Escuela Superior de Ingeniería y Tecnología
Sección Ingeniería Industrial

Trabajo Fin de Grado

Grado en Ingeniería Industrial y Automática

Diseño e implementación de controladores Beckhoff en maquetas de procesos de control

Estudiante: Jesús Elías Soto

Tutor: Santiago Torrez Álvarez

Cotutor: Juan Albino Méndez Pérez

Convocatoria: Junio 2022

Agradecimientos

Dedico este apartado para extender agradecimientos, muy merecidos, a mi familia y amigos, concretamente a mis padres, que me han brindado la oportunidad de poder educarme a este nivel, y, por parte de amigos, a Miguel y Daniel, con los que pasé gran parte de la carrera. Gracias por haberme apoyado durante tantos años, y, en especial, durante este periodo de desarrollo del proyecto, en el que me he podido implicar sin demasiados inconvenientes gracias a su invaluable presencia y ánimos, la realización de este Trabajo Fin de Grado no habría sido posible sin ellos.

Asimismo, quiero agradecer a la Universidad de La Laguna, en concreto, a todos los profesores del grado de Ingeniería Electrónica Industrial y Automática con los que he tenido el placer de aprender, en especial a mis tutores, Santiago Torres Álvarez y Juan Albino Méndez Pérez, a los que debo agradecer por su inestimable labor pedagógica.

A todos los mencionados y no mencionados que me han apoyado durante mi transcurso en la carrera, sin excepción, gracias.

Índice de contenido

Índice de figuras

Índice de tablas

Escuela Superior de Ingeniería y Tecnología
Sección Ingeniería Industrial

Trabajo Fin de Grado

Grado en Ingeniería Industrial y Automática

Diseño e implementación de controladores Beckhoff en maquetas de procesos de control

1. RESUMEN/ABSTRACT

Estudiante: Jesús Elías Soto

Tutor: Santiago Torrez Álvarez

Cotutor: Juan Albino Méndez Pérez

Convocatoria: Junio 2022

1.1 Resumen

Este Trabajo Fin de Grado (TFG) consiste en la implementación de un sistema SCADA y un entorno apropiado de control mediante un PLC de Beckhoff para ser utilizado en procesos provistos ya de sus propios dispositivos de control, con el objetivo de establecer un diseño de control versátil que ofrezca unas mayores posibilidades de monitorización, supervisión y control de los procesos industriales.

El objetivo final es establecer un entorno de control alternativo, basado en la tecnología Beckhoff, que permita un control adecuado de un sistema a través de la inclusión de un sistema SCADA, y además, dar nuevas experiencias prácticas a futuros alumnos de grado. De esta forma se tiene una visión más amplia de los sistemas de control en la industria, pasando de un análisis de los sistemas de control a nivel de campo y control, a la inclusión de niveles de supervisión y con posibilidad de conexión a niveles de operación (MES), dentro de la pirámide de automatización.

La aplicación en particular se llevará a cabo en la planta de nivel y flujo de líquidos, en primera estancia observando su comportamiento con el controlador ABB que viene por defecto, y ampliando las posibilidades de control con la adición del PLC de Beckhoff.

Con la inclusión de un SCADA se amplían las posibilidades de control al poder comunicarse instantáneamente con la maqueta, obteniendo información sobre el estado de la planta y permitiendo acceso a todos los parámetros necesarios para su visualización y control.

A la hora de llevar a cabo la programación del controlador, se usará el software TwinCAT, basado en Visual Studio, proporcionado por la propia compañía Beckhoff para el desarrollo de sus controladores.

1.2 Abstract

This Final Degree Project (FDP) consists in the implementation of a SCADA system and a proper control environment through a Beckhoff PLC, both to be used in processes that have already been provided their own control devices, with the objective of stablishing a versatile control design that offers more possibilities of monitorization, supervision and control of industrial processes.

The final aim is to have an alternative control environment, based on the Beckhoff technology, allowing for an adequate control of a system through the inclusion of a SCADA system, while also giving new practical experience to future degree students. This way, a wider vision is attained regarding the control systems in the industry, going through the analysis of the control systems on a field and control level, to the addition of supervisory levels with the possibility of connecting to operating levels (MES), within the automation pyramid.

The application will be carried out with the use of the level and flow of liquids plant, observing its behaviour with the ABB controller that the plant has installed by default and adding more control possibilities with the addition of the Beckhoff PLC.

With the addition of a SCADA, the control possibilities are increased by directly linking the SCADA to the plant, obtaining information about the plant's state, and allowing access to the parameters necessary for its control and monitorization.

When programming the controller, the TwinCAT software will be used, software based on Visual Studio, distributed by Beckhoff to develop their own controllers.

Escuela Superior de Ingeniería y Tecnología
Sección Ingeniería Industrial

Trabajo Fin de Grado

Grado en Ingeniería Industrial y Automática

Diseño e implementación de controladores Beckhoff en maquetas de procesos de control

2. MEMORIA

Estudiante: Jesús Elías Soto

Tutor: Santiago Torrez Álvarez

Cotutor: Juan Albino Méndez Pérez

Convocatoria: Junio 2022

2.1 Introducción

Los Controladores Lógicos Programables, comúnmente conocidos como PLCs (Programmable Logic Controller), tienen una historia relativamente corta dentro del entorno de la automatización y el control de sistemas. No fue hasta 1969 [1] cuando surgió el primer PLC, llamado Modicon 084, tras la propuesta enviada a General Motors por Bedford Associates, que proponía una alternativa al uso de sistemas de relés, que, hasta entonces, aunque limitada y poco eficaz debido a su complejidad (generalmente hacían falta cientos de relés para controlar maquinaria, lo que creaba una gran dificultad a la hora de gestionar el cableado y haciendo muy difícil la detección de errores) era la única manera de dar autonomía a la maquinaria con la que se trabajaba.

A partir de aquí, los PLCs fueron evolucionando con la adición de funcionalidad (entradas y salidas analógicas y digitales, funciones matemáticas…), lo que a su vez permitió la creación de nuevos artilugios de programación (en primera instancia, hardware dedicado, para más adelante predominar el uso de software) y comunicación (MODBUS, DeviceNet, Profinet, EtherCat…), así como la aparición de los primeros sistemas SCADA [5], surgidos por la necesidad de automatización a larga distancia. Hoy en día, la utilización de PLCs está muy extendida en la industria, especialmente gracias a su compatibilidad con Ethernet, permitiendo múltiples soluciones de control y manejo de datos a través de la red en una época en la que el IoT y la industria 4.0 se encuentran en alza.

Este trabajo se centrará en el diseño de la lógica de control de un PLC, concretamente de Beckhoff, para una planta de flujo y nivel, así como de un sistema SCADA que permita su comunicación, supervisión y control.

2.2 Motivación

Analizando la pirámide de automatización, es posible hacerse una idea general de cómo funciona un sistema controlado en cada capa, sin embargo, no es suficiente para entrar en específicos.

Al hacer uso de un controlador real, en un entorno controlado de laboratorio, es posible adquirir una perspectiva más concreta sobre la ciencia de la automatización, haciendo factible el desarrollo de sistemas de supervisión al diseñar el controlador, permitiendo al sistema adquirir nuevas capas de automatización y posibilidades de control fácilmente extrapolables a otros sistemas de control.

2.3 Objetivos

2.3.1 Objetivo técnico

El objetivo técnico de este Trabajo Fin de Grado es implementar la lógica de control en un PC embebido de Beckhoff y desarrollar un sistema SCADA, ambos a través del entorno de desarrollo TwinCAT, para llevar a cabo el control de una planta de flujo y nivel, permitiendo tener un controlador alternativo al ABB que usa la planta por defecto, expandiendo el repertorio de enseñanza posible en el laboratorio.

2.3.2 Objetivo académico

El objetivo académico consiste en la aplicación de lo aprendido durante el desarrollo del grado de Ingeniería Electrónica Industrial y Automática para la realización del Trabajo Fin de Grado, así como los mencionados en la Guía Docente de la asignatura:

- "Tener la habilidad de redactar un informe técnico."
- "Tener la habilidad de aplicar de manera integrada de las competencias propias del Grado."
- "Tener la habilidad de hacer una exposición pública."
- "Expresar información técnica en un idioma extranjero tanto de manera escrita como oral."
- "Tener la habilidad de trabajar de manera autónoma y tener iniciativa."

2.4 Normativa

- Norma **UNE-EN 157001:2014** " Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico". AENOR.
- Estándar Internacional **IEC 61131** para Controladores Lógicos Programables.
- Estándar Internacional **IEC 60870** para sistemas SCADA.
- Estándar Internacional **IEC 60715** para equipamiento de montaje.
- Estándar Internacional **IEC 61158** para redes de bus de campo y Ethernet.
- **UNE EN 60068-2-6/EN 60068-2-27** para vibraciones y resistencia a impactos.
- **UNE EN 61000-6-2/EN 61000-6-4** para el control de emisiones magnéticas.

2.5 Alcance

El alcance de este proyecto incluye:

- Programación en Texto Estructurado mediante el software TwinCAT de Beckhoff.
- Recogida y envío de datos a través de entradas y salidas analógicas del controlador CX5130, implementando la electrónica necesaria para la comunicación entre la planta y el controlador para su correcto funcionamiento.
- Desarrollo de un sistema de supervisión tipo SCADA a través de la creación de un proyecto de TwinCAT HMI de Beckhoff.

2.6 Marco teórico

A continuación, se desarrollan los conceptos y definiciones necesarios para la comprensión y elaboración de este Trabajo Fin de Grado.

2.6.1 PLC y PC embebido

Un PLC (*Programmable Logic Controller*) es un controlador industrial basado en microprocesadores con memoria programable capaz de guardar instrucciones y funciones que definen su comportamiento [22]. De forma general están compuestos por:

- Un procesador (CPU), que interpreta las entradas, envía las salidas y ejecuta el programa de control localizado en memoria.
- Un suministrador eléctrico, que convierte el voltaje de CA en CC.
- Una unidad de memoria que guarda los datos de las entradas y el programa ejecutado por la CPU.
- Un módulo de entradas y salidas, permitiendo al controlador recibir y enviar datos.
- Un módulo de comunicación para recibir y transmitir datos a través de la red, ya sea a otros PLC, ordenadores personales o servidores.

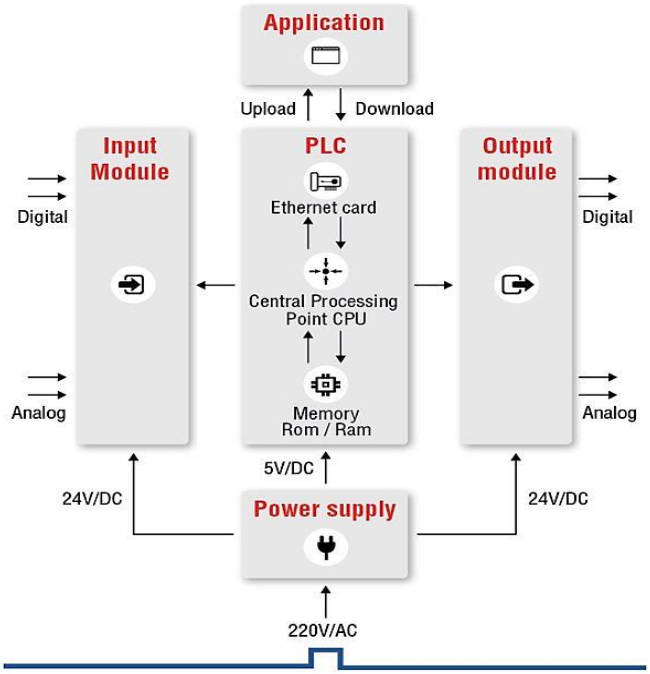


Figura 1.- Arquitectura de un PLC. Figura extraída de [22]

A diferencia de un ordenador personal, los PLCs modernos suelen trabajar con algún Sistema Operativo en Tiempo Real (RTOS), dedicados al procesado de datos y eventos en intervalos de tiempo definidos. Además, al estar diseñados para uso industrial, se fabrican con materiales que permitan su disposición en zonas de producción (zonas de impactos, altas temperaturas…).

A la hora de establecer la lógica de control y comportamiento de un PLC, así como la asignación de entradas y salidas que correspondan a cada variable, es necesaria la intervención de un dispositivo de programación, generalmente en forma de software, que permita acceso a la memoria del PLC con el que se quiera trabajar. Generalmente, las empresas suelen desarrollar su propio software dedicado que permita trabajar con los controladores que fabrican, por lo que no es extraño encontrar diferencias entre cada IDE, sin embargo, todo PLC debe regirse bajo las cláusulas expuestas en el estándar internacional **IEC-61131**, que define, por partes y sin entrar en aspectos específicos:

- La primera parte, publicada en 1992 y revisada en 2003, define los términos usados en este estándar, identificando las principales características funcionales de los PLCs.
- La segunda parte, publicada en 1992 y revisada por última vez en 2007, define los requisitos de funcionalidad y compatibilidad electromagnética necesarios para que un producto cuyo propósito es tener una función en la industria del control, incluyendo PLCs y sus asociados periféricos, pueda ejercer su función de automatización de maquinaria y procesos de control.
- La tercera parte, publicada en 1993, revisada en 2003 y pendiente de una nueva revisión con año de publicación estimado a 2024, define la sintaxis y semántica de los lenguajes de programación para controladores programables, incluyendo: Listas de Instrucciones (IL), Texto Estructurado (ST), Diagrama de Escalera (LD), Diagrama de Bloque de Funciones (FBD) y Lista Secuencial de Funciones (SFC).
- La cuarta parte, publicada en 1995, define la guía de usuario para operarios de PLCs y asiste en la selección y especificación del equipamiento de acuerdo con el estándar IEC 61131.
- La quinta parte, publicada en el 2000, define las comunicaciones con los controladores programables.
- La sexta parte, publicada en 2012, define los parámetros de seguridad necesarios para la operación de PLCs en sistemas eléctricos, electrónicos o electrónicos programables.

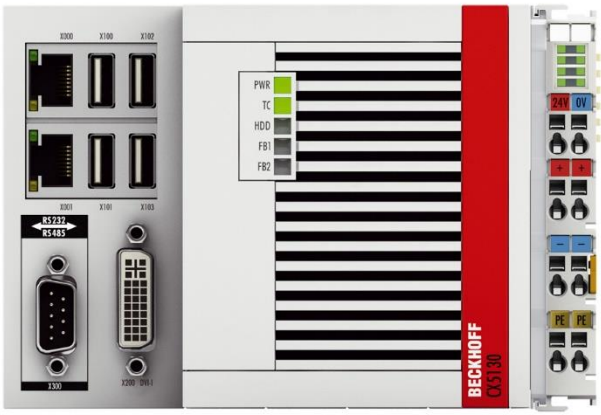


Figura 2.- PC embebido de Beckhoff CX5130. Figura extraída de [24]

El caso de los PCs (Programmable Controllers) embebidos de Beckhoff, los usados para la realización de este TFG, no son una excepción, deben cumplir con la normativa descrita.

Un sistema embebido se define como un producto similar a un ordenador en cuanto a arquitectura, pero que realiza una función específica dentro de un sistema mecánico o electrónico. Debido a esto, se pueden diseñar para reducir su tamaño y coste e incrementar su eficiencia y productividad. En concreto, los PCs embebidos de Beckhoff [24], específicamente los de la serie CX presente en el laboratorio, presentan un diseño compacto y están fabricados para ser resistentes mecánicamente y contra temperaturas, lo que les permite estar ubicados en entornos de vibraciones y golpes intensos, así como en ambientes fríos y calientes (generalmente entre -25ºC y +65ºC). En cuanto a los módulos de Beckhoff de Entrada/Salida, estos pueden ser montados en el carril DIN y conectados directamente a la derecha del PC embebido sin necesidad de usar acopladores, permitiendo ahorro de espacio y coste.

El PC ubicado en el laboratorio se encuentra montado sobre un carril que cumple con el Estándar Internacional **IEC 60715** que define sus dimensiones.

2.6.2 SCADA

Los sistemas denominados "*Supervisory Control and Data Acquisition*", comúnmente conocidos como SCADA, son sistemas de control compuestos por elementos de software y hardware que permiten, para un proceso de control, obtener, procesar, monitorizar y archivar datos en tiempo real, controlar aplicaciones industriales de forma local o remota, e interactuar con sensores y actuadores.

Generalmente suelen instalarse en servidores, permitiendo que un operador pueda acceder desde una planta automatizada a los controles integrados en el correspondiente SCADA en forma de una HMI (*Human Machine Interface)*, es decir, un panel de control que permita al operador interactuar con la planta.

El diseño recae sobre el programador al cargo de su desarrollo, siendo posible configurarlo de modo que la visualización de los datos de la instalación sea de fácil acceso para cualquier operador.

Estos sistemas se encuentran en la capa intermedia de la pirámide de automatización:

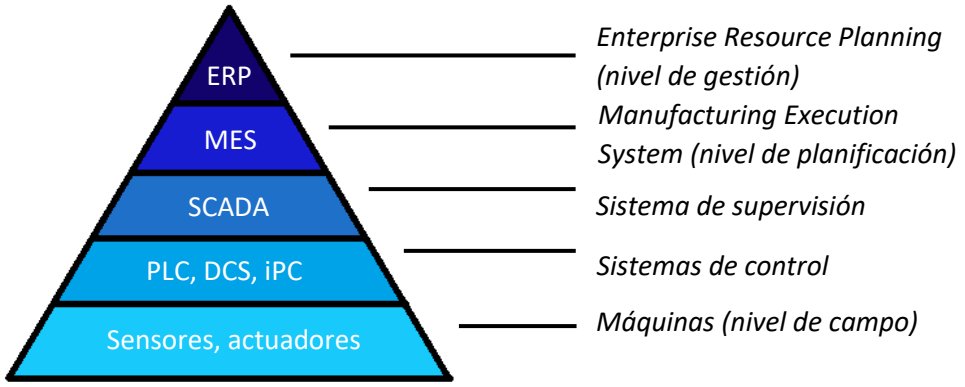


Figura 3.- Pirámide de Automatización

Dicha pirámide permite una visualización completa de la tecnología y datos que se utilizan en la industria, de manera que los tres niveles inferiores proporcionan la información necesaria para la toma de decisiones de los dos niveles superiores. Es por esto, entre otros motivos, que la información recogida suele enviarse a servidores dedicados o a la nube, donde se recogen en bases de datos, permitiendo su acceso de forma rápida y segura.

Un sistema SCADA consistirá [13], generalmente, de los siguientes elementos:

- **Ordenadores de supervisión:** Son los dispositivos, hardware y software, encargados de comunicarse con los elementos de campo, ya sean Unidades Terminales Remotas (RTUs) y/o PLCs, además de incluir el software que permite el uso de las HMI. Son el núcleo del sistema SCADA.
- **Unidades Terminales Remotas:** Unidades conectadas a sensores y actuadores, envían información al SCADA y permiten controlar los procesos.
- **Controladores Lógicos Programables:** Se conectan a los sensores y actuadores del proceso y provee la información necesaria para que un SCADA pueda efectuar la automatización de un proceso.
- **Interfaz Hombre-Máquina:** Es la interfaz que permite el control del proceso, así como permitir la visualización de datos, además de reportar la información recogida por los RTUs y PLCs en tiempo real, facilitando al operador ejercer control sobre la planta.
- **Infraestructura de comunicación:** Establece la conexión entre el sistema de supervisión y las unidades de control a través del uso de protocolos industriales.

Otra de las funciones que ofrecen los SCADA es sistemas de alarmado, proporcionando avisos al personal que corresponda de forma sonora, textual, generando e-mails, entre otras.

Antes de llegar a los sistemas SCADA de hoy en día, esta tecnología tuvo que pasar por varias generaciones [30], evolucionando al ampliar sus funciones y conectividad en cada etapa:

Generación	Arquitectura	Funcionalidad
1	Monolítica	Todo el procesado de datos, interfaz, errores, son gestionados por una única plataforma.
2	Distribuida	Las funciones del sistema SCADA están distribuidas en varias estaciones conectadas a través de una red local (LAN).
3	Conectada	Similar a la segunda generación, sin embargo, el sistema está conectado a través de una red de control de proceso (PCN), compuesta de múltiples LANs, ampliando el rango de la infraestructura.
4	Basada en la Web	Llegada de la nube y tecnologías IoT, permitiendo gran conectividad y la implementación de interfaces en la Web.

Tabla 1.- Generaciones de SCADA.

Con la llegada de aplicaciones **IIoT** (*Industrial Internet of Things*), surge la duda de qué sistema es más eficiente y accesible a la hora de realizar labores de supervisión y manejo de datos, ya que ciertas aplicaciones permiten realizar las mismas funciones que tradicionalmente realizaría un SCADA sin algunas de sus limitaciones. Sin embargo, existe una tendencia que tiende a incrementar, y es la de implementar ambas soluciones de forma híbrida, permitiendo ampliar la funcionalidad de los SCADA como, por ejemplo, con el análisis de mediciones y datos en la nube.

El desarrollo de sistemas SCADA viene acotado por el Estándar Internacional **IEC 60870**, que define, sin entrar en aspectos específicos: Información general, condiciones de operación, interfaces eléctricas, requisitos de rendimiento y protocolos de transmisión de datos.

2.6.3 TwinCAT

Los fabricantes de autómatas y controladores suelen proponer software dedicado para la configuración de sus PLCs. Este es el caso de TwinCAT (*The Windows Control and Automation Technology*), desarrollado por Beckhoff para permitir la configuración de sus **IPCs** (*Industrial Programmable Controllers*). Descargable desde su página web [19], TwinCAT, concretamente TwinCAT 3, permite el control en tiempo real de PCs y, añade, gracias a su flexibilidad debido a su integración en Visual Studio®, lenguajes de programación como C/C++, C#, VB.NET y posibilita el enlace con MATLAB®/Simulink®, dando al desarrollador un amplio margen de posibilidades de control.

TwinCAT 3 está compuesto por numerosas extensiones, funciones y utilidades que proporcionan al programador diversas opciones a la hora de desarrollar software.

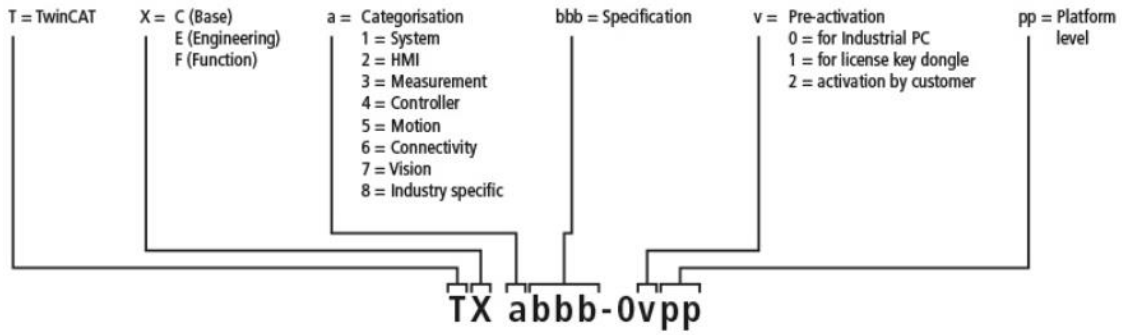


Figura 4.- Orden informativo de funcionalidad de TwinCAT. Figura extraída de [31]

La instalación del entorno de desarrollo viene dada por TwinCAT 3 XAE (*eXtended Automation Engineering*), que corresponde a **TE1000 XAE** e incluye un Shell de Visual Studio®, así como un compilador y librerías con las que empezar a programar. Al ser la instalación básica, incluye las herramientas de control necesarias para la programación del PLC.

La solución que propone Beckhoff para el desarrollo de SCADA y HMI viene dada por TwinCAT HMI, corresponde a **TE2000|HMI Engineering**, software que permite la ejecución de clientes de HMI en navegadores desarrollados en **HTML5**, permitiendo al cliente ejecutarse no solo en ordenadores personales, sino también en dispositivos móviles.

La comunicación entre la HMI y el controlador puede hacerse a través de protocolos **ADS** o de **OPC UA** (véase ***2.6.4 Comunicación*** para más información).

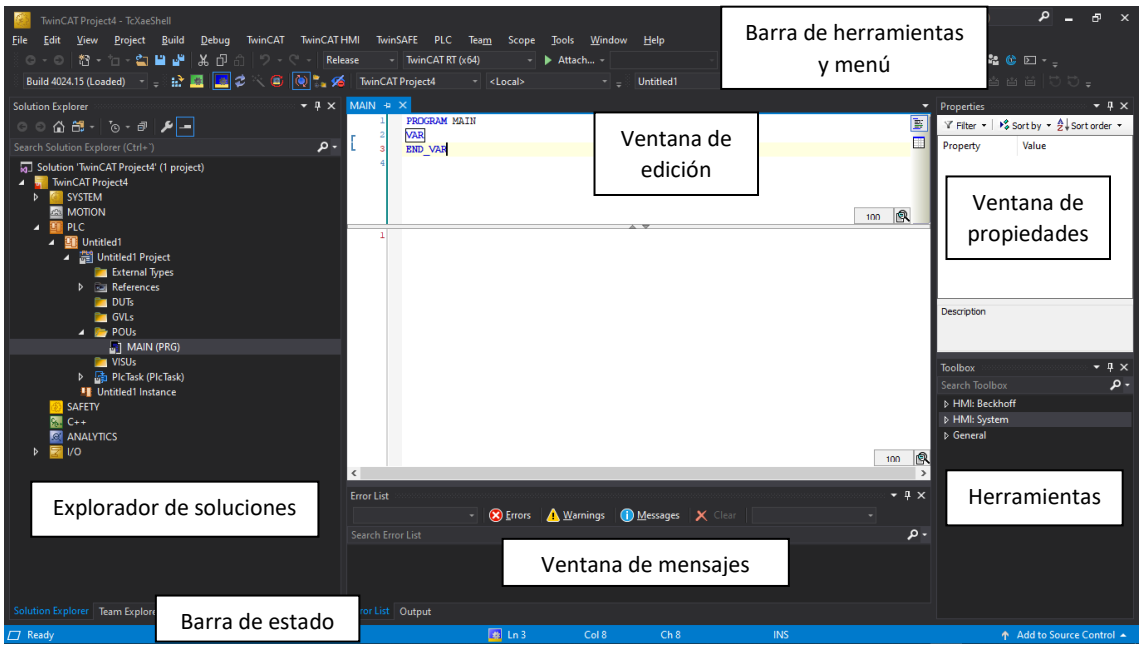


Figura 5.- Entorno de desarrollo de TwinCAT.

Además del entorno de programación dado por XAE, es necesario que el PLC incluya un **XAR** (eXtended Automation Runtime) [65], ofreciendo un entorno en tiempo real donde puedan cargarse y ejecutar los módulos de TwinCAT de forma cíclica. La comunicación entre ambos se da a través del protocolo ADS. Una vez compilado un programa en TwinCAT en la máquina con XAE instalado, se generará un ejecutable binario que podrá transferirse a la máquina con la instalación de XAR y ejecutarse en tiempo real. Debido a que la propia instalación de XAE incluye al XAR, es posible ejecutar programas de TwinCAT de forma local, permitiendo ajustar el software apropiadamente antes de ser cargado a una máquina remota.

Debido a la necesidad de presentar datos de forma gráfica con el objetivo de analizar comportamientos de control ejercidos sobre el PLC, el uso de software de medición que permita su visualización es una exigencia. Para ello, Beckhoff proporciona TwinCAT Scope View, que prepara los datos que se quieren observar de forma gráfica y, gracias a la extensión **TF3300|Scope Server**, es posible la implementación de dichas gráficas en entornos HMI.

A parte de las ya mencionadas, existen otras soluciones TwinCAT para multitud de entornos, entre los que se encuentran:

- **Control de movimiento:** Corresponde a MOTION en el explorador de soluciones y ofrece soluciones al problema de control de servomotores. (**TF5xxx**)
- **Visión:** Para el procesado de imágenes, así como la configuración de cámaras. (**TF7xxx**).

Todas las funciones que existen en TwinCAT requieren una licencia para su uso, que pueden ser activadas directamente desde el propio IDE de TwinCAT de forma gratuita (licencias de testeo) y automática tras rellenar un CAPTCHA (una semana de duración, tras ella habrá que activarlas de nuevo). En caso de que se quiera ejecutar un programa en un PLC de forma permanente, se deberá pagar por la licencia que corresponda. Esto permite a los desarrolladores ahorrar en coste de licencias hasta que se haya llegado a un punto en el que el software pueda ser comercializado.

Cabe mencionar que en este proyecto solo se trabajará con TwinCAT HMI, TwinCAT Scope View y TwinCAT PLC.

2.6.4 Comunicación

Los protocolos de comunicación son el medio por el que varios dispositivos pueden conectarse y comunicarse unos con otros al permitir el intercambio de datos. Dentro de la industria del control y la automatización, la gran variedad de protocolos de comunicación de PLCs que existen permiten flexibilidad a la hora de establecer una red de comunicación entre dispositivos, pudiendo satisfacer los requerimientos del sistema de control [34].

Protocolo	Descripción
Ethernet/IP	Provee un mecanismo de envío de datos en ambas direcciones entre PLC y dispositivo remoto. Generalmente usado por PLCs de Allen Bradley, Schneider Electric y Omron.
Ethernet TCP/IP	Transmite paquetes de datos compuestos por comandos que permiten la lectura y escritura en memoria.
Modbus RTU	Fácil uso y fiable, muy usado en Sistemas de Automatización Industrial y robótica. Usa codificación binaria y verifica errores CRC.
Modbus TCP/IP	Es un protocolo que gestiona el método de envío de datos entre varias capas sin ser afectado por el protocolo usado en la siguiente capa más inmediata. Combina una red física, que sería el Ethernet, con un estándar de red (TCP/IP) junto con el estándar Modbus de representación de datos.
Profibus.	Estándar de comunicación encargado de la conexión entre sensores y sistemas de control.
Profinet	Surge debido a la necesidad de integrar Profibus a estándares como Ethernet. Basado en TCP/IP, permite el intercambio de datos entre controladores y dispositivos.

Tabla 2.- Ejemplos de protocolos de comunicación en la industria.

En el caso de los iPCs de Beckhoff, el protocolo desarrollado por la misma empresa a partir de Ethernet, EtherCAT, se emplea como bus de campo, aunque es posible usar prácticamente cualquiera ya que existen módulos de hardware y drivers software que permiten adaptarlos a las especificaciones de TwinCAT y PLCs de Beckhoff.

Una red EtherCAT está constituida por un maestro EtherCAT, que envía datos a uno o varios esclavos EtherCAT [41]. El maestro se encarga del envío de telegramas EtherCAT que son interceptados por cada esclavo, encargados de leer los datos enviados, así como transmitir su propia información al maestro.

En este TFG, el maestro EtherCAT usado es un PC embebido de Beckhoff CX5130, compuesto por cuatro esclavos EtherCAT, que equivalen a módulos de entradas y salidas analógicas y digitales (véase ***2.7 Material/hardware/software empleado*** para más información).

En el entorno de TwinCAT, el protocolo de comunicación por defecto es ADS (*Automation Device Specification)*. Este protocolo permite el intercambio de datos y la comunicación con cualquier módulo software de TwinCAT.

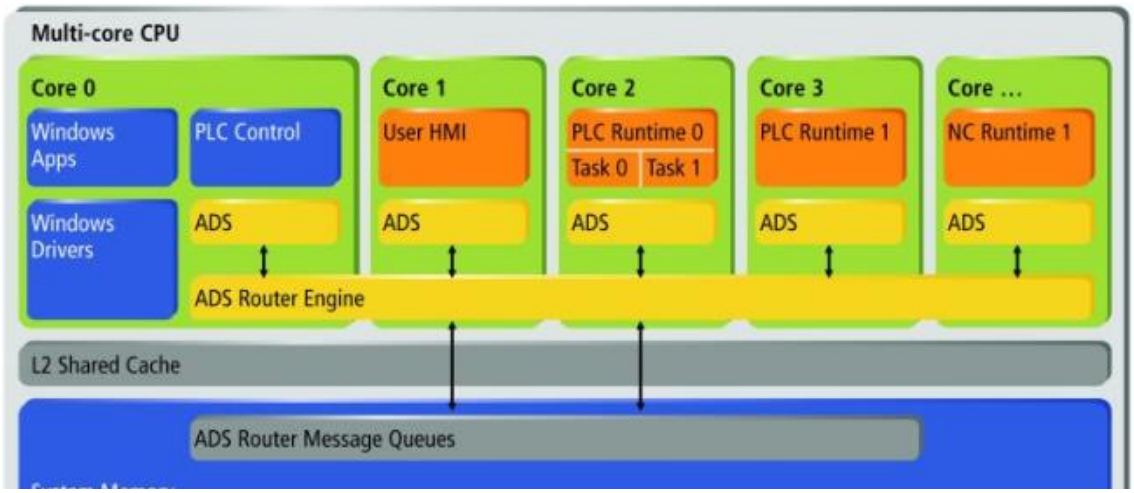


Figura 6.- Representación de arquitectura ADS en TwinCAT. Figura extraída de [44]

Si se requiere comunicación con otro dispositivo, ADS puede utilizar TCP/IP como medio para establecer un puente de comunicación.

2.6.5 Amplificador operacional

El uso de amplificadores operacionales permitirá ajustar los voltajes necesarios para establecer las conexiones entre la planta y el controlador.

Un amplificador operacional es un dispositivo electrónico que, alimentado con corriente continua, permite la amplificación de diferencia de potencial entre el voltaje a su entrada y su salida.

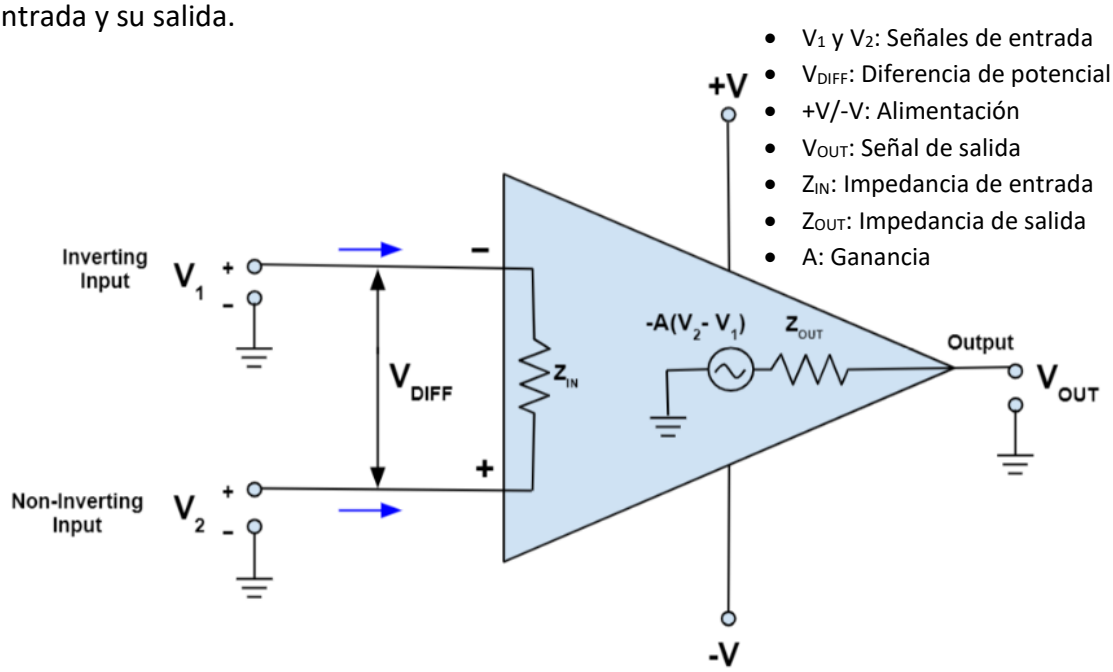


Figura 7.- Amplificador operacional. Figura extraída de [48]

Estos dispositivos tendrán un comportamiento especifico en función de si están realimentados o no. En caso afirmativo, son denominados amplificadores en configuración de lazo cerrado, la señal de salida será predecible y la realimentación negativa, estabilizando la ganancia y permitiendo su cálculo mediante componentes externos como resistencias, condensadores…

Aplicaciones comunes de amplificadores operacionales según su topología pueden observarse en la siguiente tabla:

Lazo	Aplicación	Descripción
Abierto	Comparador	Compara las señales de entrada para proporcionar una salida en función de cuál sea mayor.
Cerrado	Seguidor de tensión	Proporciona una salida equivalente a la señal de entrada.
Cerrado	Amplificador no inversor	La tensión de salida varía con la misma polaridad que la tensión de entrada.
Cerrado	Amplificador inversor	La tensión de salida varía con una polaridad contraria a la tensión de entrada.

Tabla 3.- Aplicaciones de amplificadores operacionales.

2.6.6 Controladores PID

Este tipo de controlador será empleado para regular el nivel del tanque de la planta de nivel y flujo.

Un controlador Proporcional-Integral-Derivativo (**PID**) es un mecanismo de lazo de control que emplea la retroalimentación para calcular un valor de error fruto de la diferencia entre el valor deseado (consigna) y el valor medido (variable de proceso) y aplica una corrección creando una salida con la adición de los valores proporcionales, integrales y derivativos [53].

La función de control se representa con la siguiente ecuación:

$$u(t) = K_P e(t) + K_i \int_0^t e(T)dT + K_d \frac{de(t)}{dt}$$

Donde:

- **K_P:** Representa el coeficiente proporcional.
- **K_i:** Representa el coeficiente integral.
- **K_d:** Representa el coeficiente derivativo.
- **e(t):** Representa el error.

También suele representarse como:

$$u(t) = K_P \left(e(t) + \frac{1}{T_i} \int_0^t e(T)dT + T_d \frac{de(t)}{dt} \right)$$

Cada parámetro tiene una función específica dentro del controlador. En la siguiente tabla se puede observar el comportamiento de cada coeficiente dentro del sistema:

Término	Representación	Descripción
Proporcional	$P = K_P e(t)$	Crea una señal de salida proporcional a la magnitud de la señal de error.
Integral	$I = K_i \int_0^t e(T)$	Genera una salida en función a la duración y magnitud del error. En la mayoría de los casos, anula el error en régimen permanente. Por lo general, acelera la velocidad a la que el sistema llega a la consigna.
Derivativo	$D = K_d \frac{de(t)}{dt}$	Genera una salida en función al ritmo de cambio de la señal de error. Predice el comportamiento del error del sistema, amortiguando las oscilaciones que se pudieran producir entorno a los valores de consigna.

Tabla 4.- Coeficientes de controlador PID.

2.6.7 Texto Estructurado

Este lenguaje de programación se utilizará para desarrollar un programa en el software de TwinCAT que satisfaga los objetivos propuestos.

Texto Estructurado (*Structured Text*) es uno de los lenguajes de programación recogidos en el estándar IEC 61131-3 [58]. A diferencia de otros lenguajes que se encuentran dentro del mismo estándar, es un lenguaje textual sintácticamente equivalente a Pascal, en el que se basa. Está desarrollado para parecer sintácticamente similar a otros lenguajes de alto nivel como C o Python e incluye, como estos, bucles, condiciones, operadores…

A la hora de programar en Texto Estructurado, si se quieren declarar variables, condiciones, bucles… se deberá definir un espacio de uso para cada declaración.

Por ejemplo:

```
VAR
        Testeo : INT;
END_VAR
```

Como se puede observar, el espacio de variables para este ejemplo queda delimitado entre las palabras reservadas VAR y END_VAR. Generalmente no es necesario hacer esto para funciones, bloques de funciones y programas ya que el software de programación delimita automáticamente los limites según cuanto código haya escrito el desarrollador.

El código desarrollado será llamado en cada ciclo del PLC, ejecutando una a una cada línea de código.

Existen numerosos tipos de datos a disposición del programador, entre los que cabe destacar por ser de uso más común:

Tipo	Palabra reservada
Entero	SINT, INT, DINT, LINT, UINT, USINT, LDINT, ULINT
Punto flotante	REAL, LREAL
Tiempo	TIME, DATE, TIME_OF_DAY, DATE_AND_TIME
Cadena de caracteres	STRING
Cadena de bits	BOOL, BYTE, WORD, DWORD, LWORD

Tabla 5.- Tipos de datos en Texto Estructurado.

Para operar con estos datos es necesario el uso de operadores entre los que se incluyen:

- **Aritméticos:** Adición (+), resta (-), multiplicación (*), división (/), módulo (MOD).
- **Relación:** Igual (=), menor (<), menos o igual (<=), mayor (>), mayor o igual (>=), diferente (<>).
- **Lógicos/para bits:** &/AND, OR, XOR, NOT

La ventaja de uso de Texto Estructurado sobre otros lenguajes de programación del estándar viene de su facilidad de lectura y rapidez a la hora de programar ciertas instrucciones, que pueden verse en la siguiente tabla:

Instruction	Example
Assignment	A:=B; CV := CV + 1; C:=SIN(X);
Calling a function block and use of the FB version	CMD_TMR(IN := %IX5, PT := 300);A:=CMD_TMR.Q;
RETURN	RETURN;
IF	IF D<0.0 THEN C:=A; ELSIF D=0.0 THEN C:=B; ELSE C:=D; END_IF;
CASE	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE;
FOR	FOR I:=1 TO 100 BY 2 DO IF ARR[I] = 70 THEN J:=I; EXIT; END_IF; END_FOR;
WHILE	WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;
REPEAT	REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	EXIT;
Empty instruction	;

Tabla 6.- Instrucciones en Texto Estructurado.

2.7 Material/hardware/software empleado

Para la realización de este proyecto TFG se ha hecho uso de varios elementos de hardware y software concretos para la puesta en marcha del ejercicio de control de una maqueta de regulación de flujo y nivel. A continuación, se muestra todo el material pertinente al TFG.

2.7.1 Maqueta de control

Planta de control de flujo y nivel (*véase **5.1 Anexo I** para más información*) compuesta por dos tanques de proceso conectados a válvulas manuales y solenoides. El agua es bombeada a través del sistema, pasando por un caudalímetro y una válvula de control motorizada (servo válvula) antes de llegar al tanque de proceso, donde es posible medir el nivel del tanque mediante un potenciómetro.

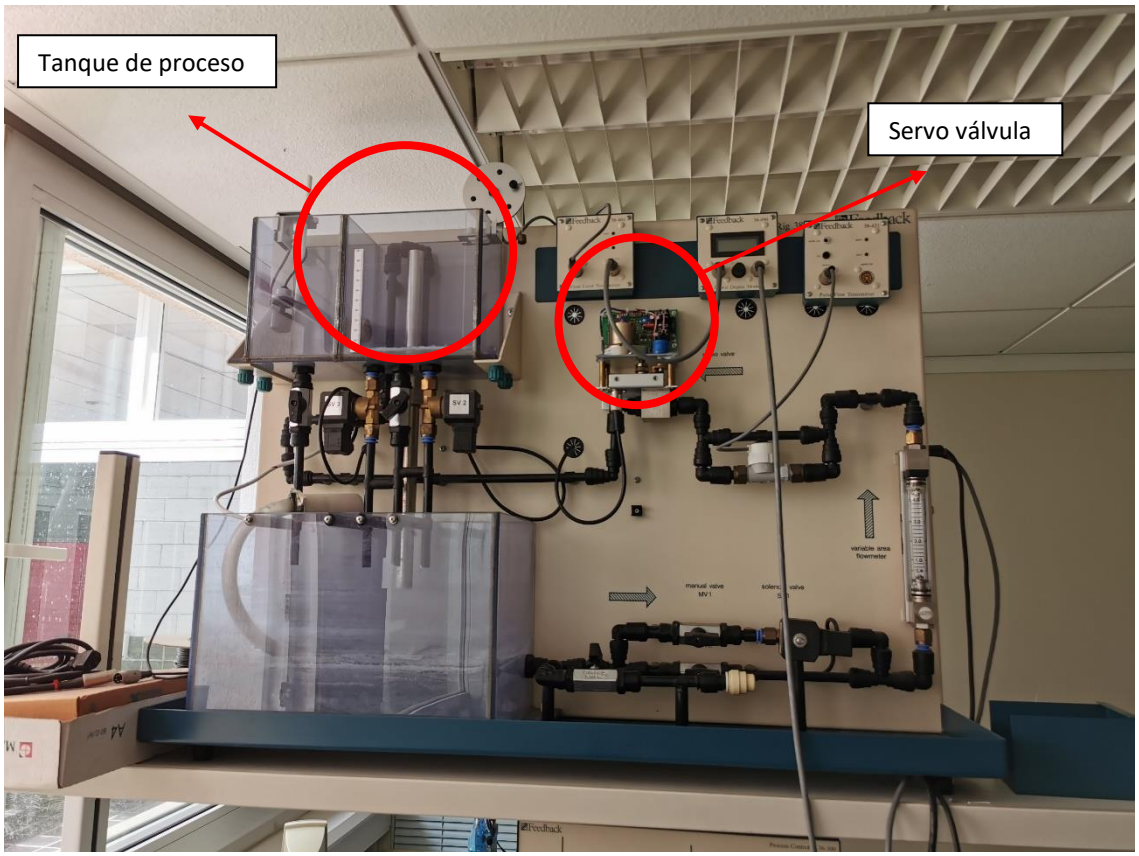


Figura 8.- Planta de control de proceso.

2.7.2 iPC de Beckhoff

Se hace uso de un PC embebido de Beckhoff, modelo CX5130 (*véase **5.2 Anexo II** para más información*), conectado a dos módulos de entrada y dos módulos de salida, unos analógico y otros digitales, denominados:

- **EL1008:** Módulo de entrada digital compuesto por 8 canales.
- **EL2008:** Módulo de salida digital compuesto por 8 canales.
- **EL3064:** Módulo compuesto por 4 entradas analógicas.
- **EL4002:** Módulo compuesto por 2 salidas analógicas.

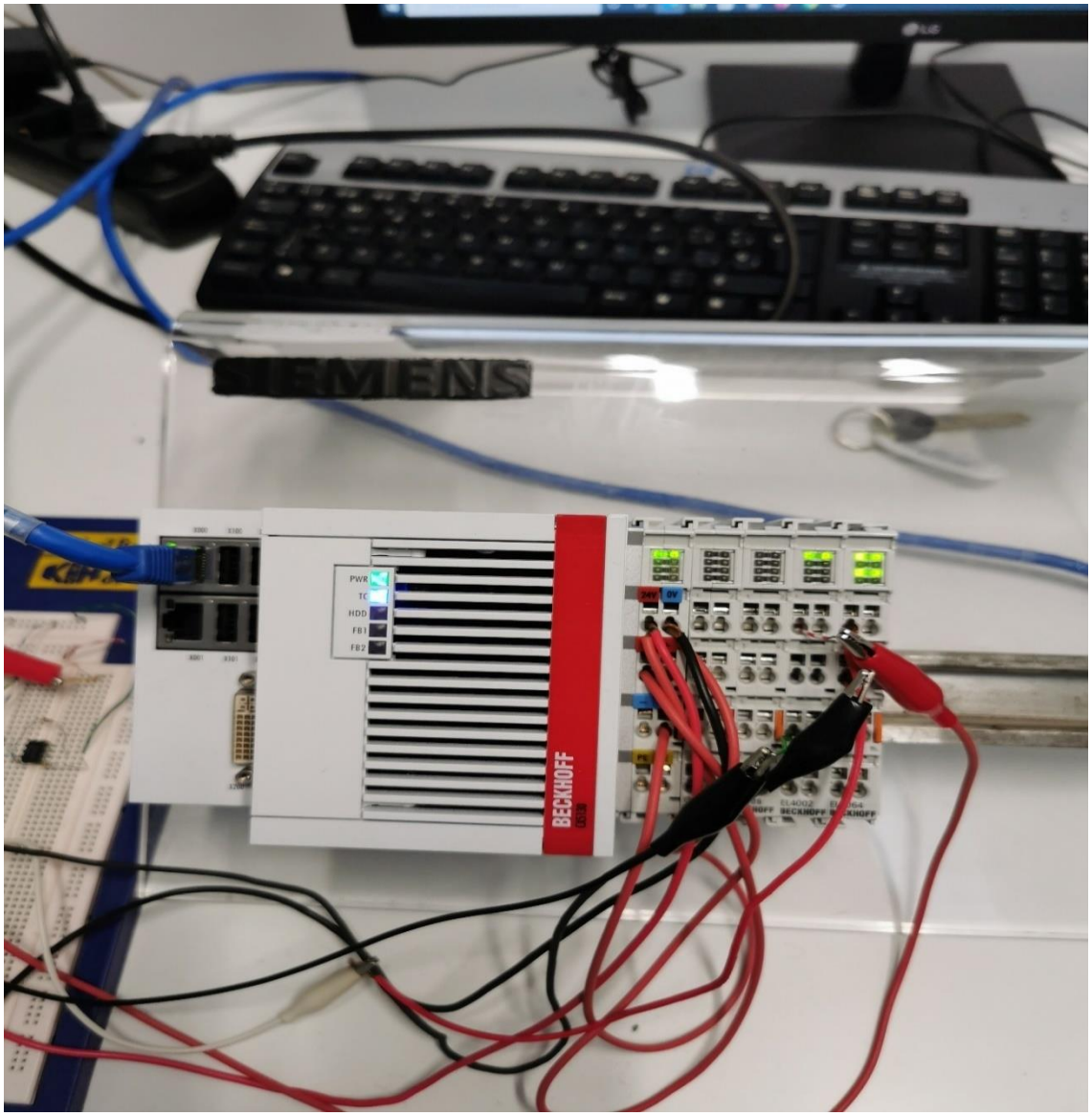


Figura 9.- PC embebido CX5130.

Concretamente se usan los módulos analógicos, de los que cabe destacar que trabajan con una resolución de 16 bits, es decir $2^{16} = 65536$. Según la representación escogida, los valores mostrados por los módulos se representarán como un valor entre -32768 y 32768 ó 0 y 65536, que deberá ser convertido a unidades legibles por el usuario.

2.7.3 Circuito de conversión

Circuito diseñado para establecer la conexión física entre la planta y el controlador, montado sobre una *protoboard* (*véase* ***2.8 Desarrollo del TFG*** y ***5.3 Anexo lll*** *para más información*).

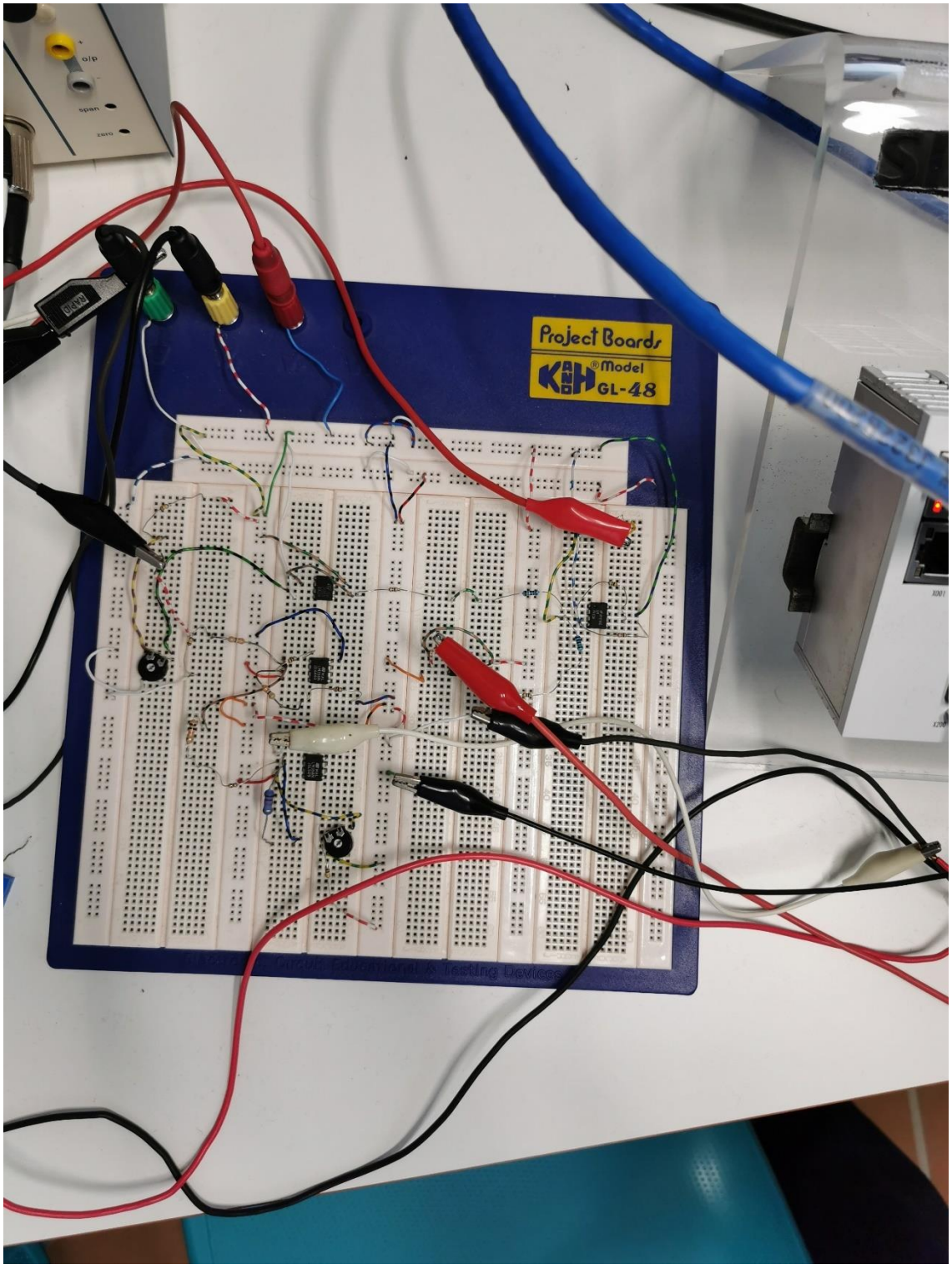


Figura 10.- Circuito de conversión.

2.7.4 Fuente de alimentación

Provee de tensión al circuito de conversión y al PC embebido de Beckhoff.

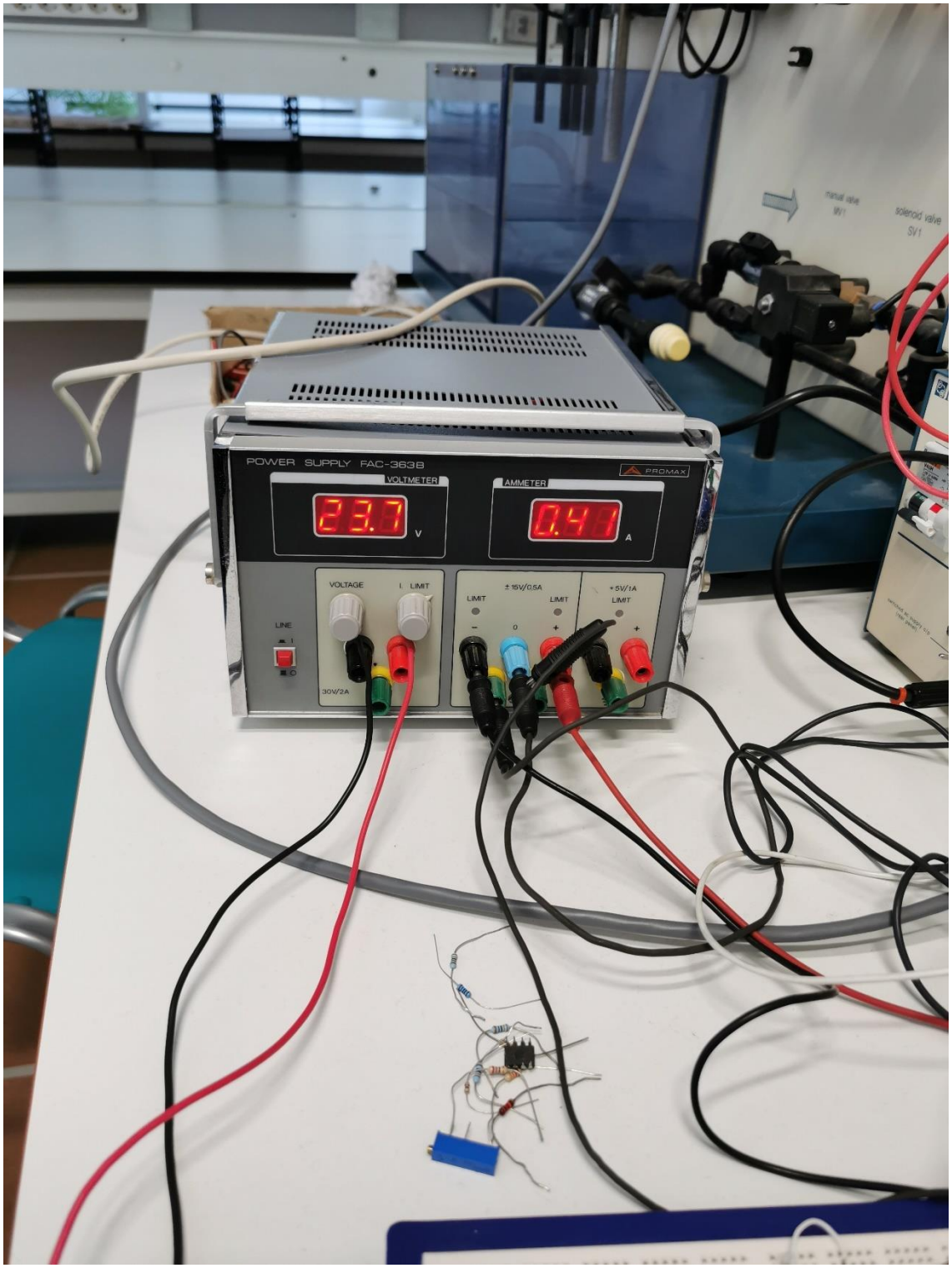


Figura 11.- Fuente de alimentación.

2.7.5 Interfaz de proceso 38-200

Interfaz conectada al sensor de nivel de la planta, así como a la servoválvula. Permite la conexión entre la planta y el controlador a través del circuito de conversión (*véase **5.1 Anexo I** para más información*).

Figura 12.- Interfaz 38-200.

2.7.6 Multímetro

Usado para la medición de voltajes y tensiones en el circuito de conversión antes del establecimiento de conexión entre la planta y el controlador.

Figura 13.- Multímetro.

2.7.7 TwinCAT

Software empleado para la programación del PC embebido de Beckhoff, así como para el desarrollo del SCADA. La versión usada en este proyecto corresponde a TwinCAT 3 **v4024.15**.

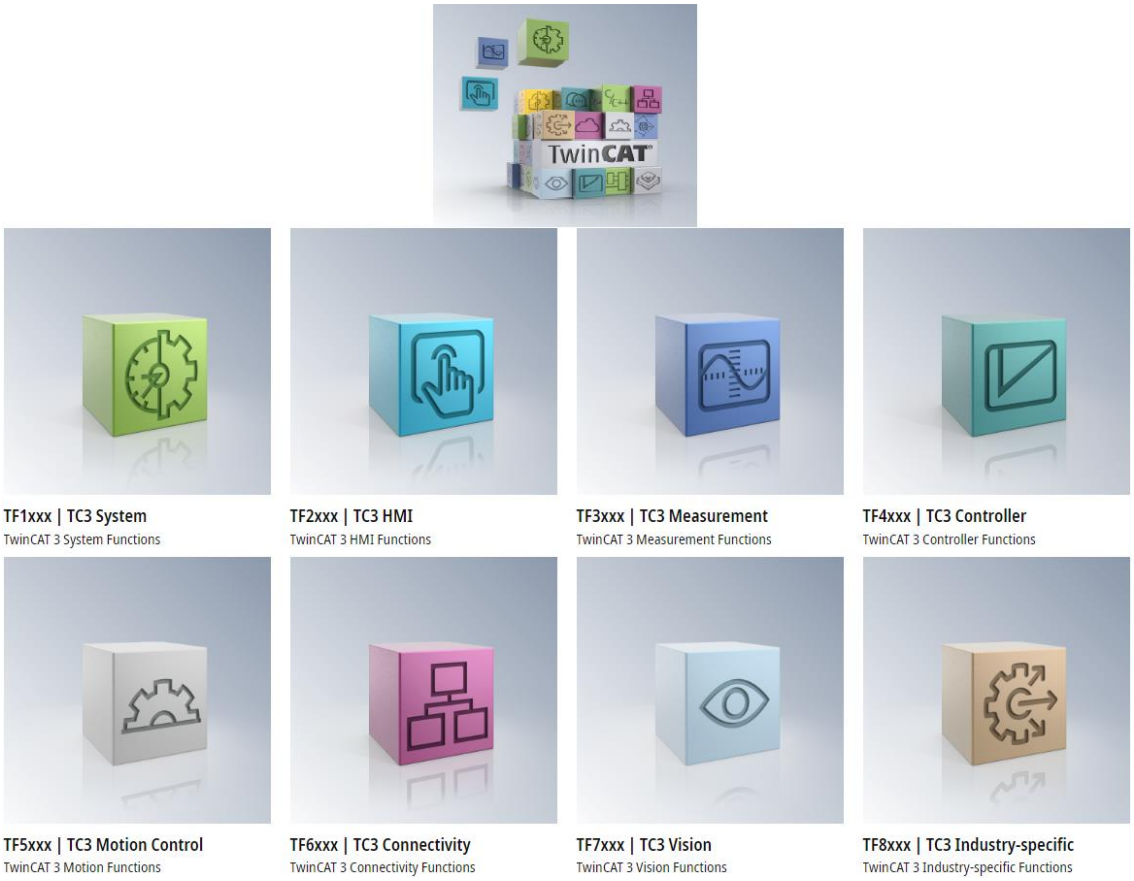


Figura 14.- TwinCAT y complementos. Figura extraída de [45]

2.7.8 LTspice XVII

Software de diseño y simulación de circuitos electrónicos utilizado para el modelado del circuito de conversión (disponible para visualización en ***5.3 Anexo III***).

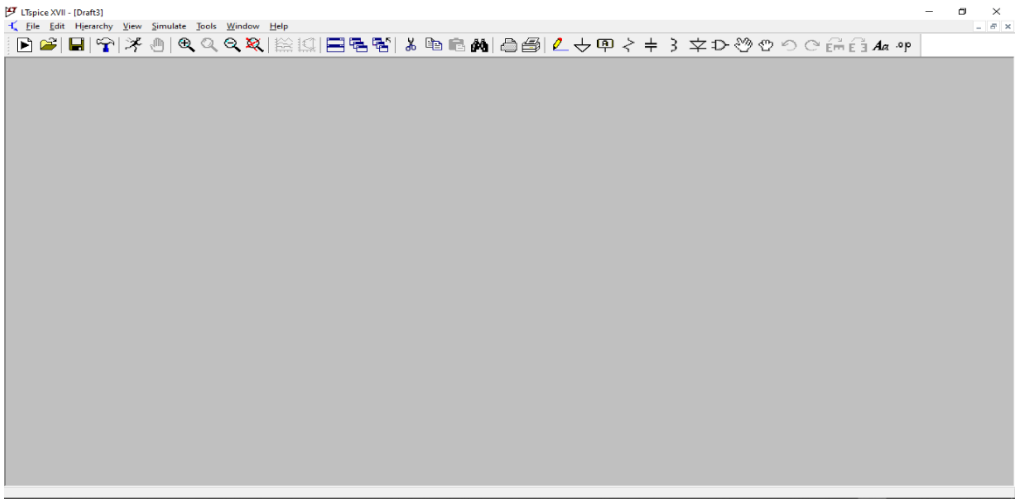


Figura 15.- Interfaz de LTspice XVII.

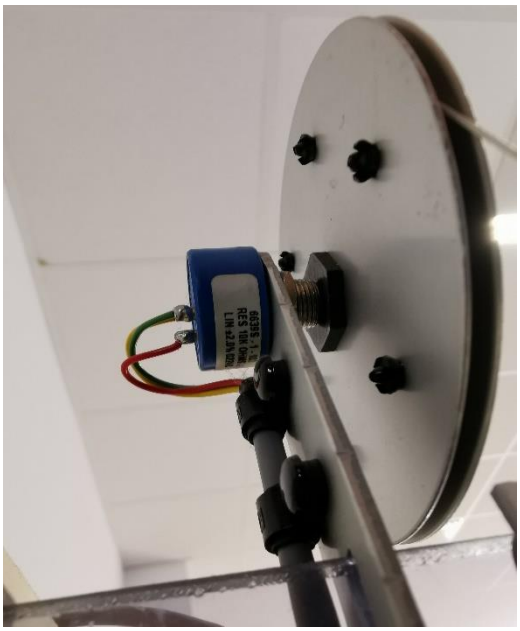
2.8 Desarrollo del TFG

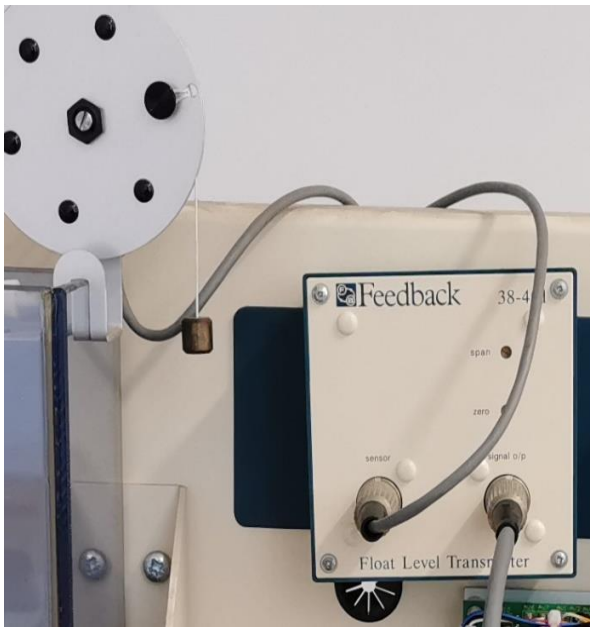
La realización de este Trabajo Fin de Grado viene marcada por distintas etapas diferenciables por su contenido y conceptos tratados, todos vistos en el Grado en Ingeniería Electrónica Industrial y Automática. A continuación, se expone la realización del TFG, incluyendo breves aclaraciones teóricas donde hiciese falta y dividido por etapas en función del contenido con el que se esté tratando.

2.8.1 Circuito de conversión analógica

Al realizar la primera toma de contacto con el material del TFG y viendo el funcionamiento de la planta con su controlador instalado por defecto (ABB), se puede observar el comportamiento que debe tomar la planta al realizar la labor de control con dicho controlador, dando una idea inicial de cómo desarrollar el proyecto.

Analizando este comportamiento, se observa que el control de la servoválvula conectada al tanque de proceso va a ser dependiente al nivel detectado por el sensor, variando su apertura en función de la cercanía a la consigna que se desea. Por lo tanto, se puede concluir que, siendo el **nivel del tanque la variable controlada**, será necesario estimular apropiadamente a la servoválvula para obtener un control estable y rápido del proceso de llenado o vaciado. Sin embargo, antes de pensar en que tipo de controlador se va a implementar, es necesario analizar el cableado y circuitería que se va a utilizar para establecer la conexión entre el PC embebido de Beckhoff y la planta de nivel. Este análisis surge debido a que la planta (concretamente la ***interfaz de proceso***) trabaja con un rango de corrientes de **4-20 mA**, es decir, tanto la señal del sensor procesada a través del transmisor (que interpreta en miliamperios el nivel detectado por el sensor de nivel) como el estímulo necesario para el movimiento de la servoválvula. Las señales de intensidad presentan mayor inmunidad frente a interferencias electromagnéticas, permitiendo mayor fiabilidad a la hora de transportar información de sensores.





Figuras 16 y 17.- Sensor de nivel (potenciómetro) y Transmisor 38-401.

Como los módulos de entrada y salida analógicos del controlador Beckhoff del laboratorio funcionan con un rango de **0-10 V**, es necesario realizar una conversión que permita la conexión entre planta y controlador de forma apropiada. Por ello, se efectúa el diseño de un circuito electrónico que permita la conversión de 4-20 mA a 0-10 V para el módulo analógico de entrada (donde es detectada la señal del sensor de nivel) y otro para la conversión de 0-10 V a 4-20 mA para el módulo de salida analógico (donde se transmite la señal de control de la servoválvula).

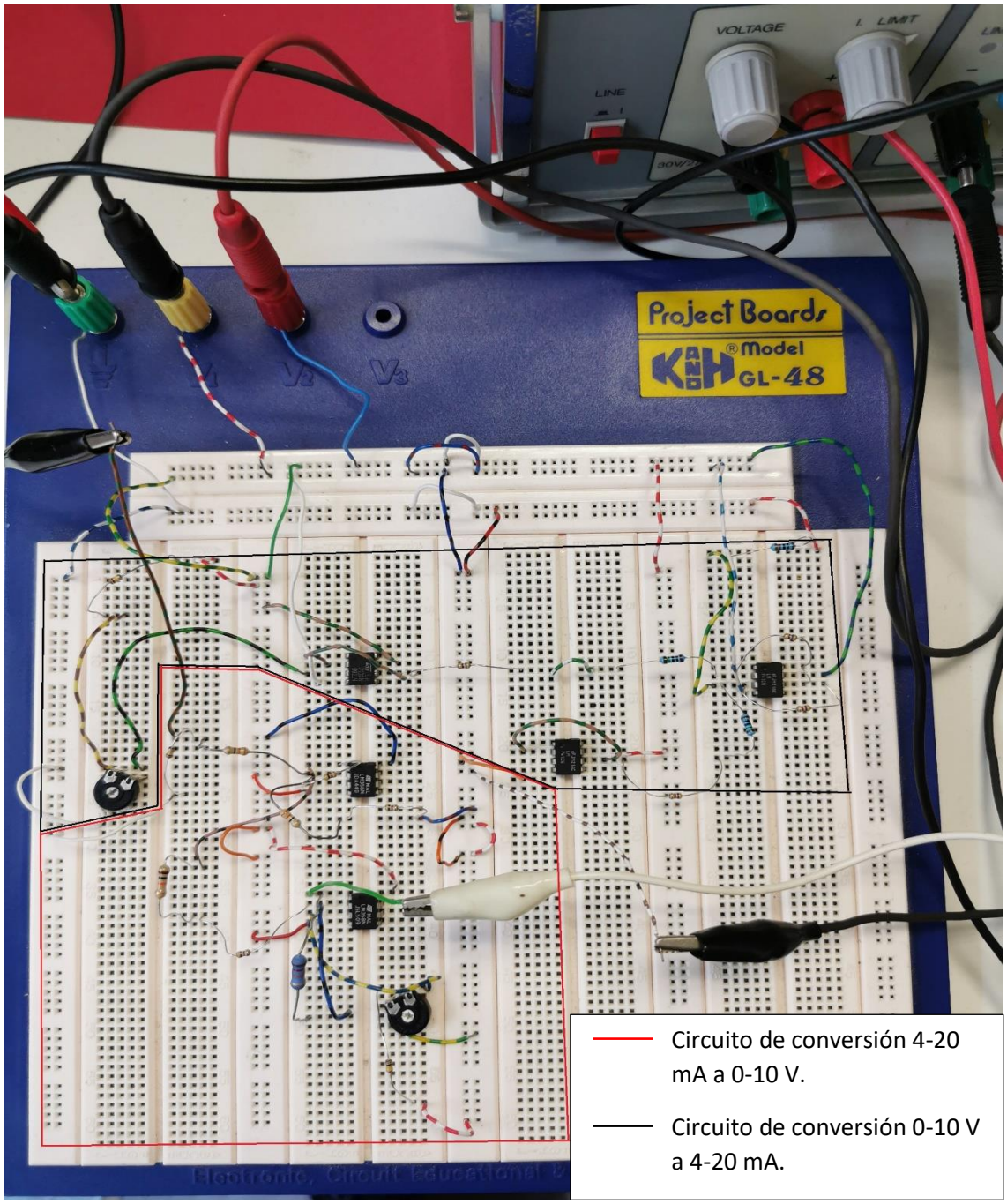


Figura 18.- Circuitos de conversión analógica.

Los amplificadores operacionales, que son alimentados a ± 15 V, nos permiten el ajuste de tensión necesario para la correcta conversión que corresponda.

Los específicos del circuito pueden observarse en ***5.3 Anexo lll***.

2.8.2 Prueba de conexión

Una vez ajustado el circuito de conversión analógica, podemos probar los módulos de entrada y salida analógicos EL3064 y EL4002, aunque no sin antes proveer al PC embebido de alimentación. Esta alimentación debe ser proporcionada a través de una corriente continua de + 24 V (admisible entre -15%/+20%), por lo que hacemos uso de la fuente de alimentación, alimentando al PC embebido y a los módulos de entrada y salida. Una vez hecho esto es posible establecer una conexión con el PC de Beckhoff a través de un cable Ethernet y con el software TwinCAT. Acceder a los periféricos es posible a través de la creación de un proyecto de TwinCAT, permitiendo acceso a la sección de **I/O** (*Input/Output*) del PLC al que se esté conectado:

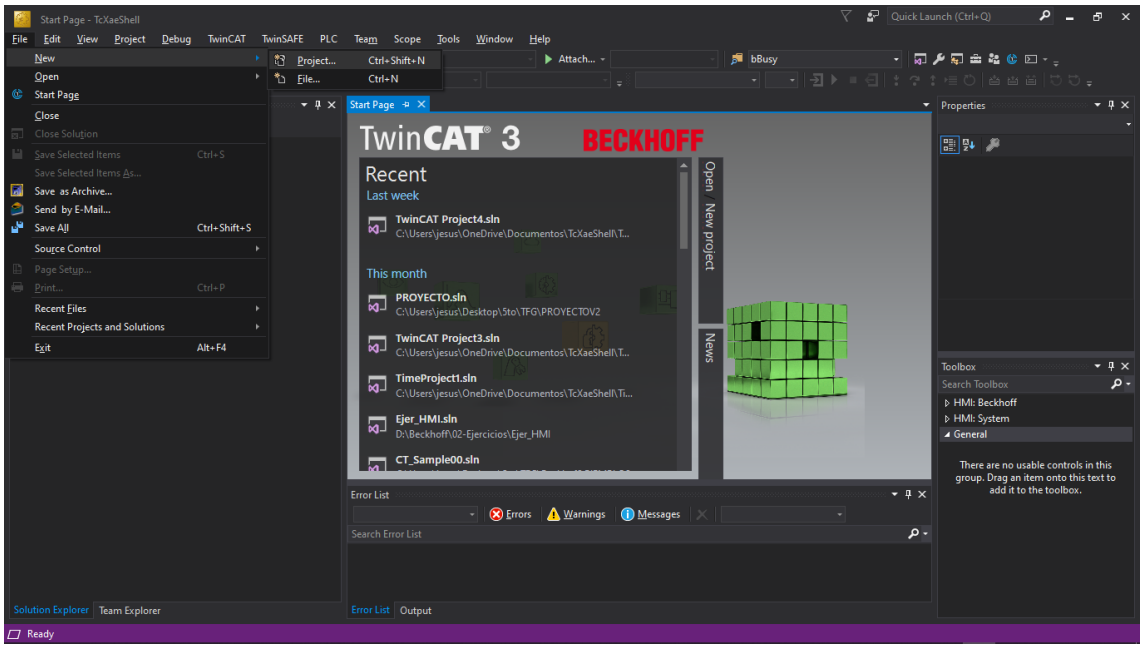


Figura 19.- Creación de proyecto.

En primer lugar, debemos crear un proyecto de TwinCAT. Conseguimos esto a través de la barra de herramientas localizada en la parte superior del entorno de desarrollo, accediendo a la opción de creación de nuevos proyectos a través de "*File/New/Project*".

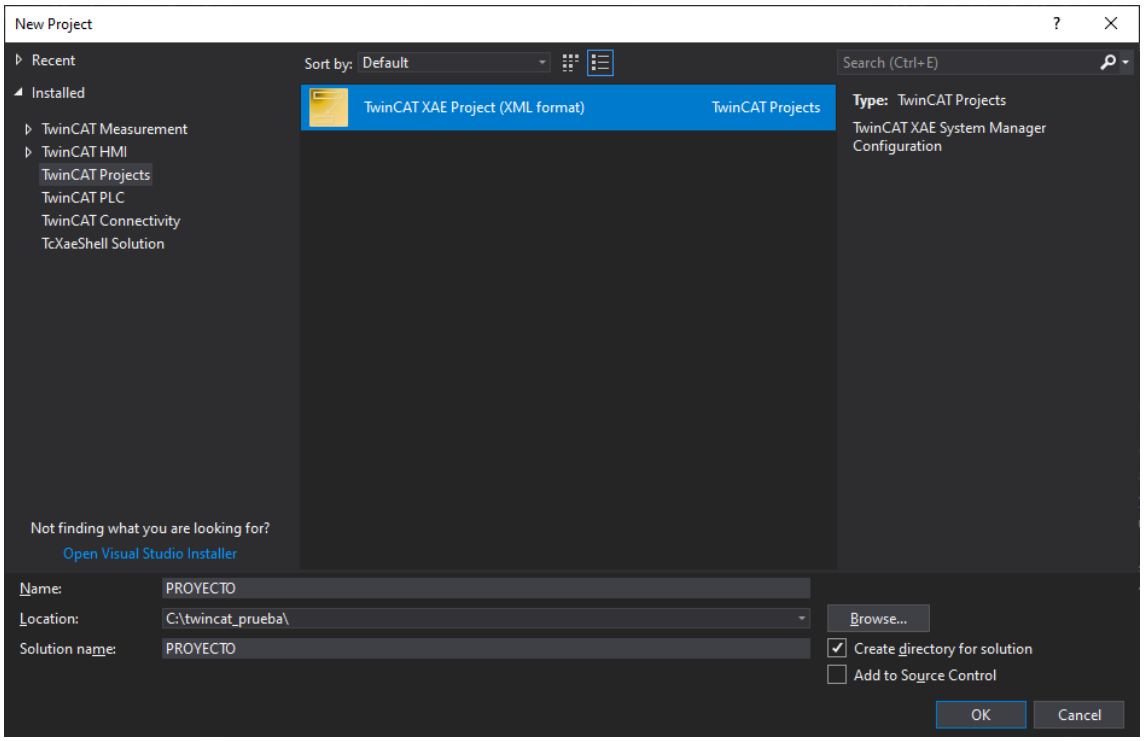


Figura 20.- Selección de solución.

Creamos un proyecto estándar, obteniendo una solución con estructura TwinCAT.

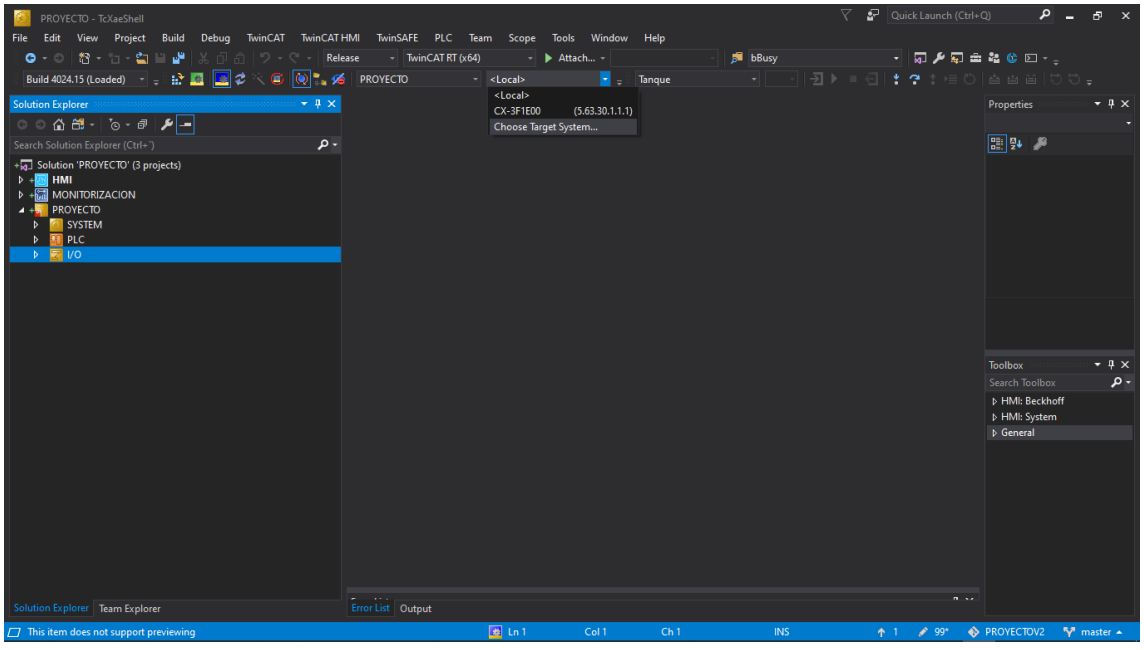


Figura 21.- Selección de conexión.

Para establecer una conexión con una CPU seleccionamos "*Choose Target System*" en el desplegable, tras lo que aparecerá una ventana que muestra las CPUs conectadas a nuestro sistema local. En esta ventana, usamos "*Search (Ethernet)…*" para buscar qué dispositivos están conectados al nuestro.

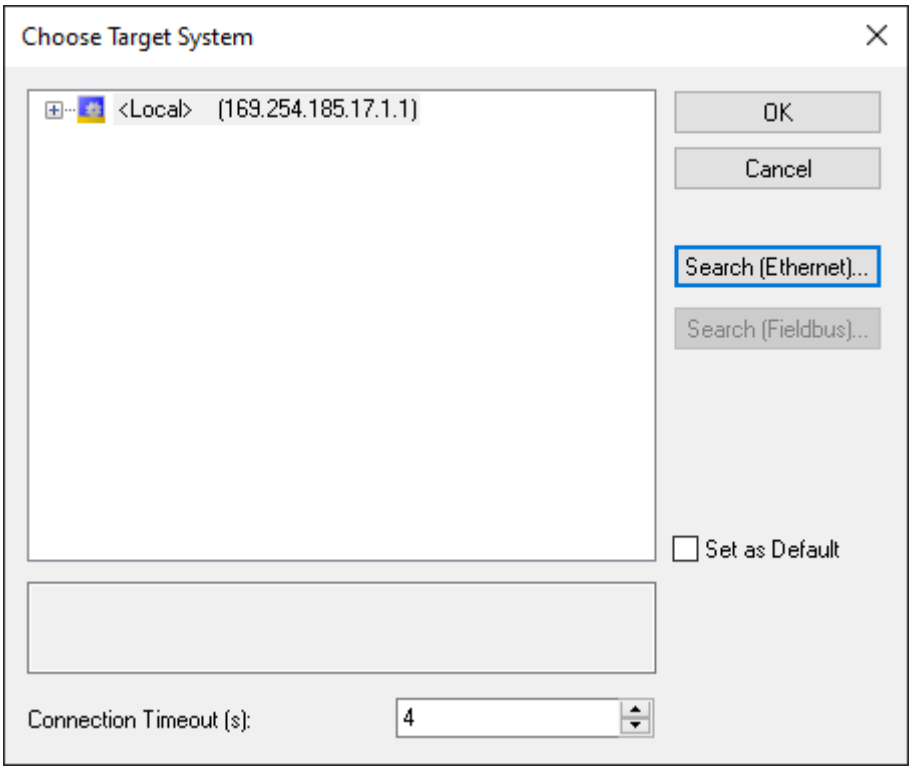


Figura 22.- Búsqueda por Ethernet.

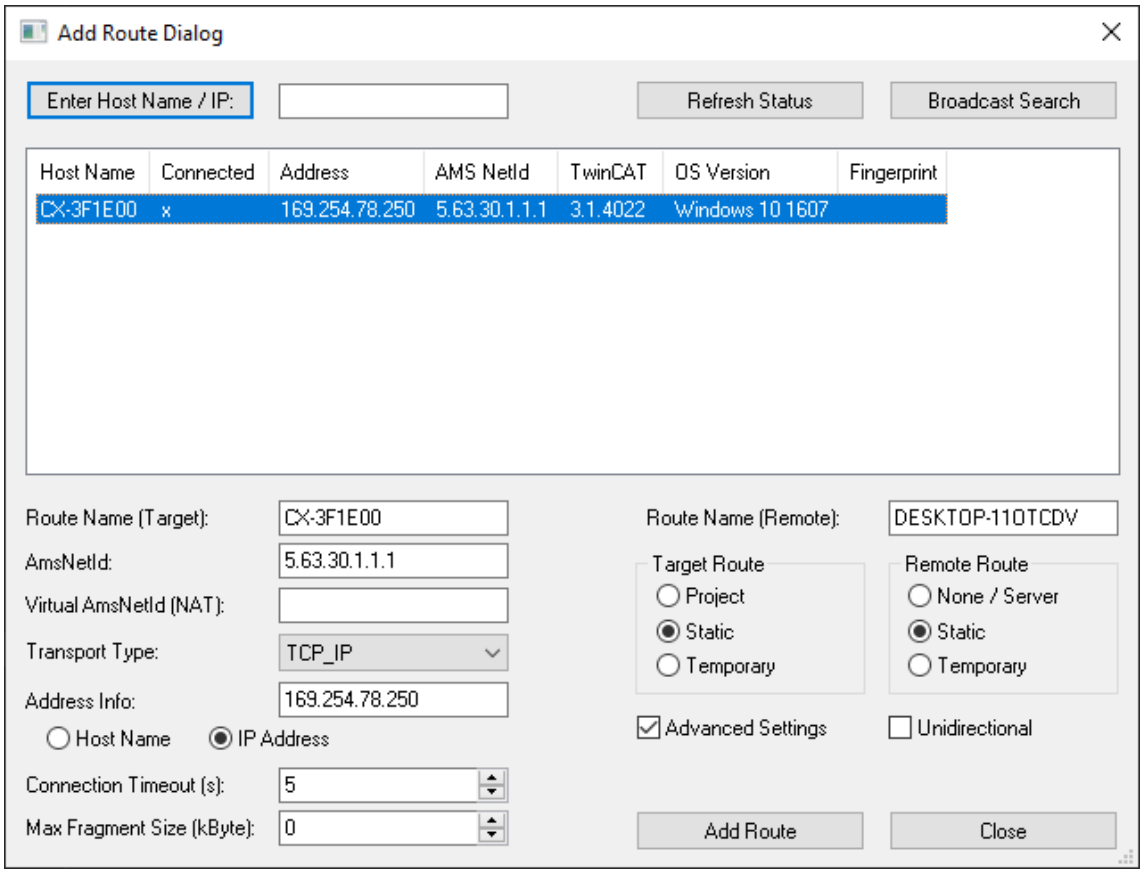


Figura 23.- Conexión a PC embebido.

Tras establecer la conexión a la CPU, podemos escanear los módulos de entrada y salida analógicos que vamos a emplear en el proyecto.

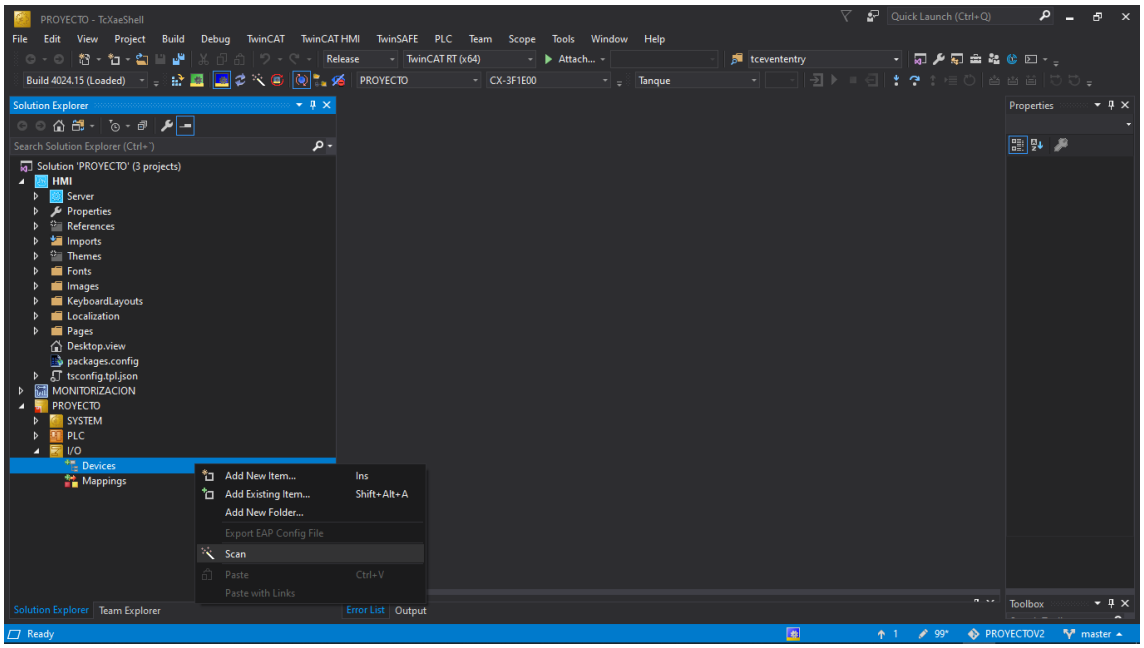


Figura 24.- Escaneo de Devices.

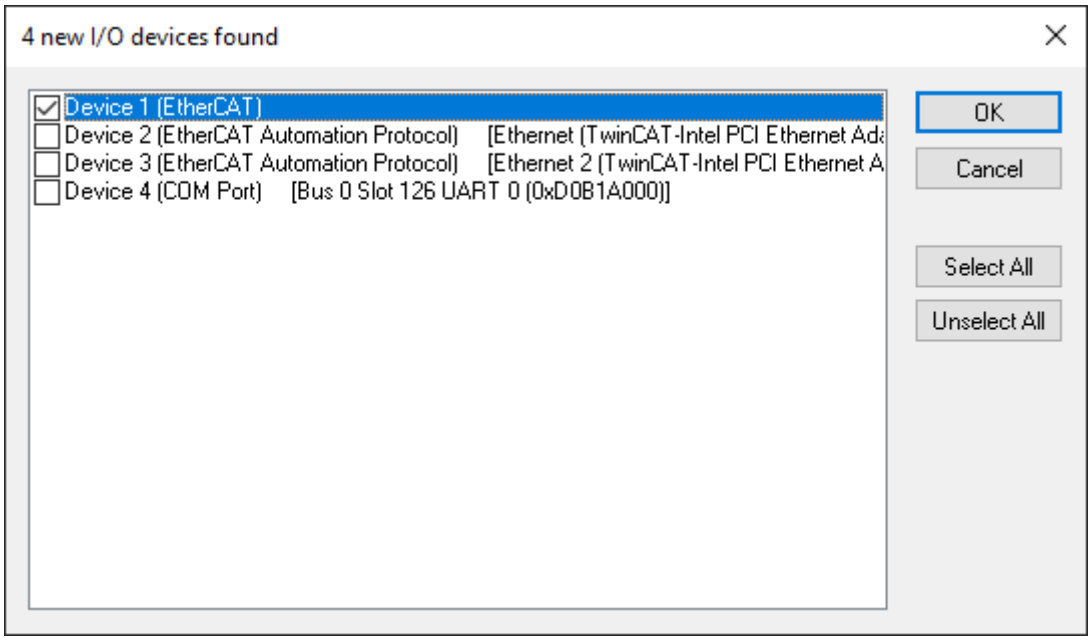


Figura 25.- Selección de Device 1.

Una vez escaneadas debemos asignar las variables que va a emplear cada canal tanto para el módulo de salida analógica como el de entrada.

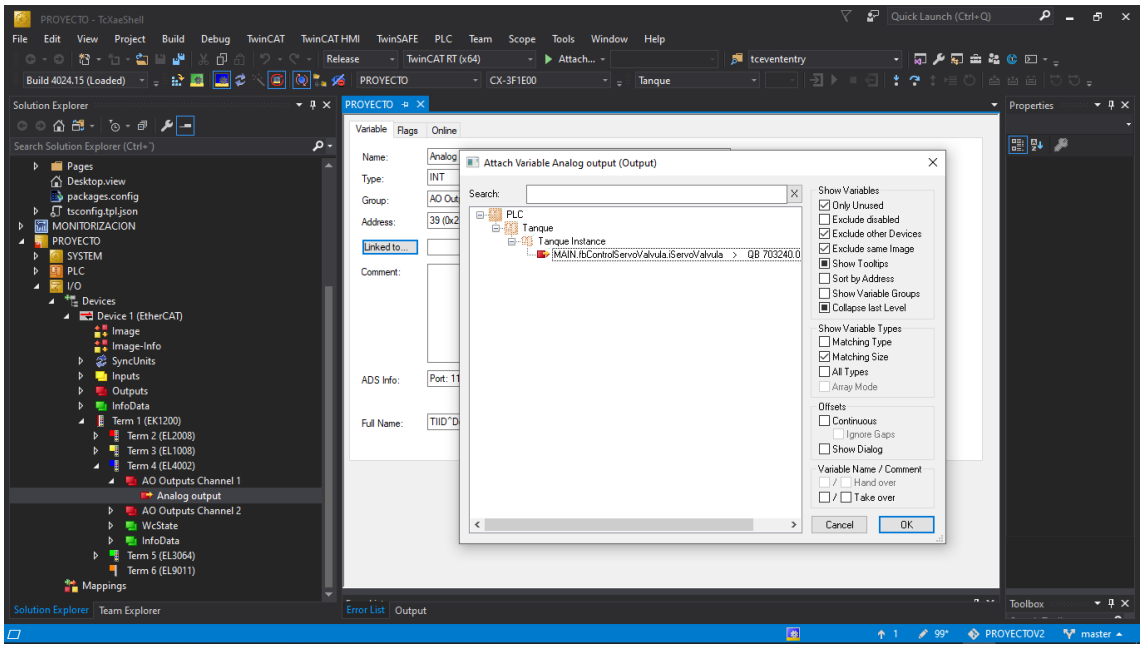


Figura 26.- Asignación de variable de salida.

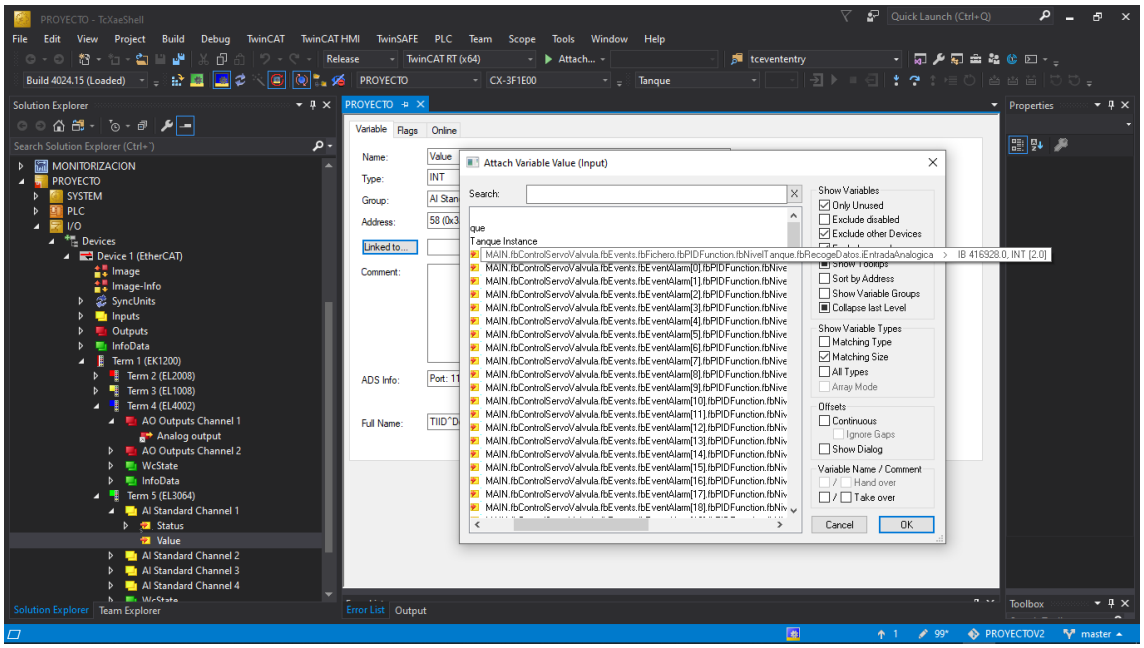


Figura 27.- Asignación de variable de entrada.

2.8.3 Programación del PLC

Tras comprobar que se detectan correctamente los módulos analógicos de entrada y salida, se puede proceder con la programación del PLC. Comenzamos con la creación de un proyecto PLC, que creará una solución con estructura TwinCAT:

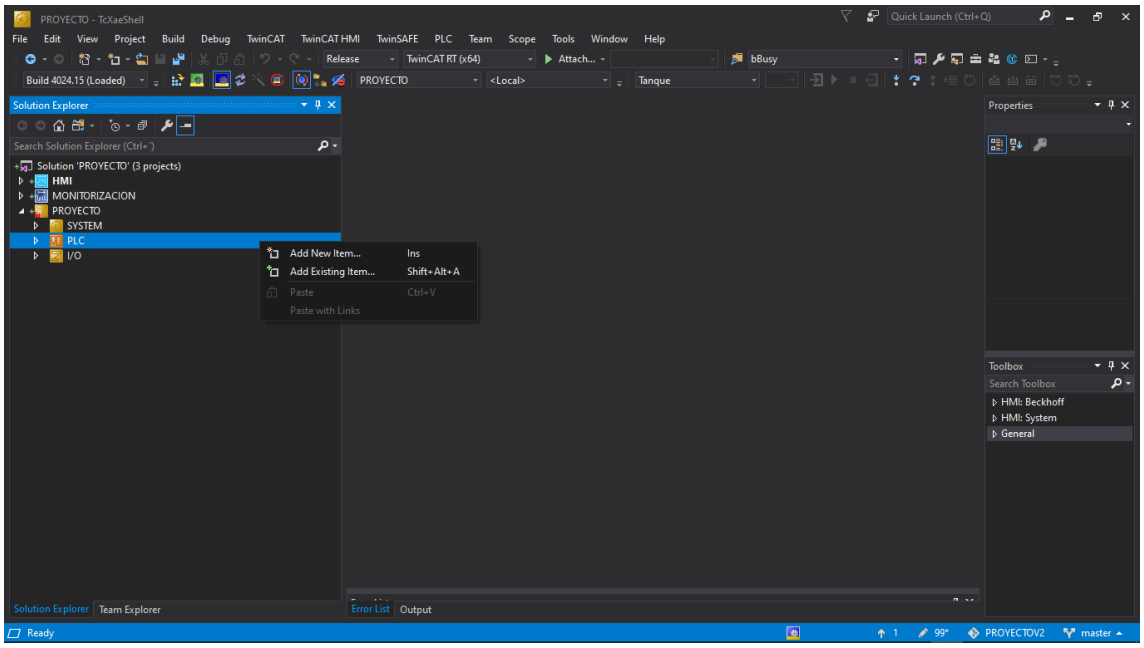


Figura 28.- Selección de proyecto PLC.

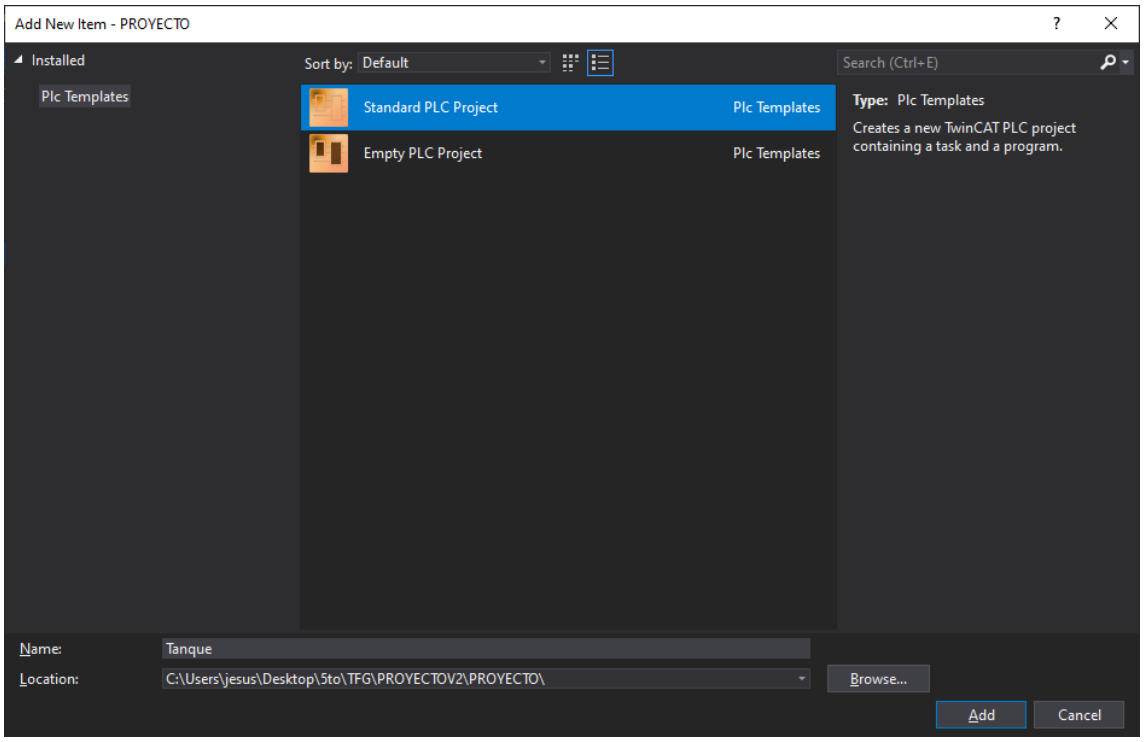


Figura 29.- Creación de proyecto PLC estándar.

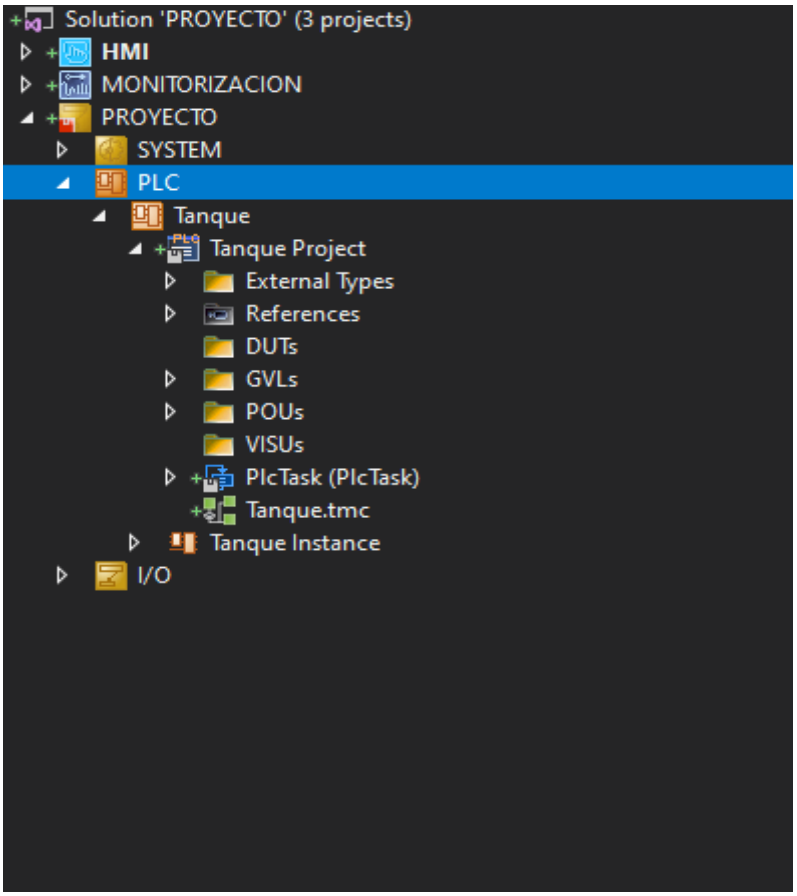


Figura 30.- Generación de solución PLC con estructura TwinCAT.

Como se puede apreciar en la **Figura 30**, la creación de un proyecto PLC genera automáticamente una serie de carpetas:

- **External Types:** Esta carpeta contiene todos los "*Type System*" usados por el PLC. Un "*Type System*" puede ser un tipo de datos o función concreta del sistema, ya sea editada por el usuario o generada por defecto.
- **References:** Aquí se guardan las librerías del estándar IEC 61131-3.
- **DUTs:** "*Data Unit Types*" es donde se definen los tipos de unidades de datos, por ejemplo, estructuras.
- **GVLs:** *"Global Variable Lists"* es la carpeta que almacena las variables globales declaradas por el usuario.
- **POUs:** "*Program Organization Units*" será utilizado para el almacenamiento del código, es decir, las funciones, bloques de funciones (**FBs**), programas…
- **VISUs:** *"Visualizations"* permite el desarrollo de HMIs de carácter básico.

Haciendo uso de estas carpetas es posible la creación de un programa que permite ejercer control sobre la planta de nivel con la que estamos trabajando.

El programa desarrollado hace uso mayoritariamente de bloques de funciones. Un bloque de funciones es un POU que devuelve uno o varios valores al ejecutarse. A diferencia de una función, es necesario que se llamen a través de una instancia, que corresponde a una copia del bloque de funciones en un POU diferente.

Esta metodología de programación es conocida como Programación Estructurada de Bloques, un paradigma de programación que da al desarrollador la opción de dividir su software en bloques, donde cada bloque realiza una función concreta y pueden ser usados de forma independiente o conjunta, anidando bloques dentro de bloques.

Una vez aclarado esto, la estructura seguida para el desarrollo de la solución PLC ha sido la siguiente (se incluyen los bloques de funciones empleados):

1. Lectura y conversión de voltios a centímetros de los datos obtenidos en el módulo de entrada analógica EL3064 (FB_RecogeDatos).
2. Implementación de asignador de consigna determinada por el usuario (FB_NivelTanque).
3. Implementación de controlador PID (FB_PIDFunction).
4. Implementación de fichero de recogida de datos relevantes de la planta y el PID (FB_Fichero) y Lectura de hora del sistema local para ser adjuntada dentro del fichero de datos (FB_HoraSistema).
5. Desarrollo de archivador de eventos (FB_Events).
6. Implementación de salida de voltaje a través de módulo de salida EL4002, haciendo uso de la salida obtenida por el PID (FB_ControlServoValvula).
7. Implementación de copiador de ficheros necesario para el guardado de ficheros en local (FB_FileCopy).
8. Paso de FBs a MAIN para la ejecución del programa.

Vista la estructura seguida, se adjuntan capturas del código siguiendo el orden mencionado:

1.

FB_RecogeDatos

1

```
FUNCTION_BLOCK FB_RecogeDatos   //Lee entrada analógica para convertir voltaje en centímetros
VAR_OUTPUT
    iNivelCentimetros : LREAL;  //Nivel del tanque (en centímetros)
END_VAR
VAR
    iEntradaAnalogica AT %I* : INT;
    rEntradaAnalogicaV : LREAL;
END_VAR
VAR CONSTANT
    rConversionVolt : LREAL := 0.000305178; //factor de conversion (es la relacion entre voltaje y dec)
END_VAR
```

```
rEntradaAnalogicaV := iEntradaAnalogica*rConversionVolt;
iNivelCentimetros := ((rEntradaAnalogicaV-2.25)/0.07);      //Ecuacion de la recta (v/cm)
                                                            //obtenida a partir de dos puntos.
IF iNivelCentimetros < 0 THEN
    iNivelCentimetros := 0;
END_IF
```

Figura 31.- Bloque de funciones FB_RecogeDatos.

La lectura de la entrada es guardada en la variable *iEntradaAnalogica*, que es convertida a un valor legible en voltios a través del factor de conversión *rConversionVolt* obtenido a través del siguiente cálculo:

Para un valor de 0 cm de llenado en el tanque se obtiene de lectura 2.251 V, que es interpretado por la entrada analógica como un valor de 7376, por lo tanto:

$$rConversionVolt = \frac{2.251}{7376} = 0.000305178$$

Realizando este cálculo para otros valores de llenado se obtiene un resultado similar o parecido.

Una vez convertida la entrada analógica a un valor de voltaje legible, podemos convertir dicho voltaje a centímetros para poder mostrarlos al usuario. Para hacer esto, se ha efectuado la ecuación de la recta entre dos puntos en la que se relacionan voltios y centímetros:

$$y = mx + b$$

$$(y - y_1) = m(x - x_1)$$

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5.75 - 2.25}{50 - 0} = 0.07$$

$$b = -mx_1 + y_1 = -0.07 * 0 + 2.25 = 2.25$$

$$y = 0.07x + 2.25 \ ; \ x = \frac{y - 2.25}{0.07}$$

Con esta ecuación se obtienen los valores en centímetros en función del voltaje recibido en la entrada analógica, guardados en la variable *iNivelCentimetros*.

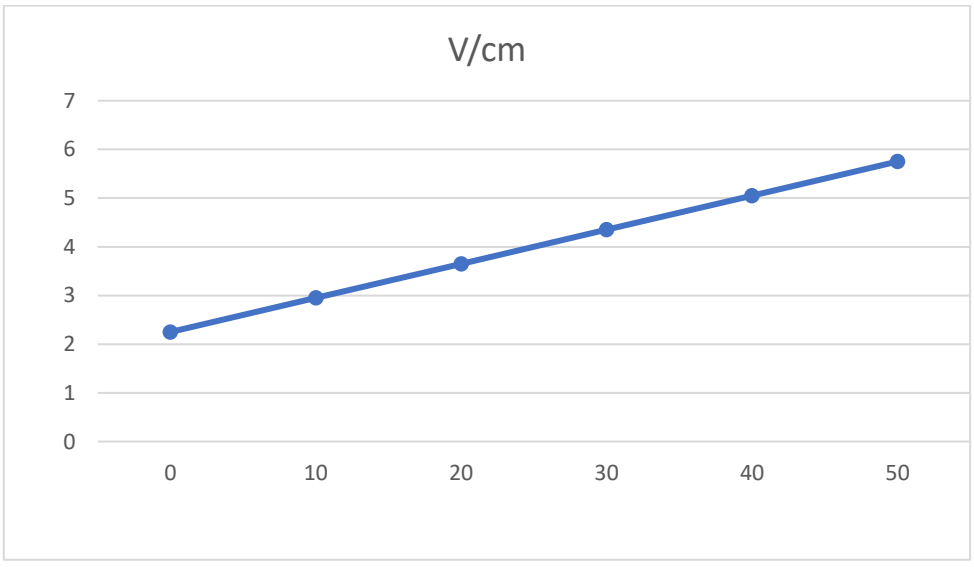


Figura 32.- Relación V/cm.

2.

FB_NivelTanque

```
FUNCTION_BLOCK FB_NivelTanque
VAR_INPUT
    fbRecogeDatos : FB_RecogeDatos;
END_VAR
VAR
    lConsigna : LREAL := 40;
    lCentimetros : LREAL;
END_VAR
```

```
fbRecogeDatos();
lCentimetros := fbRecogeDatos.iNivelCentimetros;
```

Figura 33.- Bloque de funciones FB_NivelTanque.

3.

FB_PIDFunction

```
FUNCTION_BLOCK FB_PIDFunction
VAR_INPUT
    fbNivelTanque : FB_NivelTanque;
END_VAR
VAR
    fSetpointValue              : LREAL;
    fActualValue                : LREAL;
    fManSyncValue               : LREAL;
    bSync                       : BOOL;
    DetectEdge                  : R_TRIG;

    fOut                        : LREAL;
    bHold                       : BOOL;

    eMode                       : E_CTRL_MODE;
    stCTRL_PID_PARAMS           : ST_CTRL_PID_PARAMS;
    //eState                        : E_CTRL_STATE;

    eErrorId                    : E_CTRL_ERRORCODES;
    bError                      : BOOL;
    bARWactive                  : BOOL;

    (* controller *)
    fbCTRL_PID                  : FB_CTRL_PID;


    Proporcional                : LREAL;
    Integral                    : LREAL;
    Derivativa                  : LREAL;    //Los declaro como LREAL porque TIME da problemas con la HMI.
    Modo                        : INT;


    fbGetCurTaskIdx   : GETCURTASKINDEX;
    tCycleTime        : TIME; //tiempo de ciclo del plc
END_VAR
```

Figura 34.- Ventana de declaración FB_PIDFunction.

FB_PIDFunction 3

```
fbNivelTanque();
fbGetCurTaskIdx();
tCycleTime := UDINT_TO_TIME (_TaskInfo[fbGetCurTaskIdx.index].CycleTime/10000); //conversion a ms


(* init parameter struct *)
stCTRL_PID_PARAMS.tCtrlCycleTime        := tCycletime;
stCTRL_PID_PARAMS.tTaskCycleTime        := tCycletime;
stCTRL_PID_PARAMS.fKp                   := Proporcional;            (* proportional gain Kp
stCTRL_PID_PARAMS.tTn                   := LREAL_TO_TIME(Integral);      (*  Tn
stCTRL_PID_PARAMS.tTv                   := LREAL_TO_TIME(Derivativa);       (*  Tv
stCTRL_PID_PARAMS.tTd                   := T#100MS;      (* Damping Td                                  *)
stCTRL_PID_PARAMS.fOutMaxLimit          :=  10;          (* maximum output limit *)
stCTRL_PID_PARAMS.fOutMinLimit          :=   0;          (* minimum output limit *)


(* set the mode to 0: IDLE, 1: PASSIVE, 2: ACTIVE, 3: RESET, 4: MANUAL*)
eMode := Modo;

IF NOT bSync THEN
(* call controller *)
fbCTRL_PID(     fSetpointValue      := fbNivelTanque.lConsigna,
                fActualValue        := fbNivelTanque.lCentimetros,
                fManSyncValue       := fManSyncValue,
                bSync               := bSync,
                eMode               := eMode,
                bHold               := bHold,
                stParams            := stCTRL_PID_PARAMS,
                //eState                -> eState,
                fOut                -> fOut,
                bARWactive          -> bARWactive,
                eErrorId            -> eErrorId,
                bError              -> bError
                );
END_IF

DetectEdge( CLK:=bSync );
IF DetectEdge.Q THEN
    fOut := fManSyncValue;
END_IF

(* copy var to scope var *)
ScopeSet.fSetpointValueToScope  := fbCTRL_PID.fSetpointValue;
ScopeSet.fActualValueToScope    := fbCTRL_PID.fActualValue;

(* copy var to scope var *)
ScopeSet.fOutToScope := fOut;
```

Figura 35.- Ventana de edición de código FB_PIDFunction.

La implementación de FB_PIDFunction está basada en el bloque de funciones administrado por Beckhoff **FB_CTRL_PID**, localizado dentro del paquete **TF4100|Controller Toolbox**, que debe ser instalado para su uso. Información sobre este bloque de funciones puede accederse a través de [59].

4.

FB_Fichero

```
FUNCTION_BLOCK FB_Fichero
VAR_INPUT
    fbPIDFunction   :   FB_PIDFunction;
    fbHoraSistema   :   FB_HoraSistema;
END_VAR
VAR
    eMode                       : E_CTRL_MODE;

    fbCTRL_LOG_DATA             : FB_CTRL_LOG_DATA;
    stCTRL_LOG_DATA_PARAMS      : ST_CTRL_LOG_DATA_PARAMS;

    LoggerData                  : LOGGER_DATA;

    eErrorId                    : E_CTRL_ERRORCODES;
    bError                      : BOOL;

    Temporizador                : TON;
    bInit                       : BOOL := TRUE;

END_VAR
```

4

Figura 36.- Ventana de declaración FB_Fichero.

FB_Fichero

```
F_GetSystemTime();
fbPIDFunction();
fbHoraSistema();

IF bInit THEN
    (* Inicialización. *)
    stCTRL_LOG_DATA_PARAMS.tLogCycleTime            := T#1S;
    stCTRL_LOG_DATA_PARAMS.tTaskCycleTime           := T#10MS;
    stCTRL_LOG_DATA_PARAMS.nNumberOfColumns         := 9;
    stCTRL_LOG_DATA_PARAMS.sFileName                := 'C:\test_log.csv';
    stCTRL_LOG_DATA_PARAMS.bWriteTimeStamps         := FALSE;

    (* Encabezado (STRING). *)
    stCTRL_LOG_DATA_PARAMS.arColumnHeadings[1]  := 'Hora';
    stCTRL_LOG_DATA_PARAMS.arColumnHeadings[2]  := 'Nivel';
    stCTRL_LOG_DATA_PARAMS.arColumnHeadings[3]  := 'Consigna';
    stCTRL_LOG_DATA_PARAMS.arColumnHeadings[4]  := 'fOut';
    stCTRL_LOG_DATA_PARAMS.arColumnHeadings[5]  := 'Kp';
    stCTRL_LOG_DATA_PARAMS.arColumnHeadings[6]  := 'Tn';
    stCTRL_LOG_DATA_PARAMS.arColumnHeadings[7]  := 'Tv';
    stCTRL_LOG_DATA_PARAMS.arColumnHeadings[8]  := 'Modo';
    stCTRL_LOG_DATA_PARAMS.arColumnHeadings[9]  := 'Error';

    bInit := FALSE;
END_IF
```

4

Figura 37.- Ventana de código FB_Fichero.

FB_Fichero

4

```
(* Datos a escribir en el fichero (LREAL). *)
LoggerData[ 1 ] := TO_LREAL (fbHoraSistema.sLogTime);
LoggerData[ 2 ] := fbPIDFunction.fbNivelTanque.lCentimetros;
LoggerData[ 3 ] := fbPIDFunction.fbNivelTanque.lConsigna;
LoggerData[ 4 ] := fbPIDFunction.fOut;
LoggerData[ 5 ] := fbPIDFunction.stCTRL_PID_PARAMS.fKp;
LoggerData[ 6 ] := TIME_TO_LREAL (fbPIDFunction.stCTRL_PID_PARAMS.tTn);
LoggerData[ 7 ] := TIME_TO_LREAL (fbPIDFunction.stCTRL_PID_PARAMS.tTv);
LoggerData[ 8 ] := fbPIDFunction.eMode;
LoggerData[ 9 ] := fbPIDFunction.eErrorId;


eMode := eCTRL_MODE_ACTIVE;                    //modo activo para abrir el fichero.

IF fbPIDFunction.eMode <> eCTRL_MODE_RESET THEN
    Temporizador(IN := TRUE,
                 PT := T#1S);                  //Cierre forzado del fichero cada cierto tiempo
END_IF                                         //para asegurar que se escribe en él.

IF fbPIDFunction.eMode = eCTRL_MODE_RESET OR Temporizador.Q THEN
    (* Modo: PASSIVE --> Para de escribir en el fichero y lo cierra. *)
    eMode := eCTRL_MODE_PASSIVE;
    Temporizador(IN := FALSE);
    stCTRL_LOG_DATA_PARAMS.bWriteColumnHeadings := FALSE;       //Impide la escritura de nuevos encabezados.
END_IF

(* Llamada al bloque de funciones. *)
fbCTRL_LOG_DATA(    fLogData    := LoggerData,
                    eMode       := eMode,
                    stParams    := stCTRL_LOG_DATA_PARAMS,
                    eErrorId    => eErrorId,
                    bError      => bError);
```

Figura 38.- Continuación ventana de código FB_Fichero.

FB_HoraSistema

4

```
FUNCTION_BLOCK FB_HoraSistema
VAR
    bInit           : BOOL;
    nTime           : UINT;
    tBufferTime     : TIME;
    dtBufferDT      : DT;
    nCalcBuffer     : UDINT;
    sMs             : STRING;
    sLogTime        : STRING;
    sLogTimeWithMs  : STRING;
    stSystemTime    : TIMESTRUCT;
    fbLocalTime     : FB_LocalSystemTime;
END_VAR
```

Figura 39.- Ventana de declaración FB_HoraSistema.

FB_HoraSistema

4

```
fbLocalTime(bEnable := TRUE);
IF NOT bInit
THEN
    dtBufferDT := SYSTEMTIME_TO_DT(fbLocalTime.systemTime);
    IF fbLocalTime.bValid
    THEN
        bInit := TRUE;
    END_IF
ELSE
    nTime := nTime + 1;
    tBufferTime := UINT_TO_TIME(nTime*10);
    IF tBufferTime = T#1S
    THEN
        //Add a second to your system time
        ntime := 0;
        nCalcBuffer := DT_TO_UDINT(dtBufferDT)+1;
        dtBufferDT := UDINT_TO_DT(nCalcBuffer);
        sLogTime := DT_TO_STRING(dtBufferDT);
        sLogTime := DELETE(sLogTime,14,1);
        sLogTime := REPLACE(sLogTime,'.',1,3);
        sLogTime := REPLACE(sLogTime,'.',1,6);
        sLogTimeWithMs := sLogTime;
    ELSE
        //Add ms string time-stamp
        sMs := TIME_TO_STRING(tBufferTime);
        sLogTimeWithMs := CONCAT(sLogTime,sMs);
    END_IF
END_IF
```

Figura 40.- Ventana de código FB_HoraSistema.

FB_Fichero se ha desarrollado tomando como base el bloque de funciones **FB_CTRL_LOG_DATA**, perteneciente a **TF4100|Controller Toolbox**, cuya información es accesible a través de [60]. FB_HoraSistema se emplea para mostrar la hora de escritura de cada línea dentro del fichero.

5.

FB_Events

5

```
FUNCTION_BLOCK FB_Events
VAR
    fbEventLogger       : FB_TcEventLogger;
    fbFichero           : FB_Fichero;

    bClearAlarms        : BOOL;
    bConfirmAlarms      : BOOL;
    bClearLoggedEvents  : BOOL;
    bExportLoggedEvents : BOOL;

    fbRtExport          : R_TRIG;
    fbCsvExportSetting  : FB_TcEventCsvExportSettings;
    fbExportFilter      : FB_TcEventFilter;
    sExportFilesName    : STRING := 'C:\Temp\Eventlogger-export.csv';

    fbEventAlarm        : ARRAY[0..148] OF FB_EventAlarm;
    aTestAlarm          : ARRAY[0..148] OF BOOL;
    fbEventInfo         : ARRAY[0..15] OF FB_EventMessage;
    aTestInfo           : ARRAY[0..15] OF BOOL;


    hr                  : HRESULT;
    hrLastError         : HRESULT;
END_VAR
```

Figura 41.- Ventana de declaración FB_Events.

FB_Events

```
fbFichero();

    a_010_Alarms();

    a_020_Infos();

    a_030_ConfirmLoggedAlarms();

    //a_040_ClearLoggedAlarms();

    a_050_ClearLoggedEvents();

    a_060_ExportLoggedEvents();
```

5

Figura 42.- Ventana de código FB_Events.

Debido al número de líneas de código, la acción *a_010_Alarms();* es visible a través de https://pastebin.com/Y5wzgR0q

FB_Events.a_020_Infos

```
IF fbFichero.fbPIDFunction.eErrorId = 0 THEN
    aTestInfo[fbFichero.fbPIDFunction.eErrorId] := TRUE;
END_IF
fbEventInfo [0](
    bEvent      := aTestInfo[fbFichero.fbPIDFunction.eErrorId],
    nEventID    := 0,
    );
```

5

Figura 43.- Ventana de código de acción a_020_Infos.

FB_Events.a_030_ConfirmLoggedAlarms

```
IF bConfirmAlarms THEN
    hr := fbEventLogger.ConfirmAllAlarms(0);
    IF FAILED (hr) THEN
        hrLastError := hr;
    END_IF
END_IF
```

5

Figura 44.- Ventana de código de acción a_030_ConfirmLoggedAlarms.

FB_Events.a_050_ClearLoggedEvents

5

```
//Eliminar todos los eventos recogidos del cache del PLC
IF bClearLoggedEvents THEN
    bClearLoggedEvents := FALSE;
    fbEventLogger.ClearLoggedEvents(0);
END_IF
```

Figura 45.- Ventana de código de acción a_050_ClearLoggedEvents.

FB_Events.a_060_ExportLoggedEvents

5

```
//Crear nombre de archivo
sExportFilesName := CONCAT (CONCAT('C:\Temp\',PRG_P_TIMER.sDateTime), '.csv');

//Exportar como csv los eventos. Sin un filtro se exportan todos.

fbRtExport (CLK := bExportLoggedEvents);
IF fbRtExport.Q THEN
    fbCsvExportSetting.Clear();
    fbCsvExportSetting.AddLanguage(1033, 'English');
    //fbCsvExportSetting.AddFilter(fbExportFilter);
    //fbExportFilter.Clear().IsAlarm().AND_OP().EventClass.EqualTo(TC_EVENTS.myAlarmEventClass.eCTRL_ERROR_FileDelete.uuidEventClass);
END_IF
IF fbEventLogger.ExportLoggedEvents(sExportFilesName, fbCsvExportSetting, hrErrorCode => hr) THEN
    bExportLoggedEvents := FALSE;
END_IF
```

Figura 46.- Ventana de código de acción a_060_ExportLoggedEvents.

FB_EventAlarm

5

```
FUNCTION_BLOCK FB_EventAlarm
VAR_INPUT
    fbPIDFunction   : FB_PIDFunction;
    bEvent          : BOOL;
    nEventID        : UDINT;
    sSource         : STRING;
END_VAR
VAR
    bIsInitialized  : BOOL;

    fbSourceInfo    : FB_TcSourceInfo;
    fbAlarm         : FB_TcAlarm;
    eventEntry      : TcEventEntry;
    fbRtEvent       : R_TRIG;
    fbFtEvent       : F_TRIG;

    hr              : HRESULT;
    hrLastError     : HRESULT;
END_VAR
```

Figura 47.- Ventana de declaración FB_EventAlarm.

FB_EventAlarm

5

```
//Inicialización
IF NOT bIsInitialized THEN
    bIsInitialized := TRUE;

    eventEntry := TC_EVENTS.myAlarmEventClass.eCTRL_ERROR_INVALIDCTRLCYCLETIME;
    eventEntry.nEventId := nEventID;

    fbSourceInfo.Clear();
    fbSourceInfo.sName := sSource;

    hr := fbAlarm.CreateEx(eventEntry, TRUE, ipSourceInfo := fbSourceInfo);
    IF FAILED(hr) THEN
        hrLastError := hr;
    END_IF
END_IF

//Disparador de evento
fbRtEvent(CLK := bEvent);
fbFtEvent(CLK := bEvent);

//Saltar alarma
IF fbRtEvent.Q THEN
    hr := fbAlarm.Raise(0);
    IF FAILED(hr) THEN
        hrLastError := hr;
    END_IF
END_IF

//Alarma apagada
IF fbFtEvent.Q THEN
    hr := fbalarm.Clear(0, bResetConfirmation := TRUE);
    IF FAILED (hr) THEN
        hrLastError := hr;
    END_IF
END_IF
```

90 %

Figura 48.- Ventana de código FB_EventAlarm.

FB_EventMessage

5

```
FUNCTION_BLOCK FB_EventMessage
VAR_INPUT
    fbPIDFunction   : FB_PIDFunction;
    bEvent          : BOOL;
    nEventID        : UDINT;
    sSource         : STRING;
END_VAR
VAR
    bIsInitialized  : BOOL;

    fbSourceInfo    : FB_TcSourceInfo;
    fbMessage       : FB_TcMessage;
    eventEntry      : TcEventEntry;
    fbRtEvent       : R_TRIG;

    hr              : HRESULT;
    hrLastError     : HRESULT;
END_VAR
```

Figura 49.- Ventana de declaración FB_EventMessage.

FB_EventMessage

```
//Inicialización
IF NOT bIsInitialized THEN
    bIsInitialized := TRUE;

    IF nEventID < 1 THEN
        eventEntry := TC_EVENTS.myAlarmEventClass.eCTRL_ERROR_NOERROR;
    END_IF

    eventEntry.nEventId := nEventID;

    fbSourceInfo.Clear();
    fbSourceInfo.sName := sSource;

    hr := fbMessage.CreateEx(eventEntry, ipSourceInfo := fbSourceInfo);
    IF FAILED(hr) THEN
        hrLastError := hr;
    END_IF
END_IF

//Disparador de evento
fbRtEvent (CLK := bEvent);

//Mensaje enviado
IF fbRtEvent.Q THEN
    hr := fbMessage.Send(0);
    IF FAILED(hr) THEN
        hrLastError := hr;
    END_IF
END_IF
```

Figura 50.- Ventana de código FB_EventMessage.

HRESULT

```
{attribute 'TcTypeSystem'}
{attribute 'signature_flag' := '33554432'}
{attribute 'checksuperglobal'}
{attribute 'show'}
{attribute 'no-analysis'}

    {attribute 'GUID' := '18071995-0000-0000-0000-000000000019'}
TYPE HRESULT : DINT;
END_TYPE
```

Figura 51.- External Type HRESULT.

TcEventEntry

```
{attribute 'TcTypeSystem'}
{attribute 'signature_flag' := '33554432'}
{attribute 'checksuperglobal'}
{attribute 'show'}
{attribute 'no-analysis'}

    {attribute 'GUID' := 'F00C83AD-DEC8-486E-AE99-5E0A75C26DE0'}
TYPE TcEventEntry :
    STRUCT
        {attribute 'GUID' := '18071995-0000-0000-0000-000000000021'}
        uuidEventClass : GUID;
        nEventId : UDINT;
        {attribute 'GUID' := 'B57D3F4A-0836-49B0-81C3-BED5F4817EC9'}
        eSeverity : TcEventSeverity;
        {attribute 'hide'}
        _reserved1 : UINT;
    END_STRUCT
END_TYPE
```

Figura 52.- External Type TcEventEntry.

PRG_P_TIMER

5

```
1   PROGRAM PRG_P_TIMER
2   VAR
3       fbSystemTime        : FB_LocalSystemTime;
4       dtCurrentDateTime   : DT;
5       sCurrentDateTime    : STRING;
6       sDateTime           : STRING;
7       bSecond             : BOOL;
8       nSecond             : WORD;
9       bPulse1s            : BOOL;
10
11      fbRtSecond          : R_TRIG;
12      fbLow_1s            : TON;
13      fbHigh_1s           : TOF;
14  END_VAR
15
```

Figura 53.- Ventana de declaración PRG_P_TIMER.

PRG_P_TIMER

5

```
1   fbSystemTime(
2       bEnable         := TRUE,
3       dwCycle         := 5,
4       dwOpt           := 1
5       );
6
7   dtCurrentDateTime   := SYSTEMTIME_TO_DT (fbSystemTime.systemTime);
8   sCurrentDateTime    := MID(DT_TO_STRING(dtCurrentDateTime), 19, 4);
9
10
11  sDateTime           := sCurrentDateTime;
12  sDateTime           := REPLACE(sDateTime,'-', 1, 11);
13  sDateTime           := REPLACE(sDateTime,'-', 1, 14);
14  sDateTime           := REPLACE(sDateTime,'-', 1, 17);
15
16
17
18  fbRtSecond(CLK      := fbSystemTime.systemTime.wSecond <> nSecond, Q => bSecond);
19  nSecond             := fbSystemTime.systemTime.wSecond;
20
21
22  fbLow_1s(IN         := NOT bPulse1s, PT := T#500MS);
23  fbHigh_1s(IN        := fbLow_1s.Q, PT := T#500MS, Q => bPulse1s);
24
```

Figura 54.- Ventana de código PRG_P_TIMER.

La implementación de FB_Events, FB_EventMessage y FB_EventAlarm ha sido posible gracias a la librería **Tc3_EventLogger**, que incluye todos los bloques de funciones necesarios para el desarrollo del archivador de eventos en TwinCAT 3, así como su aplicación dentro de la HMI. El manual de esta librería puede ser accedido a través de [61].

El programa **PRG_P_TIMER** se usa para nombrar los ficheros de eventos con la fecha y hora del sistema local.

El sistema de alarmado y mensajes hace uso de las constantes **TC_EVENT_CLASSES**, un sistema global de TwinCAT de listas de variables, que permite la identificación de clases de eventos y sus respectivos IDs. Es aquí donde se pueden crear mensajes y alarmas personalizadas por el usuario y donde se encuentran las usadas por TwinCAT por defecto.

6.

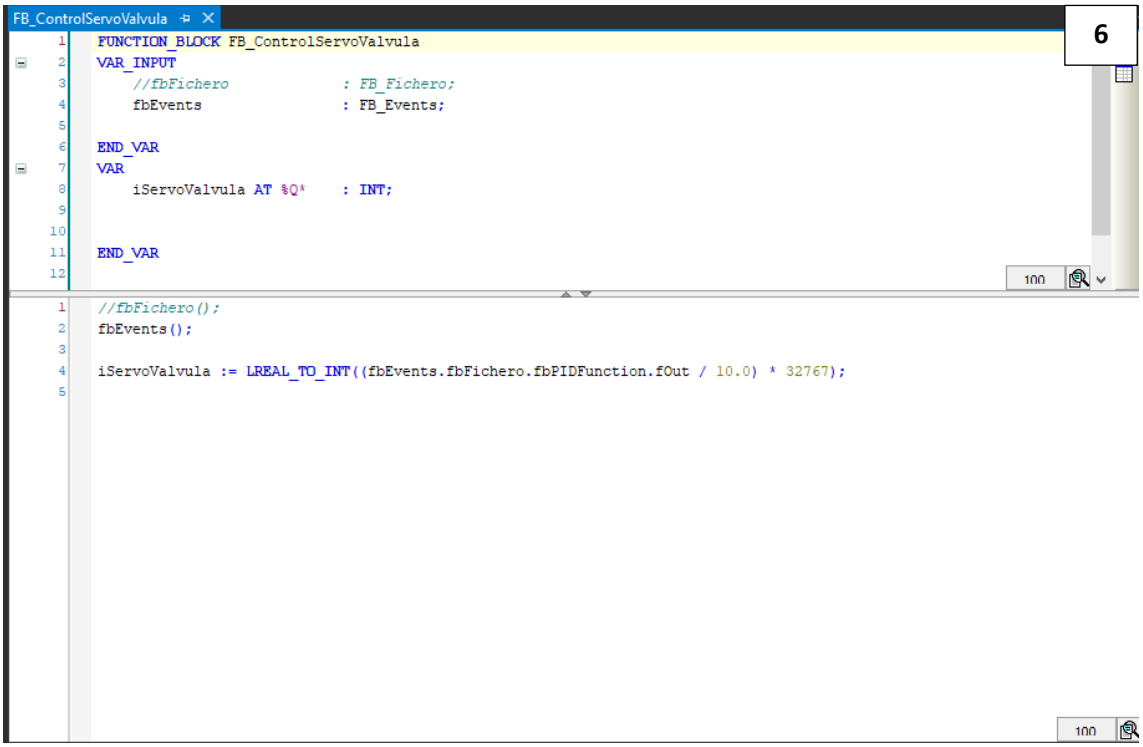


Figura 55.- Bloque de funciones FB_ControlServoValvula.

En la variable *iServoValvula* es guardado el valor en voltios que debe tomar la salida analógica EL4002 al convertir dicho valor de voltaje en un número reconocible por el PLC de Beckhoff ($0 = 0V$; $32767 = 10V$).

7.

FB_FileCopy

```
FUNCTION_BLOCK FB_FileCopy
VAR_INPUT
    sSrcNetId           : T_AmsNetId;
    sSrcPathName        : T_MaxString;
    sDestNetId          : T_AmsNetId;
    sDestPathName       : T_MaxString;
    bExecute            : BOOL;
    tTimeOut            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
VAR_OUTPUT
    bBusy               : BOOL;
    bError              : BOOL;
    nErrId              : UDINT;
END_VAR
VAR
    fbFileOpen          : FB_FileOpen;
    fbFileClose         : FB_FileClose;
    fbFileRead          : FB_FileRead;
    fbFileWrite         : FB_FileWrite;
    hSrcFile            : UINT := 0;(* File handle of the source file *)
    hDestFile           : UINT := 0;(* File handle of the destination file *)

    Step                : DWORD;
    RisingEdge          : R_TRIG;
    buffRead            : ARRAY[1..1000] OF BYTE;(* Buffer *)
    cbReadLength        : UDINT := 0;
END_VAR
```

Figura 56.- Ventana de declaración FB_FileCopy.

FB_FileCopy

```
RisingEdge(CLK:=bExecute);

CASE Step OF
     0:     (* Idle state *)
          IF RisingEdge.Q THEN
               bBusy := TRUE;
               bError:= FALSE;
               nErrId:=0;
               Step := 1;
               cbReadLength:=0;
               hSrcFile:=0;
               hDestFile:=0;
          END_IF

     1:     (* Open source file *)
          fbFileOpen( bExecute := FALSE );
          fbFileOpen( sNetId := sSrcNetId, sPathName := sSrcPathName,
                      nMode := FOPEN_MODEREAD OR FOPEN_MODEBINARY,
                      ePath := PATH_GENERIC, tTimeout := tTimeOut, bExecute := TRUE );
          Step := Step + 1;
     2:
          fbFileOpen( bExecute := FALSE );
          IF NOT fbFileOpen.bBusy THEN
               IF fbFileOpen.bError THEN
                    nErrId := fbFileOpen.nErrId;
                    bError := TRUE;
                    Step := 50;
               ELSE
                    hSrcFile := fbFileOpen.hFile;
                    Step := Step + 1;
               END_IF
          END_IF
```

Figura 57.- Ventana de código FB_FileCopy.

FB_FileCopy

7

```
3:      (* Open destination file *)
        fbFileOpen( bExecute := FALSE );
        fbFileOpen( sNetId := sDestNetId, sPathName := sDestPathName,
                    nMode := FOPEN_MODEWRITE OR FOPEN_MODEBINARY,
                    ePath := PATH_GENERIC, tTimeout := tTimeOut, bExecute := TRUE );
        Step := Step+1;
4:
        fbFileOpen( bExecute := FALSE );
        IF NOT fbFileOpen.bBusy THEN
            IF fbFileOpen.bError THEN
                nErrId := fbFileOpen.nErrId;
                bError := TRUE;
                Step := 50;
            ELSE
                hDestFile := fbFileOpen.hFile;
                Step := Step + 1;
            END_IF
        END_IF

5:      (* Read data from source file *)
        cbReadLength := 0;
        fbFileRead( bExecute:= FALSE );
        fbFileRead( sNetId:=sSrcNetId, hFile:=hSrcFile,
                    pReadBuff:= ADR(buffRead), cbReadLen:= SIZEOF(buffRead),
                    bExecute:=TRUE, tTimeout:=tTimeOut );
        Step := Step + 1;
6:
        fbFileRead( bExecute:= FALSE );
        IF NOT fbFileRead.bBusy THEN
            IF fbFileRead.bError THEN
                nErrId := fbFileRead.nErrId;
                bError := TRUE;
                Step := 50;
```

100

Figura 58.- Continuación ventana de código FB_FileCopy.

FB_FileCopy

7

```
            ELSE
                cbReadLength := fbFileRead.cbRead;
                Step := Step + 1;
            END_IF
        END_IF

7:      (* Write data to destination file *)
        fbFileWrite( bExecute := FALSE );
        fbFileWrite( sNetId:=sDestNetId, hFile:=hDestFile,
                     pWriteBuff:= ADR(buffRead), cbWriteLen:= cbReadLength,
                     bExecute:=TRUE, tTimeout:=tTimeOut );
        Step := Step + 1;
8:
        fbFileWrite( bExecute := FALSE );
        IF NOT fbFileWrite.bBusy THEN
            IF fbFileWrite.bError THEN
                nErrId := fbFileWrite.nErrId;
                bError := TRUE;
                Step := 50;
            ELSE
                IF fbFileRead.bEOF THEN (* Check if the EOF flag ist set *)
                    Step := 50;     (* Cleanup: close the destination and source files *)
                ELSE
                    Step := 5; (* Repeat reading/writing *)
                END_IF
            END_IF
        END_IF

30:     (* Close the destination file *)
        fbFileClose( bExecute := FALSE );
        fbFileClose( sNetId:=sDestNetId, hFile:=hDestFile, bExecute:=TRUE, tTimeout:=tTimeOut );
        Step := Step + 1;
```

Figura 59.- Continuación 2 ventana de código FB_FileCopy.

```
        31:
            fbFileClose( bExecute := FALSE );
            IF NOT fbFileClose.bBusy THEN
                IF fbFileClose.bError THEN
                    nErrId := fbFileClose.nErrId;
                    bError := TRUE;
                END_IF
                Step := 50;
                hDestFile := 0;
            END_IF

        40: (* Close source file *)
            fbFileClose( bExecute := FALSE );
            fbFileClose( sNetId:=sSrcNetId, hFile:=hSrcFile, bExecute:=TRUE, tTimeout:=tTimeOut );
            Step := Step + 1;
        41:
            fbFileClose( bExecute := FALSE );
            IF NOT fbFileClose.bBusy THEN
                IF fbFileClose.bError THEN
                    nErrId := fbFileClose.nErrId;
                    bError := TRUE;
                END_IF
                Step := 50;
                hSrcFile := 0;
            END_IF

        50: (* Error or ready => Cleanup *)
            IF ( hDestFile <> 0 ) THEN
                Step := 30; (* Close the destination file*)
            ELSIF (hSrcFile <> 0 ) THEN
                Step := 40; (* Close the source file *)
            ELSE
                Step := 0;      (* Ready *)
                bBusy := FALSE;
            END_IF

    END_CASE
```

7

Figura 60.- Continuación 3 ventana de código FB_FileCopy.

La necesidad de esta función de copia de ficheros surge debido a que, al seleccionar al PLC como sistema objetivo, los archivos se crean en dicho sistema en vez de en local, dificultando el acceso al usuario. Usando esta función se copian los ficheros creados en el disco duro del PLC al sistema local. Bloque de funciones basado en la documentación accesible a través de [62].

8.

MAIN

```
PROGRAM MAIN
VAR
    fbControlServoValvula   : FB_ControlServoValvula;
    fbFileCopy              : FB_FileCopy;
    bCopy                   : BOOL;
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrId                  : UDINT;
END_VAR
```

8

Figura 61.- Ventana de declaración MAIN.

MAIN

8

```
1  //llamada a programa de control
2  fbControlServoValvula();
3
4  PRG_P_TIMER();
5
6  //Funciones en HMI
7  PRG_Stop();
8
9  //Copia archivos del PLC al sistema local según la dirección que se referencie
10 fbFileCopy(sSrcNetId          := '5.63.30.1.1.1',
11           sSrcPathName       := 'C:\test_log.csv',
12           sDestNetId         := '169.254.185.17.1.1',
13           sDestPathName      :='C:\Temp2\test_log.csv',
14           bExecute           := bCopy,
15           bBusy              => bBusy,
16           bError             => bError,
17           nErrId             => nErrId);
18 //END_IF
19 IF NOT bBusy THEN
20     bCopy   := FALSE;
21 END_IF
22
23
```

Figura 62.- Ventana de código MAIN.

A continuación, se muestran las librerías utilizadas para la programación del PLC y el árbol de proyecto:

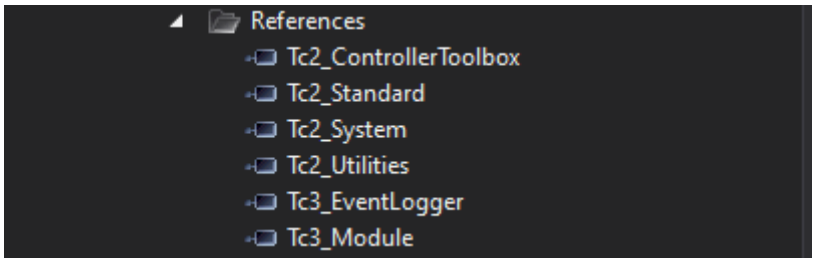


Figura 63.- Librerías TwinCAT utilizadas.

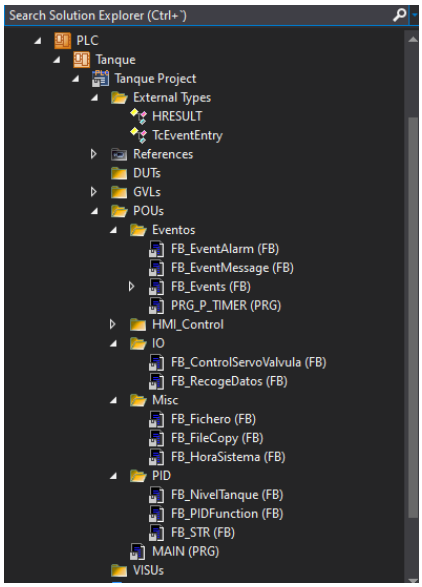


Figura 64.- Árbol de solución PLC.

La solución creada debe estar asignada a un puerto ADS identificable por el sistema. El puerto utilizado es el 851, que es el *runtime* que TwinCAT asigna por defecto.

ADS device	Port number
Cam controller	900
Runtime system 1 Runtime system 2 Runtime system 3 Runtime system 4 Runtime system 5 Runtime system n	Runtime system 1: 851 (in TwinCAT 2: 801) Runtime system 2: 852 (in TwinCAT 2: 811) Runtime system 3: 853 (in TwinCAT 2: 821) Runtime system 4: 854 (in TwinCAT 2: 831) Runtime system 5: 855 Runtime system n: 850 + n, and so on
NC	500
Reserved	400
I/O	300
Real-time core	200
Event System (logger)	100

Tabla 7.- Tabla de puertos ADS. Tabla extraída de [63]

2.8.4 Monitorización gráfica

Con el fin de observar el comportamiento del sistema debemos administrar una solución gráfica que ayude a la visualización de las variables de control. Para ello, TwinCAT Scope View, junto con **TF3300|Scope Server**, ofrece el monitorizado de gráficas según las variables escogidas por el usuario.

Creamos entonces una solución de TwinCAT Measurement:

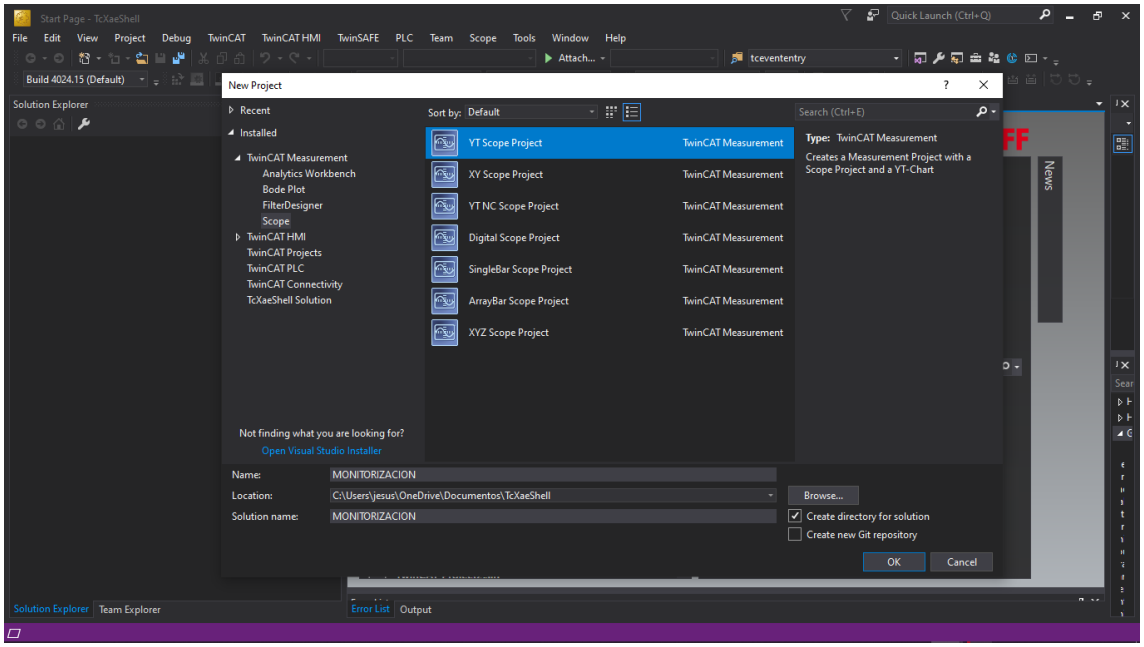


Figura 65.- Creación de proyecto TwinCAT Measurement.

Una vez generada la solución, creamos un “YT Chart”:

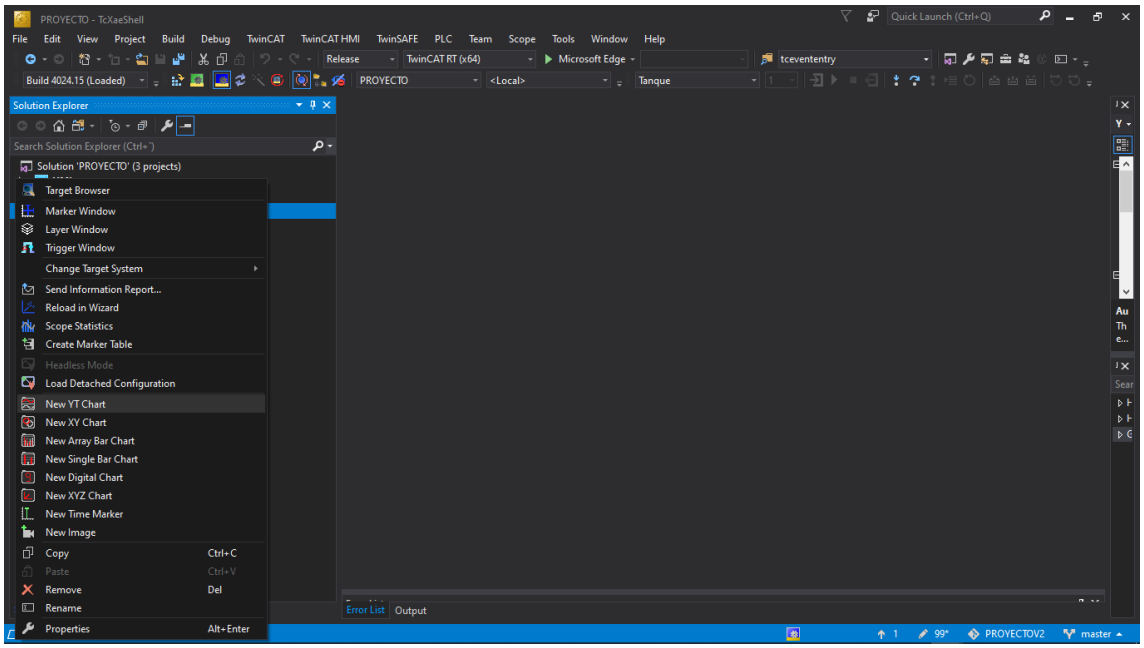


Figura 66.- Creación de YT Chart.

Esto nos crea un gráfico vacío. Para seleccionar las variables que queremos que se representen, es necesario, en primer lugar, seleccionar de qué sistema queremos que se extraigan dichas variables. En este caso tanto el PLC como el sistema local son viables para el muestreo:

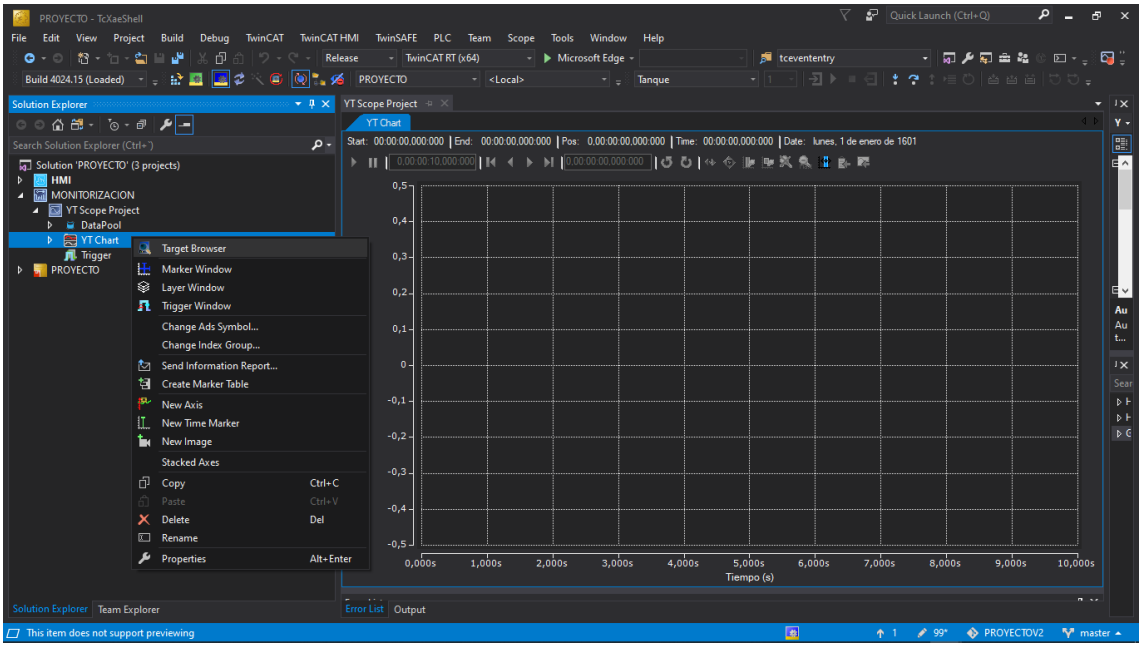


Figura 67.- Selección de sistema.

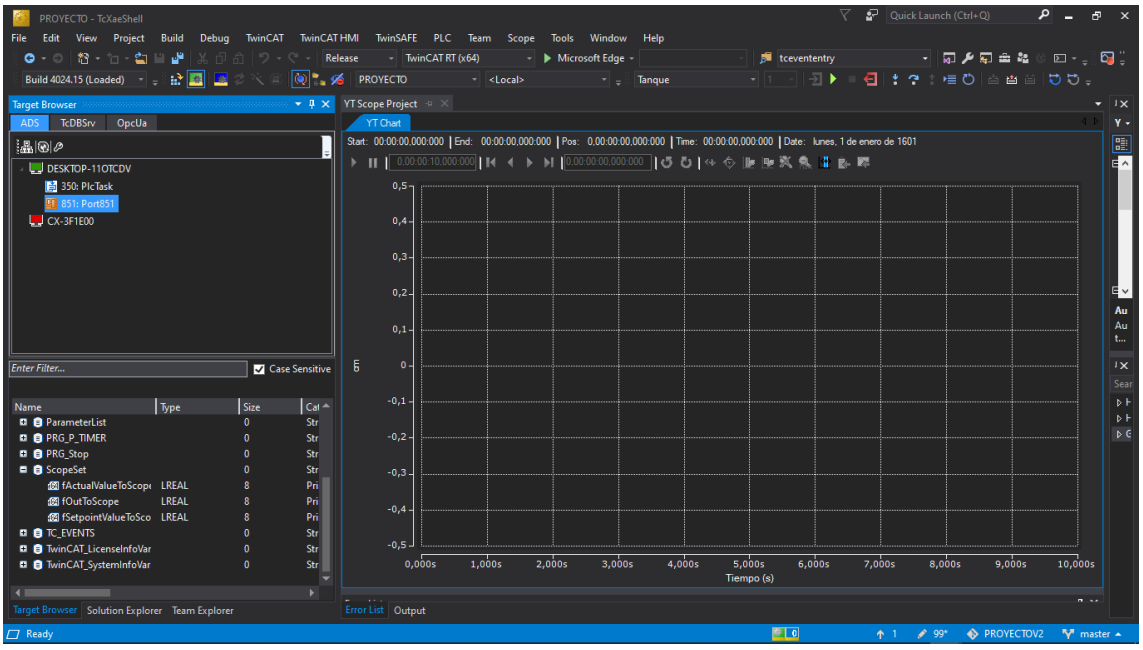


Figura 68.- Selección de puerto y variables.

TwinCAT debe estar en modo "*online*" para que las variables sean visibles.

Para acceder con facilidad a las variables deseadas (salida del PID, valor de consigna y valor actual), se han creado tres variables globales que contienen los valores que se quieren graficar:

ScopeSet

```
{attribute 'qualified_only'}
VAR_GLOBAL
    fSetpointValueToScope    :    LREAL;
    fActualValueToScope      :    LREAL;
    fOutToScope              :    LREAL;
END_VAR
```

Figura 69.- Variables globales de gráfico.

El gráfico se actualizará en tiempo real en función del valor que adquieran las variables:

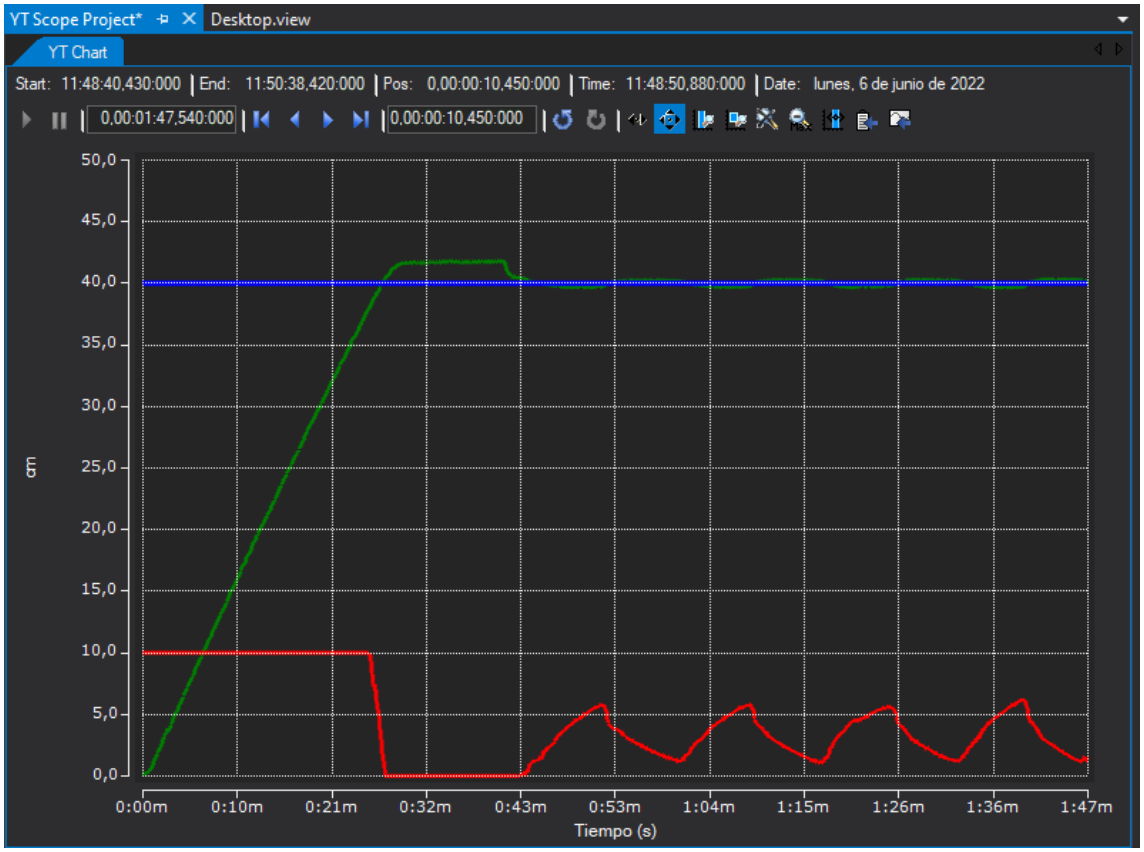


Figura 70.- Ejemplo gráfico de comportamiento.

2.8.5 SCADA/HMI

Con el desarrollo de un SCADA es posible la supervisión, tanto numérica como visual, del nivel del tanque de nuestra planta, de los valores del controlador PID implementado, ajustables desde la propia interfaz y la monitorización gráfica del sistema de control con el fin de observar su estabilidad, así como un gestor de eventos en tiempo real.

Para su incorporación dentro del entorno de TwinCAT ha sido necesaria la instalación de la extensión **TE2000|HMI Engineering**, usada para la creación de soluciones tipo TwinCAT HMI:

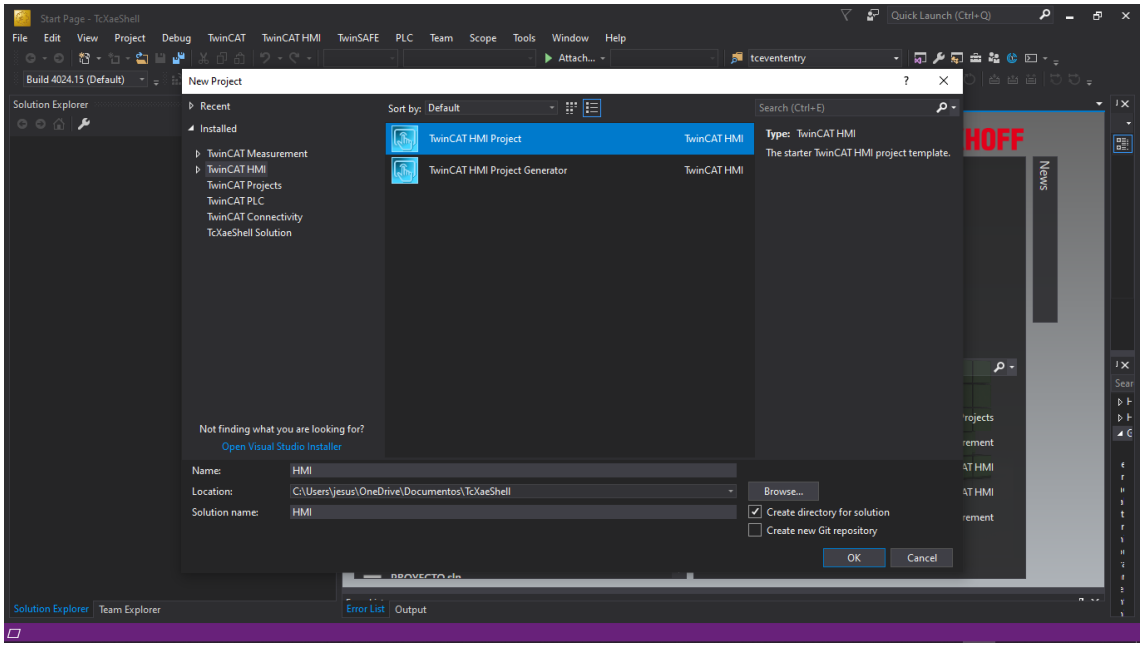


Figura 71.- Creación de proyecto TwinCAT HMI.

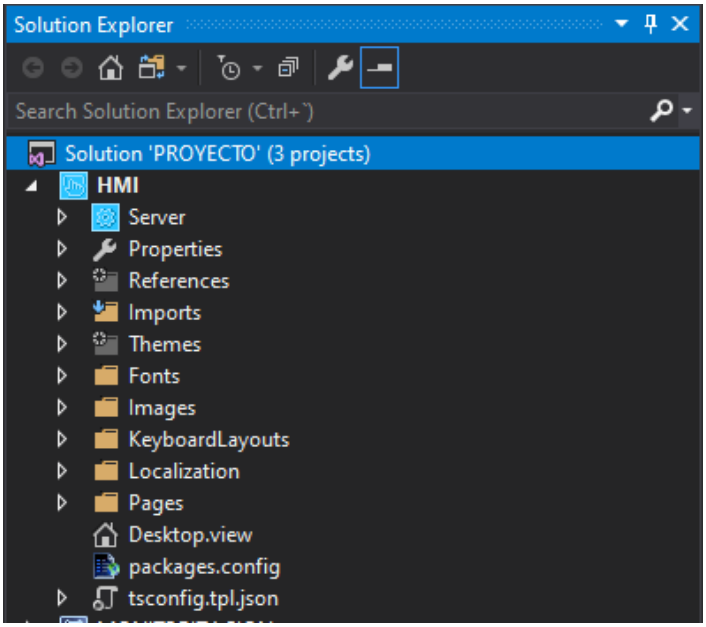


Figura 72.- Generación de solución HMI con estructura TwinCAT.

La visualización de la HMI ocurre en *Desktop.view*. Al ser una solución basada en HTML5 y JavaScript, existe la posibilidad de programar la funcionalidad y estructura en estos lenguajes, sin embargo, Beckhoff ofrece un *Toolbox* que contiene las herramientas que el desarrollador pueda necesitar (como botones, bloques de texto...) para el desarrollo de la HMI.

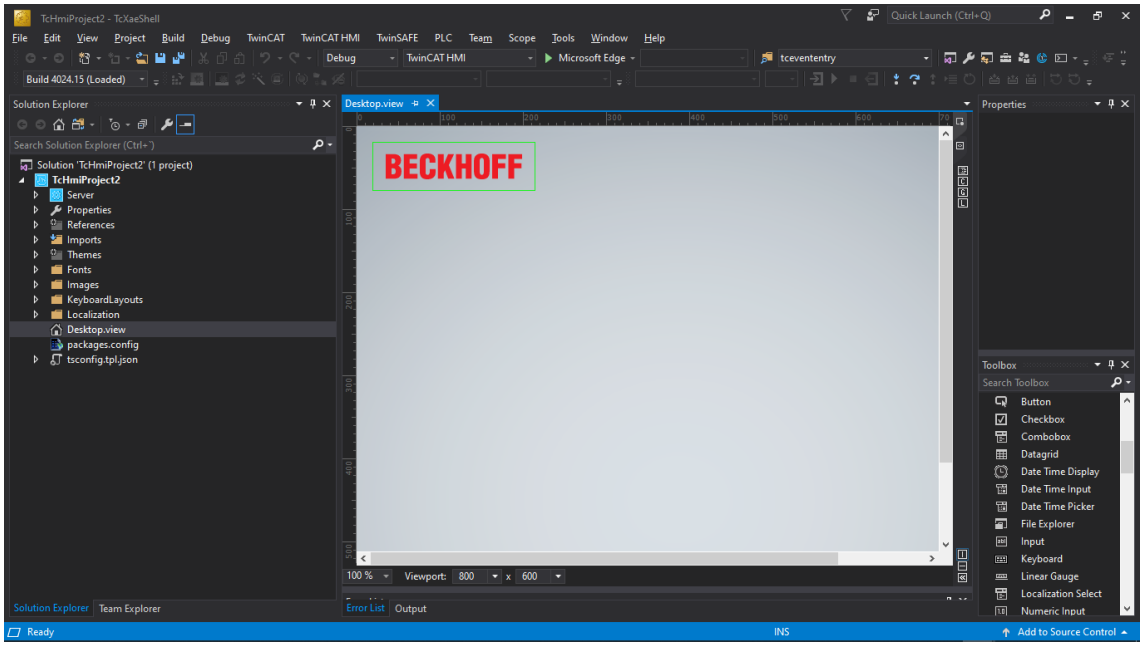


Figura 73.- Primera instancia de Desktop.view.

Los elementos del *Toolbox* empleados para nuestra solución se muestran a continuación:

- **Button:** Simple botón, su estado cambia solo al ser pulsado.
- **Toggle Button:** Botón que cambia entre dos estados.
- **Textblock:** Bloque de texto, permite la inclusión de texto dentro de él.
- **Combobox:** Menú desplegable.
- **Spinbox Input:** Entrada de valores numéricos con botones para incrementar y disminuir sus valores.
- **Region:** Contenedor de contenido, hace posible la implementación de “páginas”.
- **Date Time Display:** Muestra la fecha y hora local.
- **ADS State:** Muestra el estado de TwinCAT (Run/Config).
- **Image:** Imágenes importadas adjuntadas a la solución.
- **Linear Gauge:** Representación visual de la variable de proceso (nivel).
- **Theme Select:** Menú desplegable para la selección de Tema.
- **Localization Select:** Menú desplegable para la selección de idioma.
- **Scope Control:** Muestreo de TwinCAT Scope View.
- **File Explorer:** Explorador de archivos.
- **Event Grid:** Muestreo de eventos con opciones de filtro.
- **Container:** Contenedor de controles.

Los elementos mencionados pueden ser personalizados a través de la ventana de propiedades, donde es posible editar el tamaño, color, texto, bordes y otras propiedades a gusto del desarrollador. La ventana de propiedades también incluye una sección de eventos que habilita al desarrollador con la posibilidad de ajustar la lógica funcional de cada elemento (acciones y condiciones).

Hay que tener en cuenta que la gestión numérica y de cálculo está ocurriendo en una solución diferente (PLC), por lo que va a ser imprescindible mapear las variables que vayamos a utilizar en la HMI. Para hacer esto, accedemos a la configuración de TwinCAT HMI y seleccionamos las variables que vamos a utilizar en nuestra solución.

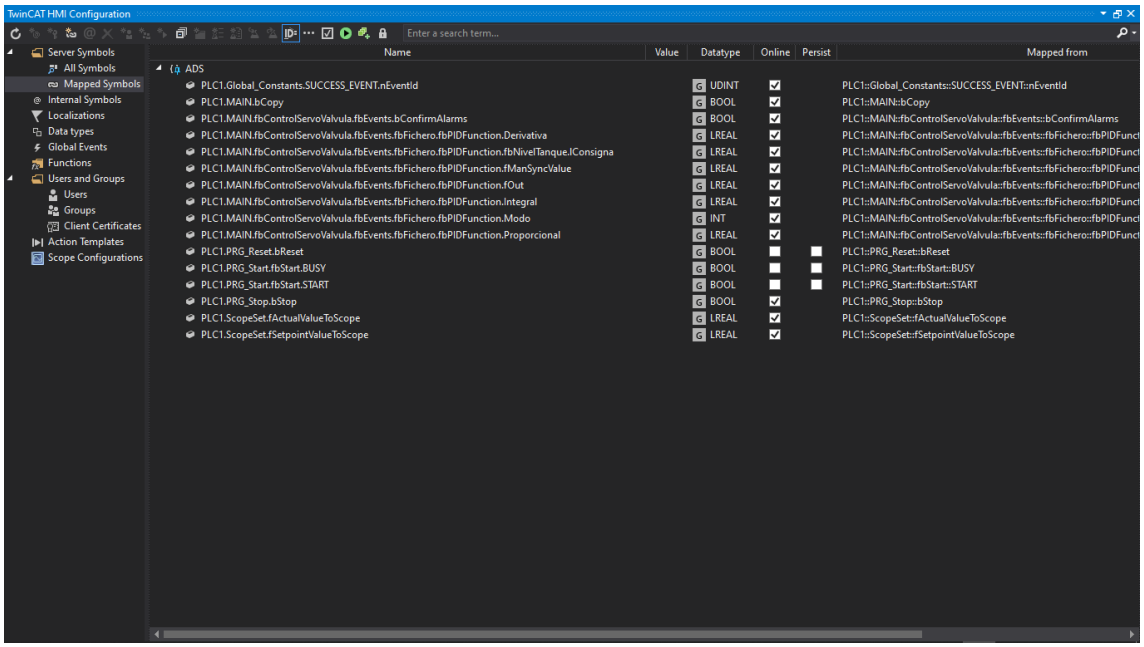


Figura 74.- Variables mapeadas en solución TwinCAT HMI.

Empleando las variables mapeadas junto a la capacidad de ajuste lógico de los elementos del *Toolbox* podemos crear una solución adaptada a nuestras necesidades de control.

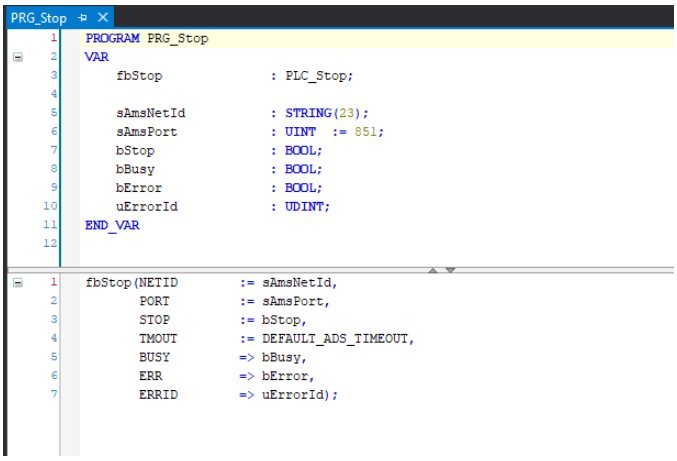


Figura 75.- Programa de botón de Stop.

Para organizar adecuadamente todo el contenido necesario en la HMI (visualización gráfica, gestor de eventos, control PID…) de modo que sea fácil de navegar, se hace uso de los contenedores *Region*, que permite la adición de contenidos (*content*) dentro de un espacio delimitado por el desarrollador en *Desktop.view*, creando efectivamente páginas accesibles a través de un menú creado con *Textblocks*.

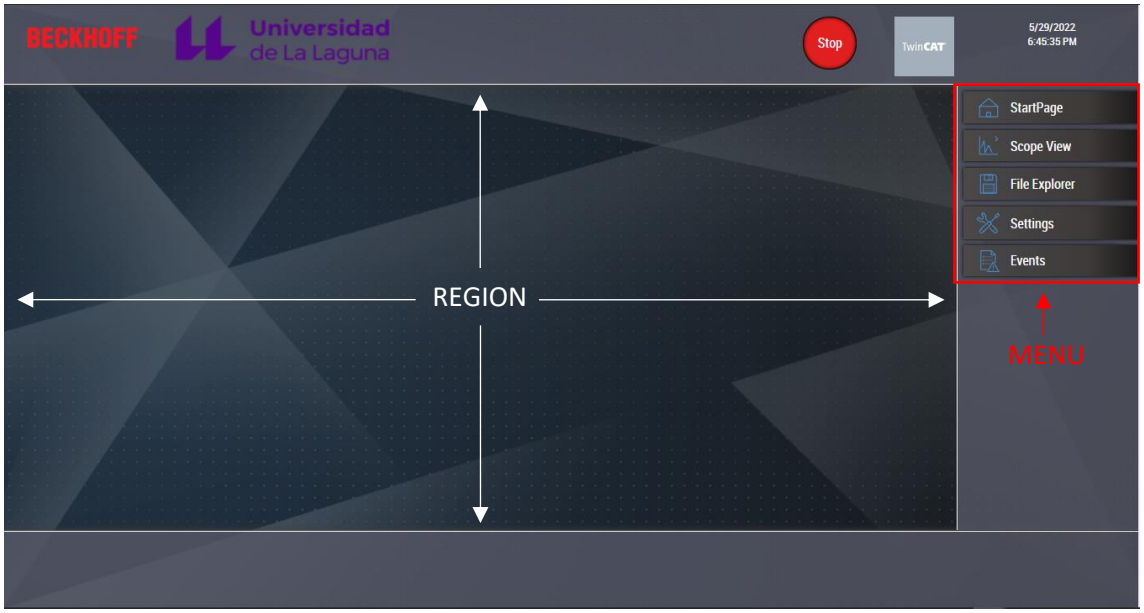


Figura 76.- Página de inicio de HMI desarrollada.

Antes de asignar la funcionalidad de cada *Textblock* del menú, hay que crear las páginas de contenido que van a ocupar la zona de región que se ha delimitado. Para realizar esto, se ha creado una nueva carpeta llamada *pages* dentro de la solución de TwinCAT HMI. Esta carpeta archivará las páginas de contenido que vamos a utilizar para la visualización de cada módulo de la HMI.

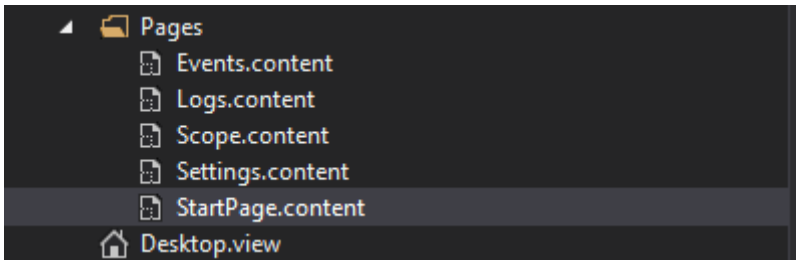


Figura 77.- Carpeta de páginas de contenido.

Una vez creadas, accedemos a las propiedades de cada *Textblock*, que vamos a emplear como menú para navegar a través de estas páginas, y adjuntamos, para cada uno, la acción de acceder a estos contenidos al ser pulsados.

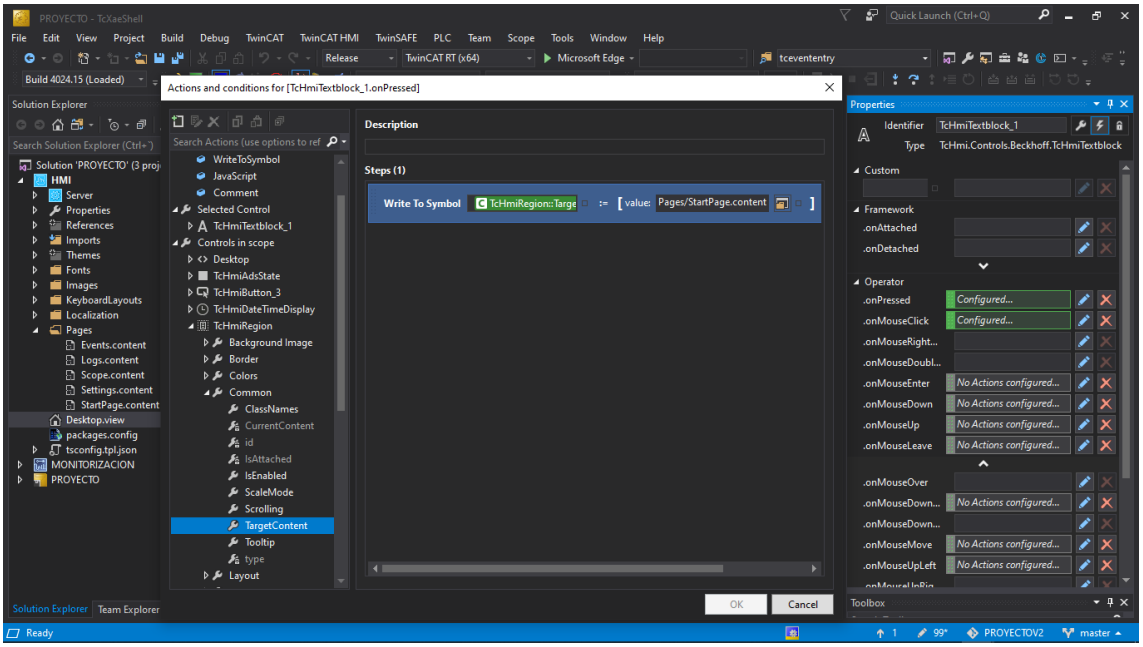


Figura 78.- Asignación de acción onPressed para StartPage.

Hacemos esto para cada *Textblock* y sus correspondientes páginas de contenido, creando así un menú completamente funcional. Debido a que solo puede existir una página de contenido por *Region,* nunca se dará la situación en la que las páginas se superponen, permitiendo que la HMI sea íntegramente modular.

Queda entonces la edición de cada página de contenido, que se ha hecho en función de a qué funcionalidad querría acceder un usuario:

1. **StartPage:** Es la página que contiene los controles del PID y la visualización de un tanque de líquido.
2. **Scope View:** Contiene la monitorización gráfica.
3. **File Explorer:** Es un simple explorador de archivos.
4. **Settings:** Contiene opciones de carácter misceláneo, como idioma y tema.
5. **Events:** Aquí se pueden visualizar los eventos, alarmas y mensajes del sistema.

Se muestran a continuación los detalles de cada página de contenido:

1.

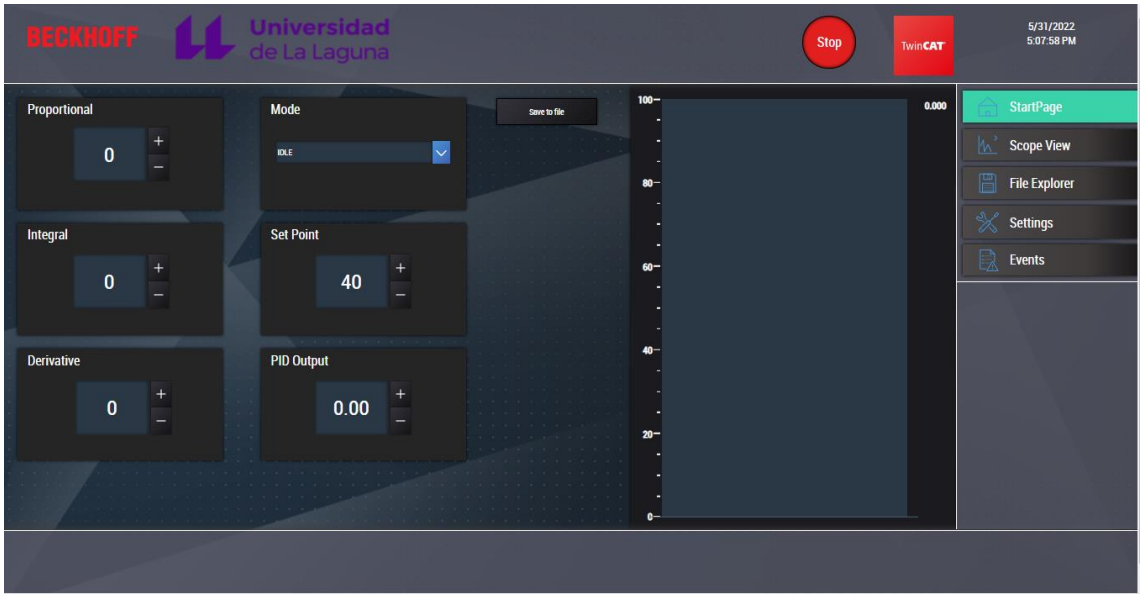


Figura 79.- Contenido de StartPage.

El contenido de *StartPage* refleja las opciones de control del controlador PID, incluyendo un visor de nivel en tiempo real. Los valores del controlador son ajustables a través de *Spinbox Input*, y, el modo, seleccionable con un menú desplegable *Combobox*. Para el ajuste lógico de cada elemento se ha procedido de la siguiente forma: *Proportional, Integral, Derivative* y *Set Point* son similares en funcionalidad (cuando el valor del PLC cambia, también debe cambiar en la HMI y viceversa), aunque las variables a las que han sido adjuntados difieren. *PID Output* es similar, con la excepción de que no debe ser ajustable a menos que el modo se encuentre en manual. *Mode* recoge los modos de control del PID (*IDLE, PASSIVE, RESET, MANUAL, ACTIVE*). Cada modo lleva asignada una ID diferente que permite diferenciarlos. En cuanto al visualizador del tanque de nivel, se ha usado una *Linear Gauge* ajustada de forma vertical y recibe tanto los valores de consigna como de valor actual. El botón *Save to File* guarda todos los valores aquí presentes en un *fichero.csv*.

2.

Figura 80.- Contenido de Scope View.

La implementación gráfica de TwinCAT Scope View al entorno TwinCAT HMI puede realizarse gracias a dos factores, la extensión ***TcHmiScope***, descargable desde el propio entorno de TwinCAT, provee al *Toolbox* el elemento *Scope Control*, utilizado para adjuntar nuestro proyecto de monitorización creado, y a **TwinCAT 3 Scope Server**, que prepara la visualización.

3.

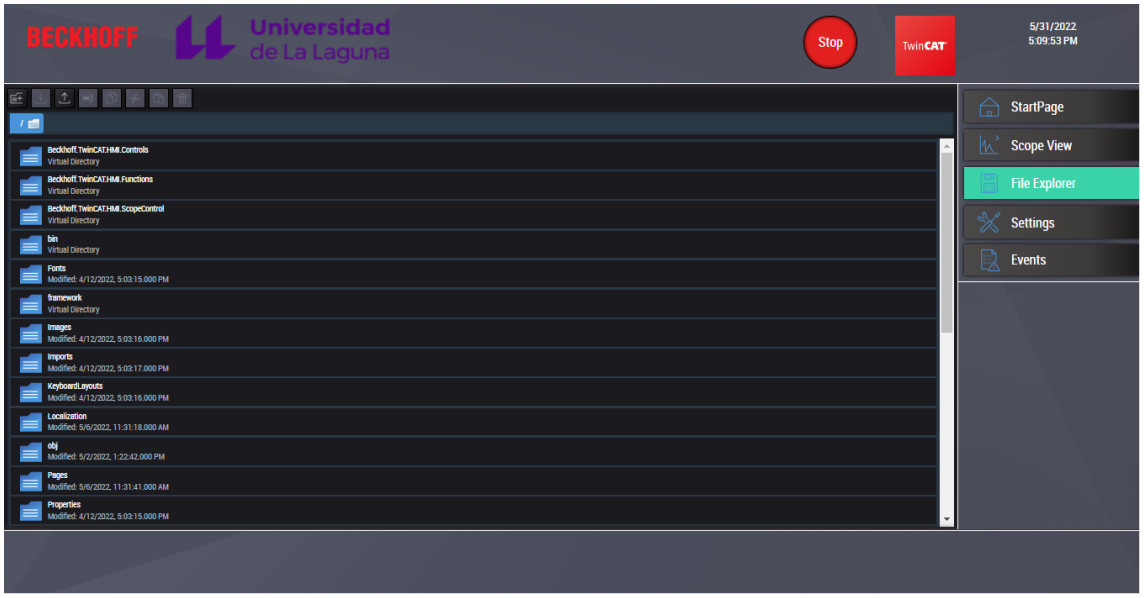


Figura 81.- Contenido de File Explorer.

4.

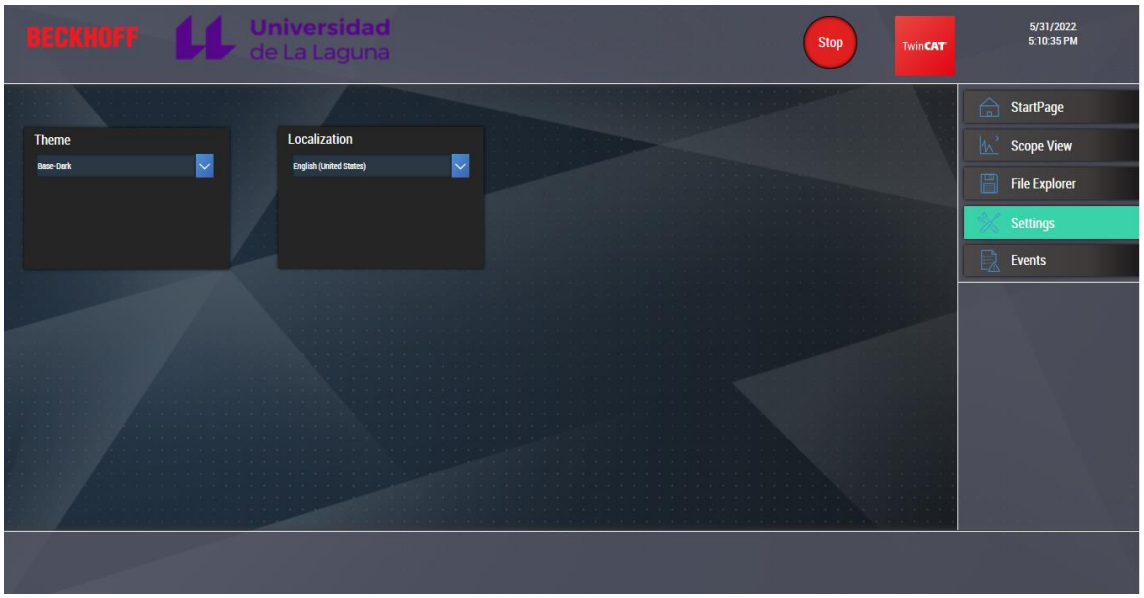


Figura 82.- Contenido de Settings.

Los temas e idiomas son seleccionables mediante un menú desplegable. Los temas utilizados son los que vienen por defecto (*Base, Base-Dark)*, ligeramente personalizados para la corrección de algunos colores, así como la adición de botones y bloques de texto personalizados.

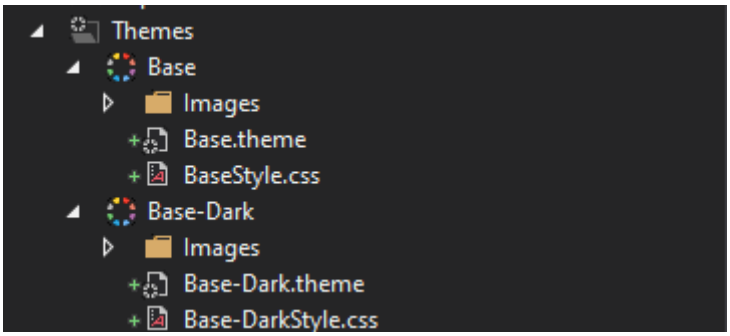


Figura 83.- Carpeta de Themes.

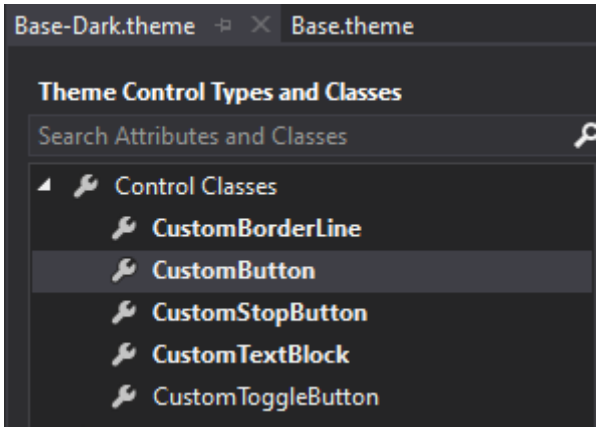


Figura 84.- Elementos personalizados.

Los idiomas son editables desde la carpeta de *Localization*, donde es posible añadir *strings* de texto identificados por IDs. Estos *strings* pueden ser adjuntados como texto en la ventana de propiedades de cada elemento que requiera de localización.

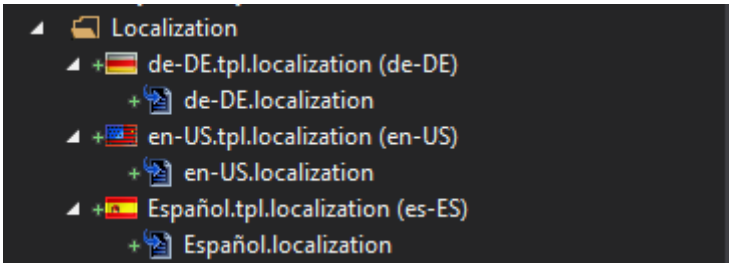


Figura 85.- Carpeta de Localization.

Key	en-US
Text3	StartPage
Text4	Scope View
Text5	File Explorer
Text6	Events
Text7	Settings
Text8	Theme
Text9	Localization
Text10	Proportional
Text11	Integral
Text12	Derivative
Text13	Mode
Text14	Set Point
Text15	PID Output
Text16	Pending alarms
Text17	PID Alarms
Text18	PID Messages
Text19	All PID events
Text20	Export events
Text21	Delete events
Text22	Confirm alarms
Text23	All ADS events
Text24	All Scope events
Text25	All Server events
Text26	Stop
Text27	All events
Text28	Save to file

Figura 86.- Localización Inglés.

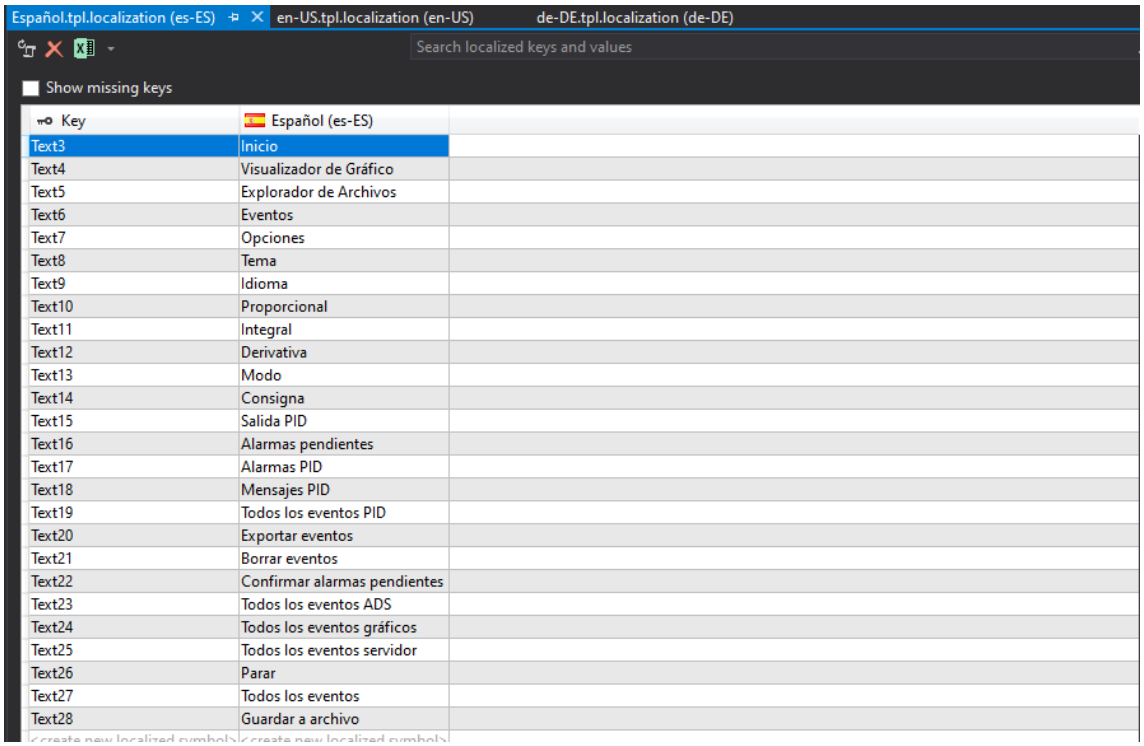


Figura 87.- Localización Español.

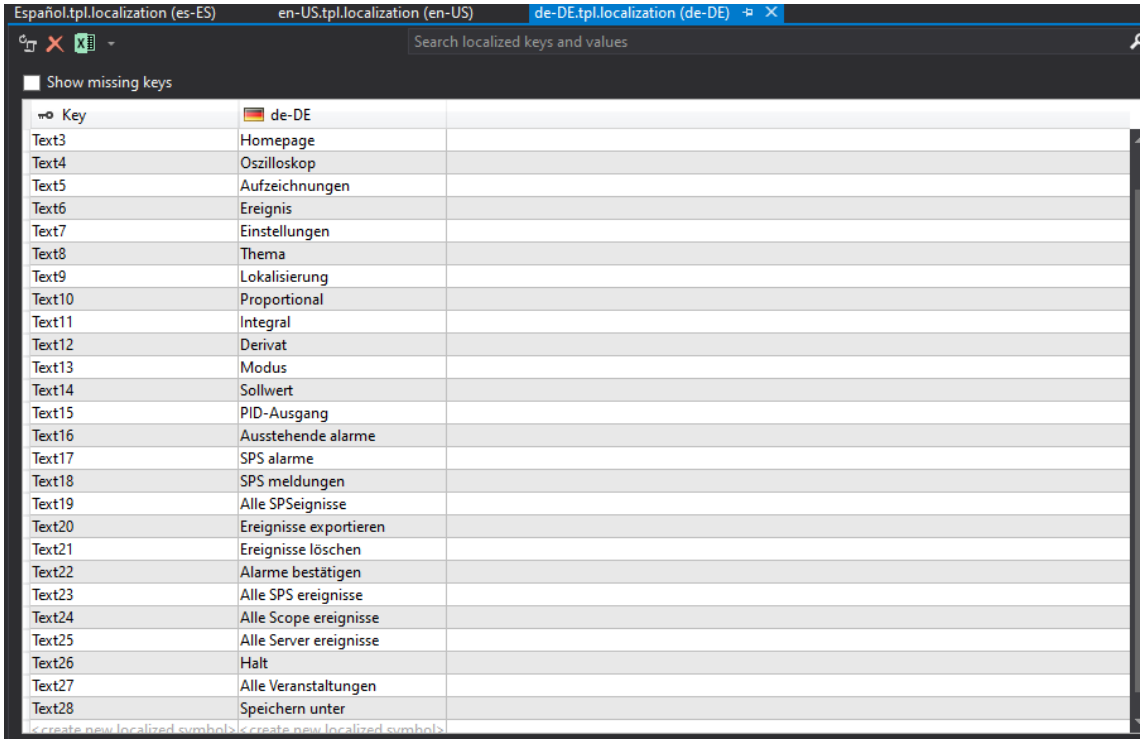


Figura 88.- Localización Alemán.

5.

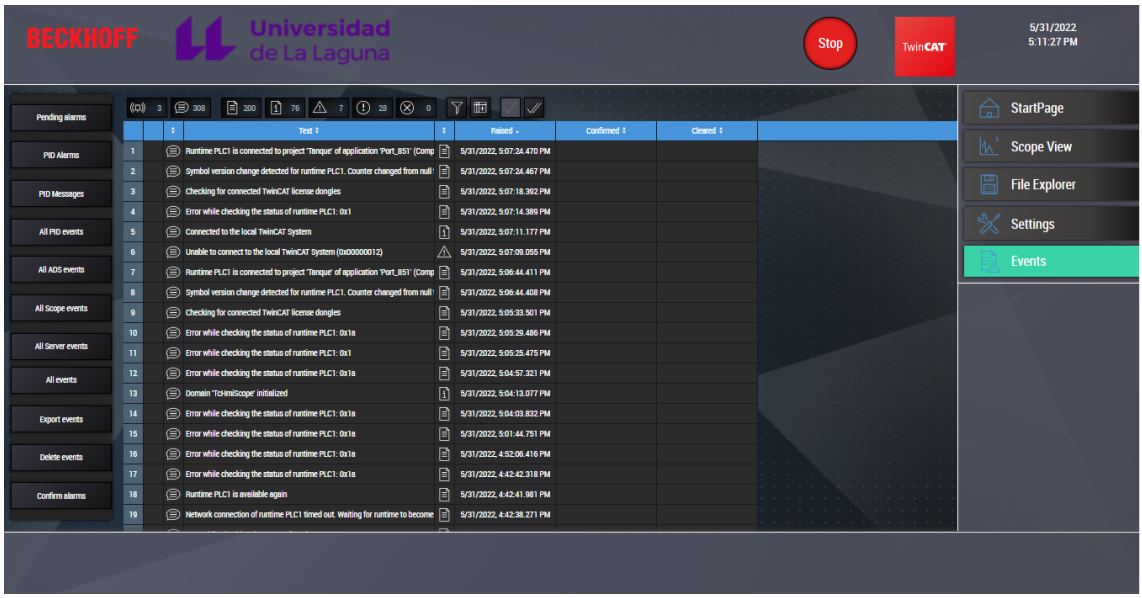


Figura 89.- Contenido de Events.

Con un *Event Grid* mostramos los eventos del PLC y, haciendo uso de la extensión ***TcHmiEventLogger***, descargable desde el entorno de desarrollo de TwinCAT, podemos mostrar los eventos recogidos por el archivador de eventos desarrollado en la solución PLC. También se incluyen *Toggle buttons* para el filtrado de eventos y *Buttons* para exportar, borrar y confirmar eventos y alarmas.

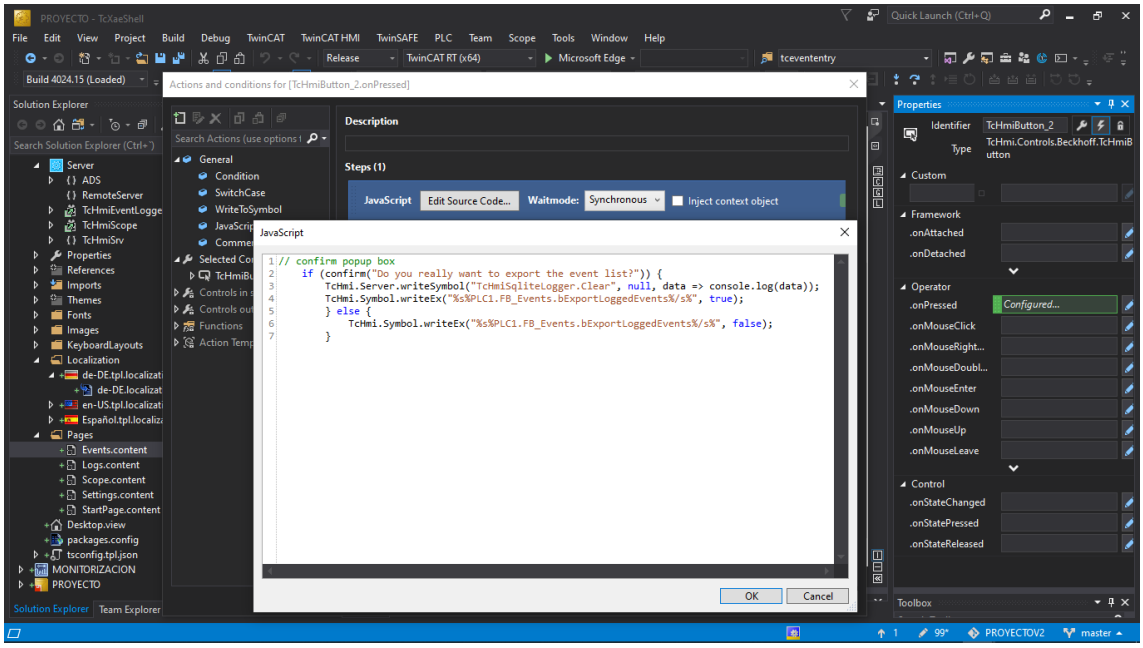


Figura 90.- Código JavaScript para exportar eventos.

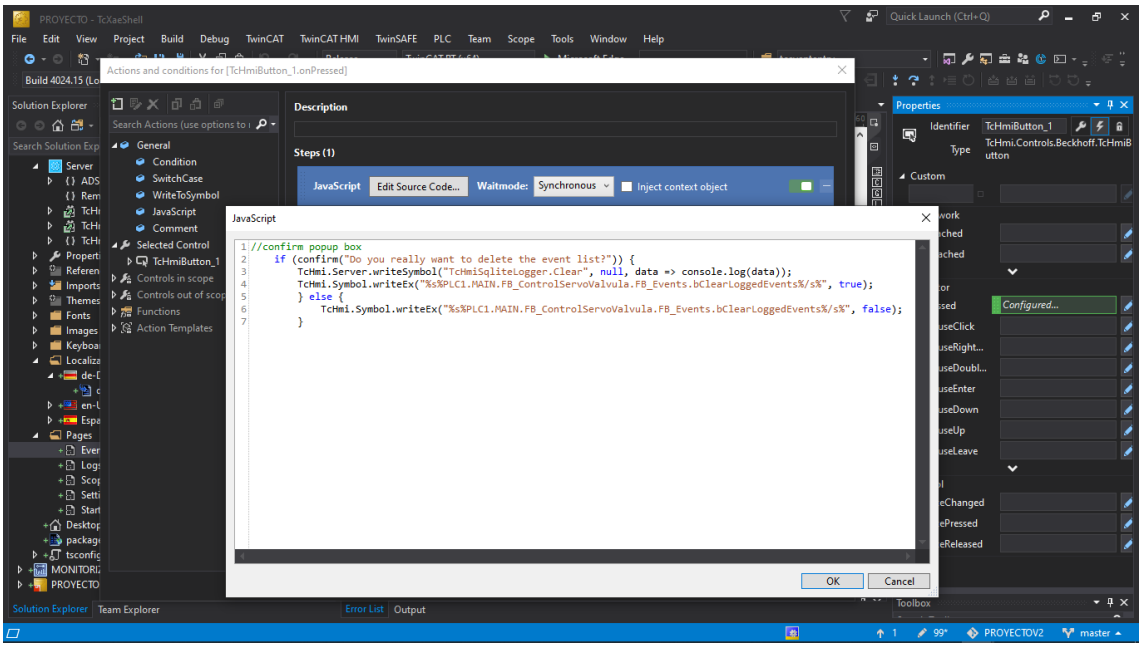


Figura 91.- Código JavaScript para borrar eventos.

El uso de JavaScript es debido a la necesidad de creación de una ventana emergente de confirmación al pulsar los respectivos botones y para actualizar el estado del *Event Grid*.

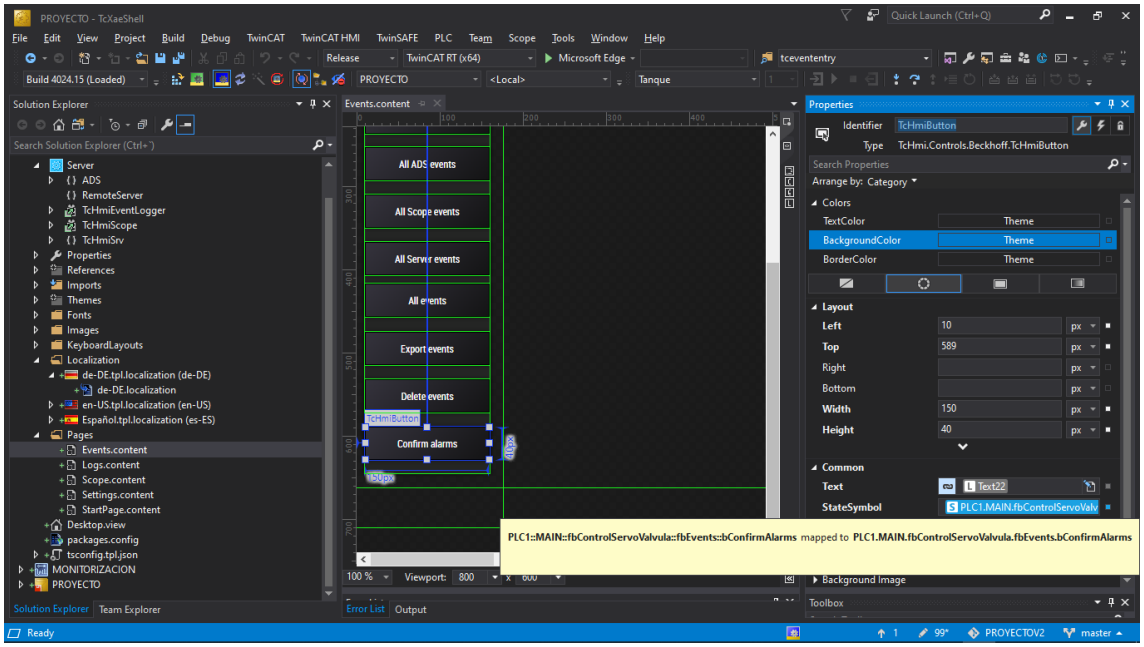


Figura 92.- Botón de confirmado de alarmas mapeado.

Una vez terminada nuestra HMI, debemos establecer la conexión ADS, a través de la configuración de servidor de TwinCAT HMI, al dispositivo al que queremos integrar la interfaz, en nuestro caso corresponde al PC de Beckhoff CX5130.

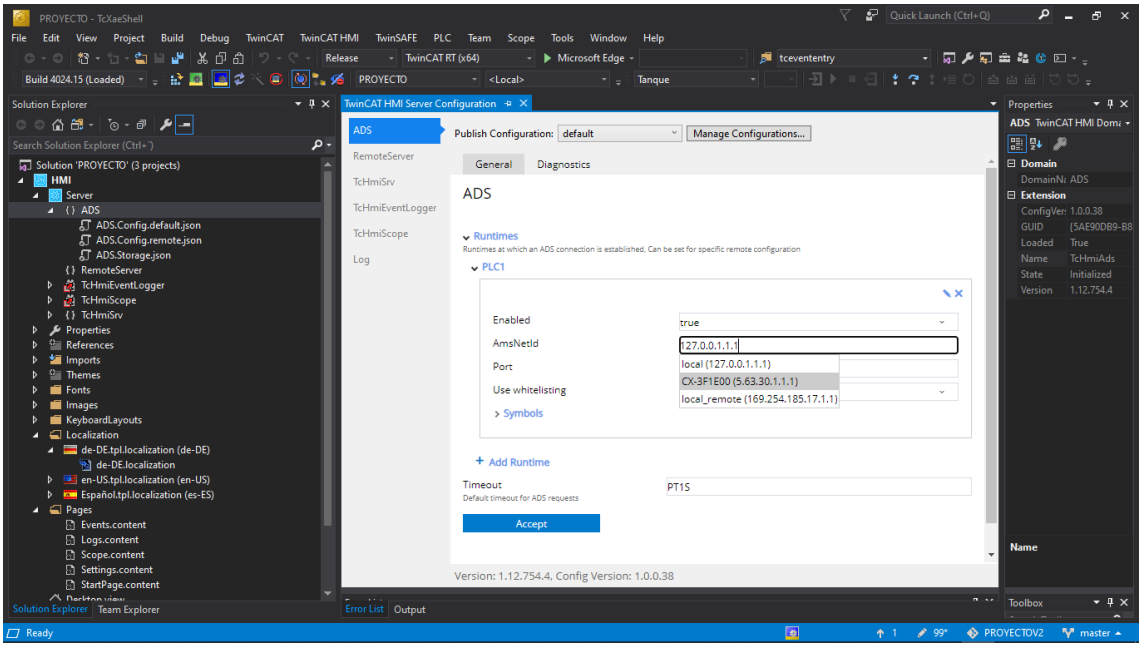


Figura 93.- Establecimiento de conexión ADS a PC embebido.

2.8.6 Panorama final

Se ha logrado el objetivo propuesto, habiendo desarrollado un programa de control, junto a un sistema de supervisión que permite el control de la planta de nivel de líquidos del laboratorio.

Las tres soluciones TwinCAT establecidas en el controlador programable industrial de Beckhoff CX5130 se complementan entre sí para dar al usuario las opciones de supervisión que pueda requerir el sistema.

El carácter modular de TwinCAT ha facilitado el desarrollo de cada solución de forma independiente, y, gracias a su flexibilidad, ha sido posible la implementación de cada una en un sistema de supervisión y control.

Escuela Superior de Ingeniería y Tecnología

Sección Ingeniería Industrial

Universidad de La Laguna

Trabajo Fin de Grado

Grado en Ingeniería Industrial y Automática

Diseño e implementación de controladores Beckhoff en maquetas de procesos de control

3. PRESUPUESTO

Estudiante: Jesús Elías Soto

Tutor: Santiago Torrez Álvarez

Cotutor: Juan Albino Méndez Pérez

Convocatoria: Junio 2022

3.1 Presupuesto

En las siguientes tablas se recogen el presupuesto del material, mano de obra y total de este Trabajo Fin de Grado.

PRESUPUESTO MATERIAL			
Equipo	Descripción	Cantidad	Importe (€)
CX5130	Controlador Programable industrial de Beckhoff	1	1.675,79
EL3064	Módulo de entradas analógicas	1	199,99
EL4002	Módulo de salidas analógicas	1	219,95
Process Rig 38-100	Planta de flujo y nivel	1	799,99
Project Board GL-48	Protoboard	1	38,45
FAC-363B	Fuente de alimentación	1	544,50
LM741	Amplificador operacional	3	1,50
LM358	Amplificador operacional	2	0,84
Resistencia		12	1,20
Potenciómetro		2	0,80
Cable banana-banana		4	13,00
Cable cocodrilo-banana		2	2,38
Cable cocodrilo-cocodrilo		2	1,95
6XV1870-3RH20	Cable Ethernet Industrial	1	23,22
		SUBTOTAL	**3.523,56**

Tabla 8.- Presupuesto material.

PRESUPUESTO MANO DE OBRA		
Horas	Coste/hora (€/h)	Total (€)
300	30,00	9.000,00

Tabla 9.- Presupuesto mano de obra.

PRESUPUESTO TOTAL	
Descripción	Importe
Material	3.523,56
Mano de obra	9.000,00
SUBTOTAL	**12.523,56**
13% Gastos generales	1.628,07
6% Beneficio industrial	751,41
SUBTOTAL	**14.902,98**
7% IGIC	1.043,21
TOTAL	**15.946,19**

Tabla 10.- Presupuesto total.

Escuela Superior de Ingeniería y Tecnología

Sección Ingeniería Industrial

Trabajo Fin de Grado

Grado en Ingeniería Industrial y Automática

Diseño e implementación de controladores Beckhoff en maquetas de procesos de control

4. CONCLUSIONES

Estudiante: Jesús Elías Soto

Tutor: Santiago Torrez Álvarez

Cotutor: Juan Albino Méndez Pérez

Convocatoria: Junio 2022

4.1 Conclusiones

En este Trabajo Fin de Grado se ha trabajado en una solución a un problema de control de una planta de nivel de líquidos haciendo uso de un controlador programable industrial de Beckhoff en conjunto a su software TwinCAT, comenzando, en primer lugar, con el conexionado entre el controlador y la planta mediante el diseño de un circuito de conversión, seguido del desarrollo del software de control, que puede ser divido en dos etapas: la solución PLC, que contiene la lógica de control y , en segundo lugar, el diseño de un SCADA empleado para facilitar la visualización y control de datos del sistema.

Se han cumplido con los objetivos principales, habiendo trabajado tanto en el entorno de desarrollo de PLCs como de SCADA/HMI al implementar una solución de control para la planta de nivel.

Los problemas encontrados durante el desarrollo del TFG están principalmente relacionados con la programación del PLC dentro del software de TwinCAT, debido a la dificultad que aparece al trabajar con entornos de desarrollo con los que no se han trabajado de forma continua con anterioridad, ligado al aprendizaje de un lenguaje de programación no visto antes, han hecho del desarrollo del proyecto algo mucho más enriquecedor, permitiéndome poner a prueba mis habilidades adquiridas durante la carrera. Dicho esto, lo más destacable para mí de este proyecto ha sido la posibilidad de adquirir unas primeras impresiones del desarrollo de SCADA Web y HMI debido a su importancia en la industria hoy en día.

4.2 Conclusions

In this Final Degree Project, a solution to a control problem for a liquid level control system has been made by making use of a Beckhoff industrial programmable controller along with its default software, TwinCAT, beginning with the connections between the controller and the system after designing a conversion circuit, followed by the development of the control software, divided into two parts: the PLC solution, which contains the control logic and, furthermore, the design of a SCADA used to facilitate the visualization and data control of the system.

The main objectives have been accomplished, having worked on both the development of the PLC and SCADA/HMI by implementing a control solution for the liquid level control system.

The problems found during the development of the FDP are mainly related to the programming of the PLC within the TwinCAT software, due to the difficulty that arises from working with development environments with which haven't been worked with frequently, adding to the fact that there was a need to learn a programming language that had not been seen before, they have made the project much more fulfilling, allowing me to put the skills that I have learnt throughout my career to the test. Having said that, the most remarkable thing for me after working on this project has been the possibility of acquiring first impressions regarding the development of SCADA Web and HMI due to its importance in the industry nowadays.

Escuela Superior de Ingeniería y Tecnología

Sección Ingeniería Industrial

Trabajo Fin de Grado

Grado en Ingeniería Industrial y Automática

Diseño e implementación de controladores Beckhoff en maquetas de procesos de control

5. ANEXOS

Estudiante: Jesús Elías Soto

Tutor: Santiago Torrez Álvarez

Cotutor: Juan Albino Méndez Pérez

Convocatoria: Junio 2022

5.1 Anexo I

Hoja de datos de planta de nivel

Feedback

Engineering Teaching Solutions

PROCON - Level/Flow Process Control 38-001

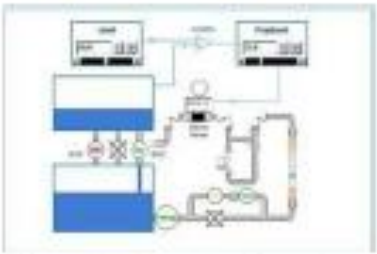
espial

Description

The Level/Flow Process Trainer is a single loop system allowing the study of the principles of process control, using liquid level and flow rates as the measured process variables. The system is a completely self-contained, low pressure flowing water circuit supported on a benchtop-mounted panel, making it suitable for individual student work or for group demonstrations.

It comprises a dual compartment process tank, linked to a sump tank by manual and solenoid operated valves. Water is pumped through the system, via a variable area flow meter and motorised control valve. Level is measured in the process tank. Flow is measured through an optical pulse flowmeter.

The PC Computer and Monitors are not supplied.



System with Proportional Control

Features

- Contains a selection of level and flow sensors & indicators
- Flow controlled by linear motorised control valve
- On/Off and proportional control
- P, PI and full PID control with autotune facility
- Couples with Temperature Trainer for dual loop control
- Modern push fittings
- Water used as the process fluid
- Comprehensive lab notes and ESPIAL Software

Curriculum Coverage

- Flow & Level familiarisation and calibration
- Interface familiarisation and calibration
- Controller familiarisation and calibration
- Float level transmitter
- Pulse flow transmitter
- On-Off control
- Study of P, PI and PID control of Level and Flow
- Tuning PID controllers
- Advanced process control

Process Controller (top) with Process Interface

Required Equipment - supplied with all Procon Training Systems

Process Interface 38-200

The Process Interface is connected to the system and provides all necessary power outlets for the Process Trainer, sensors and Process Controller. It accepts up to four 4-20 mA transmitter signal inputs and allows signal patching so that different control schemes can be quickly configured. It also provides a 4-20 mA current source, two current to voltage converters and a voltage comparator with adjustable hysteresis which can be used to provide a simple 2-state control loop in addition to the main controller loop. Protection is provided by a residual current circuit breaker.

Process Controller 38-300

The ABB Industrial Process Controller contained within the Process Controller is microprocessor based and is easily con- figured by the user to provide a range of control functions from 2-state control to 3-term PID control. It also features local or remote set-point, re-transmission of set-point or process variable, 4 logic inputs, 4 relay outputs, ramp/soak (profile sequencing) and an autotune facility which can analyse the requirements of a process and configure the control parameters for optimum performance. Together with the Process Interface, it provides a simple and convenient means of controlling the system.

Assignments

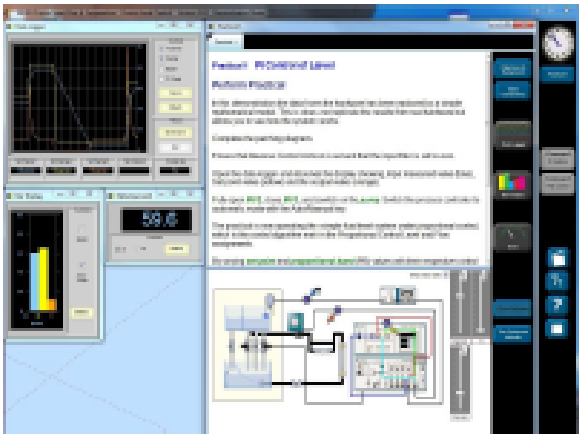
The ESPIAL Software assignments provided with the PROCON Level/Flow Process Control System are:

- **Introduction to PROCON**
- **Flow/Level Rig Familiarisation**
  The Centrifugal Pump.
  The Manual Valves and the Flow Gauge.
  The Servo Valve.
  The Solenoid Valves.
- **Flow/Level Rig Calibration**
  A Level-Volume Correspondence.
  Flow Meter Calibration.
  Servo Valve Calibration.
  Solenoid Valve Calibration.
- **Interface Familiarisation**
  Circuit Breaker & Circuit Loop Connections.
  The Servo Valve.
  The Current-Voltage Converters.
- **Interface Calibration**
  Current Source Calibration.
- **Controller Familiarisation**
  Serial Communication.
  Navigating the Controller.
  Using the Controller.
- **Controller Calibration**
  Controller Calibration.
  Controller Relays.
  Reading the Controller.
- **Float Level Transmitter**
  The Float Level Transmitter (FLT).
  Calibrating the FLT.
  A Level Control Demonstration.

- **Pulse Flow Transmitter**
  The Pulse Flow Transmitter (PFT).
  Calibrating the PFT.
  A Flow Control Demonstration.
- **On/Off Control**
  On/Off Pump Control.
  On/Off Solenoid Control.
  The Float Switch.
  Controller On/Off Control.
- **Proportional Control: Level**
  Simulation.
  Proportional Control of Level.
  Proportional Control and Offset.
  Proportional Band.
- **Proportional Control: Flow**
  Servo Proportional Control.
  Proportional Control Offset.
- **PI & PID: Level Control**
  PI Control of Level.
  Limitations of PI Control.
  PID Control of Level.
- **PI &PID: Flow Control**
  PI Control of Flow.
  PID Control of Flow.
- **Tuning PID Controllers**
  Zeigler-Nichols Tuning.
  Self-Tuning.
- **Process Controller: Advanced**
  Remote Set-Point.
  Profile Programming.
  Time Proportioned Output.

ESPIAL

The teaching content is provided by the Espial software including the underlying theory, relevant background and all instrumentation. Test instruments are initialised with settings suitable for the required measurements but can be changed. Displays and measurements may be printed or exported for inclusion in laboratory reports.



espial

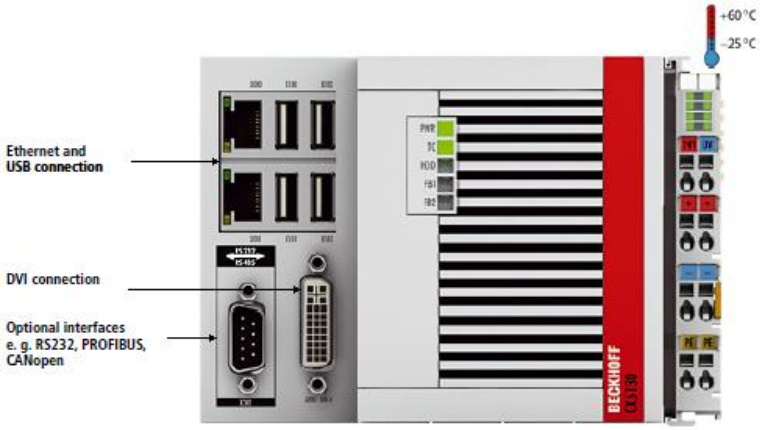
Feedback

Feedback Instruments
5 & 6 Warren Court
Park Road, Crowborough
East Sussex
TN6 2QX
United Kingdom
Tel: +44 1892 653322
Sales: sales@feedback-instruments.com
Website: www.feedback-instruments.com

For further information on Feedback equipment please contact ...

5.2 Anexo II

Hoja de datos iPC embebido CX5130



i CX5130 | Embedded PC with Intel Atom® processor

The CX5130 has an Intel Atom® multi-core processor with a clock rate of 1.75 GHz. This makes genuine multi-core technology possible in the Embedded PC segment. The hardware interfaces in this new series are oriented and implemented identically to those of the existing CX5000 series. Two independent, Gigabit-capable Ethernet interfaces as well as four USB 2.0 and a DVI-I interface are available. A multitude of further connection options and gateway functions is created by an option interface, which can be pre-equipped ex factory, as well as the I/O level, which can optionally consist of either E-bus or K-bus terminals.

The CX5130 is characterised by low power consumption and fanless design.

Depending on the installed TwinCAT runtime environment, the CX5130 can be used for implementing PLC or PLC/motion control projects with or without visualisation. The execution of motion control applications with interpolating axis movements is also possible.

Like the CX5000, the CX5100 series has a compact design; a modular device with extension modules like in the CX2000 series is not available.

Order identifier

The order identifier is derived as follows:

CX5130-01ST

- 0 = no TwinCAT
- 1 = with TwinCAT 2 PLC runtime
- 2 = with TwinCAT 2 NC PTP runtime
- 3 = with TwinCAT 2 NC I runtime
- 5 = TwinCAT 3 runtime (XAR)

- 0 = no operating system
- 1 = operating system Windows Embedded Compact 7
- 2 = operating system Windows Embedded Standard 7 P 32 bit
- 3 = operating system Windows Embedded Standard 7 P 64 bit
- 4 = Windows 10 IoT Enterprise 2016 LTSB 32 bit
- 5 = Windows 10 IoT Enterprise 2016 LTSB 64 bit
- 6 = Windows 10 IoT Enterprise 2019 LTSC 32 bit
- 7 = Windows 10 IoT Enterprise 2019 LTSC 64 bit

Further ordering options see price list.

Since not all combinations make sense, the table "Ordering information" contains a breakdown of the permissible combinations.

Optional interfaces:

- CX5130-N011 = DisplayPort interface
- CX5130-N020 = audio interface
- CX5130-N030 = RS232, D-sub plug
- CX5130-N031 = RS422/RS485, D-sub socket
- CX5130-M310 = PROFIBUS master, D-sub socket, 9-pin
- CX5130-B310 = PROFIBUS slave, D-sub socket, 9-pin
- CX5130-M510 = CANopen master, D-sub plug, 9-pin
- CX5130-B510 = CANopen slave, D-sub plug, 9-pin
- CX5130-M930 = PROFINET RT, controller
- CX5130-B930 = PROFINET RT, device, Ethernet (2 x RJ45 switch)
- CX5130-B931 = PROFINET IRT, device, Ethernet (2 x RJ45 switch)
- CX5130-B950 = EtherNet/IP slave, Ethernet (2 x RJ45 switch)
- CX5130-B110 = EtherCAT slave, EtherCAT IN and OUT (2 x RJ45)

Technical data	CX5130
Processor	Intel Atom® E3827, 1.75 GHz
Number of cores	2
Flash memory	slot for CFast card and microSD card, cards not included
Main memory	4 GB DDR3 RAM (not expandable)
Persistent memory	integrated 1-second UPS (1 MB on CFast card)
Interfaces	2 x RJ45 10/100/1000 Mbit/s, 1 x DVI-I, 4 x USB 2.0, 1 x optional interface
Diagnostics LED	1 x power, 1 x TC status, 1 x flash access, 2 x bus status
Clock	internal battery-backed clock for time and date (battery exchangeable)
Operating system	Microsoft Windows Embedded Compact 7 (supports only one CPU core), Microsoft Windows Embedded Standard 7 P, Microsoft Windows 10 IoT Enterprise 2016 LTSB, Microsoft Windows 10 IoT Enterprise 2019 LTSC
Control software	TwinCAT 2 runtime TwinCAT 3 runtime (XAR)
I/O connection	E-bus or K-bus, automatic recognition
Power supply	24 V DC (-15 %/+20 %)
Current supply E-bus/K-bus	2 A
Max. power consumption	14 W
Max. power consumption (with loading UPS)	20 W
Dimensions (W x H x D)	142 mm x 100 mm x 92 mm
Weight	approx. 960 g
Operating/storage temperature	-25…+60 °C/-40…+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protection class	IP 20
Approvals/markings	CE, UL, ATEX, IECEx
Ex marking	II 3 G Ex nA IIC T4 Gc II 3 D Ex tc IIIC T135 °C Dc Ex nA IIC T4 Gc Ex tc IIIC T135 °C Dc
TC3 performance class	Performance (40); please see here for an overview of all the TwinCAT 3 performance classes

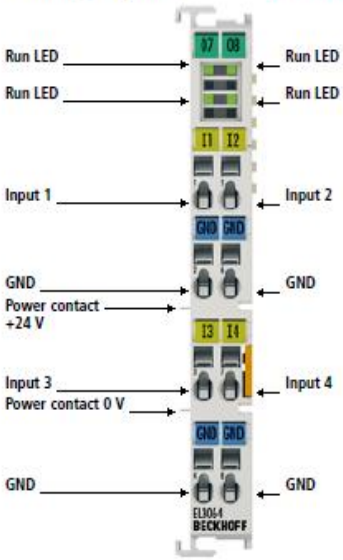
Ordering information	no operating system	Windows Embedded Compact 7	Windows Embedded Standard 7 P 32 bit	Windows Embedded Standard 7 P 64 bit	Windows 10 IoT Enterprise 2016 LTSB 32 bit	Windows 10 IoT Enterprise 2016 LTSB 64 bit	Windows 10 IoT Enterprise 2019 LTSC 32 bit	Windows 10 IoT Enterprise 2019 LTSC 64 bit	no TwinCAT	TwinCAT 2 PLC runtime
CX5130-0100	x	–	–	–	–	–	–	–	x	–
CX5130-0110	–	x	–	–	–	–	–	–	x	–
CX5130-0111	–	x	–	–	–	–	–	–	–	x
CX5130-0112	–	x	–	–	–	–	–	–	–	–
CX5130-0113	–	x	–	–	–	–	–	–	–	–
CX5130-0115	–	x	–	–	–	–	–	–	–	–
CX5130-0120	–	–	x	–	–	–	–	–	x	–
CX5130-0121	–	–	x	–	–	–	–	–	–	x
CX5130-0122	–	–	x	–	–	–	–	–	–	–
CX5130-0123	–	–	x	–	–	–	–	–	–	–
CX5130-0125	–	–	x	–	–	–	–	–	–	–
CX5130-0130	–	–	–	x	–	–	–	–	x	–
CX5130-0135	–	–	–	x	–	–	–	–	–	–
CX5130-0140	–	–	–	–	x	–	–	–	x	–
CX5130-0141	–	–	–	–	x	–	–	–	–	x
CX5130-0142	–	–	–	–	x	–	–	–	–	–
CX5130-0143	–	–	–	–	x	–	–	–	–	–
CX5130-0150	–	–	–	–	–	x	–	–	x	–
CX5130-0155	–	–	–	–	–	x	–	–	–	–
CX5130-0160	–	–	–	–	–	–	x	–	x	–
CX5130-0161	–	–	–	–	–	–	x	–	–	x
CX5130-0162	–	–	–	–	–	–	x	–	–	–
CX5130-0163	–	–	–	–	–	–	x	–	–	–
CX5130-0170	–	–	–	–	–	–	–	x	x	–
CX5130-0175	–	–	–	–	–	–	–	x	–	–

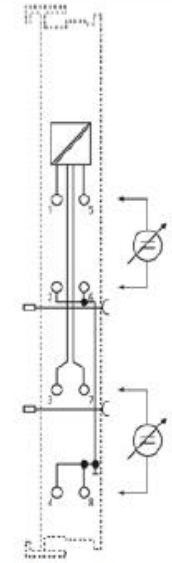
Accessories	
CX1900-0101	DVI-to-VGA passive adaptor for connecting a standard desktop VGA monitor to the CX system (singles out the VGA signals of the DVI-I interface).
CX2900-0026	20 GB CFast card, 3D flash, extended temperature range (only for Windows CE or TwinCAT/BSD operating systems)
CX2900-0038	40 GB CFast card, 3D flash, extended temperature range
CX2900-0040	80 GB CFast card, 3D flash, extended temperature range
CX2900-0042	160 GB CFast card, 3D flash, extended temperature range
CX2900-0107	Device modification for CX5120, CX5130, CX5140 and CX9020 Embedded PCs according to the requirements for ATEX and IECEx certification. The modification is mandatory for the usage of CX5120, CX5130, CX5140 and CX9020 in hazardous areas, Zone 2/22. It includes the modification and repositioning of the device label as well as a mounting bracket installed ex works for mechanical locking of the connectors. Product labeling: ATEX: II 3 G Ex nA IIC T4 Gc and II 3 D Ex tc IIIC T135 °C Dc IECEx: Ex nA IIC T4 Gc and Ex tc IIIC T135 °C Dc Read the device documentation for use in hazardous areas carefully.

Optional Interfaces	
CX5130-N010	DVI-D interface, additional DVI-D port for clone or extended desktop operation
CX5130-N011	DisplayPort interface, additional DisplayPort for clone or extended desktop operation
CX5130-N020	audio interface, 3 x 3.5 mm jack sockets, Line In, Mic In, Line Out or 5.1 Surround
CX5130-N030	RS232 interface, D-sub plug, 9-pin
CX5130-N031	RS485 interface, D-sub socket, 9-pin, configuration as an end point, without echo, termination on
CX5130-N031-0001	RS485 interface, D-sub socket, 9-pin, configuration as an end point, with echo, termination on
CX5130-N031-0002	RS485 interface, D-sub socket, 9-pin, configuration as drop point, without echo, termination off
CX5130-N031-0003	RS485 interface, D-sub socket, 9-pin, configuration as drop point, with echo, termination off
CX5130-N031-0004	RS422 interface, D-sub socket, 9-pin, configuration as full duplex end point, termination on
CX5130-B110	EtherCAT slave interface, EtherCAT IN and OUT (2 x RJ45)
CX5130-M310	PROFIBUS master interface, D-sub socket, 9-pin
CX5130-B310	PROFIBUS slave interface, D-sub socket, 9-pin
CX5130-M510	CANopen master interface, D-sub plug, 9-pin
CX5130-B510	CANopen slave interface, D-sub plug, 9-pin
CX5130-M930	PROFINET RT, controller interface, Ethernet (2 x RJ45)
CX5130-B930	PROFINET RT, device interface, Ethernet (2 x RJ45 switched)
CX5130-B950	EtherNet/IP slave interface, Ethernet (2 x RJ45 switched)

i Product announcement	CX5xxx-B950: estimated market release on request

Analog input EL3064



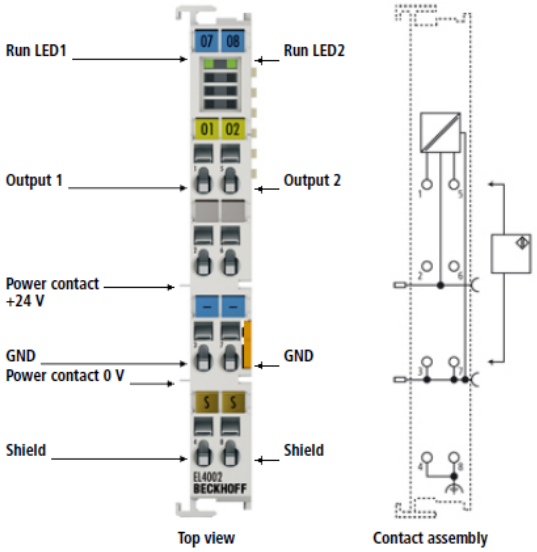


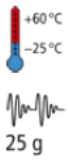
EL3064 | 4-channel analog input terminal 0…10 V, single-ended, 12 bit

The EL3064 analog input terminal processes signals in the range between 0 and 10 V. The voltage is digitised with a resolution of 12 bits and is transmitted (electrically isolated) to the higher-level automation device. The EL3064 EtherCAT Terminal features 2-wire conductors for the four single-ended inputs with a common internal ground potential. The power contacts are connected through. The signal state of the EtherCAT Terminal is indicated by light emitting diodes.

Technical data	EL3064 | ES3064
Number of inputs	4 (single-ended)
Power supply	via the E-bus
Technology	single-ended
Signal voltage	0…10 V
Distributed clocks	–
Internal resistance	> 130 kΩ
Input filter limit frequency	1 kHz
Dielectric strength	max. 30 V
Conversion time	0.625 ms default setting, configurable, multiplex
Resolution	12 bit (16 bit presentation incl. sign)
Measuring error	< ±0.3 % (relative to full scale value)
Electrical isolation	500 V (E-bus/signal voltage)
Current consumption power contacts	–
Current consumption E-bus	typ. 130 mA
Bit width in the process image	inputs: 16 byte
Special features	activatable FIR/IIR filters, limit value monitoring
Weight	approx. 60 g
Operating/storage temperature	-25…+60 °C/-40…+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/variable
Pluggable wiring	for all ESxxxx terminals
Approvals/markings	CE, UL, ATEX
Ex-Marking	II 3 G Ex nA IIC T4 Gc

Analog output EL4002





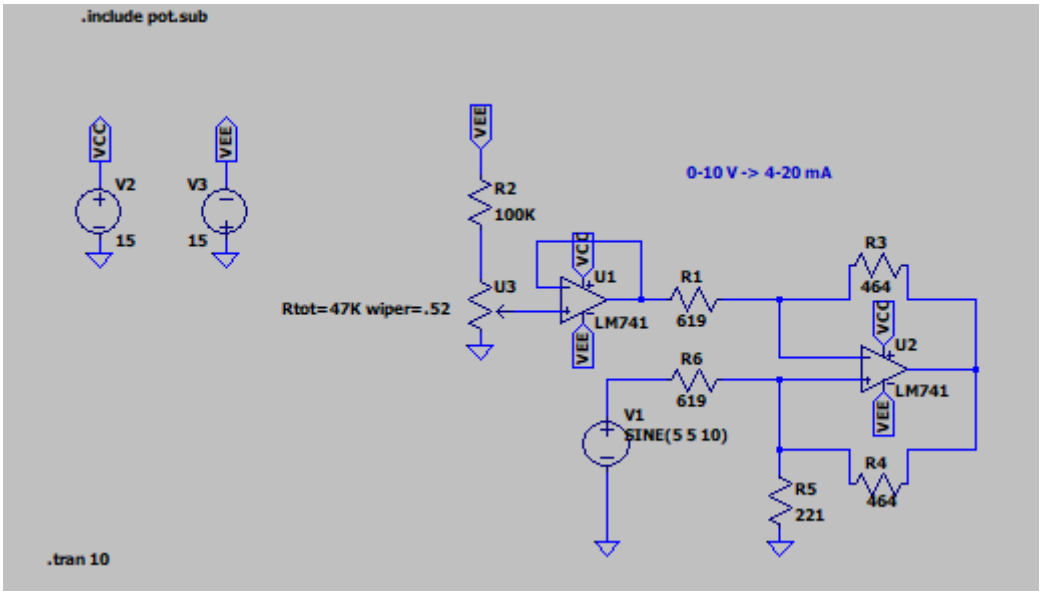
EL4002 | 2-channel analog output terminal 0…10 V, 12 bit

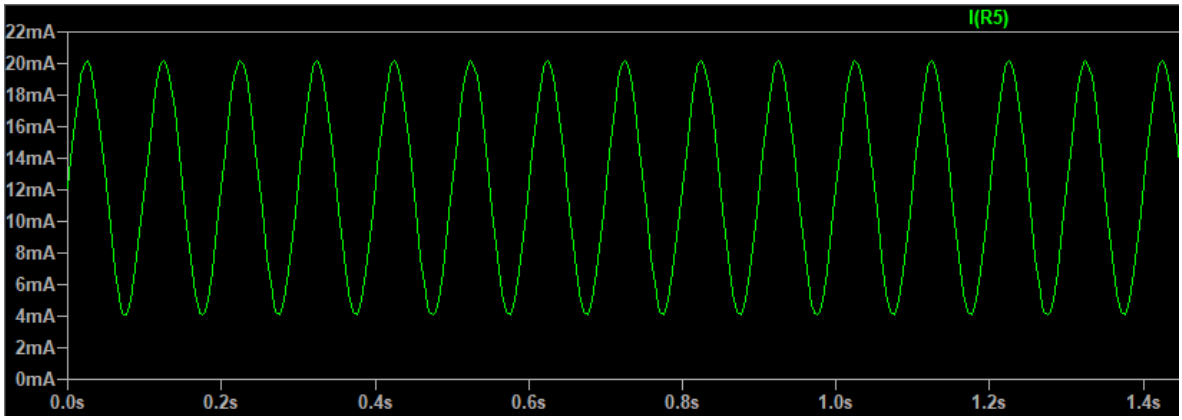
The EL4002 analog output terminal generates signals in the range between 0 and 10 V. The voltage is supplied to the process level with a resolution of 12 bits and is electrically isolated. The output channels of the EtherCAT Terminal have a common ground potential. The EL4002 combines two channels in one housing. The output stages are powered by the 24 V supply. The signal state of the EtherCAT Terminal is indicated by light emitting diodes.

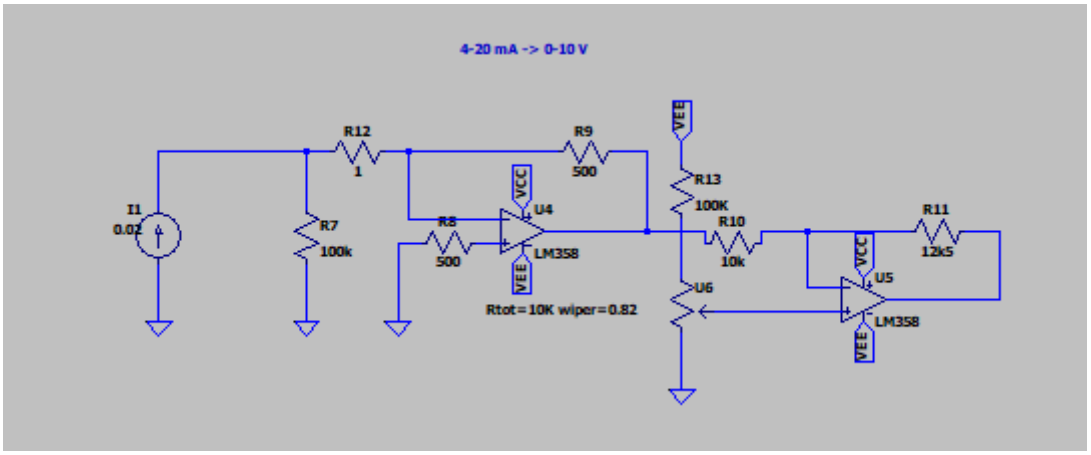
Technical data	EL4002 | ES4002
Connection technology	2-wire, single-ended
Number of outputs	2
Power supply	24 V DC via power contacts
Signal voltage	0…10 V
Distributed clocks	yes
Distributed clock precision	<< 1 µs
Load	> 5 kΩ (short-circuit-proof)
Output error	< 0.1 % (relative to end value)
Resolution	12 bit
Electrical isolation	500 V (E-bus/signal voltage)
Conversion time	~ 150 µs
Current consumption power contacts	typ. 25 mA
Current consumption E-bus	typ. 140 mA
Bit width in the process image	2 x 16 bit AO output
Special features	Optional watchdog: user-specific output value with ramp; user synchronisation can be activated.
Weight	approx. 60 g
Operating/storage temperature	-25…+60 °C/-40…+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/variable
Pluggable wiring	for all ESxxxx terminals
Approvals	CE, UL, Ex

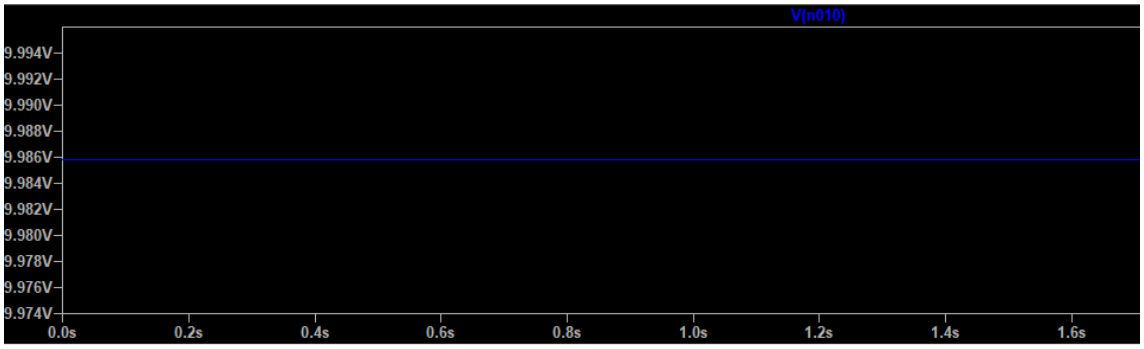
5.3 Anexo lll

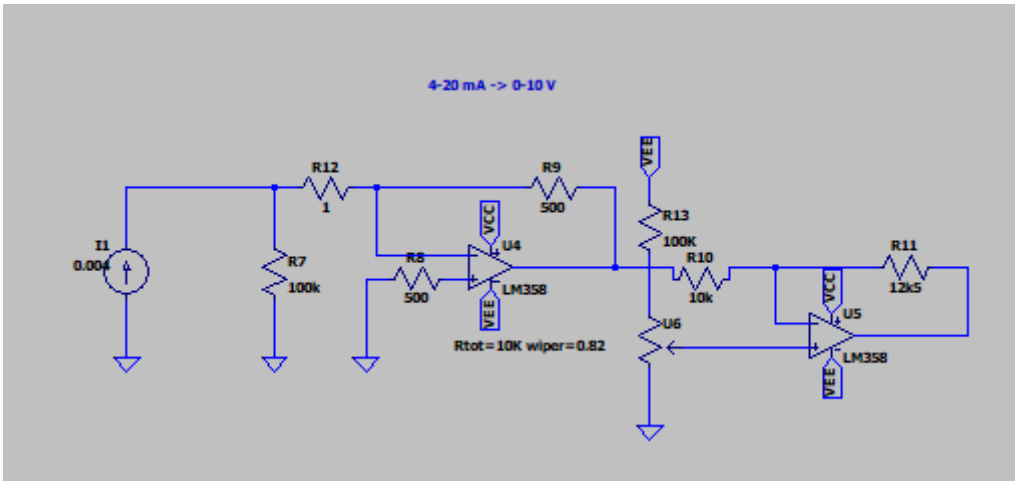
Circuitos de conversión en LTspice XVII

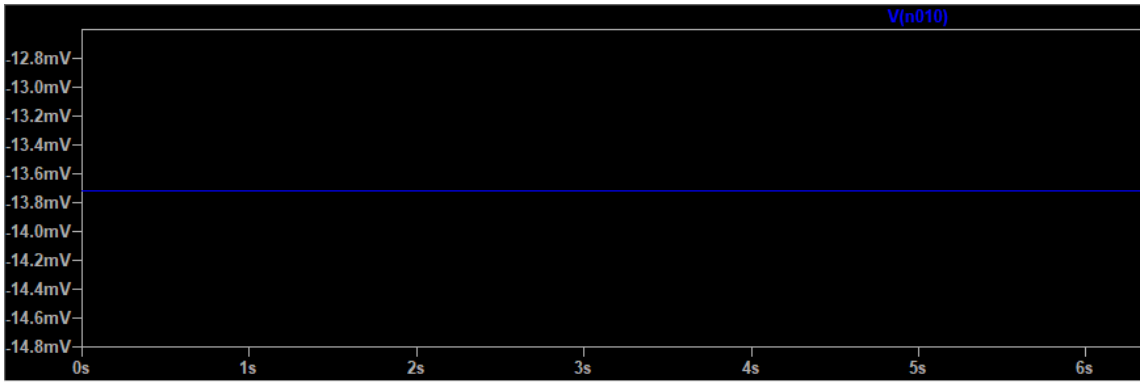

.include pot.sub
VCC
V2
15
VEE
V3
15
VEE
R2
100K
0-10 V -> 4-20 mA
Rtot=47K wiper=.52
U3
VCC
U1
LM741
VEE
R1
619
R3
464
VCC
U2
LM741
VEE
R6
619
V1
SINE(5 5 10)
R5
221
R4
464
.tran 10


I(R5)
22mA
20mA
18mA
16mA
14mA
12mA
10mA
8mA
6mA
4mA
2mA
0mA
0.0s
0.2s
0.4s
0.6s
0.8s
1.0s
1.2s
1.4s


4-20 mA -> 0-10 V
I1
0.02
R12
1
R7
100k
R8
500
R9
500
U4
LM358
VCC
VEE
Rtot=10K wiper=0.82
R13
100K
R10
10k
U6
U5
LM358
R11
12k5


V(n010)
9.994V
9.992V
9.990V
9.988V
9.986V
9.984V
9.982V
9.980V
9.978V
9.976V
9.974V
0.0s
0.2s
0.4s
0.6s
0.8s
1.0s
1.2s
1.4s
1.6s


4-20 mA -> 0-10 V
I1
0.004
R12
1
R7
100k
R8
500
R9
500
U4
LM358
VCC
VEE
Rtot=10K wiper=0.82
R13
100K
R10
10k
U6
U5
LM358
R11
12k5


V(n010)
-12.8mV
-13.0mV
-13.2mV
-13.4mV
-13.6mV
-13.8mV
-14.0mV
-14.2mV
-14.4mV
-14.6mV
-14.8mV
0s
1s
2s
3s
4s
5s
6s

Escuela Superior de Ingeniería y Tecnología

Sección Ingeniería Industrial

Trabajo Fin de Grado

Grado en Ingeniería Industrial y Automática

Diseño e implementación de controladores Beckhoff en maquetas de procesos de control

6. BIBLIOGRAFÍA

Estudiante: Jesús Elías Soto

Tutor: Santiago Torrez Álvarez

Cotutor: Juan Albino Méndez Pérez

Convocatoria: Junio 2022

[1] - “History of the PLC”. Accesible a través de: https://library.automationdirect.com/history-of-the-plc/. Accedido por última vez: 04/06/2022

[2] - Bolton, William (2015). “Programmable Logic Controllers” (6th, revised ed.). Newnes. ISBN 9780081003534.

[3] - Parr, E. A. (1998). "Computers and industrial control". Industrial Control Handbook. Industrial Press Inc. ISBN 0-8311-3085-7.

[4] - Laughton, M. A.; Warne, D. F. (2002). “Electrical Engineer's Reference Book” (16th ed.). Newnes. ISBN 9780750646376.

[5] - “A brief History of the SCADA system”. Accesible a través de: https://www.theearthawards.org/a-brief-history-of-the-scada-system/. Accedido por última vez: 04/06/2022

[6] - Norma UNE-EN 157001:2014 " Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico". AENOR.

[7] - “Structured Text programming” Accesible a través de: https://www.plcacademy.com/structured-text-tutorial/#what-is. Accedido por última vez: 04/06/2022

[8] - "El trabajo de fin de grado: guía para estudiantes, docentes y agentes colaboradores" Virginia Ferrer, Moisés Carmona. ISBN: 978-84-481-8267-0. McGraw Hill, 2012.

[9] - "Guía práctica para la realización de trabajos fin de grado y trabajos fin de máster". Mari Paz García Sanz, Pilar Martínez Clares. ISBN: 9788483719732. Universidad de Murcia, 2012

[10] - “IEC 61131-3:2013” Accesible a través de: https://webstore.iec.ch/publication/4552. Accedido por última vez: 04/06/2022.

[11] - Boyer, Stuart A. (2010). “SCADA Supervisory Control and Data Acquisition”. USA: ISA - International Society of Automation. p. 179. ISBN 978-1-936007-09-7.

[12] - “Introduction to Industrial Control Networks”. Accesible a través de: http://www.rfidblog.org.uk/Preprint-GallowayHancke-IndustrialControlSurvey.pdf. Accedido por última vez: 04/06/2022.

[13] - “Scada systems”. Accesible a través de: https://www.engineersgarage.com/scada-systems/. Accedido por última vez: 04/06/2022

[14] - “IEC 61850 – Communication Networks and Systems in Substations: An Overview of Computer Science” Accesible a través de: https://seclab.illinois.edu/wp-content/uploads/2011/03/iec61850-intro.pdf. Accedido por última vez: 04/06/2022

[15] - “IEC 61850:2022 SER Series”. Accesible a través de: https://webstore.iec.ch/publication/6028. Accedido por última vez: 04/06/2022

[16] - Gordon R. Clarke et al, “Practical modern SCADA protocols: DNP3, 60870.5 and related systems”, Newnes, 2004 ISBN 0-7506-5799-5

[17] - “IEC 60870-5:2022 SER Series”. Accesible a través de: https://webstore.iec.ch/publication/3755. Accedido por última vez: 04/06/2022

[18] - “Estructura del TFG: Qué partes comprende y qué consejos debes tener en cuenta”. Accesible a través de: https://tfg.es/estructura-tfg/. Accedido por última vez: 04/06/2022

[19] - Página principal de Beckhoff España. Accesible a través de: https://www.beckhoff.com/es-es/. Accedido por última vez: 04/06/2022

[20] - “Programmable Logic Controller”. Accesible a través de: https://www.sciencedirect.com/topics/computer-science/programmable-logic-controller. Accedido por última vez: 04/06/2022

[21] - Tanenbaum, Andrew (2008). “Modern Operating Systems”. Upper Saddle River, NJ: Pearson/Prentice Hall. p. 160. ISBN 978-0-13-600663-3.

[22] - “What is the definition of “PLC””. Accesible a través de: https://www.unitronicsplc.com/what-is-plc-programmable-logic-controller/. Accedido por última vez: 04/06/2022.

[23] - “The difference between an Industrial PC and a PLC”. Accesible a través de: https://www.cybernetman.com/blog/the-difference-between-an-industrial-pc-and-a-plc/. Accedido por última vez: 04/06/2022.

[24] - “CX5130 | Embedded PC with Intel Atom® processor”. Accesible a través de: https://www.beckhoff.com/en-gb/products/ipc/embedded-pcs/cx5100-intel-atom/cx5130.html. Accedido por última vez: 04/06/2022

[25] - "Embedded PCs – DIN rail-mountable Industrial PCs". Accesible a través de: https://www.beckhoff.com/es-es/products/ipc/embedded-pcs/. Accedido por última vez: 04/06/2022

[26] - "Embedded systems – Overview ". Accesible a través de: https://www.tutorialspoint.com/embedded_systems/es_overview.htm. Accedido por última vez: 04/06/2022

[27] - "Embedded virtualization for the design of secure IoT applications". Accesible a través de: https://ieeexplore.ieee.org/document/7909116. Accedido por última vez: 04/06/2022

[28] - "IEC 60715:2017". Accesible a través de: https://webstore.iec.ch/publication/29026. Accedido por última vez: 04/06/2022

[29] - "How to transform your field-level machine data into management-level information". Accesible a través de: https://www.ixon.cloud/knowledge-hub/how-to-transform-your-field-level-machine-data-into-management-level-information. Accedido por última vez: 04/06/2022

[30] - "IT explained: SCADA". Accesible a través de: https://www.paessler.com/it-explained/scada. Accedido por última vez: 04/06/2022

[31] - "Naming concept". Accesible a través de: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_overview/180212107.html&id=. Accedido por última vez: 04/06/2022

[32] - "HTML Living Standard". Accesible a través de: https://html.spec.whatwg.org/#html-vs-xhtml. Accedido por última vez: 04/06/2022

[33] - "HTML5 Tag Reference". Accesible a través de: https://web.archive.org/web/20120802010646/http://www.w3schools.com/html5/html5_reference.asp. Accedido por última vez: 04/06/2022

[34] - "PLC communication protocols and Its Types". Accesible a través de: https://microdigisoft.com/plc-communication-protocols-and-its-types/. Accedido por última vez: 05/06/2022

[35] - "Guía de comunicación Modbus RTU". Accesible a través de: https://www.virtual-serial-port.org/es/articles/modbus-rtu-guide/. Accedido por última vez: 05/06/2022

[36] - “Protocolo Modbus”. Accesible a través de: http://automation-networks.es/glossary/modbus-tcpip. Accedido por última vez: 05/06/2022

[37] - “PROFIBUS: Qué es y cómo funciona”. Accesible a través de: https://www.cursosaula21.com/que-es-profibus/. Accedido por última vez: 05/06/2022

[38] - “PROFINET: Qúe es y cómo funciona”. Accesible a través de: https://www.cursosaula21.com/profinet-que-es-y-como-funciona/. Accedido por última vez: 05/06/2022

[39] - “EtherCAT – the Ethernet fieldbus”. Accesible a través de: https://www.ethercat.org/pdf/ethercat_e.pdf. Accedido por última vez: 05/06/2022

[40] - “EtherCAT and TwinCAT – the next generation control”. Accesible a través de: https://www.ethercat.org/download/documents/pcc0205_ethercat_twincat_e.pdf . Accedido por última vez: 05/06/2022

[41] - “EtherCAT – the Ethernet Fieldbus”. Accesible a través de: https://www.ethercat.org/en/technology.html. Accedido por última vez: 05/06/2022

[42] - “PLC programming using TwinCAT 3 – IO (Part 10/18)”. Accesible a través de: https://www.youtube.com/watch?v=zQwiBeDbDcM. Accedido por última vez: 05/06/2022

[43] - “ADS-Communication”. Accesible a través de: https://infosys.beckhoff.com/english.php?content=../content/1033/cx8190_hw/5091854987.html&id=. Accedido por última vez: 05/06/2022

[44] - “ADS device concept”. Accesible a través de: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_ads_intro/116158859.html&id=. Accedido por última vez: 05/06/2022

[45] - “TwinCAT automation software”. Accesible a través de: https://www.beckhoff.com/es-es/products/automation/twincat/. Accedido por última vez: 05/06/2022

[46] - “Back to Basics: The Fundamentals of 4-20 mA Current Loops”. Accesible a través de: https://www.predig.com/indicatorpage/back-basics-fundamentals-4-20-ma-current-loops. Accedido por última vez: 05/06/2022

[47] - "EN IEC 61000-6-4: Emission standard for industrial environments". Accesible a través de: https://www.dlsemc.com/iec-en-61000-6-4-emission-standard-for-industrial-environments/. Accedido por última vez: 05/06/2022

[48] - "Operational Amplifiers Basics, Types and Advantages". Accesible a través de: https://www.monolithicpower.com/en/operational-amplifiers. Accedido por última vez: 05/06/2022

[49] - "Amplificador operacional ¿Qué es y cómo funciona?". Accesible a través de: https://www.centroestudioscervantinos.es/amplificador-operacional/. Accedido por última vez: 05/06/2022

[50] - "PID Control". Accesible a través de: https://www.eolss.net/ebooks/Sample%20Chapters/C18/E6-43-03-03.pdf. Accedido por última vez: 05/06/2022

[51] - Ang, K.H. and Chong, G.C.Y. and Li, Y. (2005) "PID control system analysis, design, and technology". IEEE Transactions on Control Systems Technology 13(4): pp. 559-576. Accesible a través de: http://eprints.gla.ac.uk/3817/1/IEEE3.pdf. Accedido por última vez: 05/06/2022

[52] - "Introduction: PID Controller Design". Accesible a través de: https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID. Accedido por última vez: 05/06/2022

[53] - "¿Qué es el control PID?". Accesible a través de: https://la.mathworks.com/discovery/pid-control.html. Accedido por última vez: 05/06/2022

[54] - "PID Controller Explained". Accesible a través de: https://realpars.com/pid-controller/. Accedido por última vez: 05/06/2022

[55] - "How to Tune a PID Controller". Accesible a través de: https://pidexplained.com/how-to-tune-a-pid-controller/. Accedido por última vez: 05/06/2022

[56] - "Should I limit programming to ladder logic or use all standards within IEC 61131?". Accesible a través de: https://www.controldesign.com/articles/2018/should-i-limit-programming-to-ladder-logic-or-use-all-standards-within-iec-61131/. Accedido a través de: 05/06/2022

[57] - "Structured Text Programming: A Step by Step Guide (With Examples)". Accesible a través de: https://www.plcacademy.com/structured-text-tutorial/#what-is. Accedido por última vez: 05/06/2022

[58] - "Structured Text (ST)". Accesible a través de: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/2526757515.html&id=. Accedido por última vez: 05/06/2022

[59] - "FB_CTRL_PID". Accesible a través de: https://infosys.beckhoff.com/english.php?content=../content/1033/tf4100_tc3_controller_toolbox/245435787.html&id=. Accedido por última vez: 05/06/2022

[60] - "FB_CTRL_LOG_DATA". Accesible a través de: https://infosys.beckhoff.com/english.php?content=../content/1033/tf4100_tc3_controller_toolbox/245475339.html&id=. Accedido por última vez: 05/06/2022

[61] - "TwinCAT 3 | PLC Library Tc3_EventLogger". Accesible a través de: http://ftp.beckhoff.com/download/Document/automation/twincat3/TwinCAT_3_PLC_Lib_Tc3_EventLogger_EN.pdf. Accedido por última vez: 05/06/2022

[62] - "File access from the PLC". Accesible a través de: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_system/710612235.html&id=. Accedido por última vez: 05/06/2022

[63] - "T_AmsPort". Accesible a través de: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_system/31064331.html&id=. Accedido por última vez: 05/06/2022

[64] - "Cómo elaborar un presupuesto de construcción. Paso a paso". Accesible a través de: https://www.certicalia.com/blog/como-elaborar-presupuesto-construccion. Accedido por última vez: 05/06/2022

[65] - PLC programming using TwinCAT 3 - Basics & installation (Part 2/18). Accesible a través de: https://youtu.be/0iDn9E0V1iw. Accedido por última vez: 12/06/2022.