforma IBM Cloud, pero no serían necesarios en el desarrollo final.

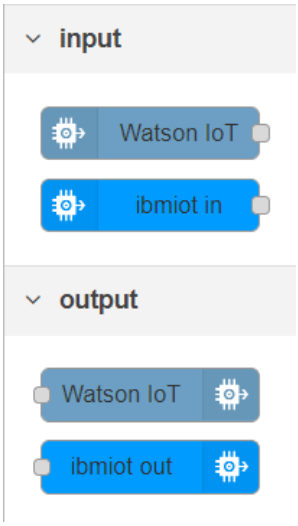

Figura 4.35 Paquete de nodos asociados a la plataforma IoT Watson de IBM Cloud.

3) **node-red-contrib-azure-iot-hub** (en este caso, v0.4.0) (Figura 4.36): Este paquete permitirá la inclusión de los nodos con los que se podrá comunicar e intercambiar información entre dispositivo y plataforma de comunicación Cloud IoT Hub de Azure.

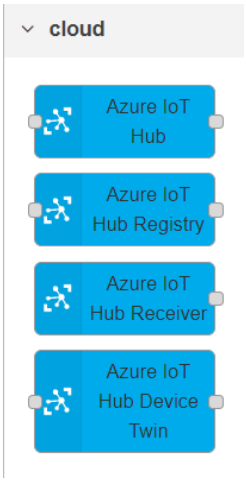

Figura 4.36 Nodos asociados a la plataforma de comunicación IoT Hub de Azure.

4) **node-red-dashboard** (en este caso, v3.5.0) (Figura 4.37): Este paquete permitirá la inclusión de nodos con los que se podrá hacer la representación gráfica de las variables en el dashboard que se accederá mediante 192.168.0.16:1880/ui.

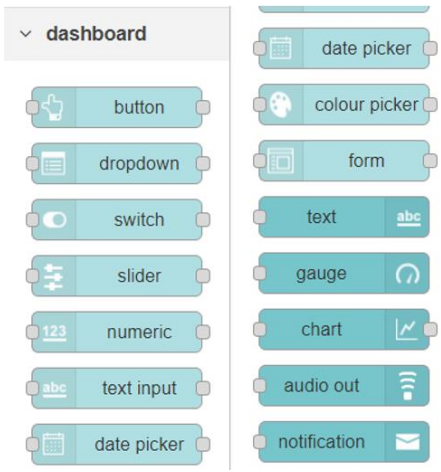

Figura 4.37 Paquete de nodos asociados al Dashboard.

Para adicionar estos paquetes de nodos citados bastará con acceder a “Manage palette” y ahí irse a ‘install’ siguiendo:  -> Manage palette -> install como se puede apreciar en la Figura 4.38.

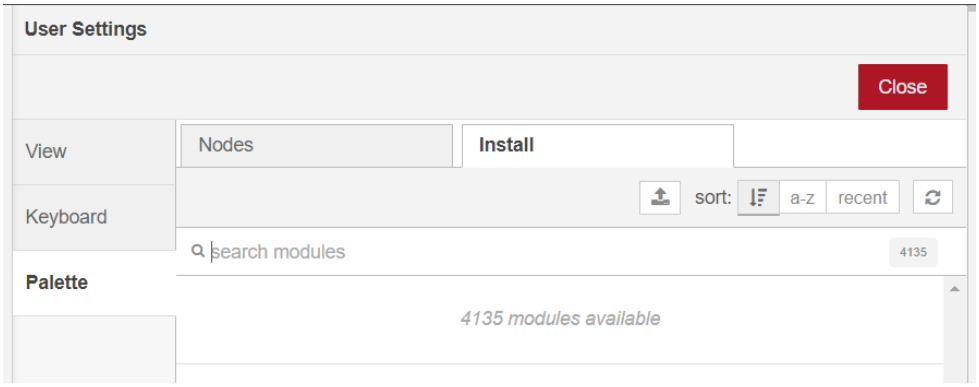

Figura 4.38 Sección de Node-RED destinada a la instalación de paquetes de nodos.

Con la inclusión de estos paquetes de nodos en el entorno Node-RED instalado en el controlador PLCnext del usuario, ya estará todo preparado en Node-RED para comenzar con la programación asociada a este entorno.

4.5.- CONFIGURACIÓN DE PLATAFORMAS CLOUD

Tras haber visto las configuraciones necesarias para Codesys, PLCnext Engineer y Node-RED, la última configuración necesaria que faltaría por revisar es la asociada al entorno Cloud empleado el cual en este caso han sido IBM Cloud en un inicio y posteriormente Microsoft Azure.

Obviamente antes de realizar cualquier acción por parte del usuario, lo primero será crear una cuenta en ambas plataformas para tener acceso a las herramientas y disponibilidades dadas por este entorno Cloud.

4.5.1- Configuración de IBM Cloud

Tras la creación del usuario, lo primero que habrá que realizar es la configuración en la plataforma Cloud fuera de la herramienta IoT Watson ya previamente introducida. Una vez que el usuario acceda con su cuenta a la plataforma cloud de IBM se podrá observar una pantalla como la de la Figura 4.39.

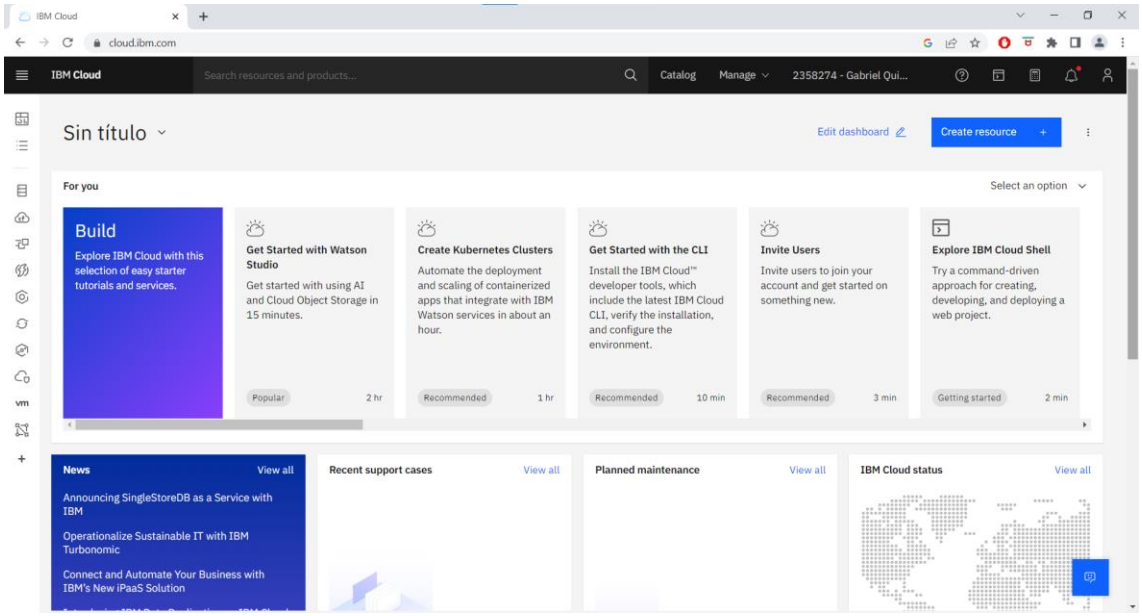

Figura 4.39 Página principal de la plataforma Cloud de IBM.

Ahora se podrá acceder a la pantalla de gestión y adición de recursos. Para ello, el usuario deberá hacer ‘click’ sobre el icono situado en la esquina superior izquierda y posteriormente dar ‘click’ en ‘Resource list’ como se puede ver en la Figura 4.40.

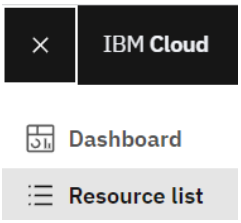

Figura 4.40 Acceso a la lista de recursos en la plataforma Cloud de IBM.

Una vez seguido este procedimiento se presentaría la siguiente pantalla visionada en la Figura 4.41. En esta pantalla se podrán crear los recursos necesarios para la aplicación del usuario. En este caso habrá que generar recursos asociados a 'Internet of Things' (Internet de las cosas) para la cual se empleará la herramienta IoT Watson y el Node-RED instalado en la plataforma Cloud de una manera semejante a la empleada en la instalación en el controlador por medio de contenedores.

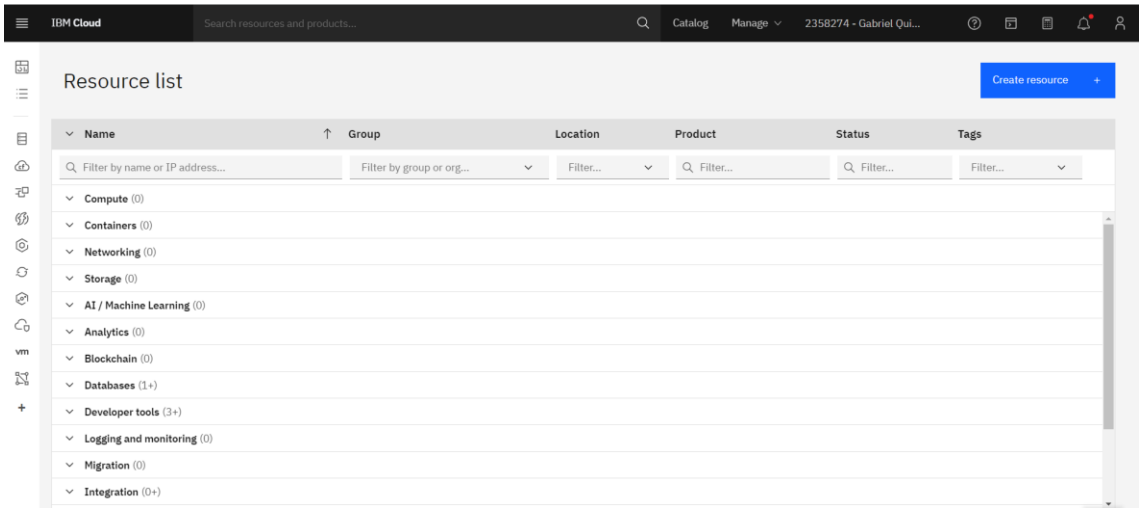

Figura 4.41 Página mostrada tras acceder al listado de recursos en la plataforma Cloud de IBM.

Para realizar la creación de los recursos necesarios para cada una de las dos acciones mencionadas habrá que hacer 'click' sobre 'Create resource' como se puede ver señalado en la Figura 4.42.

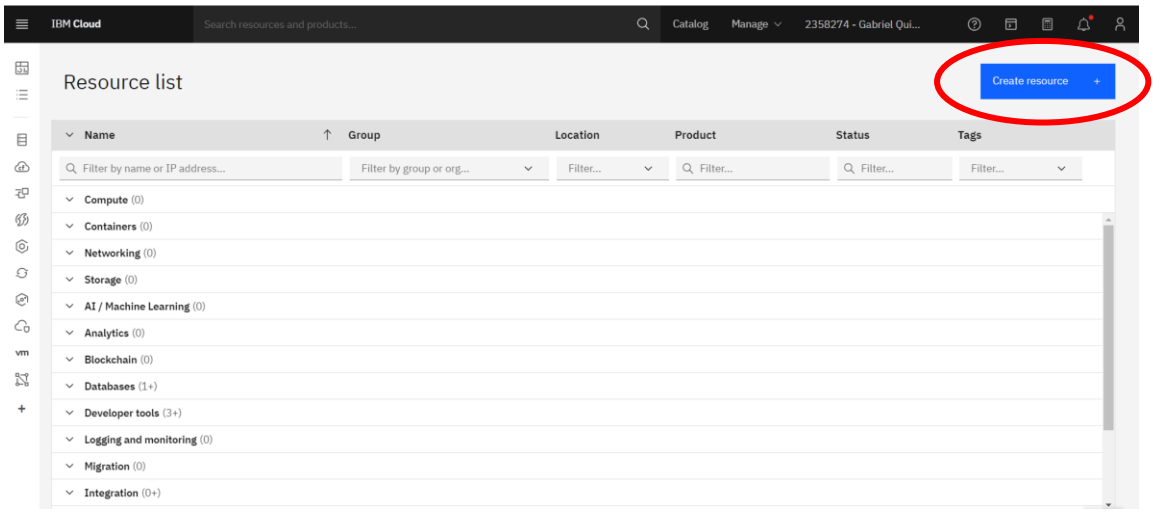

Figura 4.42 Icono de creación de un nuevo recurso en la plataforma Cloud de IBM.

Una vez que se pulse sobre este icono aparecerá una pantalla como la observada en la Figura 4.43 en la que se podrán buscar los recursos necesarios.

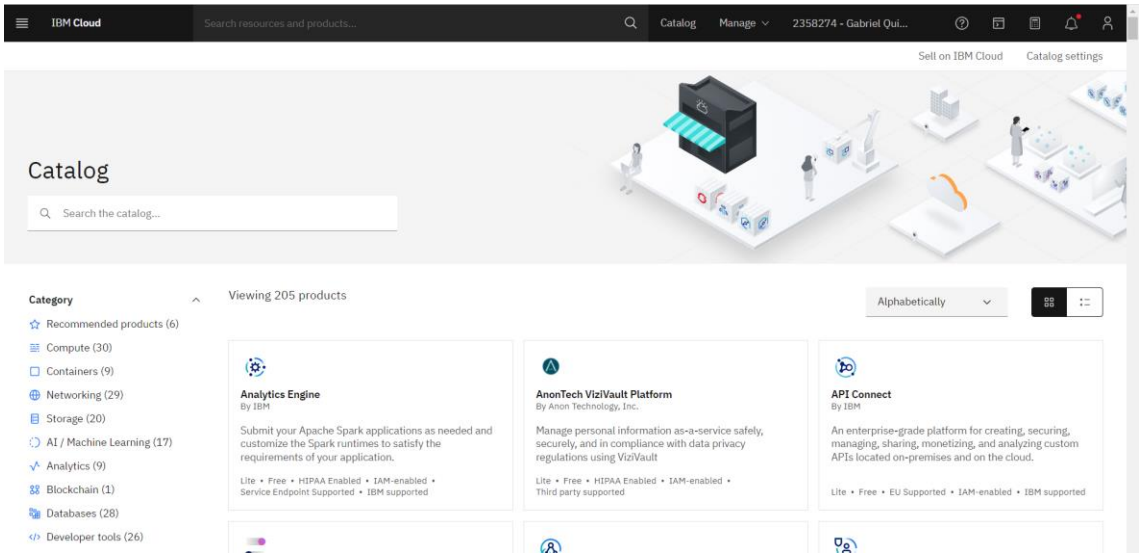

Figura 4.43 Pestaña de creación de recursos en la plataforma Cloud de IBM.

4.5.2- Configuración de IoT Watson

Una vez que ya se ha indicado la pestaña de creación de recursos se comenzará visionando la creación del recurso necesario para emplear la plataforma de comunicación

con el dispositivo, IoT Watson para cubrir el aspecto ‘Internet of Things’ ya previamente mencionado. Para ello habrá que buscar el recurso ‘Internet of Things Platform’ como se puede observar en la Figura 4.44.

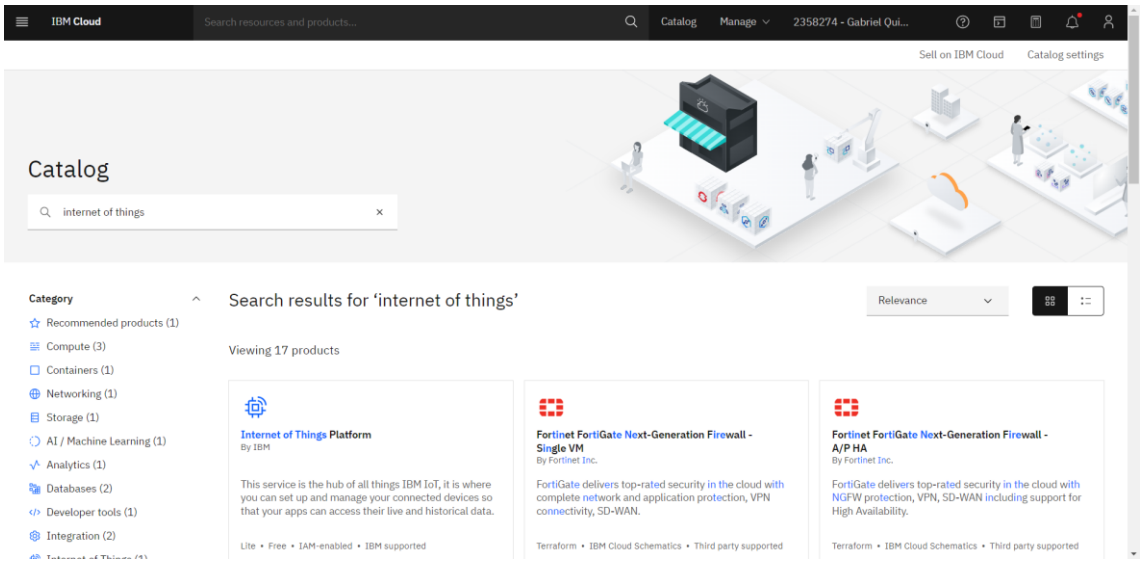

Figura 4.44 Búsqueda del recurso asociado a la plataforma de comunicación IoT Watson.

Una vez que el usuario acceda a este recurso, podrá presenciar una pantalla semejante a la siguiente visualizada en la Figura 4.45 en la que podrá crear el recurso que le dará acceso a la herramienta IoT Watson en la que generará el proyecto.

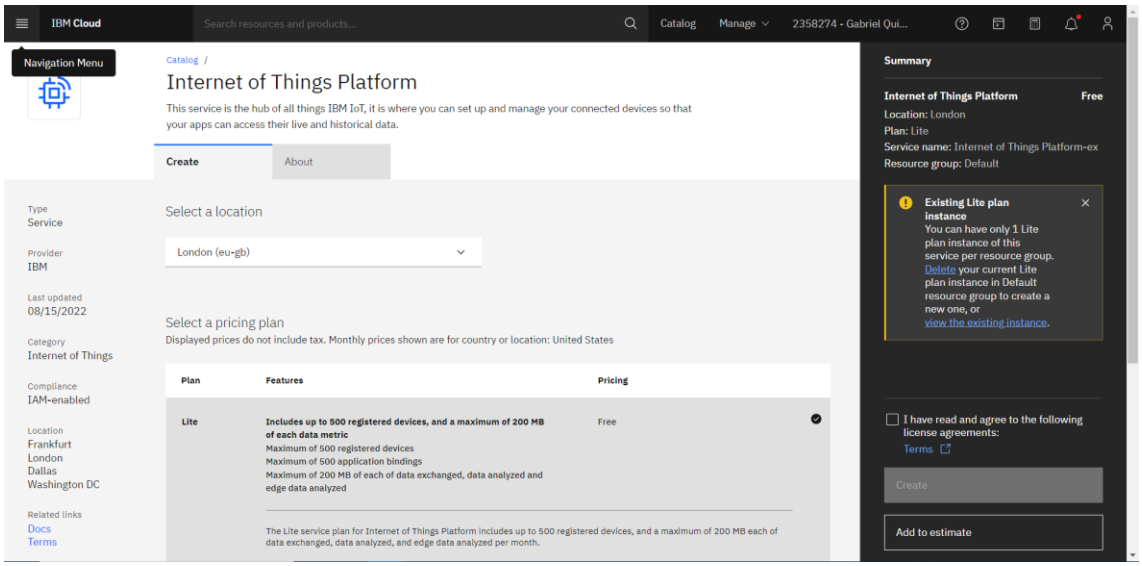

Figura 4.45 Recurso de la plataforma de comunicación IoT Watson.

Como se puede ver aquí, indica que ya existe un recurso de 'Internet of Things Platform' existente impidiendo que se pueda generar uno más debido a que el usuario está sujeto al plan gratuito ('Lite'), por el contrario, si no existiese ese recurso permitiría indicar la lectura de condiciones y dar a 'Create'. Una vez generado este recurso, debería aparecer el recurso en la lista de recursos existentes ya previamente introducida como se muestra en la Figura 4.46.

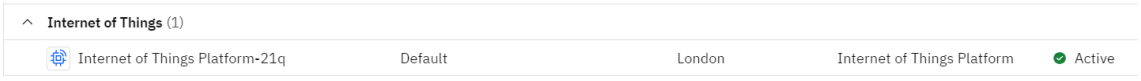

Figura 4.46 Recurso de la plataforma de comunicación IoT Watson ya generado.

Si se pulsa sobre este recurso permitirá acceder a la herramienta IoT Watson como se puede apreciar en la Figura 4.47.

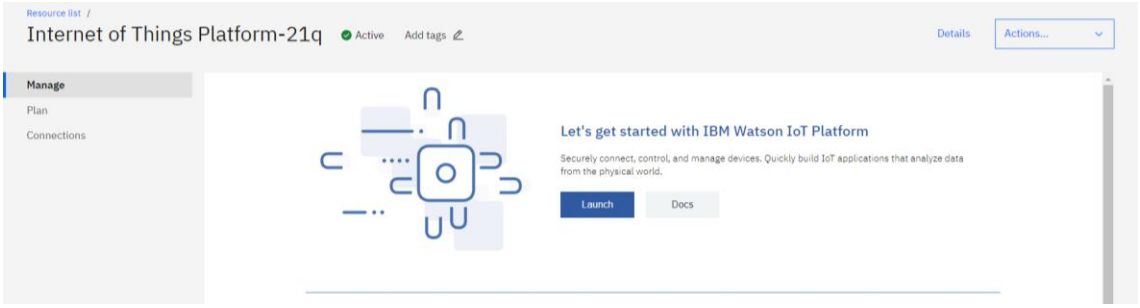

Figura 4.47 Recurso de la herramienta de IoT Watson listo para ser lanzado.

Dentro de este proceso ya se podrán realizar las tareas requeridas en la herramienta introducida. Esto se verá con más detalle en los apartados de programación y manual de usuario.

4.5.3- Configuración Node-RED en la nube IBM Cloud

Para finalizar las configuraciones necesarias en la nube para la implementación por IBM, el último paso sería la habilitación de Node-RED en la plataforma Cloud.

El servicio de Kubernetes dejará implementar un contenedor con una IP asociada en el cual se podrá contener el entorno de programación Node-RED de manera gratuita. Tras la creación del contenedor con el entorno de programación Node-RED asociado aparecerá en la pestaña de recursos la app asociada con el nombre que se haya dispuesto como se muestra en la Figura 4.48.

Figura 4.48 Contenedor para lanzamiento de Node-RED en la plataforma Cloud de IBM.

Si se accediese a la aplicación se vería una pestaña como la que se dispone en la siguiente Figura 4.49 en la cual se tiene la IP de acceso al contenedor donde se encontrará localizado el Node-RED.

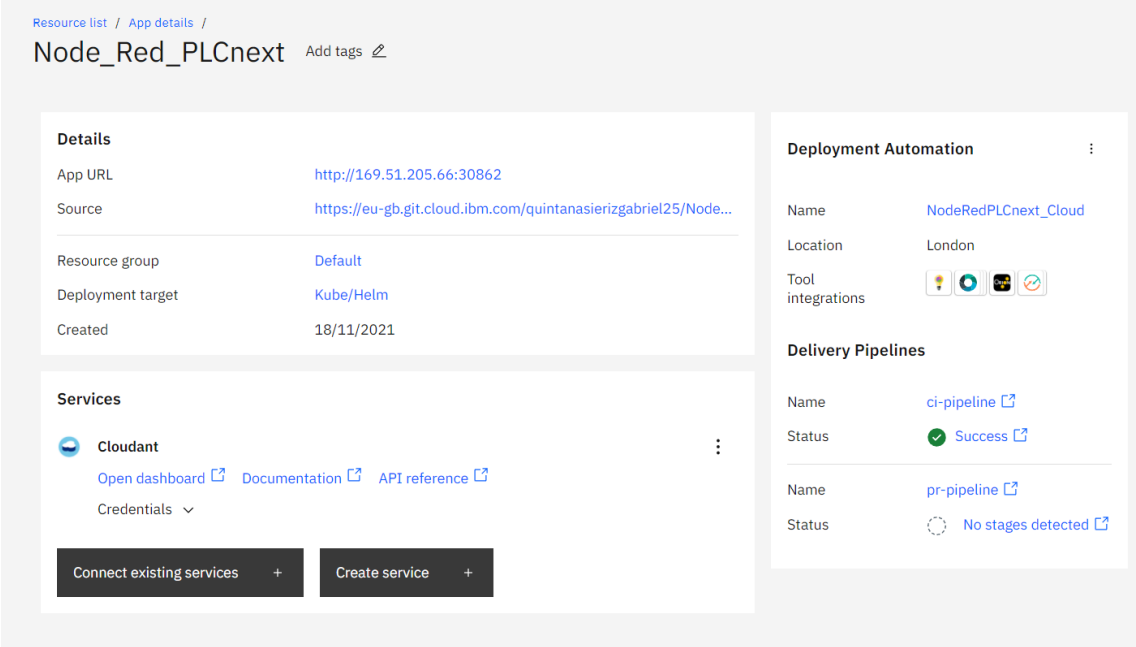

Figura 4.49 Acceso al contenedor que albergará el servicio de Node-RED.

Creado el contendor en el cual se encuentra localizado Node-RED el siguiente paso sería ver los nodos que es necesario instalar. Aquí se deberían instalar casi los mismos nodos que el Node-RED instalado en el controlador con la excepción de que el paquete de nodos asociado a OPC UA no sería necesario. Los paquetes de nodos necesarios serían los siguientes:

1) **node-red-contrib-ibm-watson-iot** (v0.2.8): Es necesario recordar que este paquete permitirá el uso de nodos que el usuario empleará para enviar y recibir datos de la herramienta de IoT Watson de IBM.
2) **node-red-dashboard** (en este caso, v3.1.1): Este paquete permitirá la inclusión de nodos con los que se podrá hacer la representación gráfica de las variables en el dashboard que se accederá mediante la dirección asociada al contenedor creado seguido de "/ui".

Con esto estaría ya preparado y descrito todo lo relacionado a las configuraciones necesarias para poder comenzar con el desarrollo y generación del programa asociado a este proyecto.

Por otro lado, como se comentó previamente, la herramienta IoT Watson deja de tener soporte y Node-RED deja de estar configurable de manera directa en la plataforma de IBM Cloud, es por eso que ahora se proceden a describir estos mismos procesos, pero para la plataforma de Microsoft Azure.

4.5.4- Configuración de Microsoft Azure

Microsoft Azure ha sido la plataforma Cloud finalmente seleccionada y con ella se podrá además comprobar que hay muchas similitudes en la adaptación de todo lo inicialmente implementado en IBM Cloud.

Lo primero, es disponer de una cuenta asociada a Microsoft para poder acceder al servicio de Azure proporcionado por este. Tras crear la cuenta se mostrará la pantalla de la siguiente Figura 4.50, pero sin los recursos que se visualizan.

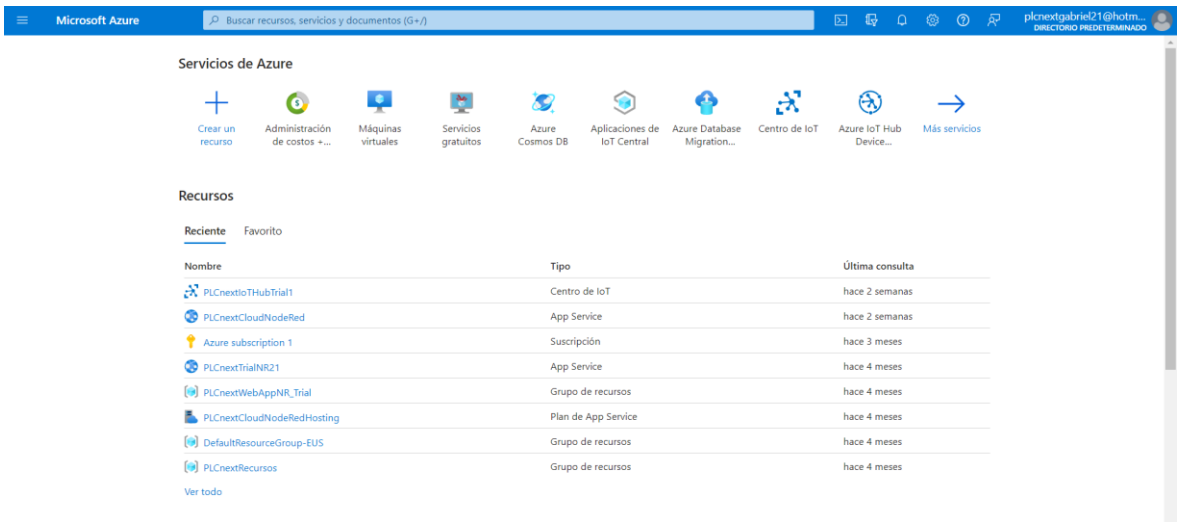

Figura 4.50 Pantalla principal realizado el acceso a la plataforma Cloud de Azure.

En la pantalla de recursos, hay que buscar los elementos necesarios para desarrollar el proyecto. Primero hay que seleccionar la plataforma de comunicación ya mencionada, IoT Hub, dependiendo si la plataforma Cloud está puesta en castellano o en inglés habrá que buscar Centro de IoT o IoT Hub respectivamente. Esta herramienta es la que permitirá la comunicación entre la nube y el dispositivo PLCnext y se asemeja a la ya introducida IoT Watson de IBM. En este caso como se dispone de la plataforma Cloud en castellano al buscar el recurso aparecerá Centro de IoT como se visualiza en la Figura 4.51 y se debe de dar a Crear para emplear este recurso.

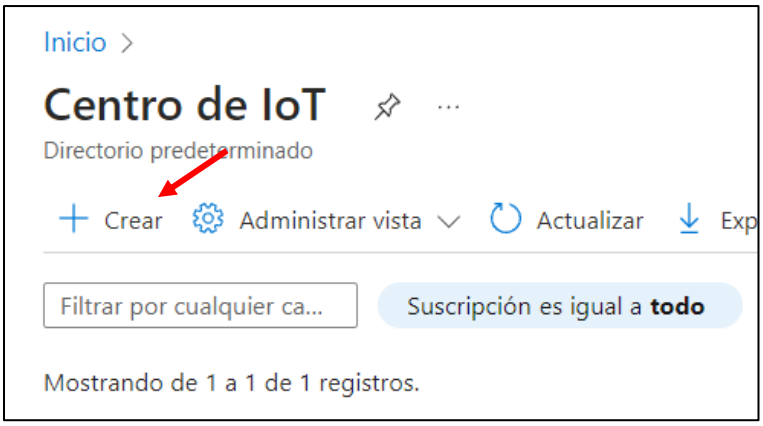

Figura 4.51 Creación de un nuevo recurso en la plataforma de Cloud de Azure.

Aparecerá entonces una pantalla como la visualizada en la Figura 4.52, se puede observar que se dispone de tres dispositivos. En este caso es porque se realiza conexión con el controlador PLCnext, un controlador externo de prueba para hacer comparativa que emplea ayuda de una Raspberry Pi para generar la comunicación con la nube y la conexión con el Node-RED hospedado en la propia plataforma Cloud.

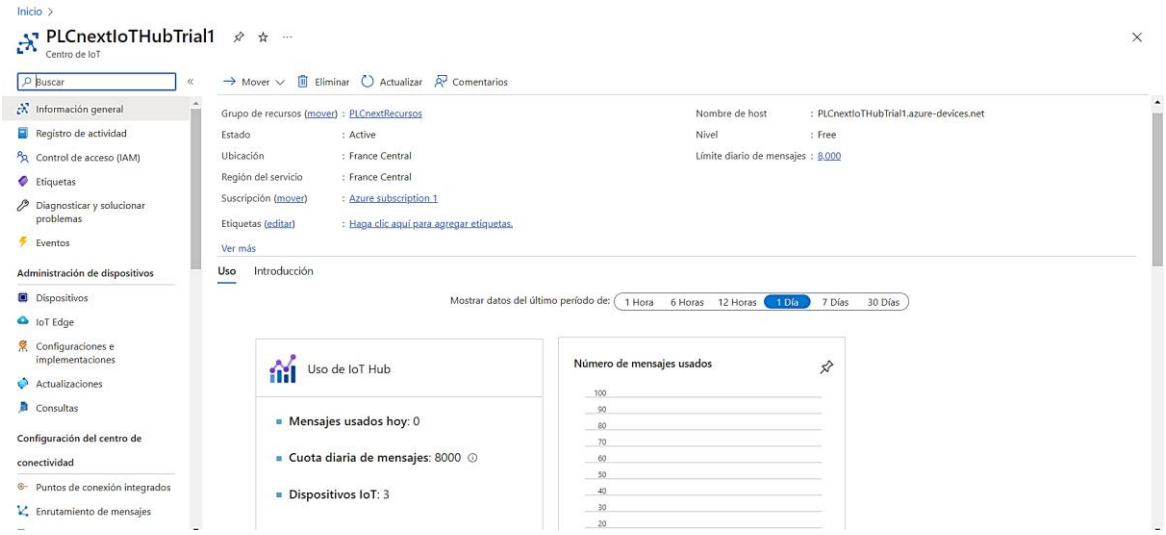

Figura 4.52 IoT Hub asociado al desarrollo del trabajo.

El siguiente paso es preparar el hospedaje para el entorno Node-RED en la plataforma Cloud. Para ello lo que se recurre es al recurso de Aplicación y sería conveniente crearlo a partir del recurso proporcionado por *jmservera* de *node-red-azure-webapp* [35], si se desplegase directamente, simplemente lo que habría que hacer es enlazar la cuenta de Microsoft Azure en la zona señalada en la Figura 4.53.

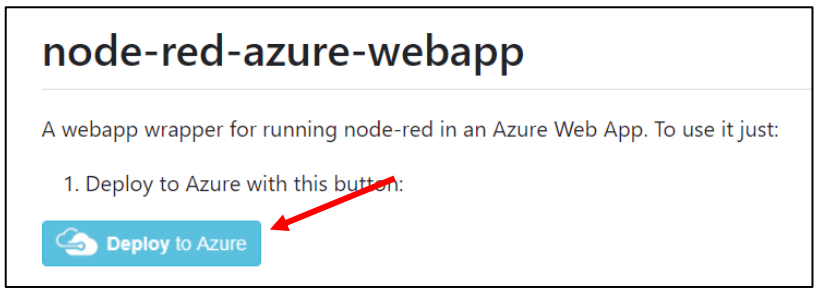

Figura 4.53 Lanzamiento de aplicación web con Node-RED desde github.

De lo contrario, lo que habrá que hacer es desplegar el recurso en la propia plataforma Cloud a través del recurso de Aplicación Web (App Services en la plataforma) visualizado en la Figura 4.54, y mientras se realiza la creación de la Aplicación Web indicar que se va a realizar el lanzamiento de la aplicación web a partir de un repositorio de origen externo

asociado al enlace: *https://github.com/jmservera/node-red-azure-webapp.git* a la plataforma Cloud como se indica en la Figura 4.55.

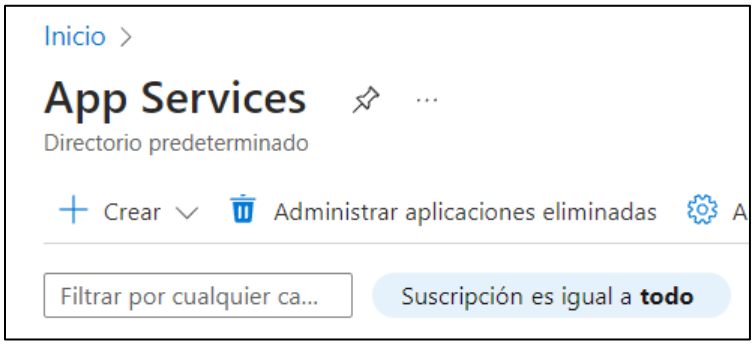

Figura 4.54 Creación de recurso de aplicación web en la plataforma Cloud de Azure.

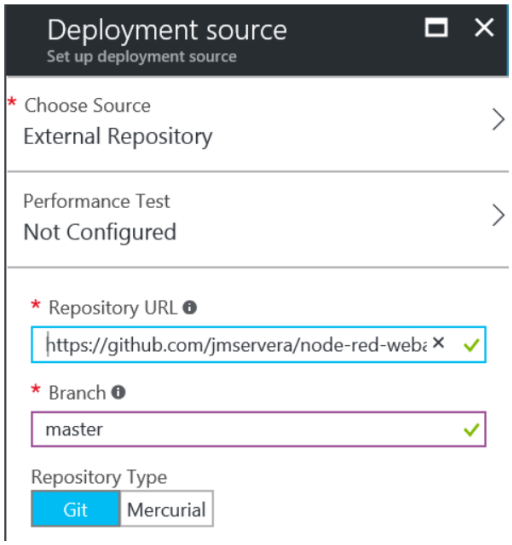

Figura 4.55 Asociación de repositorio de github a la aplicación web.

Remarcar que normalmente, el siguiente paso antes de programar sería incluir los nodos necesarios para proceder con la programación. En este caso se produce una particularidad frente el caso que se produce normalmente y es que al usar una preinstalación de Node-RED en una aplicación web de Azure, esta ya viene con los nodos que se requieren, pero con una excepción, no se dispone del nodo de enlace con la plataforma de comunicación IoT Hub de Azure, el cual casi es el más importante.

Para poder disponer de él en el Node-RED asociado a la plataforma Cloud, habrá que realizar la instalación de manera manual. Para ello habrá que acceder a la carpeta raíz de la aplicación web creada en Azure.

La carpeta raíz se accede de manera directa añadiendo *.scm* previamente a la extensión .azurewebsites.net, quedando por ejemplo en el caso particular del proyectante: https://plcnextcloudnodered.scm.azurewebsites.net. Una vez dentro, se podrá observar una pantalla semejante a la vista en la Figura 4.56.

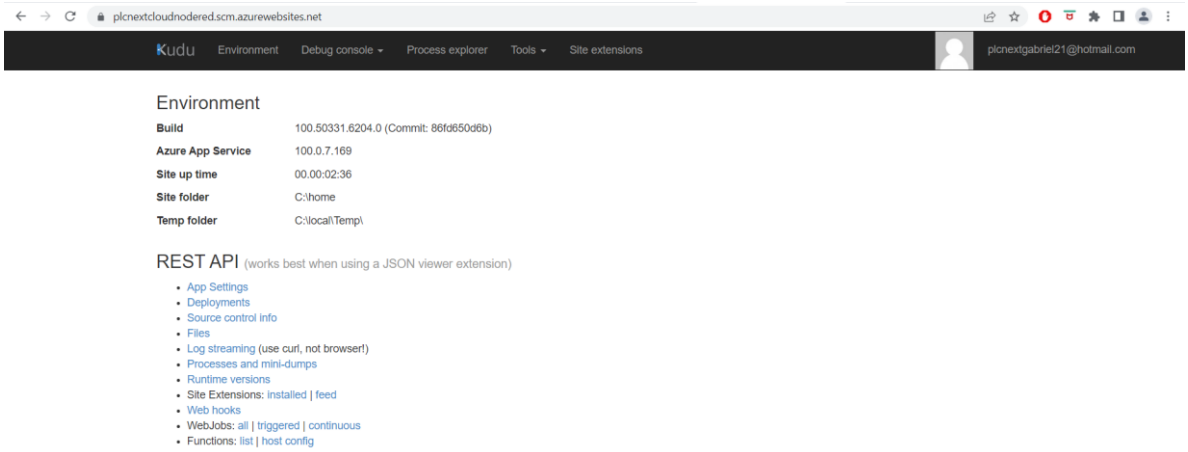

Figura 4.56 Acceso a la raíz de la aplicación web desplegada.

Una vez dentro, se podrá acceder a una ventana de consola vista en la Figura 4.57, a través de: Debug console/CMD que ya permitirá al usuario navegar libremente entre las diferentes carpetas de manera muy similar a lo ya descrito en la instalación de Node-RED en el controlador PLCnext.

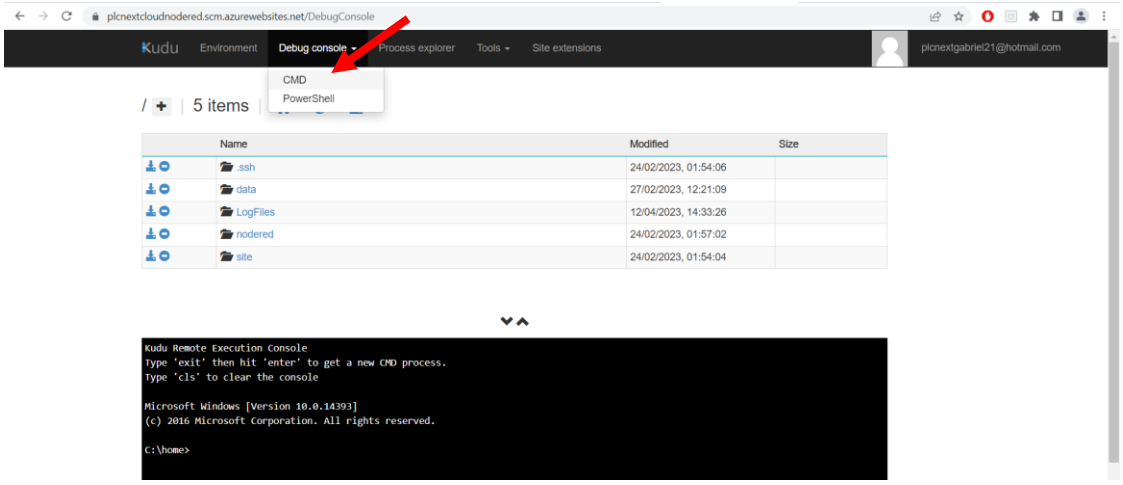

Figura 4.57 Acceso a la terminal de la aplicación web desplegada.

A través de la consola y como se venía diciendo, ya se podrá proceder de manera muy semejante a la manera de proceder que se ha aplicado para el Node-RED asociado al controlador PLCnext.

El nodo esencial a añadir, del cual no dispone de manera directa el repositorio de Node-RED incorporado en la aplicación web es el Azure IoT Hub mostrado en la Figura 4.58. Está contenido dentro del paquete de nodos: **node-red-contrib-azure-iot-hub**, que se deberá instalar de nuevo completamente a través de la mencionada ventana de comandos de la aplicación web ya previamente introducida.

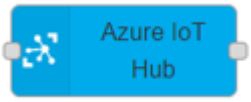

Figura 4.58 Nodo de conexión con IoT Hub de Azure.

Finalmente, el último paso de configuración a realizar, para dejar el Node-RED de la aplicación web totalmente preparado para su operación, sería el de modificar el ya comentado archivo *settings.js* añadiendo la identificación por medio de usuario y contraseña de manera semejante a la realizada para el controlador PLCnext.

4.5.5- Configuración de Proficloud.io

Para finalizar este apartado de configuración faltaría comentar muy brevemente (puesto que la configuración de Proficloud.io es casi directa) la pequeña preparación que hay que realizar en esta plataforma Cloud propia de las tecnologías PLCnext.

La preparación simplemente consiste en que una vez que se acceda a la plataforma, el añadir el dispositivo propio del usuario a esta mediante el botón de "ADD DEVICE" (añadir dispositivo) señalado en la Figura 4.59.

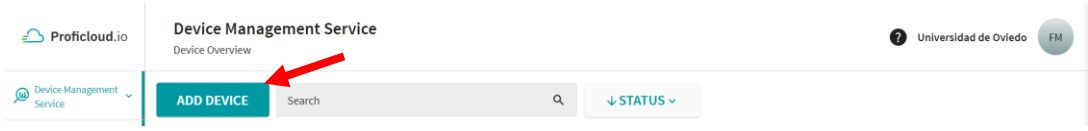

Figura 4.59 Icono de añadido de dispositivo a la plataforma Cloud Proficloud.io.

5. Programación global del sistema

En este apartado se desarrollará la explicación de toda la programación implementada para el proyecto en los diferentes lenguajes recogidos dentro de la normativa IEC 61131-3 (ST, FBD, SFC) y el asociado al entorno de programación gráfico Node-RED.

Se presentará, de manera ordenada, el desarrollo de programa implementado respetando la temporalidad del proyecto, por lo que se empezará por el desarrollo de programa en Codesys, seguido de la adaptación de código al entorno de programación PLCnext Engineer para posterior conexión, codificación y desarrollo de pruebas controlador físico. A continuación, irá seguido de la programación implementada en Node-RED, primero la generada para el Node-RED localizado en el controlador PLC y posteriormente en el entorno Cloud.

Antes de comenzar con la programación en cada una de las aplicaciones, se presenta en la siguiente Figura 5.1 una estructura general del plan de desarrollo de la programación y supervisión global del sistema automatizado, con las fases y elementos empleados en cada una de ellas.

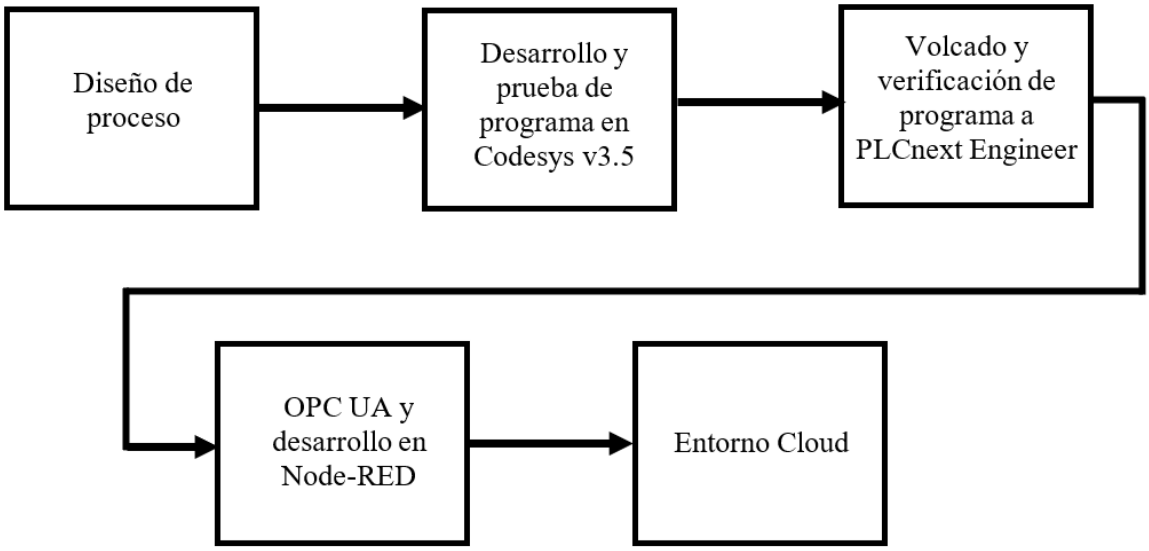

Figura 5.1 Desarrollo general esquematizado de proyecto.

5.1.- PROGRAMACIÓN EN CODESYS V3.5

Inicialmente, tras la especificación de las características de la planta objeto de este trabajo y funcionalidades a implantar, se desarrolla el programa de control con la herramienta Codesys 3.5 antes de realizar el desarrollo para el controlador final seleccionado de PLCnext mediante el entorno de programación PLCnext Engineer.

El programa desarrollado en Codesys, abarca un control completo de la planta siguiendo las normativas y guías ya previamente comentadas de ISA88, GEMMA, ISA 101… y permitiendo al usuario realizar una simulación de proceso de planta y observar el funcionamiento sin necesidad de requerir físicamente, ni de planta, ni de controlador.

Aparte de incorporar un control completo de planta, se ha generado un conjunto de pantallas (HMI) para que el operario pueda operar de manera directa con todos los recursos, elementos y modos de funcionamiento que la planta puede realizar.

5.1.1- Programa de control en Codesys v3.5

Comenzando con la fase de programa asociada al control propio de la planta introducida, antes de entrar con el código propio asociado, sería interesante visualizar el árbol de proyecto asociado a cada elemento de programa.

El árbol general de proyecto visualizado se visualiza en la Figura 5.2, donde se muestran de forma resumida todas las tareas y elementos asociados al programa que constituye todo el proyecto en Codesys, pudiendo distinguir:

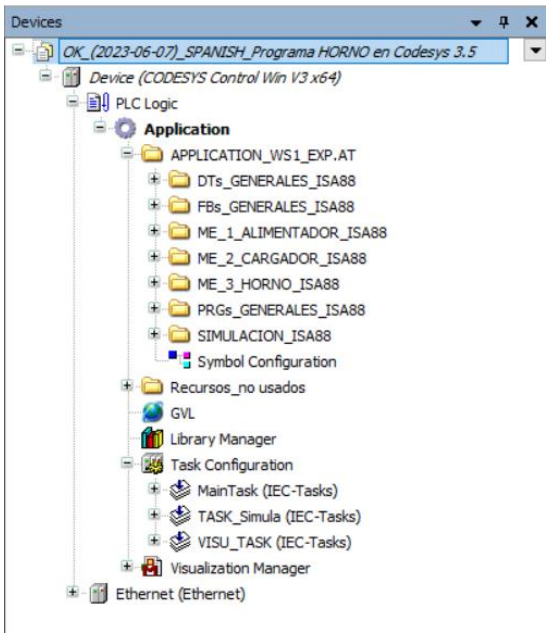

Figura 5.2 Árbol de proyecto en Codesys.

1) DTs_GENERALES_ISA88 y FBs_GENERALES_ISA88: Las estructuras de datos y bloques de función asociados al proceso general como puede ser el GDMMA, estados y el pedido que realiza el usuario.

2) ME_1_ALIMENTADOR_ISA88: Módulo de equipamiento del alimentador, el cual al igual que en el caso general poseerá su propia estructura de datos y bloques de función asociados.

3) ME_2_CARGADOR_ISA88: Módulo de equipamiento del cargador (semejante estructura que el Alimentador).

4) ME_3_HORNO_ISA88 Módulo de equipamiento del horno (semejante estructura que el alimentador y el Cargador)

5) PRGs_GENERALES_ISA88: Son un conjunto de POUs generales como funciones de escalado y programas de asignación de entradas, salidas y el programa MAIN, así como las pantallas de explotación y control de la planta.

6) SIMULACION_ISA88: Donde se localizan los módulos de programa que simulan diferentes funcionalidades de la planta física y un bloque funcional que emula el comportamiento de un sistema de primer orden.

A continuación, se procede a describir los diferentes grupos de elementos que componen el programa.

5.1.1.1 Estructuras de datos generales

Se han creado varios tipos de datos estructurados, bajo la carpeta DTs_GENERALES_ISA88, que contempla los elementos que se muestran en la Figura 5.3.

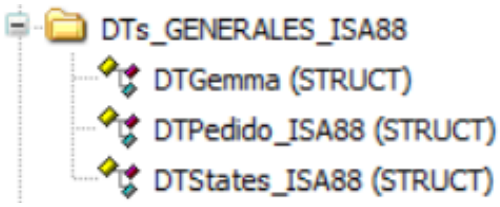

Figura 5.3 Estructuras de datos general en el programa de Codesys.

Como se puede observar se encuentran las siguientes estructuras de datos:

1) **DTGemma**: Se declaran todas las variables asociadas a los estados del GDMMA y sus transiciones, una variable de tipo STRING para el nombre de la etapa, el Reset general, la gestión de los modos de funcionamiento tanto en modo automático como en modo manual. Los diferentes grupos de variables mencionados se pueden observar en la Figura 5.4.

```
// States GEMMA
A1_IniCI, A2_PFinC, A5_PAtF, A6_IniPO: BOOL;
New_Cycle, F1_PROD, F2_PREP, F3_FIN, F4_VDes, F5_VOrd, F6_TEST: BOOL;

// Transitions GEMMA
Fin_A1, A_F2, A_F4, A_F6, Fin_A2, Fin_A5, Fin_A6: BOOL;
Fin_F1, A_A2, A_F3, A_F5: BOOL;
Fin_F2, Fin_F3, Fin_F4, Fin_F5, Fin_F6: BOOL;

// Nombre de la etapa
Step: STRING (10);

// Reset general
Reset: BOOL;

// Gestion de los modos de producción (Auto)
Start_PROD: BOOL;
With_F2: BOOL;
With_F3: BOOL;
End_Cycle: BOOL;
Instant_End: BOOL;
Cycles_1_N: BOOL;
Mode_AUTO: BOOL;  // 0 NORMAL, 1 Verificacicón orden
Tip_AUTO: BOOL;   // Transiciona con cada pulsación si Tip_MODE=1

// Gestión de los modos de producción (Manual)
Mode_MAN_TEST: BOOL;
Start_MAN_TEST: BOOL;
Stop_MAN_TEST: BOOL;
```

Figura 5.4 Código asociado a la estructura de datos DTGemma.

2) **DTPedido_ISA88**: Esta estructura de datos contiene todas las variables asociadas al pedido: su identificación (ID), inicio y estado del pedido y valores de especificación del pedido (número de piezas, peso, temperatura y tiempo del control de temperatura), tal y como se puede observar en la Figura 5.5.

```
ID: DINT := 1;              // Identificación de pedido (9 números, AAMMDDOO0, añomesdiaorden)
Inicia: BOOL;
Estado: INT;                // Código de estado: 0: Parado, 1: En producción, 2: En pausa, 3: Cancelado
SP_NPiezas: INT := 3;       // Bien por el número de piezas a cargar desde el alimentador
SP_Peso: REAL := 100.0 ;    // o por el peso de la carga (en Kg)
SP_Temp: REAL := 90.0;      // Temperatura de control en el horno a soportar por la carga (en °C)
TCT: DINT := 30;            // Tiempo de control (PID) de temperatura (en segundos) - 3 Minutos
```

Figura 5.5 Código asociado a la estructura de datos DTPedido_ISA88.

3) **DTStates_ISA88**: Esta estructura de datos que contiene todas las variables asociadas a los estados de ISA88 y sus transiciones, mensajes que indican el estado y las variables de aviso, defecto, alarma y emergencia que pueden ser

asociadas a la pantalla de explotación. Todos estos detalles se pueden encontrar en la Figura 5.6.

```
// Estados ISA88
Init : BOOL := TRUE;
Running, COMPLETED: BOOL;       // Init == Idle (Initial State)
Holding, Held, Restarting: BOOL;
Pausing, Paused: BOOL;
Stopping, STOPPED: BOOL;
Aborting, ABORTED: BOOL;

// Según Transiciones ISA-88
Start, Hold, Restart, Pause, Resume: BOOL;
Stop, Abort, Reset: BOOL;
End_Running, End_Holding, End_Restarting,
End_Pausing, End_Stopping, End_Aborting: BOOL;

// Mensaje de estado
msg_STATE: STRING ; // Texto identificador de estado

// Detalles HMI: Aviso, Defecto, Alarma, Emergencia
Aviso: BOOL;
LAviso: BOOL;
msgAviso: STRING := 'Mensaje de AVISO';

Defecto: BOOL;
LDefecto: BOOL;
msgDefecto: STRING := 'Mensaje de DEFECTO';

Alarma: BOOL;
LAlarm: BOOL;
msgAlarma: STRING := 'Mensaje de ALARMA';

Emer: BOOL;
LEmer: BOOL;
msgEmer: STRING := 'Mensaje de EMERGENCIA';
```

Figura 5.6 Código asociado a la estructura de datos referenciada bajo el nombre DTStates_ISA88.

5.1.1.2 Variables globales

Siguiendo con los elementos del programa de control implementado en Codesys, es preciso detallar el conjunto de variables globales declaradas y que pueden ser accesibles en toda la configuración del proyecto. Se encuentran bajo la denominación (carpeta) identificada por GVL (iniciales de "*Global Variables List*). De forma más pormenorizada se organizan según:

1) Variables generales asociadas a los elementos definidos de ISA88 y GDMMA, en concreto a las estructuras de datos correspondientes y las cuales se pueden observar en la Figura 5.7. Las variables auxiliares *Azul_Reset_x* se utilizan para gestionar algunos aspectos del interface de usuario.

```
Pedido_A: DTPedido_ISA88;

Alimentador_A: DTAlimentador_ISA88;
Azul_Reset_ALIM_A: BOOL;

Cargador_A: DTCargador_ISA88;
Azul_Reset_CARGA_A: BOOL;

Horno_A: DTHorno_ISA88;
Azul_Reset_HORNO_A: BOOL;

Gemma: DTGemma;
```

Figura 5.7 Variables globales de ISA88 y el GDMMA desarrollado.

2) Variables de entrada y salida de PLC asociadas al alimentador, cargador y el horno, y que se enumeran en la Figura 5.8.

```
(* Entradas / Salidas PLC del Alimentador_1 *)
// Las entradas tienen comentada la dirección para más fácil depuración
I_xA0  (*AT %IX0.0*) : BOOL;
I_xA1  (*AT %IX0.0*) : BOOL;
I_xV0  (*AT %IX0.0*) : BOOL;

Q_xAmas  AT %QX0.0 : BOOL;

(* Entradas / Salidas PLC del Cargador_1 *)
I_xFC0  (*AT %IX1.0*) : BOOL;
I_xFC1  (*AT %IX1.0*) : BOOL;

Q_xKM0  AT %QX1.0 : BOOL;
Q_xKM1  AT %QX1.1 : BOOL;

I_iChPeso (*AT %IW10*) : INT;

(* Entradas / Salidas PLC del Horno_1 *)
I_xHC  (*AT %IX2.0*) : BOOL;
I_xCG  (*AT %IX2.1*) : BOOL;
I_xKCA  (*AT %IX2.2*) : BOOL;
I_xKEA  (*AT %IX2.3*) : BOOL;
I_xTR  (*AT %IX2.4*) : BOOL;

Q_xKC  AT %QX2.0 : BOOL;
Q_xKE  AT %QX2.1 : BOOL;

I_iChTemp (*AT %IW12*) : INT;
```

Figura 5.8 Variables de entrada y salida direccionadas al PLC para Alimentador, Cargador y Horno.

3) Variables generales requeridas para la programación de modos de funcionamiento, detección de errores, mensajes, etc … de los módulos de equipamiento definidos (Alimentador, Cargador y Horno en este caso). Para más detalle consultar el código fuente de la aplicación Codesys implementada y que se incluye como Anexo de este trabajo.

5.1.1.3 Bloques funcionales generales

Estos bloques funcionales se encuentran en el árbol del proyecto bajo la carpeta de FBs_GENERALES_ISA88 descomponiéndose tal y como se ven en la Figura 5.9.

Figura 5.9 Bloques funcionales de carácter general.

- **FB_Actuator**

Este bloque funcional permite gestionar el funcionamiento general de un actuador para su activación o desactivación e incluso aplicar un valor de salida numérico. Será muy útil en accionamientos como electroválvulas, contactores, drivers de potencia, etc. Dispone también de variables de control (Iniciar, Finalizar), de habilitación (Enable) y permisivo (Permiso), así como salidas que permiten conocer el estado del accionamiento mediante mensajes y variable booleana (Done).

La declaración de las variables de este bloque funcional se muestra en la **Figura**:

Scope	Name	Address	Data type	Initialization	Comment
VAR_INPUT	**Enable**		BOOL		Permite esta funcionalidad
VAR_IN_OUT	**Actuador**		BOOL		
VAR_INPUT	**Permiso**		BOOL		Interlock
VAR_INPUT	**SP_Power**		REAL		
VAR_INPUT	**Iniciar**		BOOL		Condición para iniciar la activación de Amas
VAR_INPUT	**Finalizar**		BOOL		Condición para finalizar la activación tras el retardo, si se parametriza
VAR_INPUT	**TDone**		INT		Tiempo de permanencia de señal de Done (en segundos)
VAR_OUTPUT	**MSG**		STRING	' '	
VAR_OUTPUT	**Done**		BOOL		Confirmación de que la funcionalidad ha sido ejecutada
VAR_OUTPUT	**OUT_Power**		REAL		
VAR	**RS_Act**		RS		
VAR	**FT_Act**		F_TRIG		
VAR	**TP_Done_Act**		TP		

Figura 5.10 Listado de variables asociadas a FB_Actuator,

Este bloque de función se ha generado, por ejemplo, con la finalidad de realizar el control del actuador A+ en el alimentador o la gestión del motor del cargador a través de sus contactores (KM0 y KM1). El código en lenguaje de texto estructurado (ST) para este bloque funcional se muestra en la Figura 5.11.

```
IF Enable THEN

    MSG := '     ';
    RS_Act (SET:=Iniciar AND Permiso, RESET1:=Finalizar);

    Actuador := RS_Act.Q1 AND Permiso;

    IF Actuador THEN
        MSG := 'Act Activado...';
        OUT_Power := SP_Power;
    ELSE
        OUT_Power := 0.0;
    END_IF

    FT_Act (CLK:=Actuador);
    TP_Done_Act (IN := FT_Act.Q, PT:=TO_TIME(TDone));
    Done := TP_Done_Act.Q ;
    IF Done THEN
        MSG := 'Act Hecho... ';
    END_IF



END_IF;
```

Figura 5.11 Código en ST asociado a FB_Actuator.

- **FB_D123_DEFECTS_ST**

Este bloque de función ha sido creado para el apartado de defectos reflejados en el GDMMA con la letra D y son: D1: Parada de emergencia, D2: Diagnóstico y/o tratamiento de fallo/defecto y D3: Producción a pesar de fallo/defecto. Las variables asociadas a este bloque de función se pueden observar en la Figura 5.12.

Scope	Name	Address	Data type
VAR_INPUT	**Enable**		BOOL
VAR_INPUT	**Alim**		DTAlimentador_ISA88
VAR_INPUT	**Carga**		DTCargador_ISA88
VAR_INPUT	**Horno**		DTHorno_ISA88
VAR_OUTPUT	**OUT_Aviso**		BOOL
VAR_OUTPUT	**OUT_Defect**		BOOL
VAR_OUTPUT	**OUT_Alarm**		BOOL
VAR_OUTPUT	**OUT_Emer**		BOOL

Figura 5.12 Listado de variables asociado a FB_D123_DEFECTS_ST.

Se ha programado en lenguaje en ST y simplemente cuando se active la variable Enable se permitirá generar un valor general de aviso, defecto, alarma, emergencia correspondiente a los módulos de equipamiento definidos. El código desarrollado puede visualizarse en la Figura 5.13.

```
IF Enable THEN

    OUT_Aviso := Alim.STATES.Aviso OR
              Carga.STATES.Aviso OR
              Horno.STATES.Aviso ;

    OUT_Defect := Alim.STATES.Defecto OR
              Carga.STATES.Defecto OR
              Horno.STATES.Defecto ;

    OUT_Alarm := Alim.STATES.Alarma OR
              Carga.STATES.Alarma OR
              Horno.STATES.Alarma ;

    OUT_Emer := Alim.STATES.Emer OR
              Carga.STATES.Emer OR
              Horno.STATES.Emer ;

END_IF
```

Figura 5.13 Código en ST asociado a FB_D123_DEFECTS_ST.

- **FB_F1_PROD_GEMMA**

A continuación, irían dos de los bloques de función asociados a la parte del GDMMA de funcionamiento de planta. Serían el estado F1: Producción normal y el estado F4: Verificación en desorden. Comenzando por el bloque de función asociado a F1 y denominado con el nombre FB_F1_PROD_GEMMA se observan en la Figura 5.14 las siguientes variables empleadas para el desarrollo.

Scope	Name	Address	Data type
VAR_INPUT	**SFCTipMode**		BOOL
VAR_INPUT	**SFCTip**		BOOL
VAR	**Done_Sacar_Sp_Peso_en_Piezas**		BOOL
VAR_OUTPUT	**OUT_Fin_Auto**		BOOL
VAR_OUTPUT	**SFCCurrentStep**		STRING
VAR	**Done_Abrir_Horno**		BOOL
VAR	**Done_Sacar_Peso_en_Piezas**		BOOL
VAR	**Done_Cargar_Piezas**		BOOL
VAR	**Done_Calentar_Horno**		BOOL
VAR	**Done_Control_Temp_Horno**		BOOL
VAR	**Done_Enfriar_Horno**		BOOL
VAR	**Done_Descargar_Piezas**		BOOL
VAR	**FB_AUTO_Abrir_Horno**		FB_2_Descargar_Piezas_ISA88
VAR	**FB_AUTO_Sacar_SP_Peso_Alim**		FB_3_Sacar_SP_Peso_Alimentador_ISA88
VAR	**FB_AUTO_Cargar_Piezas**		FB_1_Cargar_Piezas_ISA88
VAR	**FB_AUTO_Calentar_Horno**		FB_1_Calentar_Horno_ISA88
VAR	**FB_AUTO_Control_Temp_Horno**		FB_3_Control_Temp_Horno_ISA88
VAR	**FB_AUTO_Enfriar_Horno**		FB_2_Enfriar_Horno_ISA88
VAR	**FB_AUTO_Descargar_Piezas**		FB_2_Descargar_Piezas_ISA88
VAR	**MSG_Sacar_SP_Peso_en_Piezas**		STRING
VAR	**F_TRIG_RUN**		f_trig
VAR	**Done_Sacar_SP_Peso**		BOOL
VAR_INPUT	**SFCReset**		BOOL

Figura 5.14 Listado de variables asociadas a FB_F1_PROD_GEMMA.

Como pequeño detalle a comentar, la variable SFCTipMode al ser de tipo BOOLEANA y encontrarse en TRUE, solo podrá realizar transición con la variable SFCTip, y todo ello por la forma de sincronización de SFCs que implementa Codesys 3.5.

El programa asociado a este bloque de función se ha desarrollado en SFC. Hay que recordar que este bloque de función es el encargado de desarrollar la función F1 del GDMMA y se corresponde con la producción normal de proceso (ver **Figura**). Esta secuencia de operaciones comienza con el inicio de proceso (Init) con la receta

proporcionada por el usuario; con esa receta, el sistema, a continuación, saca el número de piezas acorde al peso que solicitó el usuario (Sacar_SP_Peso); tras esto, las piezas son cargadas al horno (Cargar) y le sigue el proceso de calentamiento regulando a la temperatura y en el tiempo estipulados (Calentar y Contorl_Temp). Una vez completado el control de temperatura se realiza el enfriamiento de las piezas (Enfriar) y para finalizar el ciclo se descargan las piezas (Descargar). Al llegar a la posición final de recorrido del cargador, este se vacía (Vaciar) quedando el proceso terminado (Fin_Auto). El proceso representado en el programa Codesys, se puede visualizar en la Figura 5.15.

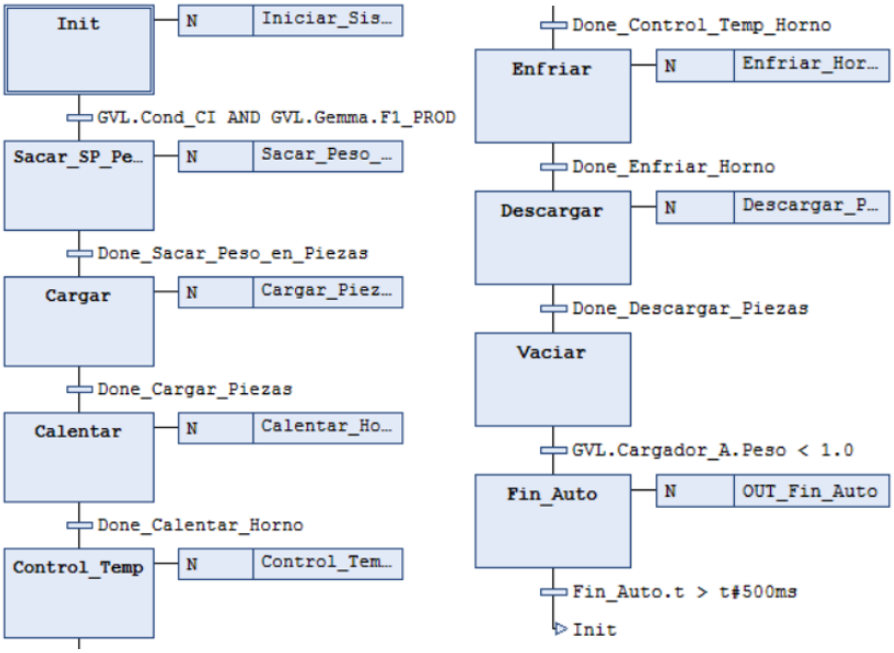

Figura 5.15 Código en SFC asociado a FB_F1_PROD_GEMMA.

Como se puede ver en el SFC, cada etapa tiene una acción a ejecutar, la cual está asociada al bloque de función F1 de la manera que se representa en la Figura.

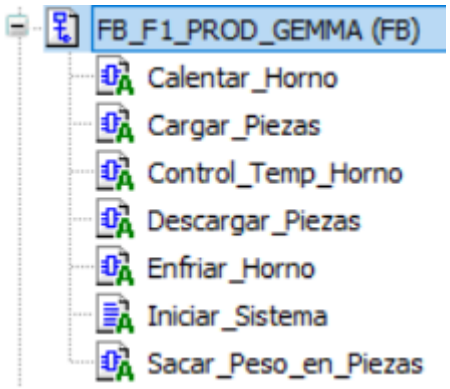

Figura 5.16 Conjunto de acciones asociadas al SFC de F1_PROD_GEMMA.

Se observa que casi todas las acciones son un módulo de programa escrito en lenguaje FBD, salvo el proceso de inicialización del sistema que se ha desarrollado en texto estructurado (ST). Sería conveniente el ver, por ejemplo, uno de los FBDs desarrollados, y el código de inicialización del sistema.

Comenzando con el FBD, se puede tomar por ejemplo el primero visualizado correspondiente a la acción Calentar_Horno. El FBD se corresponde con el que se aprecia en la Figura 5.17.

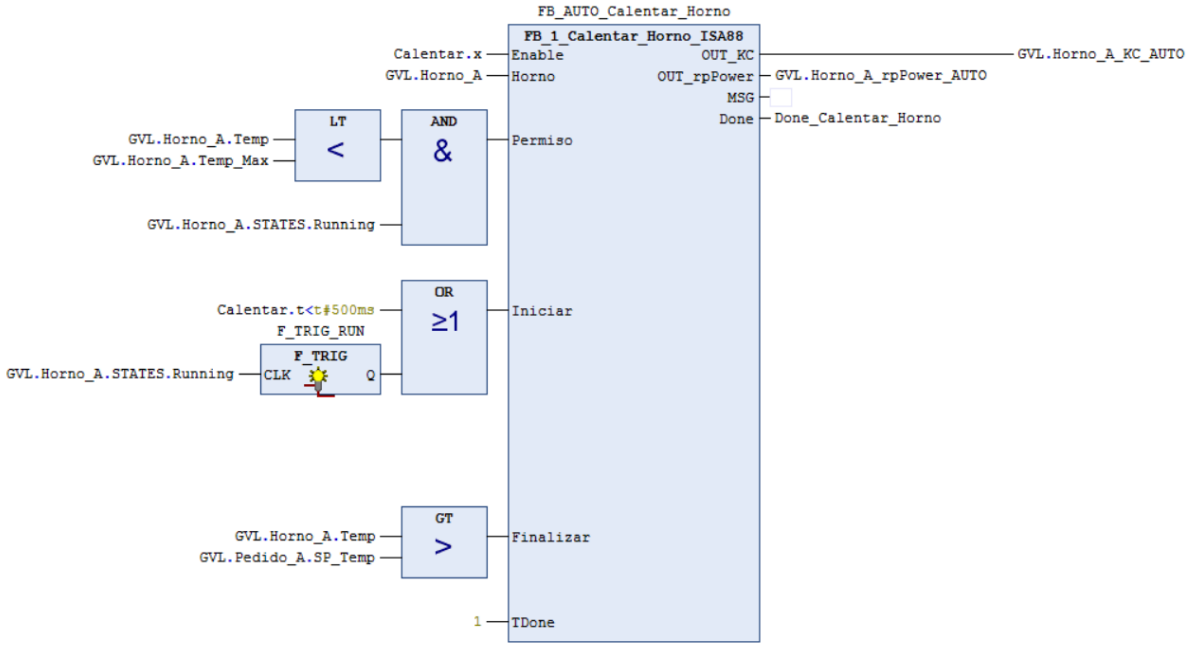

Figura 5.17 Código en FBD asociado a la acción Calentar_Horno del SFC de FB_F1_PROD_GEMMA.

El proceso que se desarrolla en esta acción es sencillo, una vez que en el SFC se llega al estado Calentar, se toma el valor de temperatura asignado por el usuario en la receta de proceso y se realiza el calentamiento de la pieza hasta ese valor. Cuando se alcanza el valor consigna de temperatura se evolucionaría a la siguiente etapa del proceso (Control_Temp, el cual iría ligado a la acción Control_Temp_Horno, en este caso).

El otro proceso de acción a comentar es el asociado a la inicialización y el cual, en el árbol de acciones recibe el nombre: Iniciar_Sistema. Como ya se había introducido, este se ha desarrollado en ST y básicamente realiza las tareas de inicialización de todos los elementos involucrados en el proceso de producción. También sirve para dar valor a la variable global Cond_CI que permite reconocer que se cumplen las condiciones iniciales del sistema (Alimentador en retroceso, Cargador en posición de descarga y el Horno frío, es decir con la temperatura por debajo de la temperatura de enfriamiento (TE). El código desarrollado en para esta acción es el aspecto mostrado en la Figura 5.18.

```
GVL.Cond_CI :=  GVL.Alimentador_A.A0 AND
            GVL.Cargador_A.FC0 AND
            GVL.Horno_A.Temp < GVL.Horno_A.TE ;


// Acciones inicializaicon del Alimentador_A
GVL.Alimentador_A_Amas_MM1 := FALSE ;
GVL.Alimentador_A_Amas_MM2 := FALSE ;
GVL.Alimentador_A_Amas_MM3 := FALSE ;
GVL.Alimentador_A_Amas_AUTO := FALSE ;

// Acciones inicializaicon del Cargador_A
GVL.Cargador_A_KM1_MM1 := FALSE ;
GVL.Cargador_A_KM1_AUTO := FALSE;

GVL.Cargador_A_KM0_MM2 := FALSE ;
GVL.Cargador_A_KM0_AUTO := FALSE;
GVL.Cargador_A_KM0_INI := Init.x AND NOT GVL.Cargador_A.FC0
                          AND (GVL.Horno_A.Temp < GVL.Horno_A.TE)
                          AND NOT GVL.Gemma.F4_VDes ;
```

```
// Acciones inicializaicon del Horno_A
GVL.Horno_A_KC_MM1 := FALSE;
GVL.Horno_A_KC_MM3 := FALSE;
GVL.Horno_A_KC_MM4 := FALSE;
GVL.Horno_A_KC_AUTO := FALSE;

GVL.Horno_A_KE_MM2 := FALSE;
GVL.Horno_A_KE_MM4 := FALSE;
GVL.Horno_A_KE_AUTO := FALSE;
GVL.Horno_A_KE_INI := Init.x AND (GVL.Horno_A.Temp > GVL.Horno_A.TE)
                      AND NOT GVL.Gemma.F4_VDes;

GVL.Horno_A_rpPower_MM1 := 0;
GVL.Horno_A_rpPower_MM2 := 0;
GVL.Horno_A_rpPower_MM3 := 0;
GVL.Horno_A_rpPower_MM4 := 0;
GVL.Horno_A_rpPower_AUTO := 0;
```

Figura 5.18 Código en ST desarrollado para la acción Iniciar_Sistema del SFC de FB_F1_PROD_GEMMA.

- **FB_F4_VDes_GEMMA**

Se conoce como el estado F4 en el GDMMA y es el proceso mediante el cual se puede realizar un mantenimiento y comprobación del funcionamiento de todos los módulos de equipamiento que incorpora el sistema de manera individualizada (verificación en desorden). En este punto es precisamente uno de los casos donde la aplicación de la norma ISA 88 más destaca, porque muestra la flexibilidad del sistema y la capacidad de todos los módulos de equipamiento de operar de manera independiente y posteriormente sincronizarse y unirse para completar una receta o una cadena de procesos solicitados por el usuario.

Las variables que se han declarado para este bloque función se muestran en la siguiente Figura 5.19.

Scope	Name	Address	Data type	Initialization	Comment
VAR_INPUT	**Enable**		BOOL		
VAR	**FB_Sacar_Pieza_Alim_A**		FB_1_Sacar_Pieza_Alim_ISA88		ME_1: ALIMENTADOR
VAR	**FB_Vaciar_Alim_A**		FB_2_Vaciar_Alim_ISA88		
VAR	**FB_Saca_SP_Peso_Alim_A**		FB_3_Sacar_SP_Peso_Alimentador_ISA88		
VAR	**FB_Cargar_Piezas**		FB_1_Cargar_Piezas_ISA88		ME_2: CARGADOR
VAR	**FB_Desargar_Piezas**		FB_2_Descargar_Piezas_ISA88		
VAR	**FB_Calentar_Horno**		FB_1_Calentar_Horno_ISA88		ME_3: HORNO
VAR	**FB_Enfriar_Horno**		FB_2_Enfriar_Horno_ISA88		
VAR	**FB_Control_Temp_Horno**		FB_3_Control_Temp_Horno_ISA88		
VAR	**FB_Calibra_TR_Horno**		FB_4_Calibra_TR_Horno_ISA88		
VAR	**R_TRIG_View_Alimentador_A**		R_TRIG		Anteriores
VAR	**RT_View_Calibra_TR**		R_TRIG		

Figura 5.19 Listado de variables asociado a FB_F4_VDes_GEMMA.

El código correspondiente ha sido desarrollado en FBD y básicamente consiste en un conjunto de bloques de función donde cada uno se corresponde con cada una de las acciones (servicios) que los diferentes módulos de equipamiento tienen asociadas. Estas acciones enumeradas son:

- Para el **Alimentador**: 1) Sacar piezas, 2) Vaciar Piezas, 3) Sacar el número de piezas según peso especificado.

- Para el **Cargador**: 1) Cargar piezas, 2) Descargar piezas.

- Para el **Horno**: 1) Calentar, 2) Enfriar, 3) Control de temperatura y 4) Calibrar termostato.

Puesto que todas siguen una estructura bastante semejante se puede analizar una de las acciones para ver cómo se implementan en el programa de verificación en desorden.

Por ejemplo, tomando el caso de la acción para sacar una pieza, básicamente habría que instanciar el bloque de función de sacar pieza, el cual se ha implementado en la carpeta

del módulo de equipamiento 1 (Alimentador) con el nombre FB_1_Sacar_Pieza_Alim_ISA88. En la llamada, tal y como se muestra en la Figura 5.20 se deben cumplir los requisitos de estar en verificación en desorden, alimentador en estado "Running" y encontrarse en modo manual, con esto al accionar el proceso se conseguirá sacar una pieza del alimentador.

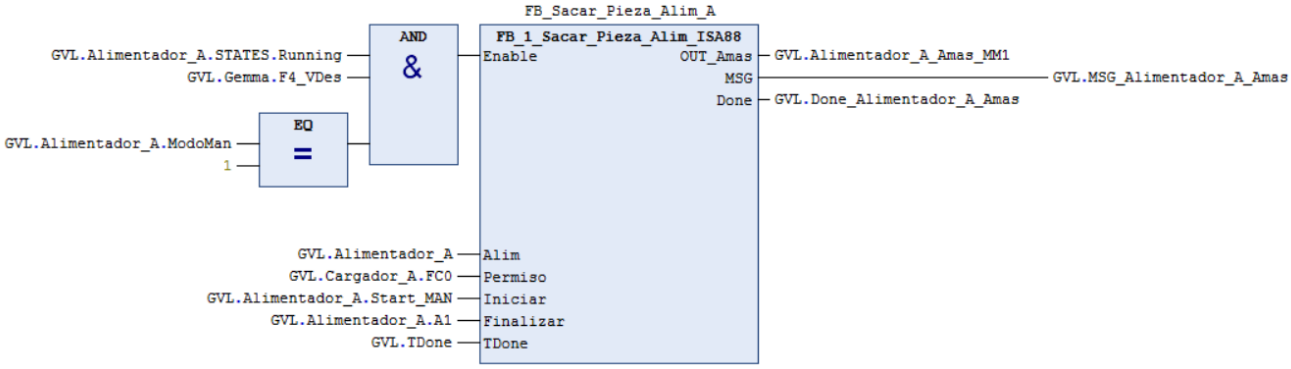

Figura 5.20 Código en FBD desarrollado para FB_F4_VDes_GEMMA focalizado en la parte en la que se llama al servicio del Alimentador de FB_1_Sacar_Pieza_Alim_ISA88.

Como se había mencionado y con el fin de no extender el documento, el resto de los procesos contenidos dentro de la verificación en desorden siguen una estructura de código similar, en todos los casos se llama al bloque de función correspondiente a la acción que debe de desempeñar un módulo de equipamiento en concreto y se le conectan las variables para cumplir las condiciones requeridas para realizar la operación deseada.

- **FB_GEMMA_AyFs**

Otro de los bloques de función generales permite gestionar el GDMMA diseñado para este proyecto, y recibe el nombre de FB_GEMMA_AyFs.

Las variables de este FB se muestran en la Figura 5.21. El código se ha realizado en lenguaje SFC (ver Figura 5.22**)** lo que permite una asociación directa con los estados del GEMMA y ha supuesto una implementación más sencilla e intuitiva.

Scope	Name	Address	Data type
VAR_INPUT	**SFCReset**		BOOL
VAR_OUTPUT	**SFCCurrentStep**		STRING
VAR_IN_OUT	**G**		DTGemma
VAR	**FB_MANUAL_ISA88_0**		FB_F4_VDes_GEMMA
VAR	**FB_PRODUCCION_ISA88_0**		FB_F1_PROD_GEMMA
VAR	**R_TRIG_F1_PROD**		R_TRIG

Figura 5.21 Listado de variables asociado a FB_GEMMA_AyFs

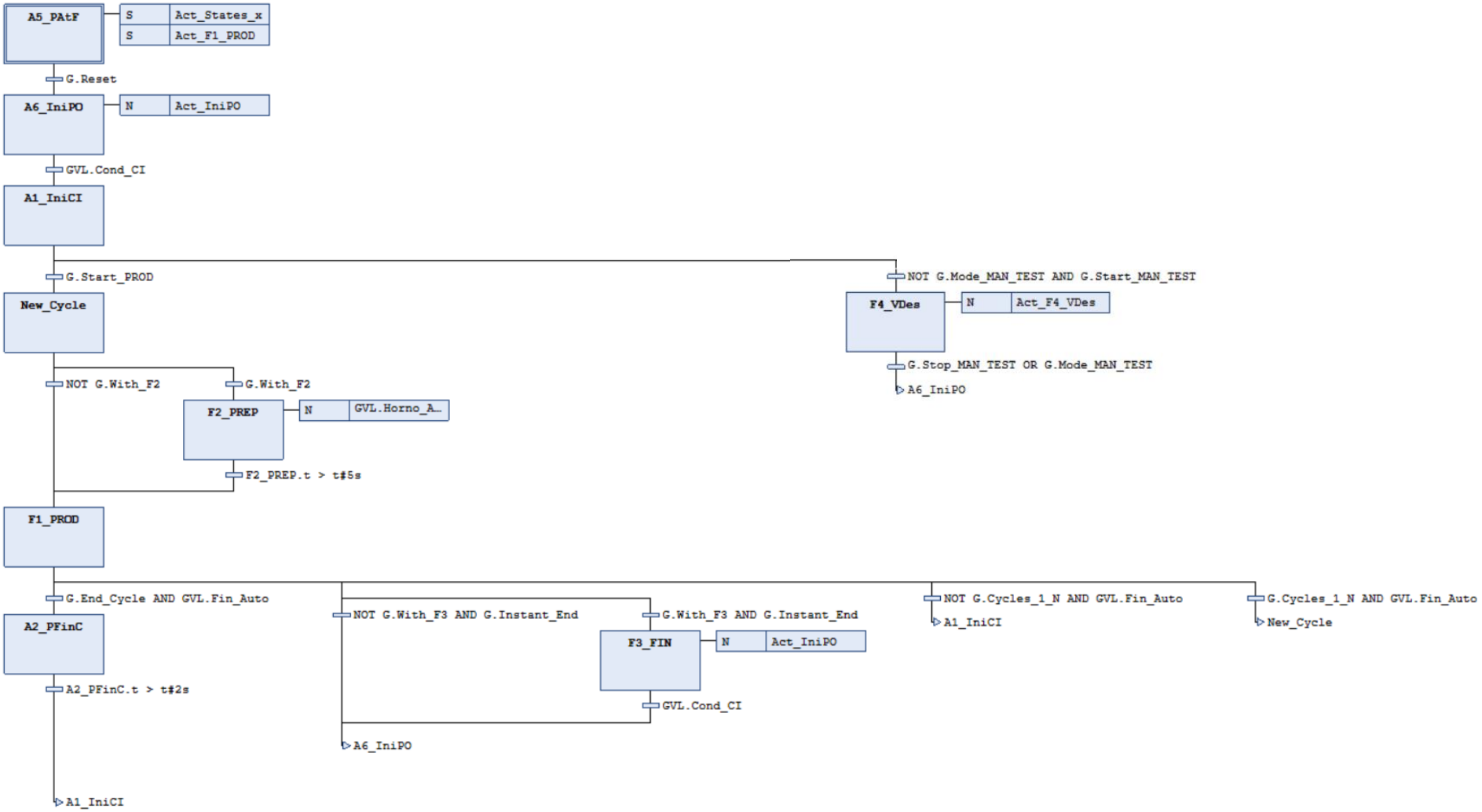

Figura 5.22 Código en SFC desarrollado para FB_GEMMA_AyFs.

Analizando el SFC contemplado en la Figura anterior, se pueden observar los diferentes estados del GDMMA. En el SFC se observa que se comienza en el estado A5 de preparación de arranque después de fallo y que tras realizar un Reset, se evoluciona hacia el siguiente estado de paro, A6 de espera a puesta del sistema en condiciones iniciales. Una vez alcanzadas las condiciones iniciales, se avanza hacia el estado A1 de paro en condiciones iniciales; tras esto el sistema ya tiene varias posibilidades de evolución dependiendo de a qué estado de funcionamiento (F) se le requiera avanzar. En esta aplicación en concreto, se puede evolucionar o hacia estado de funcionamiento normal de producción o hacia un estado de verificación en desorden.

En este caso, hay una serie de acciones asociadas a este SFC que las cuales se enmarcan en el árbol de acciones mostrado en la Figura 5.23 siguiente.

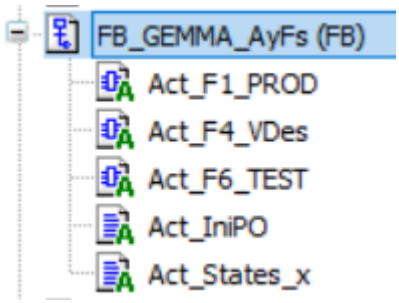

Figura 5.23 Listado de acciones asociadas a FB_GEMMA_AyFs.

En este bloque funcional las acciones referenciadas se han codificado en lenguaje de programación FBD, tanto para producción normal (Act_F2_PROD) como verificación en desorden (Act_F4_VDes); la acción Act_F6_TEST si bien está creada no ha sido implementada para este trabajo.

Por otra parte, las acciones para poner el sistema en condiciones iniciales (Act_IniPO) y para asignación de nombres a los estados (Act_States_x) se haN desarrollado en ST.

Puesto que las acciones no presentan unos bloques de función muy extensos y son relativamente sencillos de seguir, se procederá a visualizar los empleados finalmente en el proyecto.

Act_F1_PROD. Simplemente se realiza una llamada al bloque de función FB_F1_PROD_GEMMA ya introducido previamente. Su representación en Codesys se puede observar en la Figura 5.24.

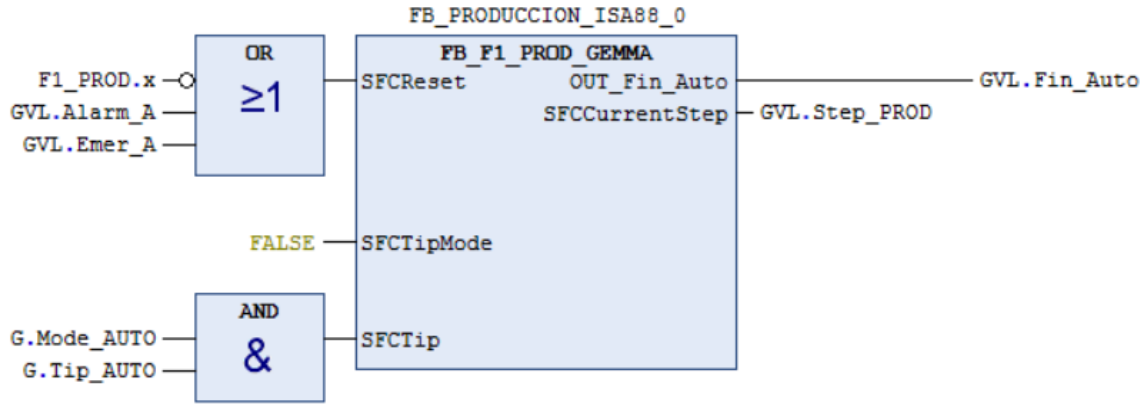

Figura 5.24 Código en FBD desarrollado para la acción Act_F1_PROD del SFC de FB_GEMMA_AyFs.

Act_F4_VDes, está asociada a la verificación en desorden del sistema. Lo único que realiza es la activación del bloque de función que gestiona el estado F4 del GEMMA, según (ver Figura 5.25):

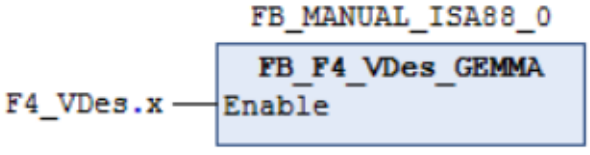

Figura 5.25 Código en FBD desarrollado para la acción Act_F4_VDes del SFC de FB_GEMMA_AyFs

Una vez indicadas las acciones desarrolladas en FBD, se mencionan las acciones desarrolladas en ST.

Act_IniPO. El código asociado a esta parte es el mismo que el ya previamente descrito en la acción Iniciar_Sistema de FB_F1_PROD_GEMMA.

Act_States_x: Es la última acción indicada en el SFC que se desarrolla en el FB_GEMMA_AyFs, permitiendo asignar los estados del Grafcet que implementa el GDMMA a las variables de la estructura DTGemma, así como identificar el nombre de la etapa utilizando la variable implícita SFCCurrentStep que utiliza Codeys v3.5.

```
// Asignacion de estados del Grafcet que implementa el GDMMA
G.Step := SFCCurrentStep;  // Y el nombre de la etapa correspondiente
G.A1_IniCI  := A1_IniCI.x;
G.A2_PFinC  := A2_PFinC.x;
G.A5_PAtF   := A5_PAtF.x;
G.A6_IniPO  := A6_IniPO.x;
// G.F5_VOrd: := F5_VOrd.x;  // Se gestiona con los parámetros SFCTipMode y SFCTip
G.New_Cycle := New_Cycle.x;
G.F1_PROD   := F1_PROD.x;
G.F2_PREP   := F2_PREP.x;
G.F3_FIN    := F3_FIN.x;
G.F4_VDes   := F4_VDes.x;
G.F6_TEST   := F6_TEST.x;
```

Figura 5.26 Código en ST desarrollado para la acción Act_States_x del SFC de FB_GEMMA_AyFs

- **FB_STATES_ISA88**

Finalizando el apartado de bloques de función generales, falta comentar el FB que gestiona los estados asociados a la norma ISA 88 que serán de aplicación para cada uno de los módulos de equipamiento definidos. Básicamente consiste en un desarrollo en ST que va asignando en qué estado se encuentra (Init, Running, Completed, Pausing, Paused…). Antes de visualizar el código, la única variable asociada a este bloque de función sería la de estado,

la cual va asociada a la estructura de datos de estados ISA 88, ya previamente referenciada. La variable se vería como se muestra en la Figura 5.27.

Scope	Name	Address	Data type
VAR_IN_OUT	**STATES**		DTStates_ISA88

Figura 5.27 Variable asociada a FB_STATES_ISA88.

El código desarrollado en ST para este bloque de función simplemente desarrolla el Grafcet que representa el diagrama de estados definido en ISA88 que se muestra en la siguiente Figura 5.28.

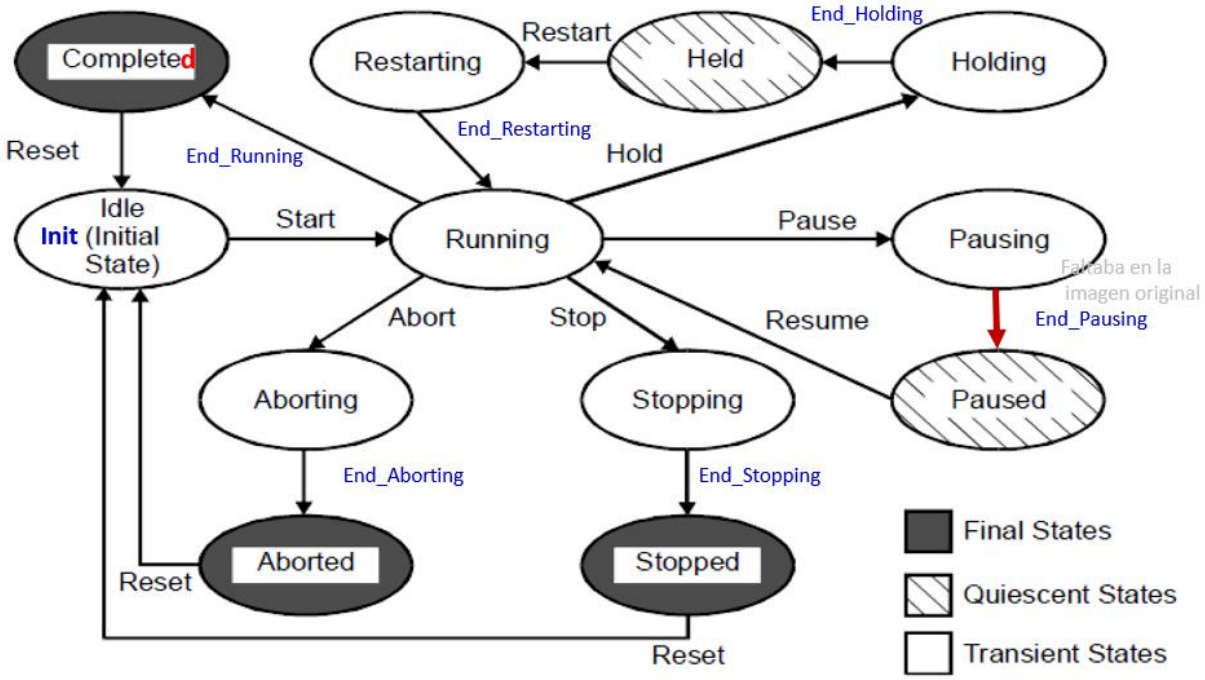

Figura 5.28 Diagrama de estados definidos por ISA88.

Se desarrollan las condiciones de activación y desactivación de cada estado y se utiliza el nombre característico que resulta muy útil para la implementación del interface de usuario. En la Figura 5.29 se aprecia un fragmento del código ST realizado para este módulo.

```
STATES.Init := ((( STATES.COMPLETED  OR STATES.STOPPED OR STATES.ABORTED ) AND STATES.Reset )
            OR STATES.Init )
            AND NOT STATES.Running ;

       IF STATES.Init THEN STATES.msg_STATE := 'Initial state'; END_IF;

STATES.Running := (((STATES.Init AND STATES.Start) OR (STATES.Paused AND STATES.Resume) OR (STATES.Restarting AND STATES.End_Restarting))
              OR STATES.Running)
              AND NOT  (STATES.COMPLETED OR STATES.Stopping OR STATES.Holding OR STATES.Aborting OR STATES.Pausing) ;

       IF STATES.Running THEN STATES.msg_STATE := 'Running'; END_IF;

STATES.COMPLETED := ((STATES.Running AND STATES.End_Running)
                OR STATES.COMPLETED )
                AND NOT STATES.Init ;

       IF STATES.COMPLETED THEN STATES.msg_STATE := 'COMPLETED'; END_IF;

STATES.Pausing := ((STATES.Running AND STATES.Pause)
               OR STATES.Pausing )
               AND NOT (STATES.Paused OR STATES.Stopping OR STATES.Holding OR STATES.Aborting);

       IF STATES.Pausing THEN STATES.msg_STATE := 'Pausing'; END_IF;
```

Figura 5.29 Código en ST desarrollado para FB_STATES_ISA88.

5.1.1.4 Programación de módulos de equipamiento

Finalizada, la explicación del apartado asociado a bloques de función y estructuras de datos de carácter general se procede igual con cada uno de los bloques de equipamiento. Puesto que en realidad son bloques que pueden operar de manera independiente, se han conceptualizado siguiendo una programación semejante, simplemente adaptándola a los servicios particulares de cada uno de los módulos. Se ha decidido desarrollar uno de los módulos de equipamiento en mayor detalle en este documento y se aconseja acudir, en caso de duda, al programa fuente implementado.

Por lo tanto, se ha tomado como base para la explicación detallada el módulo de equipamiento 1 (Alimentador) cuya estructura de carpeta en el árbol del proyecto se muestra en la siguiente Figura 5.30.

Figura 5.30 Desarrollo de programa asociada al módulo de equipamiento: Alimentador.

La estructura de datos del alimentador vendría condicionada por los estados ISA88 ya previamente introducidos, las entradas y salidas físicas asociadas al módulo de equipamiento (al ser el alimentador: A+, sensores A0, A1 y V0 y claramente el asociado a la seta de emergencia), parámetros asociados a la receta proporcionada por el usuario para operar (peso de cada pieza, número de piezas que cumplirían la selección de peso requerida por el usuario, valor de peso seleccionado por el usuario y peso real sacado por el alimentador), variables asociadas al proceso de verificación en desorden y variable asociada a la supervisión del módulo de equipamiento. La estructura de datos detallada en este párrafo se puede observar en la Figura 5.31.

```
STATES: DTStates_ISA88;      // Diagrama de estados del Alimentador (ISA88)

(* Entradas/salidas fisicas *)
Amas: BOOL;                  // Electroválvula accionamiento cilindro
A0: BOOL;                    // Detector retroceso cilindro
A1: BOOL;                    // Detector avance cilindro
V0: BOOL;                    // Detector de contenedor vacío (NA)
IxEmer : BOOL;               // Entrada de emergencia

(* Otros parámetros *)
Peso_Pieza: REAL := 40.0 ;   // Peso de cada pieza en el alimentador
SP_Piezas: UINT := 5;        // Numero de piezas a sacar del alimentador
SP_Peso: REAL := 145;        // Peso a sacar con el número de piezas  correspondiente
Peso: REAL;                  // Peso sacado

// Para gestión del modo Manual (Verificación en desorden)
ModoMan: UINT := 1 ;         // Modo Manual para Verificación en desorden
Start_MAN: BOOL;             // Pulsador para ejecución del modo Manual
TTest : INT := 5 ;           // Tiempo de ejecución de funcionalidad seleccionada en Modo Manual (en segundos)

// Para gestión de la supervisión del Modulo de equipamiento
TSup: INT := 3 ;             // Tiempo de supervisión de accionamientos (en s)
```

Figura 5.31 Estructura de datos asociada al módulo de equipamiento: Alimentador en Codesys.

Tras haber analizado la estructura de datos asociada al módulo de equipamiento descrito, tocaría comenzar a describir los bloques de función asociados al desempeño de las acciones asociadas al módulo de equipamiento correspondiente.

- **FB_1_Sacar_Pieza_Alim_ISA88**

Las variables asociadas a este servicio para alimentación de pieza se pueden observar en la Figura 5.32.

Scope	Name	Address	Data type
VAR_INPUT	**Enable**		BOOL
VAR_INPUT	**Alim**		DTAlimentador_ISA88
VAR_INPUT	**Permiso**		BOOL
VAR_INPUT	**Iniciar**		BOOL
VAR_INPUT	**Finalizar**		BOOL
VAR_INPUT	**TDone**		INT
VAR_OUTPUT	**OUT_Amas**		BOOL
VAR_OUTPUT	**MSG**		STRING
VAR_OUTPUT	**Done**		BOOL
VAR	**FB_ALIM_Amas**		FB_Actuator

Figura 5.32 Listado de variables asociadas al bloque de función FB_1_Sacar_Pieza_Alim_ISA88.

El código se ha desarrollado en FBD y se recurriría a instanciar FB_Actuator para realizar la acción de salida de pieza actuando sobre la señal A+. La representación del código que se implementa en esta acción queda reflejada en la Figura 5.33.

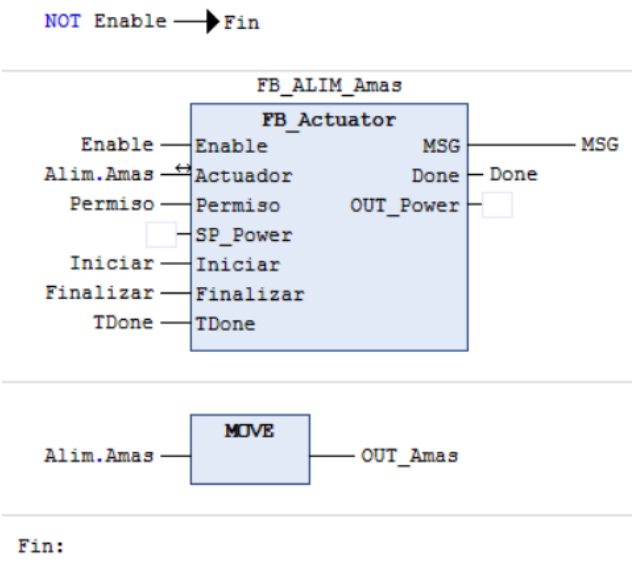

Figura 5.33 Código en FBD asociado al bloque de función FB_1_Sacar_Pieza_alim_ISA88.

- **FB_2_Vaciar_Alim_ISA88**

En este caso, el desarrollo de este servicio debido a su naturaleza repetitiva hasta detectar que no hay más piezas que sacar, se ha realizado en SFC. Siguiendo la estructuración habitual a lo largo del texto, se comienza visualizando las variables que componen esta acción que se representan en la Figura 5.34.

Scope	Name	Address	Data type	Comment
VAR_INPUT	**SFCReset**		BOOL	Permite esta funcionalidad
VAR_INPUT	**Alim**		DTAlimentador_ISA88	
VAR_INPUT	**Permiso**		BOOL	Interlock
VAR_INPUT	**Iniciar**		BOOL	Condición para iniciar la activación de Amas
VAR_INPUT	**Finalizar**		BOOL	Condición para finalizar la activación tras el retardo, si se parametriza
VAR_INPUT	**TDone**		INT	Tiempo de permanencia de señal de Done (en segundos)
VAR_OUTPUT	**OUT_Amas**		BOOL	
VAR_OUTPUT	**MSG**		STRING	
VAR_OUTPUT	**Done**		BOOL	Confirmación de que la funcionalidad ha sido ejecutada (en ms)

Figura 5.34 Listado de variables asociadas al bloque de función FB_2_Vaciar_Alim_ISA88.

En cuanto al desarrollo de código, es un proceso secuencial. Se comienza en un estado inicial, una vez que se disponga del permiso se realiza la actuación sobre el actuador A+ para sacar pieza, se retrae el actuador, se realiza la comprobación de si hay pieza y si es así se inicia de nuevo el proceso de sacar pieza, de lo contrario, para la acción, se avisa y se vuelve a condiciones iniciales. El código se ve representado en la Figura 5.35.

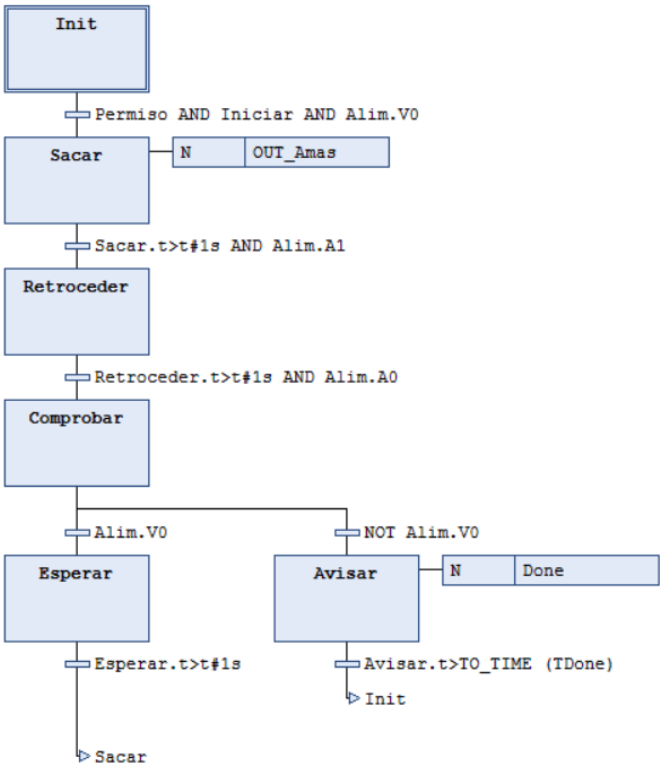

Figura 5.35 Código en SFC desarrollado para el bloque de función FB_2_Vaciar_Alim_ISA88.

- **FB_3_Sacar_SP_Peso_Alimentador_ISA88**

Este servicio realiza un procedimiento muy semejante a la acción anterior, pero en este caso sacando un número de piezas cuyo valor en peso sea igual o algo superior al especificado por el usuario o en la receta.

Las variables asociadas a este bloque funcional se muestran en la Figura 5.36.

Scope	Name	Address	Data type	Comment
VAR_INPUT	**SFCReset**		BOOL	Permite esta funcionalidad
VAR_INPUT	**Alim**		DTAlimentador_ISA88	
VAR_INPUT	**Permiso**		BOOL	Interlockd
VAR_INPUT	**SP_Peso**		REAL	Peso a sacar del alimentador
VAR_INPUT	**Iniciar**		BOOL	Condición para iniciar la activación de Amas
VAR_INPUT	**TDone**		INT	Tiempo de permanencia de señal de Done (en segundos)
VAR_OUTPUT	**OUT_Amas**		BOOL	
VAR_OUTPUT	**OUT_Peso**		REAL	
VAR_OUTPUT	**MSG**		STRING	
VAR_OUTPUT	**Done**		BOOL	Confirmación de que la funcionalidad ha sido ejecutada (en ms)

Figura 5.36 Listado de variables asociadas al bloque de función FB_3_Sacar_SP_Peso_Alimentador_ISA88.

Puesto que es un concepto muy semejante al de la acción de sacar piezas hasta que se vacíe el alimentador, esta parte de código también ha sido desarrollado en SFC.

El procedimiento de actuación sería el siguiente: se parte de condiciones iniciales, se verifica que hay pieza en el alimentador, comprobado que hay pieza, se saca la pieza y se suma el incremento de peso correspondiente en función de la pieza sacada. A continuación, se dan dos posibles divergencias por lo que hay que verificar si hay pieza en el alimentador y de ser así volver a reiterar el proceso o el caso contrario de haber alcanzado el peso consigna solicitado por el usuario y tener que parar la acción, realizar el aviso y volver a condiciones iniciales. El desarrollo del programa descrito se puede observar en la Figura 5.37.

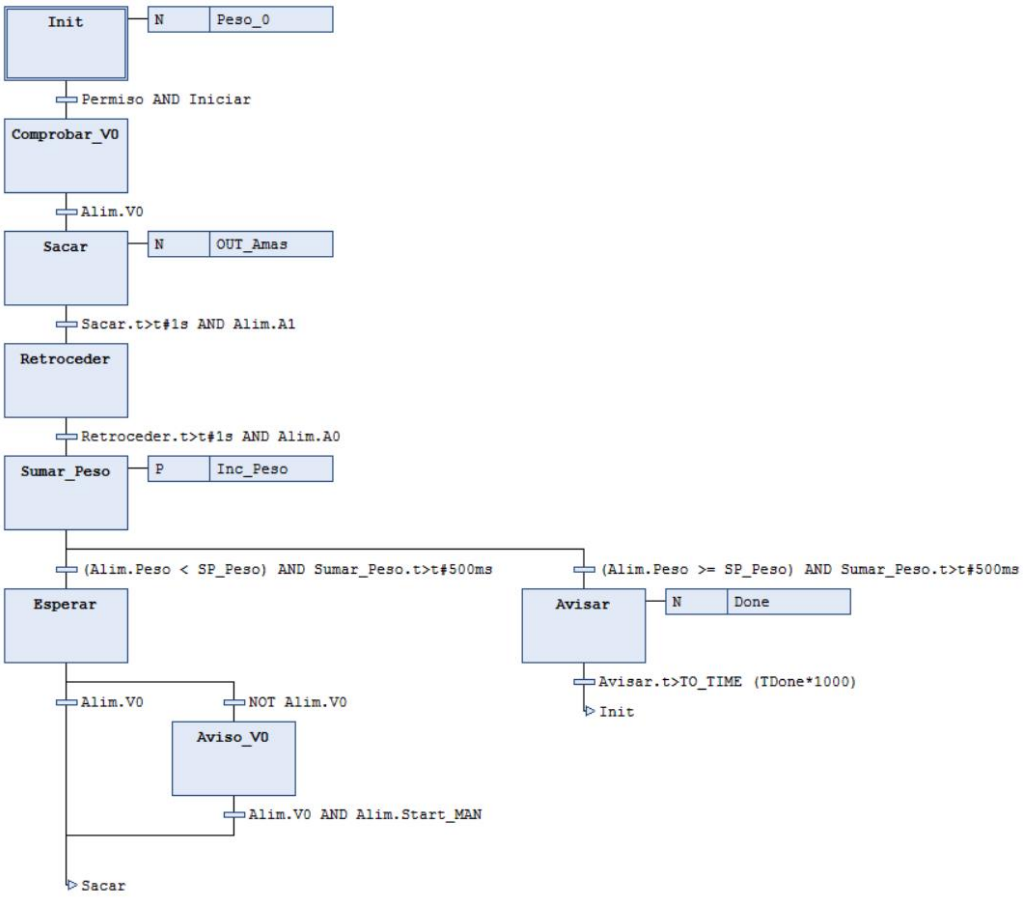

Figura 5.37 Código en SFC desarrollado para FB_3_Sacar_SP_Peso_Alimentador_ISA88.

Una vez comentadas los servicios que ofrece equipamiento asociado al alimentador por medio de los bloques funcionales correspondientes, quedan por explicar los dos últimos bloques de función asociados a cada módulo de equipamiento y que se corresponden con: 1) FB_ALIMENTADOR_ISA88_ST, es decir, la gestión de los estados del módulo de equipamiento y 2) FB_Alim_ISA88_SUP, que tiene que ver con la gestión de avisos, defectos, alarmas y emergencia.

- **FB_ALIMENTADOR_ISA88_ST**

Las variables asociadas serían las siguientes vistas en la Figura 5.38.

Scope	Name	Address	Data type
VAR	**FB_Alim_ISA88_SUP_A**		FB_Alim_ISA88_SUP
VAR	**FB_STATES_ISA88_COMPLET_ALIM_A**		FB_STATES_ISA88

Figura 5.38 Listado de variables asociadas a FB_ALIMENTADOR_ISA88_ST.

En cuanto al código, ha sido desarrollado en ST y siempre sigue el mismo orden para cada uno de los módulos de equipamiento que es el siguiente: 1) Se llama al bloque de función de supervisión del módulo de equipamiento, 2) se asocian las transiciones de los estados del módulo de equipamiento según el resultado de la supervisión y las señales correspondientes, 3) se llama al bloque de función de gestión de estados del módulo de equipamiento con asignación de mensajes, 4) se ejecutan las acciones pertinentes en los estados del módulo de equipamiento y 5) se actualizan los valores de las transiciones en los estados del módulo de equipamiento. El código que desarrolla la explicación dada viene representado en la siguiente Figura 5.39.

```
// 1 Se llama al FB de supervisión del Alimentador
GVL.Alimentador_A.IxEmer := GVL.I_xEmer;
FB_Alim_ISA88_SUP_A (Alim  := GVL.Alimentador_A);

GVL.Alimentador_A.STATES.Aviso := FB_Alim_ISA88_SUP_A.OUT_xAviso ;
GVL.Alimentador_A.STATES.Defecto := FB_Alim_ISA88_SUP_A.OUT_xDefect ;
GVL.Alimentador_A.STATES.Alarma := FB_Alim_ISA88_SUP_A.OUT_xAlarm ;
GVL.Alimentador_A.STATES.Emer := FB_Alim_ISA88_SUP_A.OUT_xEmer ;

// 2. Se asocian las transiciones de los estados del Alimentador según el resultado de la supervisión
//    y las señales correspondientes (por ejemplo, I_xEmer - Seta de emergencia)
GVL.Alimentador_A.STATES.Abort := NOT GVL.I_xEmer OR (FB_Alim_ISA88_SUP_A.Alim.STATES.Alarma);  // Pulsación seta o fallo Amas/A1
GVL.Alimentador_A.STATES.Pause := FB_Alim_ISA88_SUP_A.Alim.STATES.Aviso; // Contenedor del alimentador vacío, V0
GVL.Alimentador_A.STATES.Resume := NOT FB_Alim_ISA88_SUP_A.Alim.STATES.Aviso; // Contenedor del alimentador no vacío
GVL.Alimentador_A.STATES.Reset := GVL.Gemma.Reset;

// 3. Se llama al FB de gestión de estados del alimentador con asignación de mensajes
FB_STATES_ISA88_COMPLET_ALIM_A(STATES:= GVL.Alimentador_A.STATES );

GVL.Alimentador_A.STATES.msgAviso := FB_Alim_ISA88_SUP_A.Alim.STATES.msgAviso ;
GVL.Alimentador_A.STATES.msgDefecto := FB_Alim_ISA88_SUP_A.Alim.STATES.msgDefecto ;
GVL.Alimentador_A.STATES.msgAlarma := FB_Alim_ISA88_SUP_A.Alim.STATES.msgAlarma ;
GVL.Alimentador_A.STATES.msgEmer := FB_Alim_ISA88_SUP_A.Alim.STATES.msgEmer ;

GVL.Azul_Reset_ALIM_A := GVL.Alimentador_A.STATES.COMPLETED
                      OR GVL.Alimentador_A.STATES.ABORTED
                      OR GVL.Alimentador_A.STATES.STOPPED
                      OR GVL.Alimentador_A.STATES.Alarma
                      OR GVL.Alimentador_A.STATES.Emer ;
```

```
// 4. Se ejecutan las acciones pertinentes en los estados del alimentador; en este caso se desactiva Q_xAmas según condiciones
IF ( GVL.Alimentador_A.STATES.Pausing
     OR GVL.Alimentador_A.STATES.Stopping
     OR GVL.Alimentador_A.STATES.Aborting) THEN
     GVL.Alimentador_A_Amas_MM1 := FALSE;
     GVL.Alimentador_A_Amas_MM2 := FALSE;
     GVL.Alimentador_A.Amas := FALSE ;
END_IF

// 5. Se actualizan los valores de las transiciones en los estados del alimentador
GVL.Alimentador_A.STATES.End_Pausing := GVL.Alimentador_A.V0 AND GVL.Alimentador_A.A0 ;
GVL.Alimentador_A.STATES.End_Stopping := GVL.Alimentador_A.A0;
GVL.Alimentador_A.STATES.End_Aborting := GVL.Alimentador_A.A0 AND GVL.I_xEmer;
```

Figura 5.39 Código en ST asociado al bloque de función FB_ALIMENTADOR_ISA88_ST.

- **FB_Alim_ISA88_SUP**

La finalidad de este bloque funcional es gestionar las avisos, alarmas, defectos y emergencia que involucra su configuración y funcionamiento, aportando las causas que los generan, su clasificación y mensaje de salida correspondiente. Las variables declaradas de este bloque funcional son (ver Figura 5.40):

Scope	Name	Address	Data type
VAR_INPUT	**Alim**		DTAlimentador_ISA88
VAR_OUTPUT	**OUT_xAviso**		BOOL
VAR_OUTPUT	**OUT_xDefect**		BOOL
VAR_OUTPUT	**OUT_xAlarm**		BOOL
VAR_OUTPUT	**OUT_xEmer**		BOOL
VAR	**TON_Sup_V0**		TON
VAR	**TON_Sup_Amas**		TON
VAR	**TON_Sup_A01**		TON
VAR	**SR_Alarm**		SR
VAR	**SR_Emer**		SR
VAR	**SR_Aviso**		SR

Figura 5.40 Listado de variables del bloque de función FB_Alim_ISA88_SUP.

En este caso, el código para esta tarea ha sido desarrollado en CFC (*Continuous Function Chart*) que, aunque no pertenece al estándar IEC 61131-3 se ha considerado conveniente su uso por la facilidad de creación de bloques conectados con diferentes tipos de datos, la visualización y la depuración rápida de la lógica de control.

Realizando un análisis de las diferentes situaciones que se pueden producir, en base a los mensajes a generar, se comienza por Defecto, que si está en FALSE (Alim.STATES.Defecto := FALSE) no hay mensaje, mientras que si hay defecto (TRUE) se mostrará el mensaje: ‘AVISO: Incongruencia A0-A1’, el desarrollo de este código se ve en la Figura 5.41.

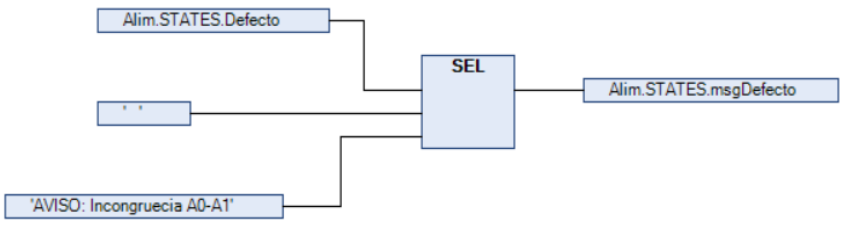

Figura 5.41 Fragmento de código en CFC asociado al mensaje de defecto generado en FB_Alim_ISA88_SUP.

Siguiendo con los casos de mensajes que se dan, se ha implementado un caso de Aviso, si el alimentador se encuentra vacío (no detecta pieza en V0). En todos los casos de mensaje, la instrucción utilizada es la misma y el código de proyección de mensaje de aviso se vería como se visualiza en la Figura 5.42.

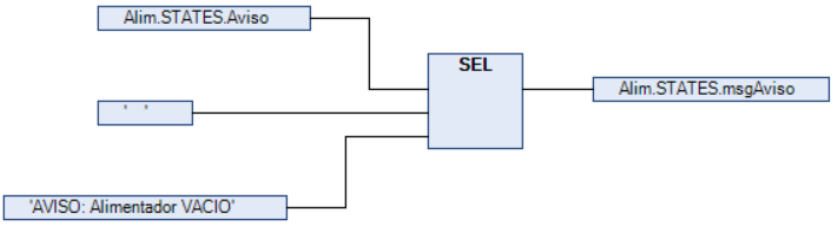

Figura 5.42 Fragmento de código en CFC asociado al mensaje de aviso generado en FB_Alim_ISA88_SUP.

Para el caso de Alarma, se ha implementado para cuando se produzca un fallo en el cilindro. Como la explicación seguiría el mismo proceso ya explicado, se procede a ver el código en la Figura 5.43.

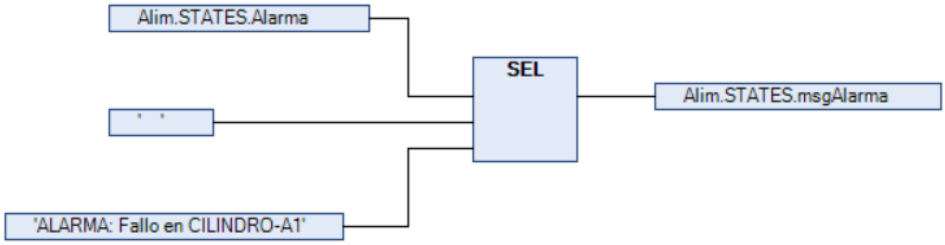

Figura 5.43 Fragmento de código en CFC asociado al mensaje de alarma generado en FB_Alim_ISA88_SUP.

Y finalmente en el caso de Emergencia, viene referenciada por la pulsación de la seta de emergencia. El código sigue el mismo que el ya citado, simplemente bastaría con visualizar la siguiente Figura 5.44 para ver similitud.

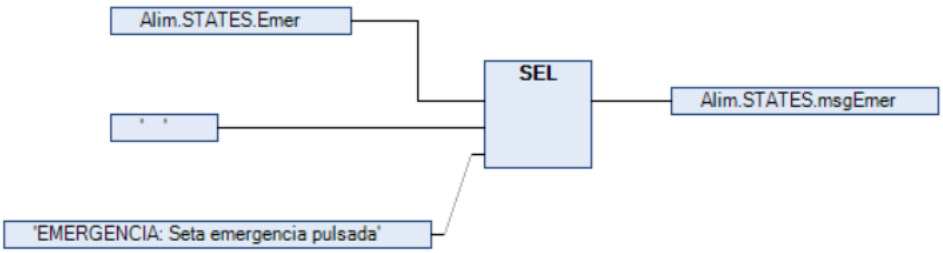

Figura 5.44 Fragmento de código en CFC asociado al mensaje de emergencia generado en FB_Alim_ISA88_SUP.

Sin embargo, esto es solo el despliegue de mensajes que sirve para observar el correspondiente defecto, aviso, alarma y emergencia que se han generado y por qué, si bien aún no se ha visto la gestión real de estos posibles estados.

El código que genera la gestión de los estados es realmente el indicado en la Figura 5.45. Por orden se visualizaría que, si se detecta que V0 no recibe señal de pieza en el tiempo establecido, se genera el caso de alimentador vacío -> Aviso; para el caso de que el actuador de cilindro esté activo, pero no se esté recibiendo señal de cilindro desplegado en A1 se genera el fallo en cilindro-A1 -> Alarma; para el caso de que se pulse la seta de emergencia -> Emergencia. Solucionado el problema, para eliminar la generación de estado de Alarma o Emergencia habrá que pulsar el Reset.

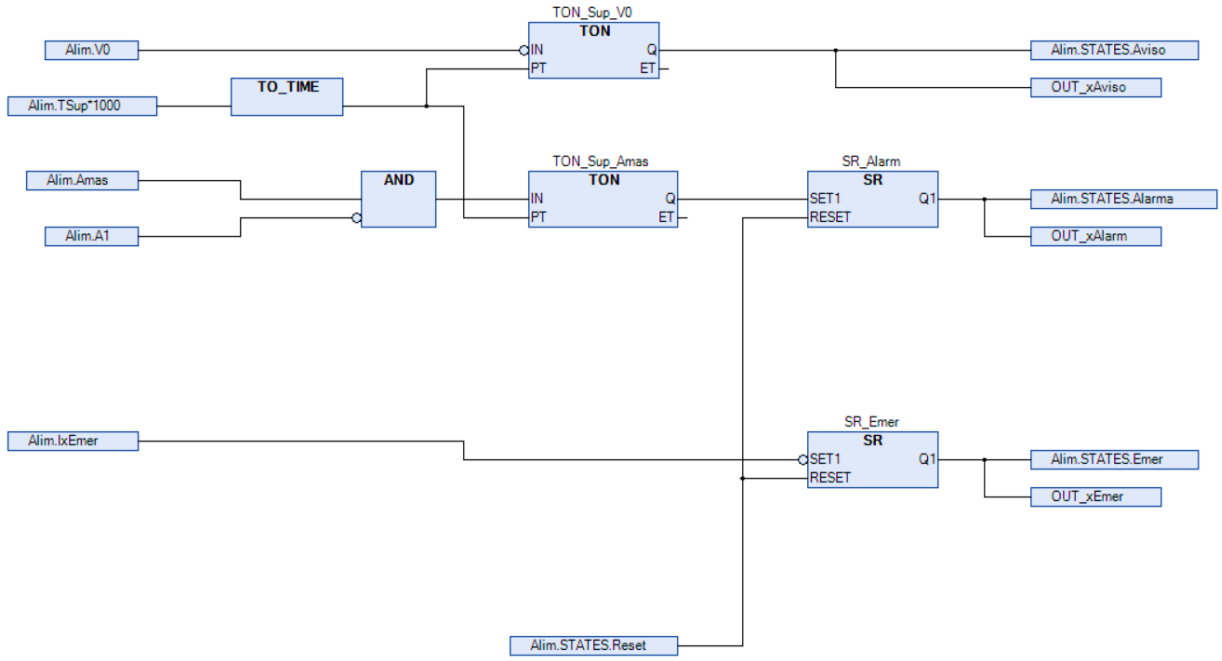

Figura 5.45 Fragmento de código en CFC encargado de gestionar aviso, alarma y emergencia en el Alimentador.

Con esto ya se ha completado la explicación el árbol de estructura de un módulo de equipamiento (Alimentador). Para los otros dos, es exactamente igual puesto que siguen la misma normativa ISA 88 en su implementación. No se va a explicar con el mismo detalle, pero sí que convendría visualizar el árbol de estructura general de cada uno de los otros módulos para ver que, cada uno con sus acciones correspondientes a la naturaleza de cada módulo, es realmente muy similar.

Por ejemplo, visualizando la estructura del módulo de equipamiento asociado a cargador y referenciado con el nombre ME_2_CARGADOR_ISA88 se vería la siguiente estructura mostrada en la Figura 5.46.

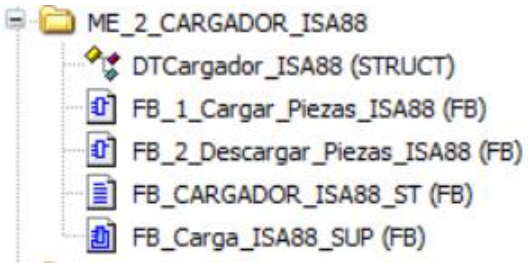

Figura 5.46 Estructura implementada en Codesys para el módulo de equipamiento: Cargador.

Como se puede ver, es exactamente lo mismo. El cargador dispone de dos servicios principales: 1) cargar piezas (FB_1_Cargar_Piezas_ISA88) y 2) descargar piezas (FB_2_Descargar_Piezas_ISA88), luego está la gestión de los estados del cargador y finalmente la gestión de avisos, defectos, alarmas y emergencia.

En el caso del último módulo de equipamiento es el horno y pasaría lo mismo que con los anteriores módulos de equipamiento, el árbol de desarrollo asociado tendría el siguiente aspecto representado en la Figura 5.47.

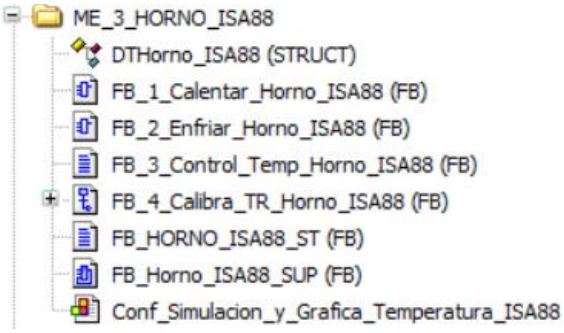

Figura 5.47 Estructura implementada en Codesys para el módulo de equipamiento: Horno.

Este módulo de equipamiento presenta cuatro servicios: 1) calentar (FB_1_Calentar_Horno_ISA88), 2) enfriar (FB_2_Enfriar_Horno_ISA88), 3) Control de temperatura a valor establecido (FB_3_Control_Temp_Horno_ISA88) y 4) la calibración de temperatura de calentamiento y temperatura objetivo de enfriamiento (FB_4_Calibra_TR_Horno_ISA88).

Tras las funcionalidades, se presenta la gestión de los estados del módulo de equipamiento (FB_HORNO_ISA88_ST, en este caso) y la gestión de defecto, alarma, aviso y emergencia (FB_Horno_ISA88_SUP). En este módulo se ha implementado un elemento de visualización del sistema de control de primer orden asociado a la evolución de la temperatura (Conf_Simulacion_y_Grafica_Temperatura_ISA88) que se verá con mayor detalle en el siguiente apartado de este capítulo de elementos de visualización desarrollados en Codesys.

5.1.2- Programas generales del sistema

Para finalizar el análisis de la programación asociada al control del sistema, se comentan a continuación un conjunto de POUs generales bajo la carpeta del árbol del proyecto identificada por PRGs_GENERALES_ISA88 (ver Figura 5.48) y que se corresponden con: funciones de escalado, programas asociados a las entradas y salidas y el programa "principal" (MAIN), aparte de las pantallas de explotación que se analizarán más adelante.

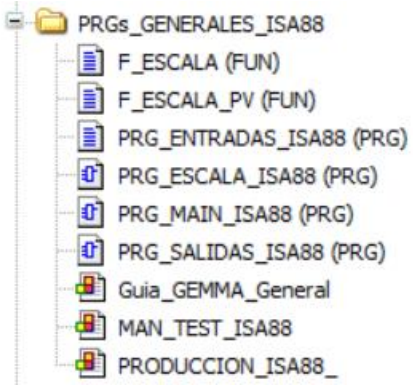

Figura 5.48 Estructura de POUs generales desarrollados en Codesys.

- **F_ESCALA_PV**

La función de escalado que se ha empleado ha sido F_ESCALA_PV que presenta las siguientes variables (Figura 5.49):

Scope	Name	Address	Data type	Comment
VAR_INPUT	**IN_CH_PV**		INT	Valor de la conversión A/D del canal
VAR_INPUT	**IN_CH_MIN**		INT	Valor máximo de la conversión
VAR_INPUT	**IN_CH_MAX**		INT	Valor mínimo de la conversión
VAR_INPUT	**IN_PV_MIN**		REAL	Valor máximo de la variable de proceso en unidades de ingeniería
VAR_INPUT	**IN_PV_MAX**		REAL	Valor máximo de la variable de proceso en unidades de ingeniería
VAR_OUTPUT	**OUT_OV**		BOOL	Valores superiores a CH_MAX del valor de conversión CH
VAR	**Pendiente**		REAL	

Figura 5.49 Listado de variables asociado a F_ESCALA_PV.

Tras observar las variables que se emplean en el desarrollo de la función, el código se ha desarrollado en ST y simplemente se calcula la pendiente de proporcionalidad y una vez obtenida la pendiente se aplica la ecuación de una recta. El código es muy sencillo y queda reflejado en la Figura 5.50.

```
// Se calcula la pendiente de proporcionalidad
Pendiente := (IN_PV_MAX - IN_PV_MIN) / INT_TO_REAL (IN_CH_MAX - IN_CH_MIN) ;
// Y se calcula el valor de salida
F_ESCALA_PV := IN_PV_MAX + Pendiente * (IN_CH_PV - IN_CH_MAX) ;
// Overflow
OUT_OV := IN_CH_PV > IN_CH_MAX ;
```

Figura 5.50 Código desarrollado en ST para función F_ESCALA_PV.

- **PRG_ENTRADAS_ISA88**

En este caso el programa está generado en ST y es un bloque asignación de valor de las entradas a las variables en caso de que haya una planta física conectada (alimentador, cargador y horno) como se puede visualizar en la Figura 5.51.

```
(* Tiene efecto solo si el PLC real está conectado a la planta en operación real,
   es decir NO en Simulación *)

// Entradas del Alimentador_1
IF GVL.Simula_Alim_1 THEN
    GVL.I_xA0 := GVL.Alimentador_A.A0 ;
    GVL.I_xA1 := GVL.Alimentador_A.A1 ;
    GVL.I_xV0 := GVL.Alimentador_A.V0 ;
END_IF

// Entradas del Cargador_1
IF GVL.Simula_Carga_1 THEN
    GVL.I_xFC0 := GVL.Cargador_A.FC0 ;
    GVL.I_xFC1 := GVL.Cargador_A.FC1 ;
    GVL.I_iChPeso := GVL.Cargador_A.Canal_Peso ;
END_IF

// Entradas del Horno_1
IF GVL.Simula_Horno_1 THEN
    GVL.I_xHC := GVL.Horno_A.HC ;
    GVL.I_xCG := GVL.Horno_A.CG ;
    GVL.I_xKCA := GVL.Horno_A.KCA ;
    GVL.I_xKEA := GVL.Horno_A.KEA ;
    GVL.I_xTR := GVL.Horno_A.TR ;
    GVL.I_iChTemp := GVL.Horno_A.Canal_Temp ;
END_IF
```

Figura 5.51 Código en ST para el PRG_ENTRADAS_ISA88.

- **PRG_ESCALA_ISA88**

Este programa aplica la función de escalado a las lecturas de las señales de la báscula (peso) y del sensor de temperatura en los canales de entrada analógicos del PLC correspondientes.

El desarrollo se ha realizado en FBD y resulta el fragmento de código mostrado en la Figura 5.52.

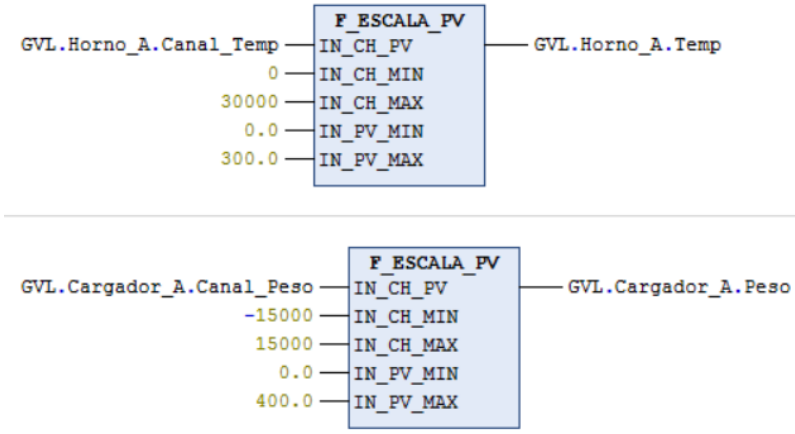

Figura 5.52 Código en FBD para PRG_ESCALA_ISA88.

- **PRG_MAIN_ISA88**

Este es el programa principal (MAIN) que se ejecuta inicialmente en cada ciclo de scan del PLC, si bien dada la metodología de programación utilizada, queda bastante reducido porque el grueso de las acciones y tareas de los módulos de equipamiento queda contenido en ellos.

Comenzando con las variables contenidas en esta POU, simplemente se vería una referencia a los bloques de función creados para la gestión de estados de cada uno de los módulos de equipamiento y en el caso de los bloques de función generales ya comentados, el de gestión de defectos y el de estados del GDMMA como se puede visualizar en la Figura 5.53.

Scope	Name	Address	Data type
VAR	**FB_ALIMENTADOR_ISA88_ST_0**		FB_ALIMENTADOR_ISA88_ST
VAR	**FB_CARGADOR_ISA88_ST_0**		FB_CARGADOR_ISA88_ST
VAR	**FB_HORNO_ISA88_ST_0**		FB_HORNO_ISA88_ST
VAR	**FB_D123_DEFECTS_ST_A**		FB_D123_DEFECTS_ST
VAR	**FB_GDMMA_AyFs_A**		FB_GEMMA_AyFs
VAR	**R_TRIG_GEMMA_AyFs_A**		R_TRIG

Figura 5.53 Listado de variables asociado a PRG_MAIN_ISA88.

Centrando el punto de vista en el código, este se ha desarrollado en FBD y quedaría solo para una habilitación haciendo TRUE en las entradas de los bloques de función asociados a la gestión de estados de los módulos de equipamiento, la habilitación del bloque de función asociado a la gestión de defectos de los tres módulos de equipamiento y la gestión del GDMMA por la presencia de alarmas o fallos. Por tanto, el asociado al programa principal se muestra en la Figura 5.54.

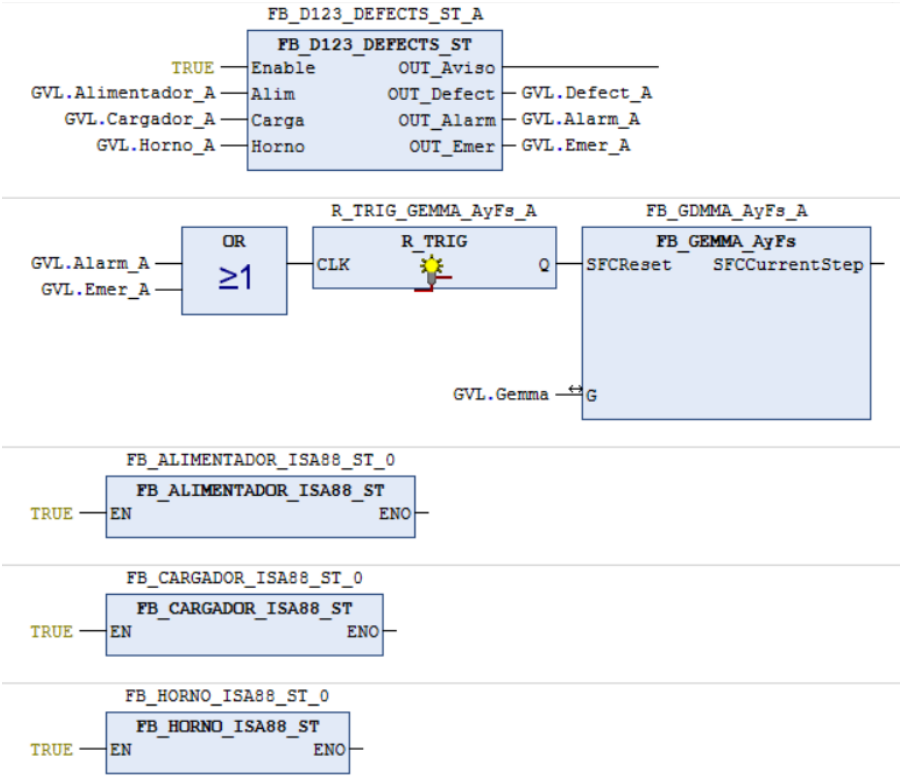

Figura 5.54 Código en FBD asociado a PRG_MAIN_ISA88.

- **PRG_SALIDAS_ISA88**

Esta parte de programa no tiene declaradas variables ni instancias a bloques funcionales pues solamente utiliza variables globales. Se ha desarrollado en lenguaje gráfico FBD pues permite un seguimiento apropiado de las condiciones que permiten la activación o no de los elementos actuadores de cada módulo de equipamiento.

Por ejemplo, para el alimentador sería actuar sobre A+ cuando correspondiese, KM1 y KM0 para ascenso y descenso del cargador respectivamente, KC y KE para calentamiento y enfriamiento respectivamente y el valor de consigna de potencia a aplicar para la realización del calentamiento en el horno.

Permite gestionar también cómoda y centralizadamente los estados en los que pueden operar los dispositivos y las seguridades en la operación de dichos componentes de proceso.

El código implementado para completar este programa se ve reflejado en la Figura 5.55, Figura 5.56 y Figura 5.57.

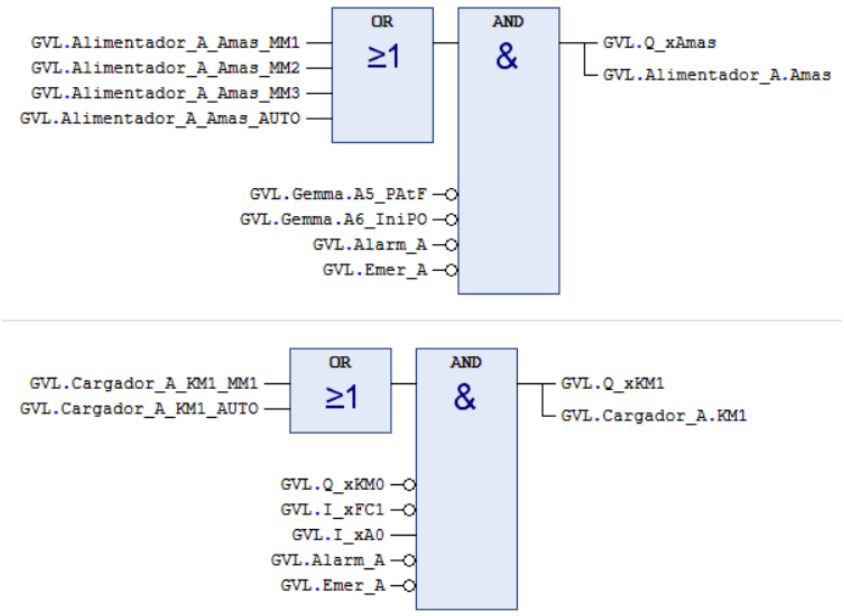

Figura 5.55 Fragmento de código en FBD de PRG_SALIDAS_ISA88 (I).

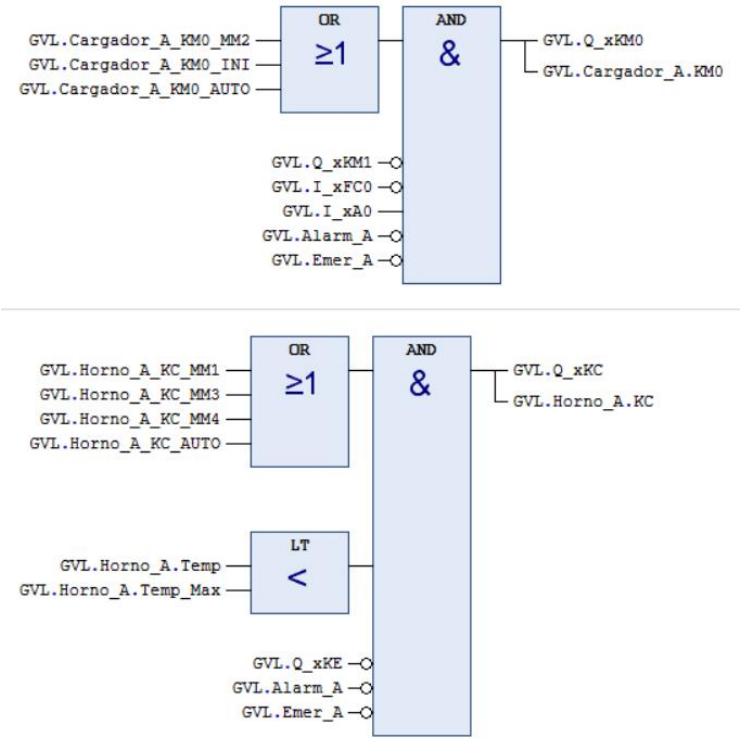

Figura 5.56 Fragmento de código en FBD de PRG_SALIDAS_ISA88 (II).

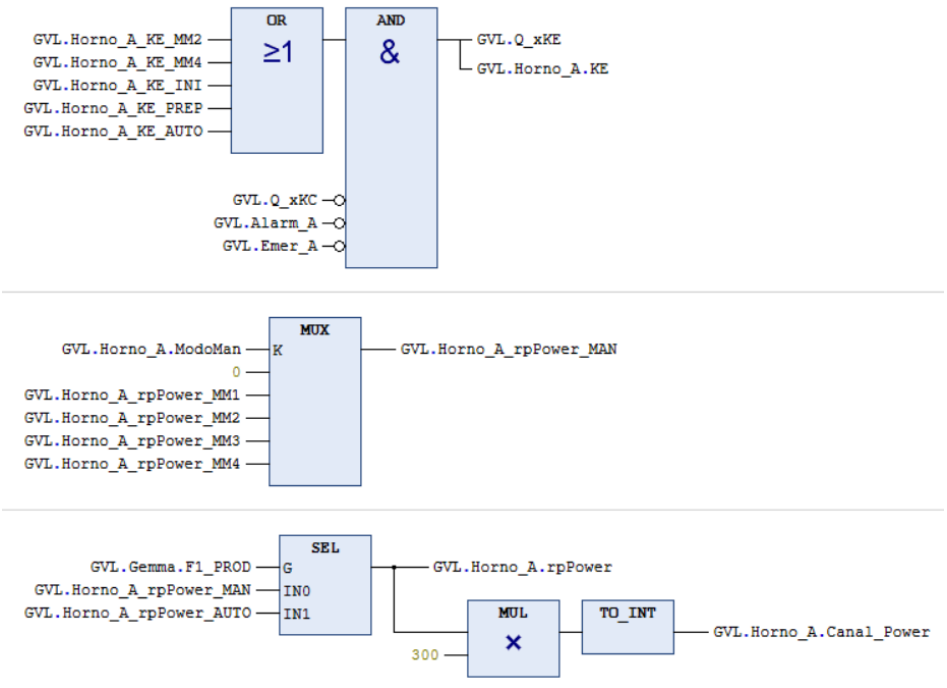

Figura 5.57 Fragmento de código en FBD de PRG_SALIDAS_ISA88 (III).

5.1.3- POUs para simulación del proceso

Antes de proceder al apartado gráfico (HMI), se van a comentar un par de POUs desarrolladas para la simulación del proceso y que incluyen algo de programación en el PLC virtual que ejecuta Codesys. Estos módulos de programación se encuentran bajo la carpeta SIMULACIÓN_ISA88 del árbol del proyecto (ver Figura 5.58).

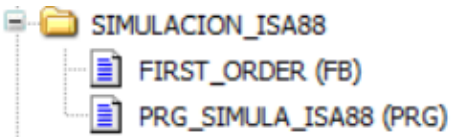

Figura 5.58 Estructura de la parte de programa destinada a simulación.

- **FIRST_ORDER**

Este bloque implementa el comportamiento de un sistema de primer orden que resulta muy habitual en el comportamiento dinámico de muchos procesos físicos, y en concreto para modelar la temperatura de un horno como el del ejemplo que se trata en este trabajo. Se utiliza la ecuación en diferencias resultante de la discretización de la función de transferencias de un sistema de primer orden con sus parámetros característicos: ganancia (K) y constante de tiempo (T).

Las variables de entrada y salida de este bloque y algunas variables locales declaradas se muestran en la Figura 5.59.

Scope	Name	Address	Data type	Comment
VAR_INPUT	**UK**		REAL	Entrada
VAR_INPUT	**K**		REAL	Ganancia del sistema
VAR_INPUT	**T**		REAL	Constante de tiempo de sistema en segundos
VAR_INPUT	**Tmms**		REAL	Periodo de muestreo en ms
VAR_OUTPUT	**YK**		REAL	Salida respuesta del sistema
VAR	**YK_1**		REAL	Salida anterior
VAR	**Tm**		REAL	
VAR	**a**		REAL	
VAR	**b**		REAL	

Figura 5.59 Listado de variables asociadas a la función FIRST_ORDER.

El desarrollo de las ecuaciones pertinentes, en texto estructurado (ST) es indica en la Figura 5.60.

```
(* Paso de periodo de muestreo a segundos *)

Tm := Tmms / 1000;

(* ECUACIÓN:    YK := a * UK + b * YK_1    *)

a := ( K * Tm ) /  (Tm + T ) ;
b :=  T /  (Tm + T ) ;

(* Cálculo de la salida *)
YK := a * UK + b * YK_1;

(* Actualización del valor anterior *)
YK_1 := YK;
```

Figura 5.60 Código en ST desarrollado para la función FIRST_ORDER.

La utilización de este módulo de programación ha sido fundamental para la programación, prueba y validación de los algoritmos de control PID en el control de temperatura del horno.

- **PRG_SIMULA_ISA88**

Finalmente, el programa que realiza, típicamente, la simulación de elementos sensores los diferentes módulos de equipamiento es PRG_SIMULA_ISA88, cuyas variables declaradas son únicamente las instancias de los bloques funcionales estándar utilizados (TON, R_TRIG y SR) y el bloque funcional FIRST_ORDER para la temperatura (ver Figura 5.61).

Scope	Name	Address	Data type	Comment
VAR	**TON_Alim_1_A0, TON_Alim_1_A1**		TON	Temporizadores para simular A0 y A1 (Alimentador)
VAR	**RT_Carga_1_FC0**		R_TRIG	Flanco ascendente FC0 para inicializar Peso (Cargador)
VAR	**RT_Alim_1_A1**		R_TRIG	Simula la carga de piezas cada vez que hay salida del alimentador
VAR	**TON_Horno_1_KCA, TON_Horno_1_KEA**		TON	Temporizadores para simular KCA y KEA (Horno)
VAR	**SR_TR**		SR	Biestable para simulación de la señal del termostato (TR)
VAR	**FIRST_ORDER_Temp**		FIRST_ORDER	

Figura 5.61 Listado de variables asociadas a PRG_SIMULA_ISA88.

El código para la representación de los elementos se ha generado en ST. Para el alimentador se han representado la simulación de los sensores que detectan el posicionamiento del cilindro (A0 y A1) como se ve representado en la Figura 5.62.

```
(* SIMULAR Detectores avance/retroceso cilindro ALIMENTADOR_A *)

IF GVL.Simula_Alim_1 THEN
    (* Simula A0 *)
    TON_Alim_1_A0 (IN := NOT GVL.Q_xAmas
                         AND NOT GVL.Error_Alim_1_A0,
                         PT := T#1S);
    GVL.Alimentador_A.A0 := TON_Alim_1_A0.Q;
    (* Simula A1 *)
    TON_Alim_1_A1 (IN := GVL.Q_xAmas
                         AND NOT GVL.Error_Alim_1_A1,
                         PT := T#1S);
    GVL.Alimentador_A.A1 := TON_Alim_1_A1.Q;
END_IF
```

Figura 5.62 Fragmento de código en ST asociado a la simulación del avance/retroceso del Alimentador.

Para el cargador se simula el movimiento de carga y descarga del cargador a efectos de visualización sobre la pantalla de operador, los sensores de indicación de posición del cargador (FC0 y FC1), el incremento de peso a medida que se realiza la carga de una pieza sobre el cargador y en el momento que se realiza la descarga de piezas del cargador tras el calentamiento, la puesta a 0 del peso tras la eliminación de las piezas del cargador. Un fragmento de lo descrito se puede ver en la Figura 5.63.

```
// SIMULACIÓN DEL CARGADOR

(* Calcula el movimiento en AVANCE del CARGADOR, a efectos de visualización, pero no modifica señales de sensores *)
IF GVL.Q_xKM1 THEN
    GVL.PosY_Carga_1 := GVL.PosY_Carga_1 - 5 ;
    IF GVL.Cargador_A.FC1
       OR GVL.PosY_Carga_1 < 0
       THEN
       GVL.PosY_Carga_1 := 0;
    END_IF
END_IF

(* Calcula movimiento en RETROCESO del CARGADOR, a efectos de visualización, pero no modifica señales de sensores *)
IF GVL.Q_xKM0 THEN
    GVL.PosY_Carga_1 := GVL.PosY_Carga_1 + 5 ;
    IF GVL.Cargador_A.FC0
       OR GVL.PosY_Carga_1 > 80
       THEN
       GVL.PosY_Carga_1 := 80;
    END_IF
END_IF

(* SIMULA las señales de los finales de carrera del CARGADOR, FC0 y FC1 *)
IF GVL.Simula_Carga_1 THEN
    GVL.Cargador_A.FC1 := GVL.PosY_Carga_1 = 0 AND NOT GVL.Error_Carga_1_FC1;
    GVL.Cargador_A.FC0 := GVL.PosY_Carga_1 = 80 AND NOT GVL.Error_Carga_1_FC0;;
```

Figura 5.63 Fragmento de código en ST asociado a la simulación de la subida/bajada y finales de carrera del Cargador.

El último fragmento de código es el desarrollado para la simulación de las señales de los sensores del horno. Se emula el cierre de la puerta del horno, los contactores de calentamiento y enfriamiento, la dinámica del proceso de calentamiento mediante el uso del FB que implementa un sistema de primer orden, la posibilidad de la inclusión de perturbaciones en el proceso de calentamiento del horno y la localización de los límites superior e inferior de la temperatura en el horno. Un fragmento del código desarrollado para lo descrito se puede visualizar en la Figura 5.64.

```
(* Simulación de señales para el HORNO *)

IF GVL.Simula_Horno_1 THEN
    // Puerta del horno cerrada con el cilindro del cargador en avance.
    GVL.Horno_A.HC := GVL.Cargador_A.FC1 ;

    (* Simula KCA *)
    TON_Horno_1_KCA (IN := GVL.Horno_A.KC AND NOT GVL.Error_Horno_1_KCA,
                     PT := T#1S);
    GVL.Horno_A.KCA := TON_Horno_1_KCA.Q;
    (* Simula KEA *)
    TON_Horno_1_KEA (IN := GVL.Horno_A.KE AND NOT GVL.Error_Horno_1_KEA,
                     PT := T#1S);
    GVL.Horno_A.KEA := TON_Horno_1_KEA.Q;

    // Detecta carga en el Horno (CG="1")
    GVL.Horno_A.CG := GVL.Horno_A.HC AND GVL.Cargador_A.FC1 AND (GVL.Cargador_A.Peso > 1.0) ;

    (* Simulación temperatura del Horno: Sma de primer orden con KC y un supuesto driver de potencia: rpPower (0-100%)*)
    // La salida es el valor del canal analógico (valor del sensor conectado al controlador
    FIRST_ORDER_Temp ( UK:= GVL.Horno_A.rpPower, K:=GVL.Horno_A.K, T:=GVL.Horno_A.T, Tmms:=GVL.Horno_A.Tmms) ;
    GVL.Horno_A.Canal_Temp := REAL_TO_INT ( FIRST_ORDER_Temp.YK * 100 );
    (* El enfriamiento se simula linealmente, bien con "KV" o el botón "HMI_Perturbacion" *)
    IF GVL.Horno_A.KE THEN // Con refrigeración se acelera la bajada de temperatura de forma lineal
        GVL.Horno_A.Canal_Temp := GVL.Horno_A.Canal_Temp - 400;
    END_IF
```

Figura 5.64 Fragmento de código en ST asociado a la simulación del proceso del Horno.

5.1.4- Pantalla de explotación y supervisión (HMI)

En cuanto a los apoyos a nivel gráfico que se han desarrollado para la operación del sistema en Codesys, se distinguen cuatro pantallas de explotación y control distintas, realizadas con las herramientas disponibles por defecto en la aplicación. Como ya se introdujo ligeramente en este documento, las pantallas tienen su base de diseño en el estándar ISA 101 buscando disponer unas pantallas de operador con elementos gráficos que sean fáciles de entender, no especialmente llamativos y donde no resulte confusa su operación.

El primer elemento gráfico que se describe es aquel que permite la visualización del control de temperatura y todo el conjunto de datos asociados al sistema de primer orden que simula la dinámica de calentamiento y enfriamiento del horno. La pantalla asociada a esta descripción se observa en la Figura 5.65.

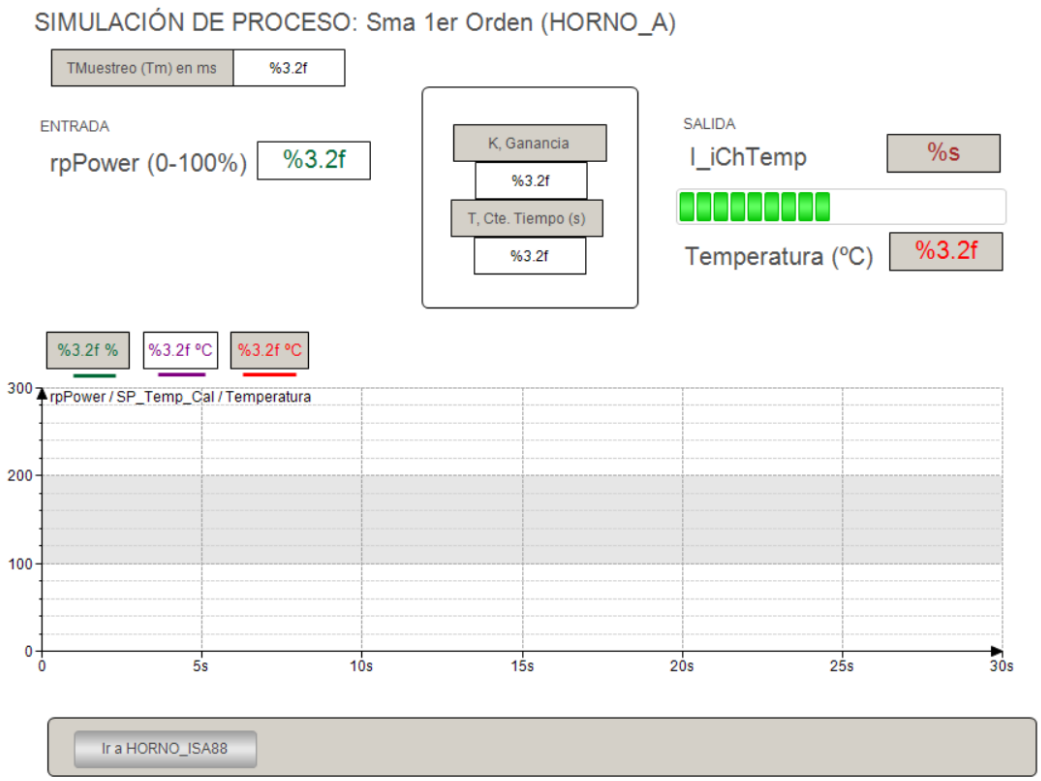

Figura 5.65 Pantalla en la que se visualiza el control de temperatura y datos al proceso de calentamiento y enfriamiento del Horno.

La siguiente pantalla (Figura 5.66) se utiliza para representar los estados del GDMMA indicando en cual se encuentra el sistema en cada momento.

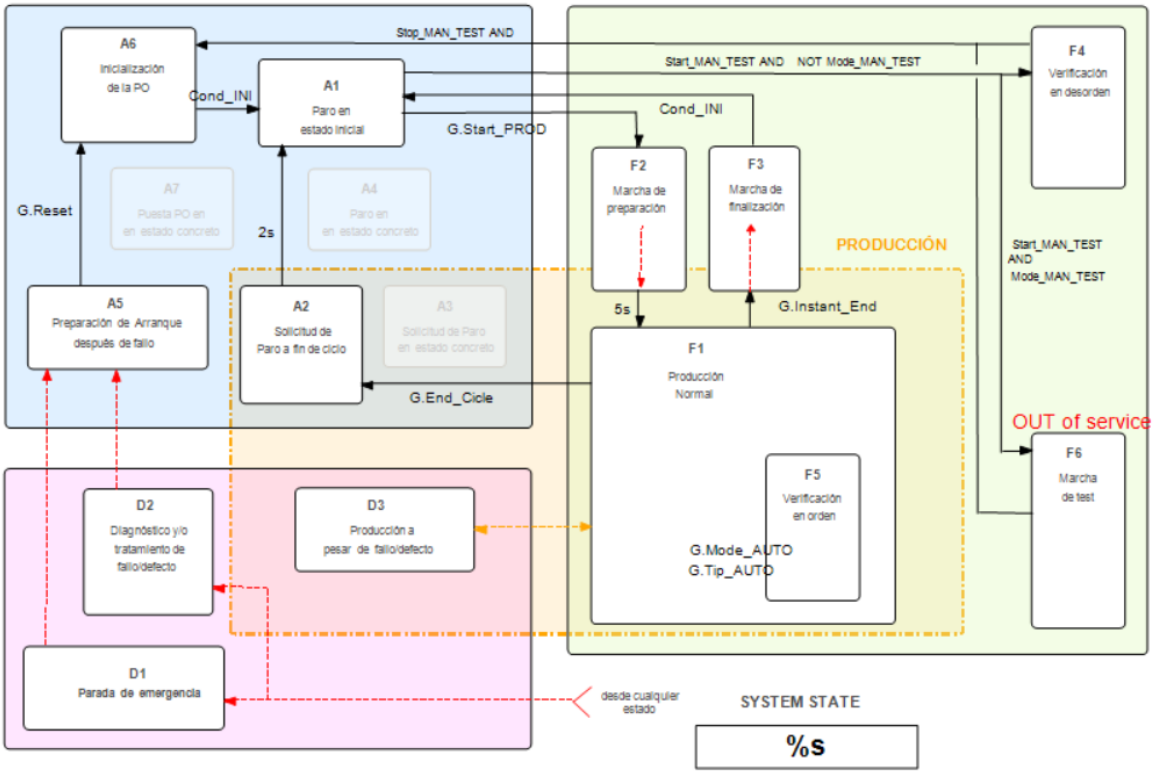

Figura 5.66 Pantalla en la que se muestra el GDMMA asociado al sistema.

La siguiente ventana de operación significativa es la que permite el control del sistema cuando este se encuentra en el modo de verificación en desorden. Parte de la pantalla desarrollada para esta tarea, se puede ver en la Figura 5.67.

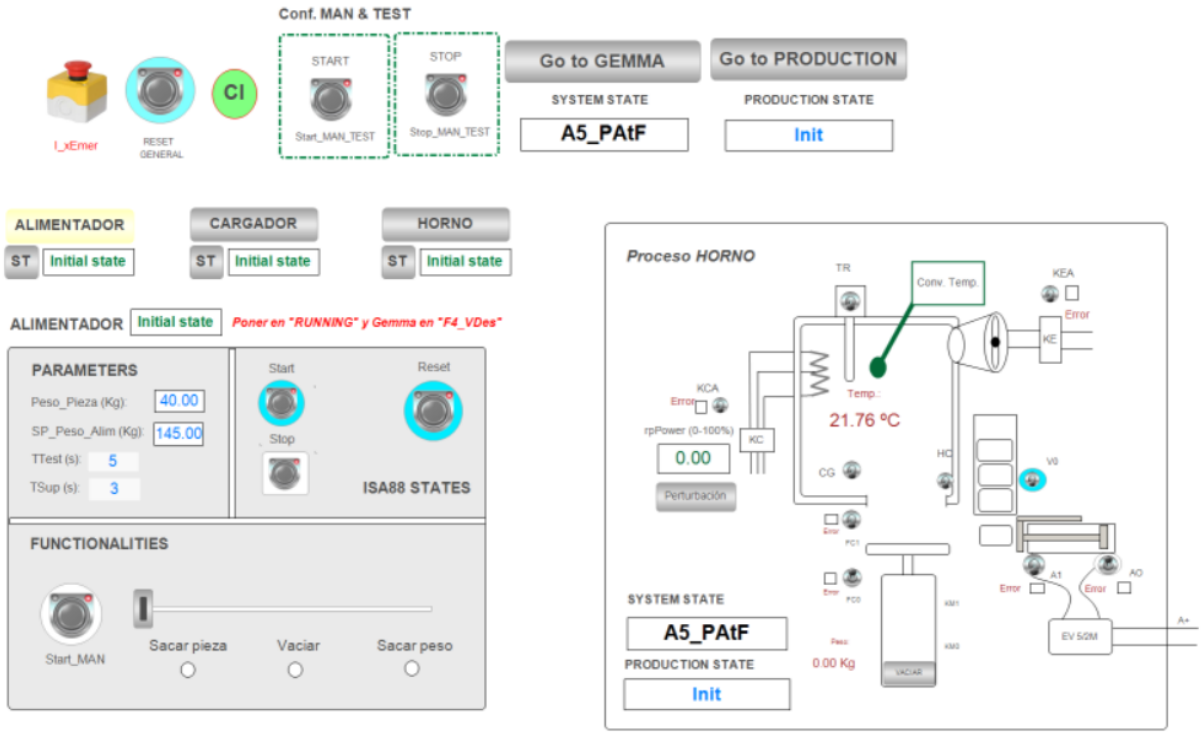

Figura 5.67 Pantalla asociada al estado de verificación en desorden.

Finalmente se ha desarrollado una pantalla dedicada a elaborar la receta de producción y observar la operación de la planta en este modo. La Figura 5.68 representa esta pantalla de explotación del sistema.

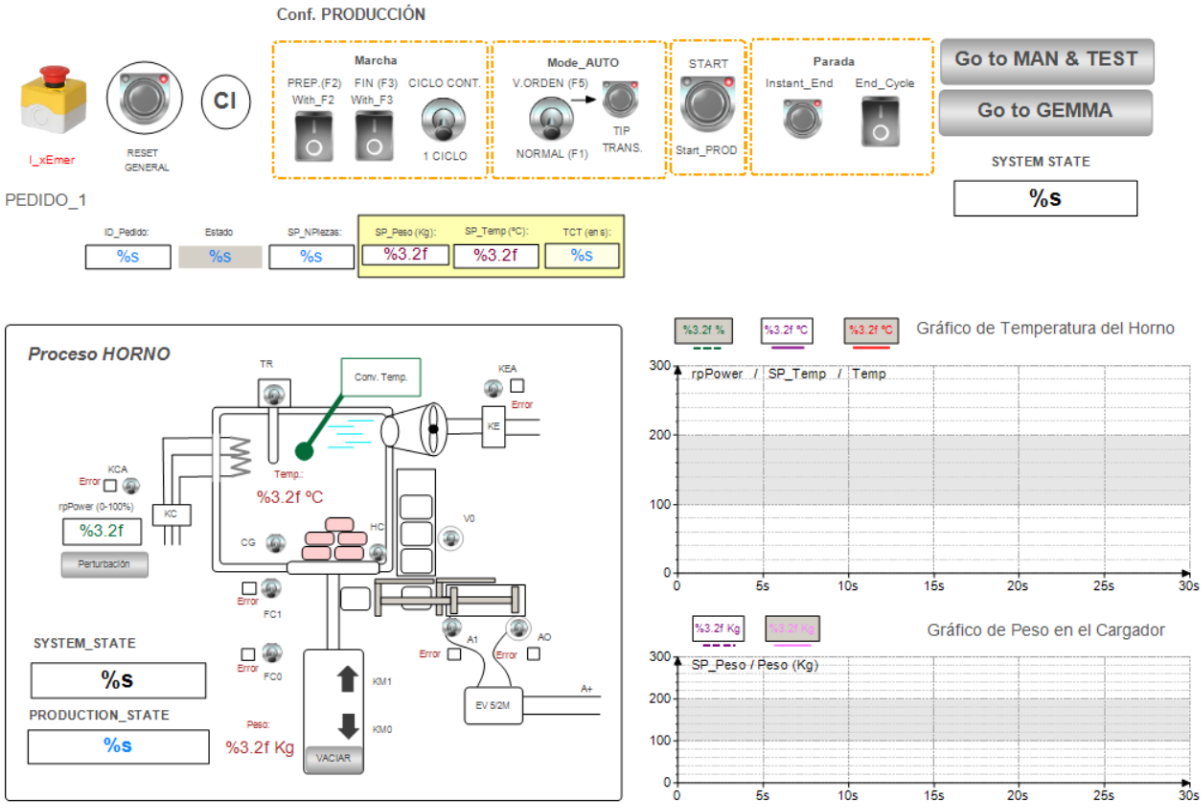

Figura 5.68 Pantalla correspondiente al modo producción.

Indicar también que estando en cualquiera de las 3 últimas pantallas mencionadas (GDMMA, verificación en desorden y la de producción) en todas ellas existe la posibilidad de navegar ente ellas mediante el uso de los botones que aparecen en la Figura 5.69, a la par que siempre se estaría visualizando el estado del GDMMA y el de producción en el que se encuentra el sistema.

Figura 5.69 Botones de navegación entre las pantallas descritas.

Finalmente, comentar también, que en cualquiera de estas tres pantallas mencionadas se encontrará el estado ISA88 de cada uno de los módulos de equipamiento como se puede

ver en la Figura 5.70, donde también aparecen mensajes que será útiles para conocer situaciones de aviso, alarma, defecto o emergencia para una operación más optimizada y segura.

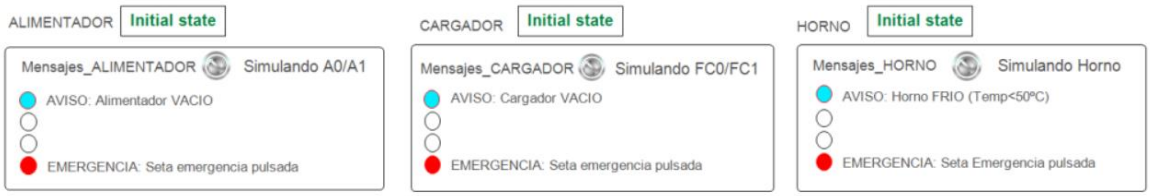

Figura 5.70 Indicación de los estados ISA88 y mensajes complementarios de cada uno de los módulos de equipamiento en las pantallas descritas.

5.2.- PROGRAMACIÓN EN PLCNEXT ENGINEER

Hasta el momento se ha explicado detalladamente el desarrollo realizado para el programa de control y supervisión mediante Codesys en modo simulación tanto en la parte PLC virtual que implementa esta herramienta como con la inclusión de código adicional para simulación del proceso.

La siguiente tarea a realizar es la implantación de la aplicación similar a la anterior en un controlador real, y particularmente en un AXC F 2152 basado en tecnología PLCnext de Phoenix Contact lo cual permite incorporar prestaciones IoT integradas en el propio sistema. Estos equipos se programan mediante la herramienta PLCnext Engineer que implementa los lenguajes de programación del estándar IEC 61131-3 en la parte PLC y un servidor web integrado que puede configurarse para el desarrollo de interfaces de usuario.

Considerando que el paso del código desarrollado para el entorno Codesys a PLCnext Engineer podría significar una carga importante de trabajo se planteó inicialmente mantener el código en Codesys y realizar una comunicación vía Modbus TCP entre la aplicación en Codesys (esclavo) y PLCnext actuando como cliente (maestro). Si bien esta podría ser una arquitectura correcta, no aprovecha todas las prestaciones de PLCnext para control en tiempo real pudiendo convivir la parte PLC con otras funcionalidades que en los PLCs

convencionales no existen y tienen que recurrir a equipos externos para incrementar en prestaciones IoT.

Si bien se llevaron a cabo algunas pruebas para la comunicación Modbus entre ambos (aplicación Codesys y CPU de PLCnext), las de pruebas de adaptación del programa Codesys a PLCnext Engineer se desenvolvía de forma más solvente y rápida de lo previsto por lo que se realizó la conversión completa del código para su implantación al controlador PLCnext.

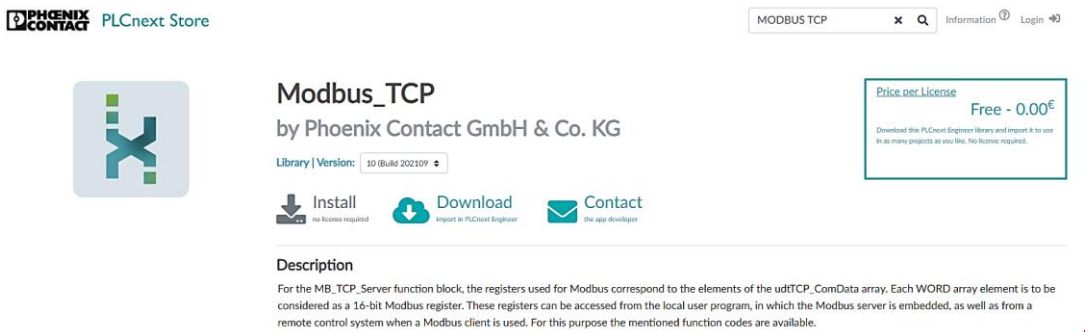

Figura 5.71 Librería de Modbus_TCP para PLCnext Engineer.

Salvo en un par de puntos importantes a tener en cuenta, que Codesys incorpora como elementos fuera del estándar IEC 61131-3, la herramienta PLCnext Engineer ha demostrado cumplir al 100% con el estándar de programación de PLCs. Estos dos aspectos son:

- En PLCnext Engineer no se puede utilizar el lenguaje CFC, mientras que en Codesys sí, por lo que hubo que adaptar esos programas.

- Para la sincronización de los SFCs, la variable SFCreset que existe en Codesys, no tiene una implementación tan directa en PLCnext Engineer y hubo que buscar un método de realización de esa inicialización en la codificación de los Grafcets.

El resto del código desarrollado en Codesys tuvo una transcripción prácticamente directa a PLCnext Engineer.

5.2.1- Programa de control en PLCnext Engineer

Se inicia la explicación partiendo de un esquema general de todos los elementos del programa y su preparación para que se ejecuten en el controlador PLCnext AXC F 2152.

En PLCnext Engineer, la disposición de generación de programa cambia un poco respecto a la generada y vista en Codesys. En este caso la realización del código se lleva primeramente en el apartado Programming visto en la Figura 5.72, situado bajo el directorio COMPONENTS -> Programming en la ventana derecha de la aplicación.

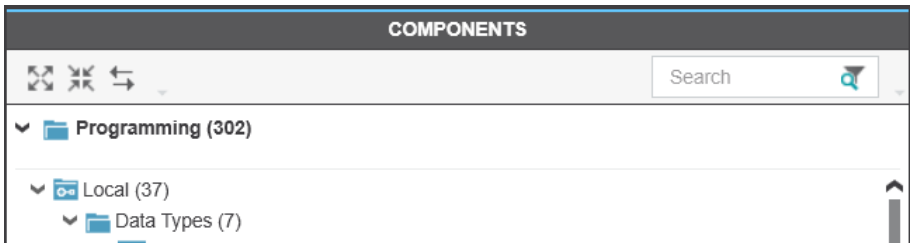

Figura 5.72 Localización de la zona asociada a la programación en PLCnext Engineer.

En este caso, se distribuye la disposición en función del tipo de elemento de programación creado: 1) Estructuras de datos, 2) bloques de función y 3) programas. Si bien es cierto que dentro de cada una de estas categorizaciones se pueden hacer agrupaciones estructurales internas como se verá para los bloques de función (que se dividirán en generales y los de los diferentes módulos de equipamiento) en Codesys ya se podía realizar una integración directa dentro de las categorías creadas externamente por el usuario.

5.2.1.1 Estructuras de datos

Comenzando en orden, lo primero sería comenzar por las estructuras de datos generadas para el programa y que van a integrar gran parte de las variables necesarias para el desarrollo de la aplicación.

En PLCnext Engineer, las estructuras de datos se encuentran localizadas todas bajo el apartado Data Types (COMPONENTS -> Programming -> Local -> Data Types) como se puede observar en la Figura 5.73.

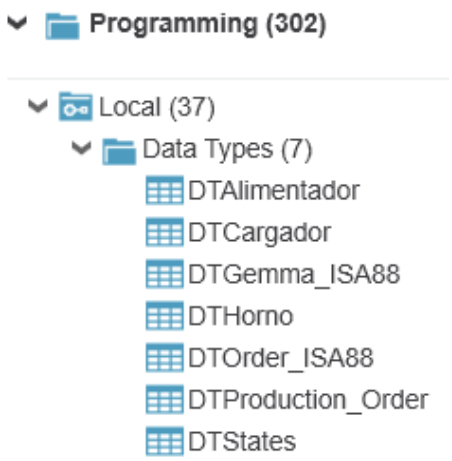

Figura 5.73 Estructuras de datos implementadas en PLCnext Engineer.

En cuanto al código asociado a estas estructuras es muy semejante tanto en el desarrollo generado para Codesys, como para el desarrollado en PLCnext Engineer más allá de pequeños detalles como el que no se pueden declarar varios parámetros separados por comas como se observa en la Figura 5.74.

```
TYPE DTStates_ISA88 :
STRUCT
    // Estados ISA88
    Init : BOOL := TRUE;
    Running, COMPLETED: BOOL;       // Init == Idle (Initial State)
    Holding, Held, Restarting: BOOL;
    Pausing, Paused: BOOL;
    Stopping, STOPPED: BOOL;
    Aborting, ABORTED: BOOL;

    // Según Transiciones ISA-88
    Start, Hold, Restart, Pause, Resume: BOOL;
    Stop, Abort, Reset: BOOL;
    End_Running, End_Holding, End_Restarting,
    End_Pausing, End_Stopping, End_Aborting: BOOL;
```

Figura 5.74 Código de estructura de datos en Codesys.

Los parámetros en PLCnext Engineer han de ser entonces declarados de manera individual como se puede captar en la Figura 5.75.

```
TYPE
    DTStates_ISA88: STRUCT
// Estados ISA88
    Init : BOOL := TRUE; // Init == Idle (Initial State)
    Running: BOOL; COMPLETED: BOOL;
    Holding: BOOL; Held:BOOL;  Restarting: BOOL;
    Pausing: BOOL; Paused: BOOL;
    Stopping: BOOL; STOPPED: BOOL;
    Aborting: BOOL; ABORTED: BOOL;

    // Según Transiciones ISA-88
    Start: BOOL; Start_LOC: BOOL; Start_REM: BOOL; //***
    Hold: BOOL; Restart: BOOL;
    Pause: BOOL; Pause_LOC: BOOL;   Pause_REM: BOOL; //***
    Resume: BOOL; Resume_LOC: BOOL; Resume_REM: BOOL; //***
    Stop: BOOL; Stop LOC: BOOL; Stop REM: BOOL; //***
```

Figura 5.75 Código de estructura de datos en PLCnext Engineer.

Por lo demás, la generación de la estructura es esencialmente de la misma forma que en Codesys y el mecanismo de acceso a las variables declaradas del tipo estructura también se hace de forma idéntica en PLCnext Engineer.

5.2.1.2 Variables globales

Para la generación de las variables globales, estas han de ir localizadas dentro del controlador que se introdujo al programa previamente en el apartado de configuración de este documento. Las variables se localizarían en el siguiente apartado: PLANT -> Project -> axcf2152 : AXC F 2152 -> Data List como se puede visualizar en la Figura 5.76.

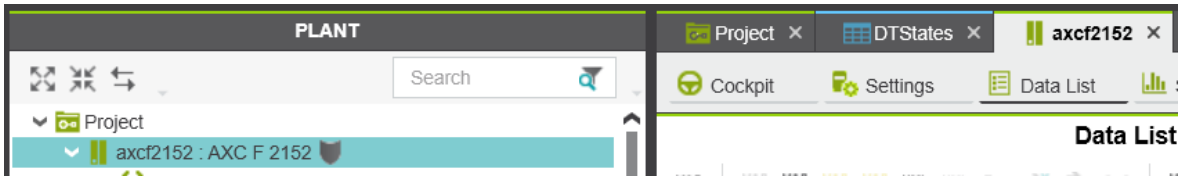

Figura 5.76 Acceso al apartado de datos del controlador AXC F 2152.

En cuanto a la declaración de las variables se lleva a cabo de manera semejante a la declaración hecha en Codesys. Por ejemplo, se ve representada en la Figura 5.77 el fragmento de las variables globales en las que se puede contemplar la variable declarada para acceso a los estados ISA88 de los módulos de equipamiento según se han definido.

Variable (PLC)	Type	Usage
Horno_A_KC_MM1	BOOL	Global
Horno_A_KC_AUTO	BOOL	Global
Horno_A	DTHorno...	Global
HMI_STATUS2	HMI_STA...	Global
HMI_CONTROL	HMI_CO...	Global
Gemma1	DTGemma	Global
Gemma	DTGemma	Global
Fin_AUTO	BOOL	Global
Estados_ISA88	DTStates...	Global
ESM_DATA	ESM_DAT	Global

Figura 5.77 Fragmento de variables globales en PLCnext Engineer.

5.2.1.3 Bloques de función y funciones

Los bloques de función y funciones sí que ya presentan ciertas particularidades con dos tipos de lenguajes de programación utilizados. La primera adaptación relativamente importante que se tuvo que realizar viene dada en los bloques de función asociados a la supervisión de cada módulo de equipamiento que habían sido desarrollados en CFC. Puesto que este lenguaje no está disponible en PLCnext Engineer, se ha adaptado a ST.

La segunda adaptación que se ha tenido que realizar ha sido para el control de los SFC. Puesto que el SFCreset no se puede realizar como en Codesys, hubo que implementar una especie de función de control asociada a esos bloques de función teniendo en cuenta la documentación proporcionada por PLCnext Engineer en este aspecto tan particular.

Pero todo esto se va a intentar ir analizando con el orden que se podría observar dentro del programa de PLCnext Engineer. Comenzando entonces con los bloques de

función y funciones generales desarrollados, que se pueden visualizar en la Figura 5.78. Aquí se han incluido todos ellos en la carpeta de FBs_y_Fs_GENERALES en lugar de distribuirlos en otras como aparecía en el árbol del proyecto Codesys, pero exactamente son las mismos POUs en ambos casos.

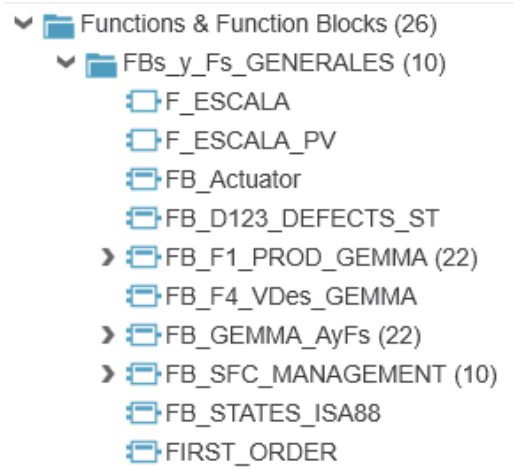

Figura 5.78 Distribución de funciones y bloques de función en PLCnext Engineer.

Se pueden apreciar unas funciones que previamente no se encontraban en la carpeta de organización general, siendo estas las de escalado y las de simulación del primer orden. Las dos funciones ya vistas en Codesys de escala (tanto F_ESCALA, como F_ESCALA_PV), transcripción es completamente directa, lo mismo sucede para la transcripción el sistema de primer orden generado para el control de temperatura (FIRST_ORDER). Es decir, en lo que se diferencian es la localización dentro del programa.

Tras describir funciones y bloques de función generales, tocaría dar paso a los ya bien conocidos módulos de equipamiento. En este caso, se procedió de una manera semejante a la del capítulo anterior destinado a la programación en Codesys 3.5. Se encuentran localizados en tres capetas de organización distintas, asociadas cada una a su propio módulo de equipamiento. Las carpetas, son casi prácticamente iguales, la única diferenciación a nivel externo que se sufre es que la estructura de datos asociada al módulo de equipamiento, ahora se encuentra fuera de esa carpeta.

Recordando que los módulos de equipamiento están tomados con una misma premisa de diseño, se procederá a analizar uno en más detalle, sobre todo porque aquí es donde uno de los bloques de función (el asociado a la supervisión de los estados de defecto, aviso, alarma, emergencia) ha tenido que adaptarse de CFC a lenguaje ST.

Siguiendo el desarrollo mostrado ya en el capítulo anterior, se tomará como ejemplo el módulo de equipamiento desarrollado para el Alimentador. La carpeta que se asocia a este módulo de equipamiento recibe el nombre de: ME_1_DOSER_ISA88 y su estructuración puede ser vista en la Figura 5.79.

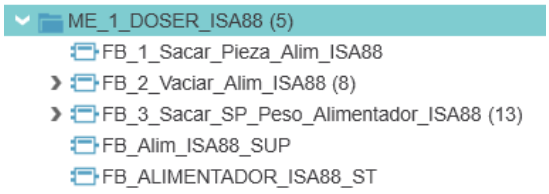

Figura 5.79 Distribución del módulo de equipamiento: Alimentador en PLCnext.

Se observa que la estructuración es la misma que en Codesys. Los servicios vienen representados por los bloques de función: FB_1_Sacar_Pieza_Alim_ISA88 (sacar una pieza), FB_2_Vaciar_Alim_ISA88 (vaciar el alimentador hasta que no haya piezas) y FB_3_Sacar_SP_Peso_Alimentador_ISA88 (sacar un número de piezas en función del peso estipulado por el usuario). Los servicios se ha realizado una transcripción directa puesto que el servicio asociado a sacar una pieza había sido desarrollado en FBD, en PLCnext Engineer, también. Los otros dos servicios de vaciar alimentador y sacar un peso especificado de piezas, desarrolladas previamente en SFC, en PLCnext Engineer también.

Luego, el bloque de función asociado a la gestión de los estados ISA88 del módulo de equipamiento, denominado: FB_ALIMENTADOR_ISA88_ST. Este bloque había sido desarrollado previamente en Codesys y para el caso de PLCnext Engineer no se produce excepción, sigue siendo desarrollado en ST.

Por lo tanto, el último que queda comentar es el asociado a la supervisión (defecto, aviso, alarma, emergencia) de los módulos de equipamiento referido con el nombre: FB_Alim_ISA88_SUP. Previamente este había sido desarrollado en CFC. Sin embargo, en PLCnext Engineer, no existe la posibilidad de emplear CFC. Toca seleccionar entonces un lenguaje alternativo entre los disponibles (LD, FBD o ST). La decisión queda dividida entre seleccionar FBD igual por su semejanza visual o ST. Se concreta que se va a emplear ST de cara a resultar más sencillo su manejo. El código entonces adaptado a ST, quedaría como se refleja en la Figura 5.80.

```
// En Codesys estaba en CFC y aquí se pasa a ST

// Aviso
TON_Sup_V0 (IN:= NOT Alim.V0, PT:= TO_TIME (Alim.TSup*1000));
Alim.STATES.Aviso := TON_Sup_V0.Q;
OUT_xAviso:= TON_Sup_V0.Q;
Alim.STATES.msgAviso := SEL (Alim.STATES.Aviso, '  ', 'WARNING: Dosser empty');

// Alarma
TON_Sup_Amas (IN:= Alim.Amas AND NOT Alim.A1, PT:= TO_TIME (Alim.TSup*1000));
SR_Alarm (SET1:=TON_Sup_Amas.Q, RESET:=Alim.STATES.Reset);
Alim.STATES.Alarma:= SR_Alarm.Q1;
OUT_xAlarm:=SR_Alarm.Q1;
Alim.STATES.msgAlarma := SEL (Alim.STATES.Alarma, '  ', 'ALARM: Cilinder fault (A1)');


// Emergencia
SR_Emer (SET1:=NOT Alim.IxEmer, RESET:=Alim.STATES.Reset);
Alim.STATES.Emer:=SR_Emer.Q1;
OUT_xEmer:=SR_Emer.Q1;
Alim.STATES.msgEmer := SEL (Alim.STATES.Emer, '  ', 'EMERGENCY: Stop button pressed');
```

Figura 5.80 Código de CFC adaptado a ST en PLCnext Engineer para la gestión de defecto, alarma, aviso y emergencia.

Descrito el módulo de equipamiento del Alimentador, de igual manera se ve reflejado en los otros dos módulos de equipamiento. El Cargador quedaría estructurado como se muestra en la Figura 5.81.

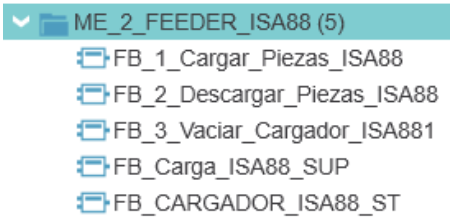

Figura 5.81 Distribución del módulo de equipamiento: Cargador en PLCnext.

Quedaría por ver el último módulo de equipamiento asociado al Horno, el cual se ve reflejado en la Figura 5.82.

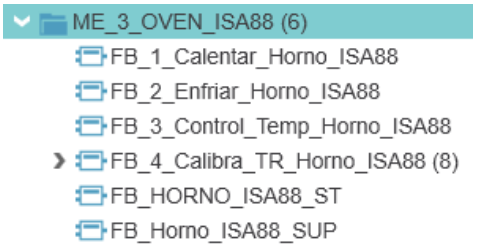

Figura 5.82 Distribución del módulo de equipamiento: Horno en PLCnext.

5.2.1.4 Programas generales del sistema en PLCnext Engineer

Para finalizar, antes de comentar unas pequeñas particularidades a tener muy en cuenta cuando se trabaje con según qué elementos de PLCnext Engineer, quedaría por comentar los programas asociados al desarrollo. Los programas se encontrarán localizados bajo el apartado PRGs_GENERALES (COMPONENTS -> Programming -> Programs -> PRGs_GENERALES) como se puede observar en la Figura 5.83.

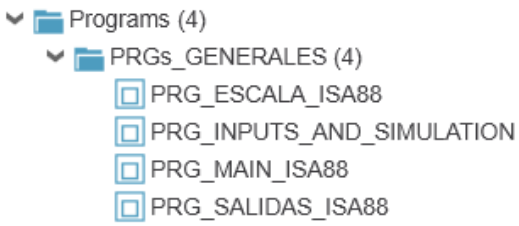

Figura 5.83 Distribución de programas en PLCnext Engineer.

Los programas, también han tenido una transcripción a PLCnext Engineer prácticamente directa. Los programas que habían sido desarrollados en FBD, los cuales están identificados como: PRG_ESCALA_ISA88, PRG_MAIN_ISA88 y PRG_SALIDAS_ISA88, no sufren mayor variación y la transcripción en directa.

Sin embargo, se puede percibir un programa que no estaba desarrollado previamente como tal (antes se separaba en dos independientes), es el referido como PRG_INPUTS_AND_SIMULATION. Este programa es el que se identifica con los PRG_ENTRADAS_ISA88 y PRG_SIMULA_ISA88, los cuales se juntaron en el anterior mencionado. El desarrollo se mantiene en ST.

Antes de proceder con los dos mayores problemas encontrados durante el paso del código a PLCnext Engineer. Sería conveniente comentar como realizar la incorporación de los programas al controlador físico. Para ello hay que incorporar los programas a dos tareas de programa (uno para el ciclo de funcionamiento principal y otro para la simulación). Para poder incorporar estos programas, hay que ir al apartado de Task and Events (tareas y eventos) (PLANT -> Project -> axcf2152 : AXC F 2152 -> PLCnext -> Task and Events) como se visualiza en la Figura 5.84.

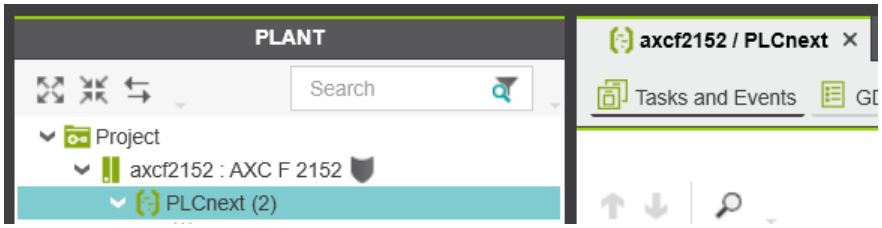

Figura 5.84 Acceso a la configuración de tareas y eventos para el controlador PLCnext.

Dentro de este apartado, se crearán dos tareas que se ejecutarán de manera cíclica. Una denominada: Main_Cyclic, al que se le asociarán los programas: PRG_ESCALA_ISA88, PRG_MAIN_ISA88 y PRG_SALIDAS_ISA88 y otro denominado: Simulation_task, al que se le asociará PRG_INPUTS_AND_SIMULATION. Esto se puede ver representado sobre la Figura 5.85.

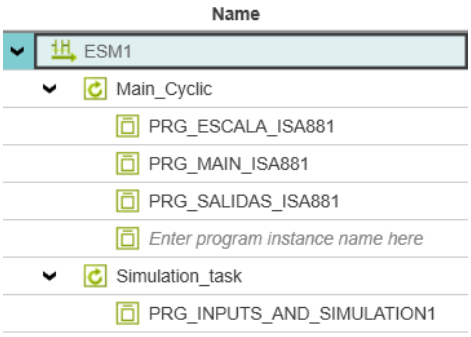

Figura 5.85 Configuración de programas en el controlador PLCnext.

5.2.1.5 Particularidades con la programación de Codesys y PLCnext Engineer

Si bien es cierto que el paso del entorno de programación de Codesys a PLCnext ha sido bastante fluido y no ha habido que realizar muchos ajustes o cambios al código original desarrollado. Hay dos situaciones que se han presentado, que son de interés a comentar.

- **Cambio en el uso de las variables asociadas a estructuras de datos de los módulos de equipamiento**

La primera se ha presentado cuando se han generado variables en los bloques de función asociadas a estructuras de datos de los módulos de equipamiento. Generalmente, se asignaría como una variable de lectura (Input). Sin embargo, tomando como ejemplo la asignación a la estructura de datos generada para el Alimentador, se presenta el siguiente error mostrado en la Figura 5.86.

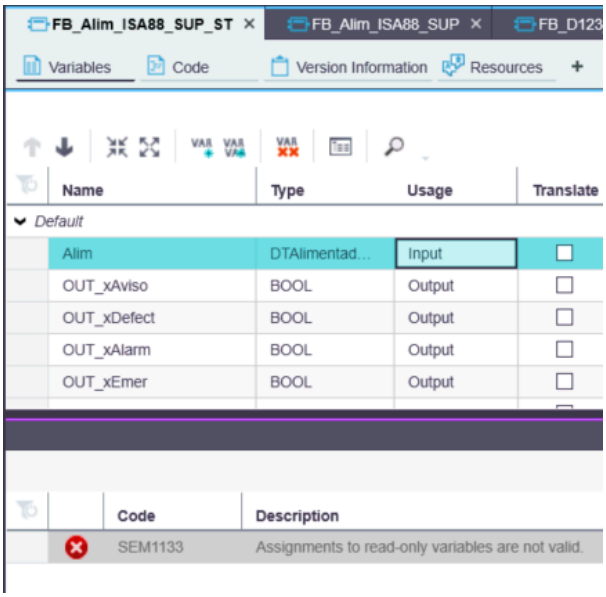

Figura 5.86 Error generado por declarar la variable asociada a la estructura de datos del módulo de equipamiento como una entrada.

Para solucionarlo, bastará simplemente con cambiar la variable a una de lectura y escritura (InOut) como se muestra en la Figura 5.87, haciendo que desaparezca el error.

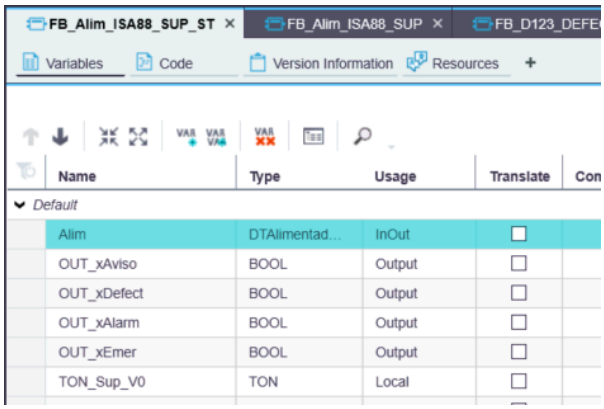

Figura 5.87 Declaración de la variable asociada a la estructura de datos del módulo de equipamiento como variable de entrada y salida.

- **Manejo de SFCs en PLCnext**

Pese a que se ha mantenido el uso de los SFCs al igual que se había implementado en el programa de Codesys. Los SFCs de PLCnext Engineer, no presentan un uso parejo a los mismos. Lo primero de todo, es que como ya se ha mencionado, requieren de una licencia externa para su uso y segundo, a pesar de que la implementación se ha buscado seguir la misma, el modo de operación de estos varía frente a Codesys.

Los detalles de cómo manejar SFCs en PLCnext se encuentran en el documento de la ayuda: *SFC Parameters, Operating Modes and Data Types.pdf*.

La llamada a un bloque funcional que se ha creado en SFC se realiza, o bien de forma expandida o comprimida. En el primer caso, todos los parámetros formales son identificables según se referencia en la Figura 5.88.

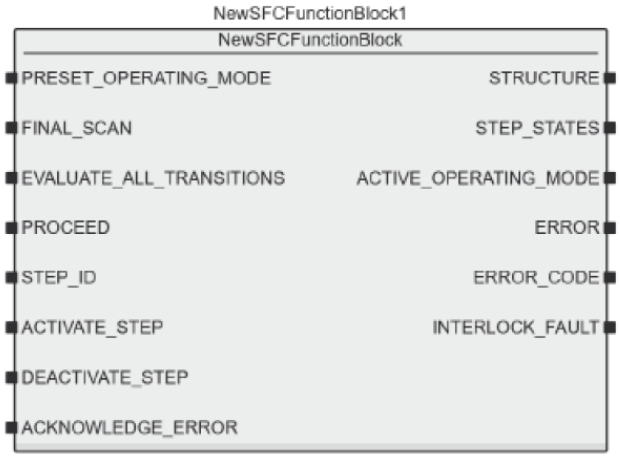

Figura 5.88 Llamada a un bloque funcional en SFC de forma expandida.

Para activar el modo de funcionamiento deseado del SFC hay que tener en cuenta que existe, por defecto, un tipo de dato ENUM denominado: SFC_OPERATING_MODE con los siguientes campos mostrados en la Figura 5.89.

```
TYPE
      SFC_OPERATING_MODE: (SFC_OPERATING_MODE_AUTO,
                           SFC_OPERATING_MODE_STEP,
                           SFC_OPERATING_MODE_STEP_FORCED,
                           SFC_OPERATING_MODE_HALT) ;
END_TYPE
```

Figura 5.89 SFC_OPERATING_MODE para activar el modo de funcionamiento deseado en SFC.

Se han declarado, por tanto, variables globales de este tipo para cada SFC de interés en su control como se puede contemplar en la Figura 5.90.

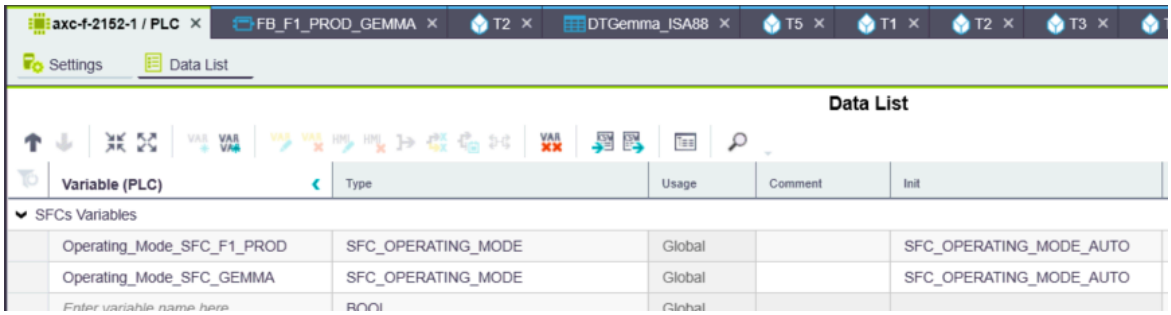

Figura 5.90 Variables globales creadas para activar el modo de funcionamiento del SFC.

La explicación de cada uno de los modos de funcionamiento se extrajo del manual de ayuda proporcionado por PLCnext Engineer dentro del programa y se muestra en la siguiente Figura 5.91.

Operating mode	Description
Automatic mode SFC_OPERATING_MODE_AUTO in Enum	The automatic mode is the initial mode after a cold/warm restart. In automatic mode, the SFC chart is evaluated normally, i.e., the steps become active/inactive and the actions are executed according to their qualifiers according to the evaluation of the transitions and depending on the states of Interlocks, if attached to the steps.
Step mode SFC_OPERATING_MODE_STEP in Enum	In step mode, the succeeding step of the SFC program is only executed if the relevant transition condition is fulfilled **and** a rising edge is detected at the 'PROCEED' input of the function block instance.
Forced (unconditional) mode SFC_OPERATING_MODE_STEP_FORCED in Enum	In forced mode, the succeeding step of the SFC program is executed if a rising edge is detected at the 'PROCEED' input of the function block instance. This happens **independently** of the transition value.
Halt mode SFC_OPERATING_MODE_HALT in Enum	In halt mode the transitions do not change the state of the SFC chart. The currently active steps are not executed and the assigned actions are not triggered. The halt mode is activated in case of an SFC error. In halt mode, a step can be activated/deactivated via the STEP_ID input (see below).

Figura 5.91 Explicación de cada uno de los modos de funcionamiento en la ayuda de PLCnext Engineer.

Por defecto está en modo AUTO ejecutando normalmente el SFC.

Los modos STEP y STEP_FORCED son equivalentes al uso de los parámetros de control del SFC en Codesys.

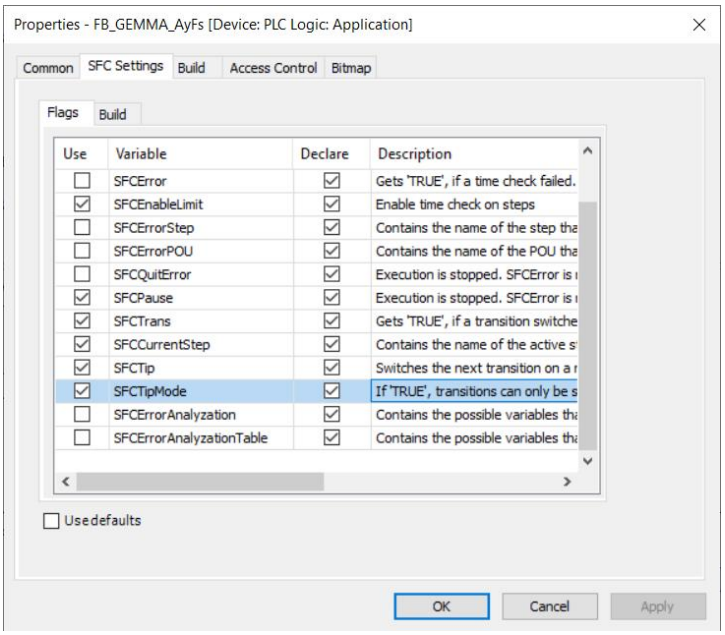

Figura 5.92 Configuración de propiedades asociadas al SFC de un bloque de función en Codesys.

SFCTip	Cambia a la siguiente transición en un flanco de subida.
SFCTipMode	Si recibe TRUE, las transiciones solo pueden ser cambiadas por SFCTip.

Tabla 5.1 Variables SFCTip y SFCTipMode de SFC y descripción en Codesys

PLCnext Engineer usa la variable de entrada PROCEED para avanzar, como la SFCTip en Codesys. Por otro lado, en función del estado de SFCTipMode:

'TRUE'	Es como activar el modo STEP en PLCnext Engineer.
'FALSE'	Semejante a activar el modo STEP_FORCED en PLCnext Engineer.

Tabla 5.2 Equivalencia de estado SFCTipMode con modos de funcionamiento de SFC en PLCnext Engineer.

El modo HALT deja el SFC "*congelado*" sin que se ejecuten las acciones asociadas a la etapa en la que se encuentre. Esencialmente es equivalente a SFCPause en Codesys,

pero, además, solo en este modo es posible llevar el SFC hasta alguna etapa concreta –si se hace a la etapa inicial, en realidad se inicializa o resetea el SFC.

Por tanto, otras entradas de interés para poder resetear o inicializar el SFC en PLCnext Engineer, es utilizar las entradas: STEP_ID, ACTIVATE_STEP y DEACTIVATE_STEP. Pero para ello según parece debe estar el SFC en modo HALT, como se ha comentado. La explicación de cada una de estas entradas proporcionada por la ayuda de PLCNext Engineer, se muestra en la siguientes Figura 5.93 y Figura 5.94.

STEP_ID	DINT	While the SFC chart is in halt mode (SFC_OPERATING_MODE_HALT), this input can be used to activate or deactivate a particular step. To **activate** a particular step: apply the corresponding step ID to this input and then switch the ACTIVATE_STEP input to TRUE. To **deactivate** a particular step: apply the corresponding step ID to this input and then switch the DEACTIVATE_STEP input to TRUE. After a successful activation/deactivation, the STEP_ID input is automatically set to the value -1. Otherwise the step number is kept. ***NOTE:*** *Alternatively, the ACTIVATE(DINT stepId) or DEACTIVATE(DINT stepId) methods of the SFC FB can be called.*
ACTIVATE_STEP	BOOL	While the SFC chart is in halt mode (SFC_OPERATING_MODE_HALT), the value TRUE at this input activates the step whose step ID is specified at the STEP_ID input (see table row above). After a successful activation, the input is automatically reset to FALSE.

Figura 5.93 Explicación asociada a las entradas STEP_ID y ACTIVATE_STEP en PLCnext Engineer.

DEACTIVATE_STEP	BOOL	While the SFC chart is in halt mode (SFC_OPERATING_MODE_HALT), the value TRUE at this input deactivates the step whose step ID is specified at the STEP_ID input (see table row above). After a successful deactivation, the input is automatically reset to FALSE.

Figura 5.94 Explicación asociada a la entrada DEACTIVATE_STEP en PLCnext Engineer.

Si se desea llevar el SFC a una etapa determinada se indica su número en STEP_ID (1, si es la etapa inicial) y, desde en el modo HALT, con un flanco en ACTIVATE_STEP se hace que evolucione pudiendo continuar desde este estado. En esencia es como efectuar generar un flanco ascendente en la entrada SFCReset en Codesys.

Para algunos casos, como puede ser el ejemplo de gestión del SFC del servicio de vaciar alimentador (FB_2_Vaciar_Alim_ISA88) visualizado en la Figura 5.95, localizado en FB_F4_VDes_GEMMA se implementa directamente. En este caso se indica para que se proceda con el servicio, el Alimentador debe cumplir el estado Running de ISA88, el GDMMA encontrarse en F4 (verificación en desorden), el Alimentador detectando piezas (A0) y Cargador abierto (FC0) y el usuario actúe sobre Start_MAN. Se procederá con el servicio hasta que no se detecte pieza o el Cargador cerrado (FC1).

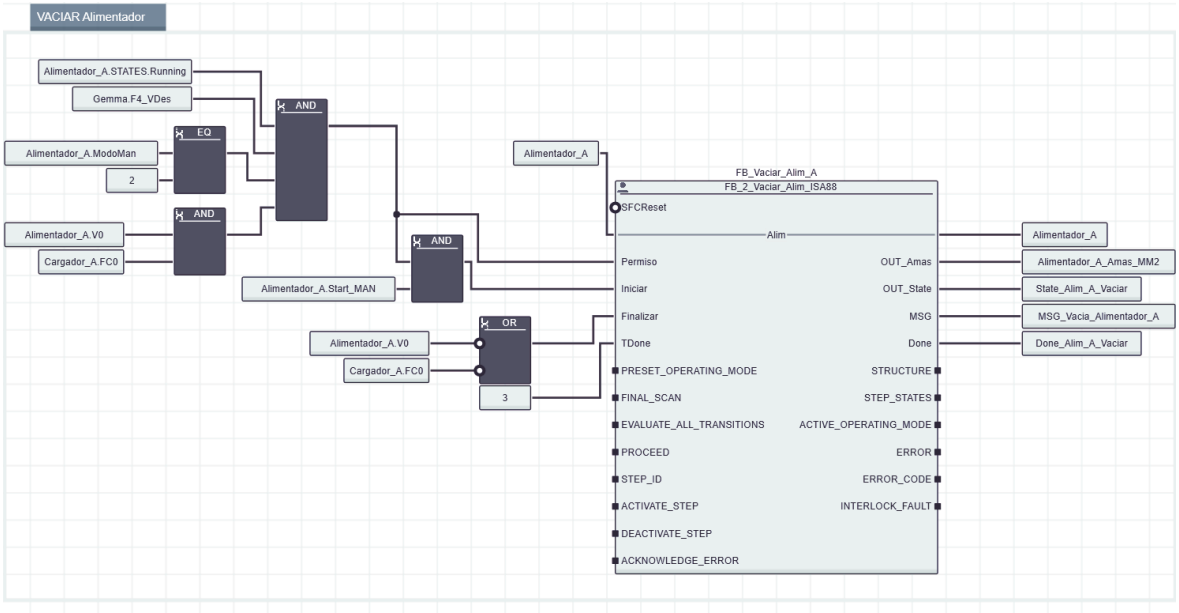

Figura 5.95 Código de gestión de SFC del servicio de vaciar alimentador

En caso de que se disponga de procesos que requieran la actuación sobre un estado el SFC en concreto, se recurrirá al bloque de manejo de SFCs como se ve en la Figura 5.96, para el caso de FB_4_Calibra_TR_Horno_ISA88, el cual permite el cambio entre el modo de funcionamiento normal del SFC o la puesta en HALT en caso de que se requiera el avance o puesta de un estado en concreto, puesto que este modo es el único que permite la libre actuación sobre el step ejecutado en el SFC.

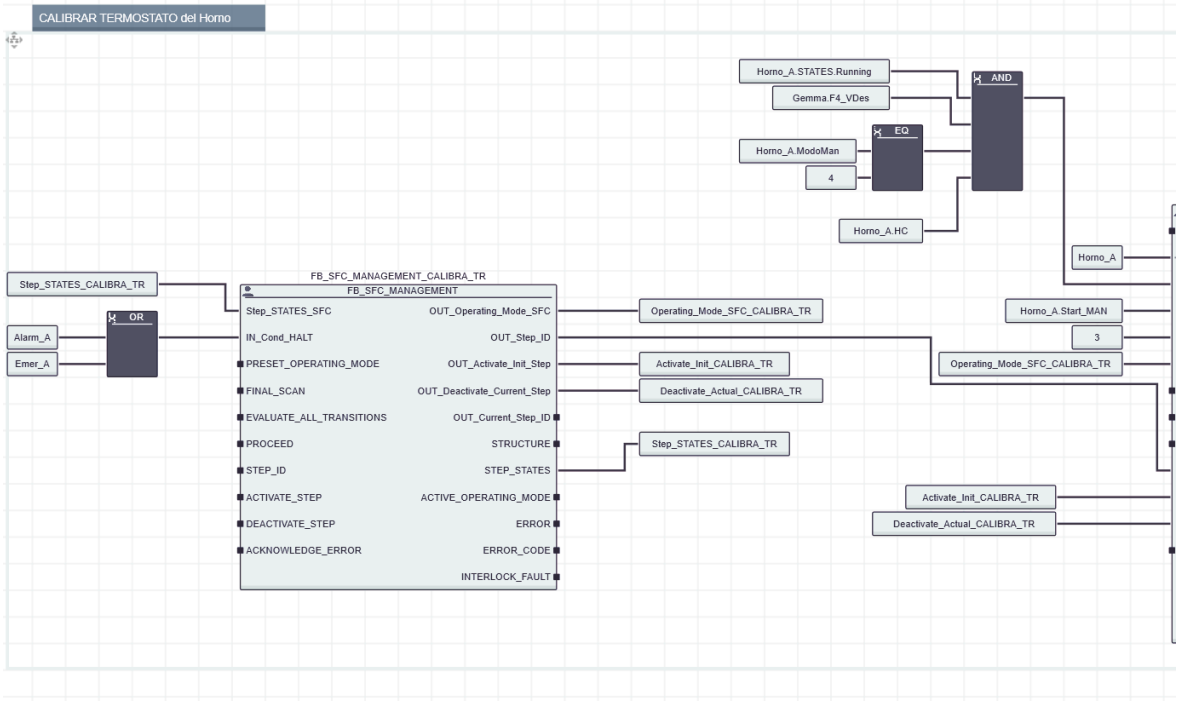

Figura 5.96 Bloque de función de manejo de SFCs asociado a un servicio.

5.2.2- OPC UA

Tras haber comentado el programa asociado al control del sistema descrito, la siguiente tarea que se ha realizado tras adaptar todo el código de Codesys a PLCnext Engineer, es configurar el servidor OPC UA de PLCnext y asociar las variables deseadas al mismo.

La manera de configurar el servidor OPC UA del controlador PLCnext ya ha sido comentada en este documento, en el apartado **"4.3.1 OPC UA Server"**. El siguiente paso consiste en adicionar las variables deseadas por el usuario al servidor OPC UA, habilitando el manejo de estas. Este proceso es realmente sencillo puesto que se hace simplemente accediendo a la pestaña donde se encuentre la variable que se desea añadir y activando la casilla OPC como se puede observar en la Figura 5.97.

Variable (PLC)	Type	Usage	Comment	Init	Retain	Constant	OPC
Gemma	DTGemma	Global			☐	☐	☑
Fin_AUTO	BOOL	Global		FALSE	☐	☐	☑
Estados_ISA88	DTStates...	Global			☐		☑

Figura 5.97 Activación de casilla destinada a OPC UA en PLCnext Engineer.

Aprovechando una sesión creada para la ya mencionada herramienta cliente UaExpert, se muestra en la Figura 5.98, un listado de las variables del programa introducidas en el servidor OPC UA que se han empleado.

Node Id	Display Name	Value	Datatype
NS5|String|Arp.Plc.Eclr/Last_Finished_Order_A.Order.ID	ID	1	Int32
NS5|String|Arp.Plc.Eclr/Last_Finished_Order_A.Order.SP_Weight	SP_Weight	125	Float
NS5|String|Arp.Plc.Eclr/Last_Finished_Order_A.Weight	Weight	{0,0,0,0,0,0,0,...	Float
NS5|String|Arp.Plc.Eclr/Cargador_A.Peso	Peso	0	Float
NS5|String|Arp.Plc.Eclr/Last_Finished_Order_A.Order.SP_Temp	SP_Temp	150	Float
NS5|String|Arp.Plc.Eclr/Last_Finished_Order_A.Temp	Temp	{0,0,0,0,0,0,0,...	Float
NS5|String|Arp.Plc.Eclr/Horno_A.Temp	Temp	21.76	Float
NS5|String|Arp.Plc.Eclr/Gemma.Step	Step	A5_PAtF	String
NS5|String|Arp.Plc.Eclr/Alimentador_A.STATES.msg_STATE	msg_STATE	Initial state	String
NS5|String|Arp.Plc.Eclr/Cargador_A.STATES.msg_STATE	msg_STATE	Initial state	String
NS5|String|Arp.Plc.Eclr/Horno_A.STATES.msg_STATE	msg_STATE	Initial state	String
NS5|String|Arp.Plc.Eclr/Step_PROD	Step_PROD	Init	String
NS5|String|Arp.Plc.Eclr/Online_Order_A.Order.ID	ID	1	Int32
NS5|String|Arp.Plc.Eclr/Online_Order_A.Order.SP_Temp	SP_Temp	150	Float
NS5|String|Arp.Plc.Eclr/Online_Order_A.Order.SP_Weight	SP_Weight	125	Float
NS5|String|Arp.Plc.Eclr/Online_Order_A.Order.TCT	TCT	30	Int32
NS5|String|Arp.Plc.Eclr/Pedido_A.ID	ID	1	Int32
NS5|String|Arp.Plc.Eclr/Pedido_A.SP_Weight	SP_Weight	125	Float
NS5|String|Arp.Plc.Eclr/Pedido_A.SP_Temp	SP_Temp	150	Float
NS5|String|Arp.Plc.Eclr/Pedido_A.TCT	TCT	30	Int32
NS5|String|Arp.Plc.Eclr/LOC_REM	LOC_REM	false	Boolean
NS5|String|Arp.Plc.Eclr/Pedido_A.Start	Start	false	Boolean
NS5|String|Arp.Plc.Eclr/Gemma.Start_PROD_REM	Start_PROD...	false	Boolean
NS5|String|Arp.Plc.Eclr/Gemma.Cycles_1_N_REM	Cycles_1_N_...	false	Boolean
NS5|String|Arp.Plc.Eclr/Gemma.Start_PROD	Start_PROD	false	Boolean
NS5|String|Arp.Plc.Eclr/Gemma.Cycles_1_N	Cycles_1_N	false	Boolean
NS5|String|Arp.Plc.Eclr/Online_Order_A.Running	Running	false	Boolean

Figura 5.98 Listado de variables manejadas en OPC UA visualizadas por cliente UaExpert.

5.2.3- Pantalla de operador (HMI Web Server)

En PLCnext Engineer también se ha desarrollado un conjunto de pantallas de operador para manejo del sistema a través de las prestaciones HMI Web Server que es posible desarrollar en el controlador PLCnext. En ese caso se ha seguido la misma estrategia que en Codesys, pantallas de operador fáciles de operar y representativas del proceso que se lleva a cabo en el controlador. Para acceder al desarrollo de estos elementos gráficos hay

que acceder al directorio: PLANT -> Project -> axcf2152 : AXC F 2152 -> HMI Web Server -> Application como se visualiza en la Figura 5.99.

Figura 5.99 Acceso a la configuración del HMI en PLCnext Engineer.

En dicha Figura 5.99 se pueden observar además todas las pantallas desarrolladas para la aplicación en PLCnext Engineer, con las mismas ventanas que en Codesys, aunque con una pantalla menos, la de observación del control de temperatura con el sistema de primer orden.

Siguiendo el orden mostrado de arriba abajo, la pantalla Login (identificación del usuario), se mostrará con cada nuevo arranque de la aplicación HMI Web Server (Figura 5.100).

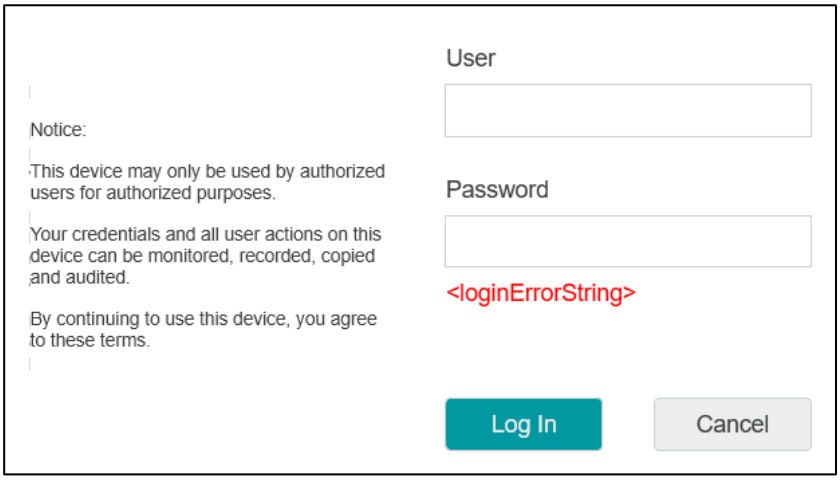

Figura 5.100 Pantalla destinada a la identificación del usuario para acceder a la pantalla de operador.

La siguiente pantalla, configurada como la pantalla por defecto (icono casa) es GDMMA_States y se activa tras introducir correctamente User y Password en la ventana Login. La implementación en PLCnext Engineer se puede visualizar en la Figura 5.101.

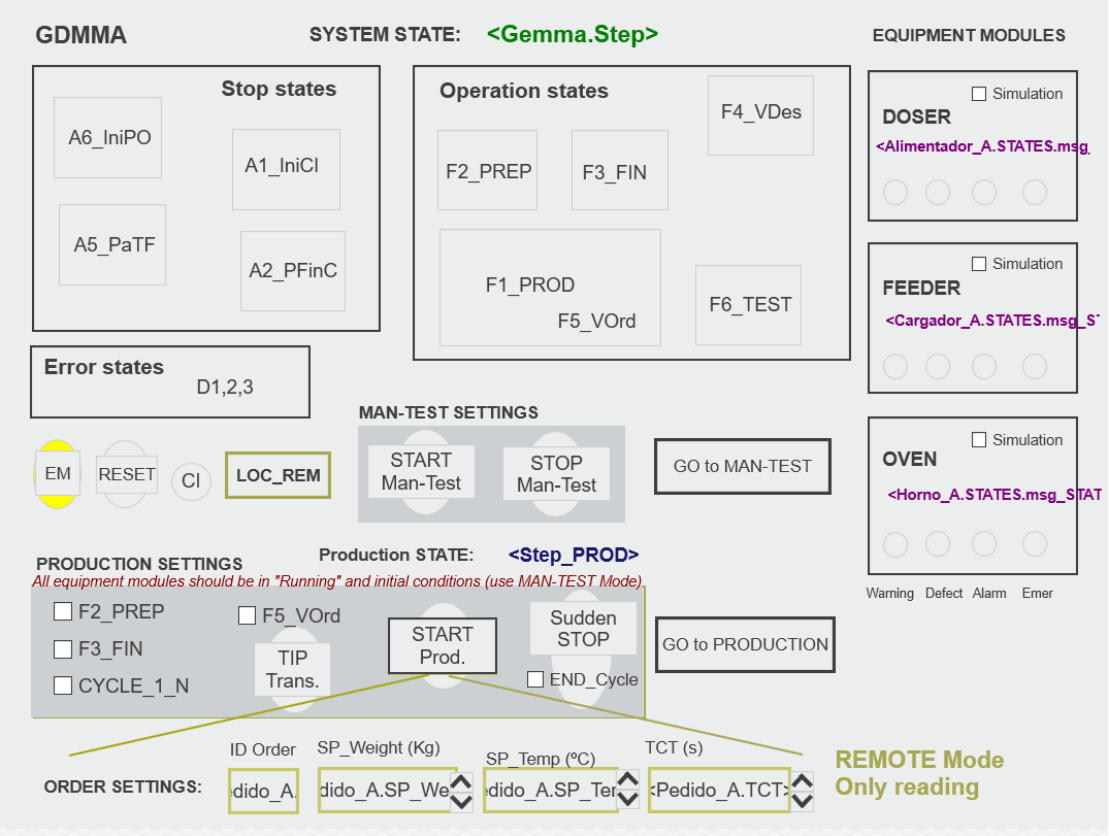

Figura 5.101 Pantalla asociada a GDMMA_States en PLCnext Engineer.

La siguiente ventana desarrollada es MAN_TEST_Mode que permite gestionar el sistema en el modo de verificación en desorden. La representación en pantalla es semejante a la previamente visualizada en Codesys y se puede ver representada sobre la Figura 5.102.

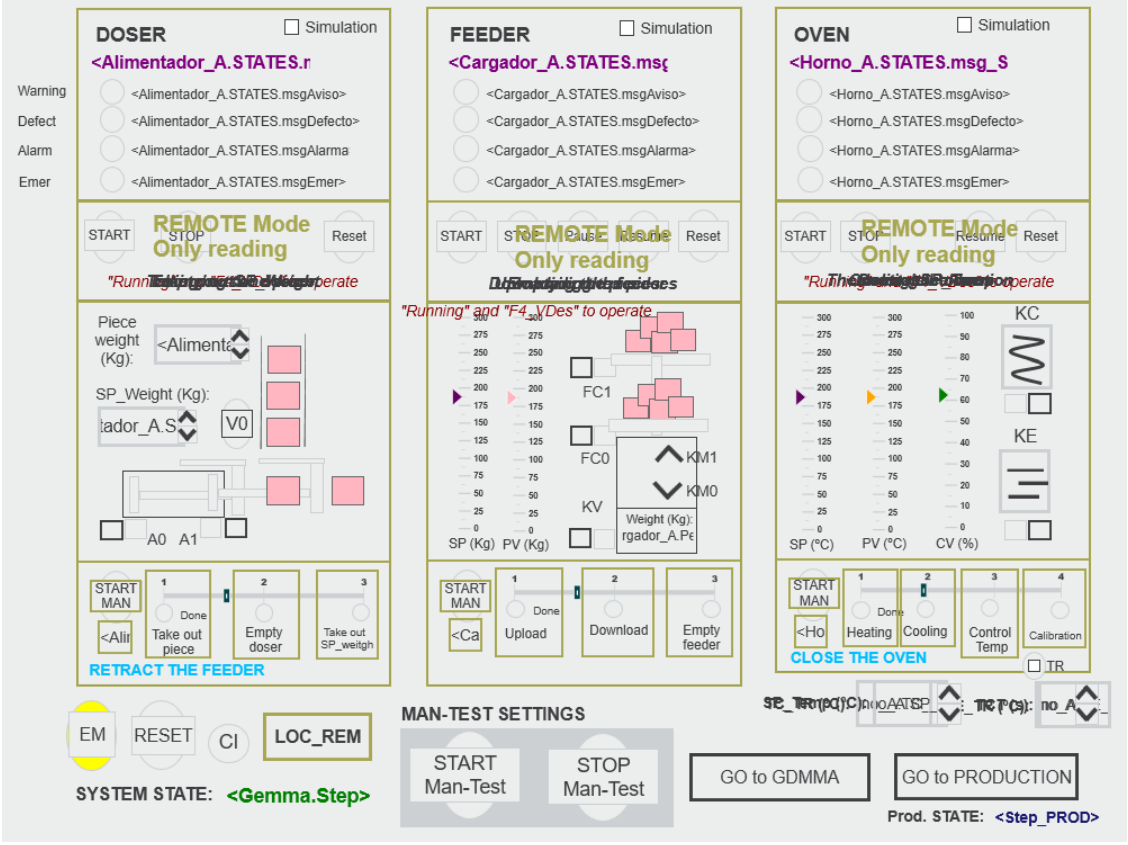

Figura 5.102 Pantalla asociada a MAN_TEST_Mode en PLCnext Engineer.

Finalmente, la última pantalla a comentar ligeramente ha sido la que se corresponde con la producción normal de planta (PRODUCTION_Mode). Ver Figura 5.103.

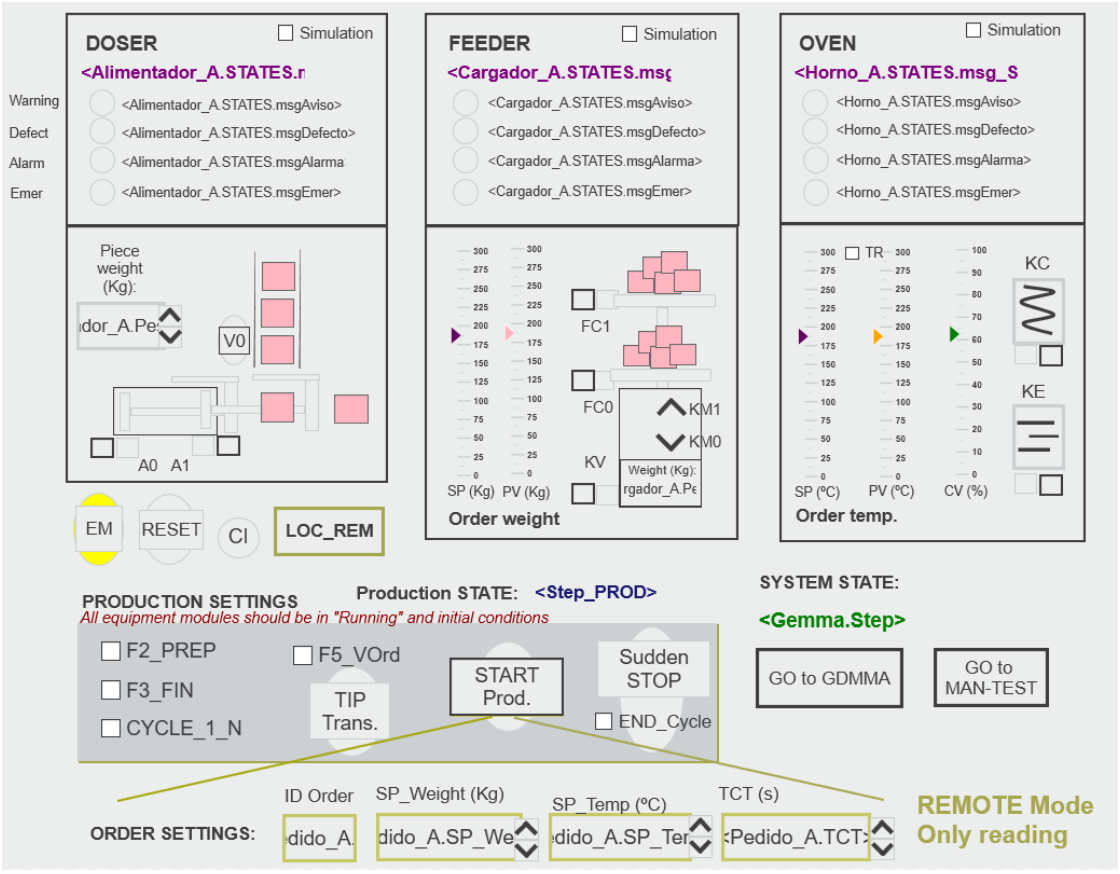

Figura 5.103 Pantalla asociada a PRODUCTION_Mode en PLCnext Engineer.

La ventana Dummy ha servido como “bando de pruebas” para el desarrollo de otras pantallas al no estar vinculada a la operación del sistema.

5.2.4- Preparación de datos para Proficloud.io

Finalmente, antes de avanzar a la programación desarrollada para el entorno Node-RED, conviene comentar que se ha realizado una prueba de la integración de los datos sobre la plataforma Cloud proporcionada por Phoenix Contact con sistemas PLCnext, Proficloud.io.

Si bien, es cierto que la integración es prácticamente directa, hay que realizar una pequeña adaptación en las variables para que estas se puedan integrar a dicha plataforma.

En este caso, se ha realizado la integración con Proficloud.io de dos variables características del sistema como son la temperatura del horno (procedente de la estructura de datos del horno, Horno_A) y el peso en el cargador (procedente de manera similar de la estructura de datos del cargador, Cargador_A).

Aunque se marque Proficloud en la declaración de la variable Horno_A (ver Figura 5.104) y supuestamente se tenga acceso a todos campos de la dicha estructura de datos donde estaría Horno_A.Temp, es preciso realizar la misma configuración para las variables independientes que en este caso se han asociado a otras dos variables auxiliares (con prefijo Proficloud_) y declaradas en el programa Main, asociadas al OUT_Port y marcadas Proficloud, según se aprecia en la Figura 5.105.

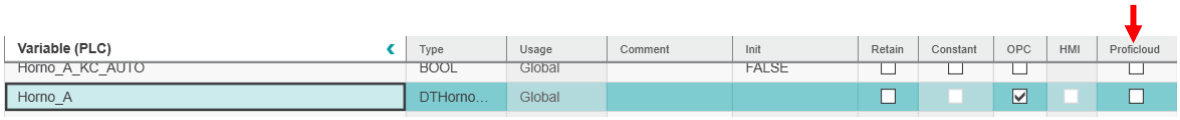

Figura 5.104 Incorporación de variable a Proficloud.io.

Name	Type	Usage	Proficloud
I_iChTemp	DINT	External	
Proficloud_Temperature	REAL	OUT Port	☑
Proficloud_Weight	REAL	OUT Port	☑

Figura 5.105 Variables auxiliares incorporadas a Proficloud.io.

Y habrá que realizar una pequeña adición al código en el que se asignen las variables de temperatura y peso de las estructuras de datos a estas variables auxiliares introducidas como se puede visualizar en la Figura 5.106.

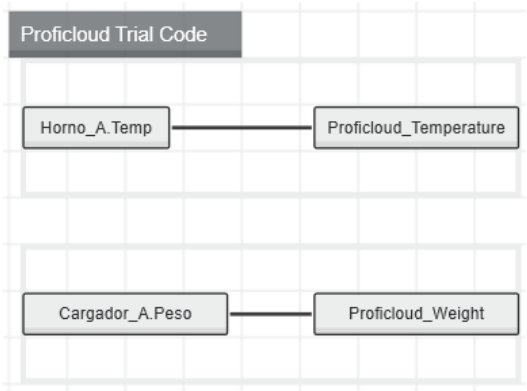

Figura 5.106 Código en FBD creado para la asignación de variable temperatura a la variable auxiliar.

Tras estos pasos, sí que se tendrán las variables listas para ser representadas en la plataforma Proficloud.io como se puede ver en la **Figura**, en la que se observan las variables tratadas, representadas sobre gráficas de la plataforma.

Figura 5.107 Representación de las variables sobre Proficloud.io.

5.3.- PROGRAMACIÓN EN NODE-RED

Vista la programación del control de la planta desde el ejemplo de desarrollo inicial en Codesys, y su adaptación de manera casi directa a PLCnext, quedaría por introducir y desarrollar un último paquete de programación generado para el entorno Node-RED, tanto en el controlador físico de PLCnext, como el desarrollado en los entornos Cloud (IBM Cloud, y finalmente Azure Cloud).

Se parte de la base que ya está completamente accesible el entorno Node-RED y con los nodos de operación necesarios para proceder con la programación puesto que ya ha sido instalado y configurado de forma correcta todo lo necesario.

5.3.1- Programación Node-RED en controlador PLCnext

Se comienza con la lectura en Node-RED de las variables localizadas en el servidor OPC UA del controlador para poder operar con ellas de la manera que el usuario desee. Para ello se requiere el conocimiento del identificador (NodeID) de cada una de las variables.

5.3.1.1 Identificadores de variables PLC (NodeID):

Para disponer del identificador de cada variable hay dos posibilidades, se han aplicado las dos, pero se recomienda una frente a la otra:

1) La primera opción es implementar sobre PLC un pequeño programa independiente que se conecte como cliente del servidor, pero con el nodo Browser de OPC UA que se llevará al Dashboard de Node-RED y permitirá navegar por la raíz del servidor, pudiendo acceder individualmente a los identificadores asociados a cada variable. El aspecto gráfico del código a implementar es el mostrado en la Figura 5.108. Sin embargo, no es el recomendable ya que es algo tosca la navegación por el servidor mediante este método.

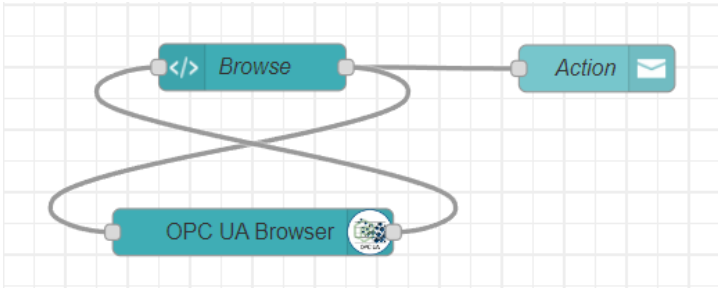

Figura 5.108 Buscador OPC UA en Node-RED.

2) La segunda opción posible, es el emplear la herramienta UaExpert, ya comentada anteriormente en este documento. Simplemente habrá que realizar la conexión con el servidor OPC y se tendrá acceso directo a lo mismo que se ha implementado en el código anterior, pero incluso con más opciones a poder aplicar si se dispone de la configuración adecuada en el servidor y con un manejo mucho más natural, cómodo y rápido por el servidor OPC UA. La Figura 5.109 muestra una conexión establecida directamente con el servidor y algunas variables del PLC ya seleccionadas.

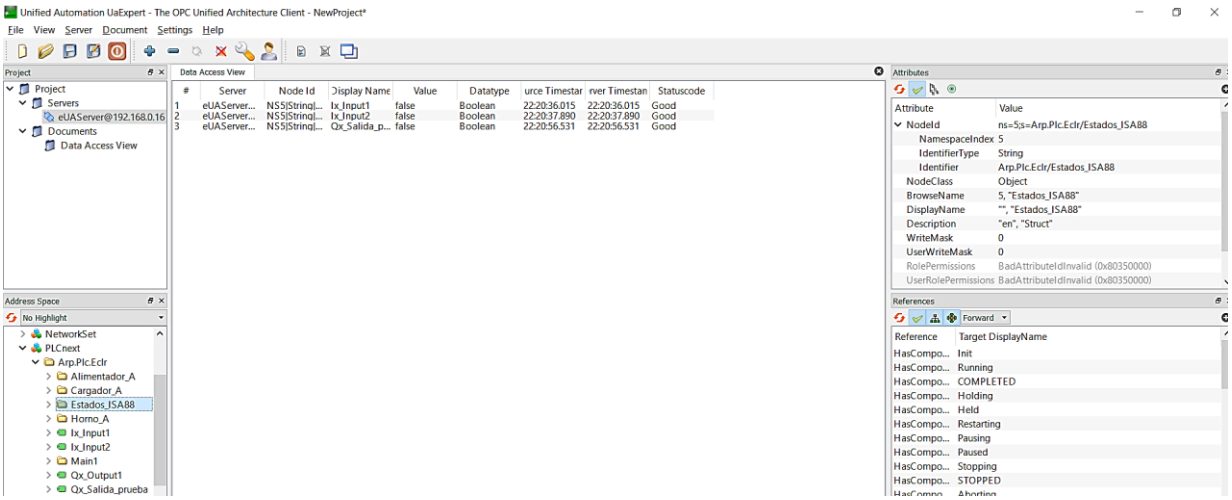

Figura 5.109 Cliente UaExpert conectado a servidor OPC UA.

Si bien es cierto que UaExpert no es en sí una programación desarrollada en Node-RED, es muy conveniente reflejar el uso de este cliente OPC UA en este punto ya que agiliza

mucho la posterior programación que se va a narrar y evita el programar de más y por lo tanto volver más complejo el código consumiendo más recursos en el controlador.

5.3.1.2 Lectura de variables del servidor OPC UA:

Tras el análisis de los métodos para la obtención de los identificadores de variable asociados al servidor OPC UA, el siguiente paso natural es leer dichas variables en Node-RED para poder operar con ellas.

Para ello, lo que hay que realizar es una inyección periódica de las variables generando así el periodo deseado de lectura. Para ello habrá que tomar un nodo de inyección de tiempo y asignarle el NodeID correspondiente a la variable.

Por ejemplo, tomando la variable de lectura de peso, el NodeID asociado sería:

ns=5;s=Arp.Plc.Eclr/Cargador_A.Peso

y habría aparte que indicar el tipo de dato que es, en este caso tipo FLOAT, quedando finalmente:

ns=5;s=Arp.Plc.Eclr/Cargador_A.Peso;datatype=Float.

Este formato se aplicará a cada una de las variables que se lean.

Visualizando como quedaría sobre el entorno de programación. Lo primero a realizar es incorporar al entorno de programación el nodo de inyección temporal (Timestamp) como se visualiza en la Figura 5.110.

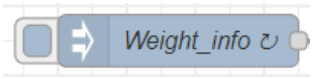

Figura 5.110 Nodo de inyección temporal en Node_RED.

Tras esto, se accede a dicho nodo y se configurar internamente con el NodeID de la variable y el tiempo tras el que se pretende obtener cada lectura como se aprecia en la Figura 5.111.

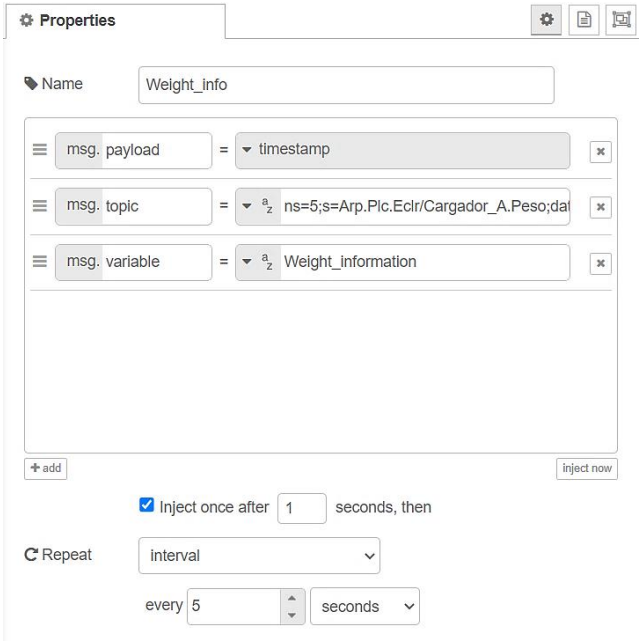

Figura 5.111 Configuración de recepción de variable del servidor OPC UA.

Con esto aún no se estará recogiendo la información del servidor OPC UA, porque aún no se ha abierto una sesión como cliente de lectura contra el propio servidor. Para ello, se requerirá incorporar al entorno de programación un nodo cliente OPC UA y que este sea configurado como lectura (READ) para ello, a su vez en este nodo, se tendrán que incorporar las credenciales de identificación de seguridad propias del servidor para que deje acceder como cliente al servidor.

El nodo cliente OPC UA en el entorno de Node-RED, tendría el aspecto mostrado en la Figura 5.112.

Figura 5.112 Nodo cliente de servidor OPC UA.

Accediendo a la configuración del mismo para incorporar los datos de inclusión detallados previamente, se estaría visualizando la siguiente imagen mostrada en la Figura 5.113.

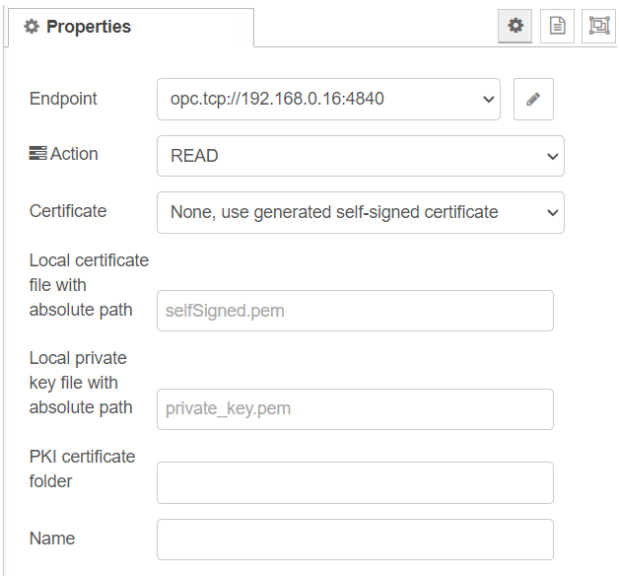

Figura 5.113 Configuración del nodo cliente OPC UA.

Sin embargo, pese que se vea la inclusión de la IP asociada al servidor OPC UA y se indique que la acción a realizar sea de lectura, faltaría por acceder a la declaración de seguridad propia del servidor y la identificación en el mismo. Para completar estas acciones, habría que acceder al apartado “Endpoint”, apareciendo lo que se muestra en la Figura 5.114 y permitiendo completar la configuración estipulada.

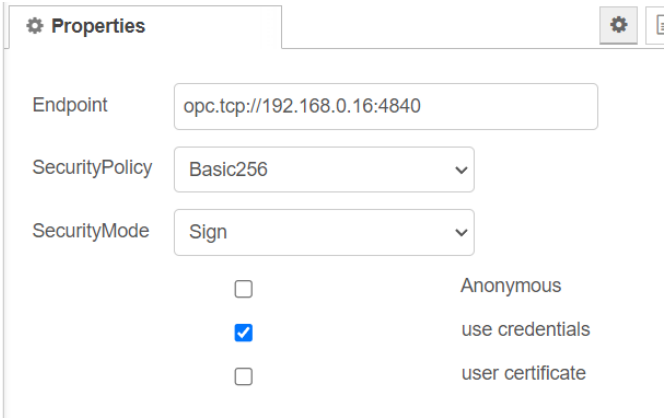

Figura 5.114 Configuración del nodo cliente OPC UA con las credenciales se seguridad apropiadas.

Uniendo ambos nodos citados se obtendría en el entorno de programación algo semejante a lo mostrado en la Figura 5.115, en la cual se está realizando la toma de los datos de peso y temperatura en el entorno Node-RED.

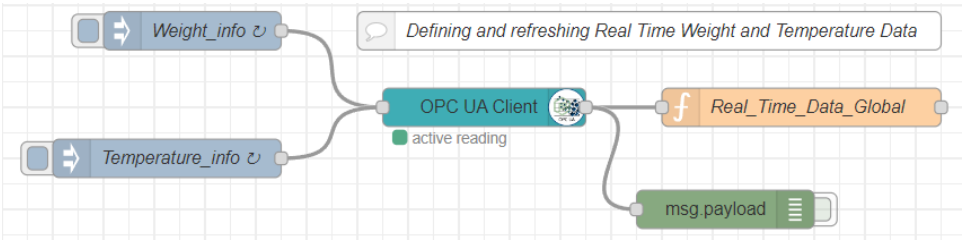

Figura 5.115 Toma de datos de peso y temperatura del servidor OC UA.

Como se puede ver, también en la Figura anterior, una vez realizada correctamente la apertura de la sesión contra el servidor OPC UA, aparecerá una señal en verde indicando que la acción que se ha solicitado está activa (en este caso, lectura activa). Se pueden percibir, dos elementos “extraños” aún no explicados. El primero a describir, en naranja, es una función generada para generar variables de carácter global en el entorno de Node-RED para trabajar más cómodamente (aunque se podría trabajar directamente con los NodeID). La función generada se puede observar en la Figura 5.116.

```
const info = Number(msg.payload);

if (msg.variable == "Weight_information") {
    global.set("Weight_information",info);
}
if (msg.variable == "Temperature_information") {
    global.set("Temperature_information",info);
}

```

Figura 5.116 Función de generación de variables globales de temperatura y peso en el entorno de Node-RED.

Y el otro elemento, en verde, es simplemente un elemento de depuración de código que se ha mantenido para verificar que se está recibiendo y generando la información de manera correcta.

Claramente, la obtención del peso y la temperatura, no han sido las únicas variables utilizadas, hay muchos más elementos que se han tomado del control asociado al PLC. Por ejemplo, se ha realizado la obtención de todos los datos de pedido, tanto en realización, como inmediatamente anterior, como en estado de preparación. La estructura implementada para este caso se puede observar en la Figura 5.117.

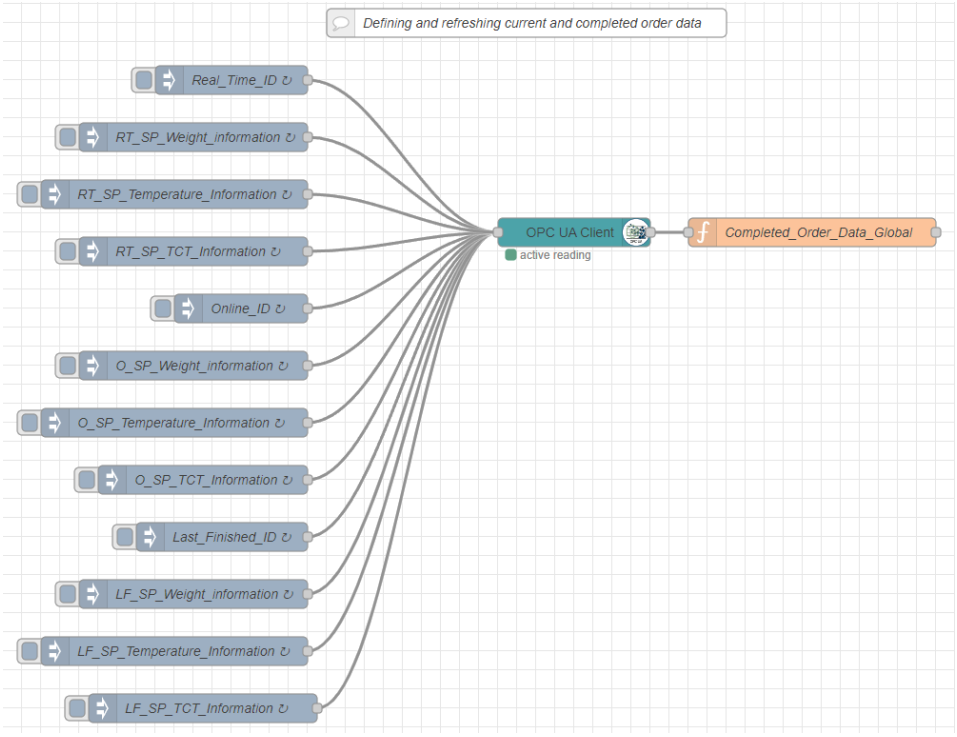

Figura 5.117 Recepción de datos de pedido por parte del servidor OPC UA.

Se han recopilado también los datos de estados del GDMMA, los estados según ISA88 de cada uno de los módulos de equipamiento y el estado en modo producción. Esta parte del código se muestra en la Figura 5.118.

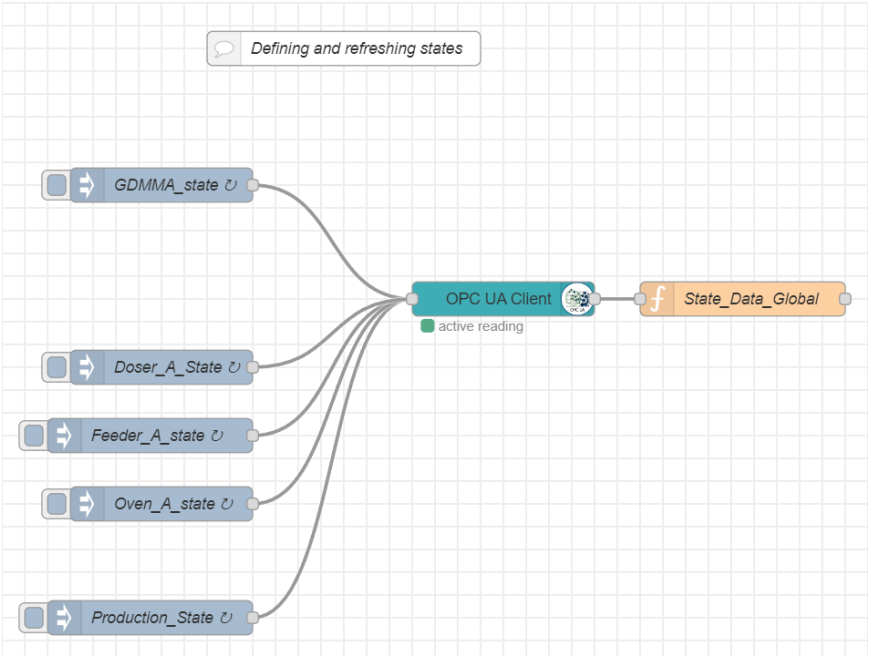

Figura 5.118 Recepción de estado del GDMMA y estados ISA88 de cada uno de los módulos de equipamiento.

Y para finalizar el proceso de lectura, unos datos de carácter general del sistema como: el estado de local/remoto (en remoto se podría controlar el lanzamiento de un pedido a través de Node-RED), si está configurada la producción en ciclo continuo, etc. Esta parte de código se corresponde con la desarrollada en la Figura 5.119.

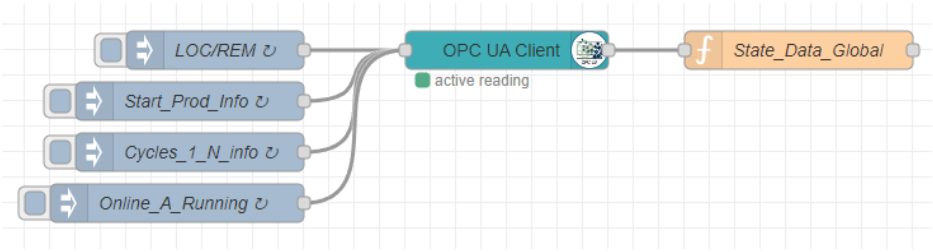

Figura 5.119 Recepción de datos de local, remoto, ciclo continuo...

Destacar que estas conexiones de cliente se han realizado cumpliendo los requisitos de seguridad impuestos por el controlador PLCnext en su servidor OPC UA, ya que a tenor de dicha seguridad, tiene estipulado que no se pueden abrir al mismo tiempo más de 5 sesiones de conexión con este. En este caso, generalmente siempre se tendrán operativas 4

sesiones de lectura y 1 de escritura que aún no se ha descrito. Remarcar que esta lectura de variables se podría haber hecho con un solo cliente de lectura, pero se volvería más difícil de visualizar la diferenciación del tipo de variables que se está recopilando.

5.3.1.3 Representación de valores en Dashboard de Node-RED:

Antes de realizar el análisis de la escritura de valores sobre variables disponibles en el servidor de OPC UA, se va a explicar el procedimiento para la representación de las variables leídas del servidor.

En todos los casos, este procedimiento es muy similar, solo que adaptándose al tipo de dato que se desea representar y que localización del Dashboard se quiere ocupar. Por ejemplo, siguiendo el caso inicial explicado para la lectura de las variables de peso y temperatura, su representación toma la siguiente estructura mostrada en la Figura 5.120 en el entorno de programación.

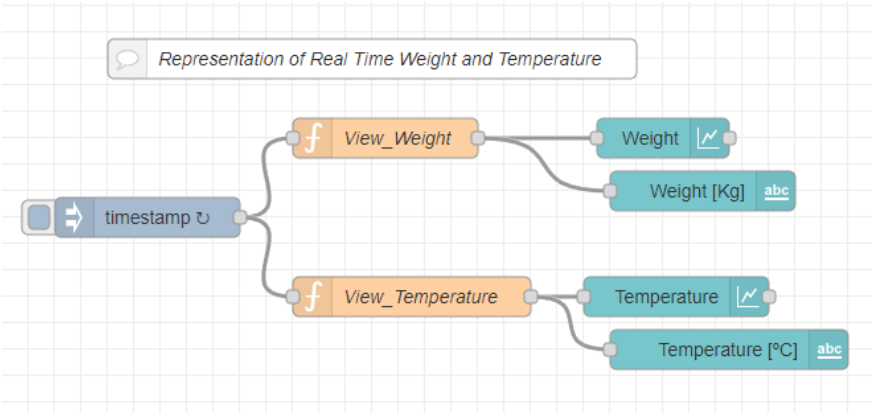

Figura 5.120 Representación sobre Dashboard de valores de peso y temperatura.

En este caso, y ya teniendo en cuenta lo explicado previamente para la lectura, se puede apreciar que se está realizando una actualización de los datos de tiempo y temperatura en el Dashboard cada un cierto periodo de tiempo (debido al *timestamp*). Se ve que esa inyección de tiempo se une con dos funciones. Estas funciones simplemente sirven para tomar el valor de la variable a representar correspondiente, cada vez que la inyección de

tiempo (*timestamp*) genera señal. La función tendría un aspecto para cada variable tan sencillo como el que muestra en la Figura 5.121.

```
1   msg.payload = global.get("Weight_information");
2   return msg;
```

Figura 5.121 Código para recopilación de valor de la variable global de temperatura.

Y finalmente habría que vincular los nodos de representación sobre el Dashboard. Por ejemplo, para el caso analizado del peso, se representarían una gráfica (nodo de arriba en la Figura 5.122) y el valor numérico inmediatamente debajo de la gráfica (nodo de abajo en la Figura 5.122).

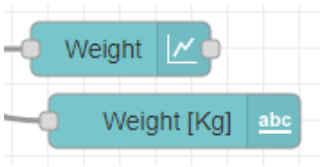

Figura 5.122 Nodos de representación de gráfica y de valor tipo texto.

Internamente estos nodos deberán tener una configuración asociada en la cual se le indica el posicionamiento que han de ocupar en el Dashboard, para ello, se han de crear clases de representación. Tomando la representación del peso en la gráfica, se representa en la pestaña que ocupa el nombre: [Real_Time_Values] y dentro de esta pestaña se localizaría en la columna asociada con el nombre: Real_Time_Data. Para generar esa configuración mencionada se tendrían que replicar las opciones visualizadas en la Figura 5.123.

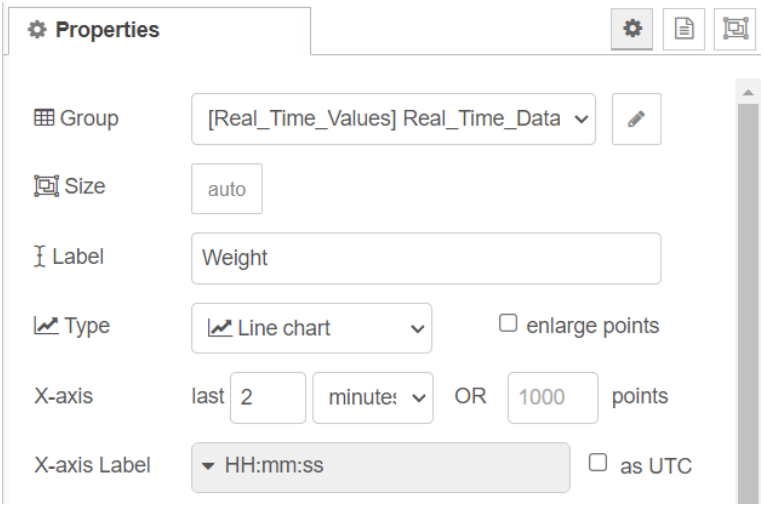

Figura 5.123 Configuración de la representación de variable sobre Dashboard.

En el Dashboard por lo tanto se localizaría en la pestaña de visualización y columna de posicionamiento descritas como se muestra en la Figura 5.124.

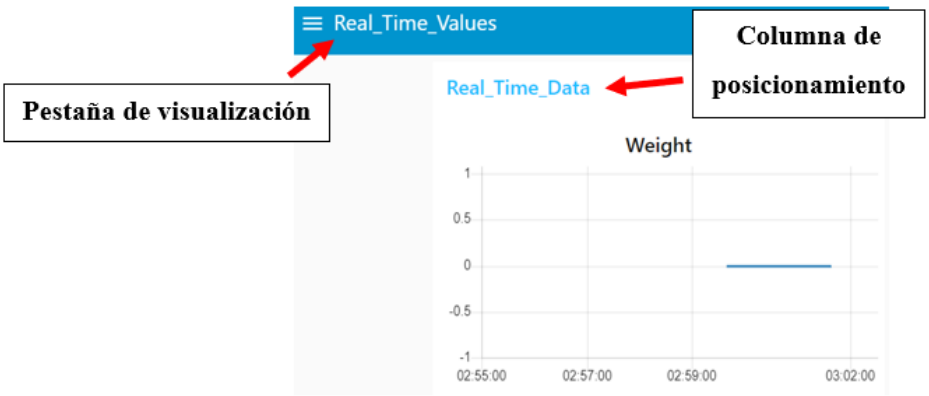

Figura 5.124 Representación de la configuración previa sobre el Dashboard.

El resto de los elementos representados siguen una representación muy similar. Por ejemplo, el código mostrado en la Figura 5.125 muestra la representación de los valores del pedido por parte del usuario que se está generando para producción de la planta en tiempo real.

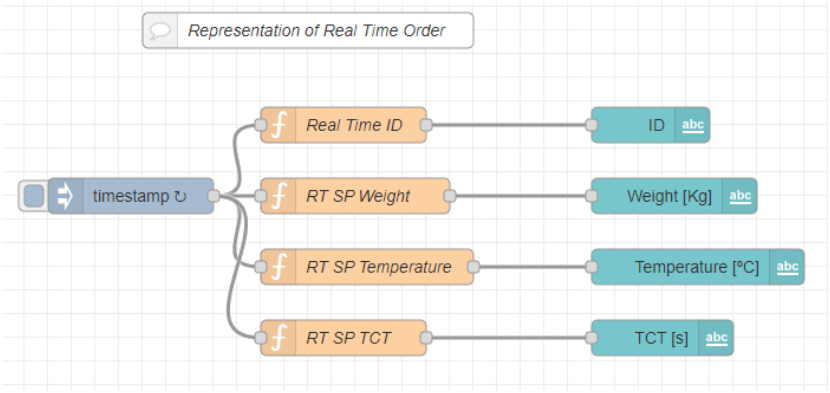

Figura 5.125 Representación de pedido en configuración sobre Dashboard.

Lo mismo sucedería para el caso de la representación de los estados del GDMMA, los estados ISA 88 de los módulos de equipamiento y el estado en modo producción de la planta como se muestra en la Figura 5.126.

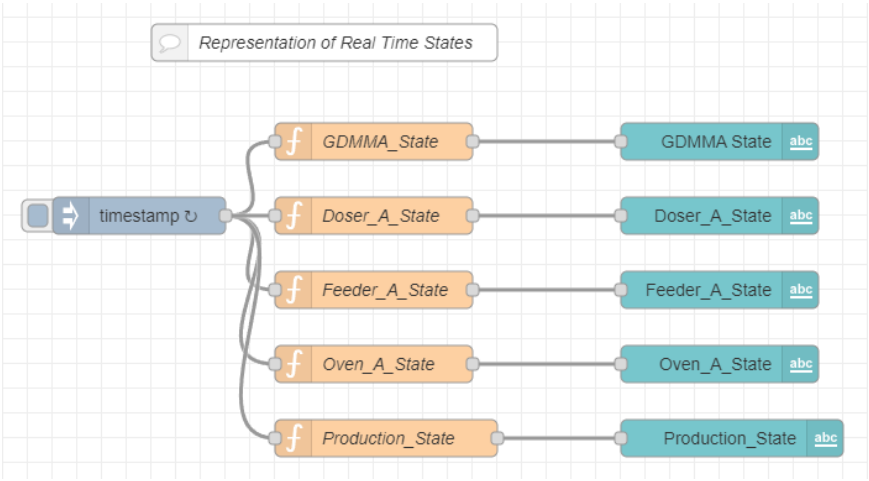

Figura 5.126 Representación de estado GDMMA, estados ISA88 de los módulos de equipamiento y estado de producción.

Aún hay más representaciones generadas para las variables tomadas, pero puesto que el código de representación es muy semejante se omite en este documento. También es importante remarcar que casi toda la representación de variables se podría asociar a un solo elemento de inyección de tiempo, pero se ha separado para la facilidad del usuario de ver

que se está preparando para visualización y que sea más fácil depurar el código en caso de producirse errores durante la generación de programa.

5.3.1.4 Escritura de valores al servidor OPC UA:

Hasta el momento se ha realizado la lectura de variables localizadas en el servidor OPC UA y su posterior representación sobre el Dashboard. No hay que olvidar el objetivo principal de paso de variables desde el controlador PLCnext a la plataforma Cloud. Sin embargo, antes de proseguir con la explicación del puente generado para conectar controlador con plataforma Cloud, también se ha implementado un control a "nivel remoto" del sistema por medio de Node-RED a través del cual el usuario puede controlar la receta de pedido y controlar varios parámetros como el lanzamiento de un pedido y control de ciclo continuo o discontinuo de producción.

Para ello hay que recurrir a otro cliente que abra sesión contra el servidor OPC UA del controlador, pero en este caso, en vez de como lectura, claramente, como se ha descrito, en modo escritura. La configuración del nodo solo cambiaría respecto al de lectura previamente descrito en lo representado en la Figura 5.127 que se ve que la acción asociada a este es la escritura ("Action": WRITE).

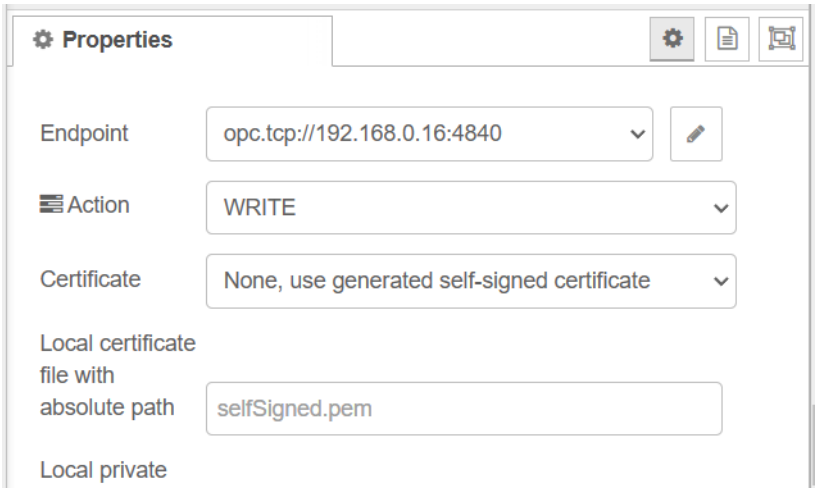

Figura 5.127 Configuración de cliente OPC UA en modo escritura.

Tras este paso, ya se podrían anexionar los nodos de control sobre las acciones descritas en el entorno de trabajo, quedando el código de envío como se muestra en la Figura 5.128.

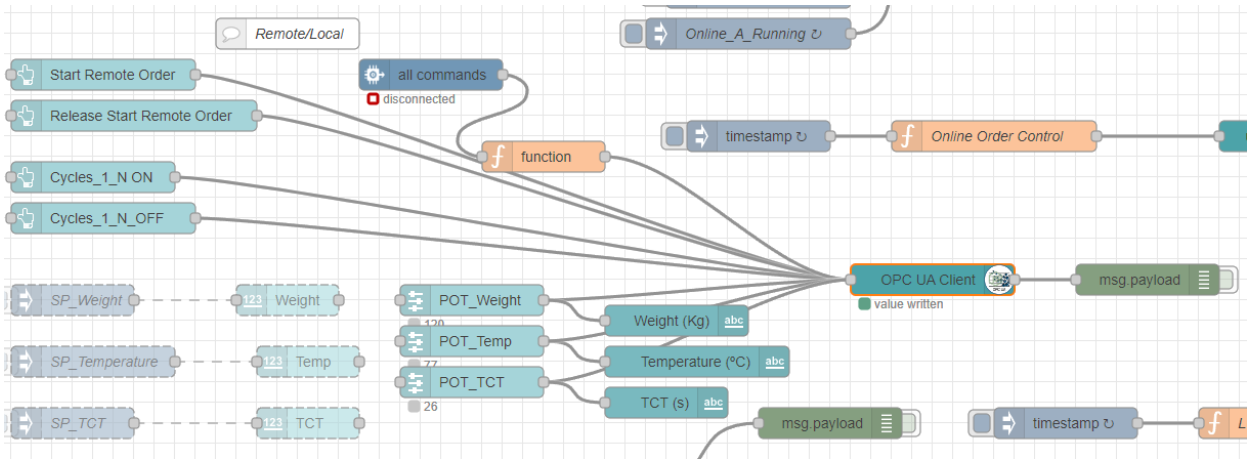

Figura 5.128 Código de escritura de variables sobre el servidor OPC UA.

Como se puede observar, hay varios elementos conectados al cliente OPC UA de escritura. Cada uno de estos elementos es un nodo asociado al Dashboard que representa una acción de escritura sobre la variable correspondiente.

De arriba abajo en la Figura 5.128 se podrían distinguir:

1) Un botón incorporado al Dashboard visualizado en la Figura 5.129 que permite al usuario el lanzamiento de un pedido en el modo producción cuando el sistema se encuentra en modo lanzamiento. En la configuración interna habrá que asociarlo con la variable del servidor OPC UA con su NodeID correspondiente como se muestra en la Figura 5.130.

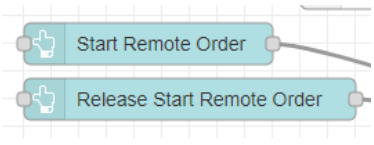

Figura 5.129 Generación de botones en el Dashboard ara lanzamiento de pedidos.

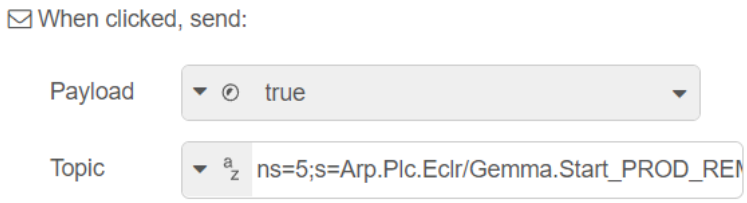

Figura 5.130 Configuración del NodeID para lanzamiento de pedido.

2) También se ha incorporado un botón para indicar si el proceso se ha de realizar en ciclo continuado o lanzamiento individual mediante interacción del usuario como se percibe en la Figura 5.131.

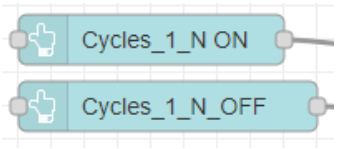

Figura 5.131 Pulsador para configuración de ciclo continuo de producción.

3) También estará la inclusión de los elementos deslizadores que permitirán la interacción por parte del usuario para la modificación de la receta generada para pedido. Puesto que el pedido viene comprendido por 3 valores sobre los que actuar: 1) Peso (POT_weight), 2) Temperatura (POT_Temp) y 3) Tiempo del control de temperatura (POT_TCT), se han añadido los elementos mostrados en la Figura 5.132. Son dos elementos, pero en realidad a la izquierda se encuentra directamente el elemento deslizador que actúa sobre la escritura de la variable y los nodos de la derecha se emplear para generar una realimentación visual del valor seleccionado. La configuración interna de las variables es como la realizada para los botones.

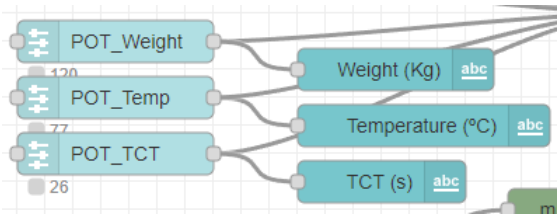

Figura 5.132 Potenciómetros para configuración de receta (peso, temperatura y tiempo de control de temperatura).

Se puede apreciar otro elemento conectado a la inyección de datos al servidor y es el correspondiente nodo de enlace con la plataforma Cloud de IBM, de la cual se traían los mismos datos que se habían implementado en la red local para el control del modo remoto, pero accesible a nivel externo por ser controlable desde el Dashboard localizado en el Node-RED de IBM. El nodo en cuestión al que se está haciendo referencia es el mostrado en la Figura 5.133.

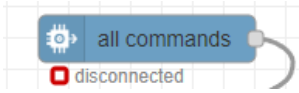

Figura 5.133 Nodo de conexión con la plataforma Cloud de IBM.

5.3.1.5 Envío de datos a la plataforma Cloud

Finalmente, para completar el proceso completo que comienza en la configuración de las variables seleccionadas para incorporar al servidor OPC UA y finaliza la preparación y envío de estas variables a la plataforma Cloud seleccionada por parte del usuario.

Entonces, una vez visto la recepción, representación e incluso modificación de las variables del controlador a nivel local, ya es momento de ascender otro nivel preparando los datos para enviar a la nube. Para ello hay que tener en cuenta que los datos han de ser tratados porque no pueden ser enviados en el formato que llegan del servidor OPC UA.

Sí que es cierto que el tratamiento de los datos es prácticamente igual para todos los entornos Cloud. Para ver la similitud se va a proceder ordenadamente a realizar la comparativa entre la disposición de los datos para la plataforma IBM Cloud previamente implementada en el proyecto y posteriormente se comparará con la finalmente adoptada Azure de Microsoft.

- **IBM Cloud:**

Comenzando entonces por la visualización del código desarrollado para el envío de datos a la plataforma IBM Cloud, se ha generado el siguiente fragmento de código mostrado en la Figura 5.134.

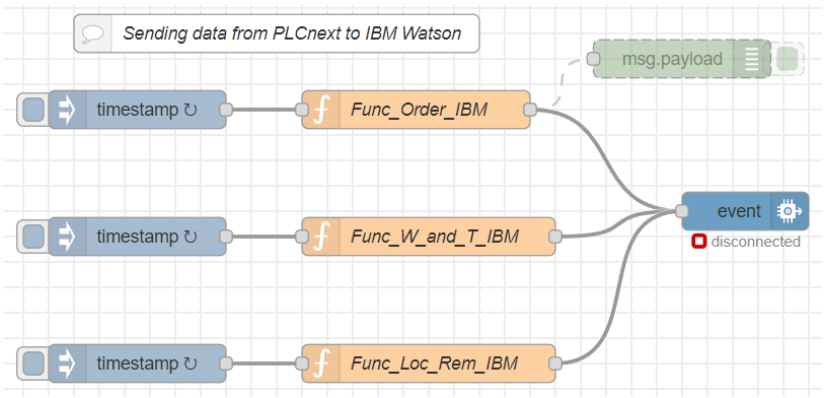

Figura 5.134 Envío de datos a la plataforma Cloud de IBM.

En cuanto a la descripción del código a nivel gráfico es muy semejante al procedimiento realizado para realizar la representación de las variables sobre el Dashboard. En este caso, hay una inyección temporal, una función y un nodo de Dashboard. Para enviar datos a la nube, hay una inyección temporal, una función y un nodo asociado a la herramienta de comunicación propia de cada plataforma Cloud (IoT Waston -> IBM Cloud e IoT Hub -> Azure Microsoft).

Entrando un poco más en detalle en la función, sí que se diferencia de la función desarrollada para la representación en el Dashboard, porque en la función asociada al envío

de datos a la nube, lo que hay que realizar es “empaquetar” los datos para que la plataforma Cloud los pueda interpretar adecuadamente.

Detallando entonces la función, que se muestra en la Figura 5.135**,** se está visualizando los datos de las variables de pedido para enviar precisamente información de dicho pedido a la nube. Tomadas las variables lo que se realiza es generar un JSON, que es un formato de texto sencillo que básicamente se puede definir como un subconjunto de datos que se envía de manera agrupada.

```
var LF_ID = global.get("Last_Finished_ID");
var LF_Weight = global.get("SP_Weight_Info");
var LF_Temp = global.get("Temp_Info");
var LF_TCT = global.get("TCT_Info");

msg.payload =
    {
        d:{
            "Last_order_finished_ID":LF_ID,
            "Last_order_finished_Weight":LF_Weight,
            "Last_order_finished_Temperature":LF_Temp,
            "Last_order_finished_TCT":LF_TCT
        }
    }
return msg;
```

Figura 5.135 Función de preparación para envío de datos a plataforma Cloud IBM.

En este caso se está generando un JSON que contiene la información del identificador de pedido y peso, temperatura y tiempo de control solicitados.

Una vez generada la función que prepara los datos lo que faltaría es realizar la conexión con la herramienta de comunicación de la plataforma Cloud. Para ello hay que realizar la introducción de las credenciales de identificación asociadas al dispositivo que se conecta dentro de la configuración del nodo de comunicación con la plataforma Cloud como se representa en la Figura 5.136.

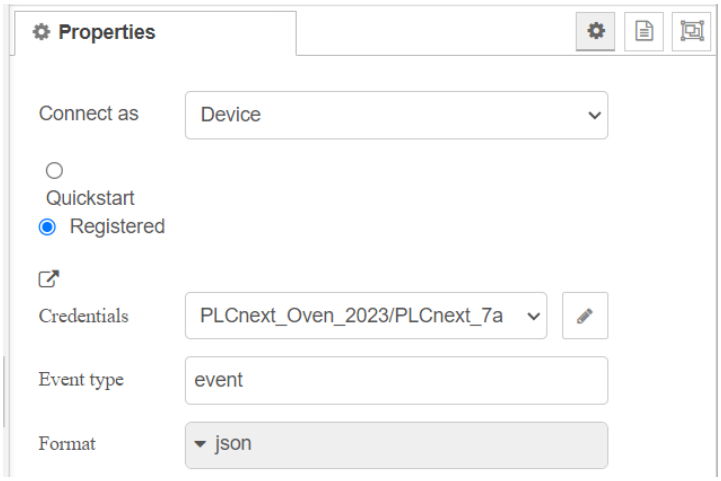

Figura 5.136 Configuración del nodo de comunicación con la plataforma Cloud de IBM con las credenciales adecuadas.

Con esto ya se estaría realizando el envío de datos a la plataforma de IBM Cloud, pero esta no ha sido la implementación final de proyecto pues va a ser sustituida por la plataforma Azure de Microsoft.

- **Azure de Microsoft**

El cambio resultó ser prácticamente directo, puesto que la estructura de envío siempre acaba siendo muy semejante, como se aprecia en la Figura 5.137. La dificultad esencialmente reside en la selección de la plataforma que más se acomode al usuario y en la adaptación a la herramienta de comunicación de dicha plataforma.

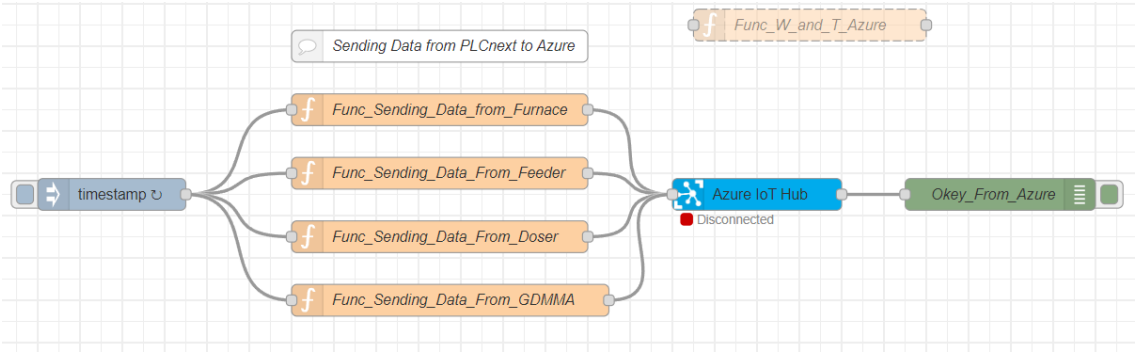

Figura 5.137 Envío de datos a la plataforma Cloud de Azure.

Ya de primeras se ve que la estructura es la misma. Un nodo de inyección temporal, los nodos de función de adaptación de mensaje a la plataforma Cloud y el nodo para enlazar con la herramienta de visualización. En este caso, se ve un nodo extra, en verde, que como ya se había comentado, solo se usan para la depuración del programa y en este caso ver que se hace la correcta recepción del mensaje en la plataforma.

Ahora tocaría analizar la estructura de la función para ver diferencias entre la previamente generada para enviar datos a IBM Cloud.

Sin embargo, antes de introducir el desarrollo de la función se va a proceder a analizar el nodo de enlace con la herramienta de comunicación puesto que este tiene una configuración que será importante tener en cuenta para el desarrollo de la función.

La configuración del nodo puede observarse en la Figura 5.138. Se puede ver que hay una identificación de credenciales muy semejante a la empleada en IBM Cloud (“Hostname”), pero, por el contrario, se puede notar que hay un apartado que manda seleccionar el tipo de protocolo de comunicación deseado en el intercambio de mensajes. En este caso se ha seleccionado el protocolo de comunicación MQTT el cual es un protocolo de red ligero de publicación suscripción muy extendido en el mundo IoT.

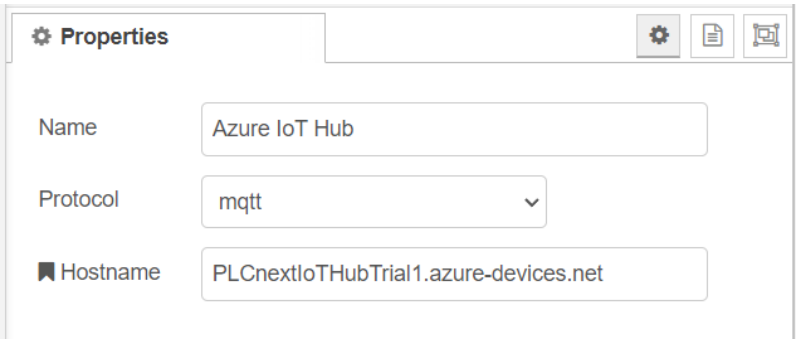

Figura 5.138 Configuración del nodo de comunicación con la plataforma Cloud de Azure con las credenciales adecuadas.

Visto que el protocolo de comunicación seleccionado es MQTT, el siguiente punto, ya si, sería analizar los entresijos de la función que genera la preparación de datos. Por ejemplo, tomando la función para envío de datos del horno se podría ver la siguiente estructura de la Figura 5.139.

```
var temp_fur_azure = global.get("Temperature_information");
var state_fur_azure = global.get("Oven_A_info");

msg.payload = {
    'deviceId': "plcnextdeviot",
    'key': "FV6ldVmygeXV4la/xZbK3VQn2zHLKKeWochhIj/mrQ0=",
    'protocol': "mqtt",
    'data': {"PLCnext" : { "Furnace" : {
        "Temp": temp_fur_azure,
        "State": state_fur_azure
    } }
}}

return msg;
```

Figura 5.139 Función de preparación para envío de datos a plataforma Cloud IBM.

En este caso, comienza exactamente igual que la función para IBM Cloud (toma de datos), pero el JSON no es exactamente el mismo que antes y es que en este caso, parte de la identificación (para mantener el protocolo de seguridad), la identificación de dispositivo que envía el mensaje y el protocolo que se está empleando va en el grueso del mensaje junto con el JSON.

Con esto ya estaría descrito el grueso de la programación de Node-RED asociado al controlador PLCnext. El siguiente paso sería narrar la programación desarrollada para el Node-RED localizado en el entorno Cloud.

5.3.2- Programación de Node-RED en el entorno Cloud

En el entorno Cloud, se ha trabajado con Node-RED tanto para IBM Cloud, como para Azure de Microsoft. La realización del código en ambas plataformas es relativamente semejante y puesto que el servicio de Node-RED no está ya disponible en el entorno IBM Cloud de manera sencilla, es más conveniente centrarse casi directa y exclusivamente en el desarrollo de programa directamente en Azure.

- **IBM Cloud:**

Echando un vistazo rápido a un esquema general que se muestra en la Figura 5.140 del programa que había sido desarrollado en el entorno de IBM Cloud.

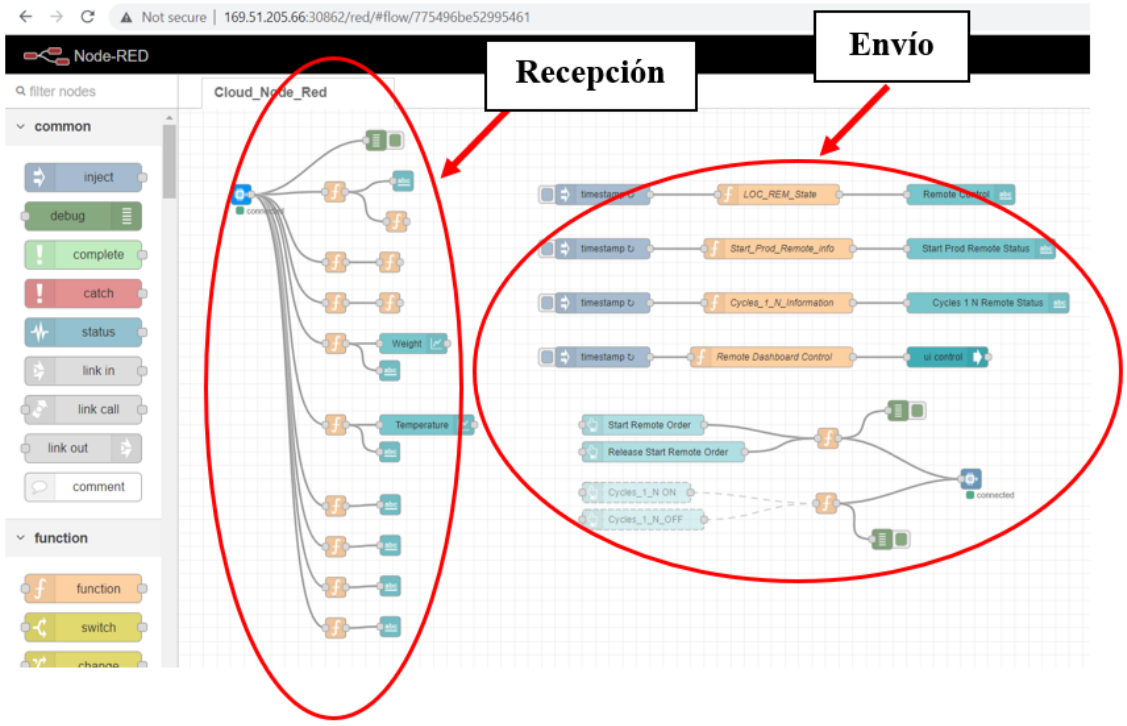

Figura 5.140 Código desarrollado en Node-RED de la plataforma Cloud de IBM.

Explicando a nivel general el programa que se está viendo, a la izquierda se está viendo la recepción de todos los datos enviados desde el controlador PLCnext a IBM Cloud. Por el contrario, a la derecha se está realizando, la representación de los datos sobre el Dashboard y el comandado de planta en modo remoto, por lo que hay que realizar un envío de datos desde el entorno Cloud al controlador PLCnext.

- **Azure de Microsoft**

Vista la programación en IBM Cloud, lo siguiente ya sería ver en mayor detalle la programación desarrollada para Azure, que si bien es cierto la recepción es muy semejante, hay un par de particularidades que se aprovechó a implementar tras el cambio de plataforma Cloud y que resulta muy interesante comentar.

El código desarrollado para la recepción de datos en Azure podría al final distribuirse de manera muy similar al implementado para el caso del controlador comenzando por la

recepción de los datos que previamente se habían enviado desde el controlador como se puede visualizar en la Figura 5.141.

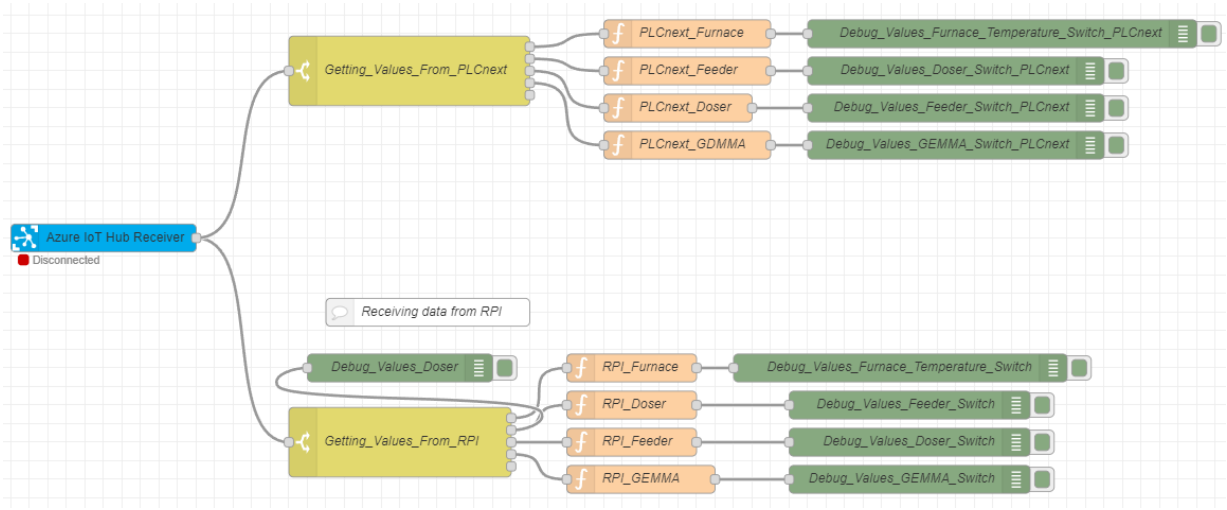

Figura 5.141 Recepción de datos del controlador en el Node-RED de la plataforma Cloud de Azure.

En este caso se puede percibir un caso particular y es que para realizar una prueba más extensa y comprobar ciertos beneficios que presenta el tener en un mismo equipo localizada toda la implementación desde el control al paso de datos a la nube se ven dos ramificaciones. La ramificación superior es la toma de datos enviados por el controlador PLCnext, mientras que la ramificación inferior es la toma de datos enviados por un controlador externo el cual realiza el envío de los mismos datos a través de una Raspberry Pi.

Se ha limitado la recepción de datos a los estados ISA 88 de los diferentes módulos de equipamiento y el estado del GDMMA para realizar una comparativa entre la recepción de datos de ambos controladores puesto que el desarrollo en el controlador externo no era llevado a cabo por el proyectante y estaba siendo desarrollado en un modelo de programa con menos funcionalidades implementadas de las que se han llevado a cabo en el controlador PLCnext para este proyecto.

Analizando la ramificación del controlador PLCnext observada en la Figura 5.142. En este caso se distinguen 3 elementos principales, de izquierda a derecha se tiene un switch que independiza los campos obtenidos en el IoT Hub por parte del controlador PLCnext. Una vez independizados los campos, se pueden observar las funciones de generación de variables globales a partir de las variables que se encuentran dentro de cada campo y finalmente el nodo generador de mensajes de depuración.

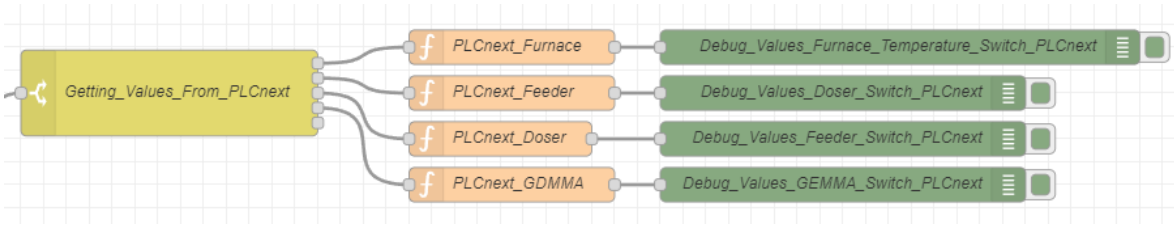

Figura 5.142 Focalización en la recepción de datos del controlador PLCnext.

Viendo en más detalle los nodos. Sobre todo, el nodo switch que ha sido la primera vez que se realiza su descripción. En la Figura 5.143 se encuentran los diferentes campos enviados por el controlador PLCnext (aquellos que habían sido enviados en formato JSON).

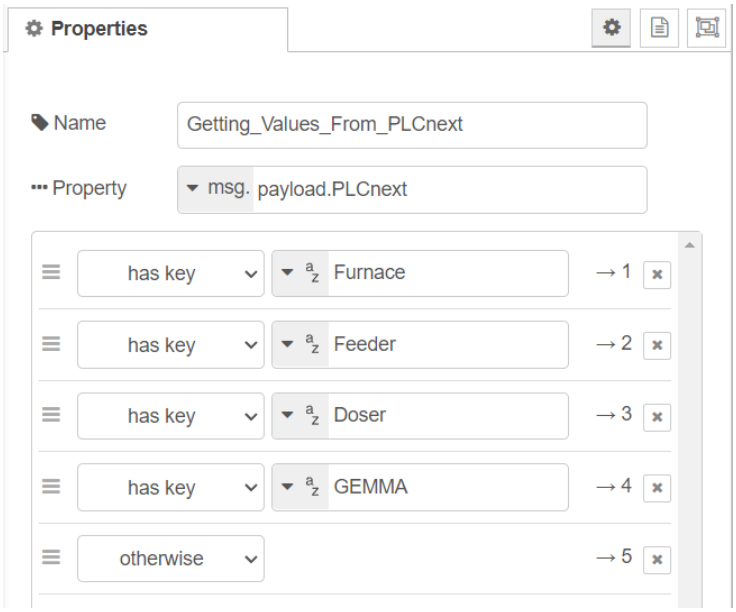

Figura 5.143 Configuración del nodo switch en el Node-RED de la plataforma Cloud de Azure.

Una vez separados los campos, ya solo quedaría generar las variables globales para posteriormente tratar en el programa según las preferencias del usuario. En el caso de la Figura 5.144, se está viendo la generación de las variables globales para las variables de temperatura y estado ISA 88 asociadas al horno.

```
1
2  global.set("Temp_Furnace_PLCnext",msg.payload.PLCnext.Furnace.Temp);
3  global.set("State_Furnace_PLCnext",msg.payload.PLCnext.Furnace.State);
4
```

Figura 5.144 Generación de variables globales con los datos recibidos de la plataforma Cloud de Azure.

Para el caso de la recepción de datos por parte del controlador externo, como ya se veía previamente, sigue la misma estructura de recepción que el controlador PLCnext, pero se puede comprobar en detalle que esto es así en la Figura 5.145.

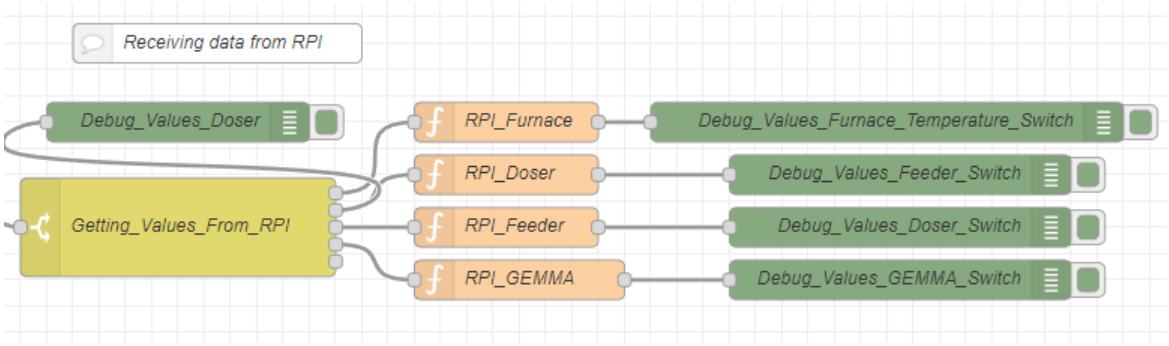

Figura 5.145 Focalización en la recepción de datos del controlador externo.

Si bien es cierto que el esquema de recepción es el mismo, la tasa de envío de mensajes por parte del controlador PLCnext y el controlador externo es distinta, siendo la de este último mucho más elevada que la de PLCnext. Puesto que la transferencia de mensajes diarios en Azure (de manera gratuita) está acotada a un valor máximo y el controlador externo no está en manos del proyectante, se ha optado por desarrollar una pequeña función que indica al controlador externo que envíe o no información a la plataforma Cloud.

La estructura de manejo de recepción de información por parte del controlador externo puede verse reflejada en la Figura 5.146. En el proceso se usa un nodo de inyección el cual se acciona desde el propio entorno de desarrollo gráfico puesto que es una acción que no conviene dejar con acceso de cualquier usuario público en el Dashboard (recordar que el entorno de programación en sí está protegido, pero el Dashboard es de libre consulta porque solo se emplea para representación de datos).

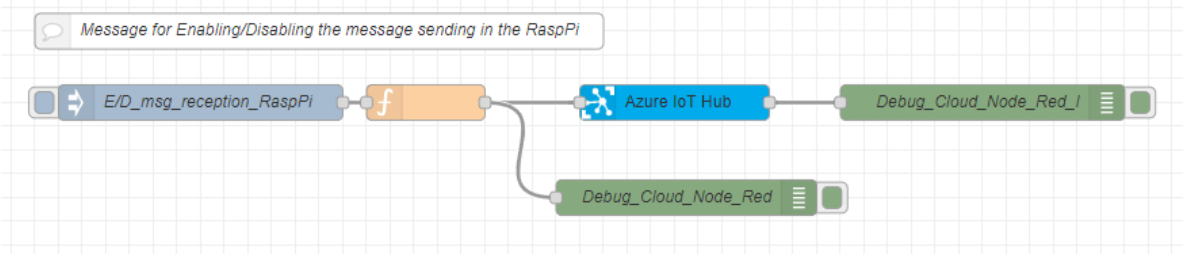

Figura 5.146 Código para habilitación de recepción de mensajes por parte del controlador externo.

El nodo de inyección temporal está directamente unido a una función implementada que prepara un mensaje tratado visualizado en la Figura 5.147 que es entendido por el controlador externo para decidir si envía mensaje o no (es decir, el controlador externo siempre está a la escucha). El funcionamiento sería tal que, si el controlador externo no está enviando mensajes, si recibe una activación sobre esta función que se acaba de describir, el controlador comenzará a enviar mensajes, mientras que, si está enviando mensajes, si se vuelve a actuar sobre ella, parará de enviarlos.

```
msg.payload = {

    "deviceId": "CloudNodeRedDevice",


    "key": "3tbjBkb6ZmAdvJ1kS+lJEpNumgn4mCekF2AWvUv2Izw=",


    "protocol": "mqtt",


    "data": { "RPI":{"toggle_IOTHUB":true}}
}

return msg;
```

Figura 5.147 Código asociado a la función de habilitación de recepción de mensajes del controlador externo.

Finalmente, el último punto a describir en este apartado de programación de Node-RED en la plataforma Cloud es la representación sobre el Dashboard de los valores recibidos tanto por parte del PLCnext, como por parte del controlador externo.

El caso de las variables recibidas por parte de PLCnext se representan de la manera en la que se puede observar en la siguiente Figura 5.148.

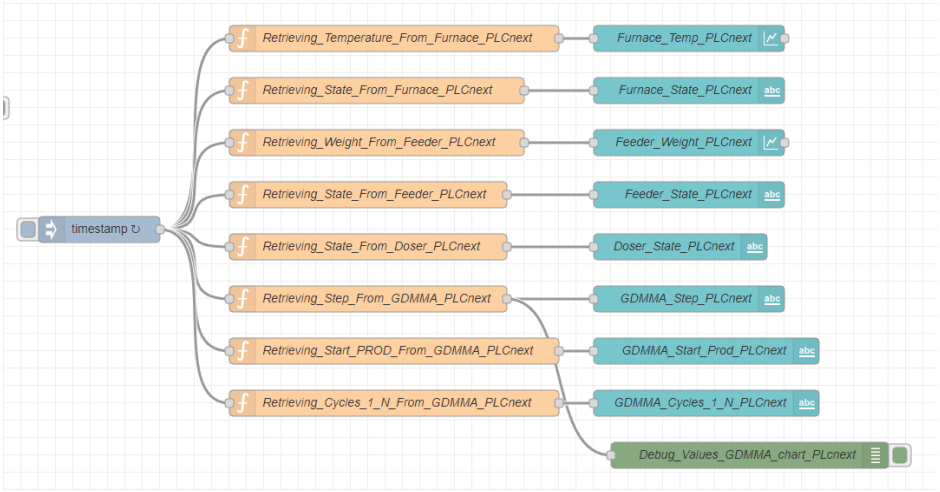

Figura 5.148 Representación sobre Dashboard de datos recibidos por el controlador PLCnext.

En este caso simplemente se realiza un caso semejante al ya visto para cualquier representación gráfica descrita en el controlador PLCnext. Es decir, se realiza una inyección temporal de variables al Dashboard de manera periódica a través de una función que toma la variable global que se desea representar (por ejemplo, la temperatura del horno en la Figura 5.149) y esta se conecta al nodo de representación en el Dashboard adecuado.

```
1  msg.payload = global.get("Temp_Furnace_PLCnext");
2  return msg;
```

Figura 5.149 Función de toma de datos de la variable global para su representación en el Dashboard.

Para el caso del controlador externo, como ya se ha dicho, es exactamente igual. Se muestra la estructura gráfica en la Figura 5.150.

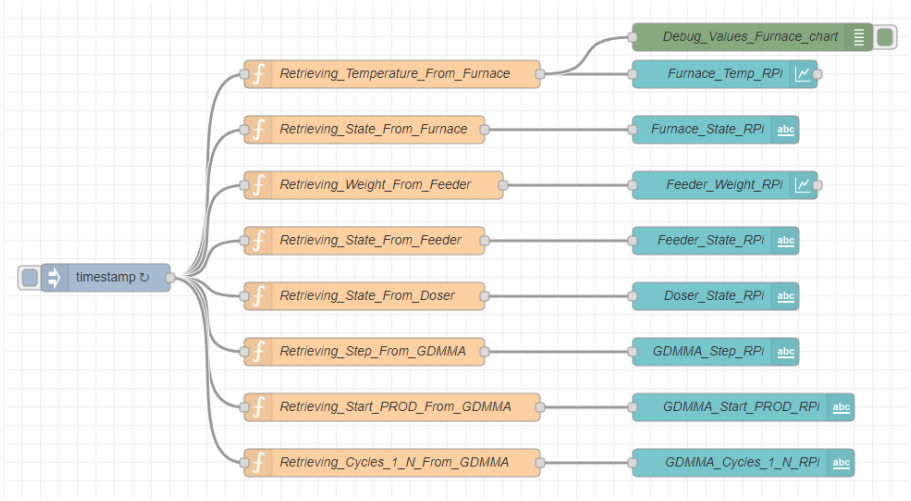

Figura 5.150 Representación de datos recibidos por el controlador externo en el Dashboard.

Antes de finalizar, cabe recalcar que puede parecer que el envío de información a la nube por parte del controlador PLCnext es igual que el controlador externo, pero este no es para nada el caso. Mientras que en PLCnext se puede realizar una interacción directa controlador-nube como se muestra en la Figura 5.151.

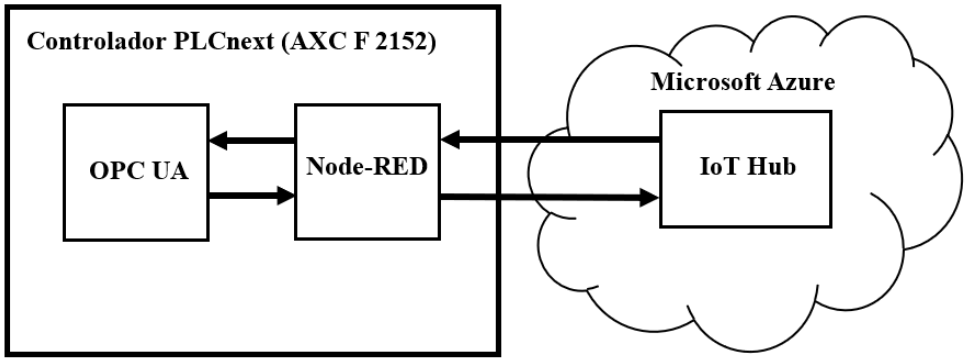

Figura 5.151 Esquema simplificado de interacción de datos entre controlador PLCnext y plataforma Cloud.

En el caso de un controlador externo, se requeriría de un intermediario como en este caso ha sido el uso de una Raspberry Pi, como se muestra en la Figura 5.152.

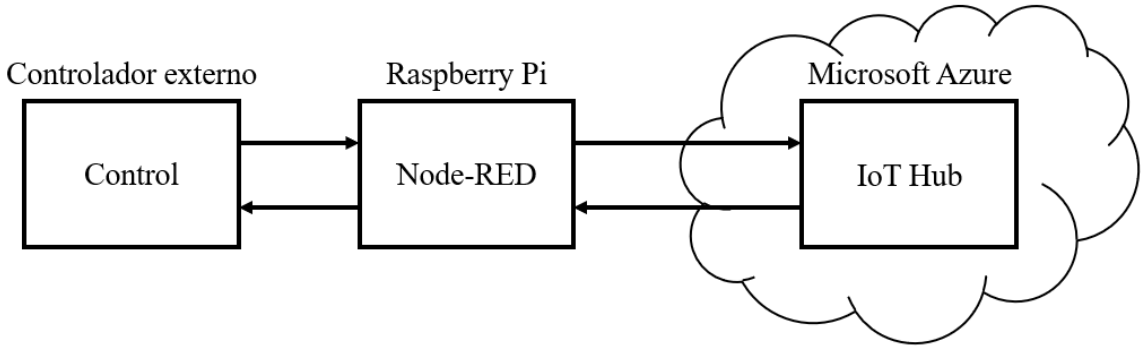

Figura 5.152 Esquema simplificado de interacción de datos entre controlador externo y plataforma Cloud.

6. Manual de usuario

Tras haber desarrollado la explicación de la programación se presenta en este documento un resumen de la manera de proceder con la operación del sistema una vez realizado el arranque.

6.1.- ARRANQUE DEL SISTEMA EN PLCNEXT

Para acotar la explicación, no reiterar dos veces lo mismo y proceder directamente, desde el controlador físico, se va a explicar el manual del operador desde su arranque en el controlador PLCnext. Es decir, si se quisiese operar en el modo de simulación de Codesys, es seguir los mismos pasos que se describirán para PLCnext.

Lo primero que hay que hacer es realizar la transferencia del programa de PLCnext Engineer: "PLCnext_Gabriel_HORNO_PLCnext_2022.6.0_Proficloud_funcionado.pcwex" al controlador físico, como se muestra en la Figura 6.1.

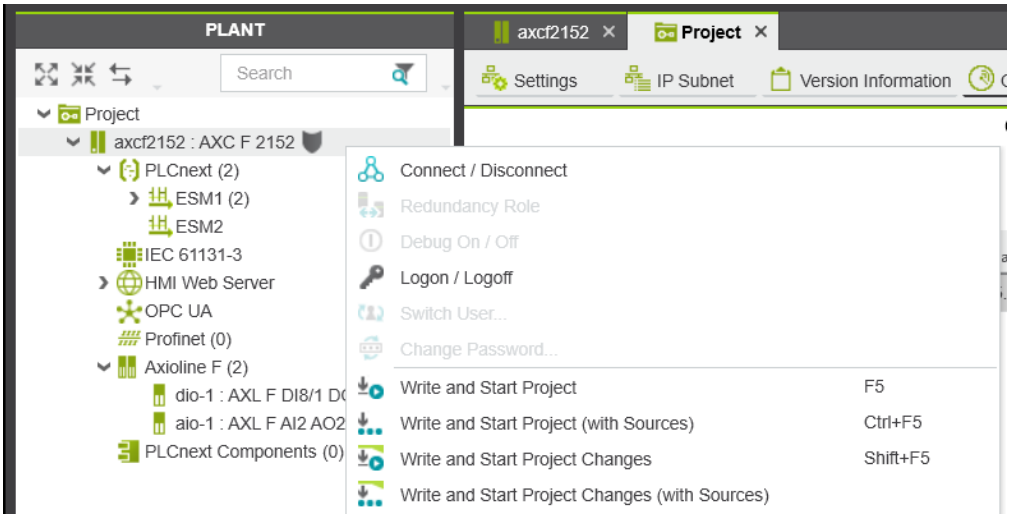

Figura 6.1 Transferencia de programa a controlador físico desde PLCnext Engineer.

Realizada la transferencia de programa al controlador físico de PLCnext, lo primero que habrá que realizar es comenzar con el acceso al HMI Web Server. Para ello, se debe introducir la IP del equipo en cualquier dispositivo (ordenador, tablet o móvil) en el

navegador de preferencia del usuario. Se mostrará entonces la pantalla reflejada en la Figura 6.2 en la que el usuario tendrá que identificarse con las credenciales (usuario y contraseña) asociadas al controlador.

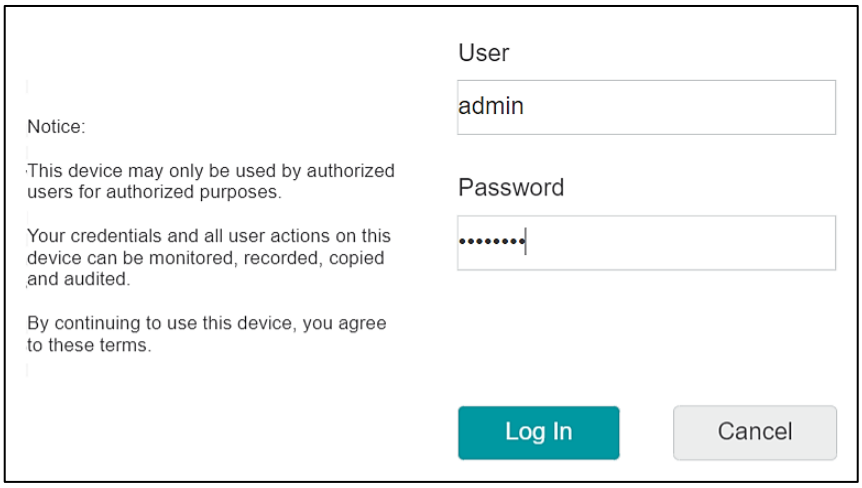

Figura 6.2 Petición de credenciales al usuario para acceder al HMI localizado en el Web Server.

Tras realizar la identificación, lo primero que se verá es la pantalla mostrada en la Figura 6.3, designada como principal por defecto (pantalla del GDMMA). La planta arranca en el estado A5 (preparación de arranque tras fallo) con la seta de emergencia pulsada (EM) y los módulos de equipamiento en el estado inicial de ISA88 (ver a la derecha de la imagen).

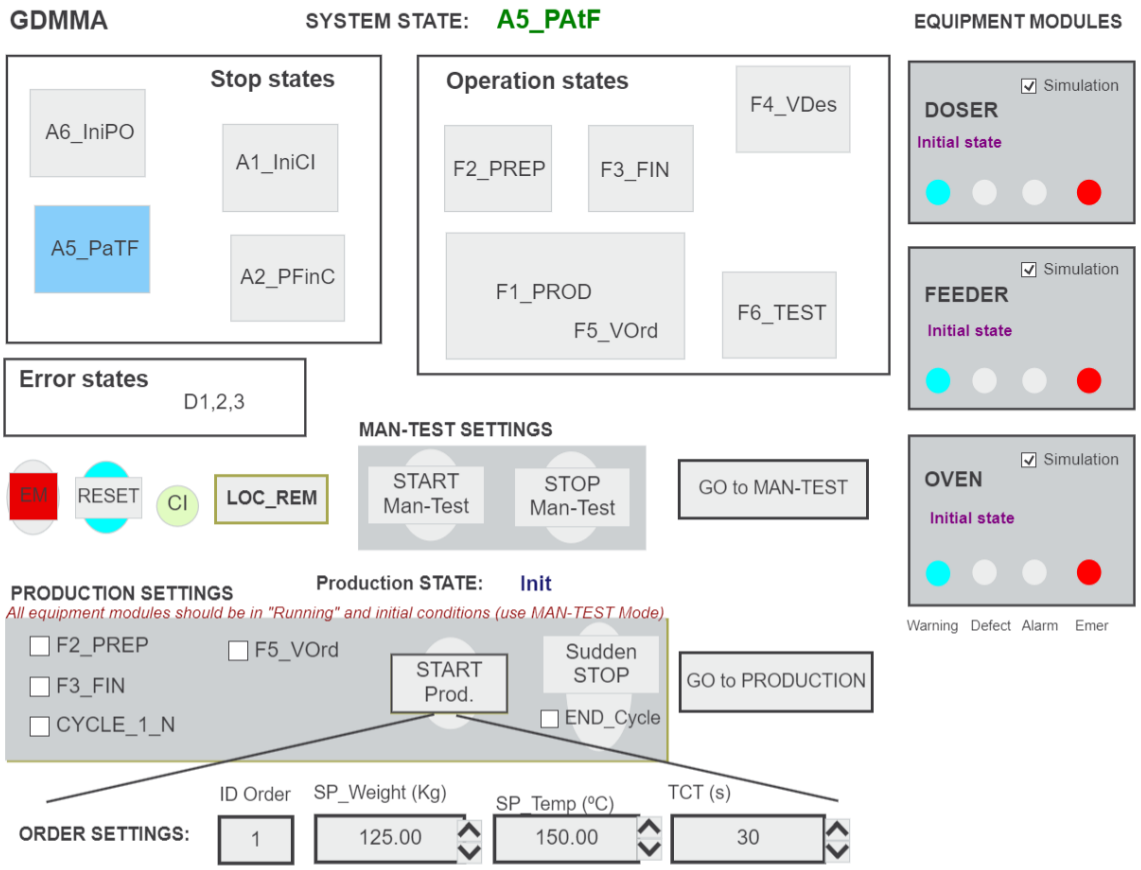

Figura 6.3 Pantalla asociada al GDMMA del proyecto.

El siguiente paso será liberar la seta de emergencia (EM), pulsar RESET y realizar el arranque (poner en Running de ISA 88) todos los módulos de equipamiento. Para arrancar los módulos de equipamiento, hay que acceder a la pestaña de verificación en desorden (pulsando en GO to MAN_TEST) y realizar el arranque de cada uno de los módulos en esta ventana (ver Figura 6.4).

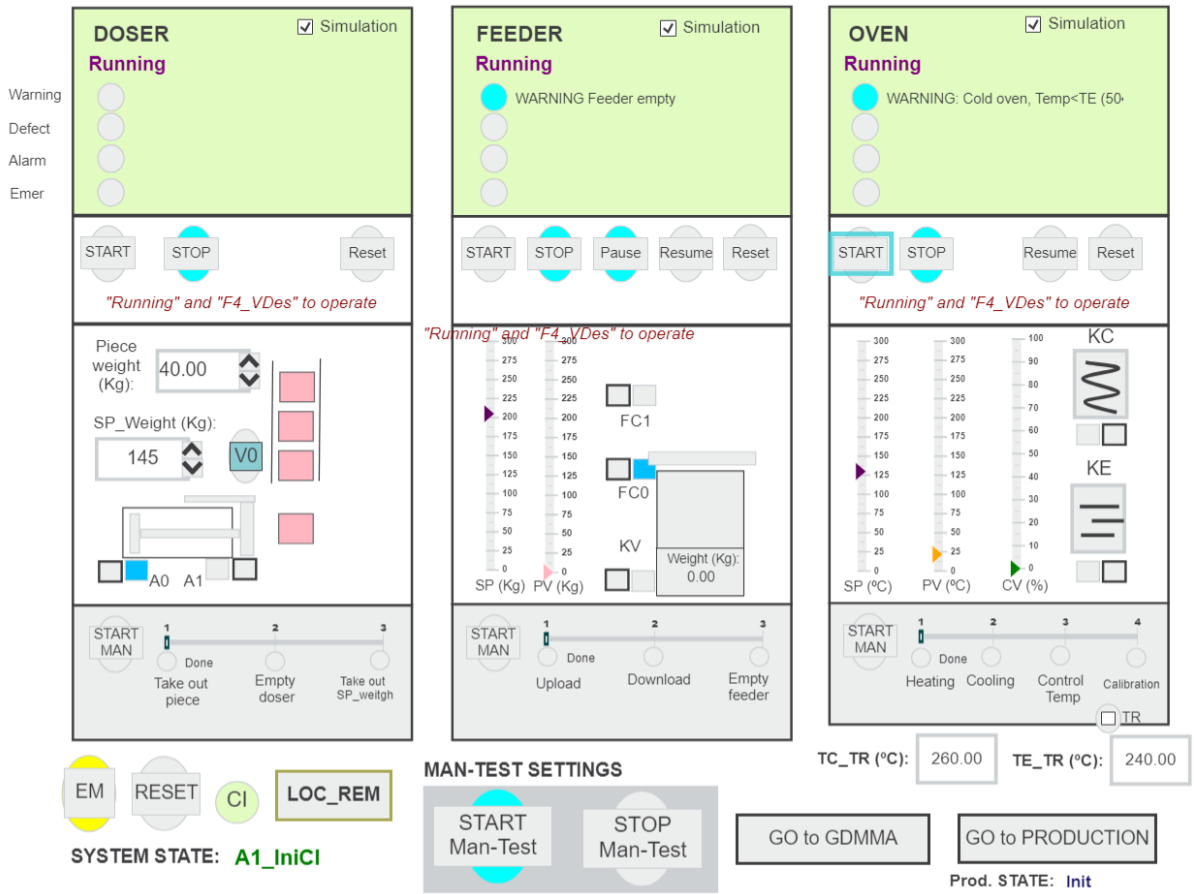

Figura 6.4 Sistema con cada uno de los módulos de equipamiento en estado Running (de ISA88) y preparado para operar.

Para arrancar el módulo de equipamiento correspondiente al Alimentador (DOSER) hay que activar la detección de piezas en el contenedor (VO), y pulsar START. Para arrancar el Cargador (FEEDER) o el Horno (OVEN) basta pulsar en el START correspondiente a cada uno de ellos. Tras completar los pasos, se tiene que observar la ventana como se ve en la Figura 6.4.

En todo momento, las marcas azules que rodean algunos botones representan la actuación coherente que el usuario puede realizar, guiando con ello de forma bastante sencilla la forma de interactuar con el sistema. Por ejemplo, se observa que el cargador estando en modo Running se podría para y pausar, o la orden general para activar el modo MAN-TEST SETTING es posible realizarla pulsando en START Man-Test (ver Figura 6.5).

Figura 6.5 Observado en detalle de las marcas destacadas en azul que guían al usuario a operar con la planta.

Si se arrancase el modo verificación en desorden, la pantalla cambiaría en aspecto como se muestra en la Figura 6.6.

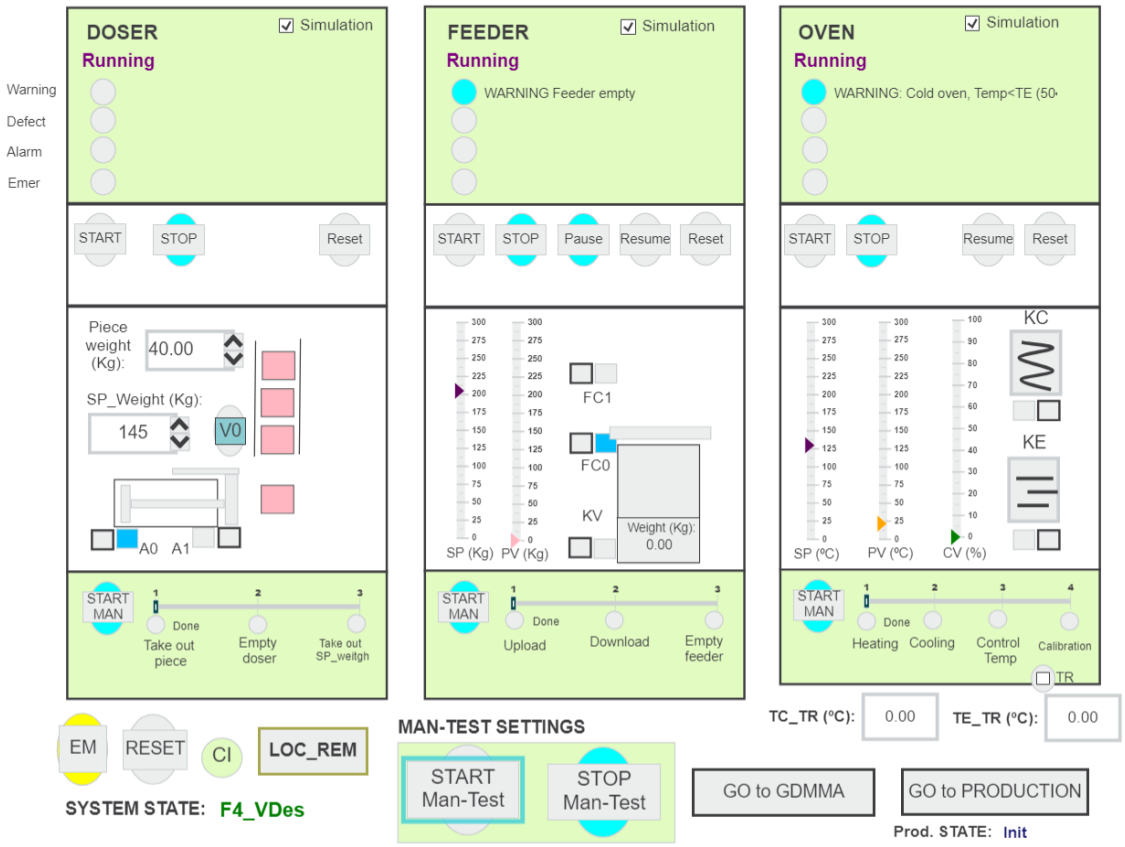

Figura 6.6 Arranque de F4 (verificación en desorden).

Si se accediese a la producción (GO to PRODUCTION), y se arranca (START Prod.) se comienza directamente con la ejecución de la receta especificada por el usuario como se representa en la Figura 6.7.

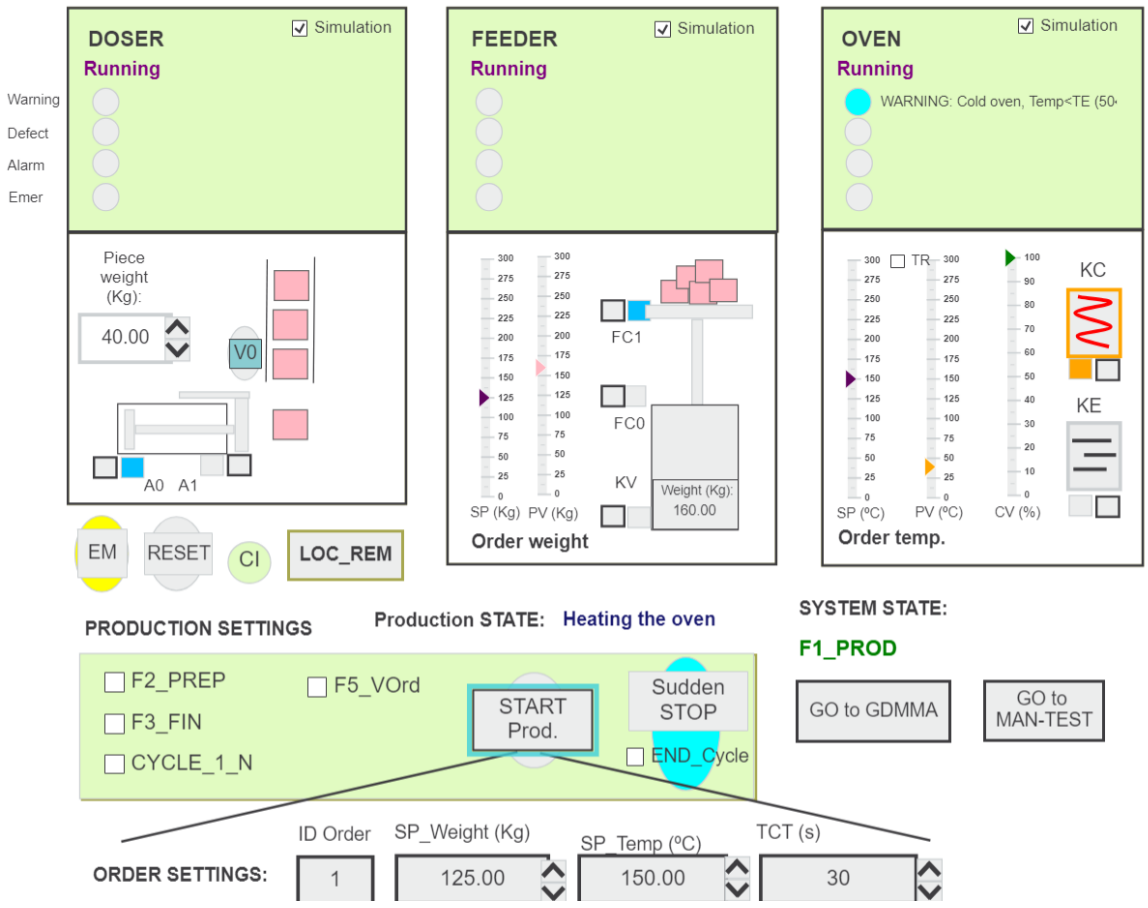

Figura 6.7 Planta en F1 (producción normal).

La receta (orden de pedido, ORDER SETTINGS) consiste en un identificado (ID Order) una cantidad de peso de piezas a tratar (SP_Weight, en Kg), una temperatura del horno (SP_Temp, en ºC) y un tiempo de control o mantenimiento de dicha temperatura (TCT, en s).

Una de las posibilidades de manejo del sistema, aparte de los ya introducidos, será poner el sistema en modo remoto para que la producción pueda ser lanzada desde Node-RED. Para configurar este modo remoto, habrá que pulsar en LOC_REM y la pantalla de

operador procederá a presentar la disposición de la Figura 6.8, donde se aprecia que el cambio a fondo azul significativo de establecimiento del modo de control remoto.

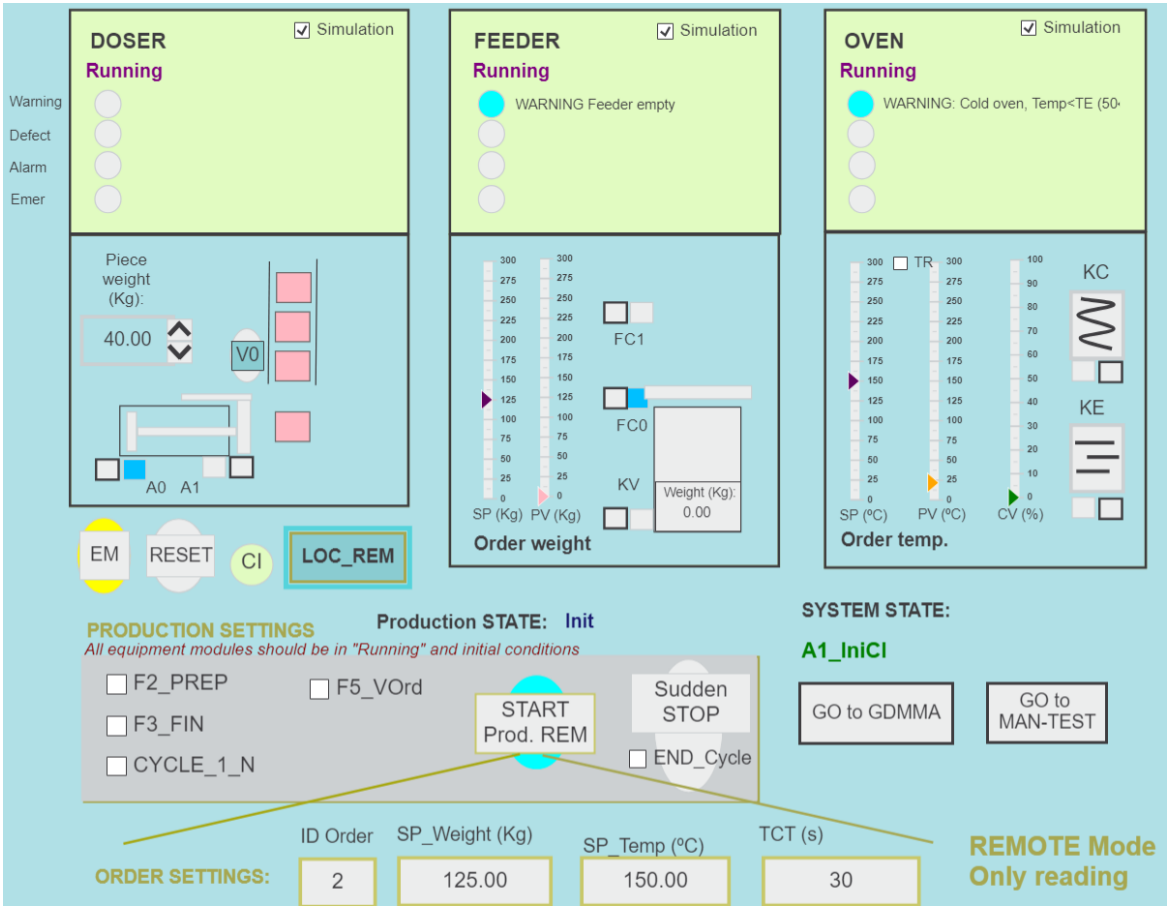

Figura 6.8 Planta en modo remoto (se controlaría desde el Dashboard de Node-RED localizado en el controlador).

Finalizadas las tareas sobre el controlador, al igual que con cualquiera de las descritas para las plataformas Cloud o el Node-RED desarrollado a nivel local, bastará con salir del navegador sobre el que se haya abierto la sesión. Para acabar cualquier tarea definitiva sobre el controlador físico, lo que habrá que realizar el cortar la tensión al mismo.

6.2.- MANEJO DEL DASHBOARD DESARROLLADO EN NODE-RED PARA EL CONTROLADOR PLCNEXT

Para acceder al Dashboard de Node-RED implementado, al igual que se ha realizado para acceder al HMI Web Server, se ha de introducir en el navegador la dirección IP del controlador PLCnext seguido del puerto donde se localice Node-RED, y terminando en *ui*, es decir:

https://192.168.0.16:1880/ui

Se mostrará entonces una primera pantalla en la que se aparecen los valores de temperatura, peso, si el sistema se encuentra en local o remoto, los estados GDMMA e ISA88 de los módulos de equipamiento, según se aprecia en la Figura 6.9.

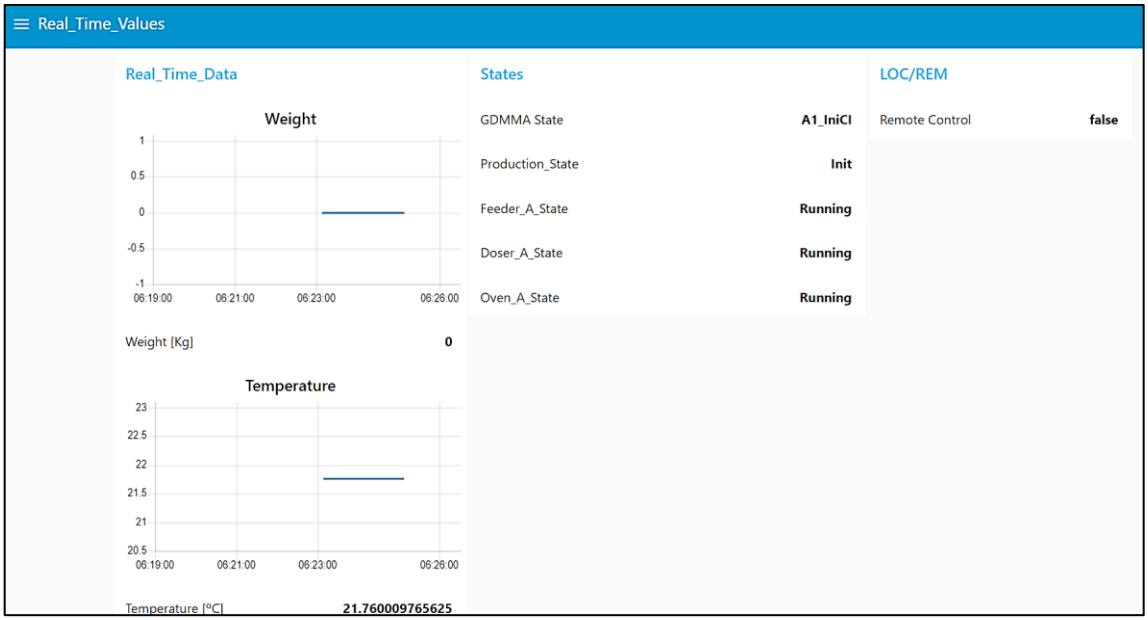

Figura 6.9 Pestaña del Dashboard destinada a la muestra de valores de temperatura y peso, estados GDMMA e ISA88 e indicación de modo local/remoto.

Si se accediese a la segunda pantalla de operador, se dispondría de la información de pedido (pedido en preparación, en funcionamiento y último pedido realizado). Por defecto,

si no está ningún pedido en operación aparecerá solo en pantalla los datos del pedido en preparación y último pedido realizado como se muestra en la Figura 6.10.

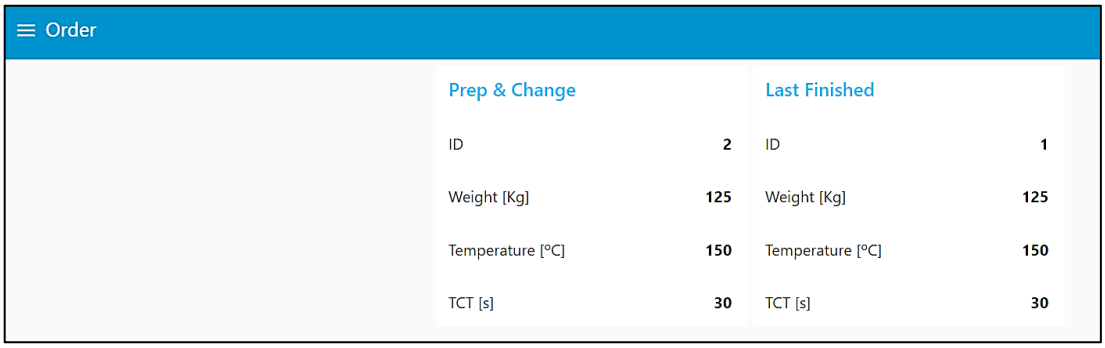

Figura 6.10 Parte de Dashboard destinada a muestra de pedido (I).

En el momento que se lance un pedido, aparecerá en la pantalla, el pedido que se está en proceso de ser completado, más los otros dos que ya aparecían como se indica en la Figura 6.11.

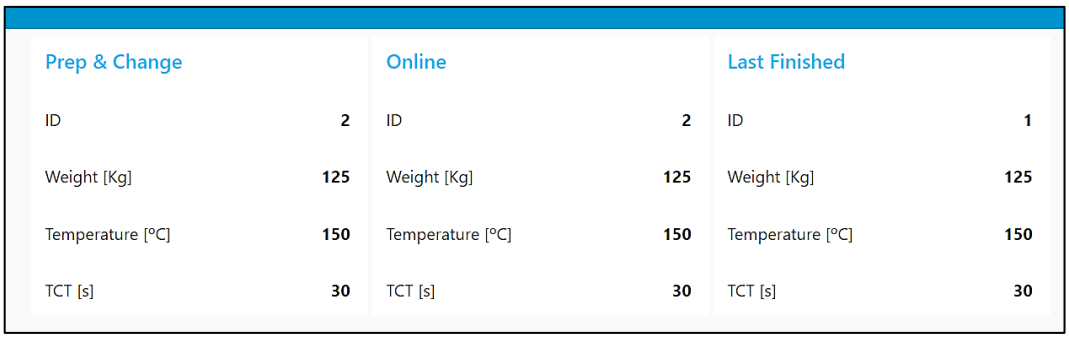

Figura 6.11 Parte de Dashboard destinada a muestra de pedido (II).

Finalmente, se podrá acceder a la pestaña de realización de pedido en remoto. Por defecto, si el modo remoto no está activo, aparecerá representado de la siguiente manera vista en la Figura 6.12.

Figura 6.12 Pestaña Dashboard de Node-RED indicando modo remoto no operativo.

En el momento que el modo remoto esté activo, se mostrarán por pantalla todas las acciones que tiene disponible el usuario para actuar y modificar los parámetros para preparar la receta y lanzar el pedido tal y como se muestra en la Figura 6.13.

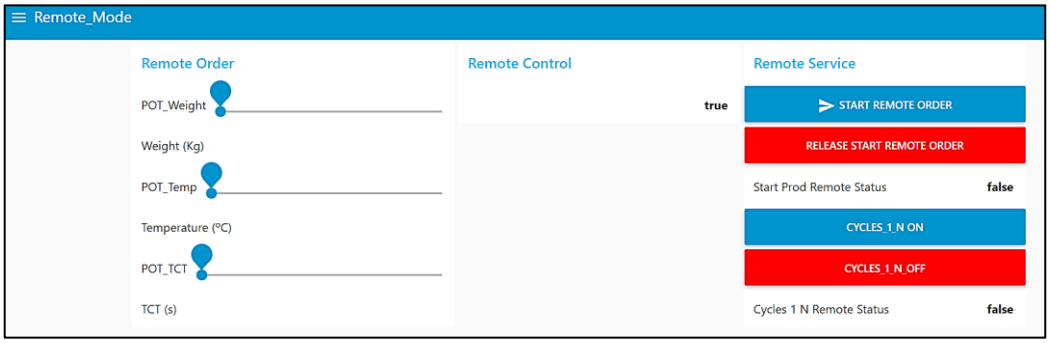

Figura 6.13 Parte del Dashboard destinada a la preparación de pedido y lanzamiento en modo remoto.

Si se lanzase un pedido en remoto y haciendo referencia a la pantalla de operador desarrollada con HMI Web Server de PLCnext Engineer ya previamente descrita, se verá que en este modo remoto pasará a tener un indicativo más en verde, como se puede observar en la Figura 6.14.

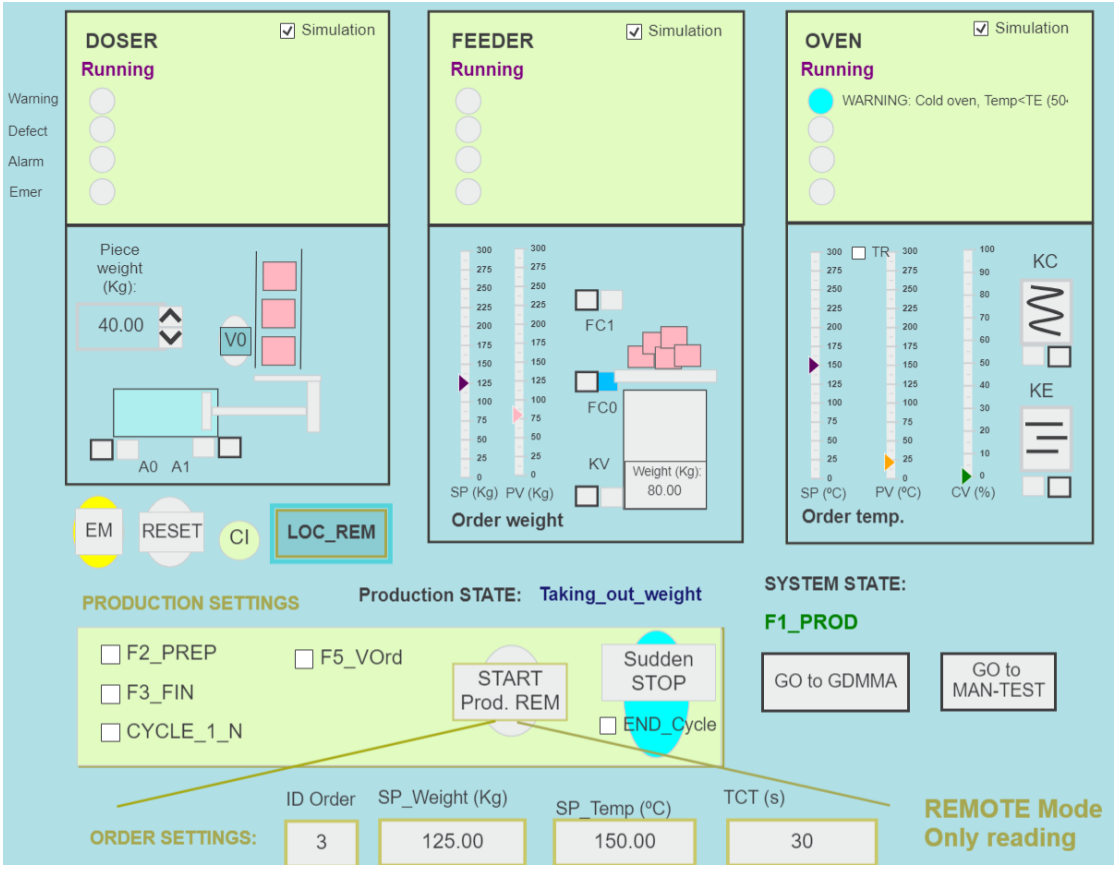

Figura 6.14 HMI mostrando F1 (producción normal) operando con un pedido lanzado en modo remoto.

6.3.- MANEJO DE DASHBOARD NODE-RED EN LA NUBE

- **Proficloud.io**

Con respecto al apartado Cloud, comenzando con Proficloud.io con los valores de temperatura y peso probados, se podrá generar una customización de Dashboard como se ve en la Figura 6.15 y agruparlos bajo una sola pestaña para observar su evolución como se muestra en la Figura 6.16.

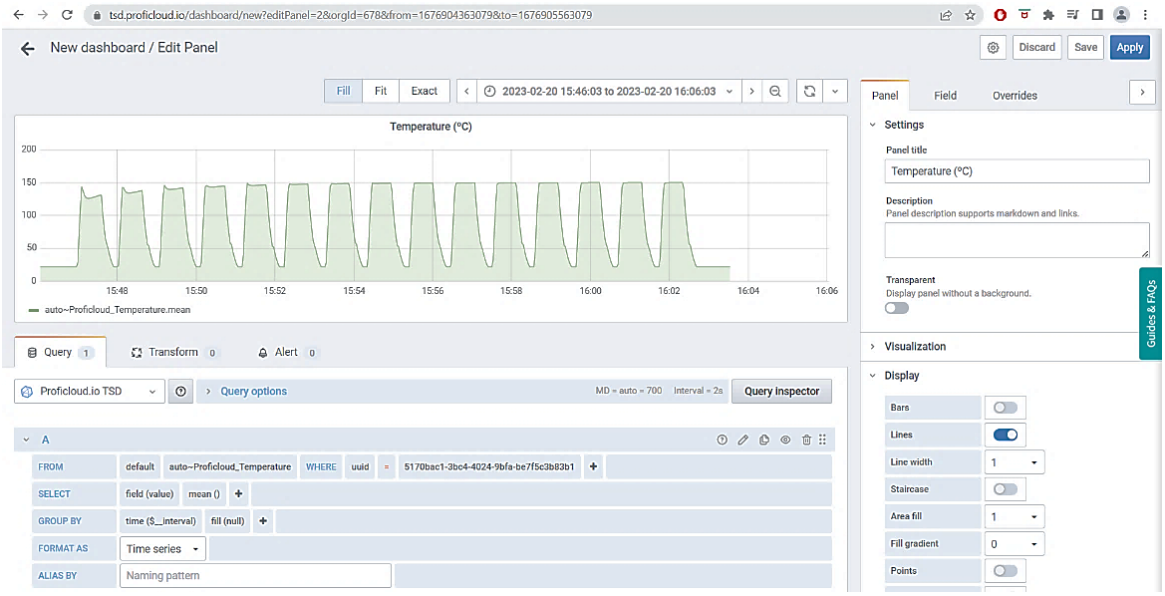

Figura 6.15 Pestaña de edición de Dashboard en Proficloud.io.

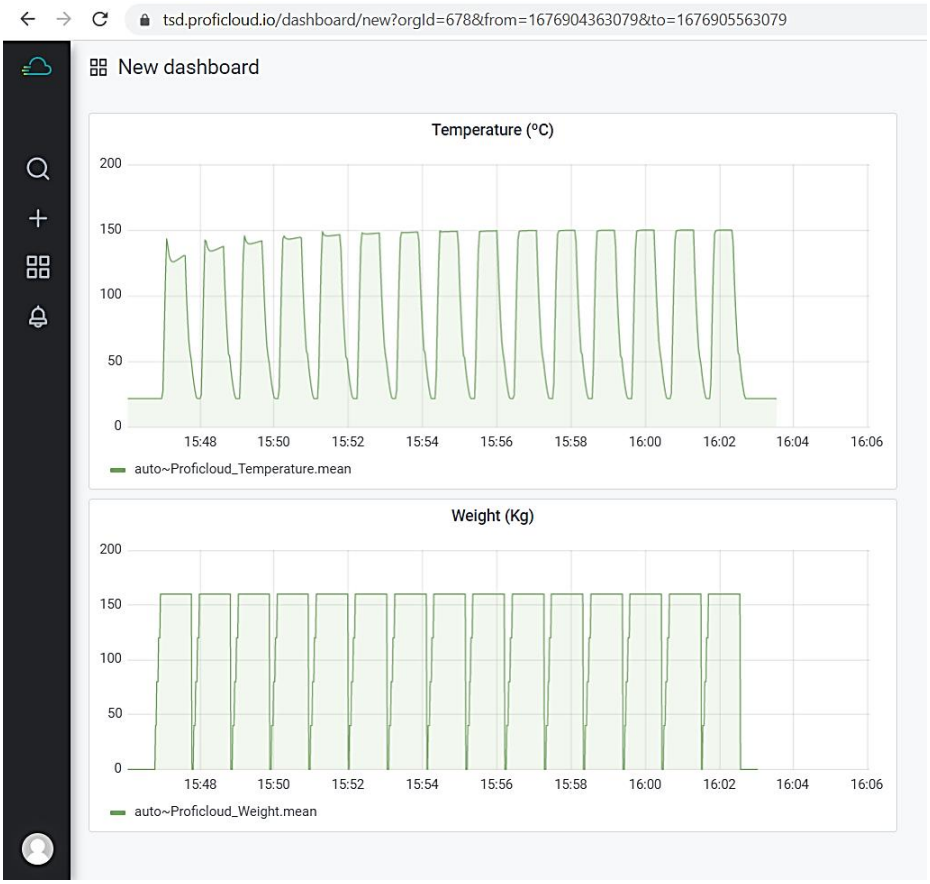

Figura 6.16 Visualización de datos de temperatura y peso en Proficloud.io.

- **IBM Cloud**

Para IBM Cloud, aunque ya no se dispone de las funcionalidades de Node-RED que expandían en gran medida las posibilidades con esta plataforma Cloud. Sigue operativo un Dashboard que muestra información referente a pedido en ejecución, valores de temperatura y peso e indicación de si el sistema está habilitado para operación en remoto. Se puede ver reflejado sobre la Figura 6.17.

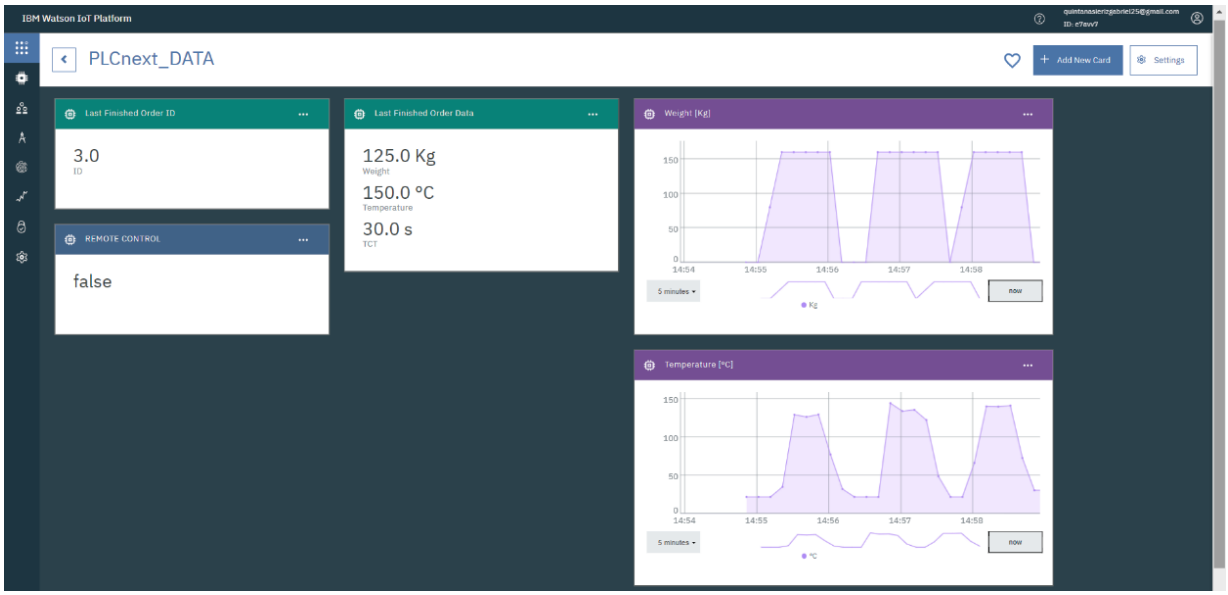

Figura 6.17 Dashboard generado en IBM Cloud (visualización de ID de pedido, local/remoto, receta de pedido, temperatura y peso).

- **Azure de Microsoft**

Finalmente se ha desarrollado un Dashboard en el Node-RED localizado en la plataforma Cloud de Azure (puesto que no se podía crear un Dashboard intermedio externo a Node-RED), el cual mostrará información referente al controlador PLCnext como se muestra en la Figura 6.18, también información asociada al controlador externo que se empleó como elemento comparador como aparece reflejado en la Figura 6.19 y una pestaña final visualizada en Figura 6.20 que compara estados del GDMMA entre ambos controladores.

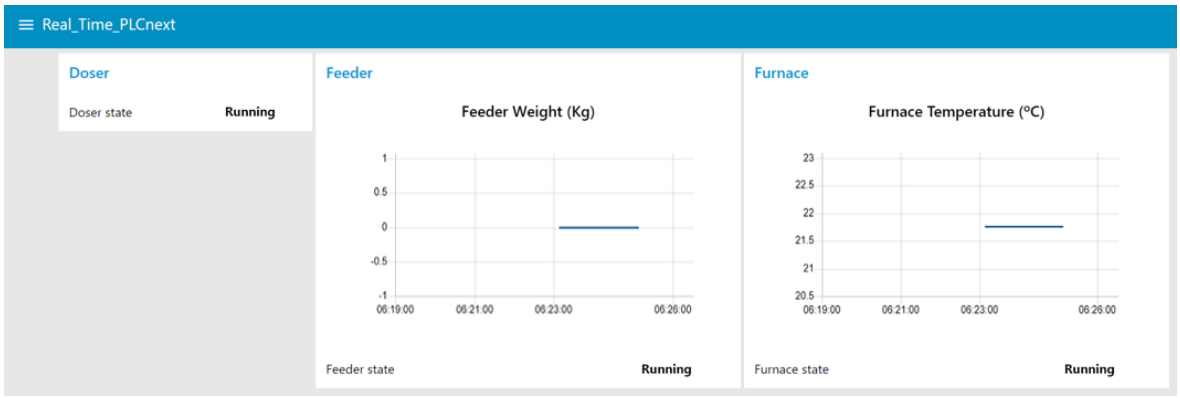

Figura 6.18 Dashboard en Node-RED de plataforma Cloud de Azure para visualización de datos del controlador PLCnext.

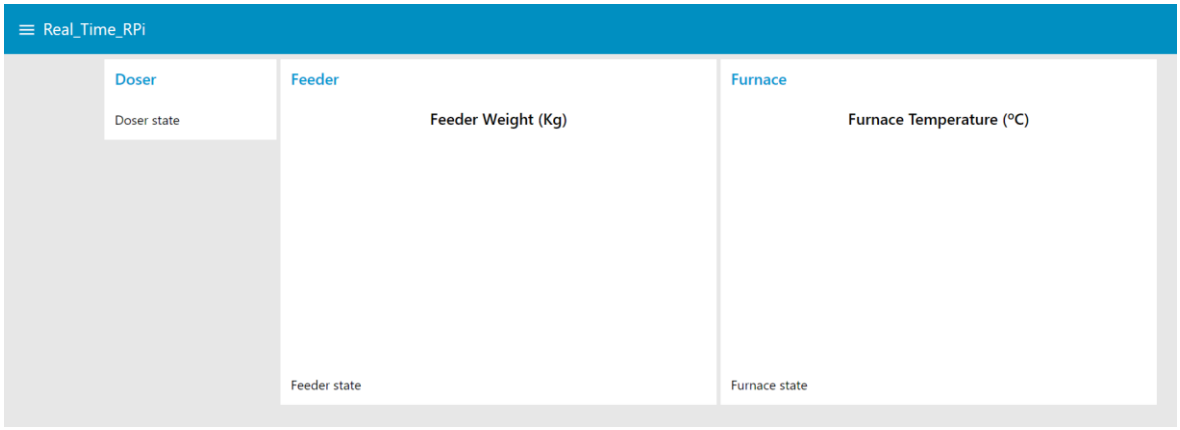

Figura 6.19 Dashboard en Node-RED de plataforma Cloud de Azure para visualización de datos del controlador externo.

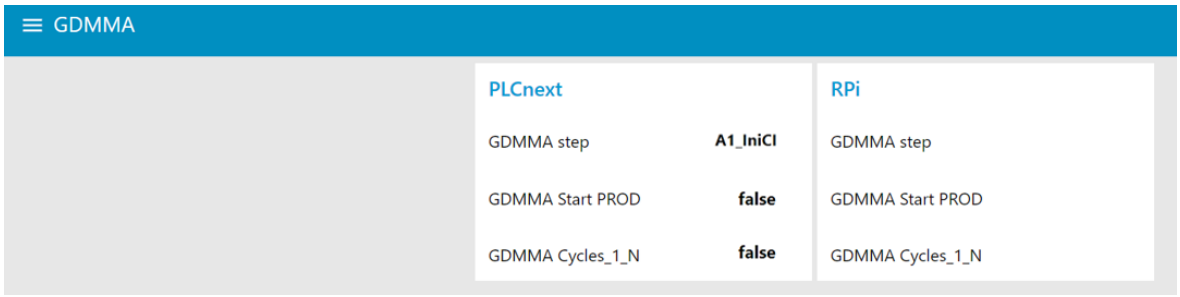

Figura 6.20 Dashboard en Node-RED de plataforma Cloud de Azure para visualización de datos GDMMA del controlador PLCnext y externo.

7. Planificación de proyecto

7.1.- DESARROLLO TEMPORAL DEL PROYECTO

Inicialmente el proyecto se comenzó disponiendo del desarrollo de una aplicación realizada con Codesys en un estado notablemente avanzado respecto a la implementación final, el cual había sido realizado por el tutor. En base a ese programa base se realizó un estudio detallado de la aplicación, ampliando y mejorando algunos aspectos. Paralelamente se abordó la adaptación de toda la programación al entorno PLCnext Engineer, también en estrecha colaboración con el tutor debido a la envergadura y complejidad del programa. Completado este paso, el autor desarrolló de manera totalmente autónoma la adaptación del programa de PLCnext para operar con el servidor OPC UA y Proficloud.io y el desarrollo de todos los elementos necesarios para que el controlador PLCnext funcionase con aplicación en Node-RED y la integración con plataformas Cloud externas a Phoenix Contact.

El proyecto se ha visto muy condicionado a un desarrollo rápido del código por dos Workshops internacionales en los que participaron tutor y proyectante donde se ha presentado las claves de diseño y el funcionamiento completo del sistema ante docentes y estudiantes universitarios.

El listado de las etapas y tareas asociadas al proyecto se ven reflejadas en el siguiente listado:

- **Labores previas:** En los que se engloba el análisis del ejemplo de partida del sistema a desarrollar y definir junto con el tutor los objetivos y tareas a realizar para completar el proyecto.

- **Finalización de la programación en Codesys:** Se completa el ejemplo en desarrollado en Codesys para ajustarse a los objetivos definidos y preparar un programa completo para su paso al entorno de PLCnext Engineer.

- **Configuración del controlador PLCnext físico:** En este apartado no se abarcan solo las configuraciones en el entorno PLCnext, si no también todas las tareas a realizar internamente sobre el controlador para prepararlo a operar con el software externo mencionado en el documento.

- **Adaptación de programa a PLCnext Engineer**: Comprende la edición y verificación del programa, inicialmente implementado en Codesys, adaptándolo al entorno PLCnext Engineer.

- **Selección de plataforma Cloud**: Comprende los procesos de selección de la plataforma Cloud externa a Proficloud.io, tanto inicialmente la selección de IBM Cloud, como el posterior cambio a Azure de Microsoft.

- **Desarrollo de código en Node-RED para el controlador:** Abarca la implementación del código inicialmente desarrollado para comunicación vía OPC UA entre controlador y entorno Node-RED, para interacción con Dashboard local y operar con IBM Cloud, como el posteriormente desarrollado para Azure.

- **Desarrollo de código en Node-RED para el entorno Cloud:** Recoge la implementación de código ya desarrollado sobre la plataforma Cloud. En primera instancia para IBM Cloud, como finalmente para Azure.

- **Integración global y pruebas finales:** Realización de carga de todos los programas sobre el entorno físico y prueba de la interacción y correcto funcionamiento del sistema.

- **Elaboración de la documentación asociada al proyecto.**

Para la representación temporal del listado de tareas, se ha empleado la herramienta Gantt Project. Inicialmente, se presenta en la Tabla 7.1 un listado detallado de las tareas.

LISTADO DE TAREAS	INICIO	FINAL	DURACIÓN
Labores previas	01/10/2021	08/10/2021	**8**
Análisis del sistema de partida y definición de objetivos	01/10/2021	08/10/2021	8
Finalización de la programación en Codesys	08/10/2021	20/10/2021	**13**
Completado de la programación de Codesys en base a los objetivos definidos	08/10/2021	20/10/2021	13
Configuración del controlador PLCnext físico	21/10/2021	10/03/2023	**57**
Configuración del controlador en PLCnext Engineer	21/10/2021	26/10/2021	6
Configuraciones externas del controlador I	21/10/2021	18/11/2021	29
Configuraciones externas del controlador II	17/02/2023	10/03/2023	22
Adaptación de programa a PLCnext Engineer	26/10/2021	03/03/2023	**32**
Transcripción de código de Codesys a PLCnext Engineer	26/10/2021	11/11/2021	17
Adiciones extras al código original sobre PLCnext Engineer (Proficloud.io)	17/02/2023	03/03/2023	15
Selección de plataforma Cloud	21/10/2021	17/02/2023	**26**
Selección de la plataforma Cloud I	21/10/2021	29/10/2021	9
Selección de la plataforma Cloud II	01/02/2023	17/02/2023	17
Desarrollo de código en Node-RED para el controlador	26/10/2021	21/03/2023	**58**
Desarrollo de código para operación con Dashboard local y plataforma Cloud I	26/10/2021	19/11/2021	25
Desarrollo de código para operación con plataforma Cloud II	17/02/2023	21/03/2023	33
Desarrollo de código en Node-RED para el entorno Cloud	01/11/2021	21/03/2023	**52**
Desarrollo de código en la primera plataforma Cloud seleccionada	01/11/2021	19/11/2021	19
Desarrollo de código en la segunda plataforma Cloud seleccionada	17/02/2023	21/03/2023	33
Integración global y pruebas finales	05/06/2023	09/06/2023	**5**
Elaboración de la documentación asociada a proyecto	21/03/2023	07/07/2023	**109**

Tabla 7.1 Listado de tareas asociadas al proyecto

A cada una de las tareas generales se les puede aplicar una media de dos horas dedicadas diariamente, dando un total de: 718 horas dedicadas. Una vez visto en detalle la distribución temporal de las tareas, en la Figura 7.1 del siguiente subapartado se presenta la distribución gráfica temporal de dichas tareas sobre un diagrama de Gantt.

7.1.1- Diagrama de Gantt

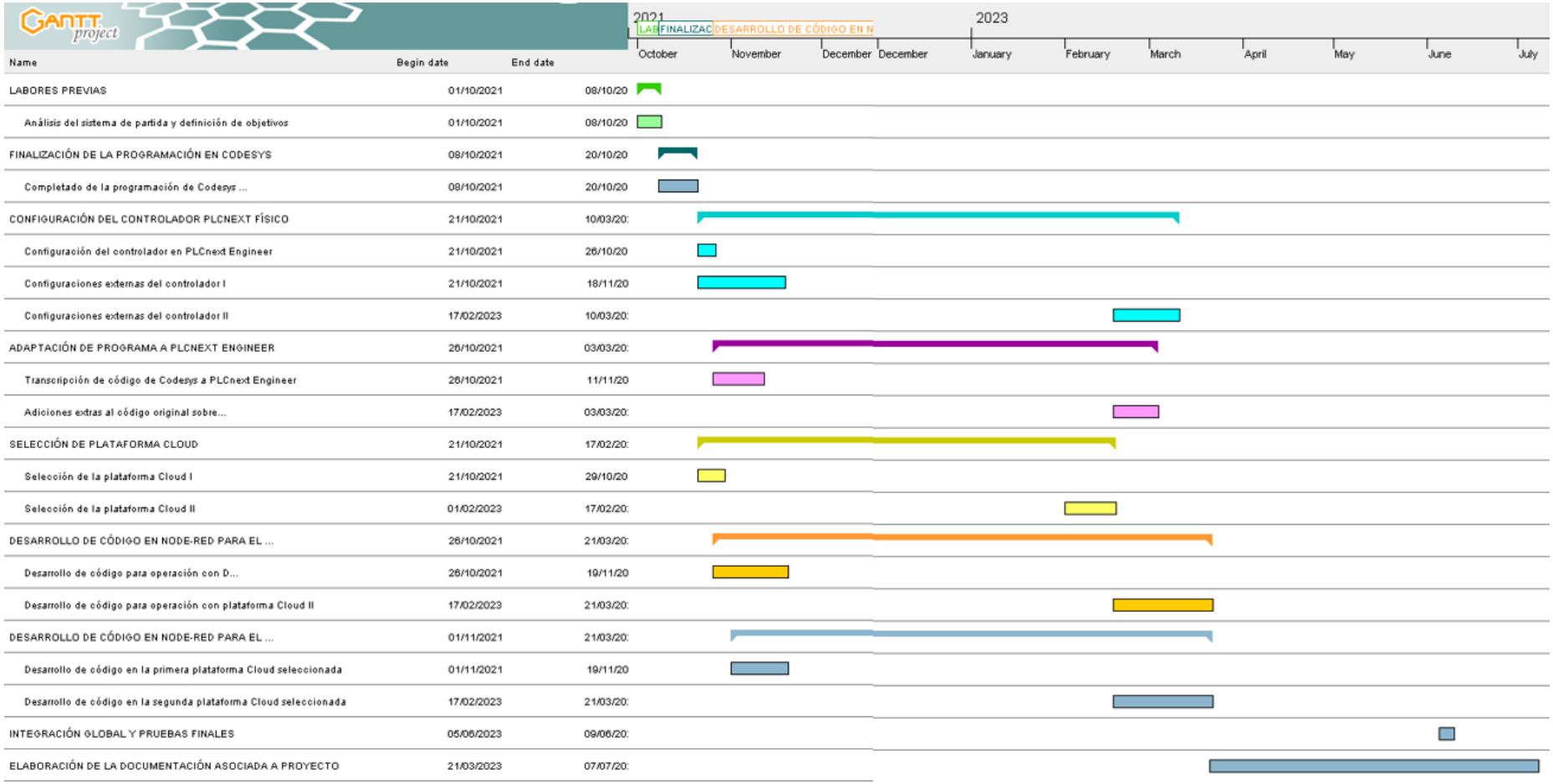

Figura 7.1 Diagrama de Gantt asociado al proyecto.

8. Presupuesto

8.1.- INTRODUCCIÓN

A continuación, se tratará el presupuesto total estimado del proyecto teniendo en cuenta los costes en materiales y mano de obra.

La licencia de Microsoft Office ha sido puesta a disposición del alumnado de la Universidad de Oviedo, así como las herramientas utilizadas: PLCnext Engineer 2022.6, Codesys v3.5 SP16 y Node-RED se encuentran disponibles de forma gratuita en la red sin necesidad de licencia, por lo que no han sido tenidas en cuenta.

Para el PC Portátil se supone una vida útil estándar de 4 años, mientras que para el EDU AXC F 2152, al ser un controlador industrial, se supone una vida útil mayor, por ejemplo, de 12 años.

8.2.- MEDICIONES DE LOS MATERIALES

Nº	Concepto	Fabricante	Unidades
1	PLCnext ENG SFC	Phoenix Contact	1

Tabla 8.1 Mediciones de software utilizado.

Nº	Concepto	Fabricante	Unidades
1	EDU AXC F 2152	Phoenix Contact	1
2	Cable Ethernet	Omron Automation	1
3	PC Portátil	ASUS	1

Tabla 8.2 Mediciones de hardware utilizado.

8.3.- MEDICIONES DE LA MANO DE OBRA

Concepto	Días Totales	Horas/Día	Horas Total
Labores previas	8	2	16
Finalización de la programación en Codesys	13	2	26

Configuración del controlador PLCnext físico	57	2	114
Adaptación de programa a PLCnext Engineer	32	2	62
Selección de plataforma Cloud	26	2	52
Desarrollo de código en Node-RED para el controlador	58	2	116
Desarrollo de código en Node-RED para el entorno Cloud	52	2	104
Integración global y pruebas finales	5	2	10
Elaboración de la documentación asociada al proyecto	109	2	218

Tabla 8.3 Mediciones de mano de obra.

8.4.- PRECIOS UNITARIOS DE MATERIALES

Nº	Concepto	Fabricante	Unidades	Precio Unidad (€)	Precio Total (€)
1	PLCnext ENG SFC	Phoenix Contact	1	200	200

Tabla 8.4 Precios unitarios de elementos software.

Nº	Concepto	Fabricante	Unidades	Precio Unidad (€)	Precio de uso (€)
1	EDU AXC F 2152	Phoenix Contact	1	1507,20	125,60
2	Cable Ethernet	Omron Automation	1	21,80	21,80
3	PC Portátil	ASUS	1	1500,80	375,20
Total Componentes de *hardware*					522,60 €

Tabla 8.5 Precios unitarios de elementos hardware.

8.5.- PRECIOS UNITARIOS DE MANO DE OBRA

Los precios unitarios de la mano de obra han sido calculados teniendo en cuenta:

Concepto	Horas Total	Coste / Hora (€)	Coste Total (€)
Labores previas	16	20	320

Finalización de la programación en Codesys	26	20	520
Configuración del controlador PLCnext físico	114	20	2280
Adaptación de programa a PLCnext Engineer	62	20	1240
Selección de plataforma Cloud	52	20	1040
Desarrollo de código en Node-RED para el controlador	116	20	2320
Desarrollo de código en Node-RED para el entorno Cloud	104	20	2080
Integración global y pruebas finales	10	20	200
Elaboración de la documentación asociada al proyecto	218	20	4360
Total			14360

Tabla 8.6 Precios unitarios de mano de obra.

8.6.- PRESUPUESTO FINAL

8.6.1- Presupuesto de ejecución material del proyecto

Concepto	**Presupuesto (€)**
Total elementos *software*	200,00
Total elementos *hardware*	522,60
TOTAL MATERIALES	722,60
Total mano de obra	14360
TOTAL DE EJECUCIÓN MATERIAL	15082,60 €

Tabla 8.7 Presupuesto de ejecución material del proyecto.

8.6.2- Presupuesto de ejecución por contrata

Se supone para el presupuesto de ejecución por contrata unos gastos generales del 13% y un beneficio industrial del 6%.

Concepto	Presupuesto (€)
Total de ejecución material	15082,60
Gatos generales (13%)	1960,74
Beneficio industrial (6%)	904,96
TOTAL PARCIAL	17948,30 €

Tabla 8.8 Presupuesto parcial del proyecto sin impuestos.

Se añade un 21% de IVA al coste parcial del proyecto en concepto de impuestos que debe abonar el cliente a la administración.

Concepto	Presupuesto (€)
Total parcial	17948,30
IVA (21%)	3769,14
TOTAL DE EJECUCIÓN POR CONTRATA	21717,44 €

Tabla 8.9 Presupuesto de ejecución por contrata del trabajo.

El presupuesto total del trabajo asciende a la cantidad de **veintiún mil setecientos diecisiete euros con cuarenta y cuatro céntimos (21.717,44€)** impuestos incluidos.

9. Conclusiones y futuras ampliaciones

9.1.- CONCLUSIONES

Tras el desarrollo del proyecto presentado, se han extraído una serie de conclusiones que se considera de interés comentar.

La primera y una de las más relevantes a destacar, es lo ventajosa que resulta la aplicación de la metodología ISA88 para el diseño y programación de la planta. La implantación de esta metodología hace que se cumplan dos premisas que resultan muy importantes en las solicitudes de la industria actual. Lo primero es que genera una enorme organización en la estructura de programación de la planta, ordenando individualmente cada uno de los elementos (módulos de equipamiento) que toman parte y jerarquizando todas las acciones que pueden llevarse a cabo sobre el sistema. La segunda y no por ello menos importante, es que oferta una flexibilidad completa a cada uno de los elementos que toman parte. Ya no es que la planta se pueda contemplar exclusivamente como un todo, si no que cada uno de los elementos que toman parte se pueden analizar y tratar de manera independiente e incluso ser capaces de extraerlos de esta instalación e integrarlos en otras que los requieran.

Para que se pueda realizar esa integración global, el conjunto de las instalaciones también tienen que estar conceptualizadas y desarrolladas bajo la normativa ISA88. En resumen, ofrece un orden y una metodología a seguir para uniformizar los desarrollos y facilitar al operario tanto la programación, como la operación física con la planta externa a la que se ha realizado el desarrollo.

La adaptación de Codesys a PLCnext Engineer, como se ha podido observar en previas páginas de este documento, ha sido exitosa, aunque si bien es cierto, la adaptación del SFC no ha sido tan sencilla, como debería ser. El ideal para mejorar este punto igual vendría fundamentado por cambiar el código de SFC en Codesys a ST en PLCnext Engineer.

GEMMA resultó también ser una incorporación exitosa en la generación de otro elemento de organización del sistema a nivel global y tener siempre en mente en qué estado de operación se encuentra.

Para las pantallas de operador y Dashboards se ha tenido en cuenta, en la medida de lo posible, la normativa ISA 101, para favorecer al entendimiento y seguimiento de dichas pantallas. Si bien es cierto que no ha habido mayor problema, el Dashboard generado en IBM Cloud no ha podido ser alterado y se ha quedado un fondo oscuro, siendo el único punto donde esta norma no se ha podido realizar del todo su apliación.

Se ha de destacar también el uso del controlador PLCnext AXC F 2152. Permitió proporcionar al desarrollo de proyecto una base sólida que permite integrar bajo un mismo dispositivo, el control completo de la planta y ser capaz de enlazarlo directamente aguas arriba con cualquier plataforma Cloud externa a este, que el usuario requiera. Esto supone una gran ventaja frente a otros controladores industriales. Como se ha verificado para el entorno Cloud de Azure, los controladores (PLCs) más convencionales requieren de un intermediario (una Raspberry Pi, en el caso estudiado) para poder comunicar la plataforma Cloud con el controlador.

Enlazando también ISA88 con la interacción de la planta con la plataforma Cloud, el establecimiento de esta metodología en la implementación permite organizar de los datos de una manera que facilita la operación, el control de los mismos y su organización en el entorno de programación Node-RED facilitando en gran medida el seguimiento del programa generado y mantener en todo momento la visualización de qué se está enviando a la plataforma Cloud.

A su vez, Node-RED en conjunto con OPC UA, han sido elementos clave para el correcto flujo de datos entre planta y plataforma Cloud, proporcionando elementos de seguridad y un fácil seguimiento del trayecto seguido por los datos.

Si bien es cierto que hasta el momento se han comentado puntos muy positivos, hay que analizar un par de aspectos observados a tener en cuenta en este tipo de aplicaciones. El

primero, es que, si bien es cierto que se ha desarrollado todo sobre un mismo controlador, el nivel de carga sobre este es muy elevado, haciendo que su hardware pueda decirse que está casi operando al límite de su capacidad. Este punto, por el contrario, puede tener una sencilla solución en la manera de programar el código del control, haciendo que opere repartido entre los dos núcleos del controlador en vez que en uno solo.

El otro punto por comentar, pero no de tan fácil solución, es que debido al desarrollo realizado en las plataformas Cloud, buscando siempre una vertiente gratuita para realizar la prueba a nivel de investigación en el envío y tratamiento de datos en la nube, nunca se tienen garantías de que ese servicio permanezca estable, ni de si va a cambiar a ser de pago o de si se va a realizar un cese de servicio. Esto se ha visto reflejado al realizar la primera selección del entorno Cloud (IBM Cloud) que por cese de servicio obligó a realizar una migración de la implementación realizada a otra plataforma. La manera más sencilla de solventar eso, queda claramente en afianzar un servicio de pago acorde a las necesidades del usuario.

Para sintetizar, PLCnext oferta un controlador ideal para desarrollar un programa basado en ISA88, que mediante el apoyo de herramientas como Node-RED y OPC UA se puede implementar una aplicación completa que cumpla requisitos solicitados por la Industria 4.0, volviéndolo una combinación de elementos muy acertada para cualquier aplicación en la industria moderna.

9.2.- FUTURAS AMPLIACIONES

Para realizar el cierre del documento asociado al desarrollo del proyecto, se apuntan algunos puntos que sería interesante desarrollar en caso de seguir trabajando con el proyecto:

- Ampliar y optimizar funcionalidades del sistema añadiendo más estados al GDMMA, añadir más elementos de alarmas, avisos, defectos, emergencias, análisis y gestión de consumos…

- Optimización del código desarrollado para el entorno PLCnext Engineer, una de esas posibles optimizaciones sería el paso de SFC a ST, puesto que el manejo de

los SFCs en este entorno no es tan sencillo como otros y el paso a un lenguaje como ST, permitiría una mayor estructuración y organización de código sin tener que crear elementos auxiliares.

- Optimizar el nivel de operación del hardware del sistema, repartiendo la carga de programa entre los núcleos del controlador.

- Guardar datos esenciales de operación de la planta en bases de datos generadas a nivel local y remoto en el entorno Cloud. Este punto se debería aplicar tras realizar la optimización del código de control entre los dos núcleos del controlador.

- Mayor estructuración de código en Node-RED del controlador generando un reparto entre diferentes pestañas de programa enlazadas entre sí para generar una mayor organización gráfica para el usuario.

- Analizar la posibilidad de incorporar sistema de seguridad (identificación de usuario) al Dashboard del Node-RED localizado en el controlador PLCnext con el fin de poder proporcionar un espacio seguro donde localizar una zona de operación de estado F4 del GDMMA (verificación en desorden) en modo remoto.

- Implementación de una planta física donde realizar la prueba de este sistema o en su defecto, una aplicación virtual sobre herramientas de simulación de planta como Factory I/O o EasyPLC Machines Simulator.

- Adquisición de un servicio de la plataforma Cloud estable que garantice una implementación más extendida de las funcionalidades asociadas a la plataforma sin el riesgo de tener que realizar una migración constante de los servicios ya desarrollados.

10.Referencias

[1] MATEOS MARTÍN, F y QUINTANA SIÑERIZ, G (2022) "*OPC-UA, Node-RED and Cloud functionalities for PLCnext Technology*"

[2] MATEOS MARTÍN, F y QUINTANA SIÑERIZ, G (2022) "*OPC-UA, Node-RED y funcionalidades Cloud con tecnología PLCnext*"

[3] "IEC 61131-3 LENGUAJES DE PROGRAMACIÓN" http://isa.uniovi.es/~vsuarez/Download/IEC%2061131-3%20(Lenguajes).pdf (Consulta: 3 de julio de 2023)

[4] INTERNATIONAL ELECTROTECHNICAL COMMISION (2013) *Programmable controllers -Part 3: Programing languajes.* BEC-CEB

[5] KARL-HEIZ, J y TIEGELKAMP, M. (2014) *IEC 61131-3: Programming Industrial Automation Systems 2nd edition*. Springer.

[6] GARCÍA, M y IRISARRI, E y PÉREZ, F y MARCOS, M (2017) "From ISA 88/95 Meta-Models to an OPC UA-Based Development Tool for CPPS under IEC 61499"

[7] CHICAZIA, F y GARCÍA, C y CASTELLANOS, E y SÁNCHEZ, C y ROSERO, C y GARCÍA, M "Arquitectura Flexible Basada en ISA-88 para el diseño del Diagrama de Control de Ejecución en Aplicaciones Distribuidas mediante IEC-61499"

[8] "ISA88: Batch Control" https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa88 (Consulta: 4 de julio de 2024)

[9] "Modos de marcha y parada. La guía GEMMA" http://isa.uniovi.es/~vsuarez/Download/GemmaTelemecanique.PDF (Consulta: 4 de julio de 2023)

[10] "08. – Guía GEMMA" https://biblus.us.es/bibing/proyectos/abreproy/4553/fichero/08+-+Gu%C3%ADa+GEMMA.pdf (Consulta: 4 de julio de 2023)

[11] AMERICAN NATIONAL STANDARD (2015) *ANSI/ISA-101.01-2015, Human Machine Interfaces for Process Automation Systems*

[12] BOHÓRQUEZ, E y PRADO, E y RAMIREZ, M (2019) *Implementación de la norma ISA 101, sobre las HMI, pertenecientes a los módulos de instrumentación de la Universidad ECCI*

[13] "8 reglas de diseño para crear un HMI de máquina atractivo" https://www.infoplc.net/blogs-automatizacion/item/101661-8-reglas-de-diseno-hmi-atractivo (Consulta: 4 de julio de 2023)

[14] "Why CODESYS? The right choice for users and device manufacturers across all industries" https://www.codesys.com/the-system/why-codesys.html (Consulta: 4 de julio de 2023)

[15] "¿Qué es la Industria 4.0?" https://www.lisdatasolutions.com/es/blog/que-es-la-industria-4-0/ (Consulta: 4 de julio de 2023)

[16] "¿Qué es la industria 5.0 y cuáles son sus 3 principales características?" https://www.advancedfactories.com/industria-5-0-caracteristicas/ (Consulta: 4 de julio de 2023)

[17] AMENDOLA, L (2022) *"¿Por qué la industria 5.0? ¿Qué opinan los profesionales?* https://es.linkedin.com/pulse/por-qu%C3%A9-la-industria-50-opinan-los-profesionales-amendola-ph-d (Consulta: 4 de julio de 2023)

[18] AUTYCOM (2022) "*Los 9 pilares de la industria 4.0"* https://es.linkedin.com/pulse/los-9-pilares-de-la-industria-40-autycom (Consulta: 5 de julio de 2023)

[19] "Ecosistema PLCnext Technology, el beneficio de nuevas opciones y formas de colaboración" https://www.lavozdegalicia.es/noticia/contenidos-patrocinados/2020/10/20/ecosistema-plcnext-technology-br-beneficio-nuevas-opciones-formas-colaboracion/00031603196566392313609.htm (Consulta: 5 de julio de 2023).

[20] "Proficloud.io: Your ticket to industrial IoT" https://proficloud.io/ (Consulta:5 de julio de 2023)

[21] "EDU AXC F 2152 – Material didáctico" https://www.phoenixcontact.com/es-cl/productos/material-didactico-edu-axc-f-2152-1046674 (Consulta: 5 de julio de 2023)

[22] "PLCNEXT ENGINEER – Software de programación" https://www.phoenixcontact.com/es-es/productos/software-de-programacion-plcnext-engineer-1046008 (Consulta: 5 de julio de 2023)

[23] "Complemento PLCNEXT" https://www.phoenixcontact.com/es-es/configurar/productos/programming-plcnext-engineer-1046008/add-ons (Consulta: 5 de julio de 2023)

[24] "UaExpert – A Full – Featured OPC UA Client" https://www.unified-automation.com/products/development-tools/uaexpert.html (Consulta: 5 de julio de 2023)

[25] "Node-RED Low-code programming for event-driven applications" https://nodered.org/ (Consulta: 5 de julio de 2023)

[26] "WinSCP" https://winscp.net/eng/download.php (Consulta: 5 de julio de 2023)

[27] "Putty" https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html (Consulta: 5 de julio 2023)

[28] "Node-RED and getting started with Docker" https://www.plcnext-community.net/makersblog/node-red-and-getting-started-with-docker/ (Consulta: 5 de julio de 2023)

[29] "IBM CLOUD. Permite resolver problemas reales y aumentar el valor de tu negocio con aplicaciones, infraestructura y servicios" https://www.cloudbuilders.es/ibm-cloud/ (Consulta: 6 de julio de 2023)

[30] "INTERNET OF THINGS: IBM Watson IoT" https://glartek.com/integrations/ibm-watson-iot/ (Consulta: 6 de julio de 2023)

[31] "Introducción a Azure" https://myclouddoor.com/introduccion-a-azure/ (Consulta: 6 de julio de 2023)

[32] "IoT Hub: IoT service hosted in the cloud" https://vecta.io/symbols/27/microsoft-azure-mono/53/iot-hub (Consulta: 6 de julio de 2023)

[33] "AXC F 2152 – Sistema de control" https://www.phoenixcontact.com/es-es/productos/mando-axc-f-2152-2404267 (Consulta: 6 de julio de 2023)

[34] "Consulta de UTC" https://time.is/es/UTC (Consulta: 6 de julio de 2023)

[35] "node-red-azure-webapp" https://github.com/jmservera/node-red-azure-webapp (Consulta: 6 de julio de 2023)