

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

“Desarrollo de una herramienta SCADA en Codesys para fotobiorreactores industriales”

Curso: 2021/2022

Modalidad TFG: Trabajo Técnico

Alumno/a:

Marina Guil Torres

Director/es:

José Luis Guzmán Sánchez

José González Hernández



Universidad de Almería
Escuela Superior de Ingeniería

Trabajo Fin de Grado
Grado en Ingeniería Electrónica Industrial
Curso 2021/2022



*Desarrollo de una herramienta SCADA
con Codesys para fotobiorreactores
industriales*

Autor Marina Guil Torres

Tutores José Luis Guzmán Sánchez
José González Hernández

Agradecimientos

A mi director y codirector del Trabajo Fin de Grado, José Luis Guzmán Sánchez y José González Hernández, por su ayuda y dedicación a lo largo del proceso.

Al centro de investigación IFAPA, por permitirme participar de este proyecto y por el buen trato recibido.

Este Trabajo Fin de Grado ha sido financiado con el Proyecto del Plan Nacional "Control híbrido y optimización de una biorrefinería sostenible para la producción industrial de microalgas (HYCO2BIO)", PID2020-112709RB-C21R del Ministerio de Ciencia e Innovación.

A mis seres queridos, familia y amigos, por su apoyo incondicional.

Glosario

OPC	OLE for Process Control (OLE para Control de Procesos).
OLE	Object Linking and Embedding.
SCADA	Supervisory, Control and Data Adquisition (Supervisión, Control y Adquisición de Datos).
IFAPA	Investigación y Formación Agraria y Pesquera.
PLC	Programmable Logic Controller (Controlador Lógico Programable).
IL	Instruction List (Lista de Instrucciones).
CFC	Continuous Function Chart (Diagrama Continuo).
SFC	Sequential Function Chart (Diagrama de Funciones Secuencial).
ST	Structured Text (Texto Estructurado).
NTC	Negative Temperature Coefficient (Coeficiente de Temperatura Negativo).
OD	Oxígeno Disuelto.
LabVIEW	Laboratory Virtual Instruments Engineering Workbench.
LD	Ladder Diagram (Diagrama de contactos).
HMI	Human Machine Interface (interfaz hombre-máquina).
FBD	Function Block Diagram (Bloques funcionales).
POU	Program Organization Unit.
DCOM	Distributed Component Object Model (Modelo de Objetos de Componentes Distribuidos).
COM	Component Object Model (Modelo de Objetos de Componentes).
RTU	Remote Terminal Unit.
ISO	Internacional Organization for Standardization (Organización Internacional de Estandarización).
PC	Personal Computer (Computador Personal).
OPTO 22	Compañía de fabricación de productos hardware y software para automatización industrial.
PAC	Programmable Automation Controller (Controlador Programable de Automatización).

IEC	International Electrotechnical Commission (Comisión Electrotécnica Internacional).
TIC	Tecnologías de la Información y la Comunicación.
HTML5	Última versión del estándar HTML.
HTML	HyperText Markup Language (Lenguaje de marcado de hipertexto).
Codesys	Controller Development System (Sistema de desarrollo del controlador).
ODM	OPC Data Manager.
RTD	Resistance Temperature Detector (Detector de Temperatura Resistivo).
VI	Virtual Instrument (Instrumento Virtual)
TOD	Time of day

Resumen

La motivación de este Trabajo Fin de Grado es desarrollar una herramienta SCADA, programada en Codesys, que sirva para la supervisión, control y adquisición de datos de fotobiorreactores industriales pertenecientes a instalaciones asociadas a la Universidad de Almería, las cuales se destinan al estudio de la producción de microalgas y su inserción en el mercado para usos alimenticios o cosméticos, así como para el tratamiento de aguas residuales o la reducción de emisiones de CO₂ provenientes de diversos procesos industriales, según proceda.

Esta herramienta pretende ser un sucesor más competente y asequible que el actualmente instalado mediante la plataforma LabVIEW. Por ello, este proyecto hace a la vez una tarea de compresión y estudio no solo del programa Codesys, sino también de este último.

Ambas partes, sistema real y herramienta SCADA, se comunicarán entre sí a partir del protocolo OPC UA que proporcionará una vía para el intercambio de información de uno a otro, permitiendo supervisar así el comportamiento de los reactores en tiempo real al mismo tiempo que se ejecutan órdenes desde el cuadro de control del operario o desde otro software o entorno externo.

Palabras clave: SCADA, OPC, microalgas, Codesys, LabVIEW, fotobiorreactores.

Abstract

The motivation of this Final Degree Project is to develop a SCADA tool, programmed in Codesys, for the supervision, control and data acquisition of industrial photobioreactors belonging to facilities associated with the University of Almeria, which are intended to study the production of microalgae and their insertion in the market for food or cosmetic uses, as well as for the treatment of wastewater or the reduction of CO₂ emissions from various industrial processes, as appropriate.

This tool is intended to be a more competent and affordable successor to the one currently installed using the LabVIEW platform. For this reason, this project makes both a task of compression and study not only of the Codesys programme, but also of the latter.

Both parts, the real system and the SCADA tool, will communicate with each other using the OPC UA protocol, which will provide a way for the exchange of information from one to the other, thus allowing the behaviour of the reactors to be supervised in real time at the same time as orders are executed from the operator's control panel or from other software or external environment.

Keywords: SCADA, OPC, microalgae, Codesys, LabVIEW, photobioreactors.

Índice General

Agradecimientos.....	3
Glosario	5
Resumen.....	7
Abstract	9
Índice General	11
Índice de Figuras.....	13
Índice de Tablas	17
Capítulo 1: Introducción.....	19
1.1. Objetivos, contexto y motivación del proyecto.....	19
1.2. Planificación temporal	20
1.3. Competencias utilizadas en el proyecto	20
1.4. Estructura del proyecto.....	22
1.5. Resumen de resultados	23
Capítulo 2: Materiales y Métodos	25
2.1. Descripción de la instalación	25
2.1.1. Reactores Raceway	25
2.1.2. Estación meteorológica	28
2.1.3. Autómatas programables (PLCs)	29
2.2. Entornos de desarrollo	29
2.2.1. LabVIEW	30
2.2.1.1. Interfaz de LabVIEW.....	31
2.2.2. Codesys	34
2.2.2.1. Norma IEC 61131-3	34
2.3. Servidor OPC	39
2.3.1. Definición y especificaciones OPC	39
2.3.2. Características y funcionalidad del servidor.....	41
2.3.3. Interfaces de comunicación	41
2.4. Matlab.....	42
2.5. Matrikon OPC Data Manager	42
Capítulo 3: Diseño del SCADA	45
3.1. Análisis de la herramienta actual.....	45
3.2. Creación del proyecto	47
3.3. Conexión con el dispositivo PLC Virtual	49
3.4. Declaración de variables globales.....	51
3.5. POU <i>Tiempo</i>	54
3.6. POU <i>PLC_PRG</i>	59

3.7. POU <i>Reseteo</i>	62
3.8. Visualizaciones	65
3.8.1. Entorno de visualización	65
3.8.2. Elementos de visualización	65
3.8.3. ImagenPool	66
3.8.4. Propiedades del elemento	67
3.8.5. Visualizaciones diseñadas	68
3.8.6. Identificadores	76
3.8.7. Representaciones gráficas	77
3.8.8. Gestor de visualización	79
3.9. Configuración de tareas	83
3.10. Conexión con el servidor	85
3.11. Configuración de símbolos	87
3.12. Conexión con Matlab	92
3.13 SCADA resultante	98
3.14 Puesta en marcha	107
Capítulo 4: Conclusiones	113
Bibliografía/Referencias	115
Anexos	117
6.1. Instalación de los programas	117
6.2. Activación de la licencia del servidor OPC	119
6.3. Descarga de la biblioteca <i>Oscat basic</i> desde Codesys Store	123
6.4. Activación de los comentarios por línea en lenguaje LD	126

Índice de Figuras

Figura 1.1. Panel del usuario resultante.	23
Figura 1.2. Esquema de las comunicaciones entre aplicaciones.	24
Figura 2.1. Vista de los fotobiorreactores raceways del centro de experimentación IFAPA en La Cañada de San Urbano, Almería.	25
Figura 2.2. Esquema de los Raceways.....	26
Figura 2.3. Localización de los sensores en cada reactor.....	27
Figura 2.4. Ejemplo de un VI en LabVIEW	31
Figura 2.5. Paleta de Controles	32
Figura 2.6. Ejemplo de terminales de control y de indicador	33
Figura 2.7. Paleta de Funciones	33
Figura 2.8. Estructura de la red de comunicación de OPC.	40
Figura 3.1. Interfaz del SCADA instalado.....	46
Figura 3.2. Pantalla principal del entorno de desarrollo de Codesys	47
Figura 3.3. Nuevo Proyecto	48
Figura 3.4. Configuración del proyecto.....	48
Figura 3.5. Pantalla principal del proyecto creado	49
Figura 3.6. Pestaña Device.....	49
Figura 3.7. Dispositivos PLC y Gateway.....	50
Figura 3.8. Desplegable de Examinar la Red.	50
Figura 3.9. Derechos de acceso al PLC.....	51
Figura 3.10. Creación de un objeto en la aplicación.	51
Figura 3.11. Lista de variables globales declaradas.....	53
Figura 3.12. Ventana POU <i>Tiempo</i>	55
Figura 3.13. POU <i>Tiempo</i>	59
Figura 3.14. POU <i>PLC_PRG</i>	62
Figura 3.15. POU <i>Reseteo</i>	64
Figura 3.16. Vista de una Ventana de Visualización	65
Figura 3.17. Recopilación de elementos empleados en Visualización.	66
Figura 3.18. Objeto <i>ImagePool</i>	66
Figura 3.19. Pestaña de propiedades del elemento Botón <i>Visualización RW1</i>	67
Figura 3.20. Visualización base.....	68
Figura 3.21. Configuración del 1ºMarco	69
Figura 3.22. Configuración del botón <i>Visualización RW1</i>	69
Figura 3.23. Visualizaciones asociadas al 1º Marco.....	70

Figura 3.24. Visualizaciones asociadas al 2ºMarco	71
Figura 3.25. Visualizaciones asociadas al 3º Marco	71
Figura 3.26. Visualizaciones asociadas a los marcos secundarios	74
Figura 3.27. Explicación de la conmutación del Botón Manual/Automático, Control CO2 RW2.....	75
Figura 3.28. Control de Nivel del Raceway 1, Modo Manual y Con Optimizador	75
Figura 3.29. Clasificación de identificadores	76
Figura 3.30. Propiedades del recuadro Hercios de RW1	77
Figura 3.31. Tipos de entrada.....	77
Figura 3.32. Trend asociado a la gráfica del Raceway 1, <i>Trend_RW1</i>	78
Figura 3.33. Configuración de la variable <i>Global.Setpoint_nivel_RW1</i> dentro del Trend_RW1.....	78
Figura 3.34. Trace <i>Traza_RW1</i> , sin ejecución.....	79
Figura 3.35. Vista del Gestor de visualización	79
Figura 3.36. Gestor de visualizaciones, sección <i>Visualizaciones</i>	80
Figura 3.37. Arquitectura de comunicación con TargetVisu.	80
Figura 3.38. Arquitectura de comunicación con WebVisu	81
Figura 3.39. Configuración de los visualizadores	82
Figura 3.40. Configuración de tareas.....	83
Figura 3.41 Configurador de cada tarea del proyecto	84
Figura 3.42. Ventana de OPCConfig	85
Figura 3.43. Configuración del PLC1	86
Figura 3.44. Configuración de la dirección del PLC	86
Figura 3.45. Configuración de la conexión	87
Figura 3.46. Tipos de derechos de acceso	87
Figura 3.47. Lista de símbolos	91
Figura 3.48. Proceso de creación de un objeto <i>Archivo Externo</i>	92
Figura 3.49. Acceso a la aplicación OPC de Matlab.	93
Figura 3.50. Pasos de la configuración de la conexión con OPC.	94
Figura 3.51. Ejemplo Item <i>ANO_ACTUAL</i>	95
Figura 3.52. Códigos de Matlab	97
Figura 3.53. Código de las funciones de Matlab	98
Figura 3.54. Opciones de ejecución de proyecto.....	99
Figura 3.55. Explicación 1	100
Figura 3.56. Explicación 2.....	100
Figura 3.57. Explicación 3	101
Figura 3.58. Explicación 4.....	103

Figura 3.59. Explicación 5	104
Figura 3.60. Explicación 6	105
Figura 3.61. Explicación 7	106
Figura 3.62. Explicación 8	106
Figura 3.63. Interfaz de Matrikon OPC Data Manager.....	107
Figura 3.64. Ejemplo de comunicación de los PLCs a SCADA.	108
Figura 3.65. Ejemplo de comunicación del SCADA a los PLCs.	109
Figura 3.66. Ejemplo de comunicación del SCADA a Matlab, escritura de una entrada en el diario de actividades.	110
Figura 3.67. Ejemplo de comunicación del SCADA a Matlab y de nuevo al SCADA, cálculo de los parámetros del optimizador 1.....	111
Figura 6.1. Tienda online de Codesys.....	117
Figura 6.2. Pestaña para la descarga de Codesys y algunas de sus posibles versiones disponibles	118
Figura 6.3. Pestaña para la descarga del servidor Codesys OPC DA Server SL Demo	118
Figura 6.4. Cuenta de usuario, pestaña Mis licencias.....	119
Figura 6.5. Selección del Target y Container del servidor	120
Figura 6.6. Activación de la licencia del servidor.....	120
Figura 6.7. Administrador de licencias	121
Figura 6.8. Mensaje de error, expiración de la licencia de uso.....	121
Figura 6.9. Mensaje de error de Codesys.	121
Figura 6.10. Carpeta GatewayPLC	122
Figura 6.11. Centro de Control de CodeMeter.....	122
Figura 6.12. Resultado de la nueva activación de la licencia del servidor OPC.....	123
Figura 6.13. Tienda Online Codesys.....	124
Figura 6.14. Administrador de paquetes.....	125
Figura 6.15. Administrador de bibliotecas	125
Figura 6.16. Ventana <i>Opciones</i>	126

Índice de Tablas

Tabla 1.1. Planificación temporal	20
Tabla 2.1. Recopilación de componentes de la estación meteorológica.....	29
Tabla 2.2. Clasificación de los tipos de datos	36

Capítulo 1: Introducción

En este primer capítulo se pondrá de relieve en qué contexto nace la necesidad de desarrollar el siguiente proyecto fin de grado, se expondrán los principales objetivos a lograr con él y las fases para su desarrollo.

1.1. Objetivos, contexto y motivación del proyecto

A día de hoy, la sociedad se enfrenta a diversos retos entre los que se incluyen el de alcanzar una mayor sostenibilidad, tanto de recursos como ambiental, y el de disminuir la contaminación global. Por ese motivo, se está trabajando en el desarrollo de nuevas tecnologías que permitan contribuir a la reducción del impacto medioambiental, mitigación del CO₂, transformación y reciclaje de residuos, depuración de aguas residuales y desarrollo de fuentes de energía alternativas.

En este contexto, una línea de investigación que está cobrando cada vez mayor relevancia es la de la producción de microalgas. Las microalgas son organismos fotosintéticos con la capacidad de crecer y reproducirse en entornos sin agua limpia o suelo fértil. Esto las convierten en una excelente fuente de biocombustible como respuesta al problema energético. Además, su propio proceso de producción también resulta útil, pues poseen la capacidad de absorber contaminantes de aguas residuales aprovechándolos para su crecimiento, así como emisiones de CO₂ que provengan de procesos industriales, por lo que se presenta también como una solución al tratamiento de aguas residuales o a la mitigación de gases invernadero. [7]

Por otro lado, las microalgas al ser fuentes de biomoléculas y metabolitos pueden usarse también en la elaboración de productos de alto valor en el campo de la cosmética, nutrición humana o alimentación animal.

La producción de microalgas es un proceso que debe ser adecuadamente planeado y realizado. Sus principales etapas son las siguientes: 1) preparación del medio de cultivo; 2) producción de biomasa en fotobiorreactores; 3) cosechado de la biomasa; 4) tratamiento de aguas para recirculación o vertido; y 5) estabilización de la biomasa o su transformación en productos finales.

Cada una de ellas posee un papel clave dentro del proceso productivo. Sin embargo, la etapa referente a la producción de biomasa en los fotobiorreactores es la más compleja y relevante debido a la gran cantidad de factores que afectan a la operación del sistema, y donde el control de procesos juega un papel fundamental para conseguir maximizar dicha producción.

El objetivo, por lo tanto, es automatizar este tipo de procesos con el fin de hacerlos competitivos en el sector industrial. Dentro de este proceso de automatización adquiere una importancia relevante el desarrollo de herramientas de supervisión, control y adquisición de datos (herramientas SCADA). En las instalaciones asociadas a la Universidad de Almería, típicamente este tipo de herramientas se han implementado con un entorno de desarrollo comercial que requiere un alto coste de mantenimiento, LabVIEW [12]. De este modo, lo que

se pretende con este trabajo técnico es desarrollar o analizar una nueva alternativa de código abierto para desarrollar una herramienta SCADA que resulte más asequible y permita aportar así una solución más competitiva y transferible a la industria.

En base a lo comentado anteriormente, los objetivos principales de este trabajo fin de grado se podrían resumir en los siguientes puntos:

1. Estudio y análisis del potencial de la herramienta Codesys para el desarrollo de herramientas SCADA.
2. Análisis de todas aquellas variables que resulten de interés de los fotobiorreactores industriales en lo que respecta a la automatización del proceso.
3. Desarrollo de conexiones de las variables a través de un servidor OPC.
4. Desarrollo completo de la herramienta SCADA conectada al servidor OPC previamente mencionado.
5. Evaluación del SCADA desarrollado en un fotobiorreactor industrial.
6. Elaboración de la memoria del Trabajo Fin de Grado.

1.2. Planificación temporal

La planificación temporal del proyecto se desglosa en la Tabla 1.1.:

Intervalo de tiempo	Actividad	Duración en horas
25 sept a 10 oct	Estudio de los entornos de desarrollo LabVIEW y Codesys. Estudio del servidor OPC.	40h (~3h/día)
11 oct a 30 nov	Comprensión del funcionamiento del SCADA actual y desarrollo del nuevo SCADA en Codesys.	250h (~5h/día)
1 dic a 17 ene	Incorporación del SCADA en la planta y redacción de la memoria del TFG.	90h (~h/día)
		380h

Tabla 1.1. Planificación temporal.

1.3. Competencias utilizadas en el proyecto

El Trabajo Fin de Grado supone la culminación de estudios de un estudiante y, por lo tanto, al igual que el resto de asignaturas cursadas, dotan a este de unas ciertas competencias una vez se supera.

Para este proyecto, en primer lugar, se adquieren competencias de carácter básico, comunes a todos los títulos de grado:

- Poseer y comprender conocimientos (CB1): demostración de que se poseen y comprenden conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.

- Aplicación de conocimientos (CB2): demostración de que se es capaz de aplicar conocimientos al trabajo o vocación de una forma profesional y se poseen las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro del área de estudio.
- Capacidad de emitir juicios (CB3): capacidad de reunir e interpretar datos relevantes (normalmente dentro del área de estudio) para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.
- Capacidad de comunicar y aptitud social (CB4): capacidad de transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
- Habilidad para el aprendizaje (CB5): desarrollo de habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.

Por otro lado, se adquieren unas competencias transversales, propias de la universidad:

- Conocimientos básicos de la profesión (UAL1): Conocimiento, habilidades y actitudes que posibilitan la comprensión de nuevas teorías, interpretaciones, métodos y técnicas dentro de los diferentes campos disciplinares, conducentes a satisfacer de manera óptima las exigencias profesionales.
- Habilidad en el uso de las TIC (UAL2): Capacidad de utilizar Tecnologías de Información y Comunicación (TICs) como herramienta para la expresión y la comunicación, para el acceso a fuentes de información, como medio de archivo de datos y documentos, para tareas de presentación, para el aprendizaje, la investigación y el trabajo cooperativo.
- Capacidad para resolver problemas (UAL3): Capacidad para identificar, analizar, y definir los elementos significativos que constituyen un problema para resolverlo con rigor.
- Comunicación oral y escrita en la propia lengua (UAL4): similar a CB4.
- Capacidad de crítica y autocrítica (UAL5): Es el comportamiento mental que cuestiona las cosas y se interesa por los fundamentos en los que se asientan las ideas, acciones y juicios, tanto propios como ajenos.
- Conocimiento de una segunda lengua (UAL7): Capacidad de entender y hacerse entender de manera verbal y escrita usando una lengua diferente a la propia.
- Capacidad para aprender a trabajar de forma autónoma (UAL9): Capacidad para diseñar, gestionar y ejecutar una tarea de forma personal.

Y, por último, unas competencias de carácter específico, propias del Grado en Ingeniería Electrónica Industrial:

- E-CT3 - Conocimiento en materias básicas y tecnológicas, que capacitan para el aprendizaje de nuevos métodos y teorías, y dotan de versatilidad para adaptarse a nuevas situaciones.
- E-CT4 - Capacidad de resolver problemas con iniciativa, toma de decisiones, creatividad, razonamiento crítico y de comunicar y transmitir conocimientos, habilidades y destrezas en el campo de la Ingeniería Industria.
- E-CB3 - Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
- E-CTEE10 - Conocimiento aplicado de informática industrial y comunicaciones.
- E-TFG - Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería Industrial de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

1.4. Estructura del proyecto

El desarrollo del proyecto se ha realizado siguiendo las siguientes fases o etapas:

1. Familiarización con la planta real.
2. Análisis de los entornos de desarrollo Codesys y LabVIEW. Comprensión del servidor OPC actualmente disponible en los reactores y posibles mejoras a realizar.
3. Estudio de la herramienta SCADA existente desarrollada con LabVIEW.
4. Desarrollo de la nueva herramienta SCADA y su conexión con la planta real.
5. Evaluación experimental en los fotobiorreactores reales.
6. Elaboración de la documentación del proyecto.

Ahora, dentro de este último punto, la estructura de la memoria se divide en 6 capítulos principales:

El primer capítulo, Introducción, recoge toda la información referente a cómo se ha desarrollado el proyecto, bajo que circunstancias y los resultados obtenidos de este, de forma resumida y concreta.

En el segundo capítulo, Materiales y Métodos, se analizan la instalación a la que hace referencia el proyecto, los entornos de programación involucrados y el servidor OPC.

El tercer capítulo, Resultados, congrega todo el desarrollo del proyecto en sí, es decir, la construcción de la herramienta SCADA para los fotobiorreactores industriales y su puesta en marcha en la planta.

Los dos capítulos siguientes, Conclusiones y Bibliografía/Referencias, se dedican al análisis de los resultados obtenidos y la referenciación de la información empleada en este trabajo, así como algunos libros recomendados que versan sobre esta materia.

Finalmente, hay un anexo donde se explican los procesos de instalación de los programas empleados y la instalación y activación de la licencia de uso del servidor OPC de Codesys. Adicionalmente, se muestra la incorporación de una biblioteca de Codesys, necesaria para desarrollar el código del SCADA en esta interfaz, y una explicación de cómo añadir comentarios en lenguaje Ladder en este entorno de desarrollo.

1.5. Resumen de resultados

Partiendo de una herramienta SCADA ya instalada en la planta de producción de microalgas del centro de investigación IFAPA, se ha desarrollado una nueva aplicación, pero esta vez haciendo uso del entorno Codesys. El objetivo de ello era comprobar que empleando este software gratuito era posible alcanzar los mismos resultados que con LabVIEW, la plataforma utilizada para el diseño de la herramienta previa y que era de uso comercial bajo licencia. Además, otra motivación para la elaboración de este proyecto era conocer más sobre la plataforma Codesys puesto que cada vez más industrias de todo el mundo optan por ella para cumplir con las labores de control y adquisición de datos de sus sistemas de producción.

A lo largo de este estudio, se ha comprobado que Codesys es una plataforma intuitiva y muy completa, que cuenta con hasta 6 lenguajes de programación distintos con los que llevar a cabo el tratamiento de datos, así como otras funcionalidades como la de crear visualizaciones con las que poder desarrollar un panel de control del usuario completo y funcional, como el mostrado en la Figura 1.1.

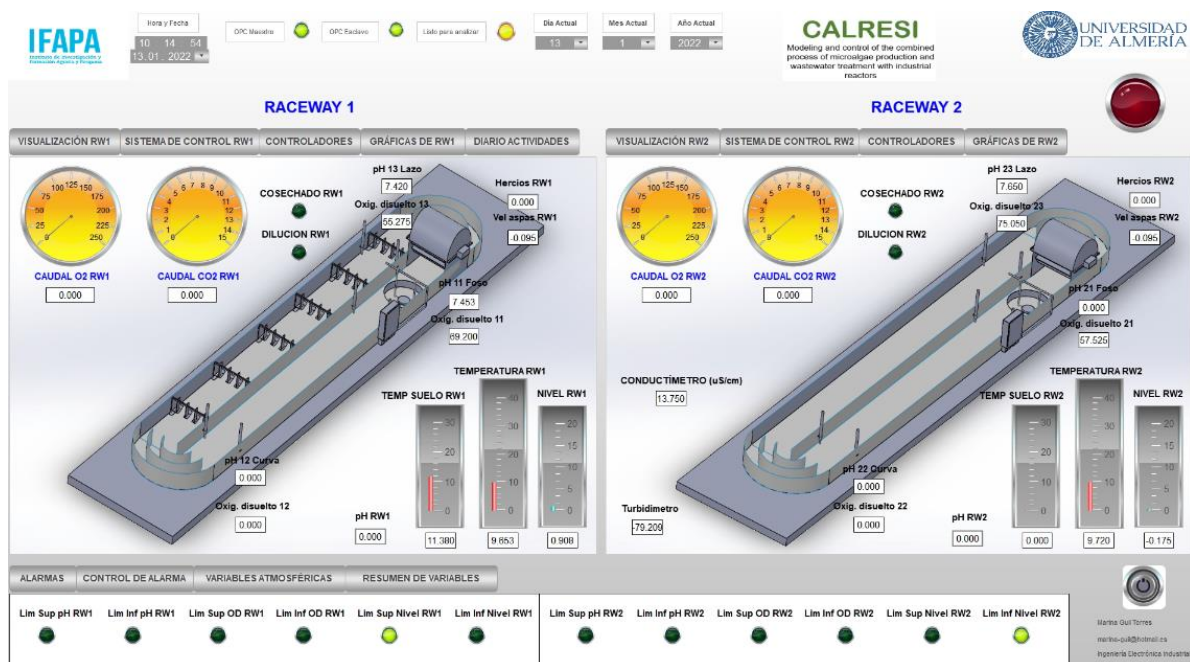


Figura 1.1. Panel del usuario resultante.

Sin embargo, este software también presenta inconvenientes como la necesidad de un PLC que se encargue de la ejecución del proyecto elaborado en dicho entorno de desarrollo. Si bien Codesys trae incorporado un PLC Virtual para tal fin, este presenta limitaciones de uso como un tiempo de ejecución máximo de 2 horas consecutivas. Para este proyecto se hizo uso de este autómatas programables puesto que para alcanzar los objetivos necesarios resultaba más que

suficiente. Sin embargo, para una implementación futura sería necesaria la adquisición de un PLC adicional en la planta.

Otro punto a tener en cuenta es que Codesys solo permite la comunicación con otras aplicaciones o dispositivos a través de su propio servidor OPC y actuando como Servidor. Es decir, debe ser desde él la declaración de las variables compartidas con otros a través del servidor. Para la comunicación con Matlab, que no es posible realizar internamente como se hacía en LabVIEW (Figura 1.2.a), esto no resulta un gran problema puesto que el envío de datos por el servidor es desde Codesys a Matlab y vuelta (Figura 1.2.b).

Sin embargo, la comunicación con los PLCs, donde se registran los datos muestreados de los reactores y la estación meteorológica, de los que se hablará en el siguiente capítulo de la memoria, y a donde se enviarán las nuevas órdenes a ejecutar por los actuadores, si se ve afectada. En tal caso se hablaría de una comunicación Servidor/Servidor, lo que no es posible efectuar directamente. Es entonces cuando entra en juego Matrikon OPC Data Manager, una aplicación que permite el intercambio de datos entre servidores distintos (Figura 1.2.b).



Figura 1.2. Esquema de las comunicaciones entre aplicaciones.

Este software, al igual que el servidor OPC de Codesys, se ha instalado como una versión de prueba gratuita, por lo que si se optase por implementar esta herramienta SCADA en la planta real debería adquirirse el producto completo.

Con todo ello, aunque en un inicio resultaba atractiva la idea de desarrollar el SCADA en el entorno de desarrollo de Codesys, el cual, si es gratuito, hay que tener en cuenta que las otras aplicaciones necesarias para poner en funcionamiento la herramienta no lo son. De esta manera, se pone en tela de juicio si esta implementación resulta más adecuada que la actualmente instalada, puesto que el principal motivo para querer sustituirla era su coste de mantenimiento. En tal caso debería hacerse un estudio en profundidad de la funcionalidad y los costes asociados a cada herramienta y sopesar que opción resulta más acertada.

Capítulo 2: Materiales y Métodos

Este capítulo alberga toda la información necesaria sobre la instalación a la que irá vinculada la herramienta SCADA resultante de este proyecto y un análisis de los instrumentos empleados para llegar a dicho resultado.

2.1. Descripción de la instalación

Conocer la instalación a la que se asociará la herramienta SCADA resulta de primordial importancia, ya que hace tomar consciencia de que variables habrá que tener en cuenta en el diseño del SCADA, además de cuáles de estas variables serán meramente de visualización en el panel del usuario y cuales servirán para tener un impacto real sobre el sistema, al cambiar, por ejemplo, una consigna o parámetro sobre el panel.

En este caso, el proyecto se centra en dos fotobiorreactores abiertos o, también llamados, raceways, los cuales a su vez cuentan con una serie de sensores y actuadores; una estación meteorológica y dos autómatas programables que comunicarán los reactores con el SCADA resultado de este trabajo.

2.1.1. Reactores Raceway

Los fotobiorreactores abiertos, los elementos principales de estudio de este proyecto, son grandes estanques con poca profundidad, diseñados de este modo con el fin de contribuir así al aumento de la productividad mediante una mejor penetración de la luz. [7]

La Figura 2.1 muestra una imagen de los dos reactores objeto de estudio de este proyecto, localizados en el centro IFAPA de la Junta de Andalucía, anexo a la Universidad de Almería.



Figura 2.1. Vista de los fotobiorreactores raceways del centro de experimentación IFAPA en La Cañada de San Urbano, Almería.

Estos reactores están divididos en tres partes: un foso bajo el suelo donde se realizan los procesos de inyección de gases, un canal o receptor solar diseñado en forma de U por donde se hacen circular las microalgas para que puedan recibir la radiación solar y realizar la fotosíntesis, y unas palas para poder impulsar el cultivo a lo largo del canal [7]. En la Figura 2.2. se muestra como es la distribución y las dimensiones concretas de estas partes constituyentes.

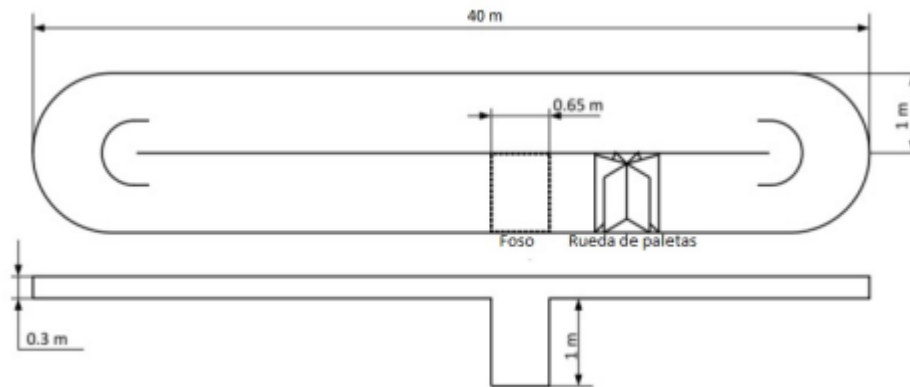


Figura 2.2. Esquema de los Raceways. Fuente [16]

Este tipo de reactores son los más ampliamente utilizados en la industria, cubriendo más del 90 % de la producción global de microalgas debido a su bajo coste (inferior a 10 €/m²) y su fácil escalado. Por otro lado, también poseen un bajo consumo energético, siendo así relevantes para su uso en aplicaciones que no requieran biomasa de alto valor, tales como el tratamiento de aguas residuales y la producción de biocombustibles. Sus principales desventajas están relacionadas con la posible contaminación debido a ser sistemas abiertos y por la escasez de control de las condiciones de operación. Por este motivo, para aquellas especies que requieran bajos niveles de contaminación se aconseja el uso de fotobiorreactores cerrados. [7]

Los reactores de este proyecto trabajan actualmente con microalgas de la cepa *Scenedesmus almeriensis*. Se caracteriza por su resistencia a los contaminantes, por tener un rango de pH que se encuentra entre 3 y 10, aunque la ratio de fotosíntesis se sitúa cercano a su máximo con valores de entre 5,7 y 8, y por un rango de temperatura que varía entre 12 y 46 °C, aunque su temperatura óptima se ubica entre 26 y 34 °C.[8]

Los procesos que tienen lugar en los reactores se pueden resumir en tres principalmente. La inyección de gases (O₂ y CO₂) y de agua en el foso con las microalgas con el fin de alcanzar unos valores óptimos que aseguren un entorno o medio adecuado para que el vegetal pueda realizar la fotosíntesis. Luego, la mezcla e impulsión de las microalgas a lo largo del foso, gracias a las palas de impulsión de la rueda, que contribuye a que estas hagan mejor la fotosíntesis. Y, por último, una etapa de cosechado del producto resultante. Para poder llevar a cabo todo ello, cada reactor cuenta con una serie de actuadores.

Una rueda de paletas accionada por un grupo motorreductor formado por un motor CEMER (IE1-MSE801-4 0,55 kW, 1370 rpm) y un reductor de WEB Iberica S.A. Este grupo es controlado eléctricamente por un variador de frecuencia (CFW 08 WEB Iberica, S.A.) para proporcionar una velocidad constante en el fluido de 0.2 m/s. [16]

La inyección de los gases en cada reactor se realiza a través de 3 discos difusores (AFD 270, EcoTec) colocados en el foso.

Cada reactor cuenta a su vez con 2 válvulas proporcionales, las cuales se caracterizan por ofrecer un control más preciso del caudal al depender de una señal eléctrica de entrada que los gobierna. Esta cualidad permite que no solo se puedan implementar controladores Todo/Nada. En este caso los controladores instalados son de tipo PID y se encargan del control de pH y de oxígeno disuelto (OD).

La válvula proporcional de CO₂ se trata de una válvula piezoeléctrica de la marca FESTO, modelo VEMD, que proporciona un caudal máximo de 20 L/min y tiene una respuesta totalmente lineal en su rango de operación. Mientras que la válvula proporcional de aire es una válvula de la marca Camozzi, modelo MX2-1/2, con un regulador electrónico de la misma marca, modelo K8P. Este conjunto proporciona un caudal máximo de 6000 L/min.

Además de estas, en cada raceway hay una electroválvula de riego de tipo Todo/Nada de la marca Cepex, modelo L10-J2, que se encarga del llenado del fotobiorreactor con agua procedente de una balsa localizada en las instalaciones del centro IFAPA. Con ella también se subirá el nivel del reactor cuando la consigna de altura del medio se encuentre por encima de la actual.

Por último, cada reactor cuenta con una bomba sumergible, instalada en el foso del fotobiorreactor, de la marca ESPA, modelo VXV 110OAS, cuyas funciones son extraer el medio hacia los tanques de cosechado y, como ocurriría con la válvula de riego, bajar el nivel del reactor cuando la consigna de altura del medio se encuentre por debajo de la actual.

Por otro lado, los reactores raceways tienen instalados una serie de sensores, según la ubicación mostrada en la Figura 2.3.

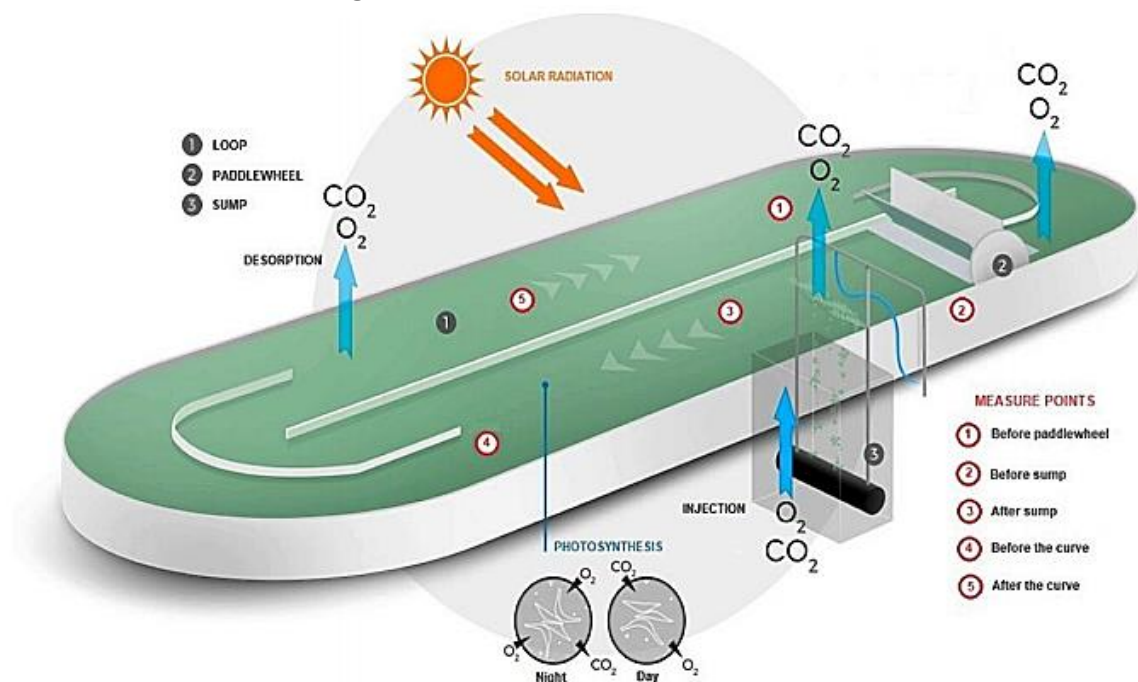


Figura 2.3. Localización de los sensores en cada reactor. Fuente [16]

En consecuencia, por cada reactor existen:

- 5 sensores de pH de la marca Crison, modelo 53 34, recomendados para aguas residuales y/o con partículas en suspensión. Este modelo proporciona valores de pH entre 0 y 14 para un rango de salida de 4 a 20 miliamperios (mA). Con ellos se obtendrá la lectura de pH del medio en diversos puntos del reactor.
- 5 sensores de oxígeno disuelto. Estos sensores se encuentran emparejados con los de pH y se encargan de medir la cantidad de oxígeno gaseoso disuelto en el agua. Los sensores instalados son de tipo membrana de la marca Mettler Toledo, modelo InPro 6050.

Estos sensores electroquímicos, o galvánicos, logran la medida a través de una membrana semipermeable que cubre dos electrodos, ánodo y cátodo, sumergidos en un electrolito. El flujo de electrones al depender de la concentración de oxígeno presente en la superficie de la membrana permite determinar el valor de oxígeno disuelto.

- Sensores de temperatura. Por un lado, para medir la temperatura del suelo donde se ubica el reactor, se hace uso de una termorresistencia (RTD), tipo PT100, cuya medida se basa en la variación de la resistencia de un conductor con la temperatura, por el principio de dispersión de los electrones en el metal.

Por otro lado, para medir la temperatura del medio, se hace uso de termistores tipo NTC (Coeficiente de Temperatura Negativo), ubicados en los sensores de membrana de oxígeno disuelto. La medición en este caso se basa en la variación de resistencia que provoca la variación de la concentración de portadores de un semiconductor al cambiar su temperatura.

- 1 sensor de ultrasonidos de la marca Wenglor, modelo UMD402U035, ubicado próximo a la zona 1. Este calculará la altura del medio, en centímetros, a partir de la distancia de este a la superficie del medio y una recta de calibración.
- 2 caudalímetros de la marca SMC. Con ellos se obtiene el valor de caudal, en L/min, de CO₂ y O₂ inyectados en el foso. En el caso del CO₂ se ha utilizado el modelo PF2M7, que permite un caudal de lectura máximo de 25 L/min. Mientras que, en el caso del O₂, se ha optado por el modelo PFMB, que alcanza lecturas de hasta 500 L/min. Ambos con un rango eléctrico de salida de 4-20mA. [16]

2.1.2. Estación meteorológica.

Además de los datos propios de cada reactor, es necesario tener un registro de las condiciones meteorológicas en las que se encuentran, sobre todo al tratarse de reactores de tipo abierto. Una estación meteorológica con sensores de diversas marcas se encarga de esta tarea. La Tabla 2.1. recopila todos los componentes que constituyen dicha estación.

Componente	Modelo	Rango de medida	Precisión
Sensor de temperatura y humedad relativa	Delta Ohm HD 9008TRR	Temp: -40 a +80 °C Hum: 0 a 100%	± 0.15 °C ±0,1 % medida ± 1,5 % (0 a 90%) ± 2 % (el resto)
Sensor de radiación global	Kipp & Zonen CM 6B	0 a 2000 W/m ²	± 5 %
Sensor de velocidad del viento	Thies Clima 4.3400.30.000	0,5 a 35 m/s	±/ 0,5 m/s o ± 5 % medida

Tabla 2.1. Recopilación de los componentes de la estación meteorológica y sus características.

2.1.3. Autómatas programables (PLCs)

Los sensores y actuadores anteriores se encuentran conectados a dos autómatas programables (PLCs) de la marca Schneider: uno de ellos actúa como esclavo, mandando datos de lectura al maestro, mientras que el otro ejerce, valga la redundancia, como maestro y contiene el código de control de los reactores.

El PLC esclavo es un modelo TM251MESE con 3 módulos de entradas analógicas. El maestro, por su parte, es un modelo TM241CE24T/U y cuenta con 5 módulos de expansión, 2 de salidas y 3 de entradas analógicas, que le permiten realizar una gran cantidad de toma de datos.

Los módulos de salidas analógicas, modelo TM3AQ4/G, permiten la conexión de 4 actuadores analógicos con salida configurable en corriente (4...20 mA o 0...20 mA) o tensión (-10...10 V o 0...10 V) y 12 bit de resolución. Los módulos de entrada analógicas son del modelo TM3AI8/G y permiten la conexión de 4 sensores analógicos con entrada configurable de iguales características que los módulos de salida.

Ambos autómatas se encuentran conectados mediante Ethernet Industrial. El maestro, a su vez, a un ordenador industrial IBOX-601 con procesador Intel i5-6200U y 16 Gb de memoria RAM. De este modo todos los valores de lectura y escritura presentes en los autómatas se encuentran disponibles en el ordenador, para su empleo en la herramienta SCADA a través del servidor OPC. [16]

2.2. Entornos de desarrollo

Para cualquier industria que se encuentre automatizada, supone de gran atractivo la idea de tener un sistema de control y supervisión que no necesite de la presencia en todo momento de un ingeniero especialista que interprete la información del sistema que este proporciona. Por ello, se introducen las herramientas SCADA o HMI mediante las cuales, sin tener un conocimiento profundo del sistema, cualquier operario que esté supervisando el proceso industrial, puede interpretar que sucede en él, extrayendo información útil sobre el mismo y, por tanto, actuar sobre el proceso y solucionar posibles incidencias.

Un SCADA es una herramienta de visualización destinada a la supervisión de un proceso

industrial, que, además, se encarga de recopilar información de interés sobre el sistema para su posterior uso o almacenamiento en registros o informes.

Los SCADA son sistemas de automatización o sistemas de control industrial que involucran control directo o la comunicación con redes de automatización industrial y máquinas; sistemas de adquisición de datos; históricos y servidores de almacenamiento de datos; sistemas de control industrial utilizando PLCs y RTUs o sistemas de seguridad y procesos, entre otros. [9]

Este método de supervisión es el más extendido en la industria, ya que ofrece todo lo necesario para realizar el seguimiento completo de una planta automatizada. A través de entornos de programación como LabVIEW o Codesys, los cuales se tratarán en este proyecto, es posible diseñar dichas herramientas.

2.2.1. LabVIEW

LabVIEW es el nombre de un entorno de programación gráfica diseñado por la compañía estadounidense National Instruments Corp., la cual se dedica al desarrollo de aplicaciones que involucren la adquisición, el control, el análisis y la presentación de datos, así como el desarrollo de sistemas de pruebas automatizadas de investigación, validación y producción. Lo que le convierte en un programa de ingeniería de sistemas ideal para la elaboración de herramientas SCADA. [12]

Aunque en sus inicios se orientaba su uso hacia la instrumentación electrónica, actualmente su alcance se extiende a otros campos como las comunicaciones, la informática y la programación.

Se asemeja a otros entornos de desarrollo comerciales que utilizan el lenguaje C o BASIC. Con LabVIEW se pueden crear algoritmos de análisis de datos y elaborar interfaces de usuario, medir sistemas físicos, crear sistemas de pruebas de producción, contrastar diseños electrónicos o establecer una estructura de comunicación inalámbrica.

Las principales ventajas de utilizar LabVIEW en lugar de otros entornos comerciales se resumen en los siguientes puntos:

- Se reduce el tiempo de desarrollo de las aplicaciones hasta 10 veces, ya que para su manejo no se requiere gran experiencia en programación, al apoyarse en símbolos gráficos en lugar de lenguaje escrito para construir las aplicaciones. Por ello resulta mucho más intuitivo que otros lenguajes de programación convencionales.
- Con un único entorno de desarrollo se integran las funciones de adquisición, análisis y presentación de datos.
- Cuenta con un compilador gráfico que logra la máxima velocidad de ejecución posible.
- Tiene la posibilidad de incorporar aplicaciones escritas en otros lenguajes, es decir, tiene conectividad a otros lenguajes y protocolos estándares de la industria.
- Tiene miles de funciones de análisis disponibles.
- Cuenta con elementos de visualización interactivos y configurables.
- Controladores para automatizar cada uno de los instrumentos y el hardware de adquisición de datos.
- A diferencia de los lenguajes C o BASIC, los cuales se basan en líneas de texto para crea

el código fuente del programa, LabVIEW emplea la programación gráfica o lenguaje G para crear programas basados en diagramas de bloques.

- Además de las funciones básicas de cualquier lenguaje de programación, LabVIEW dispone para sus usuarios de librerías específicas para la adquisición, análisis, presentación y almacenado de datos.
- Posee herramientas que facilitan la depuración de los programas.

2.2.1.1. Interfaz de LabVIEW

Los Instrumentos Virtuales o VIs son el nombre que reciben los programas de LabVIEW, llamados así porque se asemejan en apariencia y funciones a instrumentos físicos reales, como osciloscopios y multímetros.

Una VI se conforma de dos ventanas diferentes interrelacionadas entre sí: la ventana del panel frontal y el diagrama de bloques (Figura 2.4). La ventana del panel frontal es la interfaz de usuario del programa, mientras que el diagrama de bloques es la ventana donde se añade código de fuente gráfica para controlar los objetos del panel frontal.

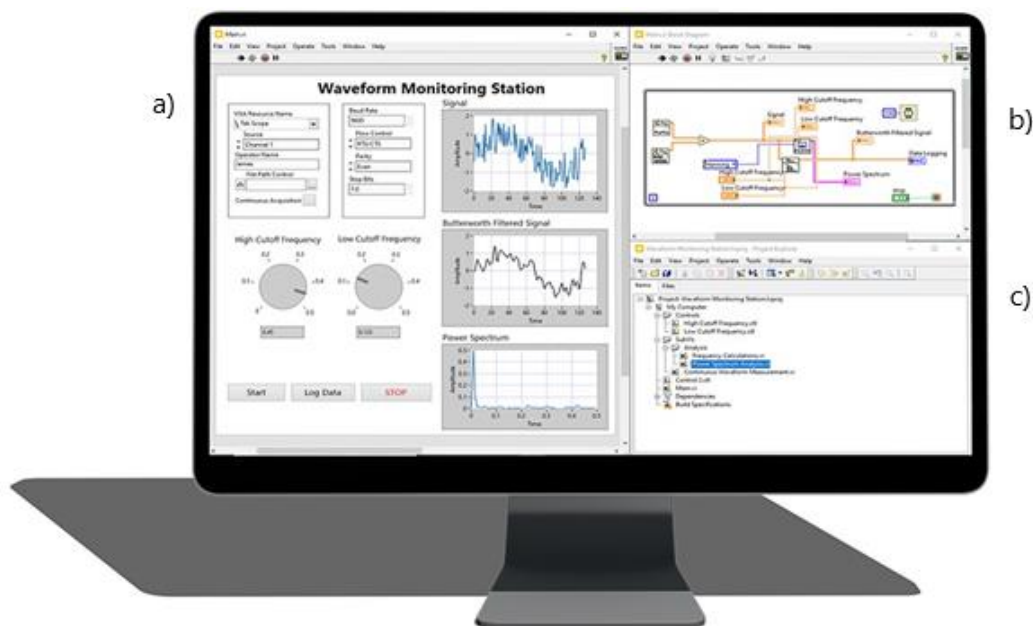


Figura 2.4. Ejemplo de un VI en LabVIEW. (a) Ventana de Panel Frontal, (b) Ventana del Diagrama de bloques y (c) Ventana principal de LabVIEW (acceso a herramientas, ventanas y otras configuraciones del programa). Fuente [12]

Con la paleta *Controls* se tiene acceso a los controles e indicadores que se utilizan para diseñar el panel frontal. Está dividida en varias categorías, como se muestra en la Figura 2.5.

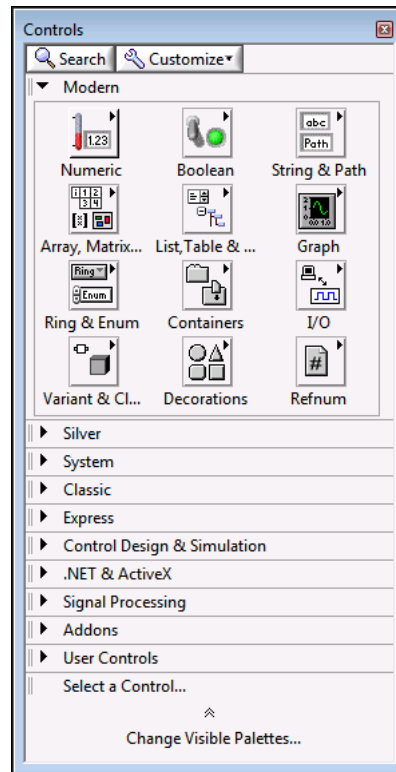


Figura 2.5. Paleta de Controles (con todas las categorías expuestas y la categoría Moderna expandida). Fuente [13]

Los controles simulan dispositivos de entrada de instrumentos y suministran datos al diagrama de bloques del VI, un ejemplo de ellos son los botones o pulsadores. Los indicadores generalmente son gráficas, tablas, lámparas Leds, y simulan dispositivos de salida de instrumentos y muestran los datos que el diagrama de bloques adquiere o genera. Así pues, cuando el usuario interactúa con el panel frontal, por ejemplo, cambiando el valor de entrada de una variable mediante un control, este nuevo dato ingresa al diagrama de bloques del VI, el cual hace el tratamiento correspondiente asociado a la variable, y la devuelve para su visualización en un indicador del panel frontal. De esta forma se puede transmitir información de uno a otro.

Tanto los controles como los indicadores, tiene un tipo de datos asociado a él. Los tipos de datos más frecuentes son numéricos, booleano y cadena de caracteres (ASCII).

Por otra parte, entre los objetos del diagrama de bloques se incluyen terminales, subVIs, funciones, constantes, estructuras y cables, los cuales transfieren datos junto con otros objetos del diagrama de bloques.

Los controles e indicadores en la ventana del panel frontal aparecen como terminales en el diagrama de bloques y, por lo tanto, son los puntos de entrada y salida de intercambio de información entre ambas ventanas. Ambos son fácilmente diferenciables puesto que cuentan con una flecha que indica el flujo de datos y el borde alrededor del terminal de los controles es más grueso que el de los indicadores (Figura 2.6).

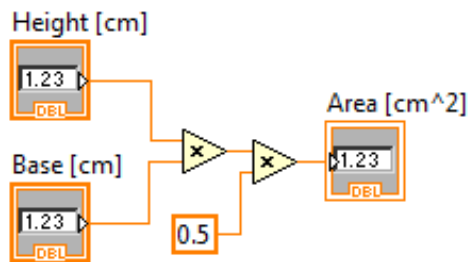


Figura 2.6. Ejemplo de terminales de control y de indicador.

Al igual que los controles e indicadores, las constantes se comportan como entradas y salidas del algoritmo del diagrama de bloques. Por otro lado, los nodos son objetos en el diagrama de bloques que tienen entradas y/o salidas y realizan operaciones cuando el VI se ejecuta. Pueden ser funciones (los elementos de operación fundamentales de LabVIEW), subVIs, Express VIs (nodos que requieren cableado mínimo y se configuran con ventanas de diálogo) o estructuras (Case, For o While). [13]

La paleta *Functions* contiene todos estos elementos que se utilizan para crear el diagrama de bloques. Y al igual que la paleta de Controles, está dividida en varias categorías como indica la Figura 2.7.

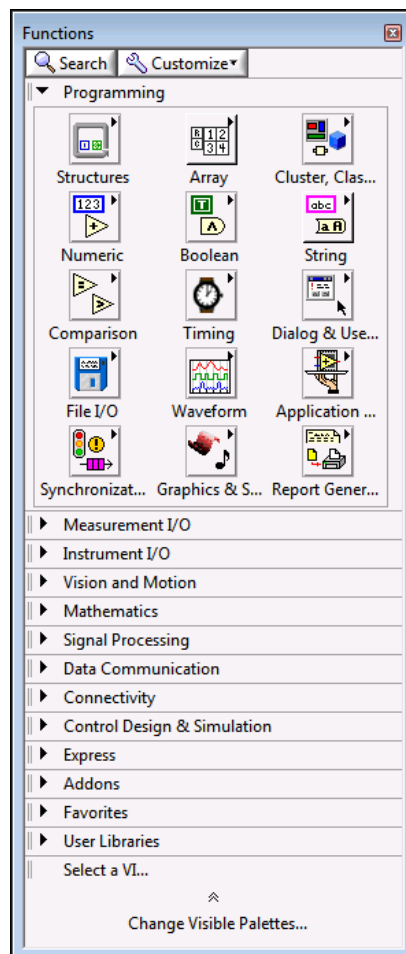


Figura 2.7. Paleta de Funciones (con todas las categorías expuestas y la categoría de Programación expandida). Fuente [13]

2.2.2. Codesys

Los desarrolladores de Codesys lo define como “*una plataforma de software orientado a las tecnologías de automatización industrial cuyo objetivo principal es proporcionar a los usuarios un soporte práctico en la implementación de sus tareas de programación de entornos de automatización.*” [11]

Codesys cubre la ingeniería del proyecto, la programación y la operación en estaciones de trabajo, pero también la ejecución, la depuración del código y de la aplicación, y la evaluación de los dispositivos de campo en el componente controlador.

Codesys es una herramienta de software con la cual poder simular, visualizar y programar equipos. Lo que le permite hacer el seguimiento completo de un proceso industrial sin la necesidad de un software externo. La plataforma se basa en el Sistema de Desarrollo Codesys, una herramienta de programación IEC 61131-3, que se puede encontrar a través de su página online oficial Codesys Store, como se explicará más adelante en “Anexos”.

Cada vez con más frecuencia, Codesys se está imponiendo en la industria frente a sus principales competidores en el sector debido a que, como menciona en su página web:

- Más de 1000 tipos de dispositivos de automatización configurables y programables de 400 prestigiosos fabricantes, como Bosch, FESTO o Schneider, apuestan por el sistema de desarrollo Codesys. Estos, gracias a las diversas funciones que el sistema Codesys posee, pueden ser escalados (modificados o ampliados) con un esfuerzo y recursos calculados.
- El entorno de programación es gratuito, no así otros elementos que ofrece la compañía, como ocurre con el Servidor Codesys OPC DA Server SL y algunas librerías. Sin embargo, existen versiones de prueba con las que poder hacer uso de ellos por un tiempo limitado, en algunos casos.
- Acceso a bibliotecas de aplicaciones y herramientas complementarias probadas, disponibles en la tienda Codesys, que aumenta la funcionalidad del entorno de desarrollo.
- La flexibilidad a la hora de programar el PLC con 6 lenguajes diferentes disponibles: ST, FBD, IL, LD, SFC y CFC. Siendo este último es más recientemente incorporado.

2.2.2.1. Norma IEC 61131-3

En el apartado anterior se mencionó que Codesys es una herramienta de programación IEC 61131-3. Esto quiere decir que se sustenta de la Norma IEC 61131-3 [1]. Para comprender que supone eso es necesario comentar que estipula la norma antes.

El estándar internacional IEC 61131 es una recopilación de estándares referentes a controladores programables y sus periféricos asociados. Su contenido se estructura en 8 partes distintas:

- **Parte 1: Información General.** Donde se establecen las definiciones y las principales características significativas en lo referente a la selección y aplicación de los controladores programables y sus periféricos asociados.

- **Parte 2: Equipo requerimientos y pruebas.** En esta sección se especifican los requisitos del equipo y pruebas vinculadas a controladores programables y sus periféricos.
- **Parte 3: Lenguajes de Programación.** La parte 3 recoge los elementos básicos de programación: las reglas sintácticas y semánticas para los lenguajes, los principales campos de aplicación, las pruebas aplicables y los medios por los cuales los fabricantes pueden expandir o adaptar estos elementos a sus propias implementaciones de controladores.
- **Parte 4: Guías de Usuario.** Se trata de un reporte técnico que sirve de guía de aplicación del estándar a los usuarios finales de los PLCs.
- **Parte 5: Especificación del servicio de Mensajería.** En ella se define la comunicación de datos entre controladores programables y otros sistemas electrónicos usando el “Manufacturing Message Specification”.
- **Parte 7: Programación en lógica difusa.** La parte 7 define los elementos básicos de programación de “lógica difusa” para su uso en controladores programables.
- **Parte 8: Guías para aplicación e implementación de lenguajes de programación.** Se trata de una guía técnica de la parte 3, referente a los lenguajes de programación, para los desarrolladores de software.

IEC 61131-3 es el primer esfuerzo real para estandarizar los lenguajes de programación empleados en automatización industrial a nivel mundial. Se corresponde con la tercera parte del estándar 61131 y es la especificación de la sintaxis y semántica de un conjunto unificado de lenguajes de programación incluyendo el modelo general del software y su estructura como lenguaje.

Una forma sencilla de verlo, es dividiendo el estándar en 2 secciones: por una parte, los elementos comunes y, por otra, los lenguajes de programación.

Elementos Comunes.

Dentro de esta sección se habla de:

- Tipos de Datos. Tipificar los datos previene de cometer, desde una etapa temprana, errores, tales como realizar operaciones entre parámetros de distinto tipo y que, por lo tanto, son inadecuados.

Los tipos de datos más comunes se recogen en la Tabla 2.2.

Clasificación	Tipo de dato	Tamaño	Valores admisibles	Ejemplos / comentario
Booleanos	BOOL	1 bit	0/1 ó TRUE/FALSE	
Cadena de bits	BYTE	1 byte	16#00 a 16#FF	Algunos fabricantes no permiten notación decimal. BYTE#16#B3 WORD#16#3BF0 BYTE#2#01001101
	WORD	2 byte	16#0000 a 16#FFFF	
	DWORD	4 byte	16#00000000 a 16#FFFFFFFF	
	LWORD	8 byte	16#0000000000000000 a 16#FFFFFFFFFFFFFFFF	
Entero decimal	SINT	1 byte	-128 a +127	En muchos PLC si no se especifica el tipo de dato se selecciona por defecto DINT DINT INT#16#4FA2 DINT#10#4350
	INT	2 byte	-32768 a +32767	
	DINT	4 byte	-2147483648 a +2147483647	
	LINT	8 byte	-9223372036854775808 a +9223372036854775807	
	USINT	1 byte	0 a 255	Valores enteros positivos UDINT#23500 USINT#16#2F
	UINT	2 byte	0 a 65535	
	UDINT	4 byte	0 a 4294967295	
	ULINT	8 byte	0 a 18446744073709551615	
Coma flotante	REAL	4 byte	7 dígitos significativos	REAL#-150.0
	LREAL	8 byte	15 dígitos significativos	
Tiempo	TIME	8 byte	T#-9223372036854.775808ms a T#+9223372036854.775807ms	TIME#60s T#1d2h15m30s500ms
Fecha y hora	DATE	8 byte	D#1970-01-01 a D#2554-07-21	Las fechas admisibles pueden variar en función del fabricante del PLC
	TIME_OF_DAY	8 byte	TOD#00:00:00.000000000 a TOD#23:59:59.999999999	TOD#12:25:30.55
	DATE_AND_TIME	8 byte	DT#1970-01-01-00:00:00.000000000 a DT#2554-07-21-23:34:33.709551615	DT#2020-10-23-19:30:00.00
Cadena	STRING	0-1985	Código de caracteres UTF8	

Tabla 2.2. Clasificación de los tipos de datos. Fuente [14]

- **Variables.** Las variables son asignadas a direcciones de hardware explícitas (como E/S) en la configuración, recursos o programas. De esta manera se les da a los programas una independencia de alto nivel del hardware, soportando el reúso del software.

Las variables pueden declararse a nivel local, lo que quiere decir que están limitadas a la unidad de organización en la cual son declaradas. Esto significa que pueden ser declaradas nuevamente en otras partes del proyecto sin ocasionar conflicto y generar errores en el programa. Si las variables requieren un alcance mayor deben ser declaradas como variables globales.

Cuando se declaran estos parámetros se les puede asociar un valor de inicialización, de modo que al arranque y al reinicio “en frío” comiencen la nueva ejecución de programas desde un valor establecido.

- **Configuración, recursos y tareas.** El estándar define que: al nivel más alto, el software completo que se requiere para solucionar un problema de control particular puede ser formulado como una configuración. Una configuración es específica a un sistema de control particular, incluyendo el arreglo del hardware, recursos de procesamiento, direcciones de memoria para los canales de E/S y otras capacidades del sistema.

Dentro de una configuración, se pueden definir una o más tareas, las cuales controlan la ejecución de un conjunto de programas y/o bloques de función. Las tareas pueden ser ejecutadas periódicamente o a la ocurrencia de algún evento disparador, por ejemplo, un cambio de consigna en una variable.

Los programas están constituidos por un conjunto de elementos de software escritos en cualquiera de los lenguajes definidos por IEC. Típicamente constan de redes (networks)

o funciones y bloques de función que son capaces de intercambiar datos. Las funciones y los bloques de función son los bloques de construcción básicos y contienen una estructura de datos y un algoritmo.

IEC 61131-3 agrega mayores capacidades a los PLC convencionales, como multiprocesamiento y conducción por sucesos.

• Unidades de Organización de Programa (Program Organization Units o POUs). Se distinguen 3 tipos:

- Funciones Estándar y Funciones Definidas por el Usuario. Las funciones estándar, tales como ADD (suma), son aquellas funciones fundamentales del propio software que comprenden operaciones comúnmente usadas en todo lenguaje de programación. Las funciones definidas por el usuario, como su propio nombre indica, son aquellas creadas por el usuario y que una vez se definen pueden ser empleadas en otros programas cuantas veces se desee.
- Bloques de Función (Function Blocks, FBs). Son equivalentes a los circuitos integrados y representan una función de control especializada, como un PID. Contienen datos, así como el algoritmo, de modo que pueden conservar información de su estado. Esto las diferencia de las funciones, las cuales siempre muestran la misma salida para las mismas entradas. Un bloque de Función no tiene por qué, ya que puede tener en cuenta cómo llegó a su estado actual.

Los Bloques de Función tienen una interfaz bien definida y su parte interna oculta (como una caja negra). Lo que permite una clara separación entre diferentes niveles de programadores o personal de mantenimiento.

Una vez definido puede ser usado reiteradas veces en el mismo programa, diferentes programas o diferentes proyectos, es decir, son reusables.

Los Bloques de Función pueden ser escritos en cualquiera de los lenguajes de programación e incluso en "C". También pueden definirse nuevos bloques basados en los ya existentes.

- Programas. Un Programa es una red de funciones y bloques de función que puede ser escrito en cualquiera de los lenguajes de programación definidos. El estándar permite dos maneras de desarrollar un programa: partiendo de una visión general para luego resolver los detalles (hacia abajo) o viceversa (hacia arriba).

Dentro del estándar en sí mismo no están especificados todos los recursos que uno puede esperar de un ambiente de programación moderno como Operación mediante "Mouse", menús desplegables o funciones de hipertexto. Por esa razón el estándar permite implementaciones parciales en varios aspectos. Estas dependen de cada suministrador, proporcionándoles gran libertad de acción, pero el usuario (o cliente) debe estar informado al respecto durante su proceso de selección.

Lenguajes de Programación

Dentro del estándar se definen sintáctica y semánticamente diversos lenguajes de programación. La elección de un lenguaje de programación u otro depende de: la formación y experiencia del programador, el problema que aborda, el nivel de descripción del problema, la estructura del sistema de control y la Interfaz con otras personas o departamentos.

Los lenguajes pueden ser de tipo textual o gráfico y se distinguen 4 principales:

- a) Texto estructurado, ST. De tipo textual, es un lenguaje poderoso de alto nivel, con sus raíces en Ada, Pascal y C, por lo que se debe tener un cierto conocimiento previo de programación para su uso. Contiene todos los elementos esenciales de un lenguaje de programación moderna, incluyendo selección del flujo de ejecución (IF y CASE) y lazos de iteración (FOR, WHILE y REPEAT), que pueden ser anidados. Por ello resulta idóneo para la definición de bloques de función complejos.
- b) Lista de instrucciones, IL. De tipo textual, se asemeja a los programas en ensamblador. Consta de una secuencia de instrucciones, cada instrucción se inicia en una línea nueva y contiene un operador y, según el tipo de operación, uno o varios operandos por comas. Delante de una instrucción puede encontrarse un identificador o marca, seguido de dos puntos, que la identifica y que puede utilizarse como destino de salto, por ejemplo. El último elemento en una línea debe ser un comentario y pueden insertarse líneas vacías entre instrucciones.
- c) Diagrama de contactos, LD. De tipo gráfico, está basado en la representación de la lógica de contactos. Se emplean variables de tipo booleana y, como ocurre en los diagramas de conexiones de circuitos eléctricos, pueden utilizarse elementos como contactos y bobinas. Este lenguaje tiene una estructura simple.
- d) Bloques funcionales, FBD. De tipo gráfico, es un lenguaje de alto nivel que permite resumir funciones o programas de una manera visual para el usuario, el cual se encarga únicamente de la programación funcional de su rutina. Es común en la industria de procesos y representa el comportamiento del programa mediante un conjunto de bloques de funciones a la manera de los diagramas de circuitos de electrónica, es decir, en términos de flujo de señales entre elementos de procesamiento.

A parte de los mencionados anteriormente, existe otros dos tipos ampliamente utilizados:

- e) Diagrama de funciones secuencial, SFC. De tipo gráfico, describe el comportamiento secuencial de un programa de control, estructurando su organización interna y descomponiéndolo en partes más fácilmente manejables, manteniendo una visión general de este. Deriva de sus antecesores "Petri Nets" y del IEC848 Grafset y se compone de pasos, bloques de funciones y transiciones.

Cada paso representa un estado particular del sistema que se está supervisando. Una transición se asocia con una condición o condiciones que cuando se cumple es causa de que el paso previo se desactive y el paso próximo se active. Los pasos están ligados a bloques de funciones, que ejecutan algunas acciones de control pertinentes a dicho paso.

Cada elemento de este diagrama puede ser programado en cualquiera de los lenguajes de programación, incluido el propio SFC.

- f) Diagrama continuo, CFC. Es un lenguaje de programación gráfico basado en el lenguaje de diagrama de bloques de funciones. Sin embargo, en contraposición a este, no hay redes. CFC permite colocar con libertad los elementos gráficos, lo que a su vez permite la presencia de lazos de realimentación. [1]

2.3. Servidor OPC

Una situación cotidiana a la que se enfrentan las empresas industriales digitalizadas es la necesidad de establecer comunicaciones entre los diversos dispositivos que conforman sus sistemas o plantas de producción, y no solo entre componentes del mismo tipo o fabricante, sino con otros con características y origen completamente diferentes. Por esta razón, se hace necesaria la incorporación de un dispositivo capaz de propiciar dicha comunicación, es decir, la incorporación de un servidor OPC. El servidor OPC hace las veces de puente de comunicación entre la planta real, el PLC que la controla y el SCADA para la recopilación de datos y actuación sobre la misma.

Para realizar la comunicación entre los fotobiorreactores y el SCADA es necesario, por lo tanto, hacer uso de un servidor OPC. El usuario, a través del SCADA, hace la función de cliente para ordenar, a partir de los PLCs que ejecutan el algoritmo de control, la siguiente acción o cambio de estado de alguna de las variables asociadas a los reactores, los cuales hacen la función de proveedor/servidor, procesando los datos recibidos y devolviéndolos para su visualización.

Esta implementación es la que existe actualmente en la planta. Sin embargo, para el nuevo SCADA es preciso no solo un servidor OPC sino dos, el actual más otro nuevo de la marca Codesys, OPC DA Server (versión 3.5.17), que se encargará de que esta interfaz sea capaz de interactuar con el resto de componentes de la comunicación.

Esta clase de servidores está ampliamente estandarizada. Existen multitud de versiones como la de MatrikonOPC para Modbus, una de las más utilizadas por la industria para la comunicación con PLCs, RTUs, analizadores y otros tipos de controladores. Sin embargo, Codesys solo admite servidores de su marca por lo que se descarta el uso de cualquier otro servidor de diferente fabricante.

2.3.1. Definición y especificaciones OPC

“OPC (Open Platform Communications) es una interfaz estándar que proporciona acceso a los datos de un proceso de automatización. La tarea principal del CODESYS OPC Server es el intercambio de datos (lectura/escritura) con el controlador, por ejemplo, para visualizaciones o para programas de registro de datos de proceso.”, define el grupo Codesys. [15]

Un servidor OPC es una aplicación de software, o driver, que se encarga de propiciar las comunicaciones entre clientes OPC (SCADAs, HMIs, etc.) conectados a él y una o más fuentes de datos (PLCs, controladores, etc.) utilizando sus protocolos propios.

aplicaciones de la empresa y a través de todas las capas empresariales. Esta es una tecnología de comunicación industrial multiplataforma, abierta, orientada a servicios, segura, y con ricos modelos de información, que se encuentra mantenida por la OPC Foundation. El servidor instalado actualmente pertenece a este protocolo.

2.3.2. Características y funcionalidad del servidor

Dentro de la Web de Codesys, el fabricante también comenta las características y funcionalidad de su producto, incluido en la configuración del entorno de programación a través de una licencia de uso:

- **Inicio automático al establecer una conexión de cliente.** El servidor OPC funciona como un programa ejecutable que se inicia automáticamente al establecerse una conexión entre el cliente, el proveedor y el PLC. A partir de ese momento, el servidor será capaz de informar al cliente sobre los valores o el estado de las variables que hayan cambiado.
- **Disparo automático cuando cambia un valor de datos o un estado de datos** (elementos OPC).
- **Examinar la lista de variables** (grupo de elementos o espacio de direcciones). El servidor OPC proporciona todas las variables (“Items in”) que están disponibles en el PLC (“Item Pool” o “Address Space”).
- **Administrar los elementos en la caché de datos**, lo que asegura un acceso rápido a sus valores y estados.
- **Es posible el acceso directo a los elementos del controlador** (sin caché).
- **Organización de los elementos en grupos** (privados o públicos). Los grupos privados pueden ser creados por el cliente arbitrariamente a partir de elementos particulares. Inicialmente no afectan a las agrupaciones que se hayan realizado en el servidor OPC, pero en caso de que sea necesario pueden transformarse en grupos públicos, aunque solo para algunas versiones del producto. Una ventaja de trabajar con grupos privados es el poder activar o desactivar ciertos grupos de variables con un solo comando, dependiendo también de si estos son accesibles.
- **Registrador de datos integrado opcional para diagnóstico.**
- **Soporte multiciente y soporte multi-PLC.** Debido a las características del estándar DCOM es posible acceder a la comunicación desde otro ordenador distinto al que la inició y con más de un solo cliente al mismo tiempo. La aplicabilidad de distintos tipos de lenguajes de programación (C++, Visual Basic, Delphi, Java) es otra de las ventajas de emplear COM.
- **También es posible la conexión OPC a controladores con CODESYS V2.3.** [15]

2.3.3. Interfaces de comunicación

El servidor OPC DA de Codesys admiten varios tipos de interfaces según la comunicación que se pretenda establecer y son: Gateway V3, Gateway V2.3, ARTI, ARTI3, SIMULATION y SIMULATION3.

La interfaz se elige atendiendo al PLC utilizado en particular y debe ser configurado para ello, como ya se indicará en el siguiente capítulo.

2.4. Matlab

MATLAB es una plataforma de programación y cálculo numérico utilizada por ingenieros y científicos para: análisis de datos, visualización de gráficas, desarrollo de algoritmos, creación de modelos y apps, conexión con hardware, cálculo paralelo, uso de MATLAB con otros lenguajes como Python, cálculo en la nube y despliegue en escritorio y web, permitiendo compartir los programas de Matlab.

Combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente.

Cuenta con Apps incorporadas que le permiten ver cómo funcionan diferentes algoritmos con sus datos y realizar iteraciones hasta obtener los resultados deseados y, después, generar automáticamente un programa de MATLAB para reproducir o automatizar el trabajo.

Matlab permite llevar cualquier idea de la investigación a la producción, convirtiendo su algoritmo a código C/C++ y HDL para su ejecución en dispositivos embebidos, siendo capaz de subir el código directamente a la nube para que otros sistemas empresariales tengan acceso a él y, gracias a su colaboración con Simulink, permitiendo el diseño en modelos empleado en simulaciones multidominio, la generación automática de código y la comprobación y verificación continuas de sistemas embebidos.

Por todo ello, Matlab puede ser empleado para sistemas de control, *Deep Learning*, procesamiento de imágenes y visión artificial, *Machine Learning*, mantenimiento predictivo, robótica, procesamiento de señales, comunicaciones inalámbricas, pruebas y medición. [17]

Por todo ello, se ha recurrido a esta interfaz para hacer la mayoría de las labores de control y tratamiento de datos de la herramienta SCADA.

2.5. Matrikon OPC Data Manager

Tradicionalmente las conexiones de entradas y salidas de los sistemas de control para la transferencia de datos se realizaban por cable. Esto provocaba que el intercambio fuera complejo de configurar y caro de mantener. La Tecnología OPC (Open Platform Communications) supone una forma más adecuada para realizar esta comunicación entre sistemas al eliminar los problemas de conectividad causados por las comunicaciones propietarias. Esta tecnología está disponible para todos los principales sistemas de control del mercado.

Los sistemas OPC están habilitados para compartir datos mediante la implementación de una aplicación Cliente/Servidor OPC. Sin embargo, existen casos en los que la aplicación no es un Cliente OPC, sino que ambos son Servidores OPC. Dos servidores OPC no pueden intercambiar datos entre sí, ya que están ideados para responder a las peticiones de Clientes OPC y no son capaces de generar peticiones. MatrikonOPC Data Manager resuelve el problema al actuar como un Cliente OPC para ambos Servidores OPC, solicitando datos a uno de los servidores y enviándolos a otro inmediatamente. De esta manera se logra una forma muy fácil de conectar servidores OPC y, por consiguiente, dispositivos.

MatrikonOPC Data Manager (ODM) es una aplicación de software que permite la transferencia

de datos de un servidor OPC a otro sin importar el fabricante. Actúa como una pasarela de datos entre dos o más sistemas de control (p. ej., PLCs), conectándolos entre sí. Con ODM, esta conectividad se puede lograr con el estándar de un software comercial.

Su diseño permite compartir datos en tiempo real, históricos y de alarmas y eventos entre dos o más sistemas de control de una manera segura y fiable. Además, la interfaz es intuitiva por lo que se agiliza y se simplifica el proceso de configuración, como ya se verá en el siguiente capítulo de esta memoria. [18]

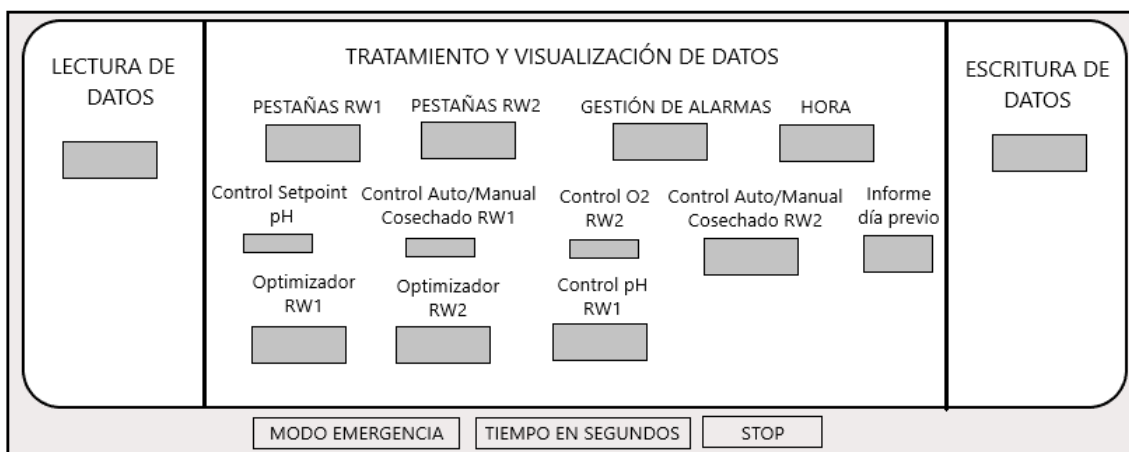
Capítulo 3: Diseño del SCADA

Como ya se ha comentado en numerosas ocasiones, el objetivo principal de este proyecto es reproducir una herramienta SCADA en Codesys, que controle y represente el sistema que conforman un par de fotobiorreactores industriales, tomando como inspiración uno ya existente desarrollado con la plataforma LabVIEW. Por ello, en un primer término, se debe interpretar el funcionamiento del SCADA que se pretende reproducir en el entorno de desarrollo de Codesys.

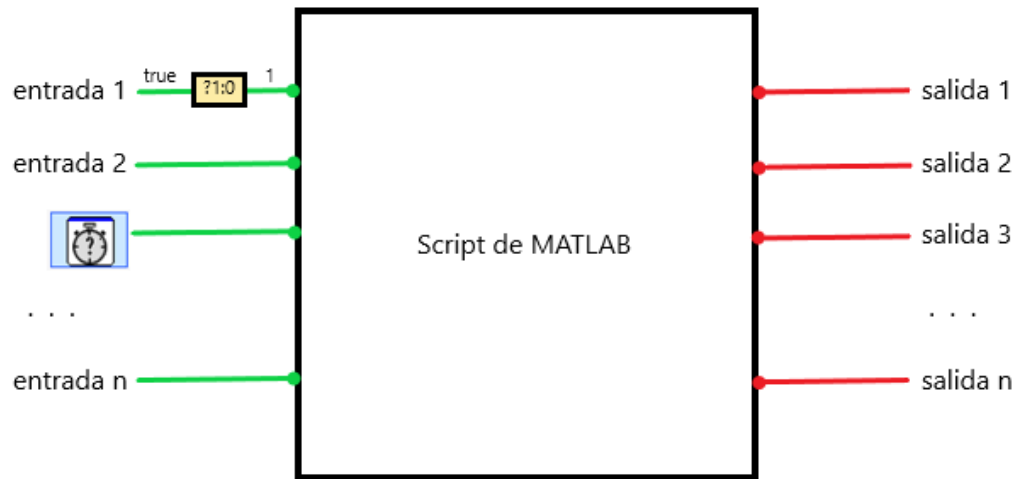
En este capítulo se explicará de manera breve el SCADA actualmente instalado y se mostrará paso a paso como se ha desarrollado la nueva herramienta en Codesys. Además, se comentará cómo realizar la comunicación OPC y la puesta en marcha de la herramienta resultante.

3.1. Análisis de la herramienta actual

El SCADA actual, desarrollado en LabVIEW, parte de un bucle “While Loop” de intervalo de repetición de 400 ms. En su interior se haya una estructura “Sequence Structure” con 3 tramas que se ejecutan en orden de izquierda a derecha: lectura de datos, tratamiento y visualización de datos y escritura de datos, respectivamente. Dentro de cada una se llevan a cabo diversas tareas, indicadas en la Figura 3. 1.a.



a) Esquema del diagrama de bloques del SCADA instalado.



b) Ejemplo de acción donde se recurre a un script de Matlab y a elementos propios de LabVIEW para conseguir sus señales de entrada.

Figura 3.1. Interfaz del SCADA instalado.

Estas tareas, como muestra el ejemplo de la figura 3.1.b, en su mayoría se llevan a cabo por medio de script de Matlab, ya que la plataforma permite la comunicación directa con este programa, y funciones propias de LabVIEW como: estructuras Case; nodos Formula; controles, indicadores y operaciones con variables matemáticas (paleta Numeric), booleanas (paleta Boolean), vectoriales (paleta Array) y con cadenas de caracteres (paleta String); o temporizadores y relojes (paleta Time & dialog). También hay tareas donde solo son necesarios elementos de LabVIEW para llevarse a cabo.

A continuación, se explica la utilidad de algunos elementos utilizados en el diagrama de bloques y que adaptan los datos a los scripts de Matlab:

Bool To (0,1) devuelve un booleano como un entero de 16 bits, esto se traduce en que convierte valores booleanos a numéricos del tipo INT siendo True=1 y False=0.

La función **Matriz booleana a número** convierte una matriz booleana en un número entero o de punto fijo con el bit menos significativo en el índice cero. De modo que, en resumidas cuentas, tiene la misma función que Bool To (0,1), pero aplicado a arrays de variables booleanas.

Elapsed time tiene diversas utilidades, pero en este caso se emplea para recrear señales de pulsos. Una vez que corre el programa, el elemento cuenta el tiempo establecido (elapsed time) y una vez llega a dicho valor genera una señal booleana TRUE que lo confirma. Al estar dentro de un bucle WHILE, cuando comienza la siguiente iteración se resetea y cuenta el tiempo de nuevo y así cíclicamente. De esta manera se genera una señal de pulsos con un periodo determinado.



Format Date/
Time String

La función **Formato de cadena de fecha / hora** muestra un valor de marca de tiempo o un valor numérico como hora en el formato que especifique utilizando códigos de formato de hora.



Decimal String
To Number

La función **Decimal String To Number** convierte los caracteres numéricos en una cadena, comenzando en el desplazamiento, a un entero decimal y lo devuelve en número.



Get Date/Time
In Seconds

Get Date/Time In Seconds obtiene la fecha y hora en segundos. [2]

3.2. Creación del proyecto

La versión instalada del programa es Codesys V3 SP17 Patch 2 (Codesys 3.5.17.2). Su proceso de instalación se describirá en el apartado “Anexos”.

Una vez se ejecuta el programa aparece la pantalla principal de Codesys, Figura 3.2, donde pulsando “Nuevo proyecto”, dentro de “Operaciones base”, permite la creación de un nuevo proyecto.

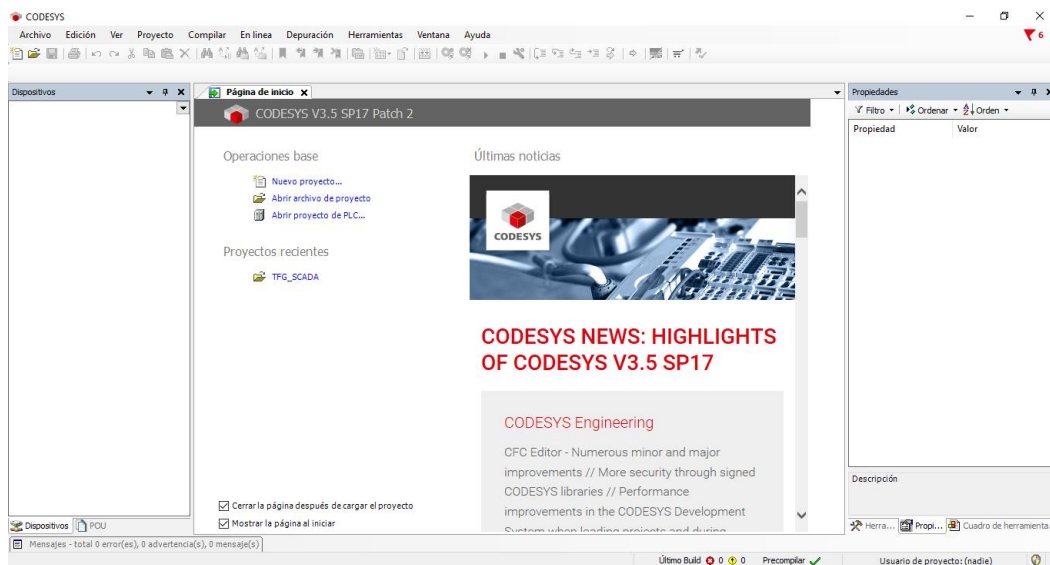


Figura 3.2. Pantalla principal del entorno de desarrollo de Codesys.

Tras eso aparecerá una ventana emergente donde se debe seleccionar el tipo de proyecto que se pretende crear:

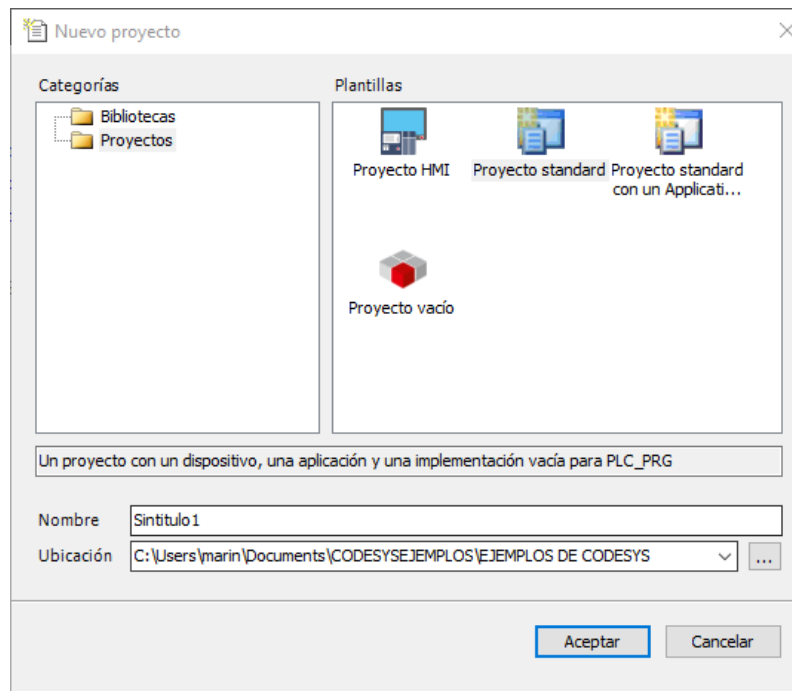


Figura 3.3. Nuevo Proyecto.

Como se observa en la Figura 3.3, hay 4 posibles plantillas de proyecto. Por generalidad se selecciona la opción “Proyecto standard”, ya que esta plantilla trae incorporadas todas las funcionalidades clásicas del programa: un dispositivo, una aplicación y una implementación vacía o unidad de organización de programa, PLC_PRG. Se le asigna un nombre al proyecto nuevo y una ubicación dentro del ordenador y se pulsa “Aceptar”.

Después de esto, se debe seleccionar el dispositivo que se va a utilizar (Figura 3.4), en este caso será un PLC Virtual que trae incorporado la interfaz con su instalación, CODESYS Control Win V3 x64, ya que la versión de Codesys instalada es la x64 bits. En este mismo paso se elige el lenguaje de programación con el que se trabajará en el POU principal que trae por defecto el proyecto estándar (PLC_PRG).

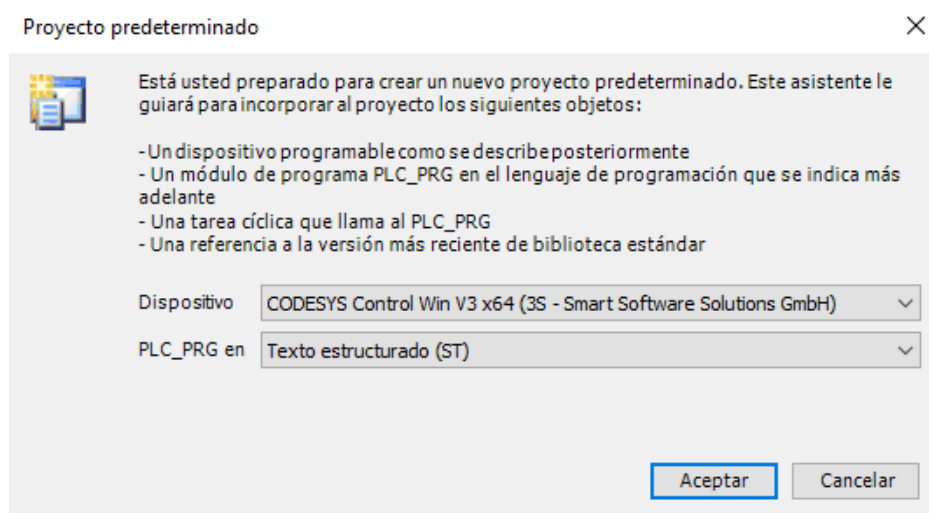


Figura 3.4. Configuración del proyecto.

Con esto se da por concluida la creación del proyecto y ya se puede empezar a trabajar con él.

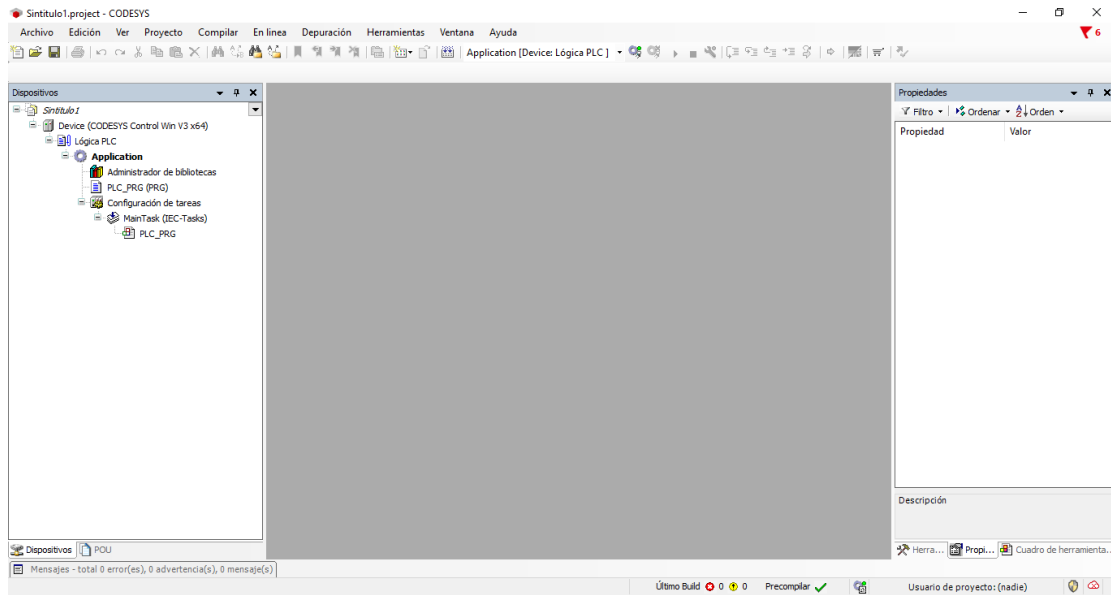


Figura 3.5. Pantalla principal del proyecto creado.

El entorno de trabajo de Codesys se divide en 3 secciones claramente diferenciadas (Figura 3.5): sección derecha, donde se agrupan las herramientas de código y visualización y las propiedades de los elementos que se introducen en el proyecto; la sección central, donde se trabaja como tal en cada unidad de programación o visualización. Y, por último, la sección izquierda, donde se recogen todos los archivos que conforman el proyecto (POUs, visualizaciones, bibliotecas, dispositivos, ...).

Además, en la parte superior del entorno de programación existen diversos menús desplegables con funciones del programa, configuraciones del compilación y ejecución, etc.

3.3. Conexión con el dispositivo PLC Virtual

El primer paso a realizar una vez está creado el proyecto es conectar el dispositivo que se seleccionó al programa para establecer las comunicaciones entre ambos. Para ello se accede a la pestaña Device, disponible en la sección izquierda de la pantalla:

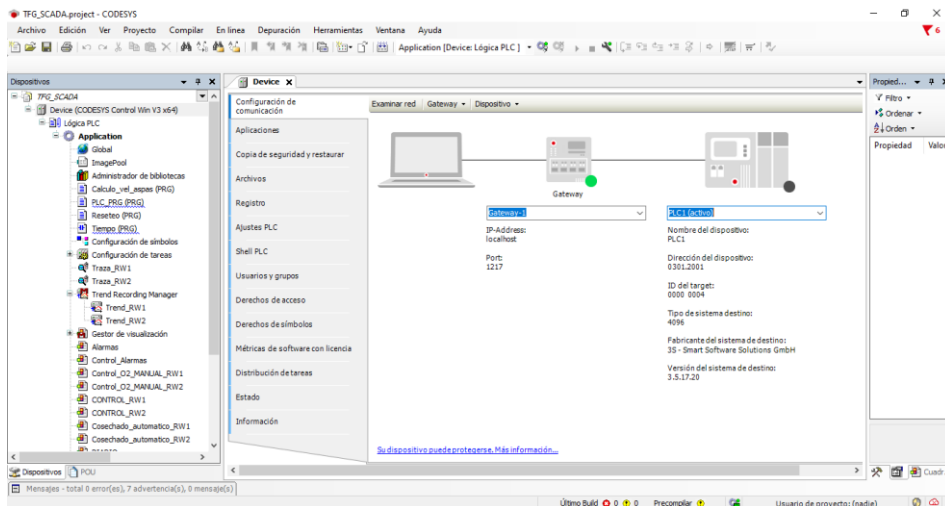


Figura 3.6. Pestaña Device.

Un esquema muy intuitivo de como son las comunicaciones dentro del proyecto se muestra en ella: el ordenador se comunica con el dispositivo PLC mediante un Gateway. Como se observa en la Figura 3.6, el Gateway está activo mientras que el PLC no (bombillas verde y gris, respectivamente), por lo que hay que realizar la conexión de este último para que el proyecto pueda ser ejecutado y puesto en línea.

Para hacerlo, antes de nada, hay que dirigirse a la parte inferior de la pantalla del ordenador y comprobar que ambos elementos estén corriendo. En el caso de que no aparezcan en el desplegable de iconos ocultos, basta con hacer una búsqueda en el ordenador de “Codesys Control Win x64 SysTray” y “Codesys Gateway SysTray”.

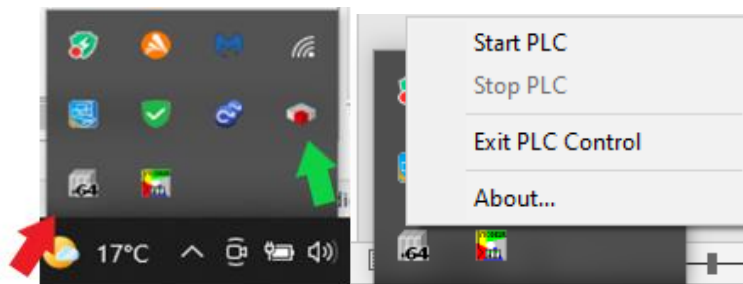


Figura 3.7. Dispositivos PLC y Gateway (fecha roja y flecha verde, respectivamente).

Confirmando lo que ya se había comentado, el Gateway si estaba activo mientras que el PLC no, como indica la Figura 3.7. Presionando sobre el icono y seleccionando “Start PLC” se consigue poner en funcionamiento. A continuación, hay que regresar a Codesys y examinar la red.

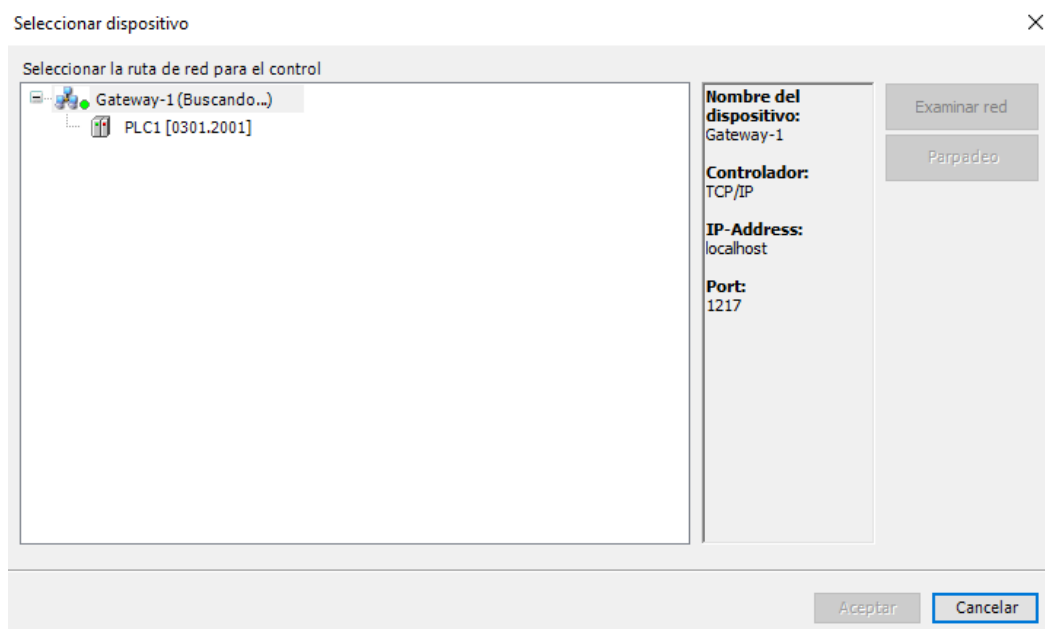


Figura 3.8. Desplegable de Examinar la Red.

Si todo se ha realizado correctamente aparecerá un dispositivo en la red (Figura 3.8). Como método de seguridad, una vez se opta por dicho dispositivo, aparece automáticamente una pantalla emergente, como la de la Figura 3.9, donde se solicita que se introduzcan un Nombre de Usuario y una Contraseña para obtener derechos de acceso al PLC. Estos se establecen la

primera vez que se conecta el PLC al proyecto y luego cada vez que se produzca la comunicación entre ambos, por lo que hay que tener cuidado de no olvidarlos.

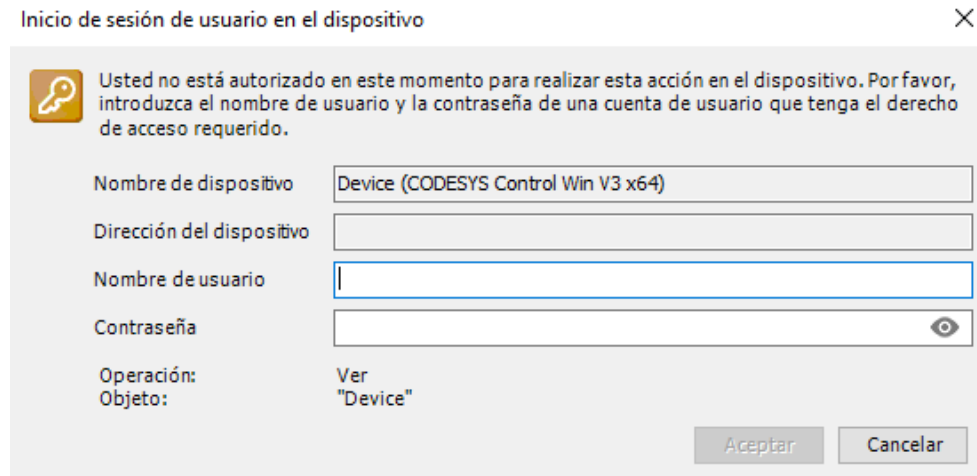


Figura 3.9. Derechos de acceso al PLC.

Cabe decir que dentro de la pestaña “Device” hay un apartado donde se puede cambiar la contraseña que se configuró inicialmente si se desea, al igual que el nombre del dispositivo PLC. En este caso se estableció como nombre del dispositivo “PLC1”, como nombre de usuario “admi” y como contraseña “contraseña”. Con esto ya estaría la comunicación establecida.

3.4. Declaración de variables globales

El siguiente paso a realizar es la declaración de variables globales. Estas, como ya se comentó anteriormente, son aquellas que al emplearse de forma continuada en diferentes partes del proyecto es preferible identificarlas como tal para evitar posibles errores de código en la aplicación.

Para ello se hace clic derecho sobre “Application” y se pulsa en “Agregar Objeto”. Aparecerá un desplegable con todos aquellos objetos que se puede añadir al proyecto (Figura 3.10).

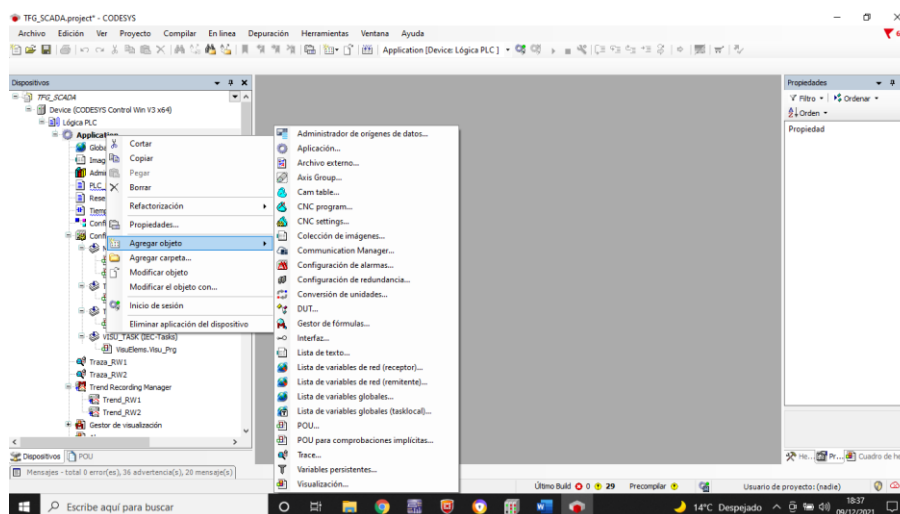


Figura 3.10. Creación de un objeto en la aplicación.

Desarrollo de una herramienta SCADA en Codesys para fotobiorreactores industriales

En esta ocasión se escoge “Lista de variables globales”. Luego se le asigna un nombre al objeto, Global. Tras ello solo hay que introducir el nombre de las variables que pertenecerán a esta categoría junto con el tipo de dato que son.

```
Global x
1 {attribute 'qualified_only'}
2 VAR_GLOBAL
3
4 // Visualizacion base
5   CommStateOK: BOOL;
6   CommStateOK1: BOOL;
7   LISTO_ANALIZAR:BOOL;
8 // -----
9 // Visualizacion RW1
10  TempSuelo1: LREAL;
11  TempRW1: LREAL;
12  NivelRW1:REAL;
13  pH_RW1:REAL;
14  pHRW11:LREAL;
15  pHRW12:LREAL;
16  pHRW13:LREAL;
17  ODRW11:LREAL;
18  ODRW12:LREAL;
19  ODRW13:LREAL;
20  HERCIOS_RW1:REAL;
21  Vel_Aspas_RW1:REAL;
22  AireRW1:LREAL;
23  CO2RW1:LREAL;
24  COSECHADO_RW1:BOOL;
25  DILUCION_RW1:BOOL;
26 // -----
27 // Visualizacion RW2
28  TempSuelo2: LREAL;
29  TempRW2:LREAL;
30  NivelRW2:REAL;
31  pH_RW2:REAL;
32  pHRW21:LREAL;
33  pHRW22:LREAL;
34  pHRW23:LREAL;
35  ODRW21:LREAL;
36  ODRW22:LREAL;
37  ODRW23:LREAL;
38  HERCIOS_RW2:REAL;
39  Vel_Aspas_RW2:REAL;
40  AireRW2:LREAL;
41  CO2RW2:LREAL;
42  COSECHADO_RW2:BOOL;
43  DILUCION_RW2:BOOL;
44  Turbidimetro:LREAL;
45  Conductimetro:REAL;
46 // -----
47 // Sistema de control RW1
48  CONTROL_NivelRW1:BOOL;
49  AUTOMATICO_CONTROL_RW1:BOOL;
50  Setpoint_nivel_RW1:LREAL;
51  Setpoint_nivel_RW1_manual:LREAL;
52  VERIF_RW1:BOOL;
53  Optimizador_RW1:BOOL;
54  BOTON_COSECHADO_RW1:BOOL;
55  BOTON_DILUIR_RW1:BOOL;
56
57   // CON OPTIMIZADOR
58   Setpoint_opt_nivel_RW1:LREAL;
59   HighLim_RW1:LREAL;
60   LowLim_RW1: LREAL;
61   cnt:INT;
62   Et_RW1:BOOL;
63 // AUTOMATICO
64 DiaSeleccionado1: ARRAY [0..6] OF BOOL;
65 NIVEL_DILUCION_RW1: LREAL;
66 NIVEL_COSECHADO_RW1: LREAL;
133 Manual_control_O2_RW2:BOOL;
134 ENTRADA_VALVULA_O2_RW2:REAL;
135 // AUTO
136 Set_OD_RW2:LREAL;
137 Q_Set_OD_RW2:LREAL;
138 kla: LREAL;
139 upid_a:INT;
140 Error_OD_RW2: LREAL;
141 Tiempo_Muestreo:BOOL;
142
143 OFF_CAUDAL:BOOL;
144 //-----
145 // Diario
146 Nombre:STRING(255);
147 Comentario:STRING(255);
148 ANADIR_ENTRADA: BOOL;
149 Diario:BOOL;
150 //-----
151 // Alarmas
152 //-----
153 // Control de alarmas
154 LIM_SUP_HP_RW1: REAL;
155 LIM_INF_HP_RW1: REAL;
156 LIM_SUP_OD_RW1: REAL;
157 LIM_INF_OD_RW1: REAL;
158 LIM_SUP_NIVEL_RW1: LREAL;
159 LIM_INF_NIVEL_RW1: LREAL;
160 LIM_SUP_HP_RW2: REAL;
161 LIM_INF_HP_RW2: REAL;
162 LIM_SUP_OD_RW2: REAL;
163 LIM_INF_OD_RW2: REAL;
164 LIM_SUP_NIVEL_RW2: LREAL;
165 LIM_INF_NIVEL_RW2: LREAL;
166 //-----
167 // Variables atmosfericas
168 Temp_atm: LREAL;
169 Hum_rel_atm:LREAL;
170 Rad_atm:LREAL;
171 Rad_PAR_atm:LREAL;
172 Vel_viento_atm:LREAL;
173 Dir_viento_atm:LREAL;
174 //-----
175 // RESET, STOP, EMERGENCIA
176 Reset_emergencia:BOOL;
177 STOP:BOOL;
178 //-----
179 // OTRAS VARIABLES QUE CONTIENE EL SERVIDOR
180 PID_Fd_RW1: LREAL;
181 PID_Kp_RW1: LREAL;
182 PID_RW1_Output: LREAL;
183 PID_Id_RW1: LREAL;
184 PID_Ti_RW1: LREAL;
185 PID_RW1_PWM_Input: UINT;
186
187 Activar_EVMedioRW1: BOOL;
188
189 Bajar_nivel_RW1: BOOL;
190 Bajar_nivel_RW2: BOOL;
191 Subir_nivel_RW1: BOOL;
192 Subir_nivel_RW2: BOOL;
193
194 Repo_EVMedioRW1: BOOL;
195 Repo_EVMedioRW2: BOOL;
196
197 Control_diurno_RW2_value: UINT;
```

```

67     HORA_DILUCION_RW1: TOD;          199
68     HORA_COSECHADO_RW1: TOD;       200
69                                     201
70     //-----
71     // Sistema de control RW2      203
72     CONTROL_NivelRW2:BOOL;         204
73     Optimizador_RW2:BOOL;          205
74     Setpoint_nivel_RW2:LREAL;      206
75     Setpoint_nivel_RW2_manual:LREAL; 207
76     VERIF_RW2:BOOL;                208
77     AUTOMATICO_CONTROL_RW2:BOOL;   209
78     BOTON_COSECHADO_RW2:BOOL;      210
79     BOTON_DILUIR_RW2:BOOL;         211
80                                     212
81     // CON OPTIMIZADOR            213
82     Setpoint_opt_nivel_RW2:LREAL;   214
83     HighLim_RW2:LREAL;             215
84     LowLim_RW2:LREAL;              216
85     Et:BOOL;                       217
86     // AUTOMATICO                 218
87     DiaSeleccionado2: ARRAY [0..6] OF BOOL; 219
88     NIVEL_DILUCION_RW2:LREAL;      220
89     NIVEL_COSECHADO_RW2:LREAL;     221
90     HORA_DILUCION_RW2:TOD;          222
91     HORA_COSECHADO_RW2:TOD;        223
92     //-----
93     // Controladores RW1          224
94     // CONTROL CO2                 225
95     Todo_Nada_CO2_RW1:BOOL;        226
96     Manual_control_CO2_RW1:BOOL;    227
97     Set_CaudalCO2_RW1:LREAL;        228
98     Pruebas_1:BOOL;                229
99     // AUTO                         230
100    Set_CaudalCO2_RW1_ESC:REAL;      231
101    ITERACION_RW1: INT;              232
102    pH_filtrado_RW1:LREAL;          233
103    RESET_RW1:BOOL;                 234
104    Aux1:LREAL;                      235
105    Aux2:LREAL;                      236
106    Aux3:LREAL;                      237
107    Aux4:LREAL;                      238
108                                     239
109    // CONTROL OD                   240
110    Manual_control_O2_RW1:BOOL;      241
111    // AUTO                           242
112    Set_OD_RW1:LREAL;               243
113    Q_Set_OD_RW1:LREAL;             244
114    Klal:LREAL;                     245
115    upid_al:LREAL;                  246
116    Error_OD_RW1:LREAL;             247
117                                     248
118    Boton_Aire_RW1:BOOL;            249
119     //-----
120    // Controladores RW2            250
121    // CONTROL CO2                   251
122    Todo_Nada_CO2_RW2:BOOL;         252
123    Manual_control_CO2_RW2:BOOL;    253
124    Set_CaudalCO2_RW2:LREAL;
125    Pruebas_2:BOOL;
126    // AUTO
127    ITERACION_RW2: INT;
128    pH_filtrado_RW2:LREAL;
129    RESET_RW2:BOOL;
130    Set_CaudalCO2_RW2_ESC:REAL;
131
132    // CONTROL OD

```

```

ControlpHRW1: BOOL;
ControlpHRW2: BOOL;

CosechandoRW1:BOOL;
CosechandoRW2:BOOL;

EP_CO2_RW2: UINT;
EP_CO2_RW2_value: UINT;

EV_AIRE_RW1_value: UINT;
EV_AIRE_RW2_value: UINT;
EV_CO2_RW1_value: UINT;
EV_CO2_RW2_value: UINT;

BombaCosechaRW1:BOOL;
BombaCosechaRW2:BOOL;
BombaCosechaRW1_F:BOOL;
BombaCosechaRW2_F:BOOL;
EVAIRERW1: BOOL;
EVAIRERW2: UINT;
EVC02RW1: UINT;
EVCosechaRW1:BOOL;
EVCosechaRW2:BOOL;
EVCosechaRW1_F:BOOL;
EVCosechaRW2_F:BOOL;
EVMedioRW1: BOOL;
EVMedioRW2: BOOL;
EVMedioRW1_F: BOOL;
EVMedioRW2_F: BOOL;

InicioCosechadorW1: BOOL;
InicioCosechadorW2: BOOL;

InyeccionCO2_RW1: BOOL;
InyeccionCO2_RW2: BOOL;

Manual_RW1: BOOL;
Manual_value_RW1: LREAL;

ReactivarEvapRW1:BOOL;
ReactivarEvapRW2:BOOL;
ReposicionEvapRW1: BOOL;
ReposicionEvapRW2: BOOL;

Selectorcontrol_RW1: BOOL;
Selectorcontrol_RW2: BOOL;
Switch_manual_CO2_RW2:BOOL;
hora_real:UINT;
Estado_optimizador_RW1:BOOL;
Estado_optimizador_RW2:BOOL;
VariableInicioCos1:BOOL;
VariableInicioCos2:BOOL;
Wacht_Dog:BOOL;

```

END_VAR

Figura 3.11. Lista de variables globales declaradas.

Como se puede observar en la Figura 3.11, para que sean fácilmente localizables las variables se han escrito por secciones, en función de a que pestaña del panel de control pertenecen y, al final del listado, se ha establecido una recopilación con aquellas que, a pesar de no ser utilizadas en el diseño del panel, están declaradas como variables configuradas en el Servidor OPC ya existente y deben conservarse.

Como la lista generada es muy extensa, resulta tedioso definir la función de cada variable de forma independiente, de modo que se describirán de forma conjunta y en función del tipo de dato que son:

En primer lugar, están las variables de tipo INT, UINT, REAL o LREAL. Todas ellas de tipo numérico, se emplean para mostrar datos referentes al estado del sistema y que son proporcionados por los sensores instalados en la planta. Como puede ser el Item “TempSuelo1” que se vincula con el sensor que mide la temperatura del suelo del Raceway 1. Para denotar los datos vinculados con setpoints que estipula el usuario, como el Item “LIM_SUP_NIVEL_RW1” que hace referencia al valor máximo que puede alcanzar el nivel del reactor 1 y que hace que una vez superado se active un LED de alarma en el panel. O para visualizar datos ligados a los diversos tratamientos que se realizarán a los reactores. Como al introducir un optimizador, que se produce un nuevo setpoint de nivel optimizado, Item “Setpoint_opt_nivel_RW1”, en el caso del reactor 1.

Las variables de tipo String están destinadas a la creación de un registro de incidencias. De esta manera diversos usuarios que interactúen con el panel pueden dejar constancia de modificaciones o revelaciones sobre el mismo que hayan sufrido mientras trabajaban con él y de los que quieran dejar constancia. Los Items “Nombre” y “Comentario” se encargan de esta misión.

Las variables de tipo TIME_OF_DAY (TOD) se emplean en esta categoría para establecer en qué instantes deben suceder determinadas tareas en los reactores. Por ejemplo, el Item “Hora_Dilucion_RW1” estipula a qué hora hay que aportar dilución al Raceway 1 cuando se trabaja con este en modo automático en su control de nivel.

Por último, las de tipo BOOL, en su mayoría se vinculan con botones, interruptores y lámparas. Con ellos se puede activar o desactivar acciones o funciones del sistema o mostrar el estado de este en cuestión de variables del tipo Todo/Nada. Un ejemplo de ello es el Item “BOTON_COSECHADO_RW1” que se encarga de activar el cosechado de producto del Raceway 1 cuando se trabaja en modo manual en su control de nivel y que además enciende en el panel de control una lámpara de confirmación de que se está realizando dicha acción.

3.5. POU Tiempo

A continuación, se creó una unidad de organización de programa, o POU, donde se congregan todas las operaciones ligadas a unidades de tiempo, fechas u horas. De esta manera al congregarse todas las operaciones del mismo tipo juntas son más sencillas de localizar para posibles modificaciones posteriores.

Para crear la unidad se sigue el mismo procedimiento que para la creación de la lista de variables globales, pero con la distinción de que en este caso se selecciona el objeto “POU” en última instancia.

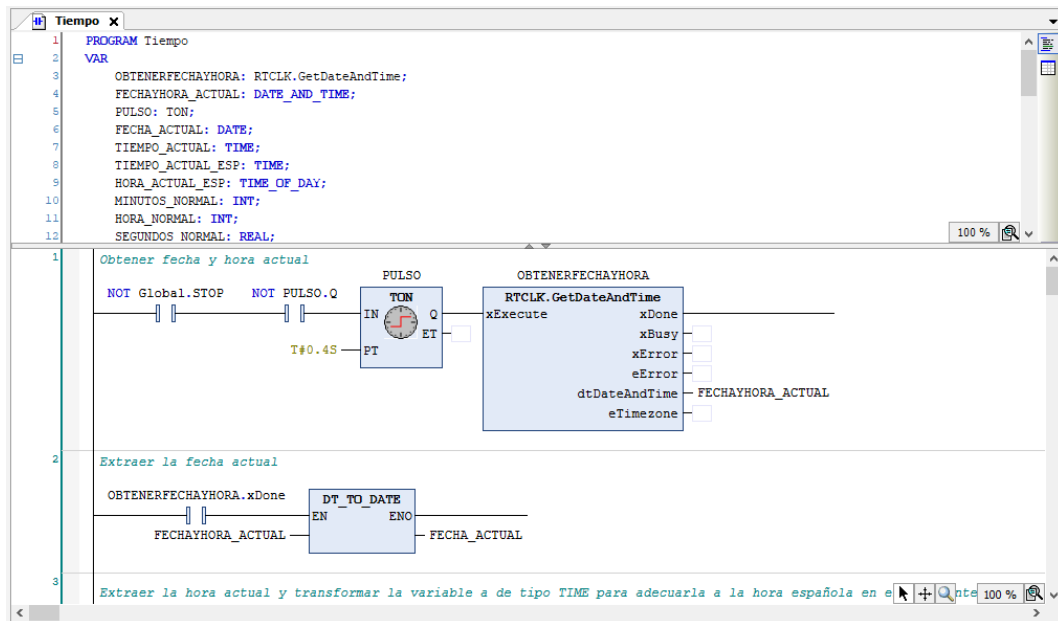


Figura 3.12. Ventana POU *Tiempo*.

Como se observa en la figura 3.12, la unidad se divide en dos subventanas. La superior está destinada a la declaración de variables locales, mientras que la inferior es donde se implementa el código del programa.

En tal caso, se ha declarado un listado de variables locales, algunas de ellas asociadas a funciones de la interfaz como son los temporizadores TON o el módulo RTCLK.GetDateAndTime, mientras que otras sirven para asignar los valores de salida de dichas funciones (Figura 3.13.a).

```
Tiempo x
1 PROGRAM Tiempo
2 VAR
3   OBTENERFECHAYHORA: RTCLK.GetDateAndTime;
4   FECHAYHORA_ACTUAL: DATE_AND_TIME;
5   PULSO: TON;
6   FECHA_ACTUAL: DATE;
7   TIEMPO_ACTUAL: TIME;
8   TIEMPO_ACTUAL_ESP: TIME;
9   HORA_ACTUAL_ESP: TIME_OF_DAY;
10  MINUTOS_NORMAL: INT;
11  HORA_NORMAL: INT;
12  SEGUNDOS_NORMAL: REAL;
13  TIEMPO_HSEG: DINT;
14  TIEMPO_MSEG: DINT;
15  TIME_IN_SEC: LREAL;
16  DIA_ACTUAL: INT;
17  MES_ACTUAL: INT;
18  AÑO_ACTUAL: INT;
19  HoraMinutos: TIME_OF_DAY;
20  TIEMPO_MHORA: INT;
21  TIEMPO_SHORA: REAL;
22  HORA_DECIMAL: REAL;
23  TON_1: TON;
24  TON_2: TON;
25  TON_3: TON;
26  TON_4: TON;
27  hora_dil_RW1:INT;
28  min_dil_RW1:INT;
29  hora_cos_RW1:INT;
30  min_cos_RW1:INT;
31  hora_dil_RW2:INT;
32  min_dil_RW2:INT;
33  hora_cos_RW2:INT;
34  min_cos_RW2:INT;
35  FECHA_STRING: STRING;
36  HORA_STRING: STRING;
37  FECHAYHORA_STRING: STRING;
```

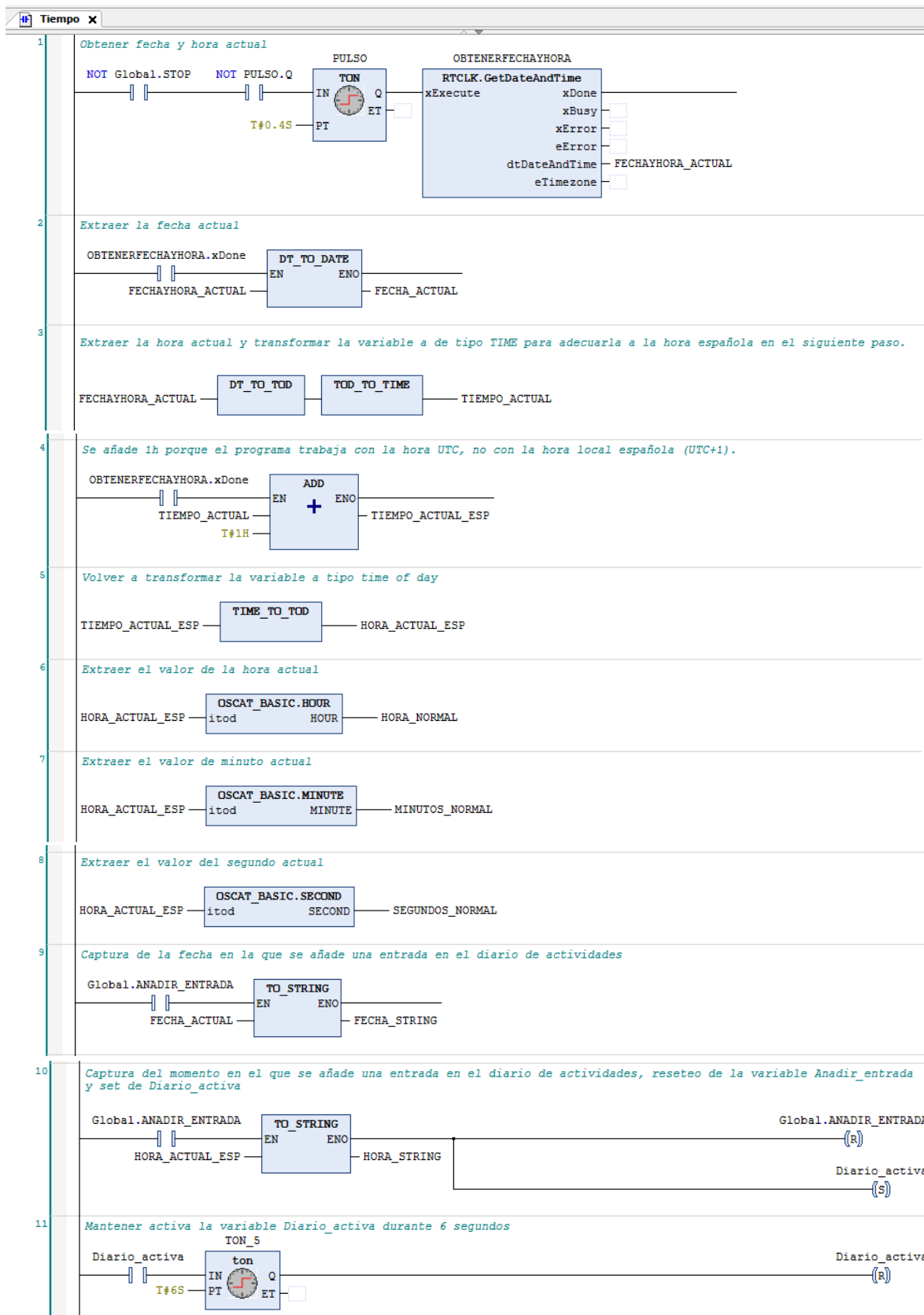


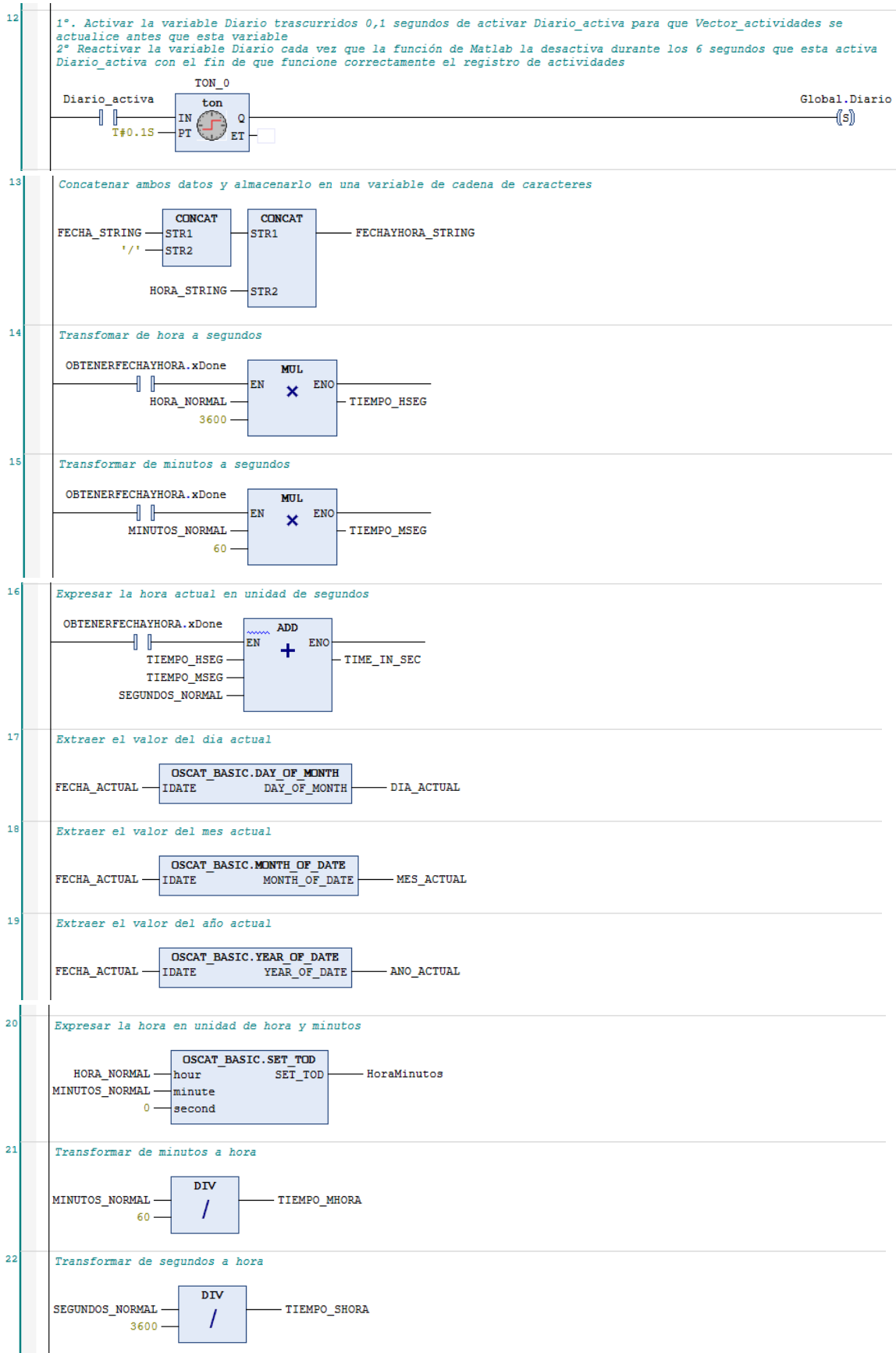
```

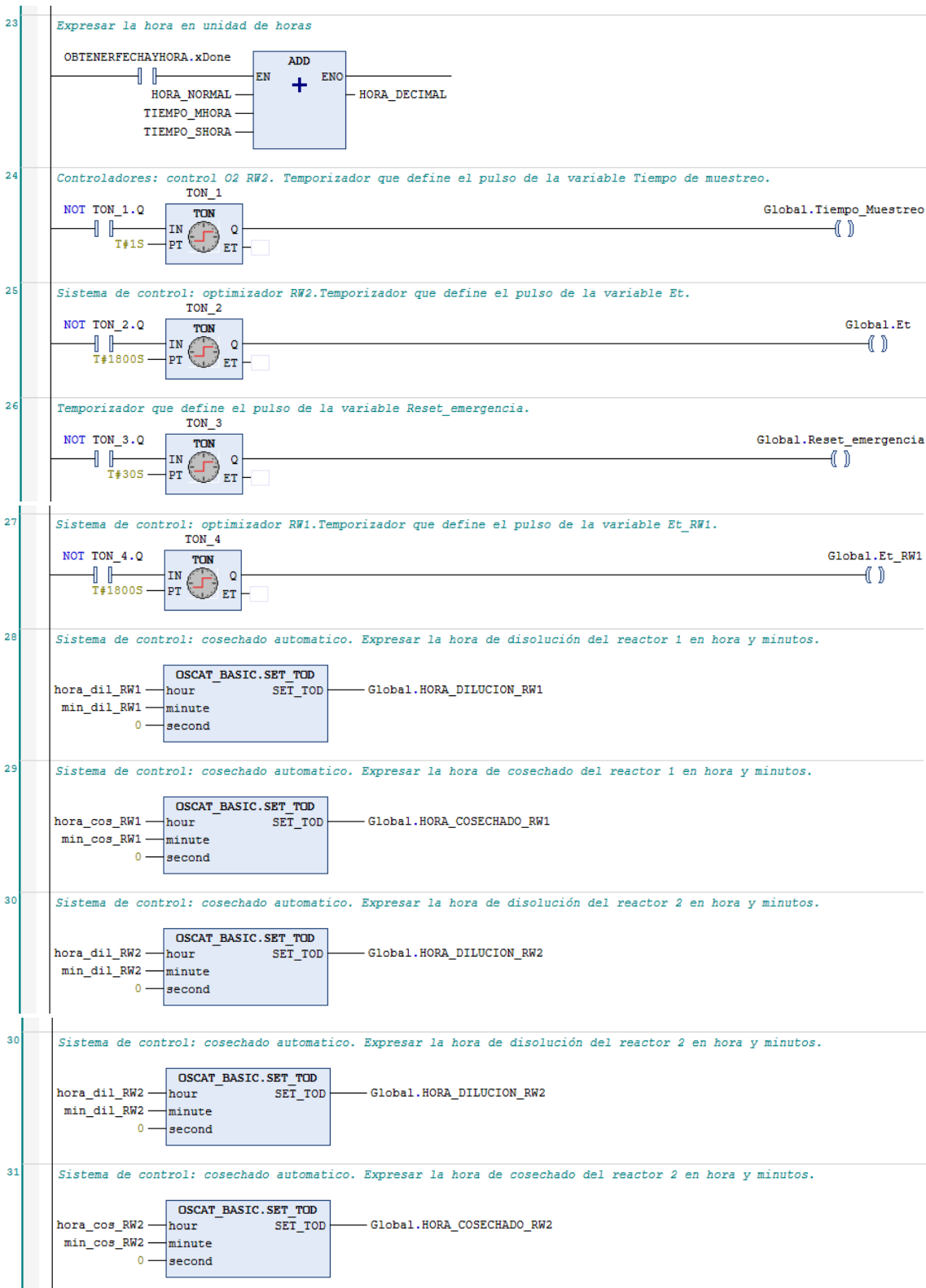
38     TON_0: ton;
39     Diario_activa: BOOL;
40     TON_5: ton;
41     END_VAR

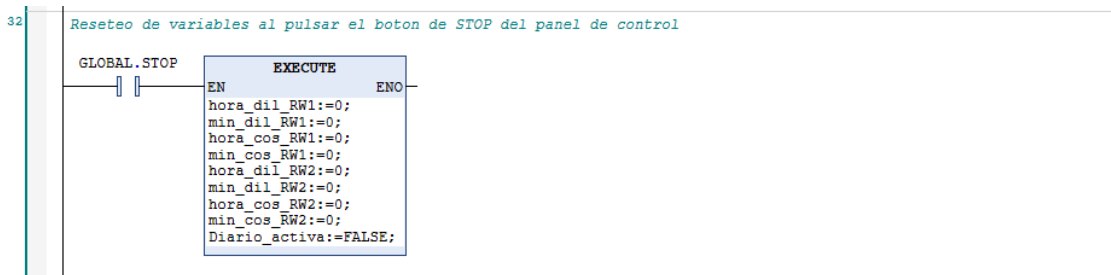
```

a) Variables locales de la unidad.









b) Código de la unidad.

Figura 3.13. POU *Tiempo*.

Como se aprecia en la Figura 3.13.b, se empleó el lenguaje de programación LD, por su sencillez y su estructuración de la información. Cada línea de código, en la parte superior, cuenta con un comentario de la función que esta cumple.

De forma resumida se puede decir que las primeras líneas de código se destinan a determinar la fecha y hora instantáneas y su fragmentación en sus unidades constituyentes (día, mes, año, horas, minutos y segundos). Las siguientes líneas se encargan de expresar el tiempo instantáneo en unidades de segundos, de horas o de horas y minutos. Luego se realiza la captura del instante en el que se introduce una entrada en el diario de actividades.

Más abajo se hace el tratamiento de ciertas variables globales para que tengan una salida en señales de pulso. Este tratamiento se realiza porque dichas variables están vinculadas a scripts de Matlab y si no se envían de esta manera el programa no ejecuta correctamente el código. Las horas de disolución y cosechado de los reactores, en el cosechado automático, han de ser en unidades de horas y minutos y los datos TOD alcanzan hasta las milésimas de segundo por lo que hay que forzar que todas las unidades por debajo del minuto sean igual a cero. Este ajuste es lo que se refleja en las últimas líneas de código de la unidad, junto con un reseteo de variables del que se hablará más adelante.

3.6. POU *PLC_PRG*

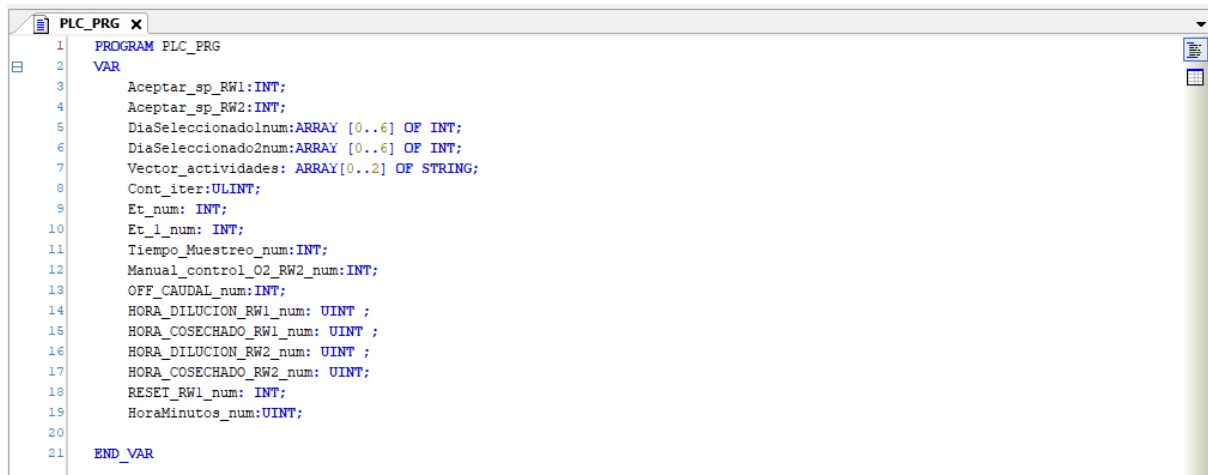
Esta unidad de programa es la que genera por defecto Codesys cuando se crea el proyecto como “Proyecto estándar” y no tiene ninguna diferencia con respecto al resto de POU que se añaden posteriormente a este.

En esta unidad se llevan a cabo aquellas operaciones necesarias para adaptar las variables, que se transmiten al resto de programas a través del servidor OPC, a dichos entornos de programación. El porqué de estos cambios se debe a que la comunicación actual entre dispositivos es: PLCs con LabVIEW a través del servidor OPC y una comunicación interna de LabVIEW con Matlab. Este tipo de comunicación interna no es posible realizarse en el entorno de desarrollo de Codesys por lo que la solución planteada es que la comunicación con Matlab sea externa, es resumidas cuentas, que sea a través del servidor.

Como ocurría en la POU “Tiempo”, donde se transformaba un valor de muestreo a una señal de pulsos de periodo el valor de muestreo, en esta unidad se hacen otras transformaciones para adecuar algunos datos al entorno de programación de Matlab. Por ejemplo, las señales tipo BOOL, que se traducen en señales biestado (TRUE/FALSE) han de transformarse a su

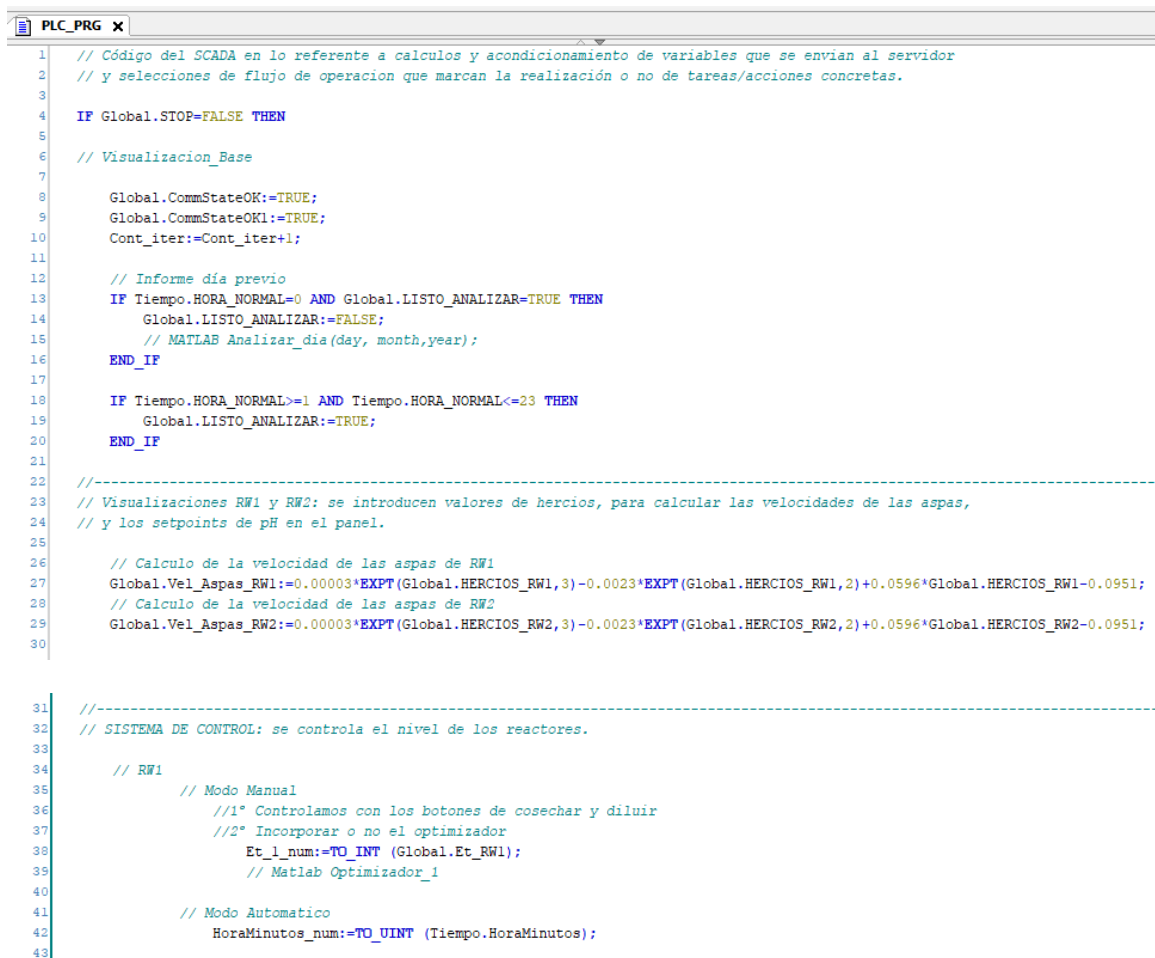
equivalente numérico, es decir, 1/0, respetivamente. También en esta POU se llevan a cabo otras acciones como el cálculo de la velocidad de las aspas de cada reactor.

En este caso, como se indica en la siguiente figura, se ha optado por el lenguaje de programación ST, dada la longitud del código que se pretendía implementar, y se ha estructurado por secciones como la lista de variables globales (ver Figura 3.14).



```
1 PROGRAM PLC_PRG
2 VAR
3     Aceptar_sp_RW1:INT;
4     Aceptar_sp_RW2:INT;
5     DiaSeleccionado1num:ARRAY [0..6] OF INT;
6     DiaSeleccionado2num:ARRAY [0..6] OF INT;
7     Vector_actividades: ARRAY[0..2] OF STRING;
8     Cont_iter:ULINT;
9     Et_num: INT;
10    Et_1_num: INT;
11    Tiempo_Muestreo_num:INT;
12    Manual_control_O2_RW2_num:INT;
13    OFF_CAUDAL_num:INT;
14    HORA_DILUCION_RW1_num: UINT ;
15    HORA_COSECHADO_RW1_num: UINT ;
16    HORA_DILUCION_RW2_num: UINT ;
17    HORA_COSECHADO_RW2_num: UINT;
18    RESET_RW1_num: INT;
19    HoraMinutos_num:UINT;
20
21 END_VAR
```

a) Variables locales de la unidad.



```
1 // Código del SCADA en lo referente a calculos y acondicionamiento de variables que se envían al servidor
2 // y selecciones de flujo de operación que marcan la realización o no de tareas/acciones concretas.
3
4 IF Global.STOP=FALSE THEN
5
6     // Visualizacion_Base
7
8     Global.CommStateOK:=TRUE;
9     Global.CommStateOK1:=TRUE;
10    Cont_iter:=Cont_iter+1;
11
12    // Informe día previo
13    IF Tiempo.HORA_NORMAL=0 AND Global.LISTO_ANALIZAR=TRUE THEN
14        Global.LISTO_ANALIZAR:=FALSE;
15        // MATLAB Analizar_dia(day, month,year);
16    END_IF
17
18    IF Tiempo.HORA_NORMAL>=1 AND Tiempo.HORA_NORMAL<=23 THEN
19        Global.LISTO_ANALIZAR:=TRUE;
20    END_IF
21
22    //-----
23    // Visualizaciones RW1 y RW2: se introducen valores de hercios, para calcular las velocidades de las aspas,
24    // y los setpoints de pH en el panel.
25
26    // Calculo de la velocidad de las aspas de RW1
27    Global.Vel_Aspas_RW1:=0.00003*EXPT(Global.HERCIOS_RW1,3)-0.0023*EXPT(Global.HERCIOS_RW1,2)+0.0596*Global.HERCIOS_RW1-0.0951;
28    // Calculo de la velocidad de las aspas de RW2
29    Global.Vel_Aspas_RW2:=0.00003*EXPT(Global.HERCIOS_RW2,3)-0.0023*EXPT(Global.HERCIOS_RW2,2)+0.0596*Global.HERCIOS_RW2-0.0951;
30
31
32    //-----
33    // SISTEMA DE CONTROL: se controla el nivel de los reactores.
34
35    // RW1
36    // Modo Manual
37    //1° Controlamos con los botones de cosechar y diluir
38    //2° Incorporar o no el optimizador
39    Et_1_num:=TO_INT (Global.Et_RW1);
40    // Matlab Optimizador_1
41
42    // Modo Automatico
43    HoraMinutos_num:=TO_UINT (Tiempo.HoraMinutos);
```

Desarrollo de una herramienta SCADA en Codesys para fotobiorreactores industriales

```
44 IF Global.AUTOMATICO_CONTROL_RW1=TRUE THEN
45     Global.BOTON_COSECHADO_RW1:=FALSE;
46     Global.BOTON_DILUIR_RW1:=FALSE;
47 END_IF
48
49     Aceptar_sp_RW1:=TO_INT(Global.VERIF_RW1);
50     HORA_DILUCION_RW1_num:=TO_UINT (Global.HORA_DILUCION_RW1) ;
51     HORA_COSECHADO_RW1_num:= TO_UINT (Global.HORA_COSECHADO_RW1) ;
52     DiaSeleccionadoInum[1]:=TO_INT(Global.DiaSeleccionado1[1]);
53     DiaSeleccionadoInum[2]:=TO_INT(Global.DiaSeleccionado1[2]);
54     DiaSeleccionadoInum[3]:=TO_INT(Global.DiaSeleccionado1[3]);
55     DiaSeleccionadoInum[4]:=TO_INT(Global.DiaSeleccionado1[4]);
56     DiaSeleccionadoInum[5]:=TO_INT(Global.DiaSeleccionado1[5]);
57     DiaSeleccionadoInum[6]:=TO_INT(Global.DiaSeleccionado1[6]);
58     DiaSeleccionadoInum[0]:=TO_INT(Global.DiaSeleccionado1[0]);
59     // Matlab Cosechado Automatico 1
60
61     ..
62     IF Global.VERIF_RW1=TRUE THEN
63         Global.Setpoint_nivel_RW1:=Global.Setpoint_nivel_RW1_manual;
64     ELSE
65         Global.Setpoint_nivel_RW1:=Global.Setpoint_nivel_RW1;
66     END_IF
67
68     // Cosechado y dilución (lámparas de Visualizacion RW1 y esquema de la instalación de Sistema de control)
69
70     IF Global.CONTROL_NivelRW1=TRUE AND (Global.BOTON_COSECHADO_RW1=TRUE OR Global.EVCosechaRW1=TRUE) THEN
71         Global.COSECHADO_RW1:=TRUE;
72     ELSE
73         Global.COSECHADO_RW1:=FALSE;
74     END_IF
75     IF Global.CONTROL_NivelRW1=TRUE AND (Global.BOTON_DILUIR_RW1=TRUE OR Global.EVMedioRW1=TRUE) THEN
76         Global.DILUCION_RW1:=TRUE;
77     ELSE
78         Global.DILUCION_RW1:=FALSE;
79     END_IF
80
81 // RW2
82 // Manual
83 //1° Controlamos con los botones de cosechar y diluir
84 // 2° Incorporar o no el optimizador
85 Et_num:=TO_INT (Global.Et);
86 // Matlab Optimizador 2
87
88 //Automatico
89 IF Global.AUTOMATICO_CONTROL_RW2=TRUE THEN
90     Global.BOTON_COSECHADO_RW2:=FALSE;
91     Global.BOTON_DILUIR_RW2:=FALSE;
92 END_IF
93
94     Aceptar_sp_RW2:=TO_INT(Global.VERIF_RW2);
95     HORA_DILUCION_RW2_num:=TO_UINT (Global.HORA_DILUCION_RW2) ;
96     HORA_COSECHADO_RW2_num:= TO_UINT (Global.HORA_COSECHADO_RW2) ;
97     DiaSeleccionado2num[1]:=TO_INT(Global.DiaSeleccionado2[1]);
98     DiaSeleccionado2num[2]:=TO_INT(Global.DiaSeleccionado2[2]);
99     DiaSeleccionado2num[3]:=TO_INT(Global.DiaSeleccionado2[3]);
100    DiaSeleccionado2num[4]:=TO_INT(Global.DiaSeleccionado2[4]);
101    DiaSeleccionado2num[5]:=TO_INT(Global.DiaSeleccionado2[5]);
102    DiaSeleccionado2num[6]:=TO_INT(Global.DiaSeleccionado2[6]);
103    DiaSeleccionado2num[0]:=TO_INT(Global.DiaSeleccionado2[0]);
104    // Matlab Cosechado Automatico 2
105
106    IF Global.VERIF_RW2=TRUE THEN
107        Global.Setpoint_nivel_RW2:=Global.Setpoint_nivel_RW2_manual;
108    ELSE
109        Global.Setpoint_nivel_RW2:=Global.Setpoint_nivel_RW2;
110    END_IF
111
112    // Cosechado y dilución (lámparas de Visualizacion RW2 y esquema de la instalación de Sistema de control)
113
114    IF Global.CONTROL_NivelRW2=TRUE AND (Global.BOTON_COSECHADO_RW2=TRUE OR Global.EVCosechaRW2=TRUE) THEN
115        Global.COSECHADO_RW2:=TRUE;
116    ELSE
117        Global.COSECHADO_RW2:=FALSE;
118    END_IF
119    IF Global.CONTROL_NivelRW2=TRUE AND (Global.BOTON_DILUIR_RW2=TRUE OR Global.EVMedioRW2=TRUE) THEN
120        Global.DILUCION_RW2:=TRUE;
121    ELSE
122        Global.DILUCION_RW2:=FALSE;
123    END_IF
124
125 //-----
126 // Controladores: se controlan los valores de O2 y CO2 de los reactores.
```

```
127 //RW1
128 // CONTROL DE CO2 principal
129 RESET_RW1_num:=TO_INT(Global.RESET_RW1);
130 // Matlab Control_pH
131
132 // CONTROL DE O2 principal
133 // introducir aqui
134
135 // Controladores adicionales de los usuarios (Pruebas)
136 // introducir aqui
137
138 //RW2
139
140 // CONTROL DE CO2 principal
141 // introducir aqui
142
143 // CONTROL DE O2 principal
144
145
146 Tiempo_Muestreo_num:=TO_INT (Global.Tiempo_Muestreo);
147 Manual_control_O2_RW2_num:= TO_INT(Global.Manual_control_O2_RW2);
148 OFF_CAUDAL_num:=TO_INT(Global.OFF_CAUDAL);
149 // Matlab Control_O2
150
151 // Cortar inyeccion de O2 cuando se inyecta CO2
152 IF Global.CO2RW2=0 AND Global.ODRW21>120 THEN
153 Global.EVAIRERW2:=Global.upid_a*TO_INT(TRUE);
154 END_IF
155
156 // Controladores adicionales de los usuarios (Pruebas)
157 // introducir aqui
158
159 -----
160 // Graficas: representacion de las variables seleccionadas graficamente.
161 -----
162
163 // Diario de actividades
164
165 // Variables Nombre y Comentario se almacenan en un vector junto con la fecha y hora en que se añade la entrada.
166 Vector_actividades[0]:=Tiempo.FECHAYHORA_STRING;
167 Vector_actividades[1]:=Global.Nombre;
168 Vector_actividades[2]:=Global.Comentario;
169
170 ELSE
171 // Reseteo del contador de iteraciones
172 cont_iter:=0;
173 END_IF
```

b) Código de la unidad.

Figura 3.14. POU *PLC_PRG*.

Este código se ha desarrollado atendiendo al esquema mostrado en la Figura 3.1. En el cual, para cada raceway, se trataban los datos de un proceso de cosechado automático y de un optimizador. Sin embargo, para las funciones de control de CO₂ (o pH) y de O₂ solo se realiza: para el primer caso, con el Raceway 1 y para el segundo, con el Raceway 2. Aun así, en la lista Global, se declararon las variables para ambos reactores, para futuras implementaciones que se produjesen.

Además de la adaptación de datos, en esta unidad se introducen elementos como selectores de flujo de ejecución que gestionan la realización o no de ciertas acciones. Por ejemplo, para que se genere el registro del día anterior se debe cumplir que la hora actual del sistema sea igual a cero y que la variable “LISTO_ANALIZAR” esté a False (Figura 3.14.b).

3.7. POU *Reseteo*

Como ocurre con cualquier sistema real, existen situaciones de emergencia en las que es necesario detener completamente la ejecución de los procesos. Con ese fin se ha creado una

unidad de programa llamada “Reseteo” que se activa únicamente cuando se pulsa el botón de STOP, localizado en la parte inferior derecha del panel de control.

Esta variable es responsable de parar la ejecución del script de Matlab; de detener las acciones de la POU “PLC_PRG” y resetear su variable de conteo de iteraciones, cont_iter; de pausar las funciones de la unidad “Tiempo”, a excepción de las señales de pulso que no son necesarias porque ya en la unidad PLC_PRG se detienen dichas señales que se envían al servidor; y, por último, de poner, través de “Reseteo” y “Tiempo”, todas las variables del sistema controladas por el usuario a 0, en el caso de las de tipo numérico, a False, las de tipo Booleano, y de eliminar el comentario, en las de tipo String. Solo la propia variable STOP y LISTO_ANALIZAR quedan exentas de este reinicio.

Para esta unidad no es necesario la declaración de variables locales como se puede ver en la Figura 3.15.

```
Reseteo x
1 // Visualizacion base
2 Global.CommStateOK:=FALSE;
3 Global.CommStateOK1:=FALSE;
4
5 // -----
6 // Visualizacion RW1
7 // Global.TempSuelo1:=0;
8 // Global.TempRW1:=0;
9 // Global.NivelRW1:=0;
10 Global.pH_RW1:=0;
11 // Global.pHRW11:=0;
12 // Global.pHRW12:=0;
13 // Global.pHRW13:=0;
14 // Global.ODRW11:=0;
15 // Global.ODRW12:=0;
16 // Global.ODRW13:=0;
17 Global.HERCIOS_RW1:=0;
18 // Global.Vel_Aspas_RW1:=0;
19 // Global.AireRW1:=0;
20 // Global.CO2RW1:=0;
21 Global.COSECHADO_RW1:=FALSE;
22 Global.DILUCION_RW1:=FALSE;
23 // -----
24 // Visualizacion RW2
25 // Global.TempSuelo2:=0;
26 // Global.TempRW2:=0;
27 // Global.NivelRW2:=0;
28 Global.pH_RW2:=0;
29 // Global.pHRW21:=0;
30 // Global.pHRW22:=0;
31 // Global.pHRW23:=0;
32 // Global.ODRW21:=0;
33
34 // Global.ODRW22:=0;
35 // Global.ODRW23:=0;
36 Global.HERCIOS_RW2:=0;
37 // Global.Vel_Aspas_RW2:=0;
38 // Global.AireRW2:=0;
39 // Global.CO2RW2:=0;
40 Global.COSECHADO_RW2:=FALSE;
41 Global.DILUCION_RW2:=FALSE;
42 // Global.Turbidimetro:=0;
43 // Global.Conductimetro:=0;
44 // -----
45 // Sistema de control RW1
46 Global.CONTROL_NivelRW1:=FALSE;
47 Global.AUTOMATICO_CONTROL_RW1:=FALSE;
48 Global.Setpoint_nivel_RW1:=0;
49 // Global.Setpoint_nivel_RW1_manual:=0;
50 Global.VERIF_RW1:=FALSE;
51 Global.Optimizador_RW1:=FALSE;
52 Global.BOTON_COSECHADO_RW1:=FALSE;
53 Global.BOTON_DILUIR_RW1:=FALSE;
54
55 // CON OPTIMIZADOR
56 // Global.Setpoint_opt_nivel_RW1:=0;
57 // Global.HighLim_RW1:=0;
58 // Global.LowLim_RW1:=0;
59 // Global.cnt:=0;
60 // AUTOMATICO
61 Global.DiaSeleccionadol[1]:=FALSE ;
135 // Global.Set_CaudalCO2_RW2_ESC:=0;
136
137 // CONTROL OD
138 Global.Manual_control_O2_RW2:=FALSE;
139 Global.ENTRADA_VALVULA_O2_RW2:=0;
140 // AUTO
141 Global.Set_OD_RW2:=0;
142 // Global.Q_Set_OD_RW2:=0;
143 // Global.kia:=0;
144 // Global.upid_a:=0;
145 // Global.Error_OD_RW2:=0;
146 // Global.Tiempo_Muestreo:=FALSE;
147 Global.OFF_CAUDAL:=FALSE;
148 // -----
149 // Diario
150 Global.ANADIR_ENTRADA:=FALSE;
151 Global.Nombre:='';
152 Global.Comentario:='';
153 Global.Diario:=FALSE;
154 // -----
155 // Alarmas
156 // -----
157 // Control de alarmas
158 Global.LIM_SUP_HP_RW1:=0;
159 Global.LIM_INF_HP_RW1:=0;
160 Global.LIM_SUP_OD_RW1:=0;
161 Global.LIM_INF_OD_RW1:=0;
162 Global.LIM_SUP_NIVEL_RW1:=0;
163 Global.LIM_INF_NIVEL_RW1:=0;
164 Global.LIM_SUP_HP_RW2:=0;
165 Global.LIM_INF_HP_RW2:=0;
166 Global.LIM_SUP_OD_RW2:=0;
167 Global.LIM_INF_OD_RW2:=0;
168 Global.LIM_SUP_NIVEL_RW2:=0;
169 Global.LIM_INF_NIVEL_RW2:=0;
170 // -----
171 // Variables atmosfericas
172 // Global.Temp_atm:=0;
173 // Global.Hum_rel_atm:=0;
174 // Global.Rad_atm:=0;
175 // Global.Rad_PAR_atm:=0;
176 // Global.Vel_viento_atm:=0;
177 // Global.Dir_viento_atm:=0;
178 // -----
179 // RESET, STOP, EMERGENCIA
180 // Global.Reset_emergencia:=FALSE;
181 //
182 // Otras variables
183 // Global.PID_Fd_RW1:= 0;
184 // Global.PID_Kp_RW1:= 0;
185 // Global.PID_RW1_Output:= 0;
186 // Global.PID_Td_RW1:= 0;
187 // Global.PID_Ti_RW1:= 0;
188 // Global.PID_RW1_PWM_Input:= 0;
189 //
190 // Global.Activar_EVMedioRW1:= FALSE;
```



```

61 Global.DiaSeleccionadol[2]:=FALSE;
62 Global.DiaSeleccionadol[3]:=FALSE;
63 Global.DiaSeleccionadol[4]:=FALSE;
64 Global.DiaSeleccionadol[5]:=FALSE;
65 Global.DiaSeleccionadol[6]:=FALSE;
66 Global.DiaSeleccionadol[0]:=FALSE;
67 Global.NIVEL_DILUACION_RW1:=0;
68 Global.NIVEL_COSECHADO_RW1:=0;
69
70 //-----
71 // Sistema de control RW2
72 Global.CONTROL_NivelRW2:=FALSE;
73 Global.Optimizador_RW2:=FALSE;
74 Global.Setpoint_nivel_RW2:=0;
75 // Global.Setpoint_nivel_RW2_manual:=0;
76 Global.VERIF_RW2:=FALSE;
77 Global.AUTOMATICO CONTROL RW2:=FALSE;
78
79 // AUTOMATICO
80 Global.DiaSeleccionado2[1]:=FALSE;
81 Global.DiaSeleccionado2[2]:=FALSE;
82 Global.DiaSeleccionado2[3]:=FALSE;
83 Global.DiaSeleccionado2[4]:=FALSE;
84 Global.DiaSeleccionado2[5]:=FALSE;
85 Global.DiaSeleccionado2[6]:=FALSE;
86 Global.DiaSeleccionado2[0]:=FALSE;
87 Global.NIVEL_DILUACION_RW2:=0;
88
89 Global.NIVEL_COSECHADO_RW2:=0;
90
91 //-----
92 // Controladores RW1
93 // CONTROL CO2
94 Global.Todo_Nada_CO2_RW1:=FALSE;
95 Global.Manual_control_CO2_RW1:=FALSE;
96 Global.Set_CaudalCO2_RW1:=0;
97 Global.Pruebas_1:=FALSE;
98 // AUTO
99 // Global.Set_CaudalCO2_RW1_ESC:=0;
100 // Global.ITERACION_RW1:=0;
101 // Global.pH_filtrado_RW1:=0;
102 // Global.RESET_RW1:=FALSE;
103 // Global.Aux1:=0;
104 // Global.Aux2:=0;
105 // Global.Aux3:=0;
106 // Global.Aux4:=0;
107
108 // CONTROL OD
109 Global.Manual_control_O2_RW1:=FALSE;
110 // AUTO
111 Global.Set_OD_RW1:=0;
112 // Global.Q_Set_OD_RW1:=0;
113 // Global.kia1:=0;
114 // Global.upid_1:=0;
115 // Global.Error_OD_RW1:=0;
116
117 Global.Boton_Aire_RW1:=FALSE;
118 //-----
119 // Controladores RW2
120
121 // CONTROL CO2
122 Global.Todo_Nada_CO2_RW2:=FALSE;
123 Global.Manual_control_CO2_RW2:=FALSE;
124 Global.Set_CaudalCO2_RW2:=0;
125 Global.Pruebas_2:=FALSE;
126 // AUTO
127 // Global.ITERACION_RW2:=0;
128 // Global.pH_filtrado_RW2:=0;
129 // Global.RESET_RW2:=FALSE;
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191 //
192 // Global.Bajar_nivel_RW1:= FALSE;
193 // Global.Bajar_nivel_RW2:= FALSE;
194 // Global.Subir_nivel_RW1:= FALSE;
195 // Global.Subir_nivel_RW2:= FALSE;
196 //
197 // Global.Repo_EVMedioRW1:= FALSE;
198 // Global.Repo_EVMedioRW2:= FALSE;
199 //
200 // Global.Control_diurno_RW2_value:= 0;
201 //
202 // Global.ControlpHRW1:= FALSE;
203 // Global.ControlpHRW2:= FALSE;
204 //
205 // Global.CosechandoRW1:=FALSE;
206 // Global.CosechandoRW2:=FALSE;
207 //
208 // Global.EP_CO2_RW2:= 0;
209 // Global.EP_CO2_RW2_value:= 0;
210 //
211 // Global.EV_AIRE_RW1_value:= 0;
212 // Global.EV_AIRE_RW2_value:= 0;
213 // Global.EV_CO2_RW1_value:= 0;
214 // Global.EV_CO2_RW2_value:= 0;
215 //
216 // Global.BombaCosechaRW1:=FALSE;
217 // Global.BombaCosechaRW2:=FALSE;
218 // Global.BombaCosechaRW1_F:=FALSE;
219 // Global.BombaCosechaRW2_F:=FALSE;
220 // Global.EVAIRERW1:= FALSE;
221 // Global.EVAIRERW2:= 0;
222 // Global.EVCO2RW1:= 0;
223 // Global.EVCosechaRW1:=FALSE;
224 // Global.EVCosechaRW2:=FALSE;
225 // Global.EVCosechaRW1_F:=FALSE;
226 // Global.EVCosechaRW2_F:=FALSE;
227 // Global.EVMedioRW1:= FALSE;
228 // Global.EVMedioRW2:= FALSE;
229 // Global.EVMedioRW1_F:= FALSE;
230 // Global.EVMedioRW2_F:= FALSE;
231 //
232 // Global.InicioCosechadoRW1:= FALSE;
233 // Global.InicioCosechadoRW2:= FALSE;
234 //
235 // Global.InyeccionCO2_RW1:= FALSE;
236 // Global.InyeccionCO2_RW2:= FALSE;
237 //
238 // Global.Manual_RW1:= FALSE;
239 // Global.Manual_value_RW1:= 0;
240 //
241 // Global.ReactivarEvapRW1:=FALSE;
242 // Global.ReactivarEvapRW2:=FALSE;
243 // Global.ReposicionEvapRW1:= FALSE;
244 // Global.ReposicionEvapRW2:= FALSE;
245 //
246 // Global.Selectorcontrol_RW1:= FALSE;
247 // Global.Selectorcontrol_RW2:= FALSE;
248 // Global.Switch_manual_CO2_RW2:=FALSE;
249 // Global.hora_real:=0;
250 //
251 // Global.Estado_optimizador_RW1:=FALSE;
252 // Global.Estado_optimizador_RW2:=FALSE;
253 //
254 // Global.VariableInicioCos1:=FALSE;
255 // Global.VariableInicioCos2:=FALSE;
256 // Global.Wacht_Dog:=FALSE;

```

Figura 3.15. POU *Reseteo*.

De esta manera mientras el botón STOP esté activo los datos que se envían al servidor son los que quedaron registrado en cada programa, Matlab y Codesys, justo antes de ser este pulsado.

Al hacer clic de nuevo sobre él y, por lo tanto, desactivarlo, las variables correspondientes pasan a tener el nuevo valor asignado tras el reseteo y se envían al servidor. De esta manera se

suspende el intercambio de nuevos valores por el servidor mientras se esté en modo STOP y se borra el rastro de las actividades del usuario anteriormente realizadas sobre el panel.

3.8. Visualizaciones

Una vez llegado a este punto, es momento de construir el panel de control. Esta es la parte más extensa y visual del proyecto y por ello se explicará su contenido por subapartados:

3.8.1. Entorno de visualización

Para diseñar un panel de control con el que el usuario pueda interactuar en Codesys se hace uso de objetos “Visualización” (Application→ Agregar objeto → Visualización). Una vez se crea, aparecerá una ventana en blanco, que será nuestro lienzo. En ella se incorporan los elementos que darán vida al panel. Dichos elementos están disponibles en la sección derecha de la pantalla, en el “Cuadro de herramientas de Visualización”.

En la parte superior de la ventana de visualización (Figura 3.16) hay un apartado donde se pueden definir variables locales (al igual que ocurría en las POU), un listado con los elementos añadidos a la visualización junto con algunas características sobre ellos como su posición y, por último, un configurador de teclado.

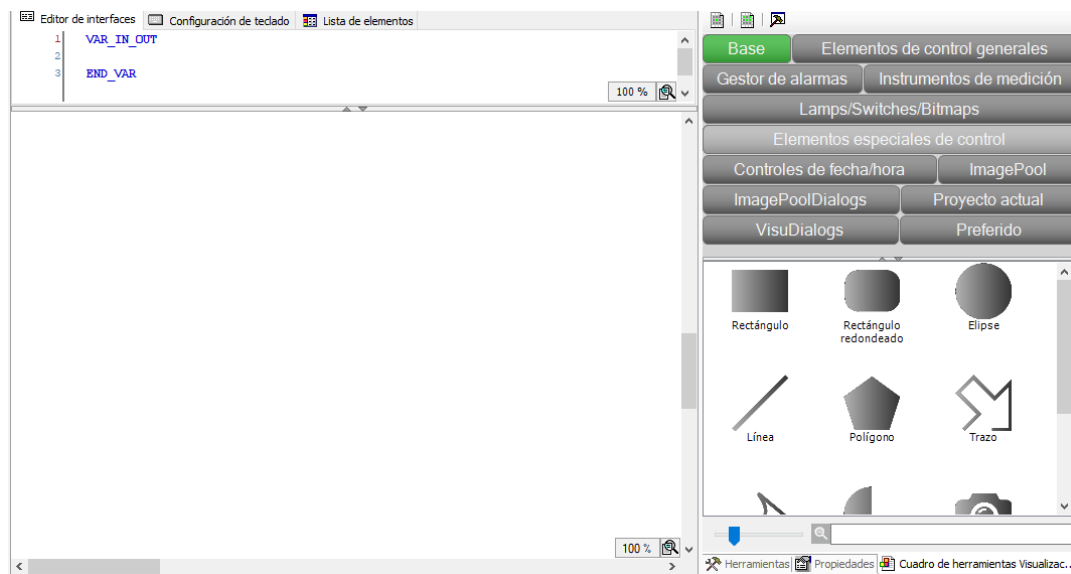


Figura 3.16. Vista de una Ventana de Visualización.

3.8.2. Elementos de visualización

Dentro del cuadro de herramientas anteriormente mencionado, existen diversas categorías en las que se clasifican los elementos disponibles para la visualización. De entre todos los elementos disponibles en este proyecto se hizo uso de los expuestos en la Figura 3.17.



Figura 3.17. Recopilación de elementos empleados en Visualización.

Para elementos como Rectángulo, Línea o Lámpara no es necesario explicar que utilidad tendrán dentro de la visualización por su obviedad, pero para otros se hará una breve aclaración de porque se han introducido.

El elemento “Tendencia” permite crear una representación gráfica, en función del tiempo, de las variables que se elijan de entre las declaradas en el proyecto. “Marco”, por otro lado, es un elemento que permite crear un conjunto de pestañas desplegadas. Para ello se crean visualizaciones independientes con el contenido que se pretende que aparezca dentro del cuadro y luego en “Configuración de visualizaciones referidas” se asocian dichas visualizaciones de modo que al pulsar un botón u otro aparezca una visualización u otra dentro del marco. Estos procesos se explicarán en mayor profundidad más adelante.

3.8.3. ImagenPool

A través de elementos Imagen se pueden introducir figuras contenidas en el proyecto en las visualizaciones. Con ese objetivo se crea primero un objeto “ImagePool” en la aplicación y desde él se añaden imágenes del computador al proyecto.

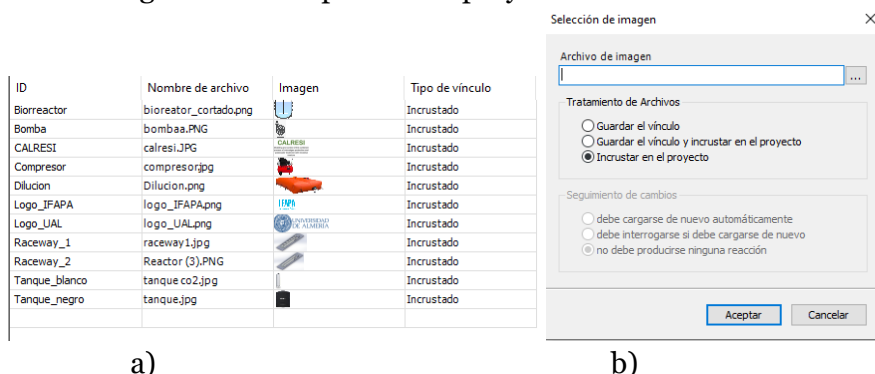


Figura 3.18. Objeto *ImagePool*. a) Ventana *ImagePool*, b) Selección de imagen.

Haciendo clic en Nombre de archivo (Figura 3.18.a) y luego en la parte derecha del recuadro, donde aparece un bloque con tres puntos, se agrega la imagen que se desea de entre las almacenadas en el ordenador y de qué modo (Figura 3.18.b). En este proyecto se optó por incrustar las figuras porque de este modo se asegura de que las imágenes no desaparezcan en caso de modificar su ubicación en el pc o si se traslada el proyecto a otro ordenador distinto.

Otro modo de incorporar dibujos a la visualización es directamente a través del desplegable “ImagePool” del Cuadro de Herramientas de Visualización, donde están disponibles todas las figuras añadidas al proyecto.

3.8.4. Propiedades del elemento

Cuando se incorpora un elemento en la ventana vacía de visualización, en la sección derecha de la pantalla aparece una pestaña de propiedades donde se pueden configurar las características del objeto como son su posición, color, movimiento, etc. Algunas de estas propiedades son comunes a todos ellos, como las recién citadas, mientras que en otras difieren de un elemento a otro. La Figura 3.19 expone un ejemplo de esta pestaña de propiedades.

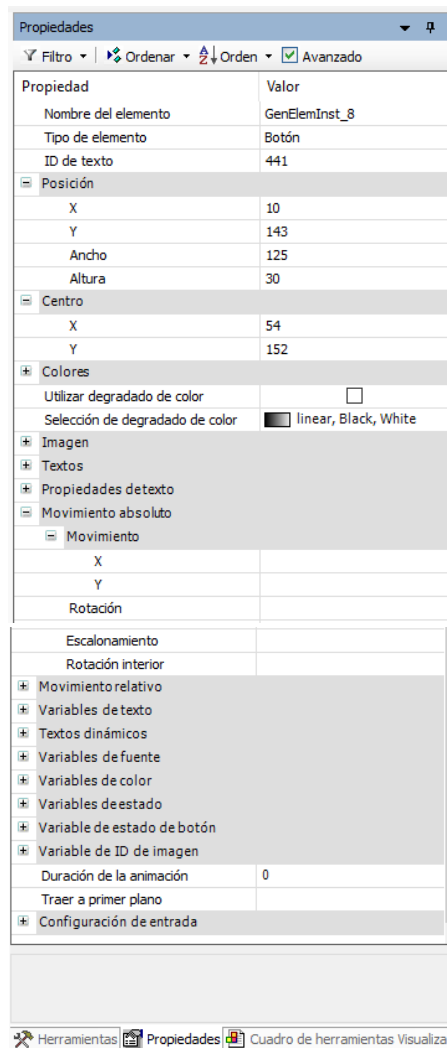


Figura 3.19. Pestaña de propiedades del elemento Botón Visualización RW1.

Cabe decir que, seleccionando el recuadro Avanzado, en la parte superior de la pestaña, se amplía el listado de propiedades que se pueden configurar de cada elemento.

3.8.5. Visualizaciones diseñadas

Una vez conocidas las nociones anteriores, ya se puede diseñar el panel de control del SCADA. En primer lugar, se añadió una visualización que se utiliza como base para el panel y a la que se le ha denominado “Visualizacion_base”.

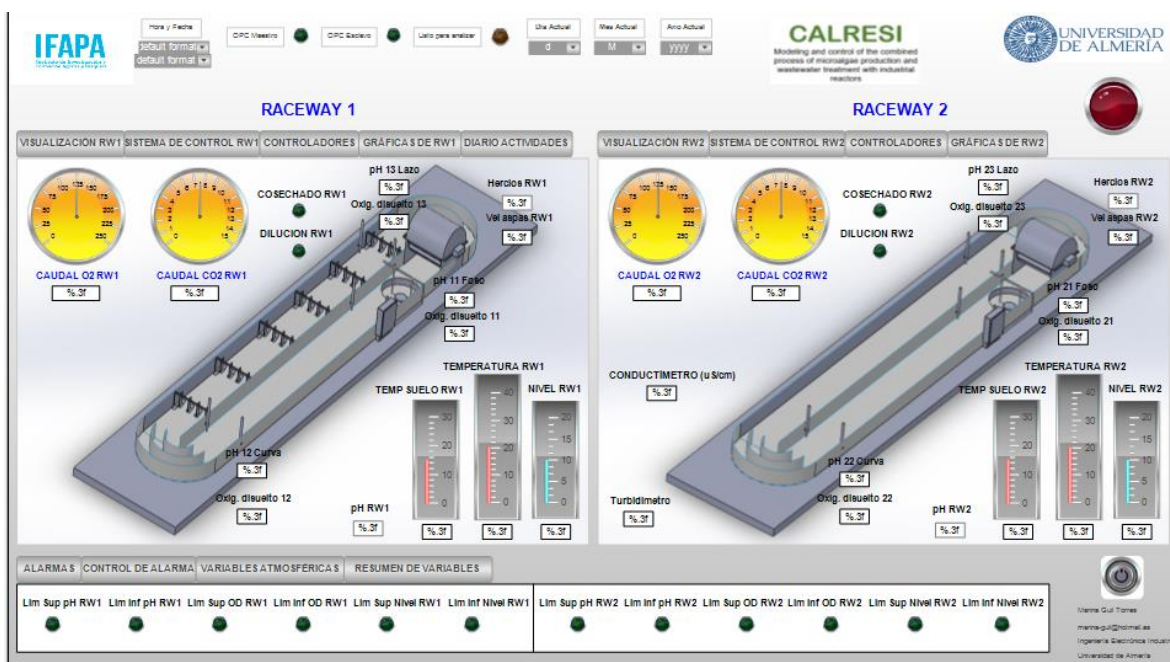


Figura 3.20. Visualización base.

En ella se distinguen 3 partes claramente diferenciadas (Figura 3.20), una superior donde se recoge información sobre la fecha y hora actuales del sistema, sobre el estado de la comunicación Cliente/Servidor y sobre que entidades están relacionadas con el proyecto. Una intermedia donde se visualizan, de forma independiente para cada reactor Raceway y a través de un conjunto de pestañas desplegadas, los datos sobre su estado, control y representación gráfica. Y, por último, una parte inferior donde se muestra información sobre el entorno, gestión de alarmas y una recopilación con las variables que resultan de mayor interés para el usuario, además de un botón de STOP del panel.

Para el diseño de la parte superior se incorporaron elementos Imagen con los logos de las entidades participantes del proyecto, Lámparas, Selectores de fecha/hora y campos de texto. Para la parte intermedia, al tratarse de dos desplegables de pestañas, se precisan, por cada reactor, de un elemento “Marco”, dentro del cual se recogen las diferentes visualizaciones que se quieren mostrar en dicho espacio, y de una botonera que lo gobierne. En la parte inferior del panel se añadieron, nuevamente, un Marco y una botonera para simular el desplegable, un botón y campos de texto.

Para añadir una nueva pestaña en el desplegable, se crea una nueva visualización con la

apariciencia que se desee, pero conservando las dimensiones del Marco creado en la “Visualizacion_base”. Luego se vincula a este último elemento mediante “Configuración de Visualizaciones referenciadas”, como se aprecia en la Figura 3.21.

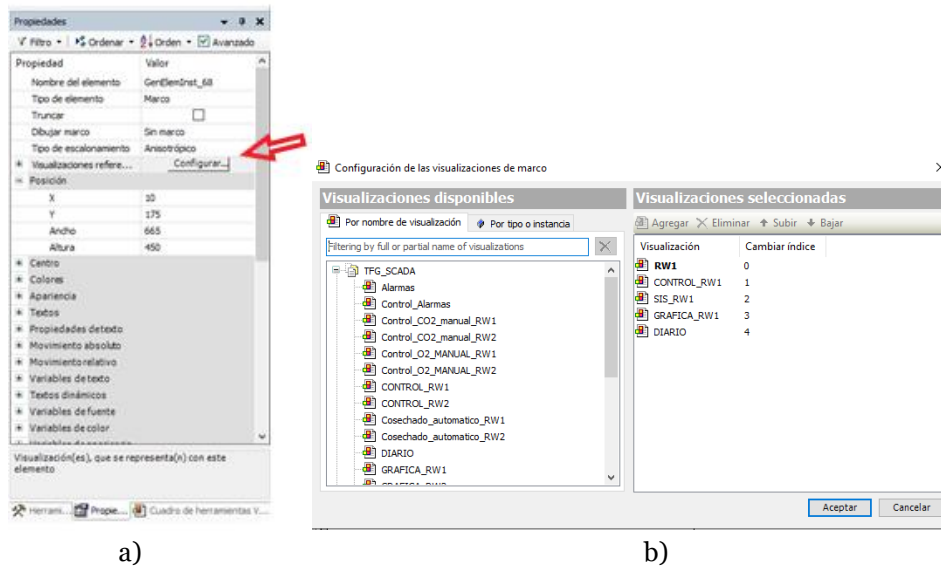


Figura 3.21. Configuración del 1º Marco. a) Propiedades del elemento b) Configuración de las visualizaciones referenciadas.

Los botones serán los encargados de indicar que pestaña se mostrará en pantalla en cada momento. Para lograr eso, dentro de las propiedades del botón, teniendo activadas las de tipo avanzado, se accede a “Configuración de entrada” en “OnClick” (Figura 3.22.a) y se selecciona, de entre las posibles opciones disponibles, el cambio de visualización de marco. Luego solo hay que escoger el Marco y, dentro de este, la visualización correspondiente que aparecerá al pulsar dicho botón del panel (Figura 3.22.b).

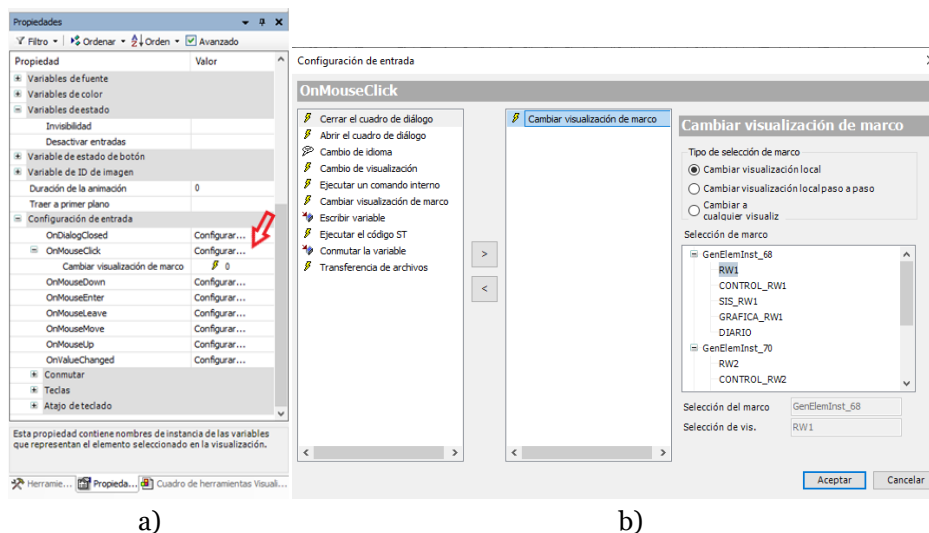
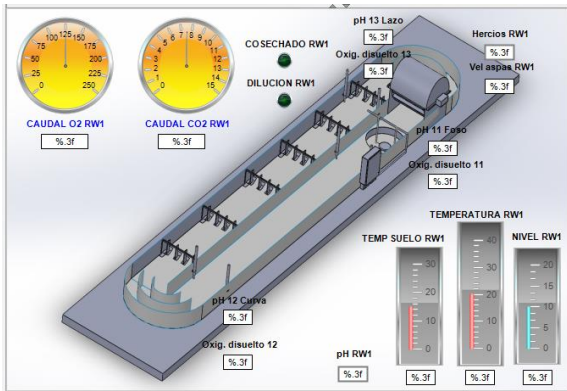


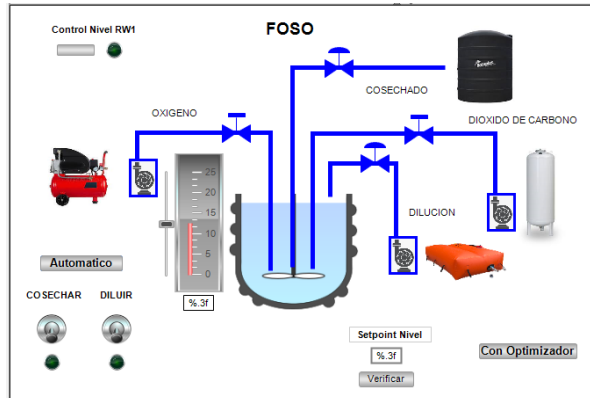
Figura 3.22. Configuración del botón Visualización RW1. a) Propiedades del elemento b) Configuración de entrada.

La visualización base, por tanto, requiere de 3 marcos distintos, los cuales a su vez cuentan con:

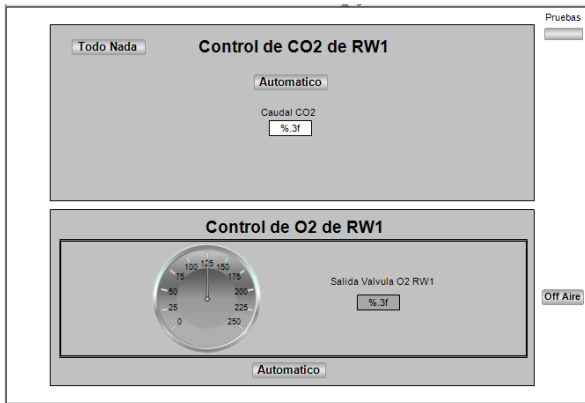
- 1º Marco: 4 visualizaciones o pestañas asociadas que muestran información del Raceway 1 y una extra para el diario de actividades (Ver Figura 3.23).



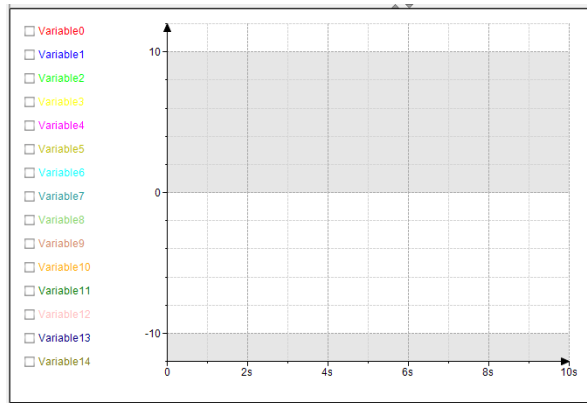
a) Visualización "RW1".



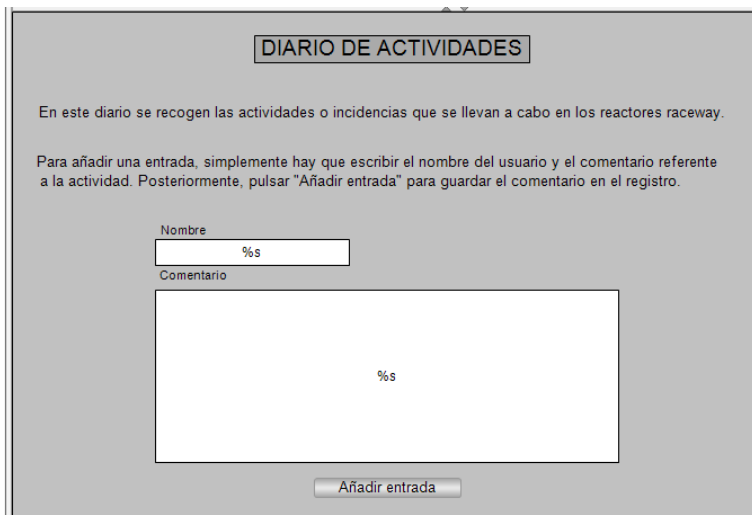
b) Visualización "SIS_RW1".



c) Visualización "CONTROL_RW1".



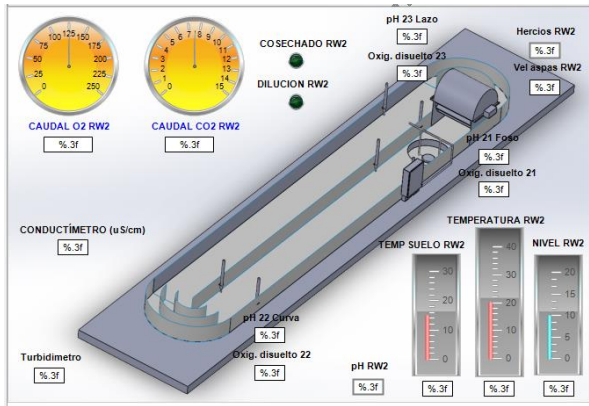
d) Visualización "GRAFICA_RW1".



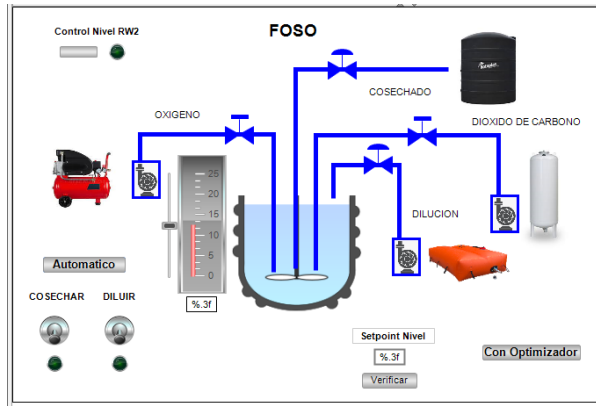
e) Visualización "DIARIO".

Figura 3.23. Visualizaciones asociadas al 1º Marco.

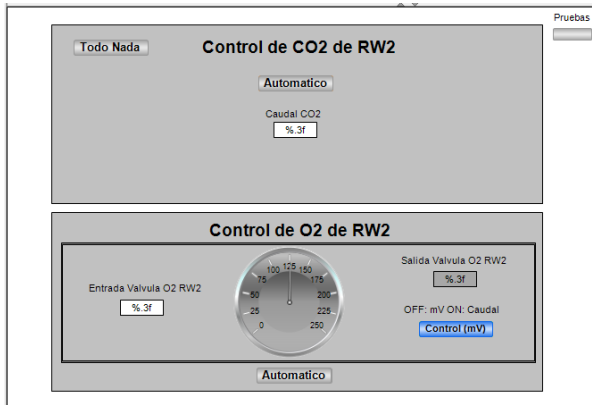
- 2º Marco: 4 pestañas para datos referentes al Raceway 2 (Figura 3.24).



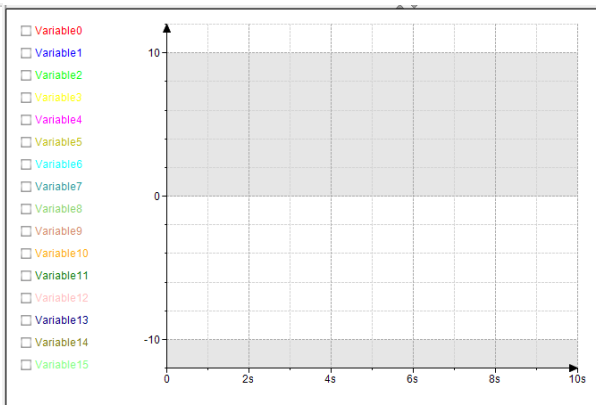
a) Visualización “RW2”.



b) Visualización “SIS_RW2”.



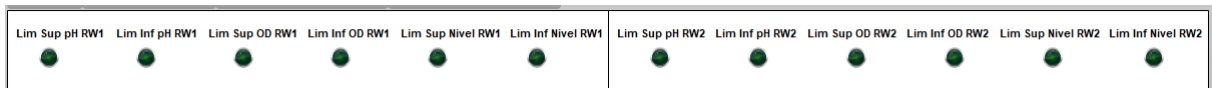
c) Visualización “CONTROL_RW2”.



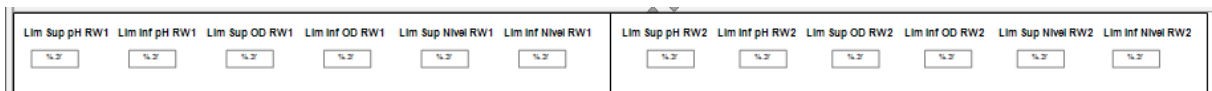
d) Visualización “GRAFICA_RW2”.

Figura 3.24. Visualizaciones asociadas al 2º Marco.

- 3º Marco: 4 visualizaciones para alarmas, condiciones ambientales y listado de variables de interés (Figura 3.25).



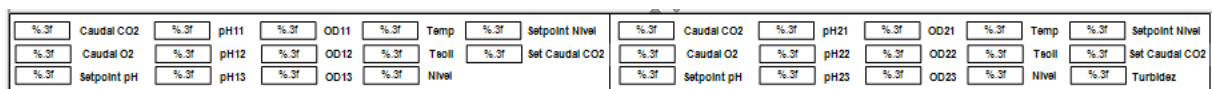
a) Visualización “Alarmas”.



b) Visualización “Control_Alarmas”.



c) Visualización “Variables_atmos”.



d) Visualización “Resumen_de_variables”.

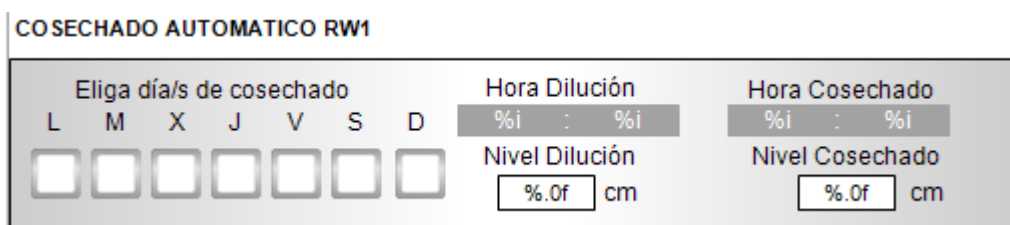
Figura 3.25. Visualizaciones asociadas al 3º Marco.

A su vez, dentro de estas visualizaciones, hay situaciones donde surge la necesidad de crear nuevos marcos, a los que se hará referencia como marcos secundarios, como por ejemplo para diferenciar el modo de trabajo Manual y Automático de los controladores. De esta manera, se procede del mismo modo que en el caso anterior: agregar marcos a las visualizaciones correspondientes, crear las nuevas visualizaciones y referenciarlas y vincular el modo de operación a los botones y los marcos.

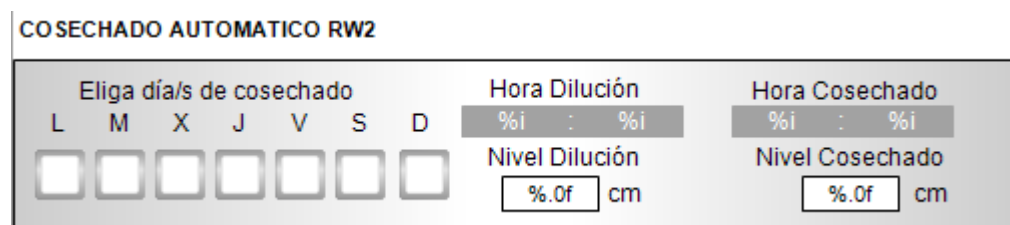
Para estos casos, al solo existir dos posibles ventanas que implementar, se ha optado por crear una única visualización con un modo de trabajo y asociarla al marco, mientras que el otro modo de trabajo se incorpora directamente sobre la visualización que sirve de base. De esta manera, se evita tener un exceso de visualizaciones en el proyecto, lo que puede provocar errores de compilación una vez se ejecute el programa.

Para conmutar de un modo a otro, en este caso se hace uso combinado de la propiedad “Traer a primer plano” y del orden de fichas de los elementos. El efecto logrado es el mismo: el primer modo de operación implementado sobre dicha visualización base permanece inalterable mientras que, a través de los botones de Manual y Automático, el marco con el otro modo de operación aparece y desaparece sobre este ocultándolo o haciendo visible, respectivamente.

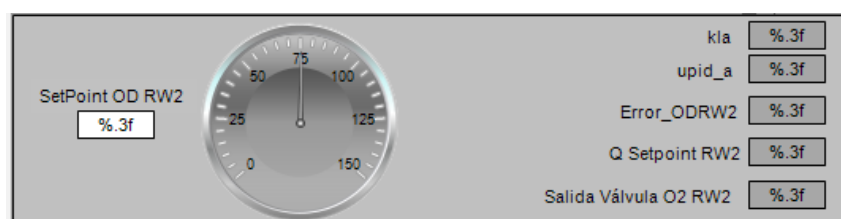
Esta técnica se empleó con el modo Manual/Automático del control de nivel, del control de CO₂ y del control de O₂ de ambos raceways. Adicionalmente, como en las pestañas “Controladores” de ambos reactores los usuarios prueban sus propias implementaciones continuamente, se creó, por cada uno de ellos, un marco y una visualización vacía exclusivamente para que hagan ahí sus experimentaciones sin romper así con la estética del SCADA original y los controladores inicialmente instalados. Las visualizaciones resultantes se recogen en la Figura 3.26.



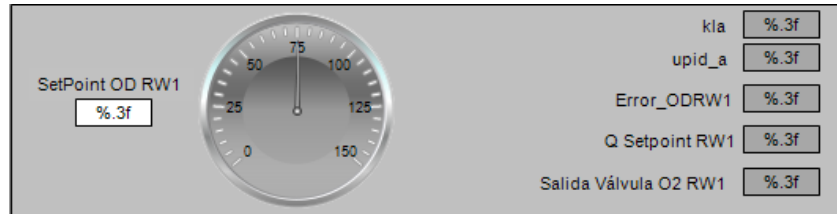
a) Visualización “Cosechado_automatico_RW1”.



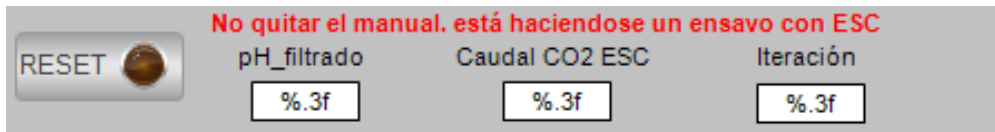
b) Visualización “Cosechado_automatico_RW2”.



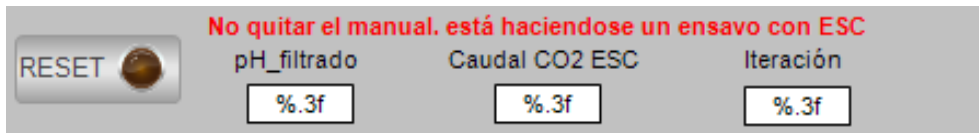
c) Visualización “Control_O2_MANUAL_RW2”.



d) Visualización “Control_O2_MANUAL_RW1”.



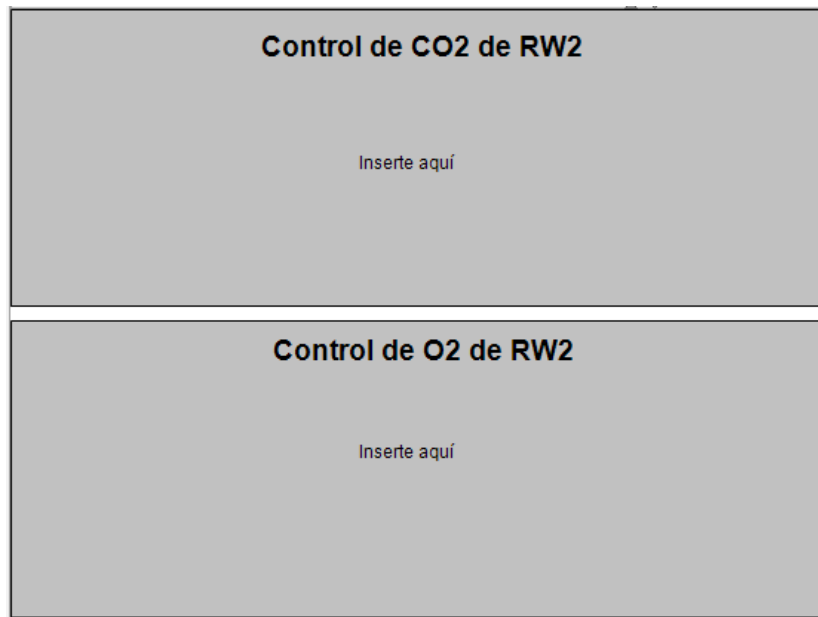
e) Visualización “Control_CO2_manual_RW1”.



f) Visualización “Control_CO2_ manual _RW2”.



g) Visualización “Pruebas_1”.

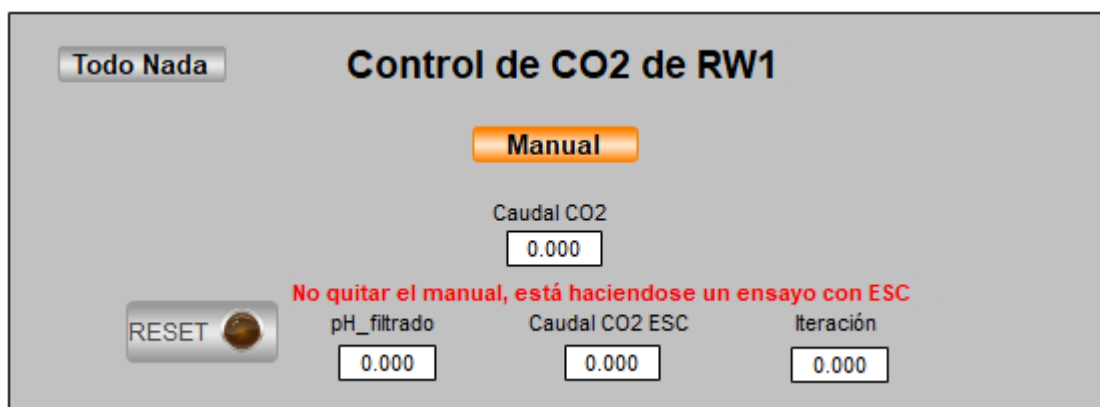


h) Visualización “Pruebas_2”.

Figura 3.26. Visualizaciones asociadas a los marcos secundarios.

Para el caso de los optimizadores, no resulta práctico hacer procesos de este tipo porque los cambios que se producen en las visualizaciones son mínimos. Por ello, a través del orden de fichas de los elementos y las propiedades “Traer a primer plano” y/o “Invisibilidad” se configuran los elementos pertinentes para logran el efecto de conmutación sin sobrecargar el proyecto de visualizaciones.

Para hacer los botones Manual/Automático (3.27.a), como no se ha encontrado una manera más eficiente de hacerlos, se han superpuesto dos de ellos: uno con el texto Automático (3.27.b), y otro, con Manual (3.27.c). Ambos gobernados con la misma variable booleana, de modo que uno la conmuta a True y otro a False cuando son pulsados. Con esta variable y gracias a la propiedad “Invisibilidad” se consigue que el botón superior aparezca o desaparezca simulando así la conmutación entre modo Automático y Manual. De igual forma se configuraron los botones de modo Con/Sin Optimizador y Control de Caudal, del Control de O2 del Raceway 2.



a) Botón conmutado a Modo Automático.

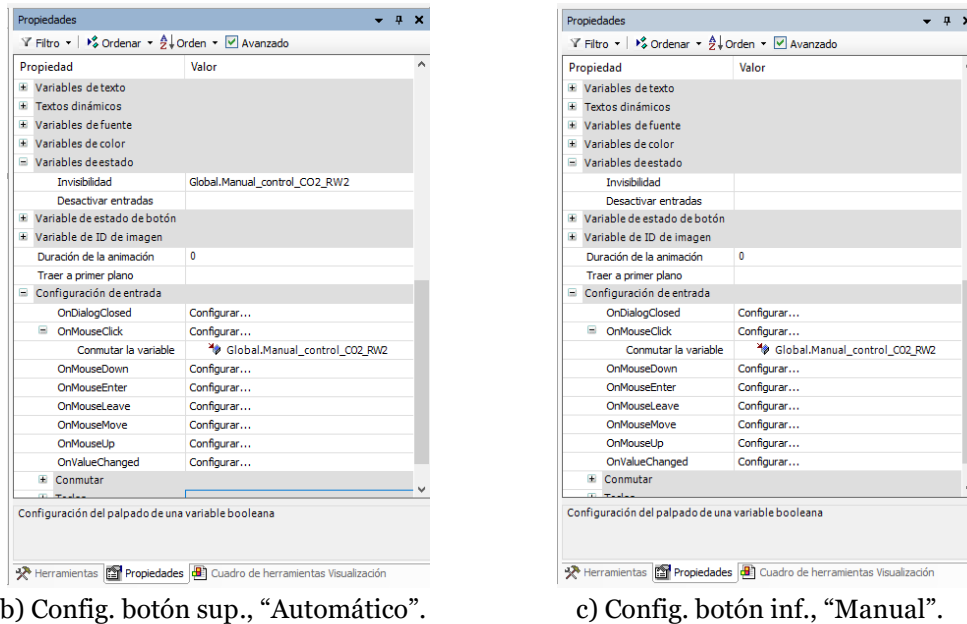


Figura 3.27. Explicación de la conmutación del Botón Manual/Automático, Control CO2 RW2.

En los optimizadores, los elementos se asocian a la variable creada para el botón de modo de operación por medio de la propiedad “Traer a primer plano”. A excepción de los campos de texto Lim_d_RW1, Lim_h_RW1 y sus homólogos para el reactor 2, que además se vinculan con el botón Manual/Automático de esta misma pestaña a través de la propiedad “Invisibilidad” porque no deben ser vistos en modo automático y el marco que debería cubrirlos en tal caso no lo lograba (Figura 3.28).

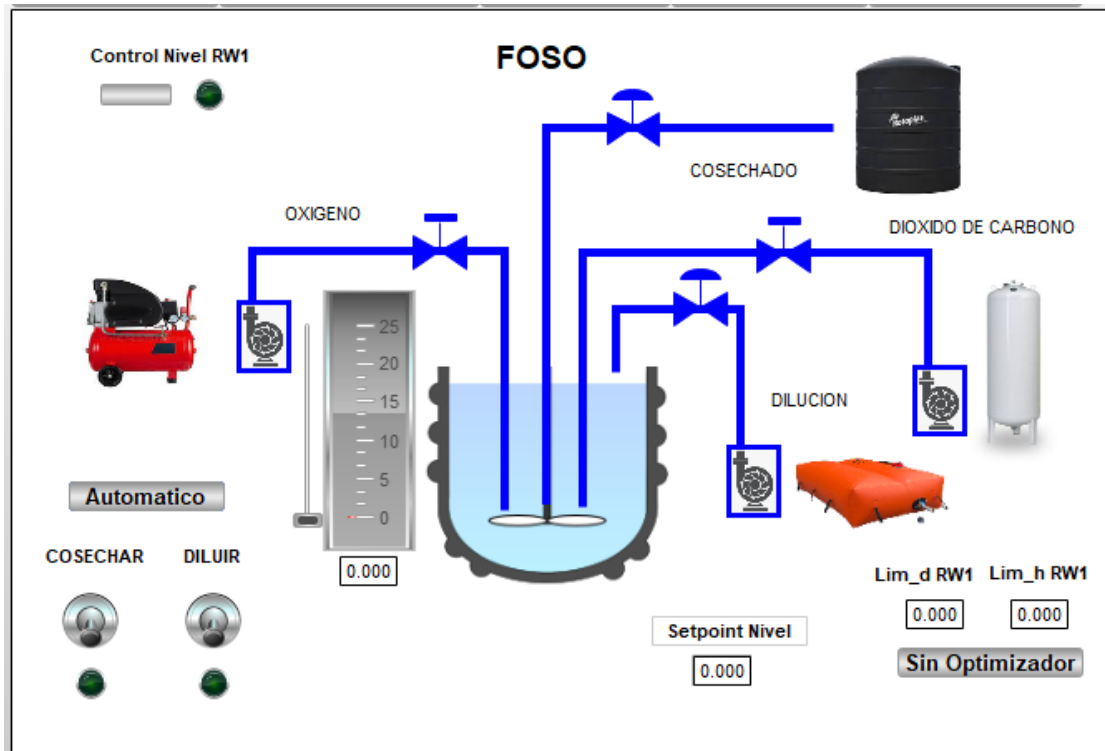


Figura 3.28. Control de Nivel del Raceway 1, Modo Manual y Con Optimizador.

En los siguientes subapartados se comentarán algunos detalles más sobre las visualizaciones y su configuración para la ejecución del proyecto.

3.8.6. Identificadores

En primer lugar, se observa que en los recuadros donde deben aparecer o escribirse valores de variables aparecen textos del tipo %f, %s, Esta es una imposición necesaria para que cuando se ejecute el programa aparezca el valor correspondiente en pantalla. Si no se realiza este ajuste, no aparecerá nada en el recuadro porque el programa hace uso de ese texto para identificar el tipo de valor que debe escribir y/o mostrar en él. La Figura 3.29 muestra la clasificación de estos identificadores.

Especificador	Significado	Especificador	Significado
%c	Caracter	%s	Cadena de texto
%d	Número entero (int)	%u	Entero sin signo
%D	Número entero long (o también %ld)	%U	Entero sin signo long (o también %lu)
%i	Número entero (int)	%x	Hexadecimal sin signo con minúsculas
%f	Punto flotante (float)	%X	Hexadecimal sin signo con mayúsculas
%e	Notación científica con e minúscula	%p	Puntero, dirección de memoria
%E	Notación científica con E mayúscula	%n	Número de caracteres
%g	Utiliza %f o %e según sea más corto	%o	Formato entero octal
%G	Utiliza %f o %E según sea más corto	%O	Formato entero octal long (o también %lo)
%o	Número octal sin signo	%lf	Formato double

Figura 3.29 Clasificación de identificadores.

Además, en los identificadores vinculados a valores numéricos se puede imponer el número de decimales del dato añadiendo en medio un punto y el valor de decimales que se desea. Por ejemplo, %.3f escribe valores numéricos con 3 decimales.

Por último, hacer un inciso para decir que estos campos se pueden hacer mediante elementos Rectángulo o Campo de texto, indistintamente, y que la visualización y escritura de datos en pantalla se logra vinculando una variable a la propiedad “Variable de texto” del elemento y seleccionando la opción Escribir variable dentro de la ventana de configuración de entrada de “OnClick”, respectivamente. La Figura 3.30 muestra cómo ha de hacerse la configuración del recuadro Hercios del reactor 1 para que se visualice o se pueda escribir dicho valor cuando se esté en ejecución.

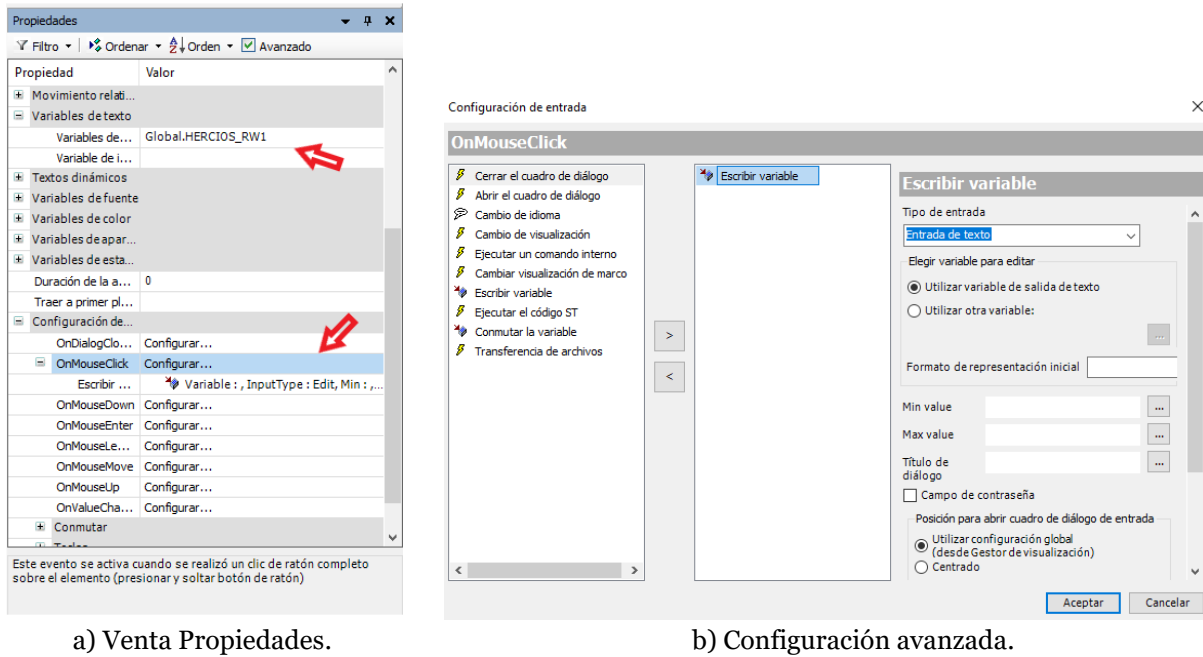


Figura 3.30 Propiedades del recuadro Hercios de RW1.

A la hora de escribir una variable se puede hacer de 5 formas distintas (Figura 3.31), según el tipo de entrada que se seleccione. Para este proyecto, se eligió la entrada de texto donde al clicar sobre el recuadro te permite directamente introducir por teclado el valor de la variable sin más. Otras opciones como `VisuDialogs.Numpad` o `VisuDialogs.Keypad` lo que hacen es mostrar en pantalla un teclado donde el usuario a través del ratón introduce la entrada correspondiente.

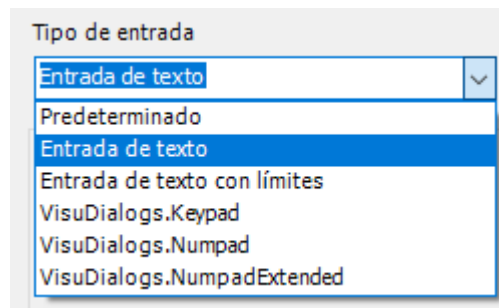


Figura 3.31 Tipos de entrada.

3.8.7. Representaciones gráficas

Para la configuración de las variables que aparecerán en las representaciones gráficas de los elementos “Tendencia” es necesario agregar un objeto “Trend Recording Manager” y crear un “Trend” por cada gráfica creada en visualización.

Aparecerá una pantalla como la de la Figura 3.32, donde se configuran el número de variables que se pueden agregar a ella, el tiempo de muestreo y el máximo de muestras que se tomarán.

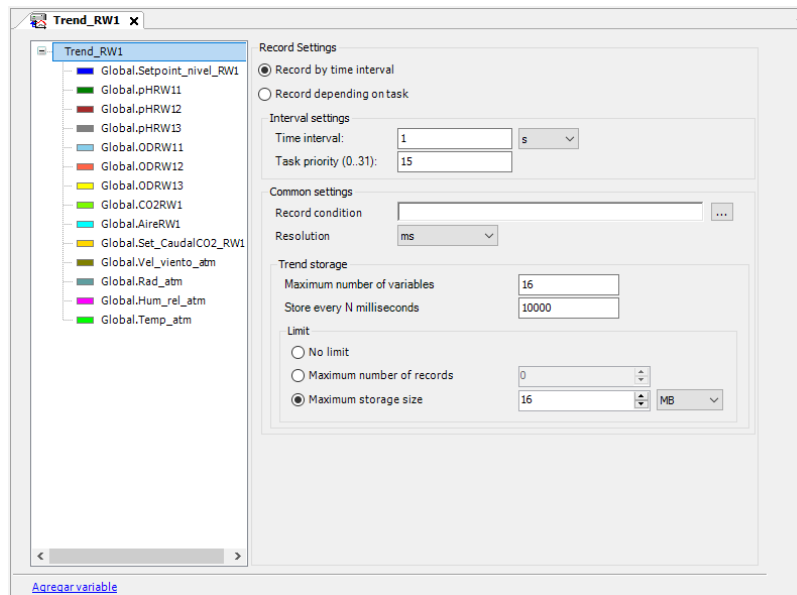


Figura 3.32. Trend asociado a la gráfica del Raceway 1, *Trend_RW1*.

Pulsando en la parte inferior, en “Agregar variable”, se pueden añadir las variables y seleccionar las características de su traza, como el tipo de línea o color (Figura 3.33).

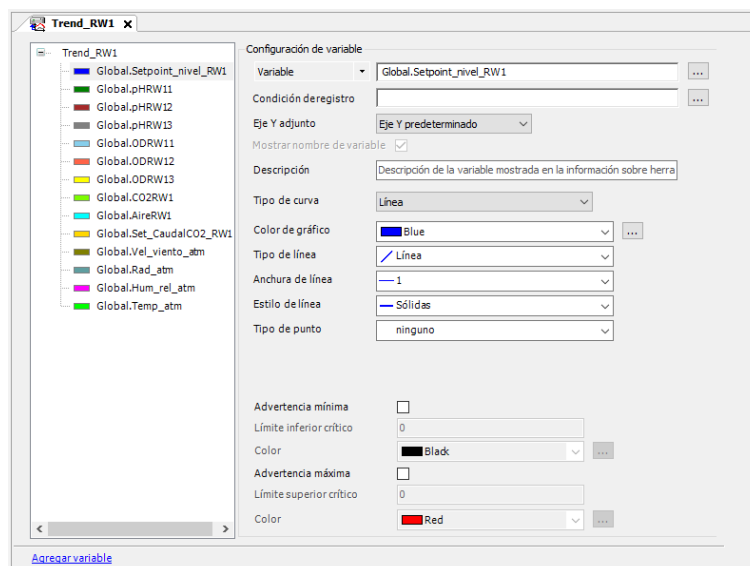


Figura 3.33. Configuración de la variable *Global.Setpoint_nivel_RW1* dentro del *Trend_RW1*.

Luego, en visualización, al incorporar el elemento Tendencia solo hay que asociar el Trend correspondiente a través de la propiedad Registro de tendencia.

Además de estas gráficas, que serán vistas en el panel de control, se han creado dos objetos “Trace” (Application → Agregar objeto → Trace), que representan las mismas trazas que las diseñadas para el panel, pero con la salvedad de que son archivos independientes del proyecto, es decir, se accede a ellas directamente y no a través de la visualización. Por otro lado, haciendo clic derecho sobre ellas se pueden guardar los datos muestreados en un archivo (txt, csv o trace) o poder cargar de nuevo uno ya existente. Su configuración es muy similar a la de los Trends, por lo que no se profundizará en cómo desarrollar uno. La Figura 3.34 muestra la traza

configurada para el reactor 1.

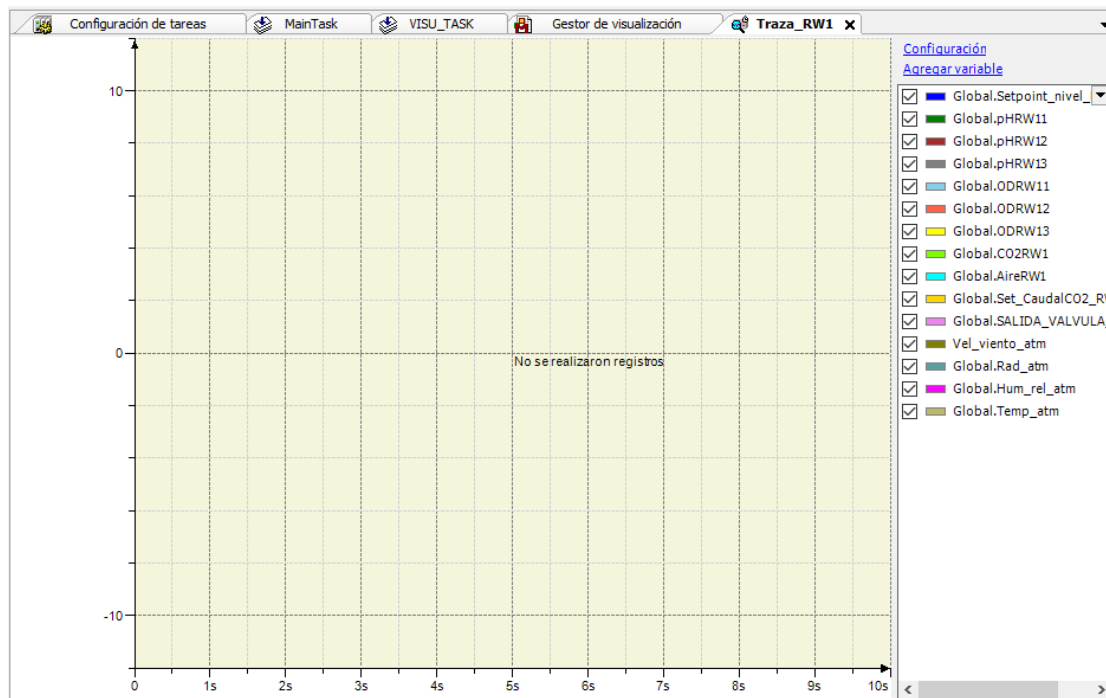


Figura 3.34. Trace *Traza_RW1*, sin ejecución.

3.8.8. Gestor de visualización

Para concluir con el apartado “3.8. Visualizaciones”, mencionar que existe una pestaña llamada “Gestor de visualización” (Figura 3.35), accesible en la sección izquierda de la interfaz, es decir, en el recopilatorio de archivos de proyecto, donde se pueden hacer configuraciones adicionales asociadas al entorno de trabajo de visualización.

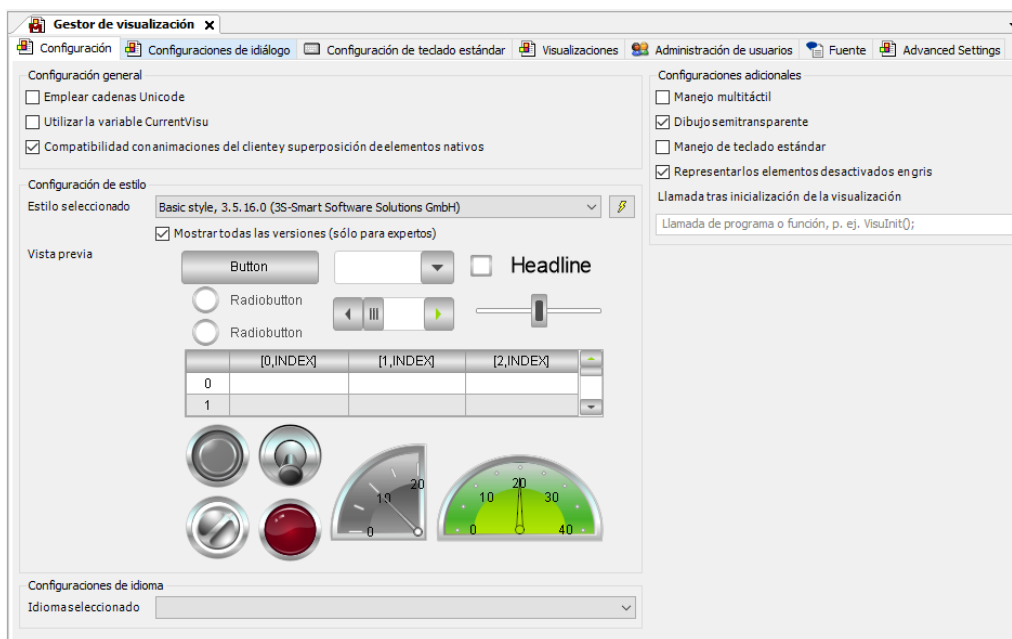


Figura 3.35. Vista del Gestor de visualización.

Dentro de la pestaña “Visualizaciones”, hay que asegurarse de que todas estén seleccionadas, como muestra la Figura 3.36, para que cuando se ejecute el programa todas ellas puedan ser llamadas y visualizadas.

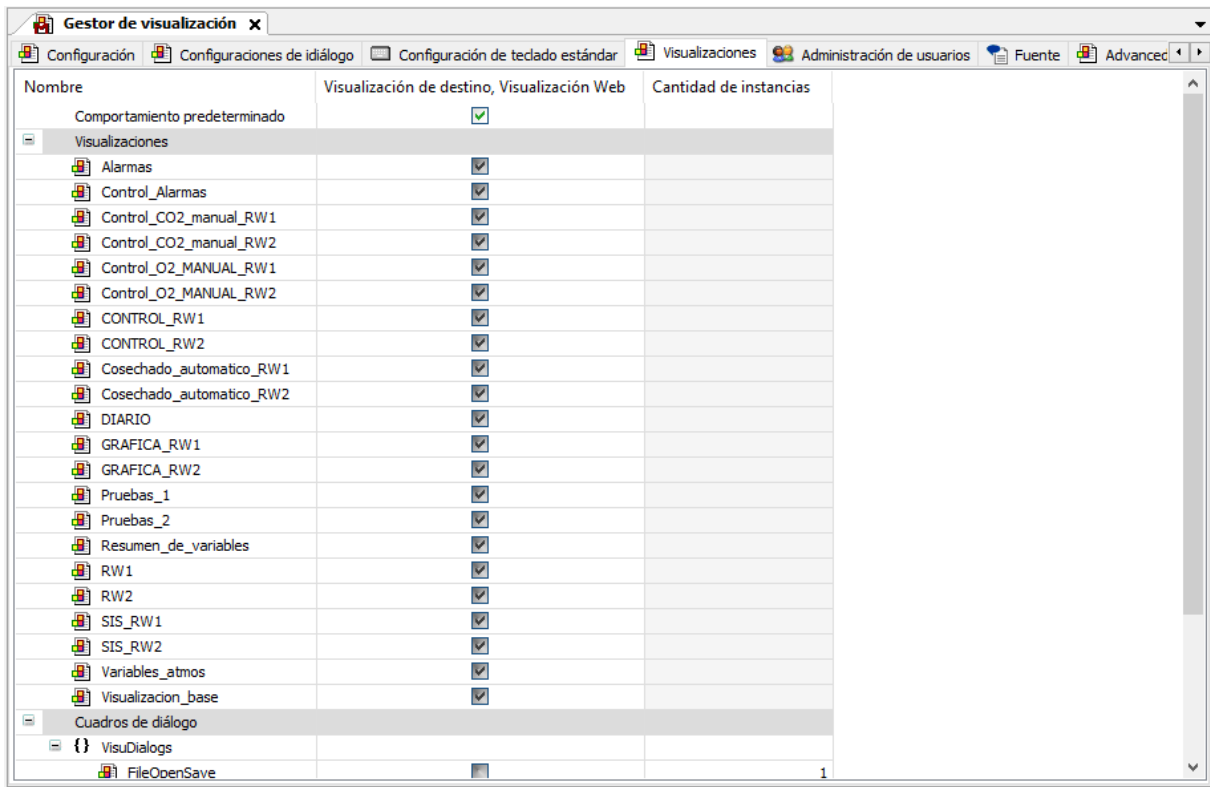


Figura 3.36. Gestor de visualizaciones, sección *Visualizaciones*.

Además del gestor, en el recopilatorio de archivos de proyecto, aparecen dos objetos más vinculados a este primero, “Visualización de destino” y “Visualización Web”.

Visualización de destino o TargetVisu es una opción de visualización que muestra las interfaces de operación directamente en el PLC o en una pantalla integrada o conectada (Figura 3.37). De esta forma se combina el controlador y la visualización en un solo dispositivo. En resumidas cuentas, las pantallas de visualización creadas se pueden mostrar en un controlador equipado con Codesys TargetVisu sin necesidad de ningún hardware adicional. Las pantallas se muestran directamente en la pantalla interna o externa del dispositivo. [5]

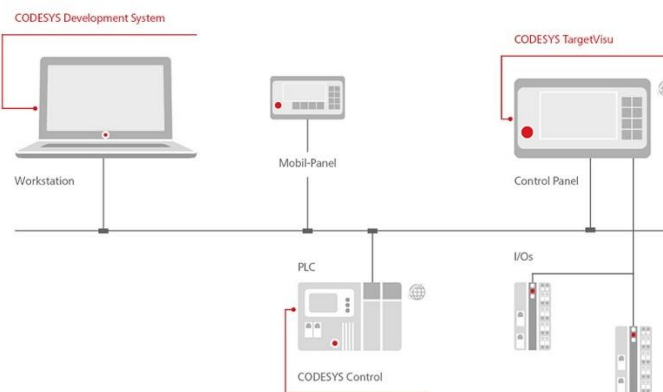


Figura 3.37. Arquitectura de comunicación con TargetVisu. Fuente [3]

Por otro lado, Visualización Web o WebVisu permite el teleacceso al panel de control a través de internet (Figura 3.38), debido a que las visualizaciones se descargan junto con la aplicación en el controlador y las visualizaciones se realizan en navegadores web estándar mediante HTML5. Para hacer uso de ello basta con llamar a la página de inicio utilizando la dirección IP / nombre de dominio del controlador en la línea de dirección del navegador de Internet en el dispositivo de visualización (`http://127.0.0.1:8080/webvisu.htm` o `http:// [Nombre de dominio]: [Dirección de puerto] /webvisu.htm`). [5]

Si se utiliza un PLC Virtual, como es el caso, se introduce la siguiente dirección en el buscador: `http://localhost:8080/webvisu.htm`.

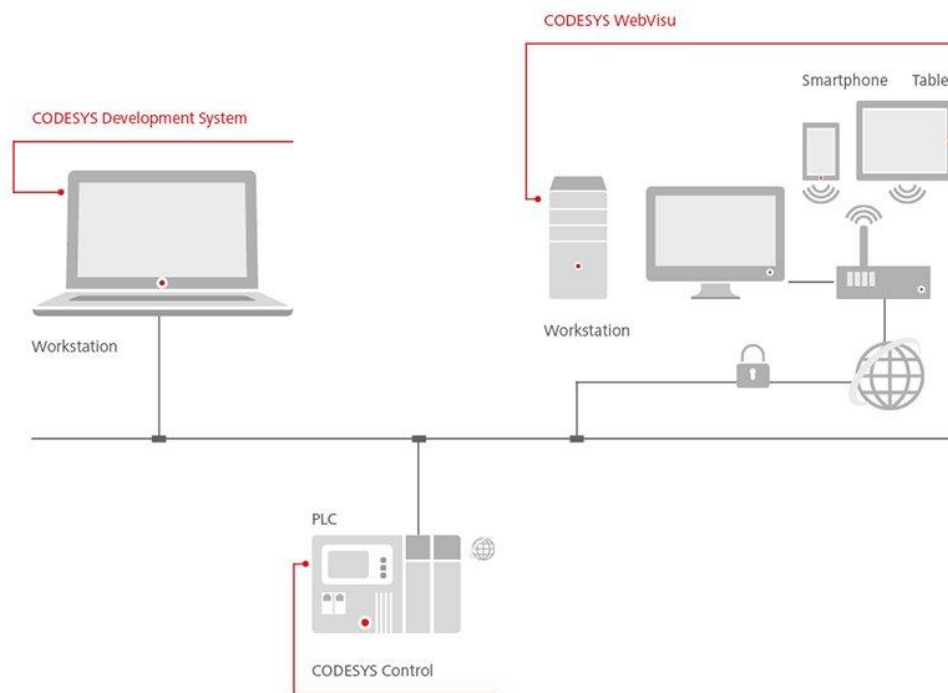
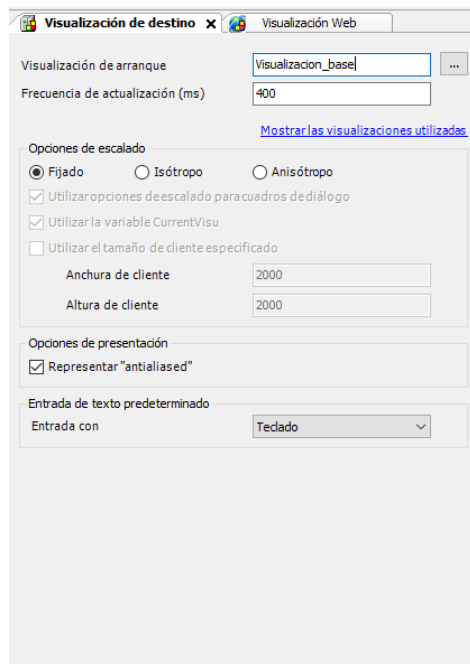


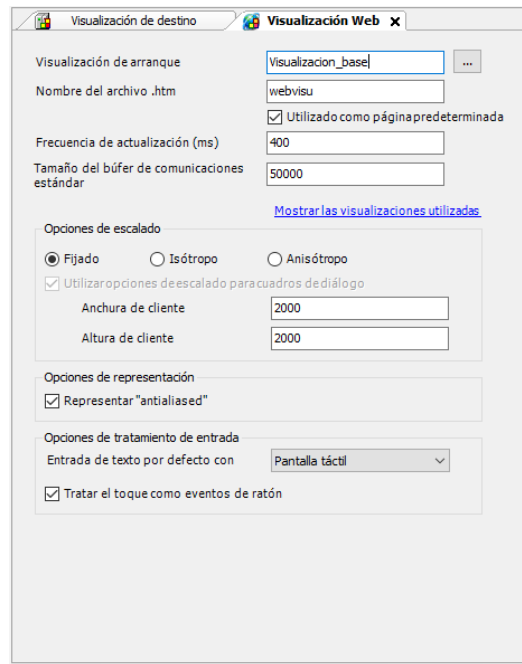
Figura 3.38. Arquitectura de comunicación con WebVisu. Fuente [4]

En ambos casos, como indica la Figura 3.39, hay que estipular una frecuencia de actualización de 400ms y que la visualización de arranque sea “Visualizacion_base”, ya que esta gobierna al conjunto de visualizaciones creadas y permite el acceso a cualquiera de ellas, ya sea directa o indirectamente.

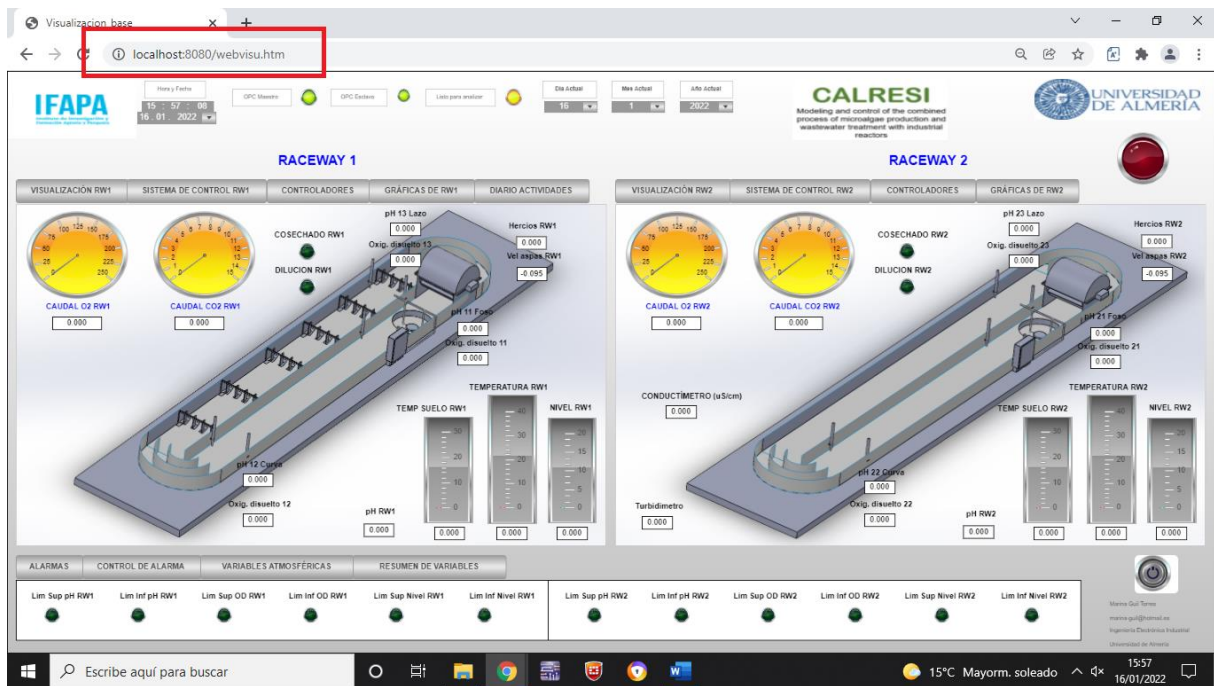
Luego, para que las proporciones de la visualización de TargetVisu y WebVisu se ajusten a la pantalla completa, se selecciona Anisotrópico en ambos casos y en TargetVisu, además, se desmarca “Utilizar el tamaño de cliente especificado”.



a)



b)



c)

Figura 3.39. Configuración de los visualizadores. a) Configuración de TargetVisu, b) Configuración de WebVisu. c) Visualización de WebVisu.

Estando en ejecución el proyecto, se podrá acceder a ambas visualizaciones, ya sea de forma espontánea como con TargetVisu o a través de Internet, con la dirección especificada anteriormente, como con WebVisu (Figura 3.39.c). Por añadidura, se puede hacer esto mismo directamente desde el entorno de desarrollo, visualización a visualización. Sin embargo, las proporciones se descuadran cuando se hace esto por lo que, para solventar el problema, hay que hacer una serie de reajustes.

En primer lugar, dentro de la pestaña Herramientas, en Opciones, se configura la opción “Derivar automáticamente estilo de visualización”, en Estilos de Visualización, y “Anisotrópico”, en Visualización.

Luego haciendo clic derecho sobre cualquiera de los objetos Visualización, disponibles en el recopilatorio de archivos de proyecto (sección derecha de la interfaz), aparece una opción llamada “Propiedades”. Dentro de ella se abre la pestaña Visualización y se activa la preferencia “Utilizar el tamaño de visualización reconocido automáticamente”, en el caso de que no lo estuviese.

Con ello se logra ajustar los límites de las visualizaciones diseñadas con los de la pantalla del entorno de visualización, una vez en ejecución.

3.9. Configuración de tareas

Para que cuando llegue el momento de ejecutar el proyecto, este lo haga cumpliendo unas especificaciones concretas de prioridad, tasa de repetición, ..., hay que asegurarse de que el administrador de tareas este configurado como corresponde.

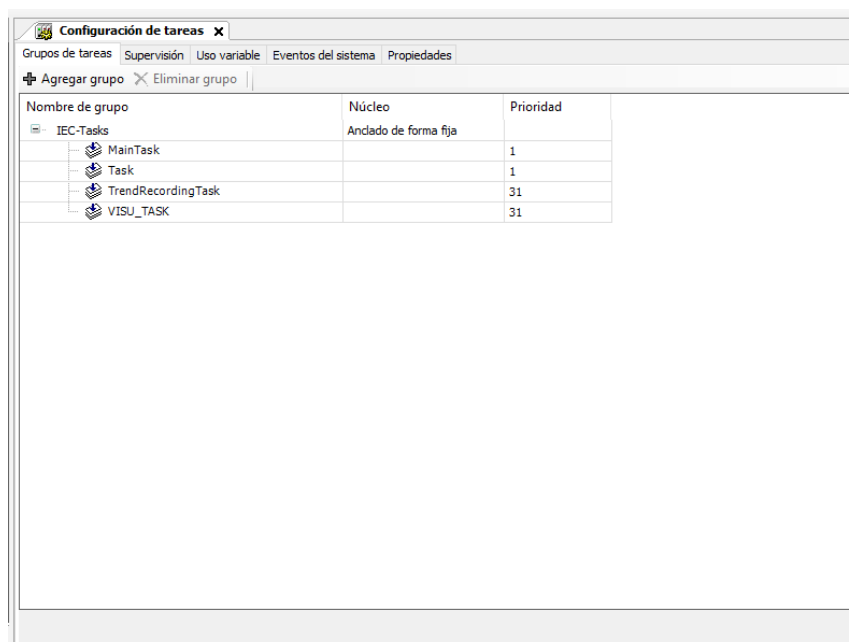


Figura 3.40. Configuración de tareas.

Como se muestra en la Figura 3.40, en este proyecto se trabaja con 4 tareas, dos de ellas vinculadas a la ejecución de visualizaciones y que se generan automáticamente al introducir una visualización en el proyecto, y otras dos relacionadas con las diferentes POUs creadas.

Al igual que “TrendRecordingTask” y “VISU_TASK”, la tarea “MainTask” se crea de forma automática, una vez se genera el proyecto por primera vez, y se vincula con la única unidad de programación que existía en ese instante, “PLC_PRG”.

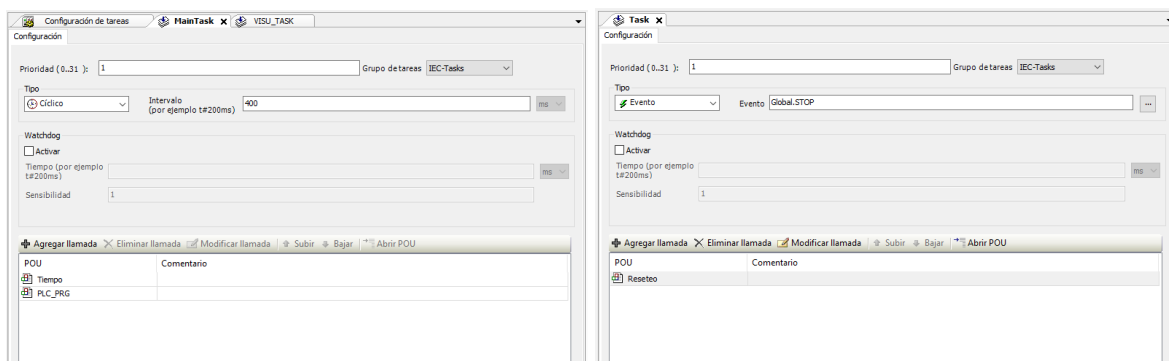
Dentro de cada tarea se pueden configurar diversos parámetros: la prioridad, el tipo de tarea o agregar un Watchdog. La prioridad hace referencia al orden en el que se ejecutarán cada una

de las tareas, siendo las próximas a 0 las de mayor prioridad y las cercanas a 31 las de menos. Es aconsejable asignar un valor único para cada tarea con el fin de prevenir errores. Generalmente desde 0 a 24 se destinan a labores del controlador mientras que el resto se dejan para tareas de fondo. En este caso se establecieron “TrendRecordingTask” y “VISU_TASK” con una prioridad de 31, como aparecieron por defecto, y “MainTask” y “Task” con prioridad 1, dado que se pretende que ambas tareas se ejecuten al mismo tiempo una vez sean llamadas.

Dentro de los tipos de tarea se diferencia 4 posibles casos: Cíclico, Evento, Ejecución libre y Estado. Mientras que los tipos Evento o Estado llaman a la ejecución de los programas cuando se produce un flanco ascendente o una salida booleana True de una variable global concreta, respectivamente; el tipo cíclico como su nombre indica ejecuta en ciclos las POU llamadas con un tiempo de ciclo que se define a través de la entrada Intervalo. Por último, con Ejecución libre la tarea llama a las unidades de programa y se ejecutan reiteradas veces cada vez que se produce una pasada completa. Esta última no tiene un tiempo de ciclo definido.

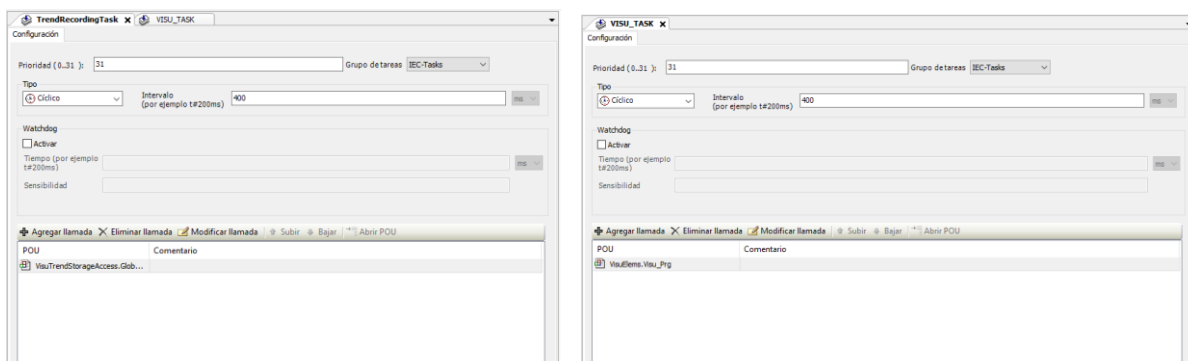
Para el proyecto se configuraron todas las tareas, a excepción de “Task”, como cíclicas con un intervalo de 400 ms, ya que en el SCADA de LabVIEW ese era el periodo de repetición del bucle While.

También se ha hecho mención a la posibilidad de activar un Watchdog, que no es otra cosa que un monitoreo del tiempo para una tarea. Si se excede dicho tiempo, la tarea se detiene y pasa a un estado de Error, al igual que el resto de tareas o aplicaciones vinculadas a este. En este caso se ha prescindido de su uso.



1) MainTask.

2) Task.



3) TrendRecordingTask.

4) VISU_TASK.

Figura 3.41. Configurador de cada tarea del proyecto.

Como se expone en la Figura 3.41, se llama a ejecución a dos POU's con la tarea MainTask, "Tiempo" y "PLC_PRG", debido a que es preciso que ambas unidades se ejecuten cíclicamente en el tiempo mientras que "Reseteo" es llamada por Task y solo se debe ejecutar si se pulsa el botón de STOP en el panel de control. Por ese motivo esta última tarea es del tipo Evento.

3.10. Conexión con el servidor.

Al igual que ocurre con el dispositivo PLC, para comunicar el proyecto de Codesys con otros programas o con un sistema real mediante un servidor OPC hay que realizar una serie de ajustes en este previamente a su uso. Con la idea de que cualquier lector de esta memoria sea capaz de hacer uso de esta interfaz de comunicación se describirá esta secuencia paso a paso.

En primer lugar, se efectúa la instalación y activación de la licencia del servidor. Este proceso se explicará más en profundidad en el capítulo "Anexos".

A continuación, se ejecuta el archivo "OPCConfig", disponible en la carpeta CODESYS OPC Server V3. Emergerá la ventana que indica la Figura 3.42.

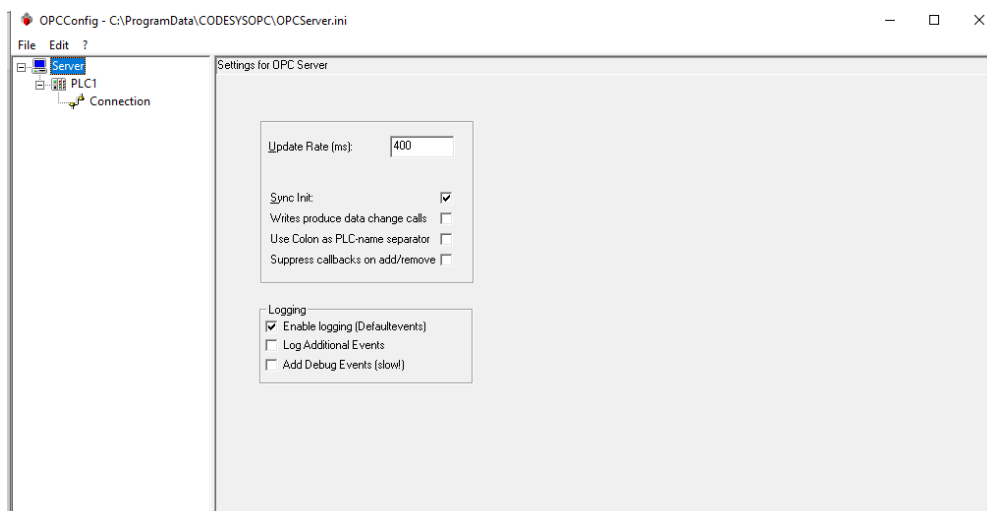


Figura 3.42. Ventana de OPCConfig.

Como se observa en ella, aparece el servidor y un PLC por defecto, que se eliminará. Dentro de la pestaña del servidor el único cambio que se ha efectuado es el valor de "Update Rate" a 200 ms, para que coincida con la tasa de actualización de datos del otro servidor con el que se comunicará.

Luego, se crea un nuevo PLC haciendo clic derecho sobre la palabra "Servidor". Se advertirá que surge una nueva pestaña de configuración referida a este (Figura 3.43). La comunicación en este proyecto se realizará a través de Gateway V3, por ser la interfaz más actual y ampliamente utilizada. En cuyo caso, se conservaron todos los datos que vienen por defecto y se introdujo el nombre de usuario y contraseña del PLC, definidos en el apartado "3.3. Conexión con el dispositivo PLC", en los campos "User" y "PLC Password". Si no se introducen dichos valores, los datos del proyecto que se asocian al servidor estarán restringidos y no aparecerán cuando se intente acceder a ellos desde otros programas en comunicación con el servidor.

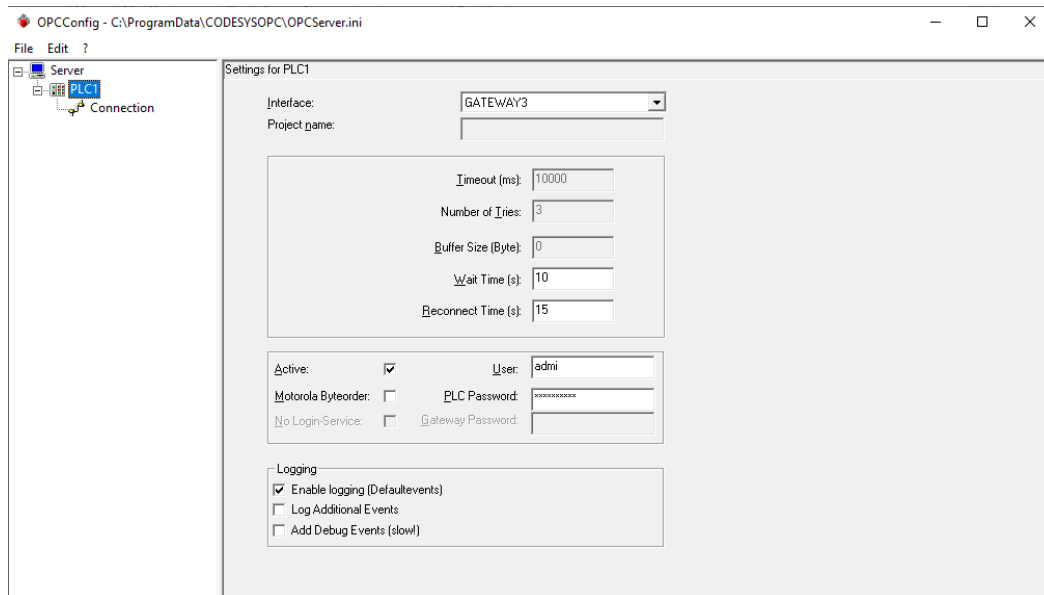


Figura 3.43. Configuración del PLC1.

Por último, pinchando sobre el desplegable “Connection” y haciendo clic sobre Edit, se puede hacer la conexión con el PLC (Figura 3.44).

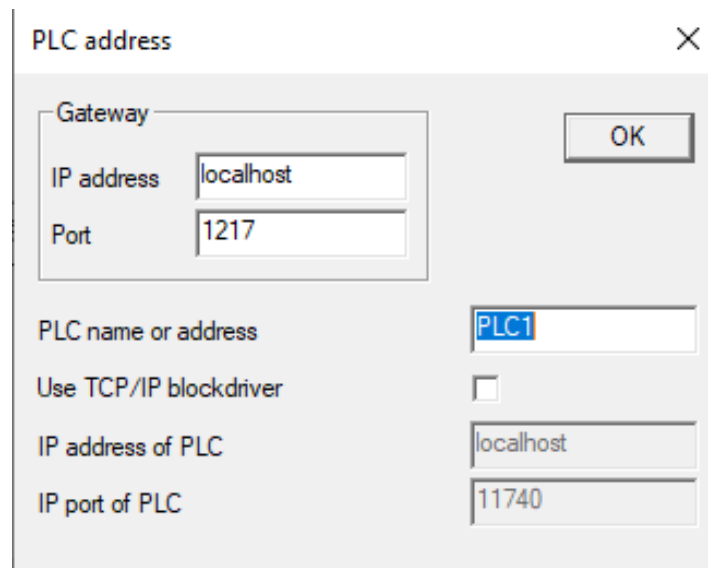


Figura 3.44. Configuración de la dirección del PLC.

Para ello se introduce el nombre o la dirección del dispositivo PLC del que se estuvo hablando en el apartado “3.3. Conexión con el dispositivo PLC”. Cabe decir que es preferible introducir el nombre ya que la dirección puede variar de una ejecución a otra, si se apaga y se reinicia el PLC de nuevo, mientras que el nombre del dispositivo es inalterable por sí mismo.

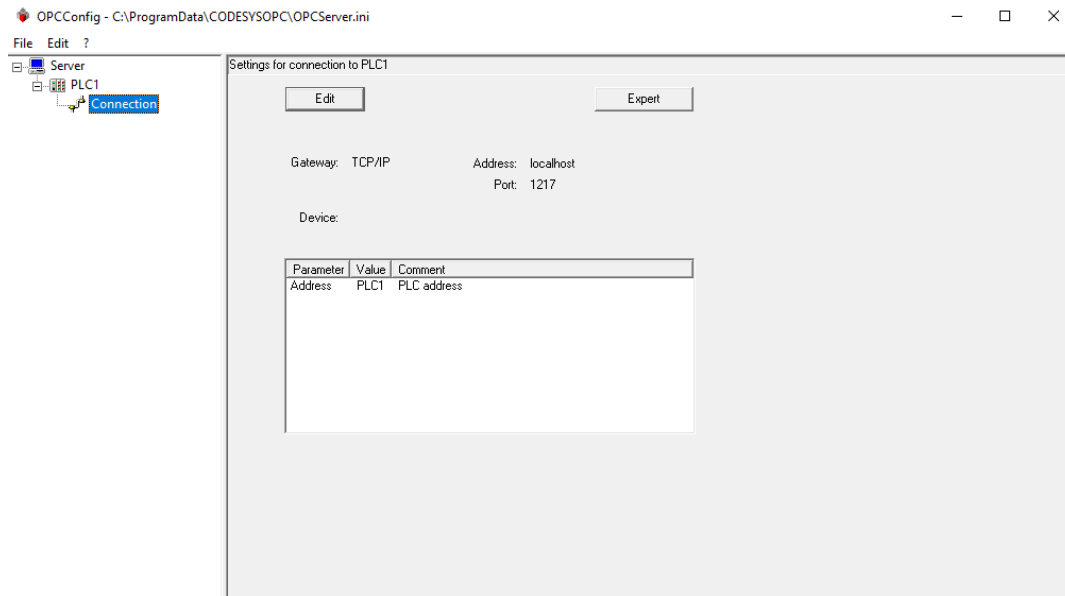


Figura 3.45. Configuración de la conexión.

Si se aceptan los cambios realizados, en la pestaña de configuración de la conexión, aparecerán actualizados los datos del PLC, como muestra la Figura 3.45.

Por último, se guardan las modificaciones efectuadas en el archivo “OPCConfig” para que se actualicen también en el archivo “OPCServer.ini”. Con esto es suficiente para propiciar la comunicación del servidor OPC con el proyecto de Codesys. En el siguiente apartado de la memoria se explicará cómo asociar elementos de este al servidor.

3.11. Configuración de símbolos

Cuando el proyecto está apto para comunicarse con el resto de aplicaciones a través del servidor, se genera una lista de símbolos en el propio entorno de programación con todas aquellas variables que serán transferidas. Para ello se crea un objeto “Configuración de símbolos” y se hace una compilación del programa para verificar que no haya errores de código (Pestaña Compilar, opción Crear Código o directamente F11).

Posteriormente, se seleccionan aquellas variables de entre todas las creadas, ya sea a nivel global como local, que se quieran compartir y se guarda el proyecto. Antes de esto, se puede establecer que derechos de acceso a cada variable tiene Codesys (Figura 3.46), pudiendo ser de entrada (lectura), de salida (escritura) o de entrada/salida (lectura/escritura).



Figura 3.46. Tipos de derechos de acceso.

Con la primera ejecución del proyecto tras este proceso estarán disponibles todas las variables compartidas en el servidor.

En la lista generada se diferenciarán dos clases de datos que han sido compartidos. Por un lado, están aquellos que coinciden con los que el servidor actualmente instalado contiene y que se han declarado como datos de entrada/salida. Y, por otro lado, unos nuevos datos que antes no se compartían, pero que ahora se hacen necesarios, que no son otros que los que se vinculan a los scripts de Matlab. Estos datos se han declarado de entrada o entrada/salida en función de si solo se leen del servidor y se envían a los scripts para que se ejecuten o si se leen y se sobrescriben con valores distintos tras pasar por estos archivos.

La Figura 3.47 recoge la declaración de variables compartidas y sus derechos de acceso.

Símbolos	Derechos de acceso	Máximo	Atributo	Tipo	Variables de miembro	Comentario
[-] Tiempo						
[-] ANO_ACTUAL	rw	rw		INT		
[-] DIA_ACTUAL	rw	rw		INT		
[-] Diario_activa		rw		BOOL		
[-] FECHAYHORA_ACTUAL		rw		DATE_AND_TIME		
[-] FECHAYHORA_STRING		rw		STRING		
[-] FECHA_ACTUAL		rw		DATE		
[-] FECHA_STRING		rw		STRING		
[-] HORA_ACTUAL_ESP	rw	rw		TIME_OF_DAY		
[-] HORA_DECIMAL	rw	rw		REAL		
[-] HORA_NORMAL		rw		INT		
[-] HORA_STRING		rw		STRING		
[-] HoraMinutos		rw		TIME_OF_DAY		
[-] MES_ACTUAL	rw	rw		INT		
[-] MINUTOS_NORMAL		rw		INT		
[-] OBTENERFECHAYHORA		rw		RTCLK.GetDateAndTime	...	
[-] PULSO		rw		TON	...	
[-] SEGUNDOS_NORMAL		rw		REAL		
[-] TIEMPO_ACTUAL		rw		TIME		
[-] TIEMPO_ACTUAL_ESP		rw		TIME		
[-] TIEMPO_HSEG		rw		DINT		
[-] TIEMPO_MHORA		rw		INT		
[-] TIEMPO_MSEG		rw		DINT		
[-] TIEMPO_SHORA		rw		REAL		
[-] TIME_IN_SEC	rw	rw		LREAL		
[-] TON_0		rw		ton	...	
[-] TON_1		rw		TON	...	
[-] TON_2		rw		TON	...	
[-] TON_3		rw		TON	...	
[-] TON_4		rw		TON	...	
[-] TON_5		rw		ton	...	
[-] hora_cos_RW1		rw		INT		
[-] hora_cos_RW2		rw		INT		
[-] hora_dil_RW1		rw		INT		
[-] hora_dil_RW2		rw		INT		
[-] min_cos_RW1		rw		INT		
[-] min_cos_RW2		rw		INT		
[-] min_dil_RW1		rw		INT		
[-] min_dil_RW2		rw		INT		

a) Lista de símbolos del POU “Tiempo”.

Símbolos	Derechos de acceso	Máximo	Atributo	Tipo
[-] PLC_PRG				
[-] Aceptar_sp_RW1	rw	rw		INT
[-] Aceptar_sp_RW2	rw	rw		INT
[-] Cont_iter	rw	rw		ULINT
[-] DiaSeleccionado1num	rw	rw		ARRAY [0..6] OF INT
[-] DiaSeleccionado2num	rw	rw		ARRAY [0..6] OF INT
[-] Et_1_num	rw	rw		INT
[-] Et_num	rw	rw		INT
[-] HORA_COSECHADO_RW1_num	rw	rw		UINT
[-] HORA_COSECHADO_RW2_num	rw	rw		UINT
[-] HORA_DILUCION_RW1_num	rw	rw		UINT
[-] HORA_DILUCION_RW2_num	rw	rw		UINT
[-] HoraMinutos_num	rw	rw		UINT

Desarrollo de una herramienta SCADA en Codesys para fotobiorreactores industriales

<input checked="" type="checkbox"/>	Manual_control_O2_RW2_num			INT
<input checked="" type="checkbox"/>	OFF_CAUDAL_num			INT
<input checked="" type="checkbox"/>	RESET_RW1_num			INT
<input checked="" type="checkbox"/>	Tiempo_Muestreo_num			INT
<input checked="" type="checkbox"/>	Vector_actividades			ARRAY [0..2] OF STRING

b) Lista de símbolos del POU “PLC_PRG”.

Símbolos	Derechos de acceso	Máximo	Atributo	Tipo
<input type="checkbox"/> ANADIR_ENTRADA				BOOL
<input type="checkbox"/> AUTOMATICO_CONTROL_RW1				BOOL
<input type="checkbox"/> AUTOMATICO_CONTROL_RW2				BOOL
<input checked="" type="checkbox"/> Activar_EVMedioRW1				BOOL
<input checked="" type="checkbox"/> AireRW1				LREAL
<input checked="" type="checkbox"/> AireRW2				LREAL
<input checked="" type="checkbox"/> Aux1				LREAL
<input checked="" type="checkbox"/> Aux2				LREAL
<input checked="" type="checkbox"/> Aux3				LREAL
<input checked="" type="checkbox"/> Aux4				LREAL
<input checked="" type="checkbox"/> BOTON_COSECHADO_RW1				BOOL
<input checked="" type="checkbox"/> BOTON_COSECHADO_RW2				BOOL
<input checked="" type="checkbox"/> BOTON_DILUIR_RW1				BOOL
<input checked="" type="checkbox"/> BOTON_DILUIR_RW2				BOOL
<input checked="" type="checkbox"/> Bajar_nivel_RW1				BOOL
<input checked="" type="checkbox"/> Bajar_nivel_RW2				BOOL
<input checked="" type="checkbox"/> BombaCosechaRW1				BOOL
<input checked="" type="checkbox"/> BombaCosechaRW1_F				BOOL
<input checked="" type="checkbox"/> BombaCosechaRW2				BOOL
<input checked="" type="checkbox"/> BombaCosechaRW2_F				BOOL
<input checked="" type="checkbox"/> Boton_Aire_RW1				BOOL
<input checked="" type="checkbox"/> CO2RW1				LREAL
<input checked="" type="checkbox"/> CO2RW2				LREAL
<input type="checkbox"/> CONTROL_NivelRW1				BOOL
<input type="checkbox"/> CONTROL_NivelRW2				BOOL
<input type="checkbox"/> COSECHADO_RW1				BOOL
<input type="checkbox"/> COSECHADO_RW2				BOOL
<input type="checkbox"/> Comentario				STRING(255)
<input type="checkbox"/> CommStateOK				BOOL
<input type="checkbox"/> CommStateOK1				BOOL
<input checked="" type="checkbox"/> Conductimetro				REAL
<input checked="" type="checkbox"/> Control_diurno_RW2_value				UINT
<input checked="" type="checkbox"/> ControlpHRW1				BOOL
<input checked="" type="checkbox"/> ControlpHRW2				BOOL
<input checked="" type="checkbox"/> CosechandoRW1				BOOL
<input checked="" type="checkbox"/> CosechandoRW2				BOOL
<input type="checkbox"/> DILUCION_RW1				BOOL
<input type="checkbox"/> DILUCION_RW2				BOOL
<input type="checkbox"/> DiaSeleccionado1				ARRAY [0..6] OF BOOL
<input type="checkbox"/> DiaSeleccionado2				ARRAY [0..6] OF BOOL
<input checked="" type="checkbox"/> Diario				BOOL
<input checked="" type="checkbox"/> Dir_viento_atm				LREAL
<input checked="" type="checkbox"/> ENTRADA_VALVULA_O2_RW2				REAL
<input checked="" type="checkbox"/> EP_CO2_RW2				UINT
<input type="checkbox"/> EP_CO2_RW2_value				UINT
<input checked="" type="checkbox"/> EVAIRERW1				BOOL
<input checked="" type="checkbox"/> EVAIRERW2				UINT
<input checked="" type="checkbox"/> EVCO2RW1				UINT
<input checked="" type="checkbox"/> EVCosechaRW1				BOOL
<input checked="" type="checkbox"/> EVCosechaRW1_F				BOOL
<input checked="" type="checkbox"/> EVCosechaRW2				BOOL
<input checked="" type="checkbox"/> EVCosechaRW2_F				BOOL
<input checked="" type="checkbox"/> EVMedioRW1				BOOL
<input checked="" type="checkbox"/> EVMedioRW1_F				BOOL
<input checked="" type="checkbox"/> EVMedioRW2				BOOL

Desarrollo de una herramienta SCADA en Codesys para fotobiorreactores industriales

<input checked="" type="checkbox"/>	EVMedioRW2_F			BOOL
<input checked="" type="checkbox"/>	EV_AIRE_RW1_value			UINT
<input checked="" type="checkbox"/>	EV_AIRE_RW2_value			UINT
<input checked="" type="checkbox"/>	EV_CO2_RW1_value			UINT
<input checked="" type="checkbox"/>	EV_CO2_RW2_value			UINT
<input type="checkbox"/>	Error_OD_RW1			LREAL
<input checked="" type="checkbox"/>	Error_OD_RW2			LREAL
<input checked="" type="checkbox"/>	Estado_optimizador_RW1			BOOL
<input checked="" type="checkbox"/>	Estado_optimizador_RW2			BOOL
<input type="checkbox"/>	Et			BOOL
<input type="checkbox"/>	Et_RW1			BOOL
<input type="checkbox"/>	HERCIOS_RW1			REAL
<input type="checkbox"/>	HERCIOS_RW2			REAL
<input type="checkbox"/>	HORA_COSECHADO_RW1			TIME_OF_DAY
<input type="checkbox"/>	HORA_COSECHADO_RW2			TIME_OF_DAY
<input type="checkbox"/>	HORA_DILUCION_RW1			TIME_OF_DAY
<input type="checkbox"/>	HORA_DILUCION_RW2			TIME_OF_DAY
<input checked="" type="checkbox"/>	HighLim_RW1			LREAL
<input checked="" type="checkbox"/>	HighLim_RW2			LREAL
<input checked="" type="checkbox"/>	Hum_rel_atm			LREAL
<input checked="" type="checkbox"/>	ITERACION_RW1			INT
<input type="checkbox"/>	ITERACION_RW2			INT
<input checked="" type="checkbox"/>	InicioCosechadoRW1			BOOL
<input checked="" type="checkbox"/>	InicioCosechadoRW2			BOOL
<input checked="" type="checkbox"/>	InyeccionCO2_RW1			BOOL
<input checked="" type="checkbox"/>	InyeccionCO2_RW2			BOOL
<input type="checkbox"/>	LIM_INF_HP_RW1			REAL
<input type="checkbox"/>	LIM_INF_HP_RW2			REAL
<input checked="" type="checkbox"/>	LIM_INF_NIVEL_RW1			LREAL
<input checked="" type="checkbox"/>	LIM_INF_NIVEL_RW2			LREAL
<input type="checkbox"/>	LIM_INF_OD_RW1			REAL
<input type="checkbox"/>	LIM_INF_OD_RW2			REAL
<input type="checkbox"/>	LIM_SUP_HP_RW1			REAL
<input type="checkbox"/>	LIM_SUP_HP_RW2			REAL
<input checked="" type="checkbox"/>	LIM_SUP_NIVEL_RW1			LREAL
<input checked="" type="checkbox"/>	LIM_SUP_NIVEL_RW2			LREAL
<input type="checkbox"/>	LIM_SUP_OD_RW1			REAL
<input type="checkbox"/>	LIM_SUP_OD_RW2			REAL
<input checked="" type="checkbox"/>	LISTO_ANALIZAR			BOOL
<input checked="" type="checkbox"/>	LowLim_RW1			LREAL
<input checked="" type="checkbox"/>	LowLim_RW2			LREAL
<input checked="" type="checkbox"/>	Manual_RW1			BOOL
<input checked="" type="checkbox"/>	Manual_control_CO2_RW1			BOOL
<input checked="" type="checkbox"/>	Manual_control_CO2_RW2			BOOL
<input type="checkbox"/>	Manual_control_O2_RW1			BOOL
<input type="checkbox"/>	Manual_control_O2_RW2			BOOL
<input checked="" type="checkbox"/>	Manual_value_RW1			LREAL
<input checked="" type="checkbox"/>	NIVEL_COSECHADO_RW1			LREAL
<input checked="" type="checkbox"/>	NIVEL_COSECHADO_RW2			LREAL
<input checked="" type="checkbox"/>	NIVEL_DILUCION_RW1			LREAL
<input checked="" type="checkbox"/>	NIVEL_DILUCION_RW2			LREAL
<input checked="" type="checkbox"/>	NivelRW1			REAL
<input checked="" type="checkbox"/>	NivelRW2			REAL
<input type="checkbox"/>	Nombre			STRING(255)
<input checked="" type="checkbox"/>	ODRW11			LREAL
<input checked="" type="checkbox"/>	ODRW12			LREAL
<input checked="" type="checkbox"/>	ODRW13			LREAL
<input checked="" type="checkbox"/>	ODRW21			LREAL
<input type="checkbox"/>	ODRW22			LREAL
<input checked="" type="checkbox"/>	ODRW23			LREAL
<input type="checkbox"/>	OFF_CAUDAL			BOOL
<input checked="" type="checkbox"/>	Optimizador_RW1			BOOL
<input checked="" type="checkbox"/>	Optimizador_RW2			BOOL
<input checked="" type="checkbox"/>	PID_Fd_RW1			LREAL
<input checked="" type="checkbox"/>	PID_Kp_RW1			LREAL
<input checked="" type="checkbox"/>	PID_RW1_Output			LREAL
<input checked="" type="checkbox"/>	PID_RW1_PWM_Input			UINT
<input checked="" type="checkbox"/>	PID_Td_RW1			LREAL
<input checked="" type="checkbox"/>	PID_Ti_RW1			LREAL

<input type="checkbox"/>	Pruebas_1			BOOL
<input type="checkbox"/>	Pruebas_2			BOOL
<input type="checkbox"/>	Q_Set_OD_RW1			LREAL
<input checked="" type="checkbox"/>	Q_Set_OD_RW2			LREAL
<input type="checkbox"/>	RESET_RW1			BOOL
<input type="checkbox"/>	RESET_RW2			BOOL
<input checked="" type="checkbox"/>	Rad_PAR_atm			LREAL
<input checked="" type="checkbox"/>	Rad_atm			LREAL
<input checked="" type="checkbox"/>	ReactivarEvapRW1			BOOL
<input checked="" type="checkbox"/>	ReactivarEvapRW2			BOOL
<input checked="" type="checkbox"/>	Repo_EVMedioRW1			BOOL
<input checked="" type="checkbox"/>	Repo_EVMedioRW2			BOOL
<input checked="" type="checkbox"/>	ReposicionEvapRW1			BOOL
<input checked="" type="checkbox"/>	ReposicionEvapRW2			BOOL
<input checked="" type="checkbox"/>	Reset_emergencia			BOOL
<input checked="" type="checkbox"/>	STOP			BOOL
<input checked="" type="checkbox"/>	Selectorcontrol_RW1			BOOL
<input checked="" type="checkbox"/>	Selectorcontrol_RW2			BOOL
<input checked="" type="checkbox"/>	Set_CaudalCO2_RW1			LREAL
<input checked="" type="checkbox"/>	Set_CaudalCO2_RW1_ESC			REAL
<input checked="" type="checkbox"/>	Set_CaudalCO2_RW2			LREAL
<input type="checkbox"/>	Set_CaudalCO2_RW2_ESC			REAL
<input type="checkbox"/>	Set_OD_RW1			LREAL
<input checked="" type="checkbox"/>	Set_OD_RW2			LREAL
<input checked="" type="checkbox"/>	Setpoint_nivel_RW1			LREAL
<input checked="" type="checkbox"/>	Setpoint_nivel_RW1_manual			LREAL
<input checked="" type="checkbox"/>	Setpoint_nivel_RW2			LREAL
<input checked="" type="checkbox"/>	Setpoint_nivel_RW2_manual			LREAL
<input checked="" type="checkbox"/>	Setpoint_opt_nivel_RW1			LREAL
<input checked="" type="checkbox"/>	Setpoint_opt_nivel_RW2			LREAL
<input checked="" type="checkbox"/>	Subir_nivel_RW1			BOOL
<input checked="" type="checkbox"/>	Subir_nivel_RW2			BOOL
<input checked="" type="checkbox"/>	Switch_manual_CO2_RW2			BOOL
<input checked="" type="checkbox"/>	TempRW1			LREAL
<input checked="" type="checkbox"/>	TempRW2			LREAL
<input checked="" type="checkbox"/>	TempSuelo1			LREAL
<input checked="" type="checkbox"/>	TempSuelo2			LREAL
<input checked="" type="checkbox"/>	Temp_atm			LREAL
<input type="checkbox"/>	Tiempo_Muestreo			BOOL
<input type="checkbox"/>	Todo_Nada_CO2_RW1			BOOL
<input checked="" type="checkbox"/>	Todo_Nada_CO2_RW2			BOOL
<input checked="" type="checkbox"/>	Turbidimetro			LREAL
<input type="checkbox"/>	VERIF_RW1			BOOL
<input type="checkbox"/>	VERIF_RW2			BOOL
<input checked="" type="checkbox"/>	VariableInicioCos1			BOOL
<input checked="" type="checkbox"/>	VariableInicioCos2			BOOL
<input checked="" type="checkbox"/>	Vel_Aspas_RW1			REAL
<input checked="" type="checkbox"/>	Vel_Aspas_RW2			REAL
<input checked="" type="checkbox"/>	Vel_viento_atm			LREAL
<input checked="" type="checkbox"/>	Wacht_Dog			BOOL
<input checked="" type="checkbox"/>	cnt			INT
<input checked="" type="checkbox"/>	hora_real			UINT
<input checked="" type="checkbox"/>	kla			LREAL
<input type="checkbox"/>	kla1			LREAL
<input checked="" type="checkbox"/>	pHRW11			LREAL
<input checked="" type="checkbox"/>	pHRW12			LREAL
<input checked="" type="checkbox"/>	pHRW13			LREAL
<input checked="" type="checkbox"/>	pHRW21			LREAL
<input checked="" type="checkbox"/>	pHRW22			LREAL
<input checked="" type="checkbox"/>	pHRW23			LREAL
<input checked="" type="checkbox"/>	pH_RW1			REAL
<input checked="" type="checkbox"/>	pH_RW2			REAL
<input checked="" type="checkbox"/>	pH_filtrado_RW1			LREAL
<input type="checkbox"/>	pH_filtrado_RW2			LREAL
<input checked="" type="checkbox"/>	upid_a			INT
<input type="checkbox"/>	upid_a1			LREAL

c) Lista de símbolos de variables globales, "Global".

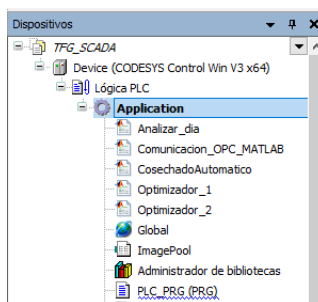
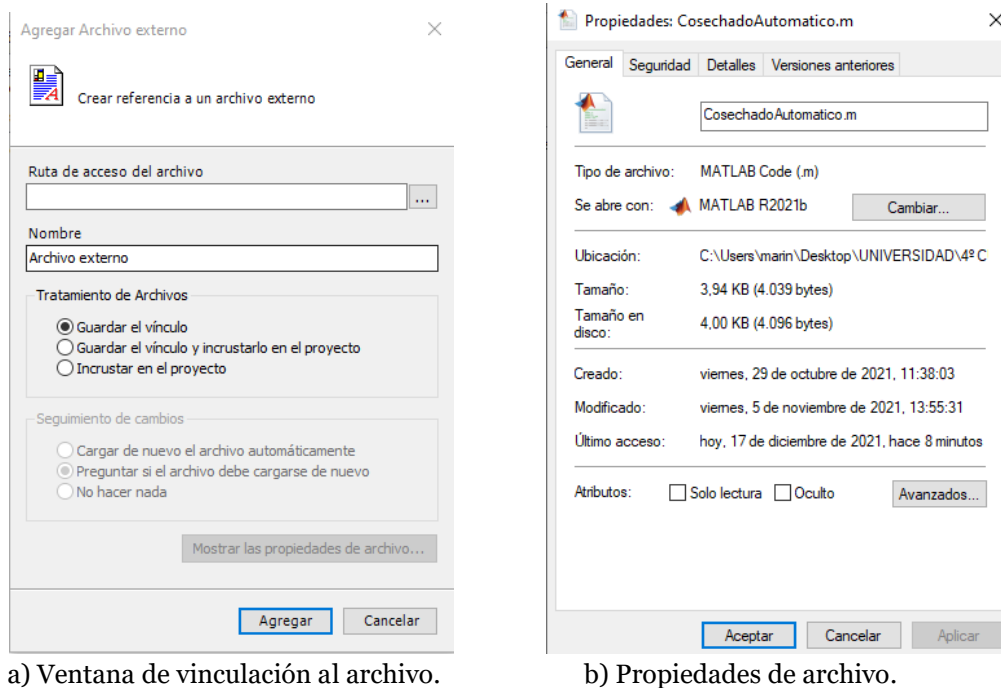
Figura 3.47. Lista de símbolos.

3.12. Conexión con Matlab

El cometido de los servidores OPC es comunicar entre sí diversos programas o aplicaciones para que interactúen, enviando y recibiendo datos de uno a otro, por ejemplo. Como ya se hizo mención, es necesario una comunicación con el programa Matlab, donde están las funciones o scripts que se encargan de ciertos tratamientos de los datos procedentes de los PLCs.

En LabVIEW, esta comunicación era directa desde la plataforma, pero con Codesys no es posible. La única función que permite este entorno de desarrollo es poder recoger estos archivos externos procedentes de otros programas en su listado de archivos de proyecto para que sean más fácilmente accesibles (Figura 3.48.c). De esta manera, no es necesario buscar la ubicación de los archivos en el ordenador ni el ejecutable del programa en sí, sino que basta con pinchar sobre el archivo y este se abre directamente con el programa correspondiente.

Esto se logra creando un objeto “Archivo externo”, seleccionando el archivo a añadir y el tratamiento que se hará al archivo (Figura 3.48.a), en este caso se han incrustado en el proyecto. Luego, en “Mostrar propiedades de archivo”, Figura 3.48.b, se elige el programa con el que se ejecutará de entre los instalados en el PC, para este caso no es otro que Matlab.



c) Localización del archivo dentro del proyecto.

Figura 3.48. Proceso de creación de un objeto *Archivo Externo*.

Retomando el asunto inicial, la solución para el problema pasa por hacer la comunicación a través del servidor. Para ello, una vez el servidor tiene cargadas las variables compartidas, se entra en Matlab, en APPS y se busca una app llamada “OPC Data Access Explorer” (Figura 3.49).

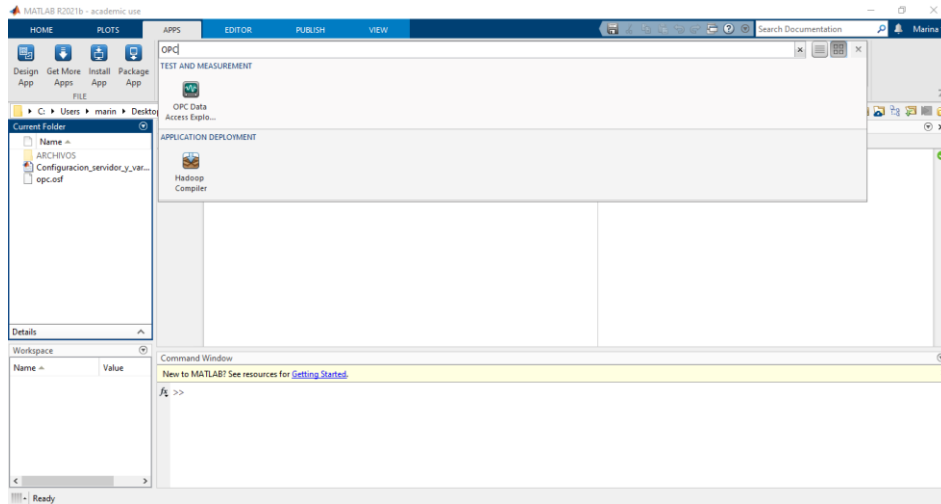
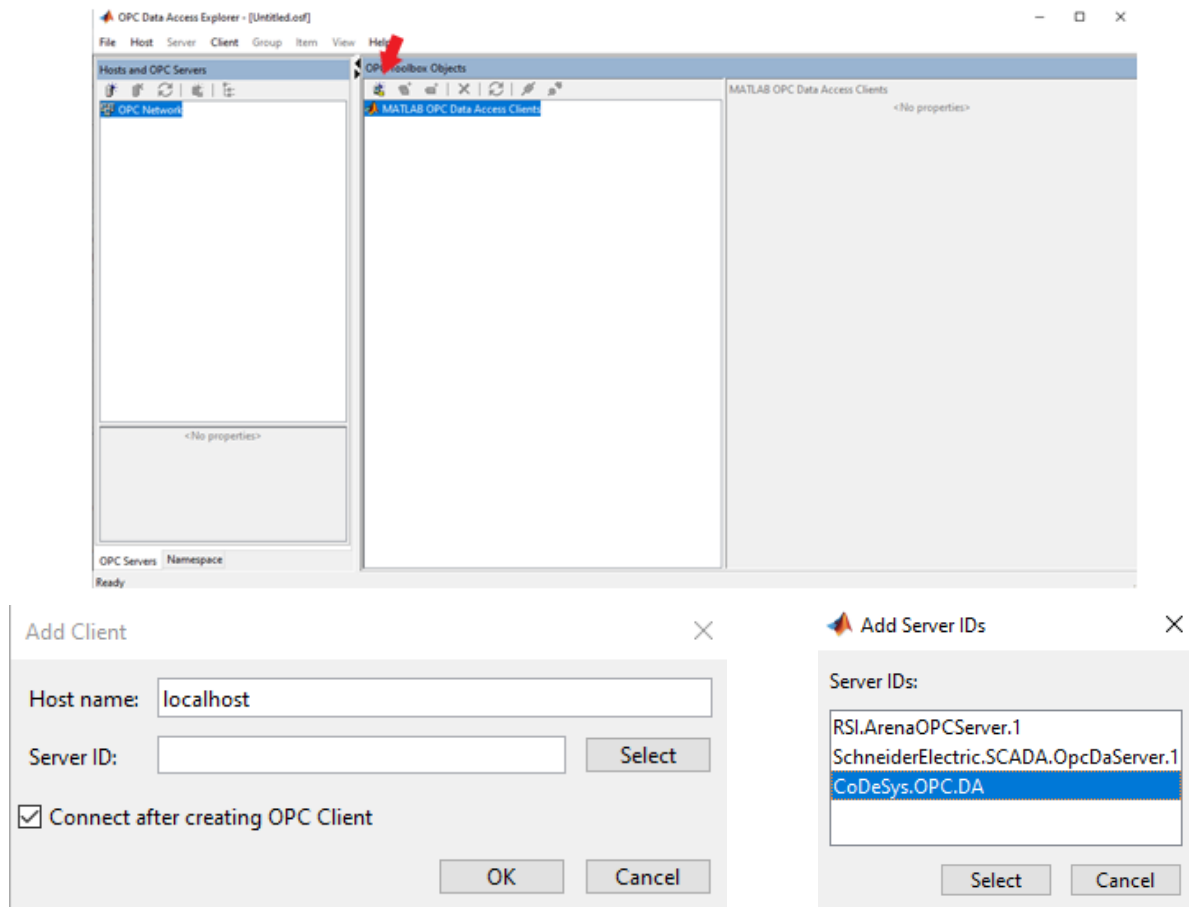
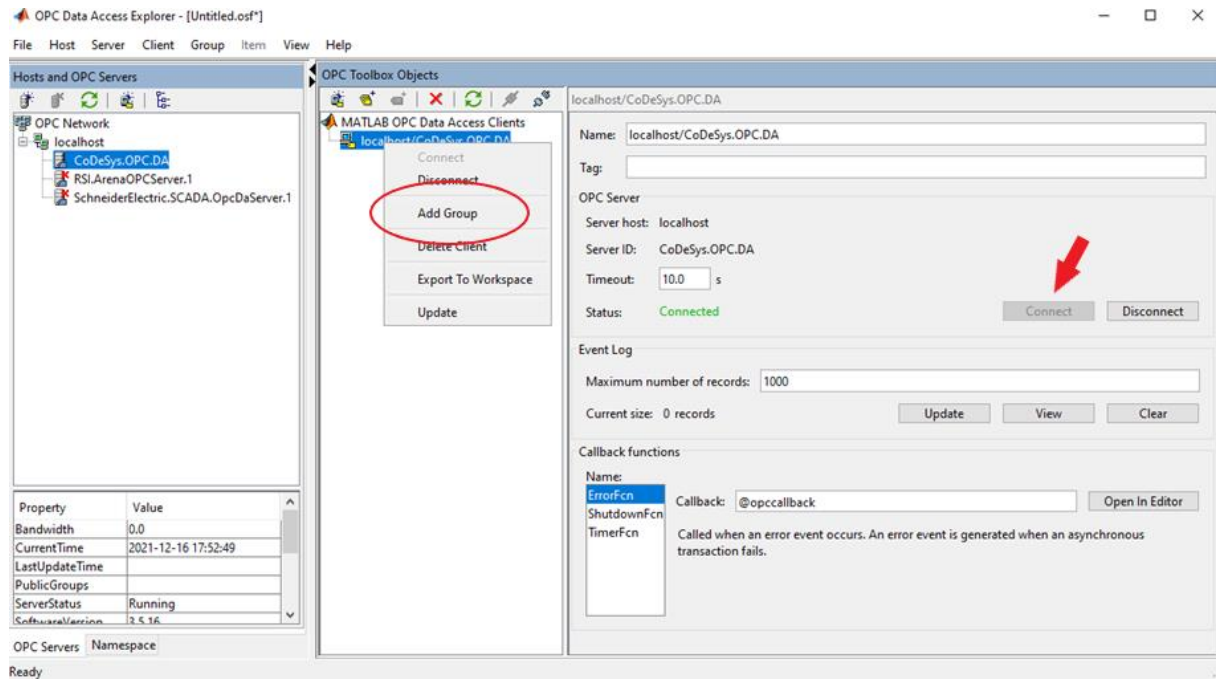


Figura 3.49. Acceso a la aplicación OPC de Matlab.

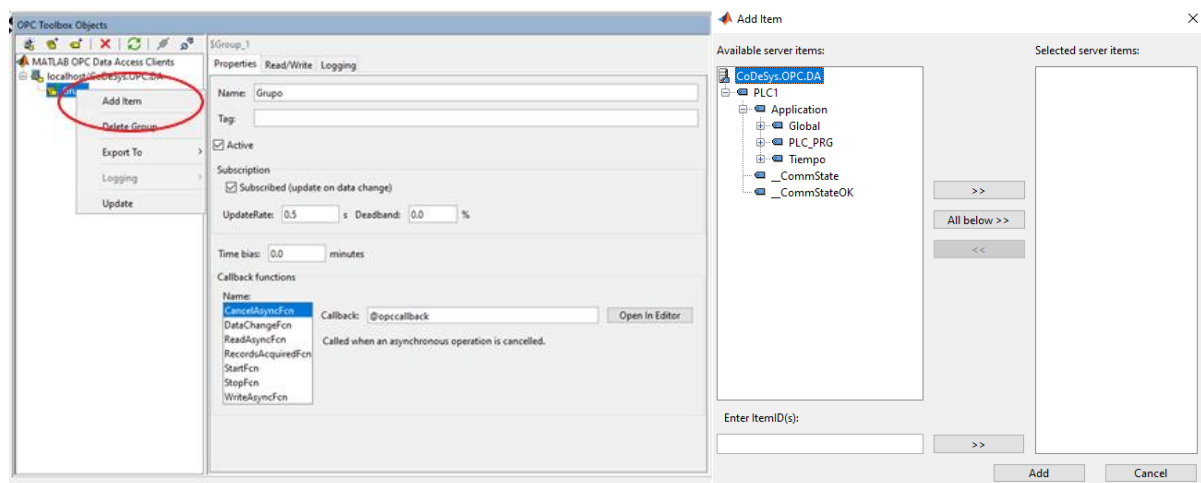
Aparecerá una ventana (Figura 3.50) donde se creará y configurará el Cliente OPC y luego, dentro de este, los Grupos con los Items, es decir, con las variables del servidor que se precisen.



1º. Creación del OPC Cliente.



2º. Conectar Servidor y Añadir un grupo.



3º. Añadir Items al grupo.

Figura 3.50. Pasos de la configuración de la conexión con OPC.

Los pasos para ello son los siguientes: en primer lugar, se crea el cliente. Para ello se selecciona el servidor ID CoDeSys.OPC.DA como se muestra en la Figura 3.50.1º. Luego se conecta el servidor y se crea un grupo donde almacenar las variables (Figura 3.50. 2º). Por último, se añaden los Item a cada grupo que se cree siguiendo los pasos descritos en la Figura 3.50.3º.

Es digno de mención que cuando se inicia la comunicación con el servidor aparece en pantalla un mensaje indicando los días restantes antes de que la licencia limitada del servidor expire. Cuando esto ocurra habrá que solicitar una nueva licencia de uso temporal como se indica en el capítulo “Anexos”.

Dentro de cada grupo, se puede acceder a sus Item y ver las características de estos, como el tipo de dato, los derechos de acceso o el valor que tienen asociado. En la Figura 3.51, se muestra

el ejemplo de la variable ANO_ACTUAL de Codesys que hace referencia al año presente, cuyo valor es igual a 2022, tiene derechos de acceso de lectura y es tipo INT16.

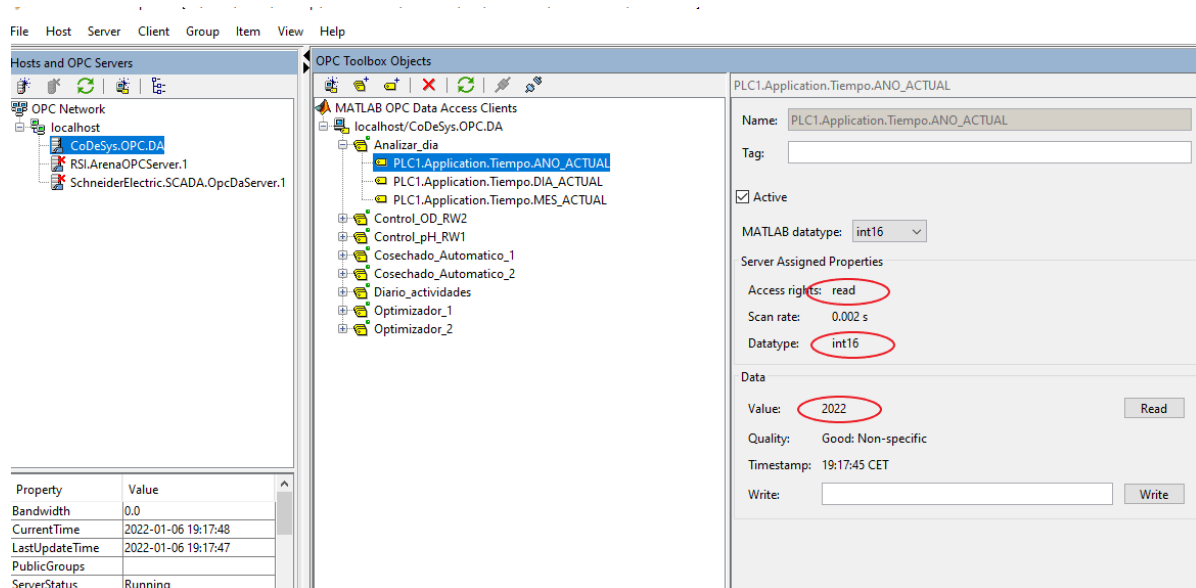


Figura 3.51. Ejemplo Item ANO_ACTUAL.

Entre las propiedades que se pueden configurar, las más destacadas son el tiempo “Timeout” y el “Update Rate”, que hacen referencia al tiempo que tarda en establecerse la comunicación con el servidor y la tasa de actualización de los datos de cada uno de los grupos creados. Se establecieron 20 s para “Timeout” y 0,4 s para “Update Rate”, para evitar error de conexión y por ser la tasa de actualización de Codesys, respectivamente.

Para mantener la información lo más estructurada posible se han creado 9 grupos, uno por cada función de Matlab a ejecutar: dos para las labores de los optimizadores, otros dos para el cosechado automático de los reactores, uno para el control de pH del reactor RW1, uno para el control de O2 del reactor RW2, otro para el diario de actividades, otro para el registro del día anterior y un último para labores relacionadas con la parada del SCADA. Dentro de cada uno de ellos se han añadido las variables pertinentes.

Una manera de trasladar esta información a un script de Matlab donde manipular los datos, es haciendo clic en “File”, en “Export Clients to” y elegir de entre las tres opciones posibles “MatLAB Code File”. Se creará una función que recoge todo el proceso anterior, pero en lenguaje de texto de Matlab.

Una vez en este formato, se manipula el texto para hacer la lectura de los datos de entrada, tratarlos en las funciones correspondientes y escribir los datos de salida de dichas funciones cíclicamente. Para conseguirlo se generó un script con el código mostrado en la Figura 3.52.a, con él se llama a la función anteriormente mencionada y que se refleja en la Figura 3.52.b, donde se leen y/o escriben las variables del servidor. Luego, estos datos pasan por las funciones y devuelven unos valores de salida que serán escritos en el servidor a través de la misma función.

Todo este proceso tendrá lugar únicamente mientras la variable STOP esté desactivada. Cuando no lo esté, se producirá la lectura inicial, pero no se ejecutarán las funciones ni se

escribirá ninguna variable en el servidor hasta que se vuelva a desactivar la variable.

```
tratamiento_matlab.m
1 clear
2 clc
3
4 while true
5     %% Inicialización de las variables de salida de los scripts
6     escritura=0; % Variable de control de escritura de datos
7     Depth_sp_RW1=0;
8     Depth_sp=0;
9     Lim_h_RW1=0;
10    Lim_d_RW1=0;
11    Lim_h=0;
12    Lim_d=0;
13    setpoint_nivel_rw1_manual=0;
14    setpoint_nivel_rw2_manual=0;
15    y_k=0;
16    it=0;
17    m=0;
18    Ju=0;
19    Ju_lim=0;
20    J=0;
21    kla_aux=0;
22    Error_ODRW1=0;
23    upid_a=0;
24    Q_setpoint=0;
25    Diariook=0;
26
27    cnt=0;
28    %% Lectura de las variables
29    [Vector_actividades,Diario,DiaSeleccionado1,HoraCosechado,HoraDilucion,HoraReal,NivelCosechado,NivelDilucion,NivelRW1,Aceptar_SP,SP_Manua
30    if STOP==true
31        continue
32    end
33    %% Ejecución de las funciones de Matlab
34    % Diario de actividades
35    [Diariook]=Diario_actividades (Vector_actividades,Diario);
36    % Optimizador 1
37    [Depth_sp_RW1,Lim_h_RW1, Lim_d_RW1,cnt]=Optimizador_1 (Temp_RW1,hour,rad,Text,Wind,Hum,Tsoil,NivelRW1);
38    % Optimizador 2
39    [Depth_sp,Lim_h,Lim_d]=Optimizador_2 (Temp_RW2, hour, rad,Text,Wind,Hum,Tsoil,nivel,Et);
40    %Cosechado automático 1
41    [setpoint_nivel_rw1_manual]=Cosechado_Automatico(Dia,DiaSeleccionado1,HoraDilucion,HoraCosechado, HoraReal, NivelDilucion, NivelRW1, Nive
42    %Cosechado automático 2
43    [setpoint_nivel_rw2_manual]=Cosechado_Automatico(Dia,DiaSeleccionado2,HoraDilucion2,HoraCosechado2, HoraReal, NivelDilucion2, NivelRW2, NI
44    %Control pH_RW1
45    [y_k,it,m,Ju,Ju_lim,J]=Control_pH_RW1(pH,setpoint_pH,rad,Input,hora,Inicializar);
46    %Control OD_RW2
47    [kla_aux,Error_ODRW1,upid_a,Q_setpoint]=Control_OD_RW2(Q_O2_RW2,Nive1RW2,Q_manual,OD_FosoRW2,OD_FosoRW1,SetPoint_OD,Tiempo_Muestreo,Manua
48    %Analizar dia
49    Analizar_dia(day,month,year,Listo_analizar);
50
51    %% Escritura de datos
52    escritura=1;
53    opc_variables(escritura,Depth_sp_RW1,Depth_sp,Lim_h_RW1, Lim_d_RW1,cnt,Lim_h,Lim_d,setpoint_nivel_rw1_manual,setpoint_nivel_rw2_manual,y_k
54    pause(0.4)
55    end
56
57
58
```

a) Código script *tratamiento_matlab*.

```
opc_variables.m
1 function [Vector_actividades,Diario,DiaSeleccionado1,HoraCosechado,HoraDilucion,HoraReal,NivelCosechado,NivelDilucion,NivelRW1,Aceptar_SP,SP_
2
3     % Create the OPCDA object - daobj1
4     daobj1 = opcda('localhost', 'CoDeSys.OPC.DA');
5     set(daobj1, 'Timeout', 20);
6     connect(daobj1);
7
8     % Create the Group object - grp1
9     grp1 = addgroup(daobj1, 'Diario_actividades');
10    set(grp1, 'LogFileName', 'opcdata.log');
11
12    % Create the Item object - itm1
13    itm1 = additem(grp1, 'PLC1.Application.PLC_PRG.Vector_actividades');
14    set(itm1, 'DataType', 'char');
15
16    Vector_actividades=read (itm1);
17    Vector_actividades=Vector_actividades.Value;
18
```

1º. Creación del cliente OPC, creación de un grupo, creación de un Item y asignación del Item a una variable que será salida de la función.

```
216
217 % Create the Item object - itm30
218 itm30 = additem(grp4, 'PLC1.Application.Global.Setpoint_opt_nivel_RW1');
219 set(itm30, 'DataType', 'double');
220
221 % Create the Item object - itm31
222 itm31 = additem(grp4, 'PLC1.Application.Global.cnt');
223 set(itm31, 'DataType', 'int16');
224
225 % Create the Item object - itm32
226 itm32 = additem(grp4, 'PLC1.Application.Global.LowLim_RW1');
227 set(itm32, 'DataType', 'double');
228
229 % Create the Item object - itm33
230 itm33 = additem(grp4, 'PLC1.Application.Global.HighLim_RW1');
231 set(itm33, 'DataType', 'double');
232
```

2º. Creación de Items que serán asignados a variables de entrada de la función y que serán escritos con los resultados obtenidos de las funciones del script *tratamiento_matlab*.

```
453
454 %% Escritura
455 if escritura==1
456 write(itm30,Depth_sp_RW1)
457 write(itm40,Depth_sp)
458 write(itm33,Lim_h_RW1)
459 write(itm32,Lim_d_RW1)
460 write(itm31,cnt)
461 write(itm43,Lim_h)
462 write(itm44,Lim_d)
463 write(itm11,setpoint_nivel_rw1_manual)
464 write(itm23,setpoint_nivel_rw2_manual)
465 write(itm51,y_k)
466 write(itm52,it)
467 write(itm55,m)
468 write(itm56,Ju)
469 write(itm57,Ju_lim)
470 write(itm58,J)
471 write(itm65,kla_aux)
472 write(itm66,Error_ODRW1)
473 write(itm67,upid_a)
474 write(itm69,Q_setpoint)
475 write(itm1_a,Diarioook)
476 end
477
478 if nargout > 0
479     out = [daobj1];
480 end
481
```

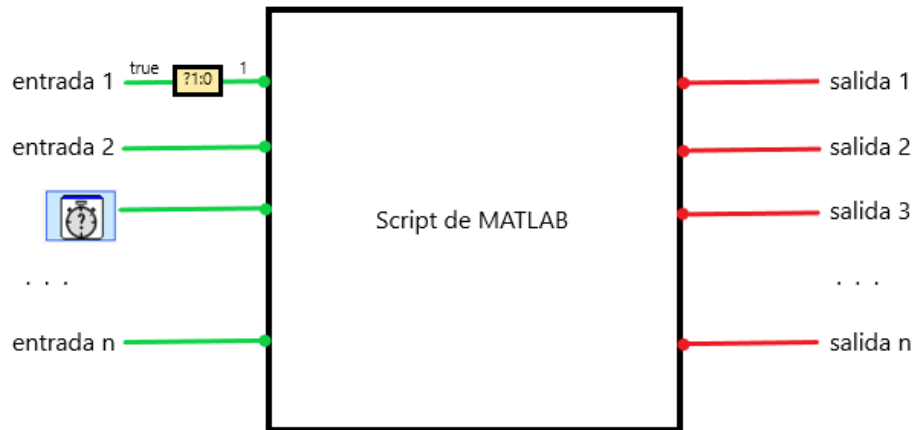
3º. Escritura de los Items de pto 2º y salida de la función.

b) Fragmentos de código de la función *opc_variables*.

Figura 3.52. Códigos de Matlab.

A excepción del diario de actividades, el resto de funciones se han construido estableciendo las variables de entrada y de salida estipuladas en LabVIEW y reutilizando la mayor parte del código de los scripts llamados, como se muestra en el esquema explicativo de la figura 3. 53.a. El registro del día anterior ya se encontraba en formato función, pero el resto de script han tenido que ser modificados para que estuviese en este formato.

El diario de actividades no sigue esta dinámica puesto que no se obtenía a través de Matlab en el SCADA anterior. Por ello se ha tenido que desarrollar un código para su obtención a través de dicho programa. Este código se ve reflejado en la Figura 3.53, en el apartado b.



```
function [salida 1, salida 2, salida 3, ..., salida n] = nombre_funcion (entrada 1, entrada 2, entrada 3, ..., entrada n)  
Codigo de script de MATLAB  
...  
end
```

a) Implementación genérica de una función de Matlab.

```
1 function [Diariook]=Diario_actividades (Vector_actividades,Diario)  
2 if Diario=true  
3     fid=fopen('ficherodatos.txt','at+');  
4     actividad=['Fecha/Hora:' Vector_actividades{1,1} ' Nombre:' Vector_actividades{1,2} ' Comentario:' Vector_actividades{1,3} newline];  
5  
6     for i=1:size(actividad,1)  
7         fprintf(fid, actividad(i,:));  
8     end  
9     fclose(fid);  
10    Diariook=false;  
11 else  
12    Diariook=Diario;  
13 end  
14  
15 end  
16 |
```

b) Código de la función Diario_actividades.

Figura 3.53. Código de las funciones de Matlab.

3.13. SCADA resultante

La mejor forma de explicar el funcionamiento del SCADA resultante de este proyecto es a partir de su panel de control y las acciones que el usuario puede tomar parte en él.

Antes de nada, hay que poner en línea el proyecto, para ello se realiza una compilación que

confirme que no existen errores de código y luego se pone en ejecución el proyecto a través de Iniciar la sesión. A continuación, se pulsa Inicio y con ello el proyecto estará en funcionamiento. Para detenerlo se pulsa Parada y para salir de la ejecución Salida. Estos comandos están disponibles en el panel de herramientas que aparece en la parte superior del entorno de desarrollo de Codesys o a través de combinaciones de teclado:

- Iniciar la sesión: Alt+F8 o icono Rueda con círculo.
- Inicio: F5 o icono Triángulo.
- Parada: Mayúscula +F8 o icono Cuadrado.
- Salida: Ctrl+F8 o icono Rueda con cruz.

Adicionalmente, Iniciar la sesión y Salida están disponibles en la pestaña “En línea”.

Cada vez que se ejecute el proyecto y haya sufrido modificaciones aparecerá una ventana con las opciones de ejecución disponibles (Figura 3.54), en tal caso se deberá elegir Iniciar sesión con descarga.

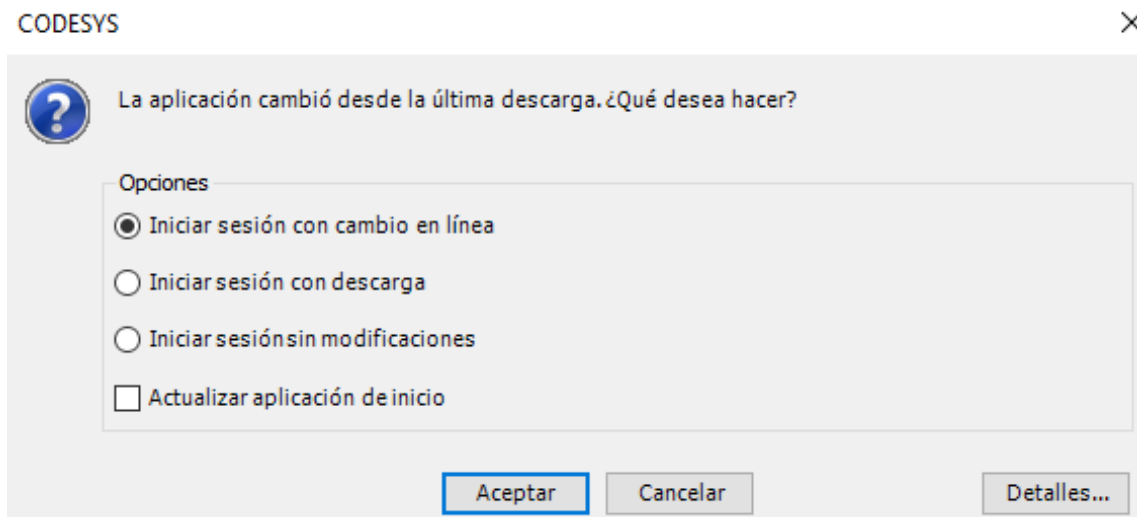


Figura 3.54. Opciones de ejecución de proyecto.

Los botones “Visualización RW1”, “Visualización RW2” y “Resumen de variables” se asocian a las pestañas mostradas en la Figura 3.55, donde se visualizan los valores instantáneos recogidos de los sensores de los reactores. Además, en las dos primeras se pueden establecer los valores de setpoint de pH y Hz para cada reactor, haciendo clic sobre su correspondiente recuadro gris. Con este último valor la POU “PLC_PRG” calcula directamente la velocidad de las aspas de impulsión de los reactores.

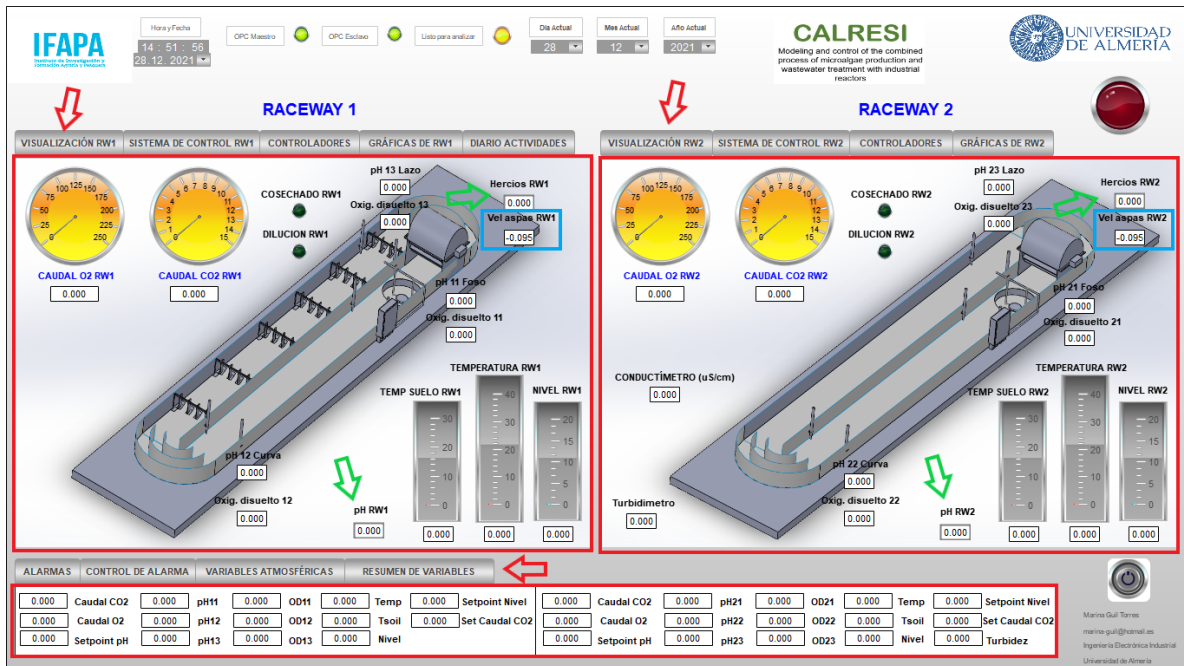


Figura 3.55. Explicación 1. Acceso a las pestañas “RW1”, “RW2” y “Resumen_de_variables” (rojo). Introducción de setpoint de variables (verde). Visualización de resultados (azul).

Por su parte, el botón “Variables atmosféricas” hace las mismas funciones, pero con los datos recogidos de la estación meteorológica. La Figura 3.56 expone este hecho:

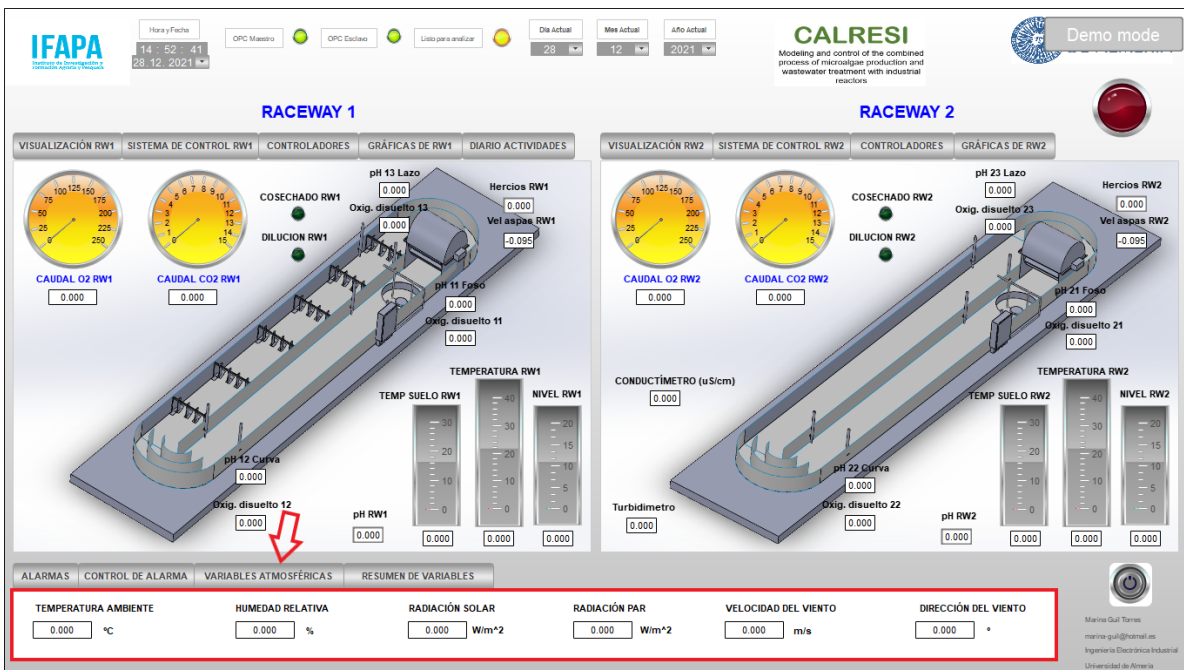
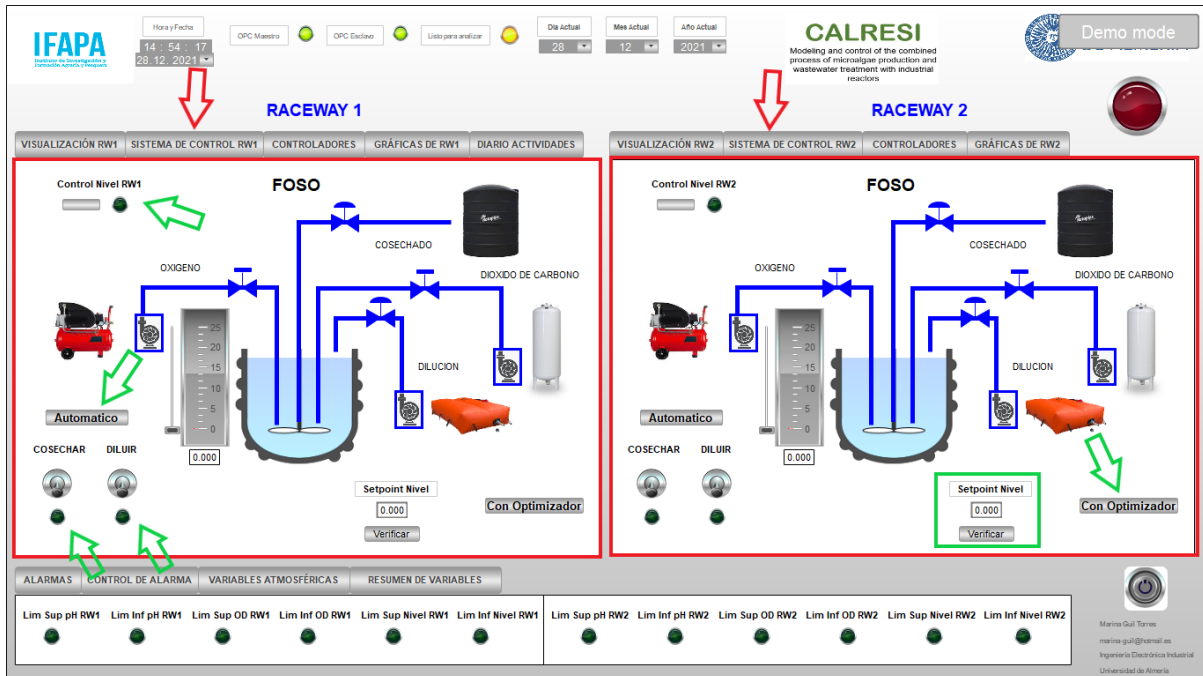


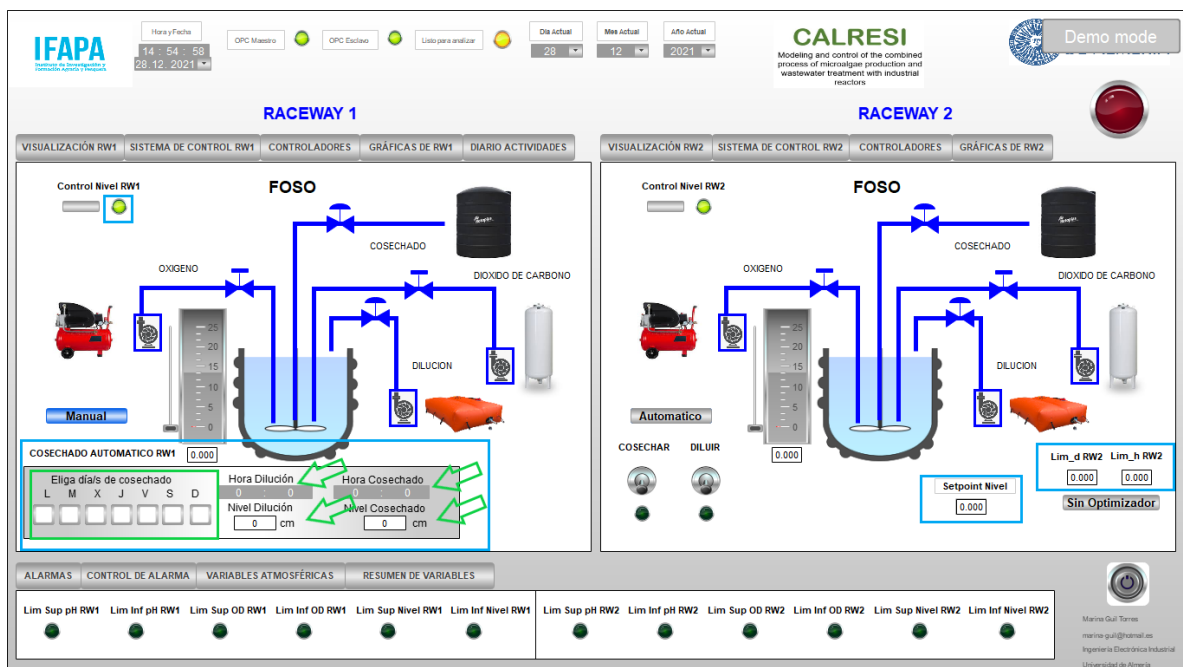
Figura 3.56. Explicación 2. Acceso a la pestaña “Variables_atmos” (rojo).

Los botones “Sistema de Control RW1” y “Sistema de Control RW2” exponen las pestañas representadas en la Figura 3.56, donde se realiza el control del nivel de los reactores mediante: inyección de O₂, inyección de CO₂, inyección de agua (dilución) y extracción de producto (cosechado). Estos dos últimos pueden controlarse manualmente o de forma automática. En

modo manual, mediante unos interruptores de cosechar y diluir (Figura 3.56.a); y en modo automático (Figura 3.56.b), con la imposición de los días, horas y nivel de cosechado y dilución que se quieren lograr. En modo manual, también cabe la posibilidad de introducir o no un optimizador que actúe a la hora de alcanzar el setpoint de nivel impuesto por el usuario.



a) Acceso a las pestañas “SIS_RW1” y “SIS_RW2” (rojo). Introducción de setpoint de variables y activación/ desactivación del control de nivel, del modo de operación y del modo manual (verde).

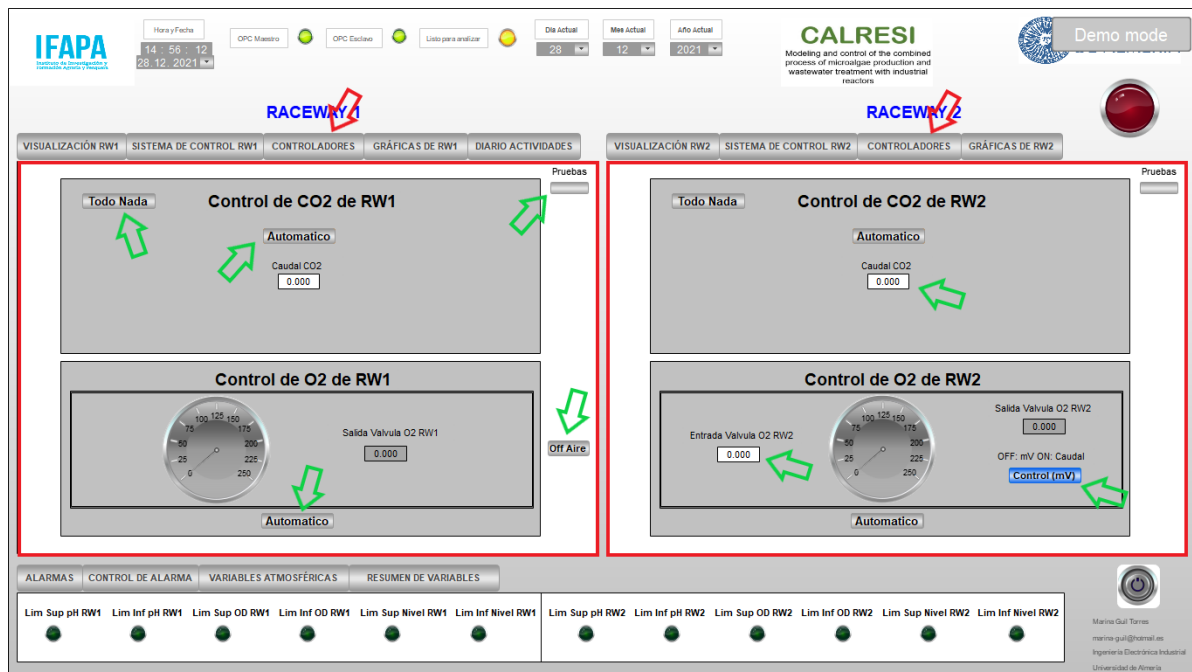


b) Control automático, resultado de activar el botón de control de nivel y del optimizador.

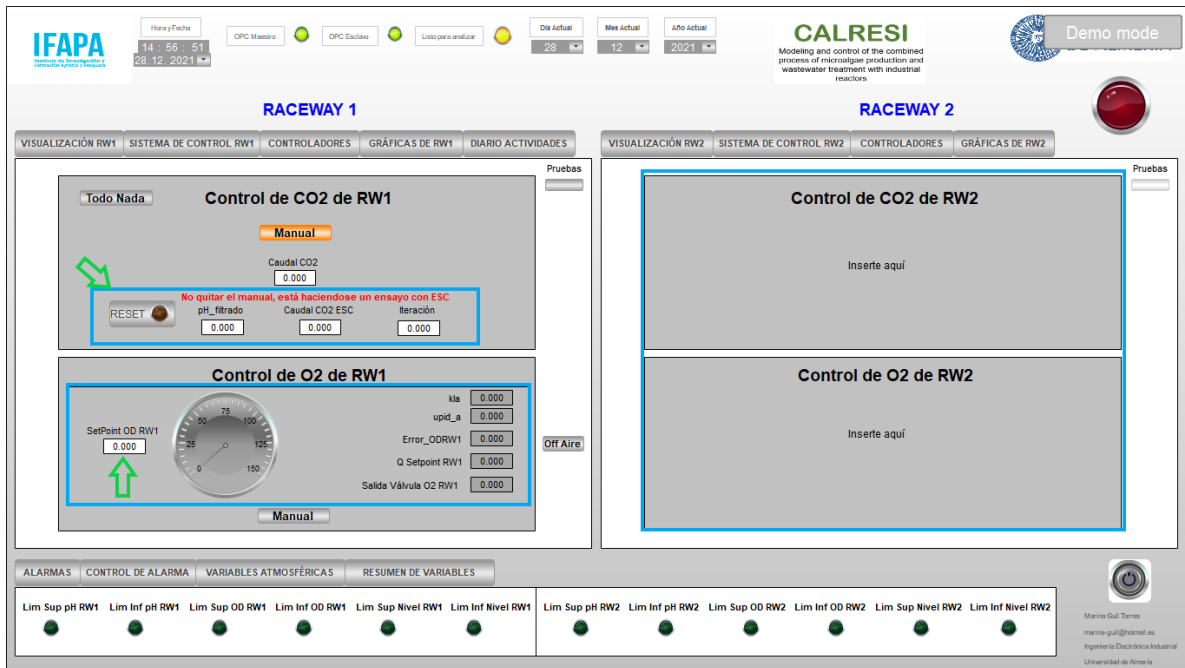
Figura 3.57. Explicación 3.

Los botones “Controladores” se vinculan a aquellas pestañas donde se hacen pruebas con diversos métodos de control para la inyección de CO₂ y de O₂ (Figura 3.58). Por defecto, existen para el control de CO₂ dos estados diferentes: un ensayo y un control Todo nada. Además, como ocurría con el control de nivel de los reactores, se cuenta con dos modos de trabajo, manual o automático. Para ambos simplemente se debe introducir el caudal de CO₂ que se desea alcanzar y el SCADA se encarga de hacer el tratamiento correspondiente, según proceda.

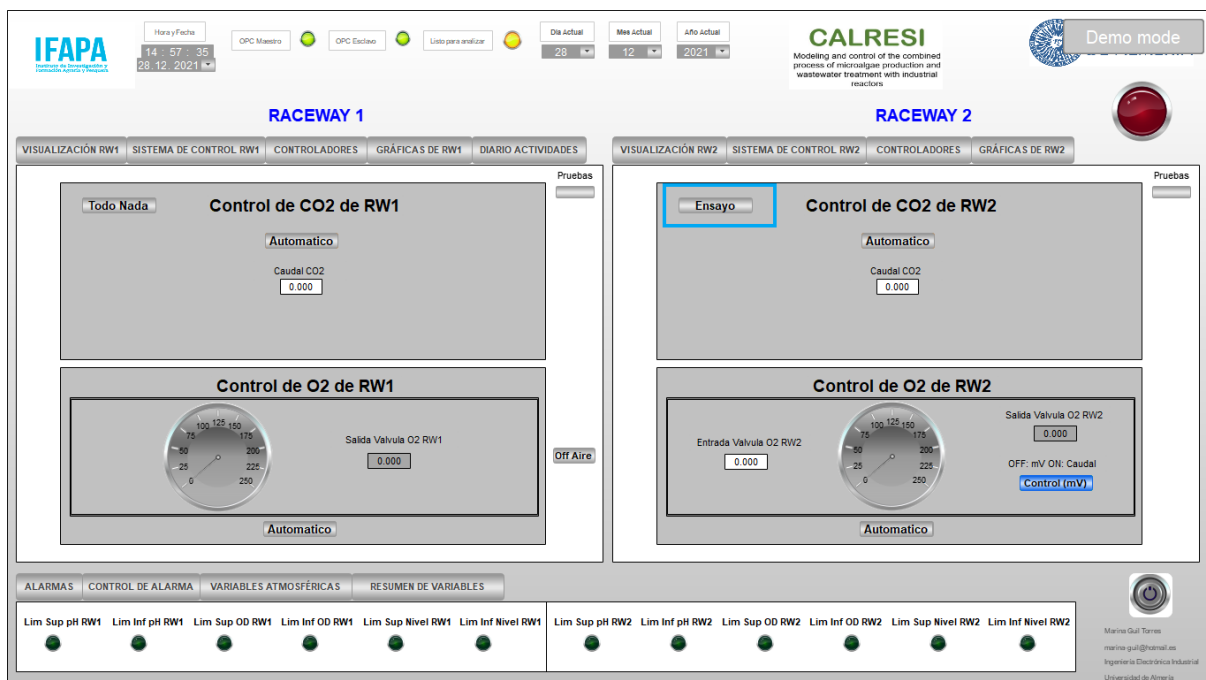
En el caso del O₂ solo se distinguen los dos modos de trabajo (manual/automático). Para el modo manual se introduce un valor de entrada a la electroválvula del O₂ y se pulsa el botón “Control” para que ejecuten las acciones pertinentes (Figura 3.58.a); y para el modo automático se introduce un setpoint de OD del reactor. Asimismo, se ha creado una pestaña vacía (Figura 3.58.b), accesible a través del botón “Pruebas” donde los usuarios pueden probar sus controladores experimentales sin perturbar los ya instaurados.



a) Acceso a las pestañas “CONTROL_RW1” y “CONTROL_RW2” (rojo). Introducción de setpoint de variables y activación/ desactivación del modo de operación y del controlador (verde).



b) Modo automático del control de CO₂ y O₂ y controlador Todo Nada del CO₂ en el reactor 1. Pestaña vacía para insertar nuevos controladores en el reactor 2.



c) Controlador Ensayo del CO₂ en el reactor 2.

Figura 3.58. Explicación 4.

Aunque se ha diseñado el panel suponiendo el control de ambas variables en los dos reactores, como ya se sabe, solo se ejecuta el tratamiento de datos en un caso para cada uno de ellos: control de CO₂ en el raceway 1 y control de O₂ en el raceway 2. Se ha hecho de esta manera para que en el caso de querer incorporar estos controladores en algún momento se amenice el

trabajo del usuario al no tener que diseñar la pestaña desde cero ni declarar más variables. Solo sería necesario introducirlos en el listado de símbolos para que se compartan en el servidor.

Los botones “Grafica de RW1” y “Grafica de RW2” muestran una pestaña donde se representan gráficamente y en función del tiempo algunas variables de los reactores y de las condiciones ambientales: valores de pH, de oxígeno disuelto, de O₂, CO₂ y nivel, y temperatura, radiación, velocidad del viento y humedad. Para el raceway 2 también se incluye la turbidez. En la leyenda disponible en el lado izquierdo se especifica cada una de las variables presentes en la gráfica y permite que estas sean activadas o desactivadas de la representación pulsando en su casilla de verificación (Figura 3.59).

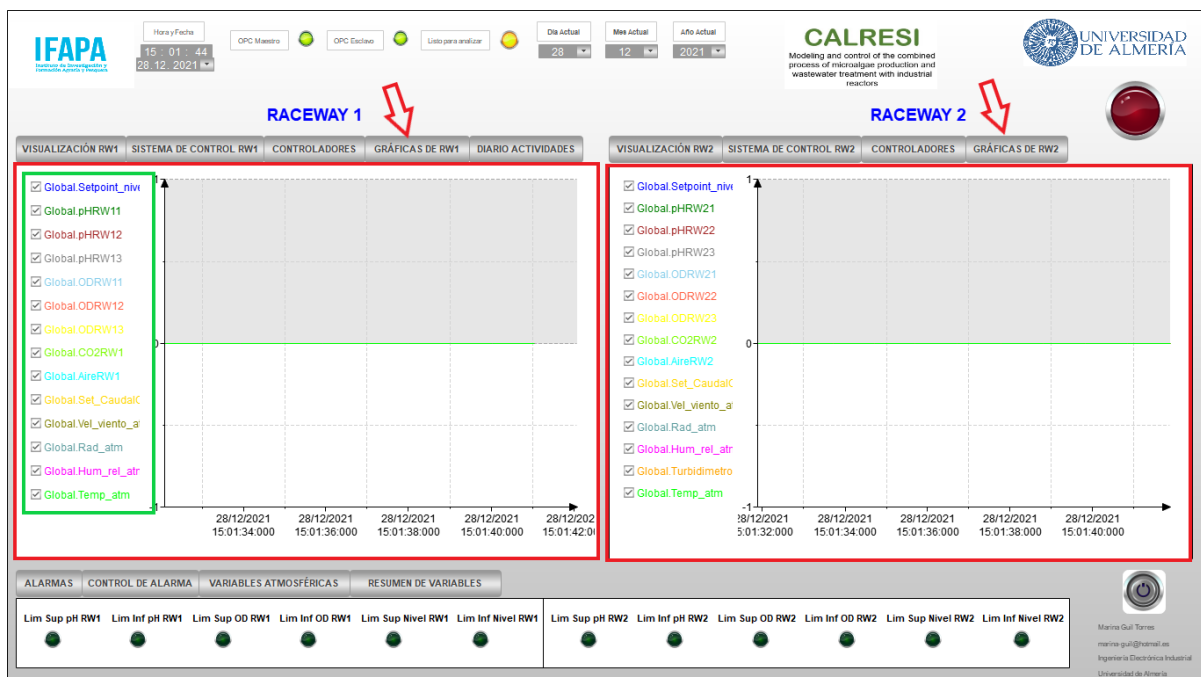


Figura 3.59. Explicación 5. Acceso a las pestañas “GRAFICA_RW1” y “GRAFICA_RW2” (rojo). Activación/desactivación de las variables visibles en las representaciones gráficas.

El botón “Diario de actividades” accede a una pestaña donde los usuarios pueden dejar mensajes sobre el estado de los reactores, posibles incidencias, etc (Figura 3.60). Basta, como indica la siguiente figura, con hacer una identificación del usuario e introducir un comentario. Una vez hecho se pulsa el botón añadir entrada y queda registrado en un archivo .txt junto con el instante en el que se añade la entrada.

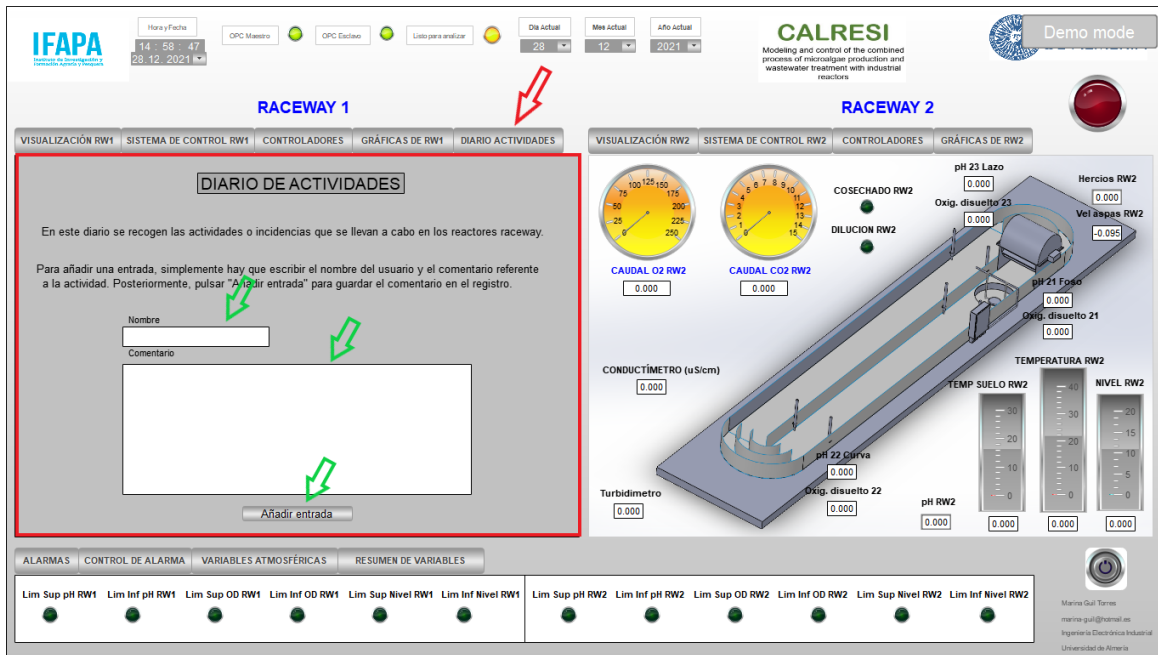
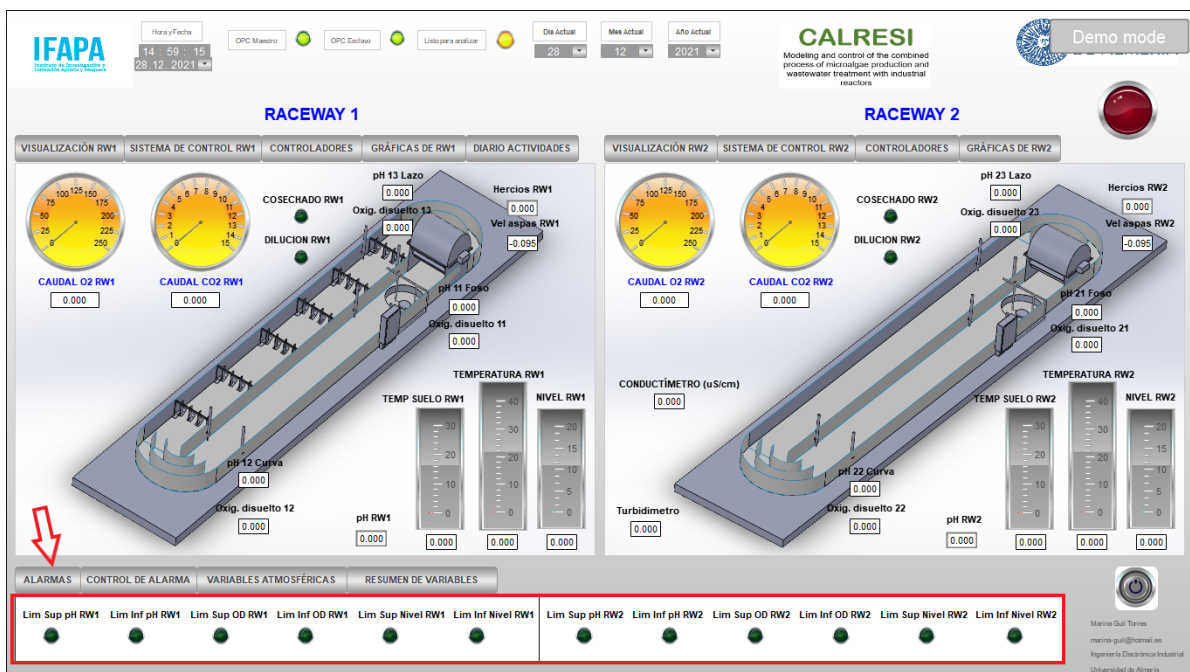
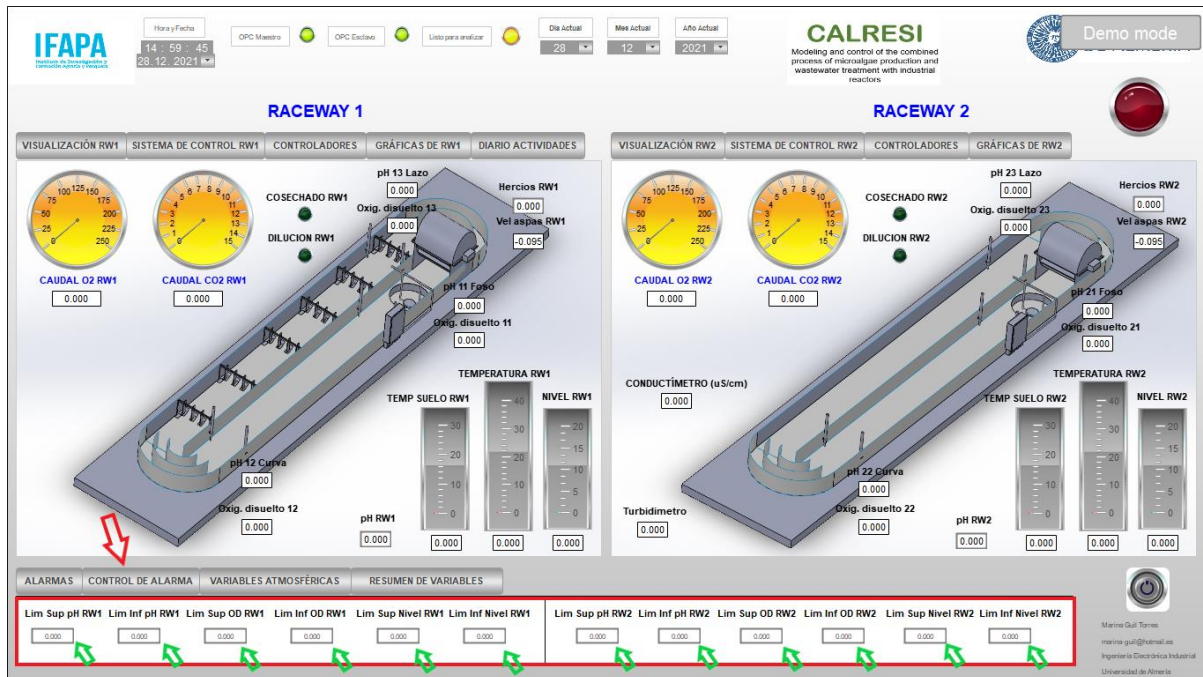


Figura 3.60. Explicación 6. Acceso a la pestaña “DIARIO” (rojo). Introducir nombre, comentario y registro de entrada (verde).

Por último, con los botones “Alarmas” y “Control de Alarmas”, se autoriza a entrar a unas pestañas donde se establecen los valores máximos y mínimos de las variables de nivel, oxígeno disuelto y pH de los reactores (Figura 3.61). Cuando los valores exceden estos límites se activan unas lámparas LED de alerta.



a) Acceso a la pestaña “Alarmas” (rojo).



b) Acceso a la pestaña “Control_Alarmas” (rojo). Establecimiento de los límites máximo y mínimo de las variables indicadas (verde).

Figura 3.61. Explicación 7.

El botón de STOP resetea todas las variables del panel gobernadas por el usuario para dejarlo como al inicio de la ejecución. A su vez, como ya se explicó con mayor profundidad en el apartado 3.7. POU “Reseteo”, se detienen las acciones de las POU’s “Tiempo” y “PLC_PRG” y de las funciones de Matlab. De modo que en el panel se visualicen, durante todo el tiempo que se esté en parada, los datos que quedaron registrados justo en el momento de ser pulsado el botón. Además, se activa una lámpara de alarma que indica el estado de activación o no de este botón (Figura 3.62).

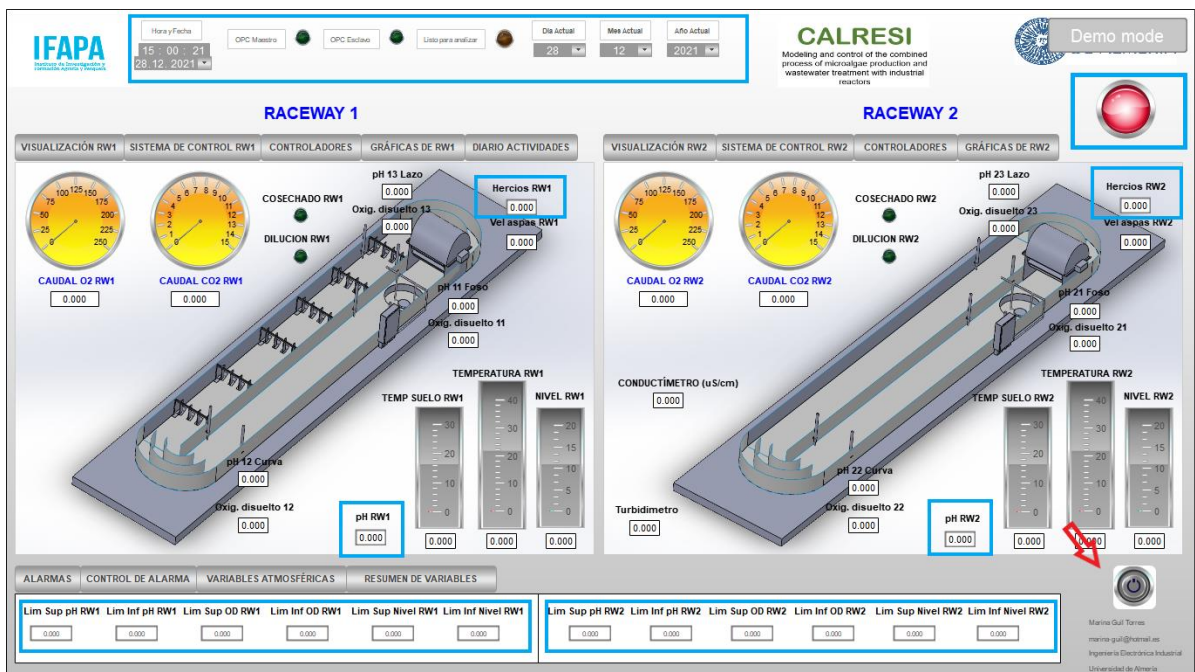


Figura 3.62. Explicación 8. Activación del STOP (rojo). Resultado de activar la parada (azul).

3.14. Puesta en marcha

El último paso a realizar para culminar este proyecto es realizar la comunicación del SCADA diseñado con el sistema real a través de los dos PLCs descritos en el primer apartado de “Capítulo 2: Materiales y métodos”. Como también se comentó en ese mismo capítulo, es preciso la utilización de una aplicación llamada Matrikon OPC Data Manager para tal fin.

Codesys para poder funcionar con facultades plenas precisa de un PLC que posibilite la ejecución del proyecto con el que se está trabajando. Esto se traduce en que la plataforma actúa como un Servidor en la comunicación OPC y no como un Cliente, que es lo que se precisa que sea respecto a los otros dos PLCs en comunicación con los reactores. La conexión directa entre servidores por sí sola no es posible por lo que se hace necesaria una herramienta que sirva de puente entre ellos y posibilite su unión. Esa herramienta no es otra que Matrikon OPC Data Manager.

Su funcionamiento es muy sencillo. Como se muestra en la Figura 3.63, la interfaz se divide en dos ventanas. En ambas se recogen todos los servidores OPC disponibles en el ordenador. Solo hay que seleccionar una variable de uno de los lados y arrastrarla hasta la variable de la otra ventana a la que corresponda. Una vez hecho se estipulan propiedades de la unión como el tipo (unidireccional o bidireccional), nombre del tag, etc.

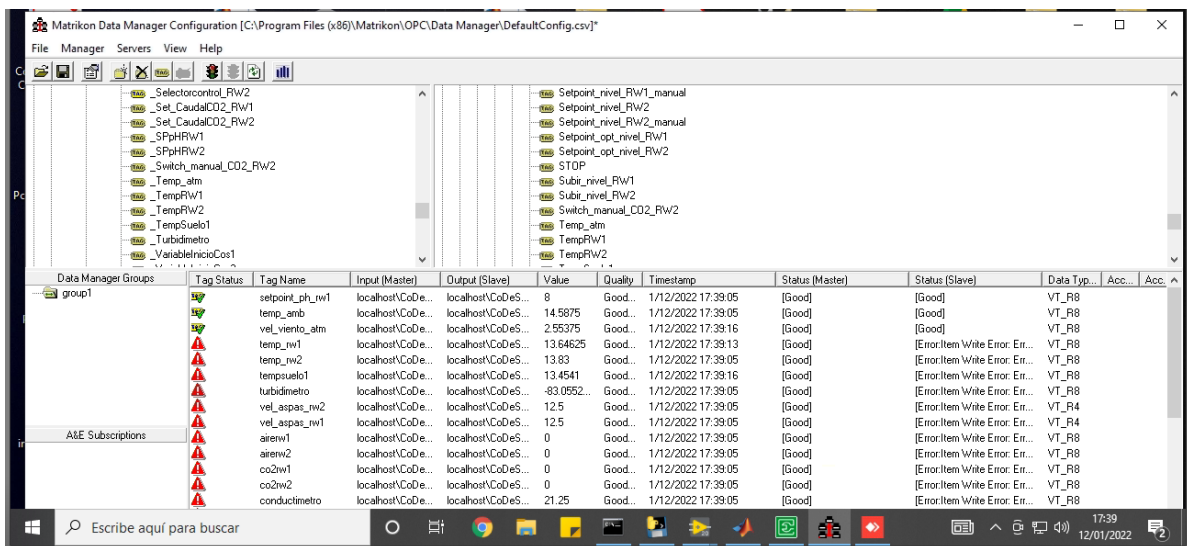
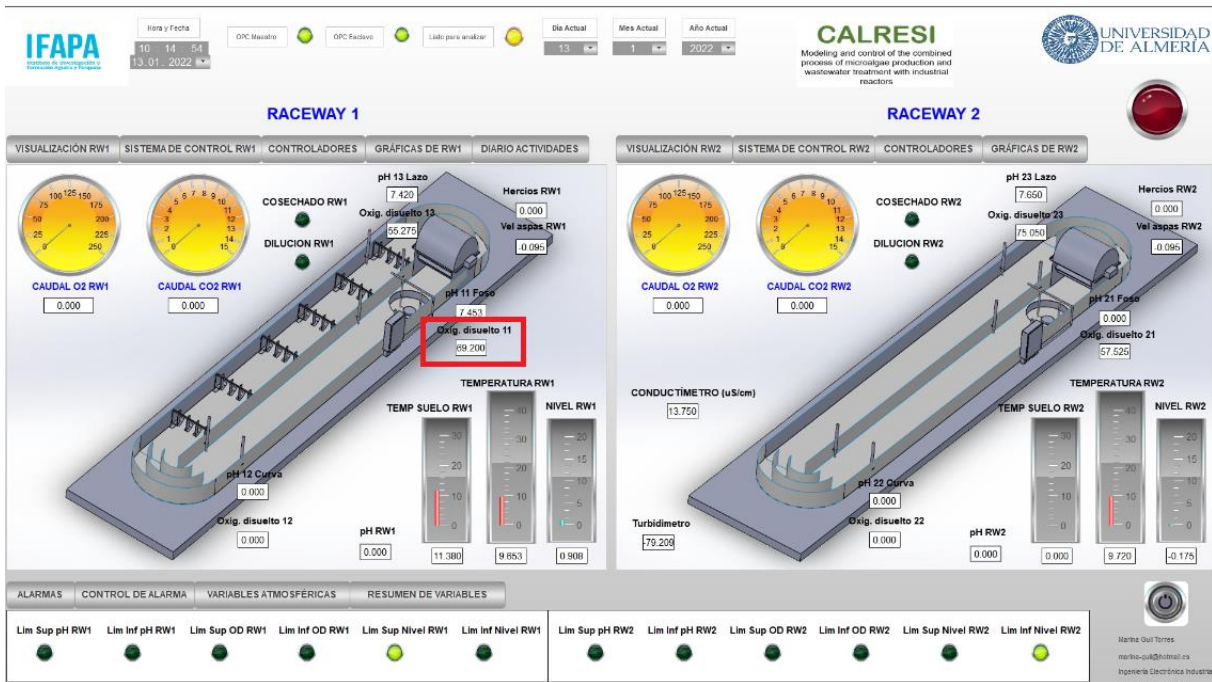


Figura 3.63 Interfaz de Matrikon OPC Data Manager.

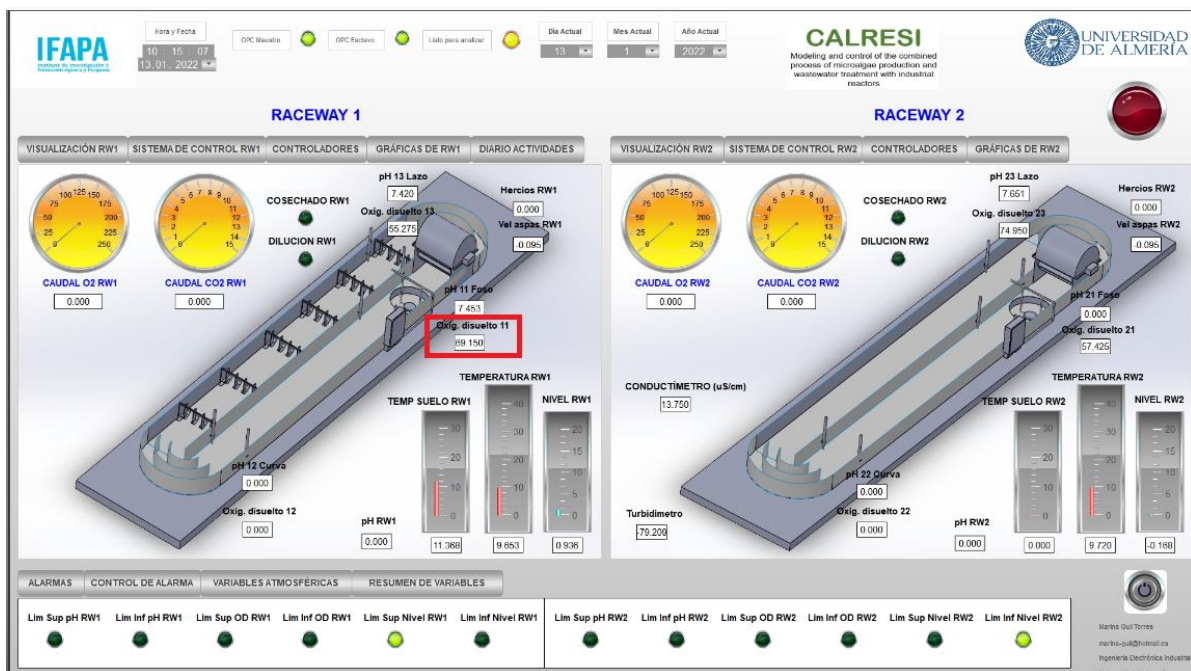
Con esto el SCADA está a punto para ser implementado. A continuación, se va a exponer una serie de ejemplos que confirmen el correcto funcionamiento de la herramienta:

1º. Comunicación entre Codesys y los PLCs: la Figura 3.64 muestra como el dato de la variable Oxígeno Disuelto 11 (ODRW11) se actualiza continuamente en el tiempo con los valores que el PLC MAESTRO está registrando del sensor. Con esto se confirma que la comunicación del PLC con el SCADA a través Matrikon se está produciendo correctamente.

Desarrollo de una herramienta SCADA en Codesys para fotobiorreactores industriales



a) Captura 1 del panel.

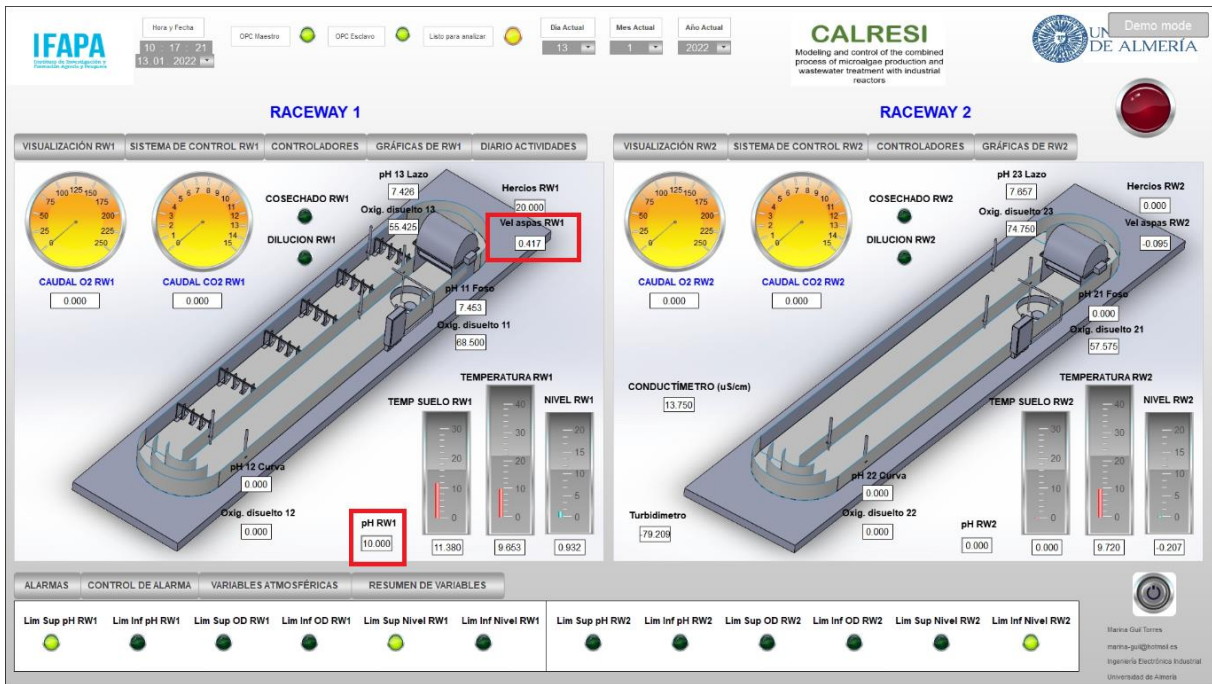


b) Captura 2 del panel.

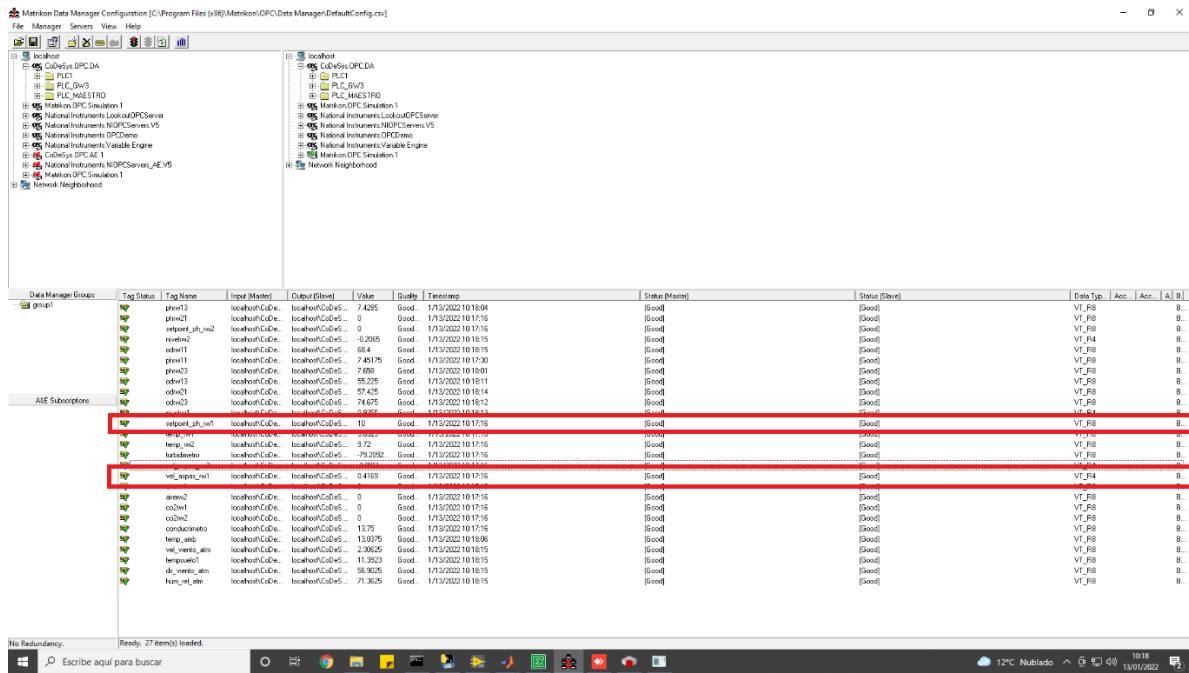
Figura 3.64. Ejemplo de comunicación de los PLCs a SCADA.

Ahora, para demostrar la comunicación contraria, es decir, el envío de datos del SCADA al PLC, se introducen dos setpoints en el panel (Figura 3.65.a), uno de ellos el valor de pH del raceway 1 (pH_RW1) y el otro el valor de Hercios del reactor 1 (HERCIOS_RW1). Este último se realiza para comprobar también que se están ejecutando las unidades de programación de Codesys, puesto que el valor que se transmite por el servidor es la velocidad de las aspas del raceway 1, Vel_Aspas_RW1 (Figura 3.65.b), y esta se calcula en la POU “PLC_PRG” a partir de los Hz.

Desarrollo de una herramienta SCADA en Codesys para fotobiorreactores industriales



a) Introducción de setpoints.



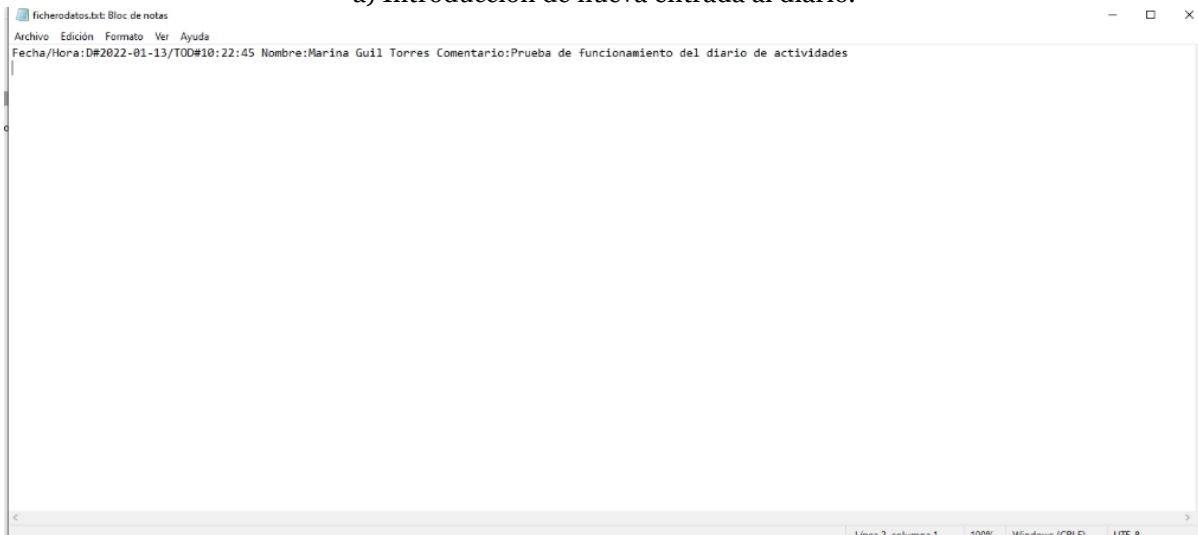
b) Actualización de los valores en Matrikon.

Figura 3.65 Ejemplo de comunicación del SCADA a los PLCs.

2º. Comunicación entre Matlab y Codesys: para confirmar que también intercambian datos estos dos programas se ha introducido una entrada al diario de actividades del panel del usuario, como se indica en la Figura 3.66.a. Esta queda registrada en un archivo de texto (Figura 3.66.b).



a) Introducción de nueva entrada al diario.



b) Archivo de texto con la entrada.

Figura 3.66. Ejemplo de comunicación del SCADA a Matlab, escritura de una entrada en el diario de actividades.

Otro ejemplo que lo demuestra se observa en la Figura 3.67, donde se ha activado el optimizador del reactor 1 y, por consiguiente, se visualizan sobre el panel los valores que recibe Codesys de la función de Matlab dedicada al cálculo del setpoint de nivel óptimo y de los valores máximo y mínimo de esta variable.

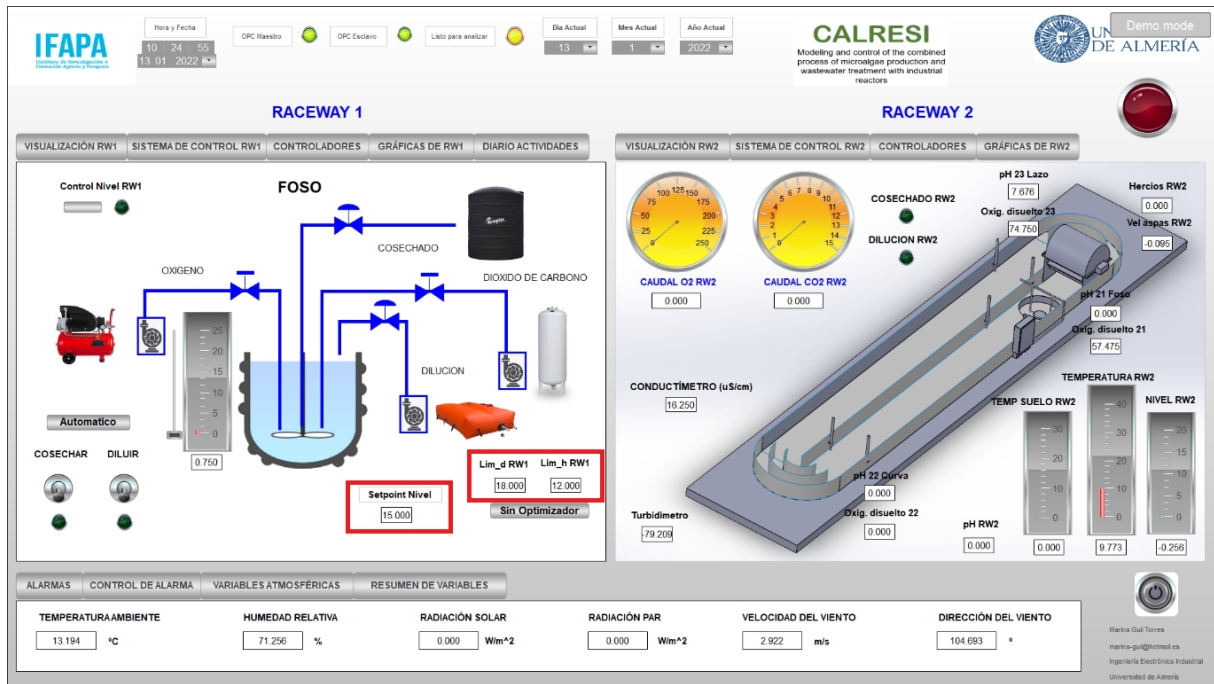


Figura 3.67 Ejemplo de comunicación del SCADA a Matlab y de nuevo al SCADA, cálculo de los parámetros del optimizador 1.

Capítulo 4: Conclusiones

De los resultados obtenidos se extrae que es posible el desarrollo de una herramienta SCADA empleando la plataforma Codesys. Sin embargo, para que esta resulte lo suficientemente competente como para ser empleada en la industria es preciso aportar capital, ya que como se comentó en alguna ocasión, aunque el entorno de desarrollo es gratuito, el servidor OPC necesario para hacer las comunicaciones no lo es. Para este trabajo fin de grado el uso de la versión de demostración de 30 días era más que suficiente, pero para una industrial real resultaría inviable porque supondría tener que renovar la licencia de uso todos los meses.

Otra limitación importante de esta plataforma es que no permite la comunicación directa con un servidor externo y mucho menos como Cliente, lo que se traduce en que Codesys no es capaz de recibir datos de otros servidores y utilizarlos en su programación. Esto supone un inconveniente cuando, como ocurre en este caso, se quiere comunicar con un PLC para el intercambio de datos de uno a otro.

Por último, otro aspecto importante es que Codesys para poder ponerse en ejecución debe conectarse a un PLC, o PLC Virtual como es el caso, donde además se carga la visualización con el panel de control del usuario. Este autómatas programable solo trabaja durante 2 horas seguidas y luego detiene la ejecución del proyecto. De modo que tendría que adquirirse un dispositivo extra que se encargase de mantener el SCADA en funcionamiento sin limitaciones, deteniéndose la ejecución de la herramienta únicamente cuando el usuario así lo quiera.

En conclusión, de todo esto se extrae que es posible implementar una herramienta SCADA haciendo uso de Codesys, considerando los aspectos anteriormente mencionados.

Como futuros trabajos a realizar para la mejora de la funcionalidad de esta herramienta SCADA se plantean:

- Sustitución del PLC Virtual de Codesys por un dispositivo real y sin limitaciones de uso.
- Sustitución de la versión de prueba del servidor DA de Codesys por el producto con funcionalidad plena.
- De igual modo se procede con la aplicación Matrikon OP Data Manager.

Asimismo, debería hacerse un estudio de los lenguajes de programación disponibles en Codesys y su compatibilidad o equiparación con el código de Matlab para poder así comprobar si es posible prescindir de este último software para hacer las labores de tratamiento de datos de los fotobiorreactores y unificar todo el código en el entorno de Codesys.

Bibliografía/Referencias

- [1] Introducción al estándar IEC 61131-3. http://www.infopl.net/files/documentacion/estandar_programacion/infoPLC_net_Intro_estandar_IEC_61131-3.pdf. Fecha de acceso: octubre de 2021.
- [2] Paleta de funciones/Programación. https://labviewwiki.org/wiki/Functions_Palette/Programming. Fecha de acceso: noviembre de 2021.
- [3] CODESYS TARGETVISU. <https://www.codesys.com/products/codesys-visualization/targetvisu.html>. Fecha de acceso: noviembre de 2021.
- [4] CODESYS WEBVISU. <https://www.codesys.com/products/codesys-visualization/webvisu.html>. Fecha de acceso: noviembre de 2021.
- [5] Visualización CODESYS (Folletos y material informativo CODESYS). <https://www.codesys.com/products/codesys-visualization/targetvisu.html>. Fecha de acceso: diciembre de 2021.
- [6] García-Rodríguez, A., TFG, Universidad de Jaén, España, mayo 2019. “Estudio de Codesys, entorno de edición de programas para PLCs y simulación de estaciones automatizadas”.
- [7] Guzmán, J.L., Ación, F.G., Berenguel, M. 2021. “Modelado y control de la producción de microalgas en fotobiorreactores industriales”. Revista Iberoamericana de Automática e Informática industrial,18, 1-18, 2021.
- [8] Rodríguez-Torres, M.J., Morillas-España, A., Guzmán, J.L., Ación, F.G. “Modelling and pH Control in Raceway and Thin-Layer Photobioreactors for Wastewater Treatment”. Energies 2021, 14, 1099.
- [9] Romero-Prada, J., TFG, Universidad de Sevilla, España, 2018 “Automatización de una planta de almacenaje y distribución de mercancías usando Factory I/O y Codesys”
- [10] Martínez-Parreño, P., TFG, Universidad de Sevilla, España, 2020 “Simulación y control de una planta multiprocesos utilizando Codesys, OPC y Matlab”
- [11] CODESYS -El paquete de software completo para la tecnología de automatización. <https://larraioz.com/codesys>. Fecha de acceso: octubre de 2021.
- [12] ¿Qué es LabVIEW? <https://www.ni.com/es-es/shop/labview.html>. Fecha de acceso: octubre de 2021.
- [13] Fundamentos del Entorno de LabVIEW. <https://www.ni.com/getting-started/labview-basics/esa/environment>. Fecha de acceso: enero de 2022.

- [14] Conceptos básicos en automatización con PLC. <http://automatizacioncavanilles.blogspot.com/2019/09/conceptos-basicos-PLC.html>. Fecha de acceso: noviembre de 2021.
- [15] Servidor Codesys OPC. <https://www.codesys.com/products/codesys-runtime/opc-server.html>. Fecha de acceso: diciembre de 2021.
- [16] González-Hernández, J., TFG, Universidad de Almería, España, 2020. “Optimización de temperatura en reactores Raceway mediante el control de altura del medio usando predicciones meteorológicas”.
- [17] MATLAB. <https://es.mathworks.com/products/matlab.html>. Fecha de acceso: diciembre de 2021.
- [18] MatrikonOPC Data Manager. <https://www.matrikonopc.es/products/opc-data-management/opc-data-manager.aspx>. Fecha de acceso: enero de 2022.

Lecturas recomendadas:

- [19] Miguel Ángel Ridao Carlini, Editorial Universidad de Sevilla “Introducción a la programación de autómatas programables usando Codesys”, 1º edición, 2016. Monografías de la Escuela Técnica Superior de Ingeniería, N° 8.
- [20] Liam Bee, ebook, “The Basics Of PLC Programming With Codesys: A beginners guide to getting started with PLCs and the Codesys environment”, 12 noviembre 2021.
- [21] Gary Pratt, ControlSphere LLC “The Book of CODESYS: The ultimate guide to PLC and Industrial Controls programming with the CODESYS IDE and IEC 61131-3.”, 19 octubre 2021.
- [22] Lane Pickrell “SCADA Tutorials: Real Time Projects And Case Studies.”, 1 abril 2021.

Recopilatorio de videos de YouTube de apoyo:

https://www.youtube.com/results?search_query=tutorial+codesys+david+mu%C3%B1oz+de+la+pe%C3%B1a
<https://www.youtube.com/c/LeonardoRodriguezOrtiz/videos>
<https://youtu.be/xIFNqiw05p4>
<https://www.youtube.com/watch?v=NqkMC71DeoQ>
<https://www.youtube.com/watch?v=q3OnLZ5YLnA>
<https://www.youtube.com/watch?v=LGnf8I2VWds>
<https://www.youtube.com/watch?v=cVS7pfXOtfY>
<https://www.youtube.com/watch?v=d6MJoeC1I5g>
<https://www.youtube.com/watch?v=Jd27xhqMOl4>
<https://www.youtube.com/watch?v=oqlkHhAWBVs>
<https://www.youtube.com/watch?v=cVS7pfXOtfY>
<https://www.youtube.com/watch?v=3abGmNBkW2Q>

Anexos

6.1. Instalación de los programas

Para la descarga del entorno de desarrollo Codesys V3 y del servidor empleado para la comunicación entre aplicaciones, Codesys OPC DA Server, solo hay que acceder a su tienda online oficial (Figura 6.1) a través del siguiente enlace <https://store.codesys.com/de/> y registrarse como usuario.



Figura 6.1. Tienda online de Codesys.

Una vez hecho esto, basta con introducir en el buscador el nombre del producto que se pretende adquirir y se procede a su descargar. Si existen diversas versiones del programa se puede instalar la que más convenga, en función del procesador del que disponga el ordenador donde se instalará (32 o 64 bits). Si no se selecciona ninguna en concreto, se instalará por defecto la más actual. En este caso se instaló la versión más reciente 3.5.17.20 (x64) bits.



Versiones planificadas

El generador de códigos CODESYS para la familia Infineon C166 / C167 ha sido descontinuado por V3.5 SP18
CODESYS Codegenerator Nit6 es discontinuado por V3.5 SP17
CODESYS Codegenerator 186 se discontinúa por V3.5 SP17

Historial de versiones

Versión	.exe (autoejecutable) 32 bits	.exe (autoejecutable) de 64 bits	.zip (archivo sin comprimir) 32 bits	.zip (archivo sin comprimir) 64 bit	Notas de lanzamiento	Fecha de lanzamiento
3.5.17.20	3.5.17.20 32 bits (.exe)	3.5.17.20 64 bits (.exe)	3.5.17.20 32 bits (.zip)	3.5.17.20 64 bits (.zip)	Actualización de calidad con pequeñas mejoras y corrección de errores. Incluye las siguientes correcciones de seguridad: CVE-2021-111 , CVE-2021-112 , CVE-2021-113	03.11.2021
3.5.17.10	3.5.17.10 32 bits (.exe)	3.5.17.10 64 bits (.exe)	3.5.17.10 32 bits (.zip)	3.5.17.10 64 bits (.zip)	Actualización de calidad con pequeñas mejoras y corrección de errores. Incluye las siguientes correcciones de seguridad: CVE-2021-09 , CVE-2021-111 , CVE-2021-112 , CVE-2021-113	23.07.2021
3.5.17.0	3.5.17.0 32 bits (.exe)	3.5.17.0 64 bits (.exe)	3.5.17.0 32 bits (.zip)	3.5.17.0 64 bits (.zip)	Actualización principal con varias mejoras y correcciones de errores. Nuevas funciones y mejoras Ingeniería: <ul style="list-style-type: none">• Editor de GFC: numerosas mejoras menores y mayores• Más seguridad a través de bibliotecas CODESYS firmadas• Mejoras generales de rendimiento en el sistema de desarrollo CODESYS• Actualizaciones de funciones mucho más rápidas y posible interfaz de usuario reducida a través de la modularización integral del sistema	28.04.2021

Figura 6.2. Pestaña para la descarga de Codesys y algunas de sus posibles versiones disponibles.

Como se observa en la figura 6.2, en el caso del entorno de desarrollo su instalación es gratuita, mientras que la del servidor OPC no. Sin embargo, Codesys proporciona un servidor de prueba (Figura 6.3) con una licencia gratuita durante 30 días que se puede renovar una vez expire esta. Este es el servidor que se empleó en este proyecto fin de grado y su nombre es Codesys OPC DA Server SL Demo o Demostración de Codesys OPC DA Server SL, en español.

store.codesys.com/en/codesys-opc-da-server-sl-demo.html

Bienvenido a CODESYS Store Internacional | EN | Iniciar sesión

Buscar la tienda entera aquí...

Servidor de automatización | SofPLC | Ingeniería | Capacitaciones y eventos | Ejemplos de

Demostración de CODESYS OPC DA Server SL

El CODESYS OPC Server V3 se basa en PLCHandler de 3S-Smart Software Solutions GmbH. Este módulo de comunicación permite una comunicación directa a todos los PLC programables con CODESYS (mayor disponible una licencia de demostración de 30 días para realizar pruebas). Al solicitar una licencia de prueba gratuita (solicitada a través del carrito de compras), es posible probar el producto durante 30 días.

Versión Actual: 3.5.17.0
Artículo No.: 2302000028

0,00 €
Más el IVA

[Descargar](#) [Iniciar sesión para agregar al carrito](#)

Descripción del producto | **Versiones** | **Reseñas**

Licencia:
Licencia de dispositivo único
El servidor OPC puede comunicarse con ambos PLC V3 y V2.3, al mismo tiempo. Además, se puede simular un PLC faltante. Luego, un archivo de símbolos con todos los símbolos en lugar del PLC proporciona al servidor OPC el grupo de símbolos (archivo XML o archivo SDB).
Puede encontrar una descripción detallada de todas las funcionalidades y la puesta en servicio en la guía del usuario de CODESYS OPC DA (parte de la configuración).

Requisitos

Figura 6.3. Pestaña para la descarga del servidor Codesys OPC DA Server SL Demo.

Entrando en la cuenta de usuario, en el apartado “Mis licencias”, está disponible la licencia de instalación, como muestra la Figura 6.4.

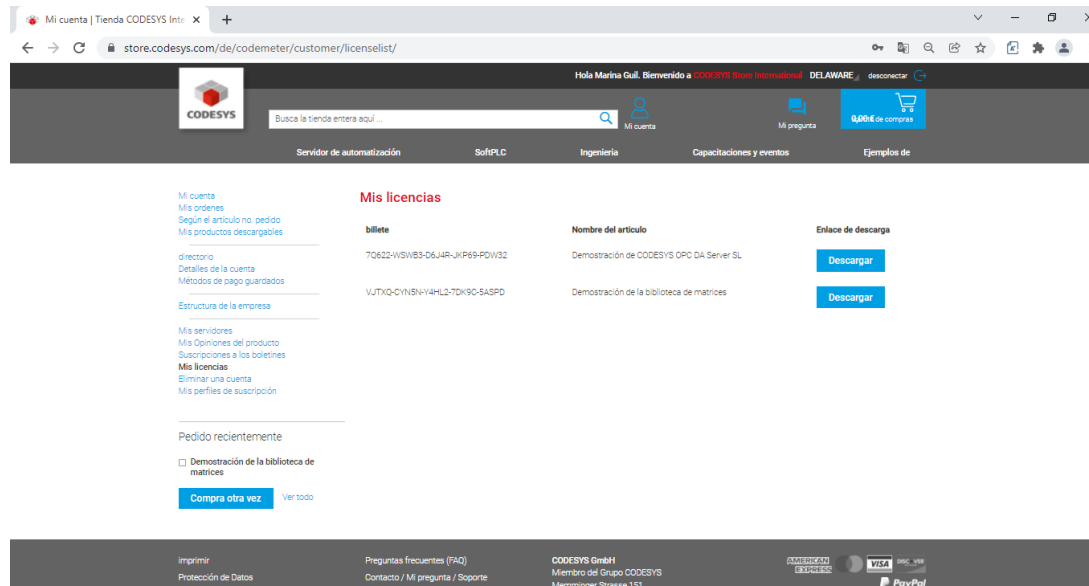


Figura 6.4. Cuenta de usuario, pestaña Mis licencias.

El código que aparece en primer lugar o “billete” se necesitará para la configuración del servidor más adelante.

Para la instalación de Matlab se siguen los siguientes pasos descritos en un PDF que proporciona la Universidad de Almería:

1. Creación de una cuenta de usuario: a través del enlace <http://es.mathworks.com> y pinchando en entrar, creamos una nueva MathWorks Account. Esta debe asociarse al correo universitario e indicar que se trata de un “Student Use”, es decir, que está destinado para el uso estudiantil. De esta manera se obtendrá la licencia gratuitamente.
2. Asociación de la MathWorks Account con la licencia para estudiantes de la Universidad de Almería: pinchando sobre tu nombre en la esquina superior derecha y eligiendo “Asociar Licencia” se puede conseguir asociar el código que proporciona la universidad.
3. Descarga e instalación del software: a través del siguiente link está disponible el programa para su descarga e instalación, <https://es.mathworks.com/downloads/>.

Matrikon OPC Data Manager está disponible a través de la página oficial de Matrikon (link: https://www.matrikonopc.com/downloads/175/software/index.aspx?utm_campaign=OPCSpain&utm_medium=referral&utm_source=google.com). Para adquirirlo solo hay que registrarse en la web y descargar el producto. Luego durante la instalación del mismo se selecciona la versión de prueba gratuita de 30 días.

6.2. Activación de la licencia del servidor OPC

Una vez descargado el servidor, hay que activar la licencia de uso del mismo. Para ello se ejecuta Codesys y se accede al apartado “Herramientas” (“Tools”) de la barra de comandos superior de la interfaz y se hace clic en “Administrador de licencias”.

Ahora se debe seleccionar donde se desea activar la licencia: “Workstation” si se quiere activar en un ordenador o “Device” si, por el contrario, la activación se va a realizar en otro dispositivo (PLC). En mi caso seleccioné Workstation.

Luego, hay que elegir el repositorio o contenedor de la licencia (“Container”): “Dongle” si se guarda en una memoria tipo USB que podrá ser utilizada en otros dispositivos, o “SoftContainer” si se desea guardar en un contenedor de licencias asociado al programa y al equipo. Para este proyecto se seleccionó esta última opción, como se observa en la Figura 6.5.

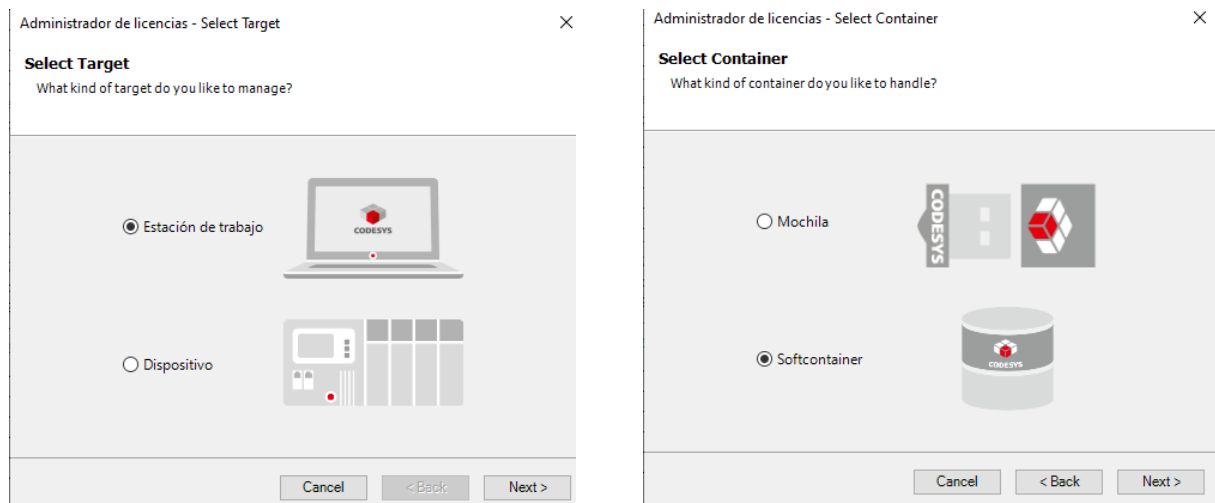


Figura 6.5. Selección del Target y Container del servidor.

A continuación, aparece una ventana llamada “Administrador de licencias” donde pinchando en “Activar licencias” te permite la opción de Activar, Solicitar o Instalar una licencia. Seleccionamos la primera opción y aparece una nueva ventana donde se debe introducir el código de activación o “billete”, del que se habló en el apartado “6.1. Instalación de programas”. La Figura 6.6 muestra este paso de la activación de la licencia.

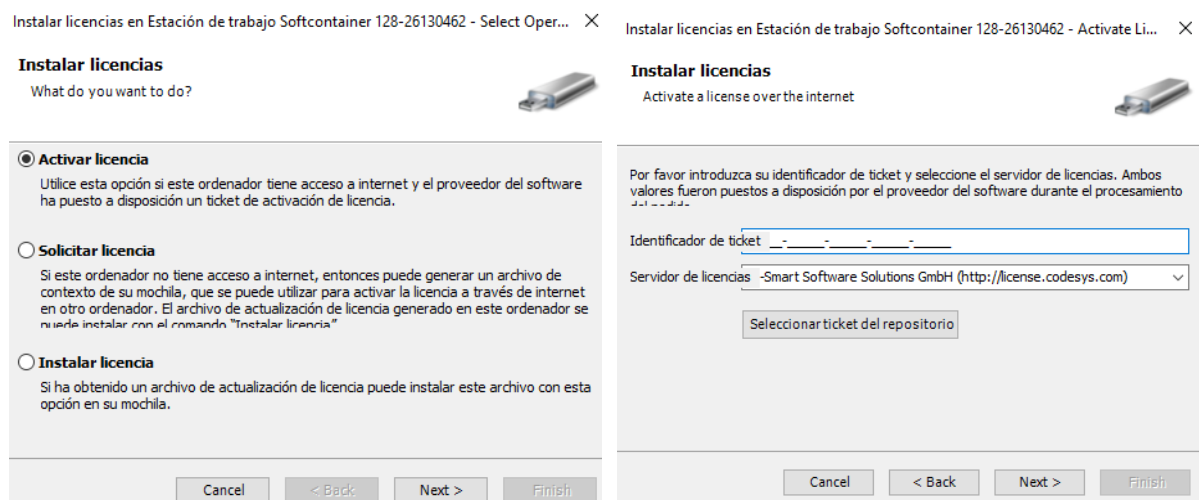


Figura 6.6. Activación de la licencia del servidor.

Una vez hecho nos devuelve de nuevo a la pestaña anterior (Administrador de licencias) y si todo el proceso se ha llevado adecuadamente deberá aparecer en la sección de productos el servidor junto con un tick verde. Como se puede ver en la Figura 6.7.

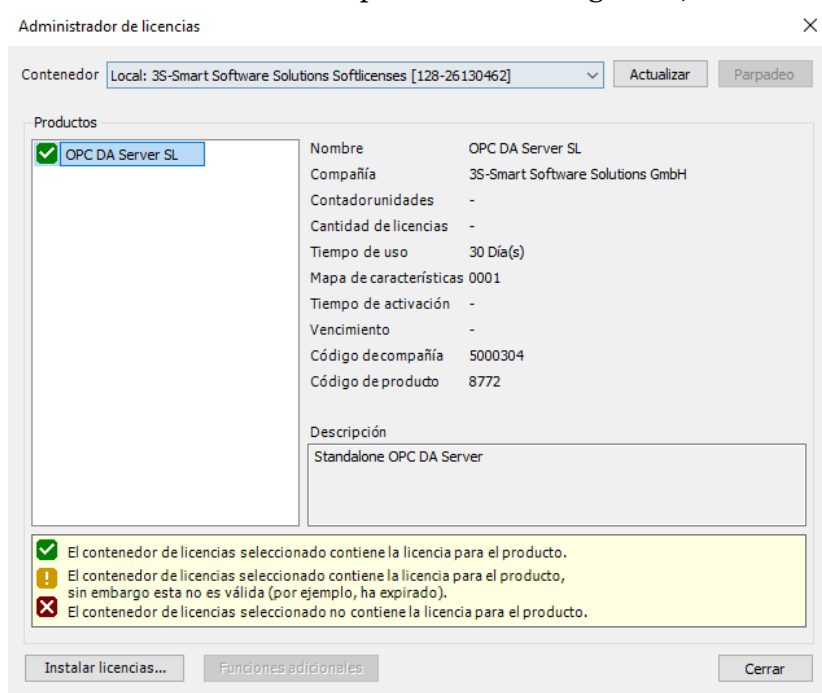


Figura 6.7. Administrador de licencias.

Una vez transcurridos los 30 días de prueba de la licencia del servidor OPC, esta expira y muestra un mensaje como el que aparece en la Figura 6.8. En ese caso hay que solicitar un nuevo “billete” a través de la tienda de Codesys, como se hizo ya anteriormente.

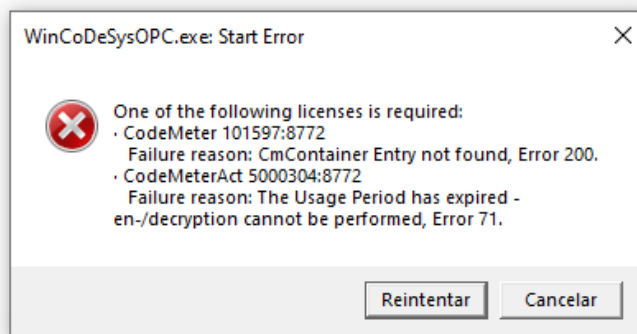


Figura 6.8. Mensaje de error, expiración de la licencia de uso.

Luego entramos en Codesys y tratamos de activar la licencia. Aparece el mensaje de error mostrado en la Figura 6.9.

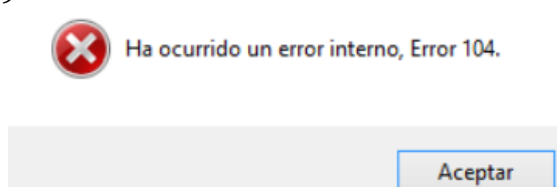


Figura 6.9. Mensaje de error de Codesys.

Este error se produce porque al parecer no es posible actualizar una licencia de prueba en el contenedor de licencias. Por ese motivo se debe crear un nuevo contenedor y vincular la nueva licencia a este. Para ello accedemos al directorio de instalación de Codesys, a la carpeta “GatewayPLC”, donde están todos los documentos referentes al uso y activación del servidor, y abrimos el archivo tipo “WISU Binary File” que se muestra a continuación en la Figura 6.10. Aparecerá una ventana donde se pregunta si se desea importar una licencia, se acepta y se abrirá el Centro de Control de CodeMeter.

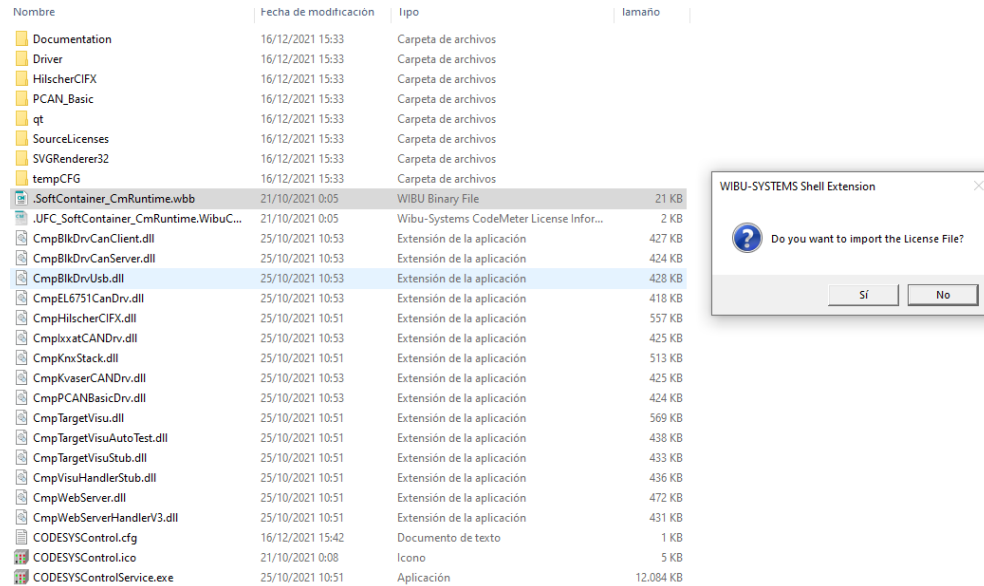


Figura 6.10. Carpeta GatewayPLC.

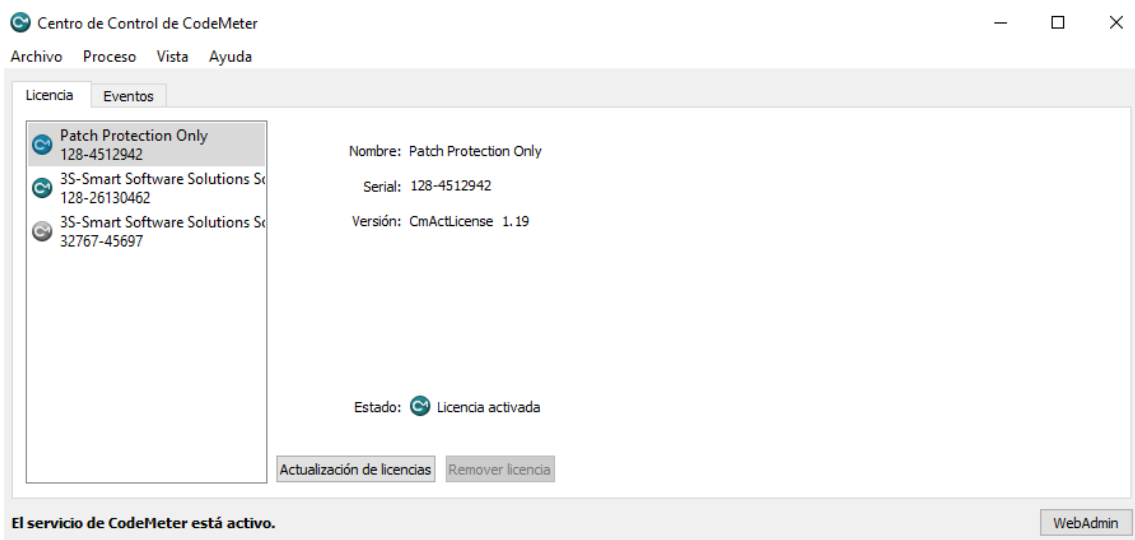
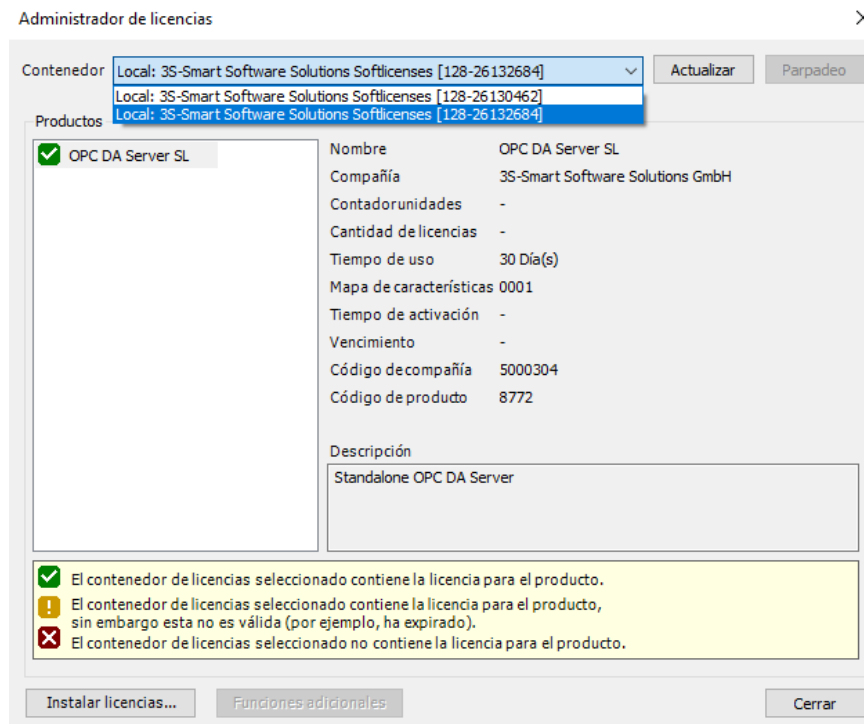


Figura 6.11. Centro de Control de CodeMeter.

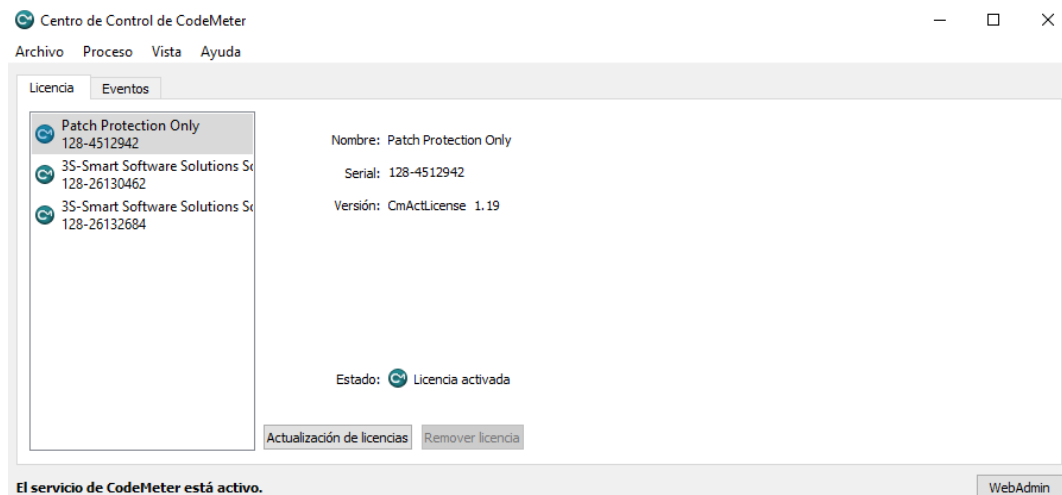
Como se atisba en la Figura 6.11, se ha creado un nuevo contenedor que aparece con el símbolo de CodeMeter en gris. Esto se debe a que no tiene ninguna licencia vinculada a él aún.

Regresando de nuevo a Codesys y refrescando el Administrador de licencias, se repiten los pasos de activación de la licencia, pero para este nuevo contenedor que se ha generado. Si todo se ha hecho correctamente, el Administrador de licencias y el Centro de Control de CodeMeter

deberán tener el aspecto mostrado en la Figura 6.12.



a) Administrador de licencias.



b) Centro de Control de CodeMeter.

Figura 6.12. Resultado de la nueva activación de la licencia del servidor OPC.

6.3. Descarga de la biblioteca *Oscat basic* desde Codesys Store

Para la realización de este trabajo fue necesaria la descarga de una biblioteca adicional llamada “Oscat basic” desde la tienda online de Codesys. Esta cuenta con un gran número de funciones y operaciones entre las que se encuentran unas que resultaban necesarias en la POU “Tiempo”.

Como ya se mencionó, en esa unidad de programación se congregaban la mayoría de acciones referidas a la obtención de la fecha y hora de nuestro sistema, así como temporizadores que proporcionasen señales de pulso precisas para el correcto comportamiento de las funciones de Matlab cuando este recibe, a través del servidor, las variables compartidas. Entre esas operaciones necesarias estaba, por ejemplo, separar la fecha actual en el día, mes y año correspondientes, y esta función solo se podía lograr introduciendo las aplicaciones que proporciona esta biblioteca adicional.

Su instalación resulta muy sencilla, basta con entrar en la tienda Codesys Store y hacer una búsqueda de la biblioteca (Figura 6.13). Está disponible de forma gratuita así que no hay que solicitar ninguna licencia para su uso, sino solo descargar el software.

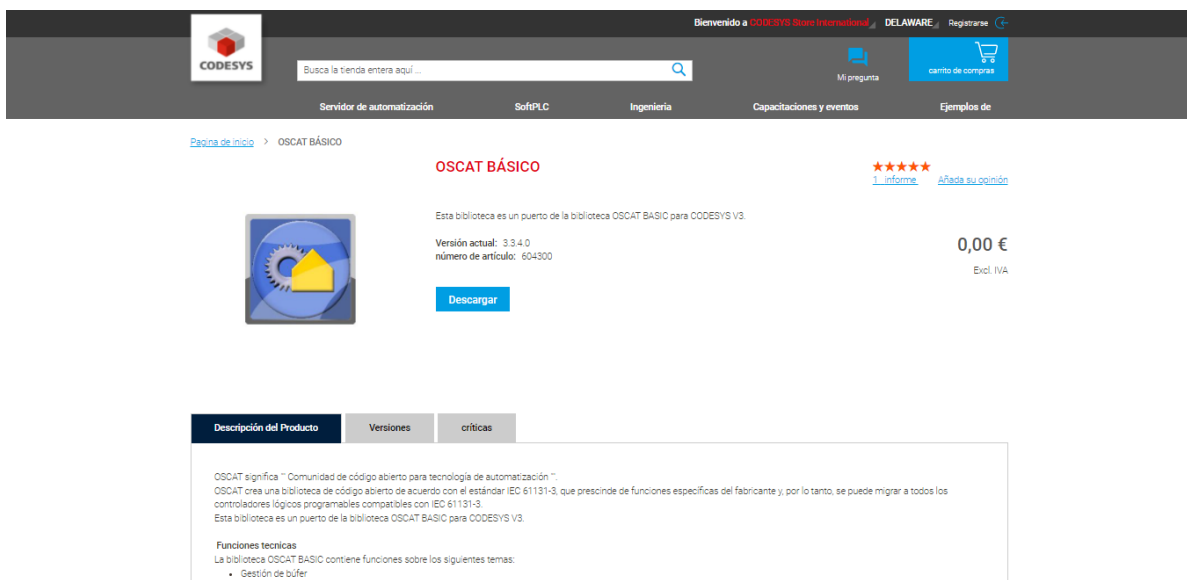


Figura 6.13. Tienda Online Codesys.

Una vez descargado en el ordenador se inicia Codesys. Dentro de la pestaña “Herramientas” se selecciona la opción “Administrador de paquetes”. Aparecerá una ventana, como la que muestra la Figura 6.14, con el listado de todas las bibliotecas que trae el software por defecto instaladas. Desde ella, haciendo clic en “Instalando...”, se busca la biblioteca que se había descargado de la tienda en el pc y se ejecuta el instalador.

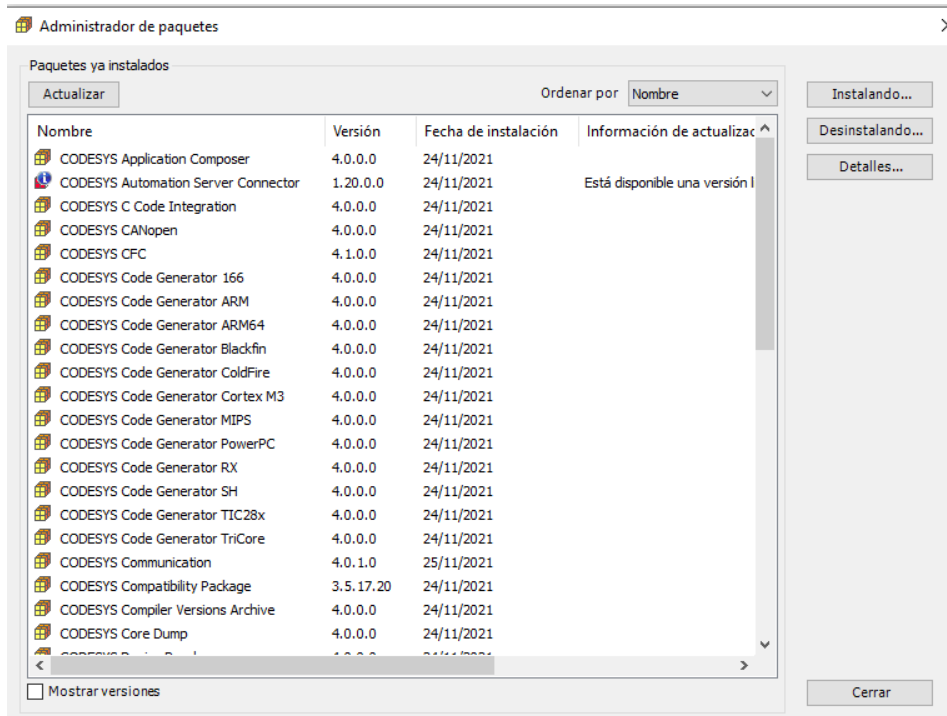


Figura 6.14. Administrador de paquetes.

Una vez el paquete ha sido instalado en Codesys, hay que agregarlo también en el proyecto para poder hacer uso de sus funciones. Para ello, en la parte izquierda de la pantalla, se selecciona el archivo “Administrador de bibliotecas” y se pulsa “Agregar biblioteca”. Aparecerá el listado de todas las bibliotecas disponibles, Figura 6.15, que coinciden con los paquetes de la Figura 6.14. Se busca “Oscat Basic” y se pulsa “Aceptar”.

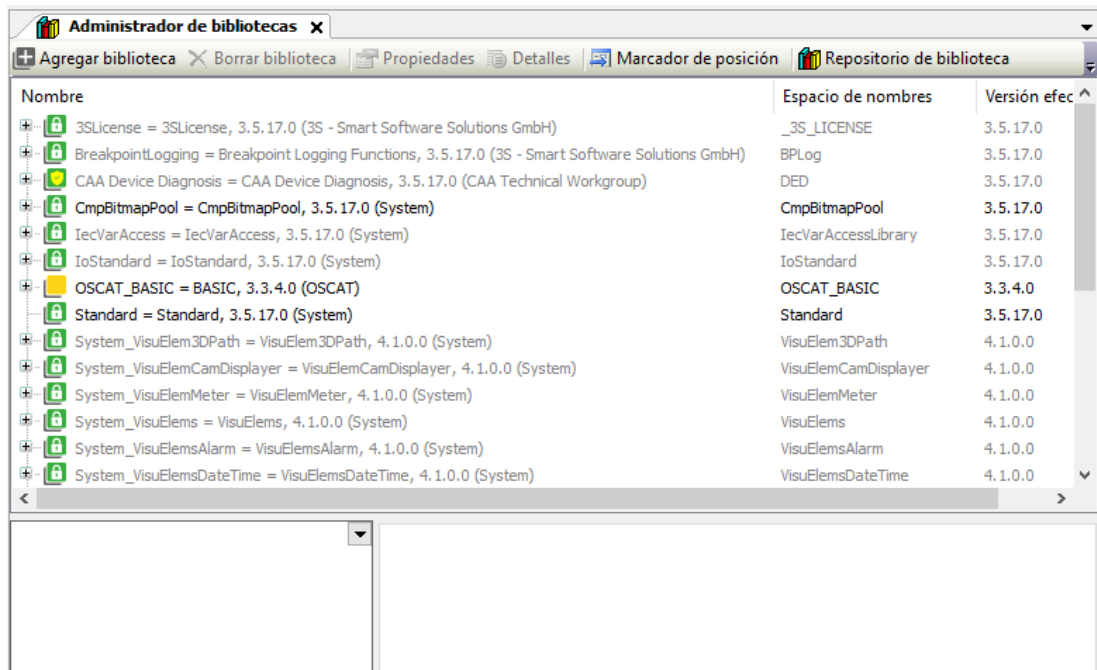


Figura 6.15. Administrador de bibliotecas.

Además de “Oscat basic”, se añadieron las bibliotecas “Útil”, “Time and Date” y “VisuDialogs” (esta última se hace de forma automática con la creación de la primera visualización dentro del proyecto). Todas ellas estaban entre las bibliotecas disponibles de Codesys por lo que no hay que descargarlas desde la página oficial.

6.4. Activación de los comentarios por línea en lenguaje LD

Para que, mientras se trabaja en lenguaje de programación LD, se puedan introducir comentarios explicando cómo se está desarrollando el código, basta con acceder al menú “Herramientas”, a “Opciones”. Aparecerá una ventana donde se pueden seleccionar diversas configuraciones del entorno de trabajo. Marcando la opción “Mostrar comentario de red”, dentro de “FB, LD y IL” (Figura 6.16), se consigue una línea de texto en la parte superior de cada red del código LD donde se pueden escribir comentarios.

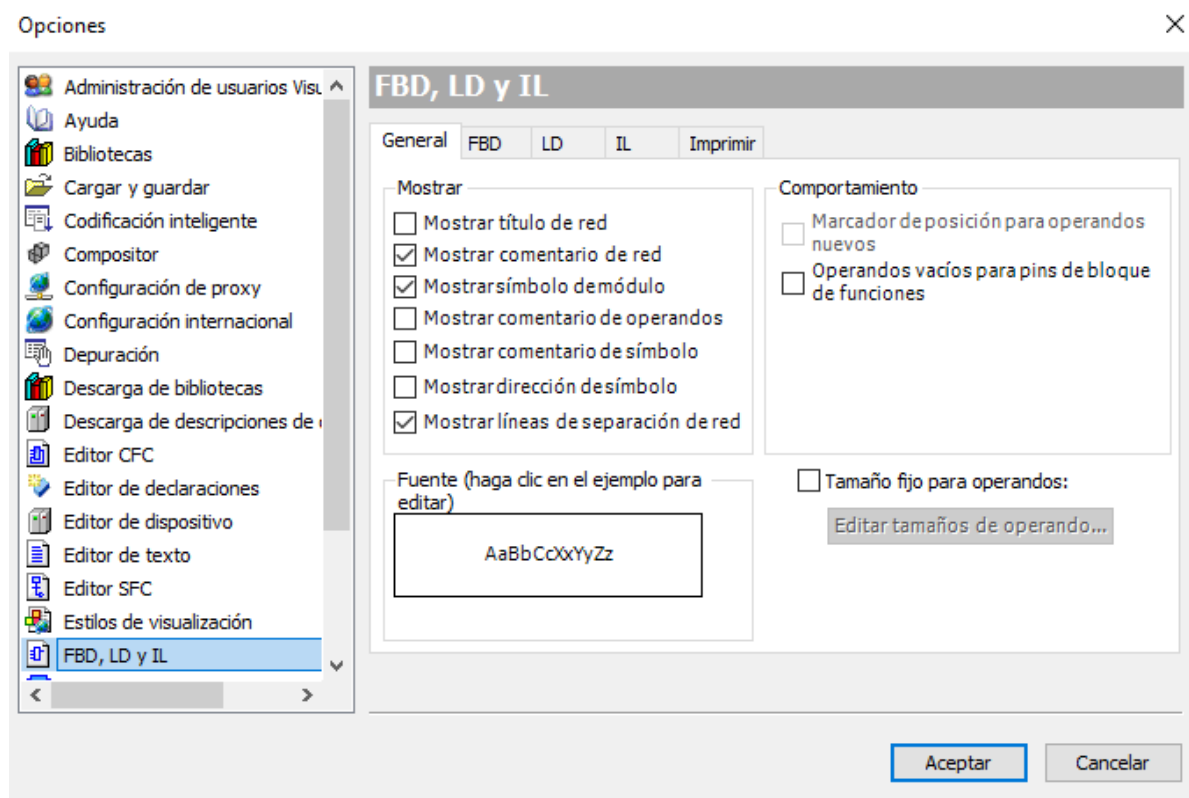


Figura 6.16. Ventana *Opciones*.



Resumen/Abstract

La motivación de este Trabajo Fin de Grado es desarrollar una herramienta SCADA, programada en Codesys, que sirva para la supervisión, control y adquisición de datos de fotobiorreactores industriales pertenecientes a instalaciones asociadas a la Universidad de Almería, las cuales se destinan al estudio de la producción de microalgas y su inserción en el mercado para usos alimenticios o cosméticos, así como para el tratamiento de aguas residuales o la reducción de emisiones de CO₂ provenientes de diversos procesos industriales, según proceda.

Ambas partes, sistema real y herramienta SCADA, se comunicarán entre sí a partir del protocolo OPC UA que proporcionará una vía para el intercambio de información de uno a otro, permitiendo supervisar así el comportamiento de los reactores en tiempo real al mismo tiempo que se ejecutan órdenes desde el cuadro de control del operario o desde otro software o entorno externo.

The motivation of this Final Degree Project is to develop a SCADA tool, programmed in Codesys, for the supervision, control and data acquisition of industrial photobioreactors belonging to facilities associated with the University of Almeria, which are intended to study the production of microalgae and their insertion in the market for food or cosmetic uses, as well as for the treatment of wastewater or the reduction of CO₂ emissions from various industrial processes, as appropriate.

Both parts, the real system and the SCADA tool, will communicate with each other using the OPC UA protocol, which will provide a way for the exchange of information from one to the other, thus allowing the behaviour of the reactors to be supervised in real time at the same time as orders are executed from the operator's control panel or from other software or external environment.