Trabajo Fin de Grado
en Ingeniería Electrónica, Robótica y Mecatrónica

Control de electrolizador mediante IEC 61499

Autor: Samuel López Castro

Tutor: Juan Manuel Escaño González

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2025

Trabajo Fin de Grado
en Ingeniería Electrónica, Robótica y Mecatrónica

Control de electrolizador mediante IEC 61499

Autor:
Samuel López Castro

Tutor:
Juan Manuel Escaño González
Profesor titular

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2025

Autor: Samuel López Castro

Tutor: Juan Manuel Escaño González

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2025

El Secretario del Tribunal

A mi familia

A mis compañeros

A mis amigos

Agradecimientos

Quiero expresar mi más sincera gratitud a todas las personas que han formado parte de este camino y que, de una forma u otra, han hecho posible la realización de este proyecto.

En primer lugar, a mi familia, quienes han estado ahí siempre que lo he necesitado. En especial a mis padres quienes siempre me han apoyado, proporcionado todos los medios necesarios, madrugado cuando no tenían por qué hacerlo, animado y aconsejado; y sobre todo han sido un grandísimo ejemplo a seguir, enseñándome el valor del trabajo y la perseverancia. También a mi hermano por cuidar de mí y por recordarme que en incluso en momentos difíciles siempre hay espacio para una sonrisa (o una partida de Brawl Stars).

A mi pareja Nuria, por su paciencia y comprensión, por sus ánimos incansables, por celebrar cada pequeño logro y por levantarme en los días más duros.

A mis compañeros de carrera; sobre todo a Pedro, Dani, Mario y Pablo, por acompañarme paso a paso, compartiendo largas jornadas de estudio, proyectos, risas, nervios antes de los exámenes, felicidad por superar retos juntos… (y sobre todo los apuntes).

A mis amigos que conozco desde pequeño, por ser un pilar fundamental en esta etapa, sacándome sonrisas en momentos difíciles y compartiendo experiencias inolvidables juntos.

A los profesores de la escuela, por su excelente labor y por ayudarme a adquirir los conocimientos que han hecho posible la realización este proyecto.

Y, por último, a mi tutor Juan Manuel, por su guía y dedicación para orientarme, resolver mis dudas y enriquecer este trabajo con su experiencia y conocimientos.

Samuel López Castro

Sevilla, 2025

Resumen

Este proyecto se centra en el diseño de un sistema de control para un electrolizador PEM utilizando la norma IEC 61499, con el objetivo de optimizar su eficiencia energética, alargar su vida útil y asegurar un funcionamiento estable. La motivación surge de la necesidad de avanzar en tecnologías limpias como el hidrógeno verde, cuya producción eficiente requiere sistemas de automatización avanzados capaces de integrarse con fuentes renovables intermitentes.

Para ello, se emplea un modelo simulado del electrolizador en Simulink y la plataforma EcoStruxure Automation Expert, conectados mediante el protocolo OPC UA para garantizar la interoperabilidad. A través de este enfoque, se busca validar la viabilidad de aplicar el estándar IEC 61499, el cual aún no ha sido ampliamente aplicado hasta la fecha, en entornos industriales de hidrógeno verde, contribuyendo al desarrollo de soluciones industriales sostenibles y escalables en el ámbito de la automatización distribuida.

Abstract

This project focuses on the design of a control system for a PEM electrolyzer using the IEC 61499 standard, with the aim of optimizing energy efficiency, extending lifespan, and ensuring stable operation. The motivation stems from the need to advance clean technologies such as green hydrogen, whose efficient production requires advanced automation systems capable of integrating with intermittent renewable sources.

To this end, a simulated model of the electrolyzer is used in Simulink and the EcoStruxure Automation Expert platform, connected via the OPC UA protocol to ensure interoperability. This approach seeks to validate the feasibility of applying the IEC 61499 standard, which has not yet been widely studied, in green hydrogen industrial environments, contributing to the development of more sustainable and scalable solutions in the field of distributed automation.

Índice

Agradecimientos **ix**

Resumen **xi**

Abstract **xiii**

Índice **xiv**

Índice de Tablas **xvi**

Índice de Códigos **xviii**

Índice de Ecuaciones **xx**

Índice de Figuras **xxii**

1 Introducción **1**
1.1 Objetivos y metodología del trabajo *1*

2 Fundamentos teóricos de la electrólisis **3**
2.1 Hidrógeno *3*
2.2 Electrolizadores. Tipos de electrolizadores *6*
2.2.1 Alcalinos (AEL) 7
2.2.2 PEM (Proton Exchange Membrane) 8
2.2.3 SOEC (Solid Oxide Electrolyzer Cell) 9
2.2.4 AEM (Anion Exchange Membrane) 10
2.3 Aplicaciones del hidrógeno verde *11*
2.4 Control de los procesos del electrolizador *11*

3 Norma IEC 61499 **13**
3.1 Introducción a la norma IEC 61499 *13*
3.2 Comparación con la norma IEC 61131-3 *16*
3.3 Modelo basado en bloques funcionales. Tipos de FB *17*
3.3.1 Basic Function Block (BFB) 17
3.3.2 Composite Function Block (CFB) 18
3.3.3 Service Function Block (SIFB) 18
3.4 Herramientas de desarrollo compatibles: EcoStruxure Automation Expert *19*
3.4.1 Comunicación OPC UA 20
3.4.2 CAT e instancias CAT 23

4 Características del sistema **25**
4.1 Especificaciones técnicas *26*
4.2 Modelo electroquímico *26*
4.2.1 Cálculo de la tensión de circuito abierto 28
4.2.2 Cálculo del sobrepotencial de actuación 29

4.2.3 Cálculo del sobrepotencial óhmico 30
4.3 Modelo térmico *30*
4.3.1 Cálculo del calor generado 32
4.3.2 Cálculo de las pérdidas térmicas en el ambiente 32
4.3.3 Cálculo del calor extraído por el sistema de refrigeración (*Qcool*) 33

5 Desarrollo e implementación **35**
5.1 Prueba de conexión OPC UA con Matlab *35*
5.2 Control de la temperatura desde Matlab *42*
5.2.1 Cálculo de la función de transferencia 42
5.2.2 Diseño del controlador 44
5.3 Control de la temperatura desde EAE *50*
5.3.1 Configuración en Matlab 50
5.3.1.1 Sample time 50
5.3.1.2 Funciones de lectura y escritura 51
5.3.2 Diseño del PI desde EAE 54
5.3.2.1 Bloque IThis 56
5.3.2.2 Bloque RENDEZVOUZ 57
5.3.2.3 Bloque ERROR 57
5.3.2.4 Bloque INTEGRAL 59
5.3.2.5 Bloque PI 60
5.3.2.6 Bloque FILTRO 62
5.3.3 Realización de simulaciones 63

6 Resultados y análisis **65**

7 Conclusiones y trabajos futuros **69**

Referencias **71**

Glosario **74**

ÍNDICE DE TABLAS

Tabla 1. Comparación entre los colores del hidrógeno en términos de costs y emisiones de carbono 5
Tabla 2. Eventos predeterminados IEC 61499 14
Tabla 3. Especificaciones técnicas del electrolizador 26

ÍNDICE DE CÓDIGOS

Código 1. Algoritmo del bloque 'Producto' 37
Código 2. Script de Matlab para comunicación OPC UA [25] 39
Código 3. Script de Matlab 'DibujaBodeGba.m' 45
Código 4. Script de Matlab 'conexion.m' 51
Código 5. Función de Matlab 'writeRef.m' 52
Código 6. Función de Matlab 'writeTemp.m' 52
Código 7. Función de Matlab 'readQcool.m' 53
Código 8. Algoritmo 'Resta' del bloque 'Error' 59
Código 9. Algoritmo 'Calculo' del bloque 'Integral' 60
Código 10. Algoritmo `Pid' del bloque 'PI' 62
Código 11. Algoritmo 'Calculo' del bloque 'Filtro_tau' 63

ÍNDICE DE ECUACIONES

Ecuación 1. Relación estequiométrica de la electrólisis 3
Ecuación 2. Relación estequiométrica en el ánodo 7
Ecuación 3. Relación estequiométrica en el cátodo 7
Ecuación 4. Relación estequiométrica en el ánodo 8
Ecuación 5. Relación estequiométrica en el cátodo 8
Ecuación 6. Relación estequiométrica de la reacción resultante en PEM 8
Ecuación 7. Relación estequiométrica de oxidación en ánodo 9
Ecuación 8. Relación estequiométrica de oxidación en cátodo 9
Ecuación 9. Relación estequiométrica de la reacción resultante en SOEC 9
Ecuación 10. Cálculo del voltaje en el stack 27
Ecuación 11. Ecuación de Nernst 28
Ecuación 12. Cálculo del voltaje estándar 28
Ecuación 13. Cálculo del sobrepotencial de activación 29
Ecuación 14. Ley de Ohm 30
Ecuación 15. Cálculo de la resistencia óhmica 30
Ecuación 16. Cálculo de la conductividad de la membrana 30
Ecuación 17. Balance térmico en el electrolizador 31
Ecuación 18. Cálculo del calor generado 32
Ecuación 19. Cálculo del voltaje termoneutral 32
Ecuación 20. Cálculo del coeficiente adimensional 32
Ecuación 21. Cálculo de la presión parcial del vapor de agua 32
Ecuación 22. Cálculo del coeficiente *Y* 32
Ecuación 23. Cálculo de las pérdidas térmicas en el ambiente 33
Ecuación 24. Cálculo del calor extraído por el sistema de refrigeración 33
Ecuación 25. Ecuación de control PID 42
Ecuación 26. Expresión de la función de transferencia de sistema de primer orden 43
Ecuación 27. Cálculo de la ganancia estática 43
Ecuación 28. Cálculo de la constante de tiempo 43
Ecuación 29. Función de transferencia del modelo 44
Ecuación 30. Expresión de GBA(s) 44

Ecuación 31. Cálculo del tiempo de subida en bucle cerrado 45
Ecuación 32. Cálculo de la frecuencia de corte 45
Ecuación 33. Cálculo del margen de fase actual 46
Ecuación 34. Cálculo del aporte de fase 46
Ecuación 35. Expresión del controlador PI 46
Ecuación 36. Cálculo de la constante de tiempo del controlador 47
Ecuación 37. Cálculo del polo a alta frecuencia 47
Ecuación 38. Cálculo de la ganancia k 47
Ecuación 39. Expresión final del controlador PI 48
Ecuación 40. Expresión del prefiltro 48
Ecuación 41. Expresión de un PI 61
Ecuación 42. Cálculo de los parámetros del PI 61
Ecuación 43. Cálculo del filtro a alta frecuencia 63

ÍNDICE DE FIGURAS

Figura 1. Ilustración esquemática de un sistema básico de electrólisis de agua [10] 7
Figura 2. Celda de electrolizador alcalino [11] 8
Figura 3. Representación esquemática de los procesos de electrólisis PEM y de pila de combustible [11] 9
Figura 4. Celda de electrolizador SOEC [11] 10
Figura 5. Celda de electrolizador AEM [11] 10
Figura 6. Ejemplo de BFB en IEC 61499 13
Figura 7. Ejemplo de ECC de un FB en IEC 61499 14
Figura 8. Conexión en recursos 15
Figura 9. Conexiones y mapeado en programas IEC 61499 15
Figura 10. Etapas para la implementación en cada uno de los estándares 16
Figura 11. Comparación entre FBs de IEC 61131-3 vs. IEC 61499 17
Figura 12. BFB vs. CFB 18
Figura 13. Ecostruxure Automation Expert 19
Figura 14. Interfaz EAE [26] 20
Figura 15. Elementos de la comunicación OPC UA 21
Figura 16. Ejemplo de nodos y referencias de nodos 22
Figura 17. Ejemplo de comunicación Publicador-Suscriptor 22
Figura 18. Modelo Simulink del sistema 25
Figura 19. Modelo electroquímico y modelo térmico 26
Figura 20. Modelo electroquímico 27
Figura 21. Sub-modelo para el cálculo de *E* 28
Figura 22. Sub-modelo para el cálculo de *Vact* 29
Figura 23. Sub-modelo para el cálculo de *Vohm* 30
Figura 24. Modelo térmico 31
Figura 25. Propiedades del servidor OPC UA 35
Figura 26. Red de bloques del CAT ‘prueba’ 36
Figura 27. Datos y eventos del bloque ‘Producto’ 36
Figura 28. Datos y eventos del bloque ‘prueba_HMI’ 37
Figura 29. Red de bloques de la aplicación 37
Figura 30. Red de bloques del recurso 38

Figura 31. Implementación y diagnóstico 38
Figura 32. Lista de servidores encontrados por Matlab 40
Figura 33. Propiedades del cliente OPC UA (desconectado) 40
Figura 34. Propiedades del cliente OPC UA (conectado) 41
Figura 35. Resultado del producto 41
Figura 36. Resultado de la lógica de ejemplo EAE 42
Figura 37. Temperatura del electrolizador ante escalón a la entrada 43
Figura 38. Comparación del modelo con el sistema real 44
Figura 39. GBA(s) con C(s) = 1/1 46
Figura 40. GBA(s) con C(s) = (776.9194s+1)/(159.1549s+1)s 47
Figura 41. GBA(s) con C(s) = 0.0082(776.9194s+1)/(159.1549s+1)s 48
Figura 42. Simulación programada en Simulink 49
Figura 43. Resultados de la simulación en Matlab 49
Figura 44. Señal de habilitación de los bloques de lectura y escritura 50
Figura 45. Error por el uso de variables globales en las funciones 53
Figura 46. Subsystem ‘Escritura Ref’ 54
Figura 47. Diagrama de bloques de Simulink para la conexión con EAE 54
Figura 48. Red de bloques de la aplicación 55
Figura 49. Red de bloques del recurso 55
Figura 50. Red de bloques del CAT ‘drawwork’ 56
Figura 51. Configuración OPC UA de las variables compartidas en ‘controlDrawwork_HMI’ 56
Figura 52. Datos y eventos del bloque ‘controlDrawwork_HMI’ 57
Figura 53. Datos y eventos del bloque ‘E_REND’ 57
Figura 54. Datos y eventos del bloque ‘Error’ 58
Figura 55. Secuencia de estados del bloque ‘Error’ 58
Figura 56. Datos y eventos del bloque ‘Integral’ 59
Figura 57. Secuencia de estados del bloque ‘Integral’ 60
Figura 58. Datos y eventos del bloque ‘PI’ 61
Figura 59. Secuencia de estados del bloque ‘PI’ 61
Figura 60. Datos y eventos del bloque ‘Filtro_tau’ 62
Figura 61. Secuencia de estados del bloque ‘Filtro_tau’ 63
Figura 62. Resultados de la simulación 1 en EAE 65
Figura 63. Resultados de la simulación 2 en EAE 66
Figura 64. Resultados de la simulación 3 en EAE 66
Figura 65. Resultados de la simulación 4 en EAE 67
Figura 66. Valores de la perturbación *I_el* en la simulación 5 67
Figura 67. Resultados de la simulación 5 en EAE 68

1 INTRODUCCIÓN

En el ámbito de la automatización industrial, la creciente complejidad de los sistemas distribuidos plantea importantes desafíos en términos de diseño y gestión. Como señala Vyatkin (2009), "el diseño de sistemas de automatización distribuida se complica debido a la falta de un único lenguaje de programación que permita al desarrollador describir el sistema completo, incluyendo la lógica de cada nodo de control y sus interacciones" (p.41). A medida que los sistemas de automatización industrial se vuelven más complejos y distribuidos, este problema se va intensificando; es por ello que surge la norma IEC 61499. Esta arquitectura propone un lenguaje de diseño a nivel de sistema para sistemas distribuidos de medición y control, acortando así la distancia entre los lenguajes de programación PLC más populares y los sistemas distribuidos [1].

La norma IEC 61499 es un estándar internacional desarrollado por el Comité Técnico 65 de la IEC (International Electrotechnical Commission) a partir de un proyecto iniciado en 1990. Su objetivo fue definir un modelo arquitectónico común basado en bloques de función (function blocks) para sistemas distribuidos de medición y control de procesos industriales. Este estándar se diseñó para complementar tanto a los sistemas Fieldbus (IEC 61158) como a los lenguajes de programación definidos en la IEC 61131-3 para controladores programables. Debido a la estrecha relación con la IEC 61131-3, muchos expertos de ese proyecto participaron también en el desarrollo de la 61499. Tras varios años de trabajo y revisiones internacionales, se publicaron entre 2002 y 2004 las primeras especificaciones públicas (IEC/PAS 61499-1, -2 y -4) y un documento de apoyo (IEC/TR 61499-3). Posteriormente, con la experiencia acumulada, en 2005 estas PAS se convirtieron en normas oficiales IEC, y en 2012 se publicó la segunda edición [2].

Si bien la IEC 61131 permitió establecer un marco común y homogéneo para el desarrollo de aplicaciones de automatización industrial, su enfoque estaba principalmente orientado a sistemas centralizados, limitando su capacidad de adaptación a entornos más dinámicos y distribuidos. En contraposición, la IEC 61499 introduce una arquitectura basada en bloques funcionales distribuidos, lo que facilita la portabilidad, la interoperabilidad y la reutilización de componentes en sistemas heterogéneos.

Esta transición representa un cambio de paradigma: mientras que la IEC 61131 centra su atención en el ciclo de escaneo secuencial de los PLC, la IEC 61499 propone un modelo orientado a eventos que resulta más adecuado para la comunicación entre nodos y la coordinación de procesos en sistemas distribuidos. Gracias a esta orientación, la norma se posiciona como una herramienta clave para el diseño de arquitecturas de automatización flexibles, escalables y modulares, capaces de responder a los nuevos retos de la industria digitalizada.

Aunque la IEC 61499 ya ha demostrado su utilidad en ámbitos como la robótica, la automatización de procesos y, más recientemente, en el contexto de la Industria 4.0, su potencial aún está lejos de haberse explotado por completo. En particular, la aplicación de este estándar al control de electrolizadores para la producción de hidrógeno verde se encuentra todavía en una fase incipiente. Este vacío representa una oportunidad para investigar y evaluar su eficacia en un contexto estratégico para la transición energética. En este marco, el presente proyecto propone el diseño y desarrollo de un sistema de control de un electrolizador basado en la norma IEC 61499, con el objetivo de analizar sus ventajas frente a modelos tradicionales y contribuir al avance de soluciones tecnológicas en el ámbito de la energía sostenible.

1.1 Objetivos y metodología del trabajo

La creciente necesidad de reducir las emisiones de gases de efecto invernadero ha impulsado la transición hacia fuentes de energía renovables y tecnologías más limpias. En este contexto, el hidrógeno verde ha surgido como una solución prometedora para el almacenamiento de energía, la descarbonización de procesos industriales y la

promoción de una movilidad sostenible [3].

La obtención este hidrógeno verde se basa en la electrólisis del agua, proceso que consiste en separar el agua en hidrógeno y oxígeno utilizando electricidad, por medio de unos dispositivos denominados electrolizadores [4]. Entre los distintos tipos existentes, los electrolizadores PEM destacan por su alta eficiencia y rápida respuesta, lo que los hace ideales para su acoplamiento con fuentes de energía intermitentes. Sin embargo, esta producción requiere de sistemas de control avanzados que garanticen tanto la eficiencia energética como la seguridad del proceso, siendo aquí donde la automatización distribuida juega un papel clave.

Este proyecto tiene como objetivo principal el control de la temperatura de un electrolizador PEM utilizando la arquitectura IEC 61499; con el fin de optimizar su eficiencia energética, prolongar su vida útil y garantizar la estabilidad del proceso.

Para realizar este control, se dispone de un sistema de *Simulink* (herramienta de modelado de MATLAB R2024a) de un electrolizador PEM industrial y la aplicación *EcoStruxure Automation Expert* (EAE) de Schneider Electric, desde la cual se realizará el control de la temperatura. Simulink solo actuará como intermediario, aportando un modelo simulado de la planta.

La comunicación entre ambos entornos se realizará empleando el protocolo de comunicación OPC UA (Open Platform Communications Unified Arquitecture), garantizando la interoperabilidad entre sistemas. Este protocolo de comunicación será explicado en apartados posteriores.

Este enfoque permitirá validar la aplicabilidad de la norma IEC 61499 en el control de electrolizadores, contribuyendo así al desarrollo de sistemas más eficientes para la producción de hidrógeno verde.

Este proyecto se estructura en distintos capítulos que permiten abordar de manera progresiva el control de un electrolizador mediante la norma IEC 61499. En primer lugar, se presentan los fundamentos teóricos de la electrólisis, donde se explican los principios físicos y químicos que sustentan el proceso. Posteriormente, se expone la norma IEC 61499, destacando su relevancia para el diseño de sistemas de automatización distribuidos. A continuación, se detallan las características del sistema propuesto, describiendo tanto sus componentes como su arquitectura. Seguidamente, se aborda el desarrollo e implementación del control, explicando las etapas prácticas seguidas para llevar la teoría a la aplicación real. En el capítulo de resultados y análisis se evalúa el rendimiento del sistema implementado. Finalmente, se incluyen las conclusiones y trabajos futuros, donde se resumen los principales logros alcanzados y se proponen posibles líneas de mejora y ampliación.

2 FUNDAMENTOS TEÓRICOS DE LA ELECTRÓLISIS

La electrólisis es un proceso químico que consiste en la descomposición en iones de una sustancia en la disolución por medio de una corriente eléctrica [5]. Si concretamos un poco más, en nuestro caso la definiremos como el proceso industrial mediante el cual separamos la molécula de agua (H_2O) en sus gases fundamentales; en este caso en, en las moléculas de hidrógeno y oxígeno gaseoso (H_2 y O_2 respectivamente). La relación estequiométrica se puede ver en la siguiente ecuación química.

$$H_2O \rightarrow H_2 + \frac{1}{2} O_2$$

Ecuación 1. Relación estequiométrica de la electrólisis

Este proceso va a ser el fundamento para el desarrollo del electrolizador, un dispositivo que se aprovecha de la electrólisis para generar hidrógeno [6].

2.1 Hidrógeno

El hidrógeno es el elemento más abundante y el más ligero del universo. Es un elemento reactivo y, por tanto, se puede usar como combustible. Este es uno de los principales atractivos del uso del hidrógeno, ya que es un elemento relativamente fácil de obtener con una gran disponibilidad. Sin embargo, aunque esto sea así, el hidrógeno no se suele encontrar en la naturaleza en su forma de gas (H_2), por lo que se tiene conseguir como resultado de un proceso anterior.

En función de su forma de obtención podemos definir distintos tipos de hidrógeno:

- **Hidrógeno Verde**[1]. Se produce mediante la electrólisis del agua utilizando electricidad procedente de fuentes de energías renovables, como la eólica o la solar. La razón por la que se llama verde es que al utilizar energía renovable, su producción no genera emisiones de dióxido de carbono en ningún momento.

  Actualmente, el hidrógeno verde solo representa un pequeño porcentaje de la producción total de hidrógeno debido a los altos costos de su proceso. Sin embargo, tiene una excelente proyección de futuro, al ser el tipo de hidrógeno más limpio, lo que contribuirá a cumplir los planes de cero emisiones netas de carbono [7].

- **Hidrógeno Morado (Violeta), Rosa y Rojo**. El hidrógeno rosa se produce mediante la electrólisis del agua utilizando la electricidad de una central nuclear. El hidrógeno púrpura se obtiene mediante el uso de energía nuclear y calor mediante la electrólisis combinada y la descomposición termoquímica del agua. El hidrógeno rojo se genera mediante la descomposición catalítica del agua a alta temperatura utilizando energía nuclear térmica como fuente de energía [8].

[1] Este proyecto se centrará en la producción de este tipo de hidrógeno a través de energía solar

Estos métodos no liberan gases de efecto invernadero, pero dan como resultado residuos nucleares, los cuales son muy contaminantes.

- **Hidrógeno Amarillo**. Se produce mediante la electrólisis del agua utilizando la electricidad de la red eléctrica. Las emisiones de carbono varían significativamente con el tiempo, dependiendo de las fuentes de energía de la red [7].
- **Hidrógeno Gris**. El hidrógeno gris se produce a partir de combustibles fósiles y suele emplearse el método de reformado de metano con vapor (SMR) o el método de oxidación parcial. Durante este proceso, se produce CO_2 y finalmente se libera a la atmósfera. Compone la mayor parte del hidrógeno producido [7] [8].
- **Hidrógeno Turquesa**. En este proceso también se emplea el metano como materia prima, pero en este caso, se produce el hidrógeno mediante la división térmica del metano mediante la pirólisis del mismo. Este proceso (aunque en etapa experimental) elimina el carbono en forma sólida en lugar de gas CO_2, lo que lo hace más fácil de almacenar y reutilizar [7] [8].
- **Hidrógeno Negro/Marrón**. Producido a partir del carbón, los colores negro y marrón se refieren al tipo de carbón bituminoso (negro) y lignito (marrón). La gasificación del carbón es un método utilizado para producir hidrógeno. Sin embargo, es un proceso muy contaminante, y se producen CO_2 y monóxido de carbono como subproductos que se liberan a la atmósfera [8].
- **Hidrógeno Azul**. El hidrógeno azul se refiere al hidrógeno derivado del gas natural, un combustible fósil. Sin embargo, la mayor parte (aunque no la totalidad) del CO_2 emitido durante el proceso se capturaría y almacenaría bajo tierra (secuestro de carbono) o se consolidaría en un producto sólido para su posterior utilización. Esto se denomina captura, almacenamiento y utilización de carbono (CCSU). Al producirse a partir de combustibles fósiles, el hidrógeno azul actualmente tiene menores costos que el hidrógeno verde [8].
- **Hidrógeno acuoso**. La tecnología de hidrógeno acuoso consiste en introducir oxígeno en un depósito de combustible sellado entre granos de roca, utilizando petróleo no barrido como combustible y el suelo como recipiente del reactor. El proceso comienza separando el aire atmosférico en nitrógeno y oxígeno mediante unidades de separación de aire. El oxígeno se envía a un depósito subterráneo, donde se aumenta la temperatura, lo que promueve la transferencia de agua-gas, la gasificación en caliente y las reacciones de termólisis en agua dentro del depósito de combustible, generando así gas de síntesis.

  Finalmente, el H_2 se extrae mediante membranas en los pozos de producción. Estas membranas están compuestas de una aleación de paladio, donde los óxidos de carbono permanecen bajo tierra y el H_2 reacciona con el paladio. El proceso es altamente eficiente porque toda la conversión de energía se realiza bajo tierra. Esta tecnología podría ser una forma limpia de generar hidrógeno utilizando reservas de petróleo económicamente irrecuperables, extrayendo únicamente hidrógeno a la superficie [7] [9].
- **Hidrógeno Blanco**. Se refiere al hidrógeno natural en su estado más natural. Se encuentra en la naturaleza como gas libre en las capas de la corteza continental, en las profundidades de la corteza oceánica o en gases volcánicos, géiseres y sistemas hidrotermales.

  Los procesos que intervienen en la formación natural del hidrógeno no se comprenden bien. Algunas hipótesis son la desgasificación del hidrógeno del núcleo terrestre, la reacción del agua con rocas ultrabásicas (es decir, serpentinización) o con agentes reductores en el manto, la radiólisis natural (es decir, la disociación del agua por uranio o plutonio) y la descomposición de la materia orgánica.

  Como se ha mencionado, el reformado con vapor produce hidrógeno gris con alta eficiencia y bajo coste, pero también con una huella de carbono catastrófica. La electrólisis, por su parte, genera hidrógeno verde a partir de fuentes de energía renovables, pero pierde casi el 70 % del aporte energético durante el proceso.

  El hidrógeno blanco formado naturalmente se presenta como una fuente prometedora, abundante y libre de carbono, que requiere una infraestructura mínima para su explotación. Debido al escaso nivel de investigación sobre el tema, aún resulta difícil evaluar los recursos mundiales de hidrógeno blanco [7] [8].

Se pueden realizar comparaciones de costos y emisiones para los diferentes colores de hidrógeno identificando

sus especificaciones y recopilando datos de su evaluación económica. El factor principal a considerar en esta comparación es el rango entre el costo y las emisiones de CO_2. Los precios se expresan en USD/kgH_2 y las emisiones se expresan en kg de CO_2 liberado a la atmósfera. La Tabla 1 muestra un resumen de los resultados obtenidos entre los años 2021-2022.

Color	**Coste [USD/kgH_2]**	**Emisiones [kgCO_2/kgH_2]**
Verde	2.28-7.39	0
Morado, Rosa y Rojo	2.18-5.92	0
Amarillo	6.06-8.81	*[2]
Gris	0.67-1.31	8.5
Negro/Marrón	1.2-2.0	20
Turquesa	2.0	**[3]
Azul	0.99-2.05	1-2
Acuoso	0.23	0

Tabla 1. Comparación entre los colores del hidrógeno en términos de costs y emisiones de carbono

Según los datos de la Tabla 1, el color de hidrógeno más económico sería el acuoso. Sin embargo, cabe destacar que no se ha probado adecuadamente y aún se encuentra en una etapa inicial. Los siguientes colores de hidrógeno más económicos son el gris, el negro y el azul, que consumen combustibles fósiles, desde gas natural hasta carbón, y, por lo tanto, dependen de sus precios. Por lo tanto, estos son los más contaminantes, mientras que el hidrógeno negro es menos ecológico, seguido del hidrógeno gris. El hidrógeno azul aún genera algunas emisiones, pero el sistema de captura y almacenamiento de carbono (CAC) las reduce considerablemente. El hidrógeno turquesa tiene un costo considerablemente bajo, a pesar de ser una tecnología que requiere más pruebas. Su subproducto puede venderse para diversas aplicaciones y, al ser sólido, no emite emisiones de GEI. Los demás colores tienen los costos más altos, empezando por el hidrógeno púrpura o rosa, cuya energía proviene de una planta nuclear. El método de producción suele ser la electrólisis del agua, que requiere mejoras. Le sigue el hidrógeno verde, con altos costos de producción. Estos deben reducirse significativamente para establecerse como el método de producción líder. El hidrógeno amarillo es el más caro, dependiendo de la ubicación y, por lo tanto, de la combinación energética utilizada. La composición de la combinación energética también determina la cantidad de emisiones generadas. Algunos de estos métodos han sido tecnologías maduras durante años, como las técnicas de gasificación por SMR. En cambio, otros comenzaron a destacarse en las últimas décadas o ni siquiera se han explorado mucho, como es el caso de la extracción de hidrógeno blanco, por ejemplo. Todas estas son fuentes de hidrógeno que pueden explorarse más para producir una mayor cantidad de hidrógeno y convertirlo en el principal vector energético del futuro. Lamentablemente, todas las tecnologías que emiten emisiones significativas de GEI deben abandonarse gradualmente y optar por soluciones más sostenibles [7]. Debido a esto, este proyecto se centrará en la obtención del hidrógeno verde mediante la electrólisis del agua.

[2] * Las emisiones de hidrógeno amarillo dependen de la ubicación y su respectiva combinación energética
[3] ** El hidrógeno turquesa produce un subproducto de carbono sólido.

2.2 Electrolizadores. Tipos de electrolizadores

Por lo explicado anteriormente, se define a un electrolizador como un dispositivo en el que se lleva a cabo el proceso de electrólisis. Podemos definir ciertas partes fundamentales en un electrolizador:

- **Stack (o Pila de hidrógeno)**. Es el corazón del electrolizador. En él se produce el proceso de electrólisis y por tanto es el elemento más importante de este. El resto de los sistemas se construyen entorno al Stack para conseguir las condiciones oportunas para su correcto funcionamiento. Para este proyecto se van a definir dos tipos de nomenclaturas, cuando nos refiramos al dispositivo que transforma el agua en hidrógeno y oxígeno mediante electricidad lo mencionaremos como Stack mientras que cuando hablemos del dispositivo que realiza el proceso inverso nos referimos como pila de hidrógeno. Esto no es rigurosamente así ya que una pila de hidrógeno puede realizar tanto el proceso propio como el inverso, pero se va a definir así para que sea más claro.
- **Bomba hidráulica**. Al trabajar en un circuito hidráulico, es necesario el uso de bombas hidráulicas que ayuden al movimiento del agua por el electro para tener un control sobre el flujo y la presión.
- **Intercambiadores de calor**. La temperatura a la que se desarrolla el proceso es fundamental tanto a la hora de obtener un mayor rendimiento del sistema como de trabajar en un punto de operación que no lastime los dispositivos. Esta parte del dispositivo es crucial para el control de la temperatura que se desarrollará en este proyecto.
- **Separadores**. Durante el proceso se va a trabajar con mezclas de gases y líquidos, y es importante que los separemos de la mejor manera posible. Esto es crucial ya que el hidrógeno es un gas explosivo y hay que evitar a toda costa que este circule por sitios a los que no está destinado. Para evitar esto último se incorporarán todas las medidas de seguridad necesarias.
- **Purificación**. Como veremos más adelante, la calidad del hidrógeno debe seguir unos estándares marcados por la ISO 14687. Para ello va a ser necesario instalar a la salida de hidrógeno una etapa de purificación.
- **Planta de Tratamiento de agua**. Se requiere de una calidad mínima del agua para que el Stack funcione correctamente, por este mismo motivo se mejora la calidad del agua hasta que alcance límites aceptables.
- **Filtros y Potes de condensado**. Acumulan las impurezas y la humedad en partes del sistema donde es vital mantener una serie de condiciones de calidad.
- **Rectificador**. El Stack trabaja con un voltaje determinado y con corriente continua. Para ello necesitamos un rectificador que convierta la corriente y voltaje de entrada a los valores de funcionamiento nominal. Exploraremos las condiciones más adelante.
- **Instrumentación y Válvulas de control**. Para monitorizar y controlar el funcionamiento de la planta electrolizadora.

Como ya se ha mencionado, en el Stack se produce el proceso de electrólisis. Como se lleva a cabo dicho proceso depende de la tecnología con la cual se haya desarrollado el mismo. Todas comparten una fisiología externa, todos poseen un ánodo por el que entra el agua y sale el oxígeno y un cátodo por el que sale el hidrógeno [6].

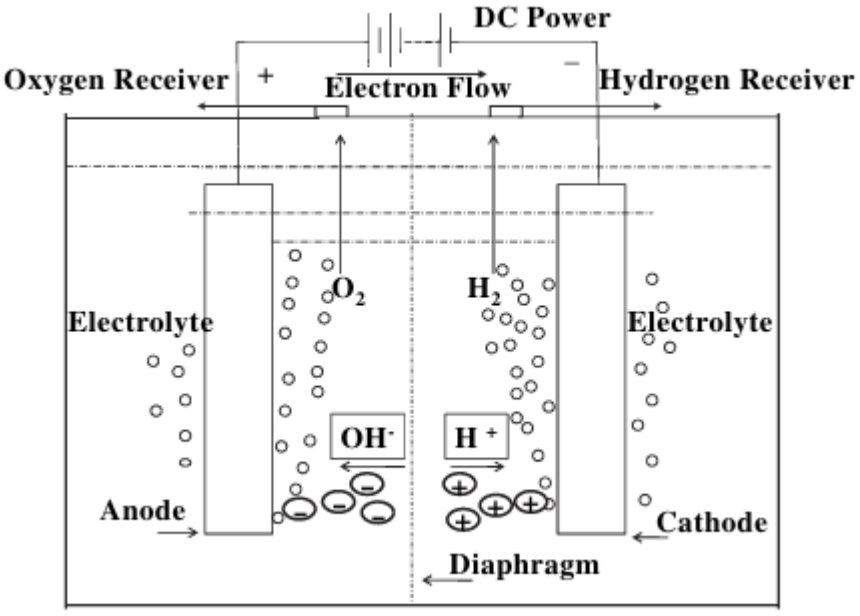

Figura 1. Ilustración esquemática de un sistema básico de electrólisis de agua [10]

Podemos diferenciar cuatro tipos de tecnología prominente de Stack.en la actualidad.

2.2.1 Alcalinos (AEL)

Los electrolizadores alcalinos representan una de las tecnologías más consolidadas para la producción de hidrógeno por electrólisis. Su funcionamiento se basa en una solución líquida de hidróxido de potasio (KOH) o hidróxido de sodio (NaOH) como electrolito, y utilizan un separador poroso entre ánodo y cátodo. Las reacciones electroquímicas principales son:

$$4OH^- \rightarrow O_2 + 2H_2O + 4e^-$$

Ecuación 2. Relación estequiométrica en el ánodo

$$4H_2O + 4e^- \rightarrow 2H_2 + 4OH^-$$

Ecuación 3. Relación estequiométrica en el cátodo

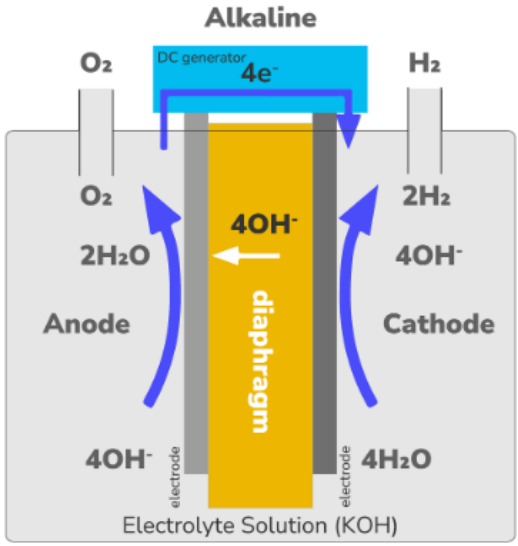

Figura 2. Celda de electrolizador alcalino [11]

Los materiales utilizados en los Stacks suelen ser níquel o acero inoxidable, lo cual contribuye a reducir los costes de fabricación. Aunque su eficiencia ronda entre el 60 y el 70 %, presentan ciertas limitaciones, como una menor densidad de corriente y tiempos de respuesta más lentos frente a cargas dinámicas, lo que dificulta su integración directa con fuentes renovables intermitentes [10].

2.2.2 PEM (Proton Exchange Membrane)

Los electrolizadores de membrana de intercambio de protones (PEM) utilizan una membrana polimérica sólida, típicamente Nafion®, que actúa como electrolito. Esta membrana permite el paso de protones (H^+) desde el ánodo al cátodo, donde se produce el hidrógeno, mientras bloquea el paso de gases como el oxígeno. Las reacciones electroquímicas son:

$$H_2O \rightarrow \frac{1}{2}O_2 + 2H^+ + 2e^-$$

Ecuación 4. Relación estequiométrica en el ánodo

$$2H^+ + 2e^- \rightarrow H_2$$

Ecuación 5. Relación estequiométrica en el cátodo

$$H_2O + 2e^- \rightarrow H_2$$

Ecuación 6. Relación estequiométrica de la reacción resultante en PEM

La electrólisis PEM es un proceso inverso de un proceso de pila de combustible PEM (PEM fuel cell process) (Fig. 3). El agua se divide en oxígeno, protones y electrones en un electrodo (ánodo) aplicando un voltaje de CC superior a un voltaje termoneutro (1,482 V). Los protones pasan a través de la membrana del electrolito de polímero y en el cátodo se combinan con los electrones para formar hidrógeno. El paso de protones a través de la membrana está acompañado por el transporte de agua (arrastre electroosmótico).

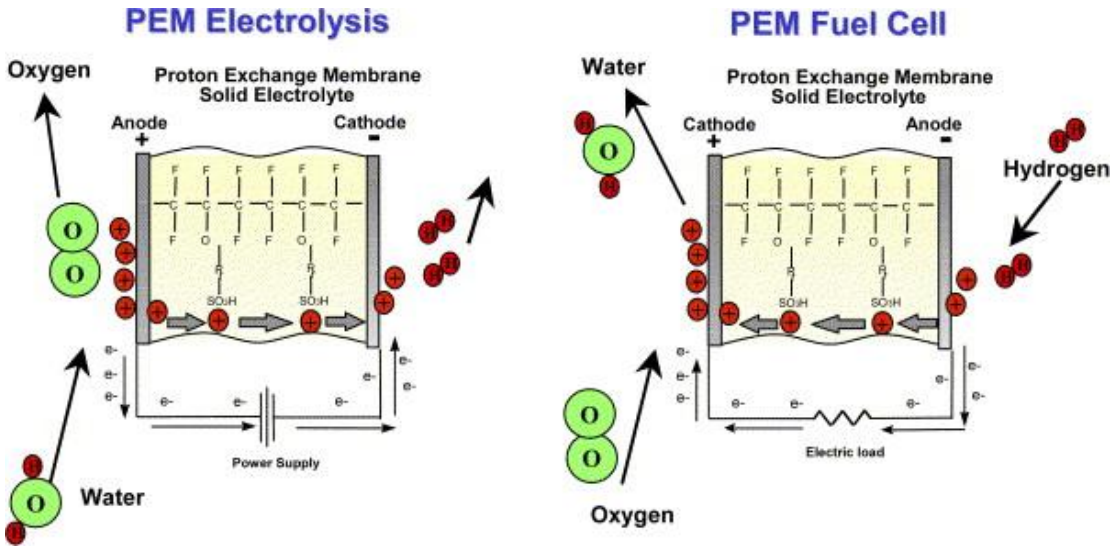

Figura 3. Representación esquemática de los procesos de electrólisis PEM y de pila de combustible [11]

Se propuso como una mejora respecto a la electrólisis alcalina; siendo más segura, mayor eficiencia resistiendo presión y mezcla de gases y un mayor rendimiento, siendo capaz de trabajar con mayores intensidades y con flujos de entrada y salida más grandes. Sin embargo, el empleo de materiales nobles como el platino e iridio en los electrodos y titanio en las placas bipolares encarece significativamente el coste del sistema [6] [11].

En este proyecto vamos a trabajar con un electrolizador con tecnología PEM.

2.2.3 SOEC (Solid Oxide Electrolyzer Cell)

Los stacks de tipo SOEC, están formados de tres capas cerámicas. Una capa densa de electrolito y dos capas porosas de electrodos correspondientes al ánodo y cátodo. La capa cerámica actúa como como separador de gases y electrolito y cuando se aplica un voltaje, se produce la separación de H_2 y O_2. Los electrodos utilizados tienen que ser altamente conductivos de electrones e iones y se ha de maximizar la superficie de contacto. Las capas se pueden apilar de distintas maneras, pero la función es la misma.

Los electrolizadores de óxido sólido operan a altas temperaturas (entre 600 y 850 °C) y utilizan un electrolito cerámico sólido conductor de iones de oxígeno, como el óxido de zirconio estabilizado con itrio (YSZ). A estas temperaturas, es posible realizar la electrólisis del vapor de agua con una eficiencia termodinámica superior al 90 % si se aprovecha el calor residual de otras fuentes industriales. En estas condiciones se consigue una oxidación de agua en el lado del ánodo y una reducción de agua en el lado del cátodo. Lo que nos da una reacción neta de separación de hidrógeno y oxígeno en su forma gas [6] [12].

$$2O^{2-} \rightarrow O_2 + 4e^-$$

Ecuación 7. Relación estequiométrica de oxidación en ánodo

$$H_2O + 2e^- \rightarrow H_2 + O^{2-}$$

Ecuación 8. Relación estequiométrica de oxidación en cátodo

$$2H_2O \rightarrow H_2 + O_2$$

Ecuación 9. Relación estequiométrica de la reacción resultante en SOEC

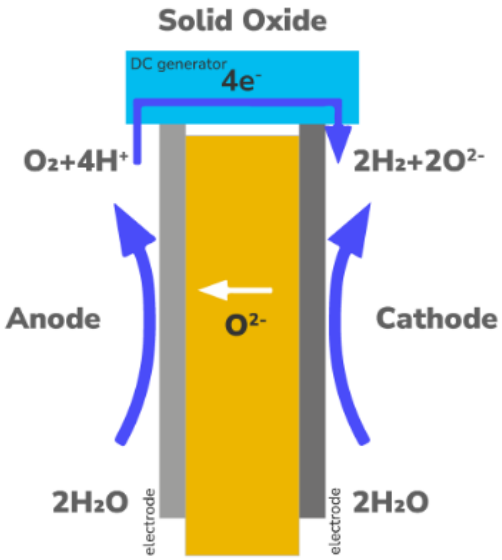

Figura 4. Celda de electrolizador SOEC [11]

2.2.4 AEM (Anion Exchange Membrane)

Los electrolizadores de membrana de intercambio aniónico (AEM) constituyen una tecnología emergente que combina las ventajas de los sistemas alcalinos y PEM. En lugar de protones, la membrana AEM transporta aniones OH^-, lo que permite el uso de catalizadores menos costosos, como metales no nobles, y soluciones alcalinas diluidas.

A diferencia de los AEL, los electrolizadores AEM utilizan una membrana polimérica sólida, lo que mejora la compacidad y reduce riesgos de corrosión asociados a soluciones líquidas concentradas. Su eficiencia y estabilidad aún están en desarrollo, pero presentan un gran potencial para producir hidrógeno verde a bajo coste con materiales más sostenibles [13].

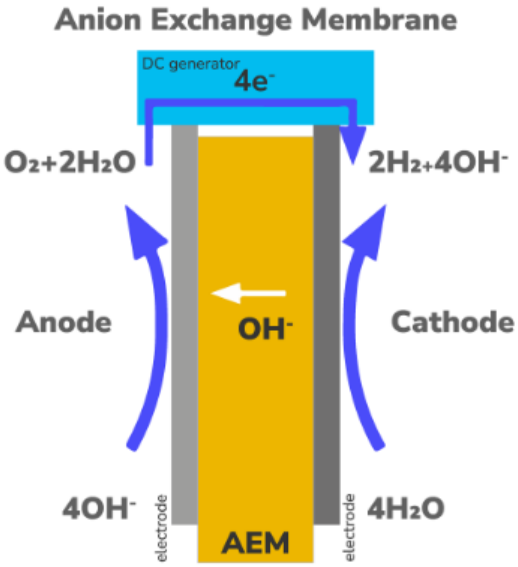

Figura 5. Celda de electrolizador AEM [11]

2.3 Aplicaciones del hidrógeno verde

El hidrógeno verde presenta un amplio abanico de aplicaciones potenciales y reales en diversas industrias, no solo por sus propiedades energéticas, sino también por su capacidad para reducir emisiones en procesos intensivos en carbono. A continuación, se describen las principales áreas de aplicación, basadas en el estudio de Jiménez Sáez (2020), complementadas con fuentes académicas relevantes.

- **Industria química**. Históricamente, la industria química ha sido uno de los principales consumidores de hidrógeno, utilizándolo como insumo clave en la producción de compuestos como el amoniaco (NH_3), metanol, y como catalizador en síntesis orgánicas y procesos farmacéuticos [14]. La generación de hidrógeno verde permitiría descarbonizar estos procesos, reduciendo significativamente su huella ambiental.
- **Procesos térmicos e industriales**. El hidrógeno también puede actuar como fluido térmico en ciclos combinados de generación eléctrica, siendo útil para enfriar turbinas o motores y mejorar la eficiencia energética. En algunos casos, puede participar como combustible en turbinas secundarias, maximizando la recuperación de energía térmica [14].
- **Refinación de combustibles fósiles**. Si bien la refinación de combustibles se lleva a cabo mediante procesos químicos, dado el tamaño de este mercado y la escala de producción de que representa vale la pena considerar la industria de los combustibles por separado [14]. En esta industria, el hidrógeno se utiliza en procesos como el hydrotreating y el hydrocracking, necesarios para purificar hidrocarburos pesados y descomponer moléculas grandes en compuestos más ligeros. El uso de hidrógeno verde en esta industria permitiría reducir las emisiones asociadas al tratamiento de combustibles [15].
- **Minería**. En sectores como la minería del cobre, el hidrógeno se emplea en procesos de pirorefinación para eliminar impurezas como oxígeno, contribuyendo así a la obtención de un metal más puro. Además, puede sustituir al diésel en maquinaria pesada, reduciendo emisiones y dependencia de combustibles fósiles [14].
- **Sector energético y almacenamiento**. El hidrógeno verde es una solución clave para el almacenamiento estacional de energía, particularmente en sistemas eléctricos con alta penetración de energías renovables variables como la solar o eólica. En este contexto, permite absorber excedentes de producción y generar electricidad posteriormente mediante celdas de combustible [16]. También puede prestar servicios de control de frecuencia gracias a su rápida respuesta operativa.
- **Transporte y electromovilidad**. La electromovilidad basada en hidrógeno (Fuel Cell Electric Vehicles, FCEV) es una alternativa prometedora para reemplazar combustibles fósiles en vehículos pesados, buses y trenes. Los vehículos a hidrógeno pueden superar las limitaciones de autonomía de los modelos basados únicamente en baterías [3].
- **Generación eléctrica autónoma y micro-redes**. En zonas aisladas o de difícil acceso, las micro-redes energéticas pueden beneficiarse del uso de hidrógeno para el almacenamiento y generación eléctrica, actuando como vector energético limpio y flexible para asegurar el suministro [14].

2.4 Control de los procesos del electrolizador

La electrólisis del agua se integra como una etapa crítica dentro de sistemas más amplios, donde la eficiencia y la seguridad operativa son prioritarias. Dado que los procesos industriales implican variables como temperatura, presión, caudal y consumo eléctrico, se requiere una estrategia de control robusta y flexible [17]. En un electrolizador se pueden controlar el flujo y la pureza del agua de alimentación, la **temperatura** del sistema y la presión del hidrógeno y oxígeno producidos para optimizar la eficiencia y seguridad del proceso. Otros parámetros clave incluyen el voltaje, la corriente eléctrica, el nivel de electrolito y la concentración de gas en la salida, además de la detección de fugas para garantizar la seguridad. Como ya se ha mencionado anteriormente, este proyecto se centra en el control de la temperatura del sistema.

La implementación de control distribuido sobre un electrolizador permite no solo una mejor supervisión operativa, sino también la integración con sistemas de gestión energética y unidades de almacenamiento,

facilitando la interoperabilidad y la respuesta adaptativa ante eventos del sistema [18]. Además, la capacidad de realizar diagnósticos y corrección de fallos en tiempo real mejora la disponibilidad y confiabilidad de la planta industrial. Por tanto, comprender el papel del electrolizador dentro de un proceso industrial, así como sus requisitos de control, resulta clave para el desarrollo de sistemas eficientes, seguros y normativamente compatibles.

La automatización industrial ha sido tradicionalmente implementada mediante arquitecturas centralizadas conforme a normas como IEC 61131, sin embargo, el crecimiento en la complejidad de los sistemas, la necesidad de escalabilidad y la integración con fuentes renovables han incentivado la adopción de soluciones distribuidas, como las que propone la norma IEC 61499 [1]. Esta norma permite el diseño de aplicaciones modulares, reutilizables y descentralizadas, adecuadas para procesos complejos y heterogéneos.

3 NORMA IEC 61499

Como ya se ha explicado en apartados anteriores, la norma IEC 61499 surge por la necesidad de trabajar con arquitecturas distribuidas y sistemas cada vez más complejos, solucionando las capacidades del tradicional estándar IEC 61131-3.

3.1 Introducción a la norma IEC 61499

La norma IEC 61499 es un estándar internacional desarrollado por la Comisión Electrotécnica Internacional (IEC) que define un estándar para el diseño de sistemas de automatización industrial distribuidos. Su enfoque se basa en bloques de función (Function Blocks, FBs) que encapsulan tanto lógica de control como interfaces de comunicación, permitiendo que las aplicaciones se estructuren como redes de bloques conectados mediante eventos y datos. Esta separación estructural entre eventos y datos es una de las características más distintivas del estándar, ya que proporciona una mayor claridad en el diseño del flujo de control [1] [19].

El funcionamiento se basa en cadenas de bloques de función que solo funcionan cuando se activan por eventos. Cada FB tiene entradas y salidas de eventos y de datos, los cuales no se pueden combinar. El bloque no realizará ninguna acción a menos que un evento lo autorice. Los eventos se definen para asociarse con datos y solo se utilizarán los datos asociados con el evento que los active (esto se aplica tanto para las entradas como para las salidas) [20]. Un evento de salida de un bloque funcionará como evento de entrada de otro, permitiendo ejecutar el bloque cuando se active el evento. Los eventos y los datos siempre fluirán de las salidas a las entradas y, por lo general, las entradas de bloque no se pueden conectar a varias salidas de bloque, ya que se produciría un conflicto de estados [20].

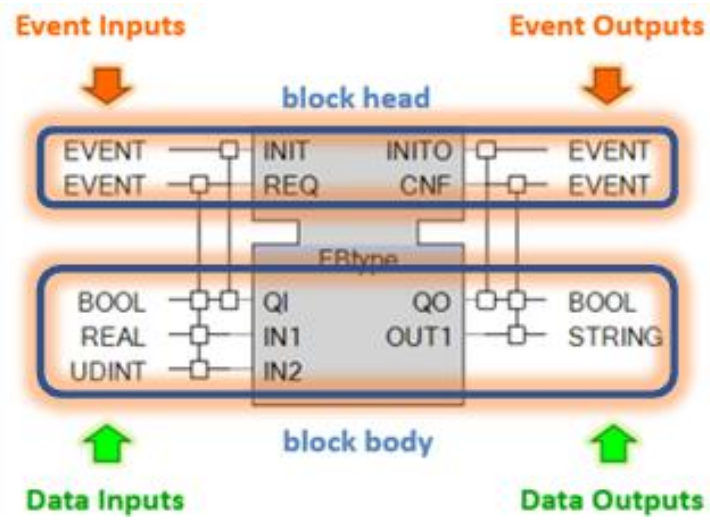

Figura 6. Ejemplo de BFB en IEC 61499

Evento	Tipo	Función
INIT	Entrada	Inicializar el bloque
INITO	Salida	Confirmar que fue inicializado
REQ	Entrada	Ejecutar la lógica del bloque
CNF	Salida	Indicar que la ejecución fue completada

Tabla 2. Eventos predeterminados IEC 61499

Dentro de los FBs, el Diagrama de Control de Ejecución (ECC) define qué sucede cuando se activa uno de los eventos de entrada de un bloque de función. La mayoría de los bloques de función tienen un ECC que se puede definir para el bloque. Consiste en una serie de estados que están conectados mediante transiciones, las cuales determinan qué estados se ejecutan en cada instante [20].

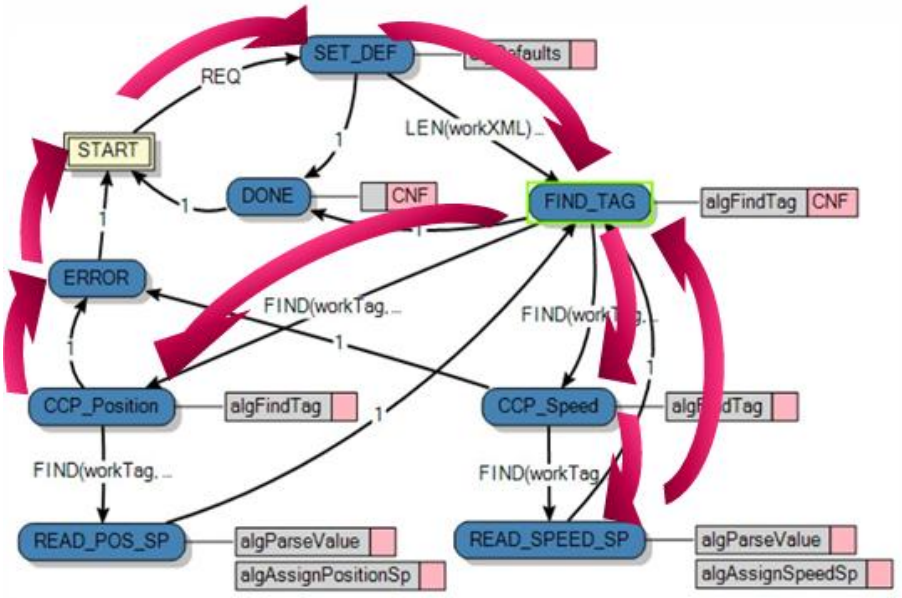

Figura 7. Ejemplo de ECC de un FB en IEC 61499

Dentro de la norma, existen tres tipos diferentes de bloques de función: Basic Function Block (bloque de procesamiento básico), Composite Function Block (compuesto por varios Basic FBs) y Service Interface Function Block (código interno oculto) [20]. En los apartados posteriores se profundizará más en ellos.

La comunicación en los programas según el estándar IEC 61499 se estructura jerárquicamente en tres niveles principales: Aplicación (Application), Recurso (Resource) y Dispositivo (Device). A continuación, se describe cómo funciona cada uno y cómo se relacionan entre sí:

- **Aplicación**. Es la unidad de software que contiene el código lógico que debe ejecutarse. Esta puede incluir múltiples FBs, y representa el comportamiento general que se desea logra. Las aplicaciones pueden ser mapeadas a múltiples recursos, permitiendo su distribución en varios dispositivos dentro de un sistema.
- **Recurso**. Representa una unidad de ejecución lógica dentro de un dispositivo físico. Puede ejecutar una parte de la aplicación asignada. El recurso administra la conexión con (Fig.8):
  - El hardware físico mediante el mapeo de proceso (Process mapping).

- Otros dispositivos mediante el mapeo de comunicación (Communication mapping).

- **Dispositivo**. Es el hardware físico (como un PLC) al que se le pueden desplegar aplicaciones. Cada dispositivo contiene uno o más recursos. Tiene dos tipos de interfaz:
  - Process Interface, para el E/S físico.
  - Communication Interface, para enlazarse con otros dispositivos.

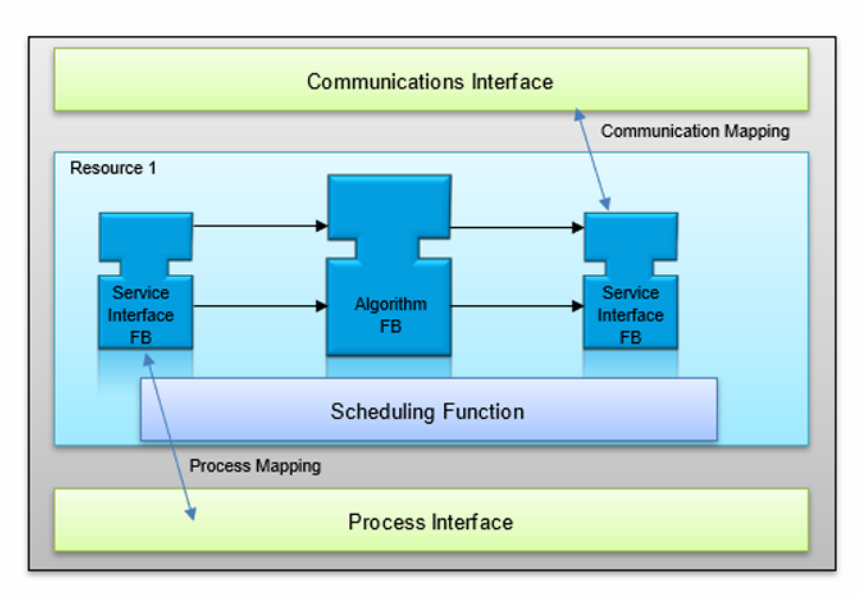

Figura 8. Conexión en recursos

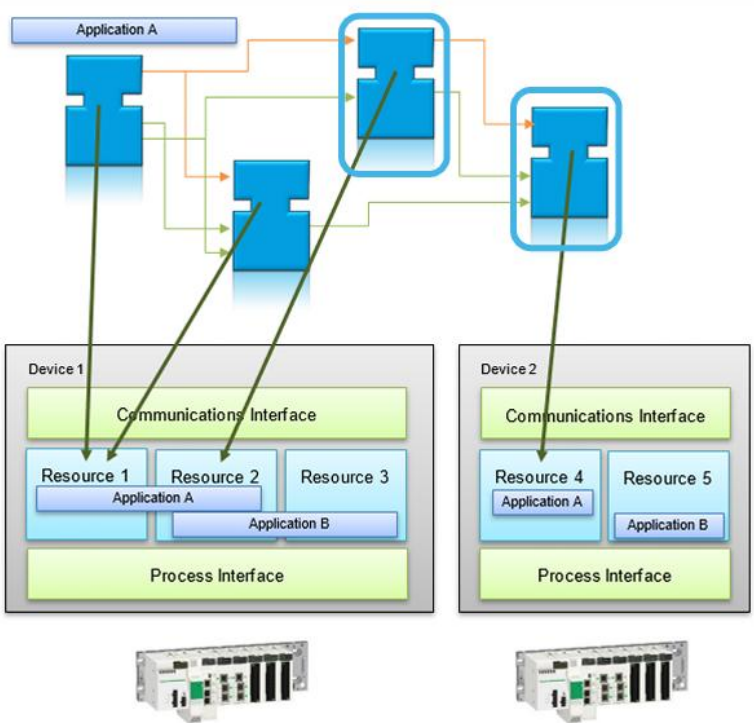

Figura 9. Conexiones y mapeado en programas IEC 61499

3.2 Comparación con la norma IEC 61131-3

Los estándares IEC 61131-3 e IEC 61499 representan dos paradigmas distintos en la programación de sistemas de automatización industrial. Mientras que la norma IEC 61131-3 ha sido históricamente la más adoptada en la industria, especialmente en el desarrollo de PLCs, IEC 61499 ha surgido como respuesta a los nuevos retos que impone la Industria 4.0, como la necesidad de sistemas distribuidos, flexibles y reconfigurables.

Tal como se ilustra en la Figura 10, la implementación bajo IEC 61131-3 está orientada a sistemas de control centralizado, donde toda la lógica de control se ejecuta en un único controlador, de manera cíclica y secuencial, como fue el caso del Micrologix 1000 usando plataformas como RSLogix. En contraste, IEC 61499 se basa en una arquitectura distribuida , como se evidenció con el uso de herramientas como nxtSTUDIO y controladores DCSmini, en la que diferentes partes de la aplicación pueden ejecutarse en diferentes dispositivos interconectados por red, facilitando la modularidad y escalabilidad del sistema [21].

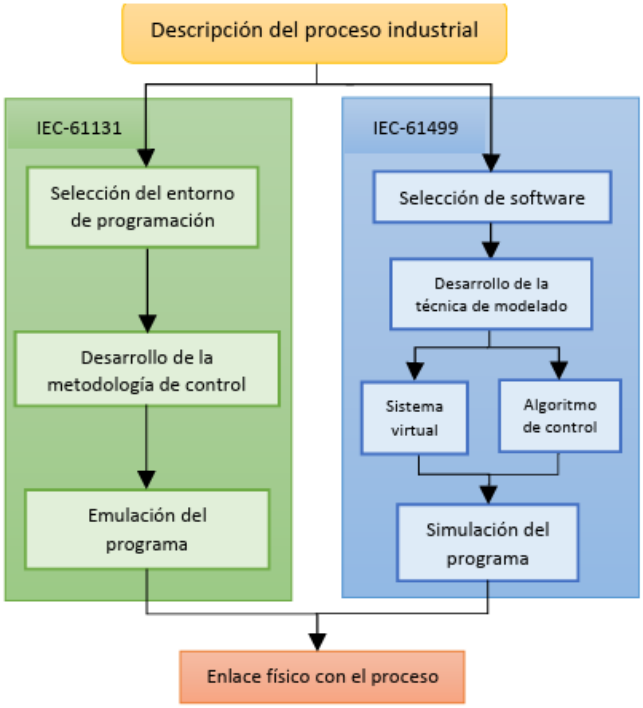

Figura 10. Etapas para la implementación en cada uno de los estándares

En IEC 61131-3, los bloques de función se ejecutan de forma cíclica, siguiendo un orden implícito definido por la herramienta de desarrollo. Por el contrario, como se explicó en el apartado anterior, IEC 61499 introduce un modelo basado en eventos, donde los bloques solo se activan cuando reciben un evento de entrada. Esto permite una ejecución más reactiva y eficiente en sistemas distribuidos [21].

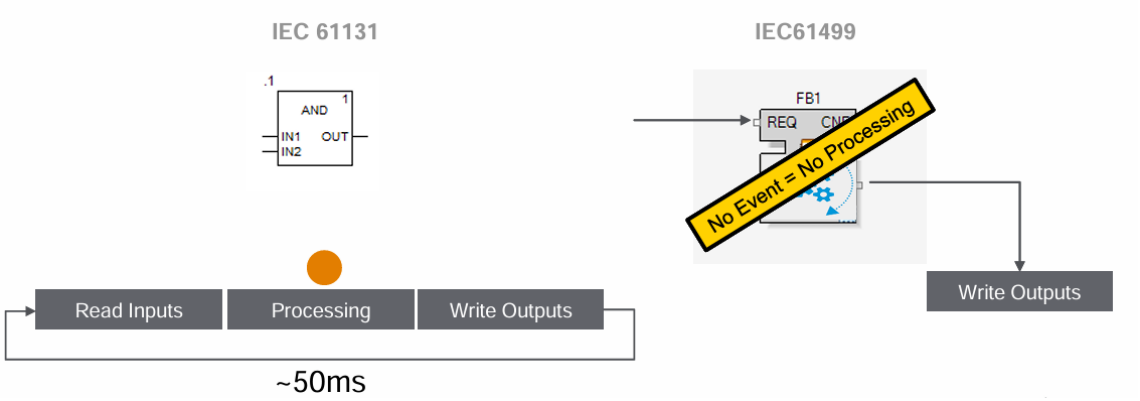

Figura 11. Comparación entre FBs de IEC 61131-3 vs. IEC 61499

IEC 61499 incorpora características de programación orientada a objetos de forma más natural, como encapsulamiento, reutilización de componentes y composición jerárquica. Esto permite modificaciones más ágiles al sistema, como la incorporación de nuevos módulos sin alterar el resto de la lógica. Por su parte, IEC 61131-3 es menos adaptable a cambios, lo cual puede incrementar significativamente el esfuerzo de desarrollo en sistemas complejos [21].

Un aspecto crucial en aplicaciones industriales críticas es la continuidad operativa. En sistemas IEC 61131-3, un fallo en el único PLC que ejecuta la lógica detiene todo el sistema. En cambio, en IEC 61499, el fallo de un dispositivo afecta únicamente a la parte del sistema que le ha sido asignada, aumentando la disponibilidad global del sistema [21].

Como conclusión, la norma IEC 61499 responde mejor a los principios de la Industria 4.0: portabilidad, interoperabilidad, modularidad e inteligencia descentralizada. Aunque IEC 61131-3 ha comenzado a incorporar algunos elementos de orientación a objetos, sus características siguen limitadas frente a las necesidades actuales.

3.3 Modelo basado en bloques funcionales. Tipos de FB

Como se ha explicado en apartados anteriores, el estándar IEC 61499 se fundamenta en una arquitectura orientada a bloques funcionales (FB) como unidad principal de diseño, modelado y ejecución en sistemas de automatización industrial. En IEC 61499 los FB son entidades reactivas, modulares y orientadas a eventos, permitiendo el desarrollo de aplicaciones distribuidas, portables y reutilizables, ideales para los desafíos de la Industria 4.0. En este apartado se profundizará más a fondo en el contenido y funcionamiento de los FB, así como los diferentes tipos de bloques.

Según lo explicado en el apartado 3.1, un Function Block en IEC 61499 encapsula:

- Una interfaz de eventos (entrada/salida)
- Una interfaz de datos asociada a esos eventos
- Variables internas
- Un comportamiento dinámico especificado mediante un Execution Control Chart (ECC).

El FB solo ejecuta su lógica cuando recibe un evento de entrada. En ese momento, procesa los datos asociados y puede generar eventos y datos de salida, siguiendo las transiciones del ECC (Fig. 7).

IEC 61499 define varios tipos de bloques funcionales, cada uno con características y finalidades específicas. A continuación se detallan.

3.3.1 Basic Function Block (BFB)

“A Basic Function Block consists of an Execution Control Chart (ECC), a set of algorithms, and a set of internal

variables" (IEC 61499-1:2012, Clause 4.5.2).

Es el tipo más simple de FB (Fig. 6). Contiene un ECC que define su comportamiento en términos de estados y transiciones. Cada estado puede ejecutar uno o más algoritmos, escritos en lenguajes como Structured Text (ST) o incluso C, y activar eventos de salida; predefinidos en acciones (se pueden asociar varias acciones a un mismo estado) [22].

Las transiciones gestionan el flujo del ECC desde el estado inicial hasta el final. Se gestionan a través de condiciones que activan las transiciones al producirse. En el caso de que una condición active más de una transición, se ejecutará aquella que tenga mayor prioridad. Esta prioridad de selección de transiciones puede ser modificada por el programador en cualquier instante [22].

Este tipo de bloque es el más adecuado para representar operaciones lógicas, cálculos y pequeñas máquinas de estado.

Algunos ejemplos típicos de podrían ser un controlador ON/OFF, una puerta lógica, un comparador o un bloque de retardo.

3.3.2 Composite Function Block (CFB)

Los CFB son bloques de función que contienen una red de otros bloques de función (basic, service o incluso composite) [20]. La ventaja de este tipo de bloques es que permite agrupar lógica compleja dentro de un bloque encapsulado, lo que facilita la comprensión de la aplicación y reutilizar el código sin necesidad de realizar pruebas cada vez. Esta encapsulación se puede anidar para proporcionar capas de funcionalidad [23].

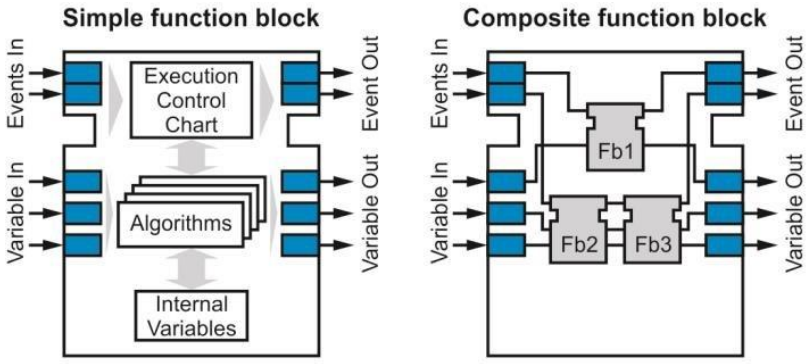

Figura 12. BFB vs. CFB

Los bloques de función compuestos en IEC 61499 no contienen directamente entradas y salidas digitales, ya que los elementos de automatización complejos no pueden incluirse de forma directa dentro de ellos. Para permitir la interacción con el entorno externo o con otros bloques, es necesario agregar interfaces explícitas. Estas interfaces actúan como puntos de conexión y pueden enlazarse tanto con bloques funcionales internos como con otros elementos del sistema. Además, también es posible definir nuevas interfaces dentro del propio bloque compuesto, configurando sus nombres, tipos de datos asociados y eventos correspondientes. Una vez definidas, estas interfaces pueden ser conectadas libremente según las necesidades del diseño del sistema [23].

3.3.3 Service Function Block (SIFB)

"Service interface function blocks provide access to non-standard services, such as communication or process I/O, that are outside of the control application" (IEC 61499-1:2012, Clause 4.5.4).

Los SIFBs permiten acceder a servicios del sistema que se encuentran fuera del ámbito directo de la aplicación de control, como la entrada/salida de procesos, la comunicación entre dispositivos o la temporización. A diferencia de otros tipos de bloques funcionales, su comportamiento no está definido mediante un ECC, sino

que su lógica interna se encuentra encapsulada y es gestionada directamente por el entorno de ejecución (run-time). Esto significa que el código que ejecuta sus funciones está oculto para el desarrollador. Los SIFBs se utilizan comúnmente para interactuar con el hardware del sistema (como sensores, actuadores, buses de campo o redes industriales), actuando como una interfaz entre la lógica de control y el entorno físico [24].

3.4 Herramientas de desarrollo compatibles: EcoStruxure Automation Expert

El desarrollo de aplicaciones conforme a la norma IEC 61499 requiere herramientas específicas que interpreten correctamente su modelo de bloques de función distribuidos, permitan el diseño gráfico de aplicaciones y gestionen la asignación de recursos a distintos dispositivos físicos. A lo largo de los últimos años, varias plataformas han sido desarrolladas para dar soporte a este estándar, entre ellas destacan 4diac IDE, ISaGRAF, ** nxtControl** (actualmente descontinuado) y, más recientemente, EcoStruxure Automation Expert, una solución comercial avanzada de Schneider Electric.

EcoStruxure Automation Expert (EAE) es una plataforma industrial relativamente reciente orientada a IEC 61499 que ha sido lanzada como un producto comercial plenamente integrado dentro de un entorno de automatización de propósito general. Esta herramienta pasa del estándar definido en IEC 61131-3, desarrollada en LADDER, SFC y ST, a diseñar aplicaciones de control industrial distribuidas mediante un entorno gráfico basado en bloques de función activados por eventos, donde cada componente puede ser desplegado en diferentes dispositivos a través de una red industrial.

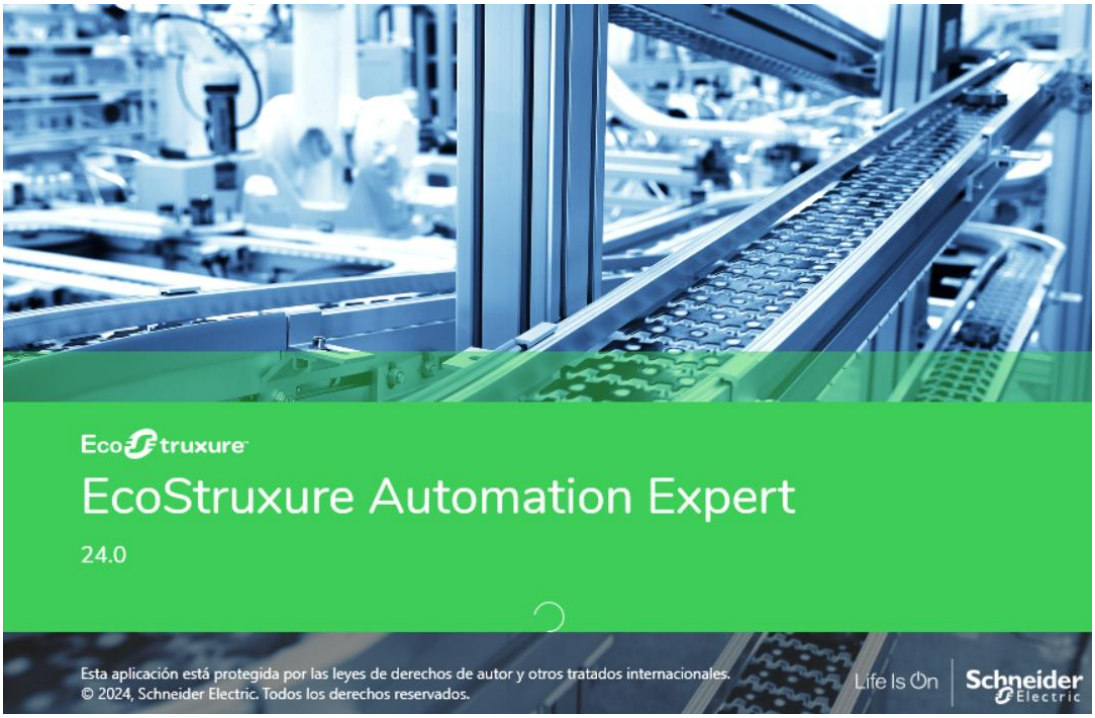

Figura 13. Ecostruxure Automation Expert

La herramienta implementa el concepto de automación universal (universal automation) o automatización centrada en software, con bloques reutilizables que pueden integrarse con arquitecturas IT/OT sin depender del hardware físico. Esta independencia permite reducir tiempos de desarrollo y aumentar la interoperabilidad entre componentes [25].

Además, EAE facilita la integración de múltiples tecnologías y estándares, permitiendo la interoperabilidad entre IEC 61499 e IEC 61131-3, lo que resulta especialmente útil en entornos de automatización híbridos. También incluye herramientas para diagnóstico, simulación, trazabilidad de variables y despliegue remoto, lo que reduce significativamente los tiempos de desarrollo y puesta en marcha.

EAE soporta una amplia gama de protocolos de comunicación industrial, integrando tanto estándares abiertos

como tecnologías propias del ecosistema Schneider Electric. Entre los protocolos compatibles se incluyen Modbus, MQTT, HTTP/HTTPS, EtherNet/IP y OPC UA, el cual será empleado en este proyecto como protocolo de comunicación y explicado en el siguiente sub-capítulo.

Gracias a su diseño modular y su soporte completo del modelo IEC 61499, EcoStruxure Automation Expert se posiciona como una solución alineada con los principios de la Industria 4.0, como la flexibilidad, la digitalización del ciclo de vida, el mantenimiento predictivo y la integración IT/OT. En consecuencia, se ha convertido en una de las plataformas líderes para la implementación práctica de soluciones distribuidas en el ámbito de la automatización industrial.

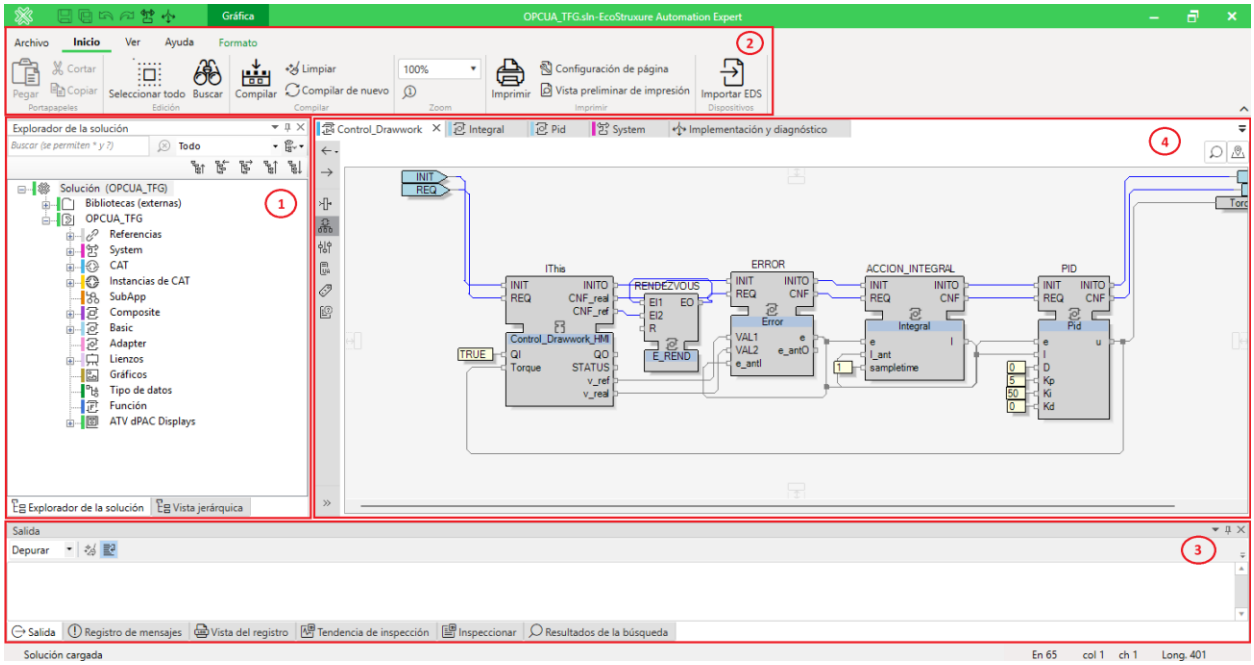

Figura 14. Interfaz EAE [26]

En la Figura 14, se muestra la interfaz de la aplicación con una solución de ejemplo con varios bloques funcionales. Dentro de la interfaz, se pueden distinguir 4 zonas señalizadas con números [26]:

- Figura 14,1. Se muestran de forma ramificada todos los elementos de la solución. En primer lugar, aparecen dentro de la carpeta 'Bibliotecas (externas)' todas las librerías añadidas a la solución, permitiendo añadir más librerías previamente instaladas si fuera necesario.
- Figura 14,2. En la parte superior del espacio de trabajo hay varios menús: 'Archivo', 'Inicio', 'Ver' y 'Ayuda'. En 'Archivo' nos encontramos las funcionalidades típicas de cualquier programa como son crear, cargar o guardar una solución, ajustes, bibliotecas, etc. En 'Inicio' tenemos funcionalidades de compilación, limpieza, etc. La pestaña 'Ver' permite localizar ventanas. Es especialmente útil para abrir las propiedades de un elemento o crear tendencias de inspección. Por último, en 'Ayuda' el usuario dispone de la documentación del programa y de las librerías instaladas.
- Figura 14,3. En la parte inferior hay una ventana donde aparecerán salidas de compilación y depuración. Si se cambia a la pestaña 'Registro de mensajes' se pueden buscar errores, advertencias o mensajes específicos
- Figura 14,4. Finalmente, en la parte central se encuentra el espacio de trabajo propiamente dicho. Es aquí donde se configuran y conectan los distintos bloques y diagramas.

3.4.1 Comunicación OPC UA

El estándar OPC (Open Platform Communications), anteriormente conocido como OLE for Process Control, surgió en 1996 con el objetivo de resolver los problemas de comunicación entre sistemas HMI y PLCs de

diferentes fabricantes en entornos de automatización industrial. En aquel momento, cada fabricante utilizaba su propio driver, lo que complicaba la integración. Para solucionar esto, se propuso una arquitectura cliente-servidor: el servidor gestionaba los distintos drivers de los PLCs, mientras que los HMIs actuaban como clientes, conectándose solo al servidor. Esta capa intermedia permitía centralizar la comunicación y facilitaba la escalabilidad del sistema, ya que el reemplazo de dispositivos no requería reconfigurar todo el sistema, sino simplemente actualizar los drivers en el servidor [27].

Este enfoque ayudó a estandarizar la comunicación industrial y resolvió muchos de los problemas de interoperabilidad entre dispositivos heterogéneos. En 2006, el protocolo evolucionó hacia una versión más avanzada, conocida como OPC Unified Architecture (OPC UA), dejando a la versión original bajo el nombre de OPC Classic [28].

La principal mejora de OPC UA frente a OPC Classic es su independencia del sistema operativo, ya que OPC Classic solo era compatible con Windows. Además, OPC UA introduce medidas de seguridad robustas, como el cifrado de extremo a extremo y el uso de certificados digitales mediante infraestructuras de clave pública (PKI), garantizando así la autenticación y la integridad de los datos. Otra mejora importante es el modelado de la información: mientras que OPC Classic se limitaba a transmitir datos simples, OPC UA permite definir modelos de información complejos con objetos, propiedades y relaciones, lo que lo convierte en una solución mucho más flexible y adaptada a las necesidades actuales de la industria.

Como se ha mencionado y se puede observar en la Figura 15, este protocolo se basa en la comunicación Cliente-Servidor, en la cual los clientes se conectan al servidor para solicitar los servicios que ofrece.

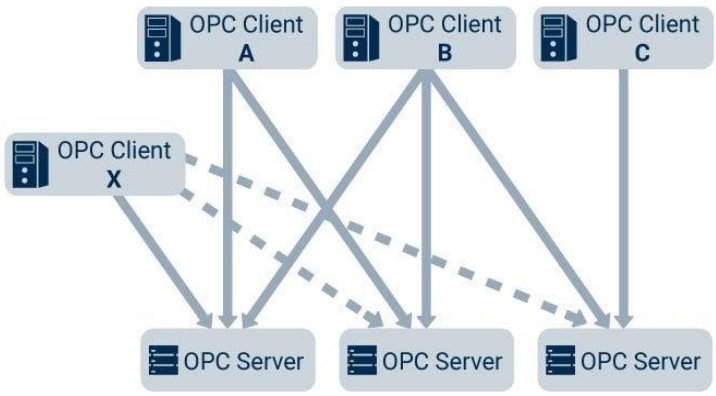

Figura 15. Elementos de la comunicación OPC UA

Una de las principales características de OPC UA es que se basa en una arquitectura orientada a servicios (SOA). A diferencia de OPC Classic, donde solo se exponían variables aisladas, en OPC UA lo que se ofrecen son servicios completos que permiten una interacción mucho más rica con los datos. Algunos ejemplos de estos servicios son la lectura y escritura de variables, la suscripción a cambios en variables (como notificaciones cuando un valor varía), o la exploración de los nodos del servidor.

En esta arquitectura, los datos no se gestionan de forma plana, sino que están organizados en nodos. Estos nodos son la unidad básica de comunicación dentro de un servidor OPC UA y pueden representar variables, objetos, métodos ejecutables, etc. Cada nodo pertenece a una clase concreta y está definido por una serie de atributos. Entre los atributos más relevantes se encuentran el 'NodeId', que actúa como identificador único dentro del servidor; el 'NodeClass', que indica la clase del nodo; y el 'BrowseName', que es el nombre utilizado para localizar el nodo al explorar el servidor [29].

Además, los nodos se relacionan entre sí mediante referencias (sin atributos) que establecen conexiones semánticas, permitiendo estructurar la información de forma jerárquica y coherente. También existe la opción de anidar nodos dentro de un mismo nodo, para poder agrupar fácilmente nodos que cumplan ciertas propiedades.

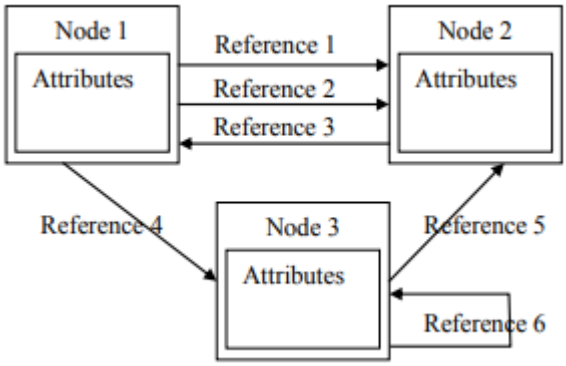

Figura 16. Ejemplo de nodos y referencias de nodos

Otra funcionalidad destacada de OPC UA es su capacidad para implementar un modelo de comunicación tipo Publicador-Suscriptor. En este esquema, un suscriptor puede recibir automáticamente notificaciones sobre eventos o cambios en variables específicas a las que está suscrito, como por ejemplo en la Figura 17; en la que el 'Publisher' (Publicador) toma los datos 'DataSets', los empaqueta en 'DataSetMessages', los agrupa en 'NetworkMessages' y los envía por un protocolo de transporte para que el 'Subscriber' (Suscripctor) escuche esos mensajes, los descomponga y acceda a los 'DataSets' que necesita.

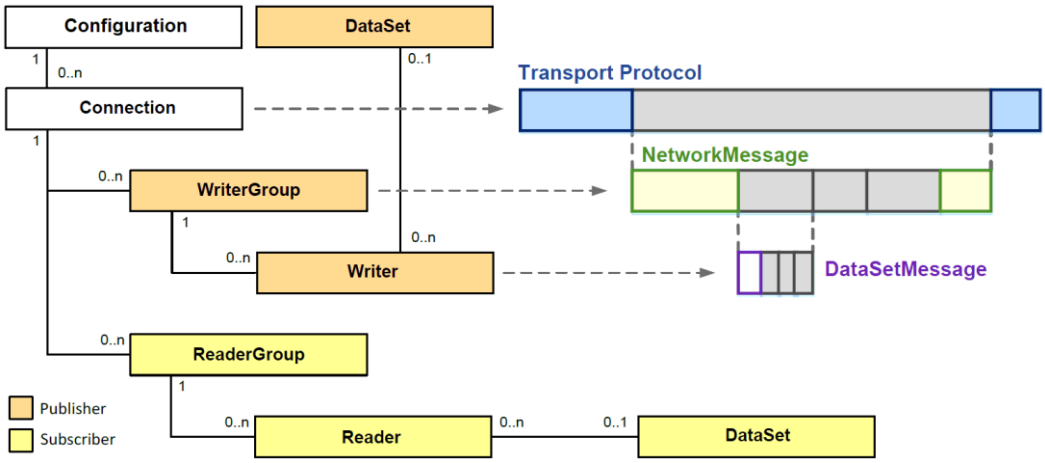

Figura 17. Ejemplo de comunicación Publicador-Suscriptor

En resumen, OPC UA presenta varias ventajas que resultan muy útiles para aplicaciones en tiempo real o sistemas distribuidos frente a otros protocolos de comunicación [28]:

- **Descentralización**: se pasa de la antigua estructura piramidal de comunicación, en la que los dispositivos de bajo nivel necesitaban comunicarse con dispositivos de medio nivel para llegar hasta los de alto nivel, a una red de malla donde la comunicación entre el nivel bajo y alto es directa.
- **Independencia de la plataforma**: como se dijo anteriormente, OPC UA permite el uso de cualquier sistema operativo, pudiendo incluso desplegarse en la nube.
- **Escalabilidad**: su arquitectura orientada a servicios y su modelo de información dinámico facilita la integración de nuevos equipos y sistemas sin necesidad de reconfigurar toda la red. Esto permite que una planta industrial pueda crecer o adaptarse fácilmente.
- **Interoperabilidad**: permite la comunicación entre dispositivos de distintos fabricantes.

3.4.2 CAT e instancias CAT

Los bloques CAT (Component Automation Type) son componentes reutilizables que encapsulan otros bloques funcionales, ya sean Basic Function Blocks, Composite Function Blocks, u otros bloques CAT. Su función es similar a la de los Composite Function Blocks, pero con un enfoque más marcado hacia la orientación a objetos y la interacción con el operador, ya que están diseñados para integrarse fácilmente con interfaces HMI o sistemas SCADA [26].

Cabe aclarar que los CAT no son parte de la norma IEC 61499, sino que son una extensión específica de EAE.

Cuando se crea un bloque CAT, se genera automáticamente un bloque asociado llamado 'IThis', que define la interfaz del componente con el HMI. Este contiene las variables de entrada y salida del CAT que podrán ser representadas visualmente en la interfaz y modificadas por el operario. Gracias a esto, los bloques CAT son ideales para modelar comportamientos físicos o lógicos específicos, como válvulas, motores, transportadores o sensores, que requieren supervisión o actuación directa desde el entorno de control. Es necesario saber que toda variable que quiera ser expuesta a la comunicación OPC UA debe estar creada en este bloque [26].

Además, por cada bloque CAT que se diseñe, se pueden crear múltiples instancias, llamadas 'Instancias de CAT'. Estas instancias son las que se arrastran a los lienzos de diseño de aplicaciones y que, posteriormente, se mapean a dispositivos físicos dentro del sistema.

En resumen, un bloque CAT estará compuesto por un bloque IThis que conectará con el HMI, integrado en una red de bloques que forman una lógica secuencia.

4 CARACTERÍSTICAS DEL SISTEMA

En este capítulo se describirá el funcionamiento del sistema a controlar, así como sus límites y especificaciones.

Se dispone del modelo de Simulink de un electrolizador PEM, modelo Hamilton-STD SPE-HG de 1 kW, integrado en la microrred experimental HyLab de la Universidad de Sevilla [30]. Este tipo de tecnología ha sido seleccionada por su rápida respuesta dinámica, alta densidad de corriente, y su idoneidad para su integración con fuentes renovables intermitentes.

Este sistema está compuesto por una pila de celdas PEM montadas en serie, y un conjunto auxiliar (Balance of Plant, BoP) que incluye elementos para el acondicionamiento del agua, separación de fases, control de presión, alimentación eléctrica, sensores, sistema de control, y sistema de refrigeración por ventilador, el cual resulta esencial para el control térmico.

Este sistema funciona con energía solar, es decir, mediante el efecto fotovoltaico la energía de la radiación solar se convierte en energía eléctrica, por lo que la intensidad de entrada irá variando en el tiempo en función esta radiación.

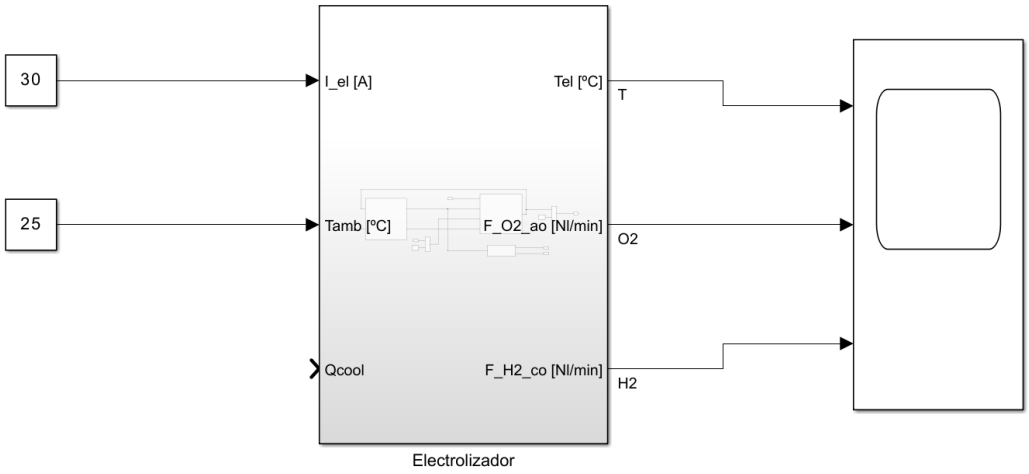

Figura 18. Modelo Simulink del sistema

El sistema cuenta varias entradas: '*Qcoo*l', que será la variable a controlar; y dos perturbaciones, intensidad de entrada en Amperios '*I_el*' y temperatura ambiente en ºC '*Tamb*'. *Qcoo*l está relacionado con la potencia suministrada al ventilador del electrolizador en vatios, es decir, cuanto mayor sea, más rápido girarán las aspas del ventilador, y por tanto, más enfriará.

Las salidas son: la concentración de oxígeno en Nl/min '*F_O2_ao*', la concentración de hidrógeno en Nl/min '*F_H2_ao*' y la temperatura del electrolizador en ºC '*Tel*'. Este último dato es el que nos interesa para realizar el control de la temperatura.

El sistema se compone de 2 sub-sistemas: modelo electroquímico (Fig. 19,1) y modelo térmico (Fig. 19,2).

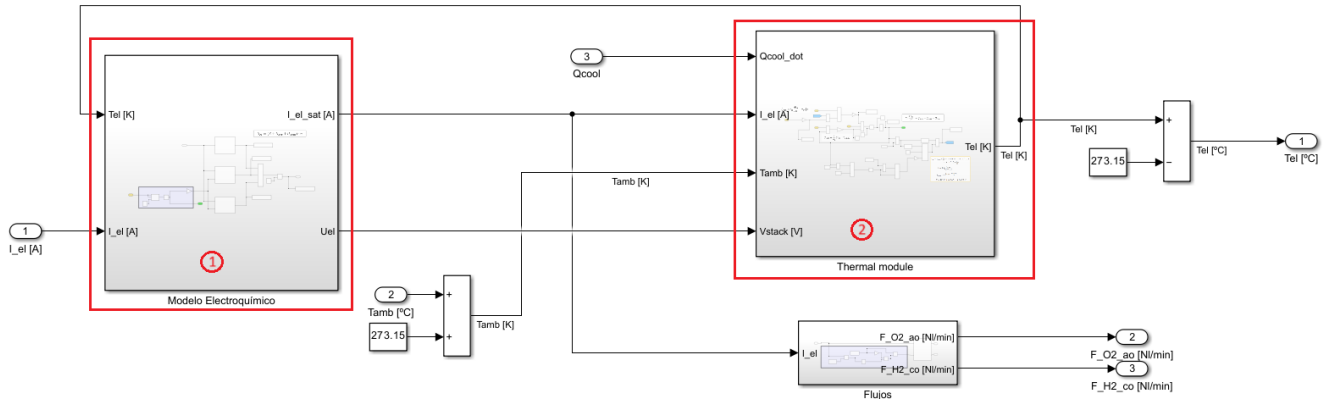

Figura 19. Modelo electroquímico y modelo térmico

4.1 Especificaciones técnicas

Las principales características técnicas del electrolizador se resumen en la siguiente tabla [30] (Tab. 3):

Parámetro	Valor
Potencia máxima	1 kW
Caudal de hidrógeno	0.23 Nm³/h @ 1 kW
Presión parcial H_2 / O_2	6.9 bar / 1.3 bar
Voltaje / corriente de operación	9–11 V / 5–80 A
Número de celdas en serie	6
Temperatura nominal	60 ºC
Área de la pila	212.35 cm²
Densidad de corriente de intercambio (ánodo / cátodo)	$1.06 \cdot 10^{-6}$ / $1 \cdot 10^{-3}$ A/cm²
Espesor de membrana	178 µm (Nafion 117)

Tabla 3. Especificaciones técnicas del electrolizador

Se asumen simplificaciones para mantener la utilidad del modelo en aplicaciones de control, como gases ideales, homogeneidad de temperatura en la pila, y desprecio de efectos menores (como la difusión a alta corriente).

4.2 Modelo electroquímico

El modelo electroquímico tiene como finalidad describir la relación entre la corriente de entrada y el voltaje del stack del electrolizador PEM, teniendo en cuenta la temperatura de operación y las presiones de hidrógeno y oxígeno. Esta relación se conoce como curva de polarización, y permite analizar la eficiencia del proceso y las

pérdidas por sobrepotenciales.

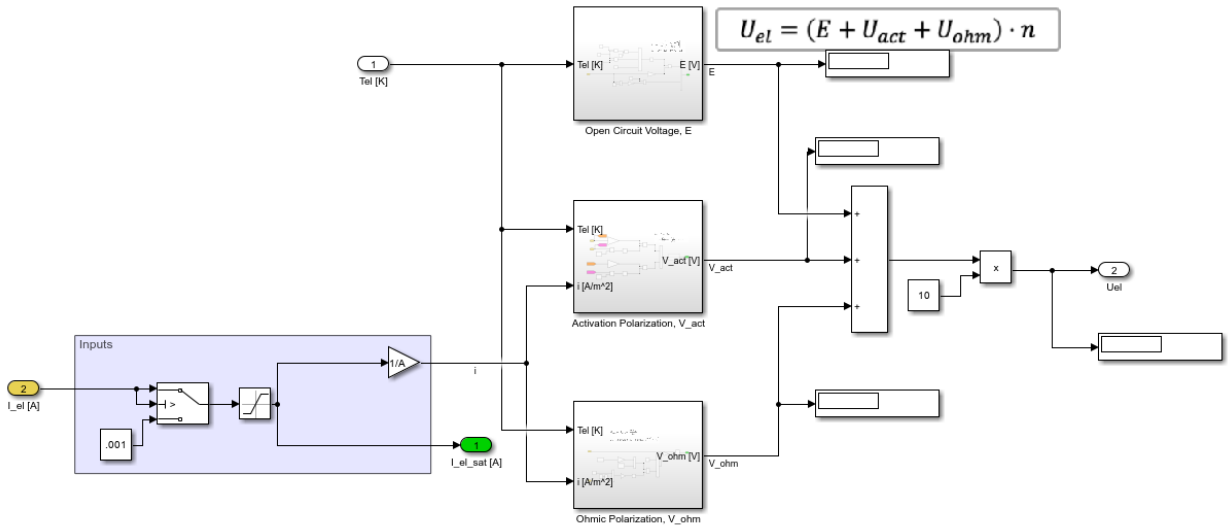

Figura 20. Modelo electroquímico

Se adoptan las siguientes hipótesis:

1. La difusión se desprecia por trabajar con densidades de corriente bajas.
2. La conductividad de la membrana depende solo de la temperatura.
3. Presión constante (no se considera dependencia de la presión en los sobrepotenciales).
4. Membrana totalmente hidratada, con un valor de σ_m fijo.
5. Las celdas son idénticas y conectadas en serie, compartiendo comportamiento térmico y eléctrico.

La tensión total en el stack requerida para la reacción química se calcula como:

$$\boldsymbol{Vel} = (E + Vact + Vohm + Vdiff) \cdot n$$

Ecuación 10. Cálculo del voltaje en el stack

Donde:

- *E* es la tensión de circuito abierto (reversible), obtenida con la ecuación de Nernst (Ec. 11).
- *Vact* es el sobrepotencial de activación.
- *Vohm* es el sobrepotencial óhmico.
- *Vdiff* es obrepotencial por difusión (se desprecia en este modelo por operar a densidades < 3 A/cm²).

Estos parámetros se calculan dentro de sus respectivos sub-modelos.

4.2.1 Cálculo de la tensión de circuito abierto

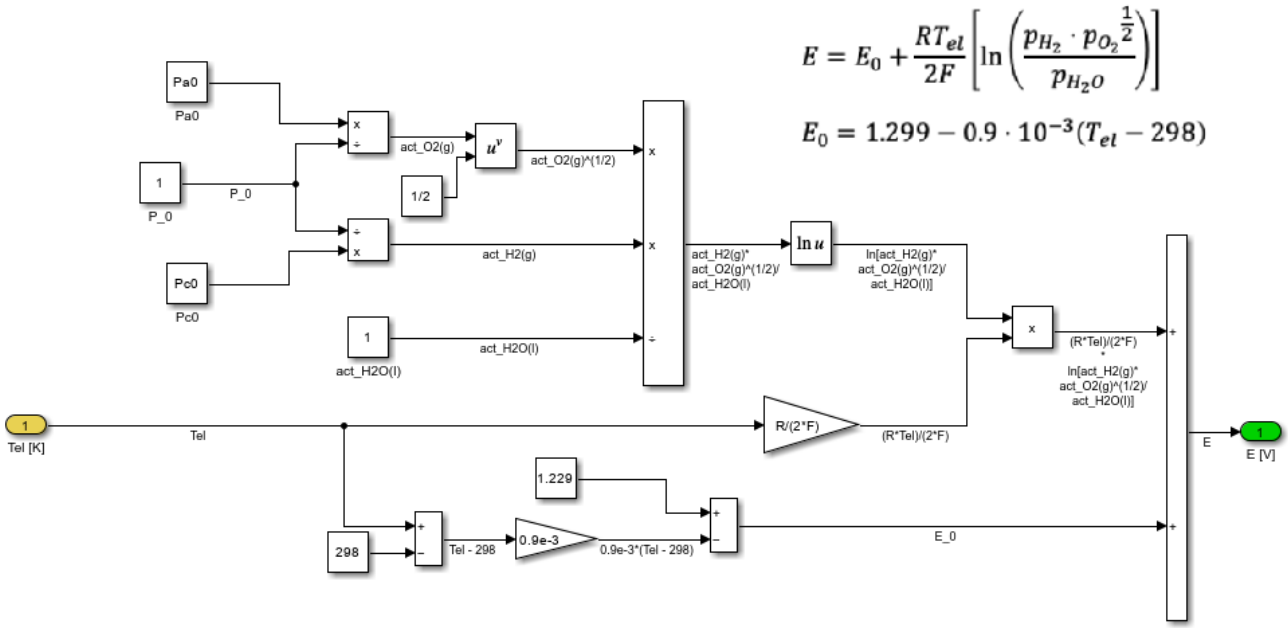

Figura 21. Sub-modelo para el cálculo de *E*

La tensión de circuito abierto se obtiene de la ecuación de Nernst:

$$\boldsymbol{E} = E_0 + \frac{R \cdot Tel}{2F}\left[\ln\left(\frac{a_{H_2(g)}\, a_{O_2(g)}^{\frac{1}{2}}}{a_{H_2O(g)}}\right)\right]$$

Ecuación 11. Ecuación de Nernst

Donde:

- E_0: voltaje estándar (≈1.229 V a 25 °C).
- R: constate universal de los gases (8.314 J/(mol·K)).
- Tel: temperatura del electrolizador en k.
- F: constante de Faraday (96.485 C/mol).
- pi: presión parcial de cada componente.

E_0 viene dado por la siguiente ecuación:

$$E_0 = 1.299 - 0.9 \cdot 10^{-3}\,(Tel - 298)$$

Ecuación 12. Cálculo del voltaje estándar

4.2.2 Cálculo del sobrepotencial de actuación

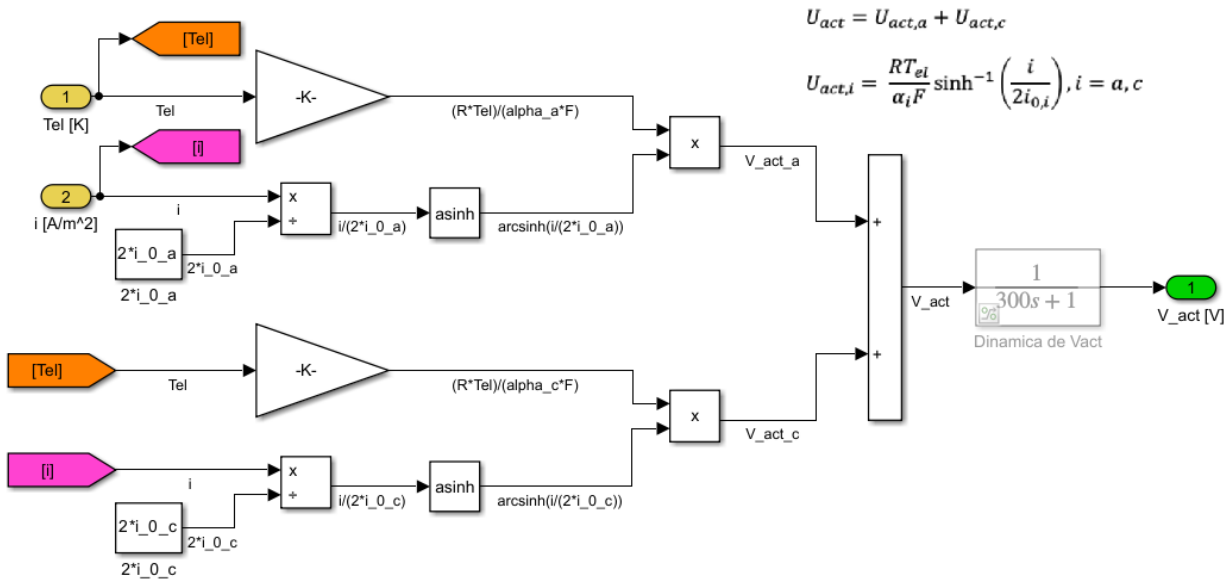

Figura 22. Sub-modelo para el cálculo de *Vact*

El sobrepotencial de activación se calcula como:

$$Vact,i = \frac{R \cdot Tel}{\alpha_i F} \, sinh^{-1}\left(\frac{i}{2i_{0,i}}\right) \, , i = a, c$$

$$\boldsymbol{Vact} = Vact,a + Vact, c$$

Ecuación 13. Cálculo del sobrepotencial de activación

Donde:

- i: densidad de corriente (A/cm²).
- i_0 : densidad de corriente de intercambio (anódica/catódica).
- α: coeficiente de transferencia de carga.

4.2.3 Cálculo del sobrepotencial óhmico

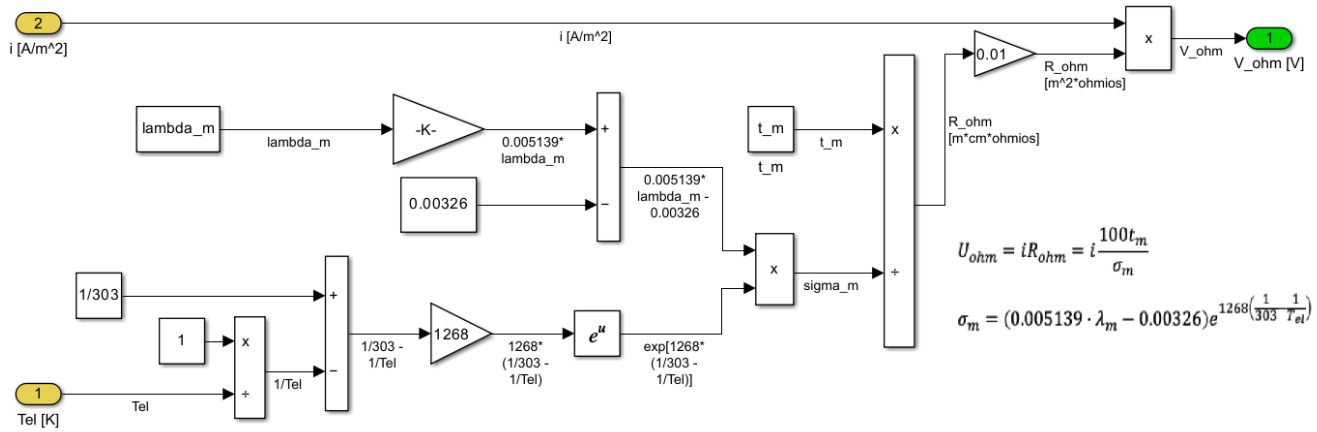

Figura 23. Sub-modelo para el cálculo de *Vohm*

El sobrepotencial óhmico se debe a diferentes tipos de resistencias, en particular la de membrana (iónica) y la eléctrica. Para su cálculo, se utiliza la ley de Ohm, como se indica en la ecuación 14.

$$\boldsymbol{Vohm} = i \cdot Rohm = i\,(Relec + Rmem)$$

Ecuación 14. Ley de Ohm

Sin considerar la resistencia eléctrica, la resistencia óhmica, Rohm ($\Omega \cdot cm^2$), se calcula según la ecuación 15. Para obtenerla, es necesario calcular previamente el valor de la conductividad de la membrana σm ($\Omega^{-1}\ cm^{-1}$), según la ecuación 16.

$$Rohm = \frac{100\ t_m}{\sigma_m}$$

Ecuación 15. Cálculo de la resistencia óhmica

$$\sigma_m = (0.005139\ \lambda_m - 0.00326)\ e^{1268\left(\frac{1}{303} - \frac{1}{Tel}\right)}$$

Ecuación 16. Cálculo de la conductividad de la membrana

Donde:

- t_m: espesor de la membrana (m).
- λ_m: contenido de agua en la membrana ((7-14) H_2O/mol SO_3^-)
- *i*: densidad de corriente del electrolizador, definida como la corriente entre el área de la celda (*i=Iel/Ael* (A/cm^2))

4.3 Modelo térmico

El modelo térmico tiene como propósito describir la evolución dinámica de la temperatura del stack del

electrolizador en función de las condiciones de operación, especialmente la potencia de entrada, la temperatura ambiente y la refrigeración aplicada. Dado que la dinámica térmica es más lenta que la electroquímica, este modelo resulta esencial para el diseño de estrategias de control de temperatura, especialmente bajo condiciones variables típicas de fuentes renovables.

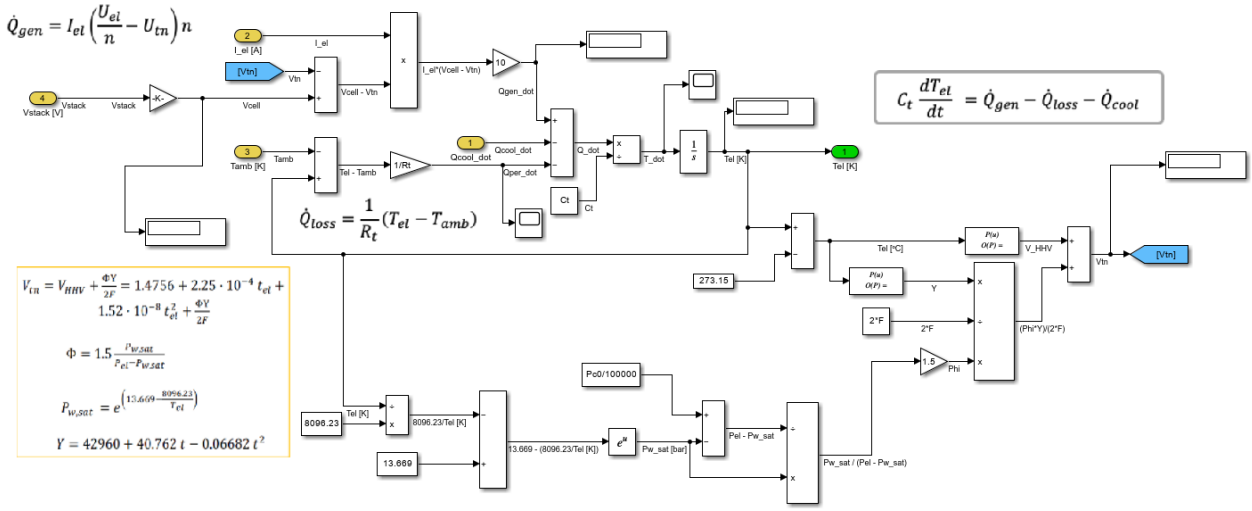

Figura 24. Modelo térmico

Se adoptan las siguientes hipótesis:

1. Modelo térmico concentrado: temperatura homogénea en todo el stack.
2. Flujos de salida (agua, H_2, O_2) no contribuyen térmicamente.
3. Temperatura ambiente constante durante el experiment.
4. Efecto Joule directo despreciado (incluido ya en $\dot{Q}_{gen}$).
5. El sistema de refrigeración actúa solo si se supera la temperatura nominal.
6. La interacción térmica con el entorno se da solo por convección y radiación.

Con estas consideraciones, el modelo se basa en un balance energético simplificado, expresado como una ecuación diferencial de primer orden:

$$C_t \frac{dT_{ez}}{dt} = \dot{Q}_{gen} + \dot{Q}_{loss} + \dot{Q}_{cool}$$

Ecuación 17. Balance térmico en el electrolizador

Donde:

- C_t: capacidad térmica del stack (J/K).
- $\dot{Q}_{gen}$: calor generado por pérdidas electroquímicas (W). Se obtiene de la ecuación 18.
- $\dot{Q}_{loss}$: pérdidas térmicas al ambiente (W). Se obtiene de la ecuación 23.
- $\dot{Q}_{cool}$: calor extraído por el sistema de refrigeración (W). Variable a controlar.

4.3.1 Cálculo del calor generado

El calor se produce por las irreversibilidades del proceso:

$$\dot{Q}_{gen} = I_{el}\,(V_{cell} - V_{tn})\,n$$

Ecuación 18. Cálculo del calor generado

Donde:

- *Vcell*: voltaje en la celda ($= V_{el}/n$, con V_{el} obtenido del proceso electroquímico).
- V_{th}: voltaje termoneutral. Obtenido de la ecuación 19.
- La corriente se considera una perturbación rápida y medible.

$$V_{th} = V_{HHV} + \frac{\phi Y}{2F} = 1.4756 + 2.25 \cdot 10^{-4}\, t_{el} + 1.52 \cdot 10^{-8}\, t_{el}^2 + \frac{\phi Y}{2F}$$

Ecuación 19. Cálculo del voltaje termoneutral

Donde:

- V_{HHV}: '*Voltaje de mayor valor alto*'.
- t_{el}: temperatura del electrolizar en ºC (T_{el} en Kelvin).
- ϕ: coeficiente adimensional (Ec. 20).
- $P_{w,sat}$: presión parcial del vapor de agua (bar) (Ec. 21).
- Y: coeficiente con unidades de J/mol (Ec. 22).

$$\phi = 1.5\,\frac{P_{w,sat}}{P_{el} - P_{w,sat}}$$

Ecuación 20. Cálculo del coeficiente adimensional

$$P_{w,sat} = e^{\left(13.669 - \frac{8096.23}{T_{el}}\right)}$$

Ecuación 21. Cálculo de la presión parcial del vapor de agua

$$Y = 42960 + 40.762\,t - 0.06682\,t^2$$

Ecuación 22. Cálculo del coeficiente *Y*

4.3.2 Cálculo de las pérdidas térmicas en el ambiente

Se modelan como un flujo térmico proporcional a la diferencia de temperaturas:

$$\dot{Q}_{loss} = \frac{1}{R_t}(T_{el} - T_{amb})$$

Ecuación 23. Cálculo de las pérdidas térmicas en el ambiente

Donde:

- T_{amb}: temperatura ambiente en Kelvin, considerada una perturbación lenta y medible.
- R_t: resistencia térmica que modela el intercambio de calor por radiación y convección (K/W).

4.3.3 Cálculo del calor extraído por el sistema de refrigeración (*Qcool*)

Finalmente, si se alcanza la tensión nominal, el sistema de refrigeración elimina calor:

$$\dot{Q}_{cool} = \dot{Q}_{gen} - \dot{Q}_{loss}$$

Ecuación 24. Cálculo del calor extraído por el sistema de refrigeración

Como se ha mencionado al principio del capítulo $\dot{Q}_{cool} = Q_{cool}$ es la variable de control, que depende de la velocidad del ventilador y flujo refrigerante. Cuanto mayor sea, más calor es extraído.

Q_{cool} está determinado por la capacidad máxima del sistema de refrigeración. Para el electrolizador Hamilton-STD SPE-HG (1 kW), basado en la validación experimental, el modelo opera en un rango donde $\dot{Q}_{gen}$ ≈100−300W, por lo que $Q_{cool,máx}$ debe al menos igualar $\dot{Q}_{gen} - \dot{Q}_{loss}$ para mantener T constante. Entonces, $Q_{cool,máx} \approx 300$ W es un buen límite de diseño estimado.

Por otro lado, el sistema de refrigeración solo es capaz de enfriar el electrolizador, por lo que una Q_{cool} negativa carecería de sentido. Es por ello que se el limite inferior se establece en $Q_{cool,mín} = 0$ W.

De esta manera, Q_{cool} queda limitado en un rango de **[0 - 300] W**.

La temperatura ideal de operación de los electrolizadores PEM se encuentra generalmente entre los **50** y **70** ºC. Rebajar este límite inferior podría causar una saturación en Q_{cool} debido a la capacidad máxima de refrigeración o disminuir la eficiencia del sistema; mientras que superar el límite superior puede acelerar la degradación de la membrana PEM y otros componentes del electrolizador, disminuyendo su vida útil. Es por ello, que a la hora de controlar la temperatura del electrolizador, se buscará que esta se mantenga dentro de estos límites, a pesar de que en las primeras simulaciones se fuerce al sistema a operar en temperaturas menores para realizar un control más robusto ante cambios bruscos en la referencia.

5 DESARROLLO E IMPLEMENTACIÓN

En este capítulo se llevará a cabo tanto la conexión entre ambas aplicaciones, Matlab y EAE; como el control de la temperatura del sistema.

Estos procesos se llevarán a cabo en varios sub-capítulos siguiendo el siguiente orden: Se realizará la conexión entre Matlab y un programa de ejemplo de EAE, se controlará la temperatura del electrolizador desde Matlab; y finalmente se adaptará la conexión al sistema del electrolizador de Simulink y se controlará desde EAE.

5.1 Prueba de conexión OPC UA con Matlab

En primer lugar, como propone Javier Prieto en su proyecto [26], el cual nos servirá de guía, se realizará la conexión entre Matlab y un programa de prueba de EAE mediante el protocolo OPC UA.

Como se dijo en el tercer capítulo, el protocolo OPC UA se basa en la comunicación entre servidor y cliente. Tanto EAE como Matlab pueden tomar ambos papeles. En este proyecto, se ha decidido que EAE actúe como servidor y Matlab se conecte a él como cliente.

La comunicación se realizará con el PLC virtual predeterminado de EAE 'EcoRT_0', el cual deberá tener activada la opción *Servidor OPCUA > Habilitar servidor* (Fig. 25).

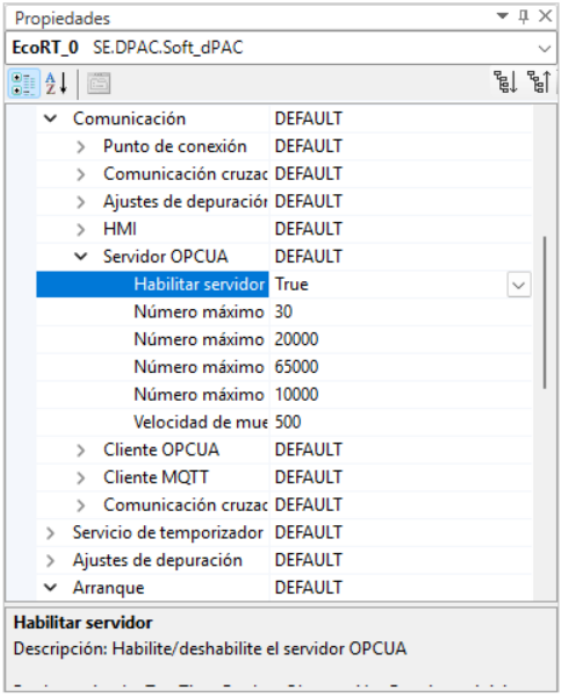

Figura 25. Propiedades del servidor OPC UA

La lógica del programa consiste en tomar dos valores ‘OUT1’ y ‘OUT2’ y realizar el producto de ambos. El resultado se guarda en ‘res’. El objetivo es compartir las variables ‘OUT1’ y ‘OUT2’ para que un cliente les de valores, y posteriormente exponer la variable ‘res’ para que el cliente conozca el resultado. Es importante recordar que toda variable que quiera ser expuesta debe estar creada en un bloque IThis, explicado en el tercer capítulo. También hay que tener en cuenta que las variables de entrada al IThis permiten solo lectura y las de salida permiten tanto lectura como escritura [26].

Por lo tanto, ‘OUT1’ y ‘OUT2’ deben ser variables de salida, de tal forma que el cliente pueda escribir sobre ellas, y ‘res’ debe ser de entrada, para que el cliente solo pueda leerla y no altere el resultado por error.

Es necesario marcar la casilla *Expuesto* de cada variable, en la pestaña *Servidor OPCUA*, para poder compartirlas; dando acceso de lectura a las entradas y acceso de escritura a la salida.

En la Figura 26 se muestra el bloque CAT llamado ‘prueba’.

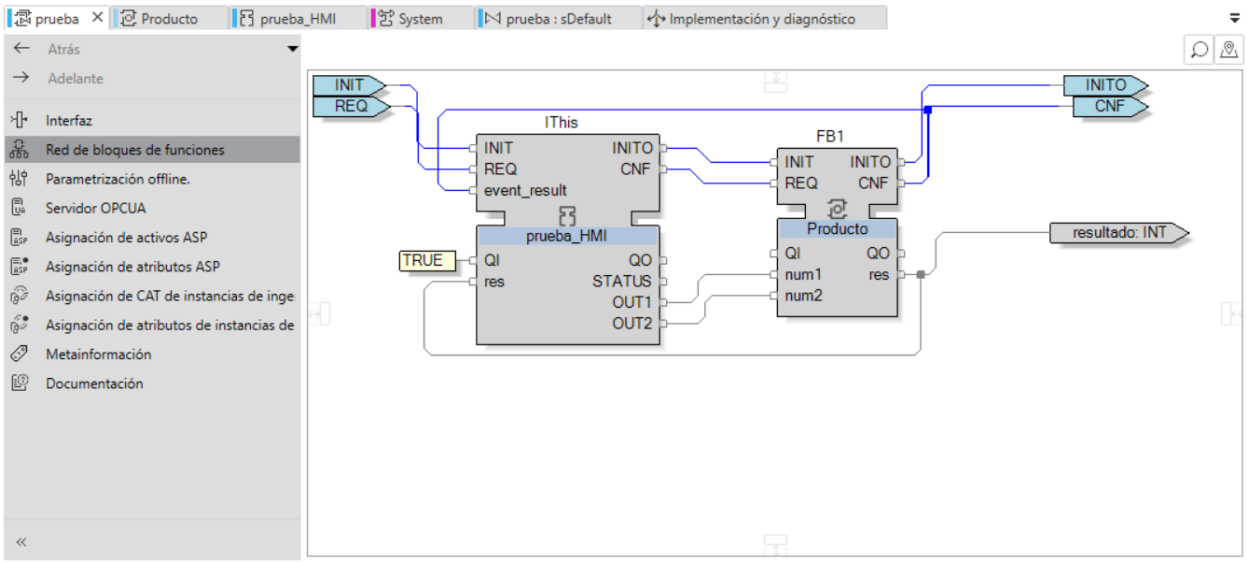

Figura 26. Red de bloques del CAT ‘prueba’

El BFB ‘Producto’ recibe ‘OUT1’ y ‘OUT2’, realiza su producto y devuelve ‘res’ al bloque IThis. Por tanto, su configuración será la siguiente:

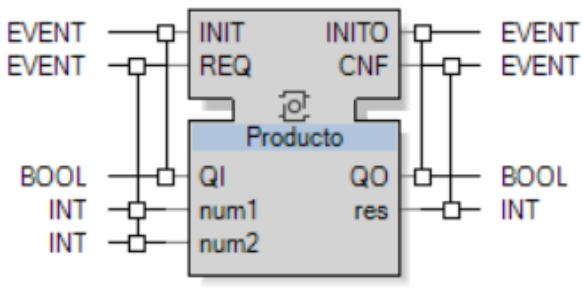

Figura 27. Datos y eventos del bloque ‘Producto’

```
ALGORITHM REQ IN ST:
res := num1 * num2;
END_ALGORITHM
```

Código 1. Algoritmo del bloque 'Producto'

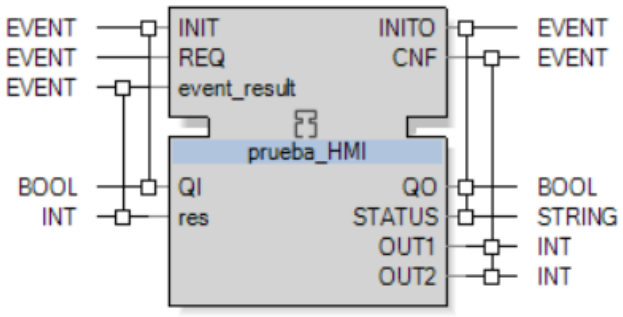

Figura 28. Datos y eventos del bloque 'prueba_HMI'

Hay bloques que necesitan ser inicializados internamente mediante los eventos 'INIT' e 'INITO', como es el caso de los bloques IThis. Por ello, se necesita un bloque auxiliar que empiece la cadena de inicialización. Este se llama 'plcStart'. Este bloque se conecta al CAT 'prueba' como se muestra en la Figura 29.

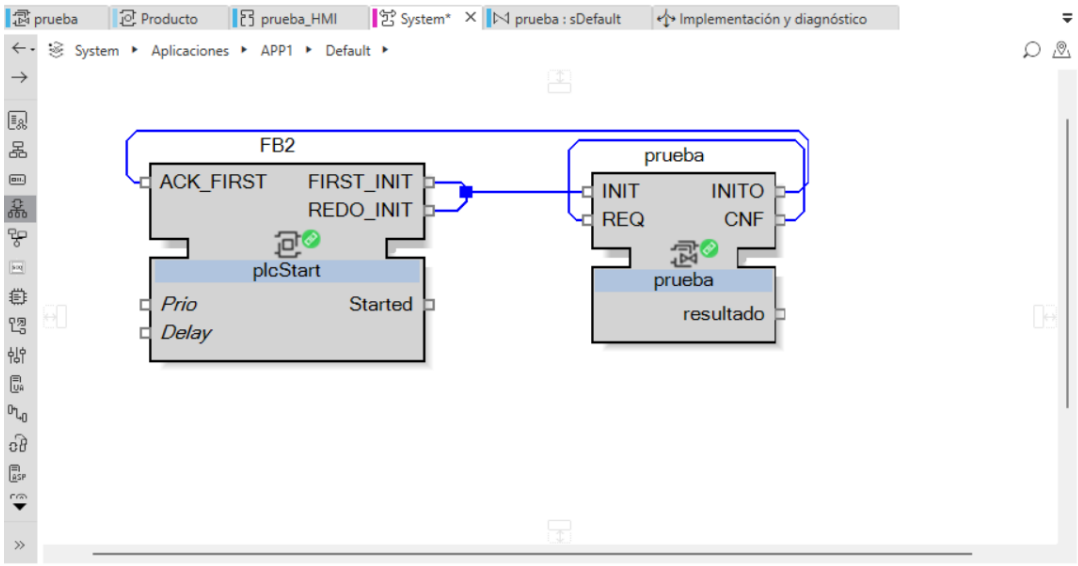

Figura 29. Red de bloques de la aplicación

Una vez creada la aplicación, es necesario mapear los bloques al recurso.

Los recursos deben tener siempre dos bloques fundamentales: 'E_RESTART' y 'DPAC_FULLINIT'. El primero genera los eventos de arranque del recurso y el segundo es un bloque de funciones compuestas que se utiliza para facilitar la secuencia de inicialización de los CAT del hardware y la aplicación.

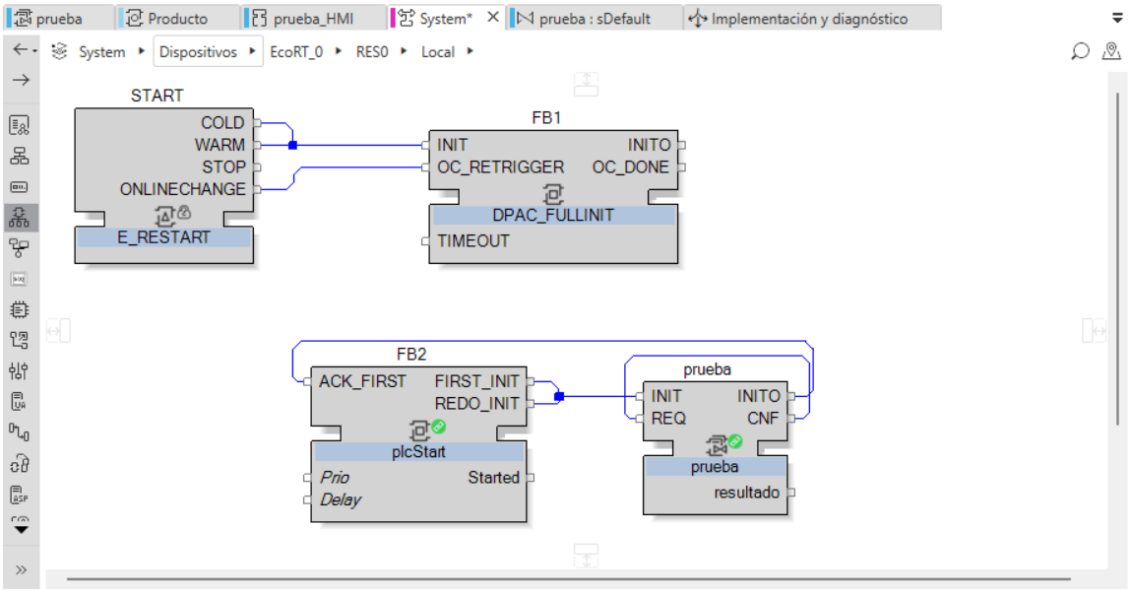

Figura 30. Red de bloques del recurso

Para poder ejecutar el programa y que Matlab pueda conectarse al servidor es necesario acceder a la ventana de 'Implementación y diagnóstico' y compilar, iniciar sesión en el dispositivo, implementar, establecer el proyecto de arranque y ejecutar. Se puede comprobar que los pasos se han realizado correctamente cuando los iconos del campo 'Estado' se encuentren todos de color verde (Fig. 31).

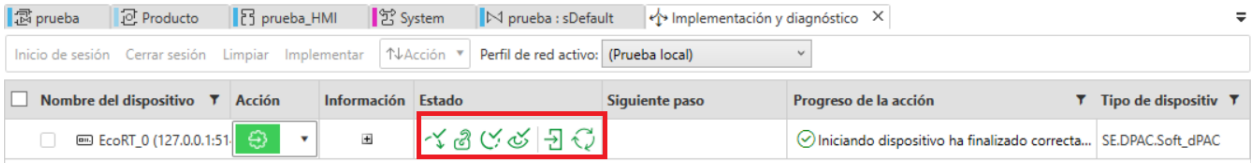

Figura 31. Implementación y diagnóstico

Por otro lado, en Matlab se ejecutará el siguiente script (Cód. 2), haciendo uso del Add-On 'Industrial Communication Toolbox', para poder leer y escribir mediante comunicación OPC UA.

```
% 1. Búsqueda del servidor
serverName = 'EcoStruxure';
serverList = opcuaserverinfo('localhost') % también se puede usar la URL
hsInfo = findDescription(serverList, serverName);

% 2. Crear cliente y conectar
uaClient = opcua(hsInfo)
connect(uaClient);
uaClient

% 3. Buscar nodo 'OUT1' navegando jerárquicamente
nodePath = {'prueba', 'IThis', 'OUT1'};
nodoActual = uaClient.Namespace;
for i = 1:length(nodePath)
    nodoActual = findNodeByName(nodoActual, nodePath{i});
    if isempty(nodoActual)
        error("No se encontró el nodo '%s' en la ruta.", nodePath{i});
    end
end
```

```
% 4. Escribir sobre la variable ‘OUT1’
factor1 = 5;
writeValue(uaClient, nodoActual, factor1);

% 5. Buscar nodo ‘OUT2’ navegando jerárquicamente
nodePath = {'prueba', 'IThis', 'OUT2'};
nodoActual = uaClient.Namespace;
for i = 1:length(nodePath)
    nodoActual = findNodeByName(nodoActual, nodePath{i});
    if isempty(nodoActual)
        error("No se encontró el nodo '%s' en la ruta.", nodePath{i});
    end
end

% 6. Escribir sobre la variable ‘OUT2’
factor2 = 6;
writeValue(uaClient, nodoActual, factor2);

% 7. Buscar nodo ‘res’ navegando jerárquicamente
nodePath = {'prueba', 'IThis', 'res'};
nodoActual = uaClient.Namespace;
for i = 1:length(nodePath)
    nodoActual = findNodeByName(nodoActual, nodePath{i});
    if isempty(nodoActual)
        error("No se encontró el nodo '%s' en la ruta.", nodePath{i});
    end
end

% 8. Leer la variable ‘res’
Resultado = readValue(uaClient, nodoActual)

% 9. Desconectar
disconnect(uaClient);
```

Código 2. Script de Matlab para comunicación OPC UA [26]

Este programa puede encontrar el servidor de EAE fácilmente ya que se encuentra ejecutándose en el mismo PC ‘localhost’. Luego crea el cliente OPC UA y consigue acceder a los datos navengando jerárquicamente a través de los nodos, para finalmente darles un valor o leerlos.

Al ejecutarlo tenemos varios mensajes en la ventana de comandos de Matlab:

```
serverList =

OPC UA ServerInfo 'EcoStruxure Runtime':

   Connection Information:
              Hostname: 'PORTATIL-SAMUEL'
                  Port: 4840
             Endpoints: [1×1 opc.ua.EndpointDescription]

   Security Information:
    BestMessageSecurity: None
    BestChannelSecurity: None
         UserTokenTypes: {'Anonymous'}
```

Figura 32. Lista de servidores encontrados por Matlab

```
uaClient =

OPC UA Client:

   Server Information:
                    Name: 'EcoStruxure Runtime'
                Hostname: 'PORTATIL-SAMUEL'
                    Port: 4840
             EndpointUrl: 'opc.tcp://PORTATIL-SAMUEL:4840'

   Connection Information:
                 Timeout: 10
                  Status: 'Disconnected'
             ServerState: '<Not connected>'

   Security Information:
      MessageSecurityMode: None
    ChannelSecurityPolicy: None
                Endpoints: [1×1 opc.ua.EndpointDescription]
```

Figura 33. Propiedades del cliente OPC UA (desconectado)

Al principio el cliente aparecerá como desconectado (Fig. 33). Para conectarlo se usa el comando 'connect(Uaclient)' (Cód. 2, línea 8).

```
OPC UA Client:

   Server Information:
                        Name: 'EcoStruxure Runtime'
                    Hostname: 'PORTATIL-SAMUEL'
                        Port: 4840
                 EndpointUrl: 'opc.tcp://PORTATIL-SAMUEL:4840'

   Connection Information:
                     Timeout: 10
                      Status: 'Connected'
                 ServerState: 'Running'

   Security Information:
          MessageSecurityMode: None
       ChannelSecurityPolicy: None
                   Endpoints: [1×1 opc.ua.EndpointDescription]

   Server Limits:
              MinSampleRate: 0.0001 sec
               MaxReadNodes: 0
              MaxWriteNodes: 0
         MaxHistoryReadNodes: 0
     MaxHistoryValuesPerNode: 0
```

Figura 34. Propiedades del cliente OPC UA (conectado)

Tras ese comando, podemos ver que el cliente esta conectado y el servidor de EAE se encuentra activo. También se guarda el sample time mínimo de EAE, que es 0.1 ms.

Después de dar valores a 'OUT1'=5 y 'OUT2'=6 y que EAE realice el producto, se comprueba que se ha realizado correctamente la operación leyendo el dato 'res' que se guarda en 'Resultado' (Cód. 2, línea 50) (Fig. 35).

```
Resultado =

  int16

    30
```

Figura 35. Resultado del producto

Finalmente, para comprobar el correcto funcionamiento de la conexión, en EAE se puede usar la función de 'Inspección' para ver en tiempo real el valor de las entradas y salidas de los bloques, así como el número de eventos producidos (Fig. 36).

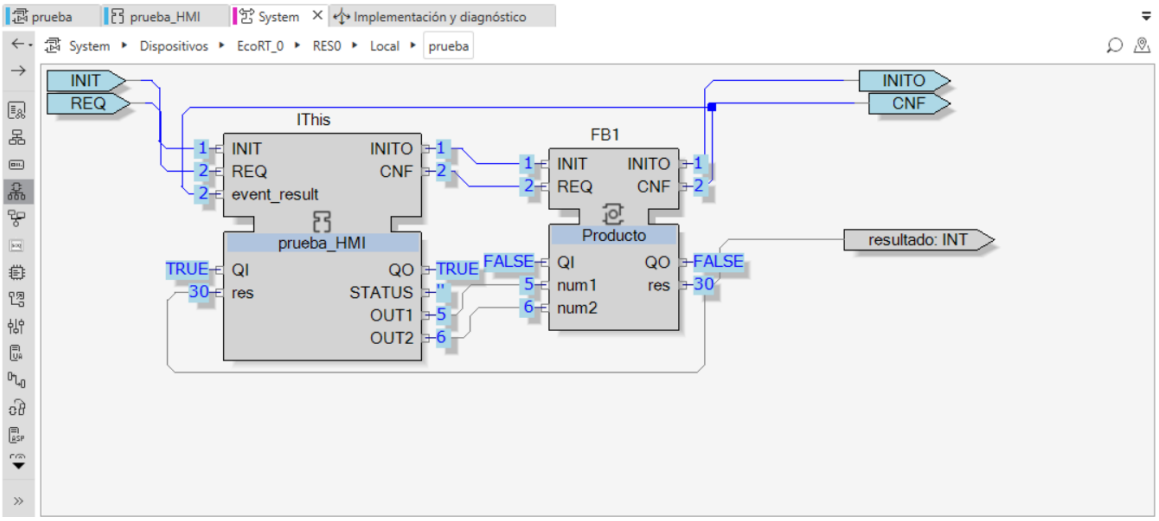

Figura 36. Resultado de la lógica de ejemplo EAE

5.2 Control de la temperatura desde Matlab

Para controlar la temperatura se empleará una técnica clásica de control como es el control PID. El control PID se basa en la siguiente ecuación:

$$u = (kp \cdot e) + (ki \cdot I) + (kd \cdot D)$$

Ecuación 25. Ecuación de control PID

Donde:

- *u* es la señal de control. En este caso corresponde a *Qcool*.
- *e* es el error. Es la diferencia entre la referencia *Ref* y la salida *T_el*.
- *I* es la integral del error.
- *D* es la derivada del error.
- *kp* es la ganancia proporcional. Afecta la rapidez de respuesta al error.
- *ki* es la ganancia integral. Ayuda a eliminar el error en estado estacionario.
- *kd* es la ganancia derivativa. Reduce la oscilación y mejora la estabilidad.

Para calcular estos parámetros del PID se optará por diseñar un controlador sin cancelación de dinámica por el dominio de la frecuencia, teniendo como especificaciones un tiempo de subida aproximadamente del orden del tiempo de respuesta del sistema, para evitar inestabilidad; errores en régimen permanente nulo y sobreoscilación nula (si esta última fuera posible).

5.2.1 Cálculo de la función de transferencia

Para ello en primer lugar, se calcula la función de transferencia del modelo mediante el siguiente experimento (Fig. 37): manteniendo las perturbaciones constantes *I_el* = 30 A, *Tamb* = 25ºC; se realiza una simulación de 40000 segundos en la que *Qcool* está inicialmente a 0 y a los 20000 segundos toma el valor 150.

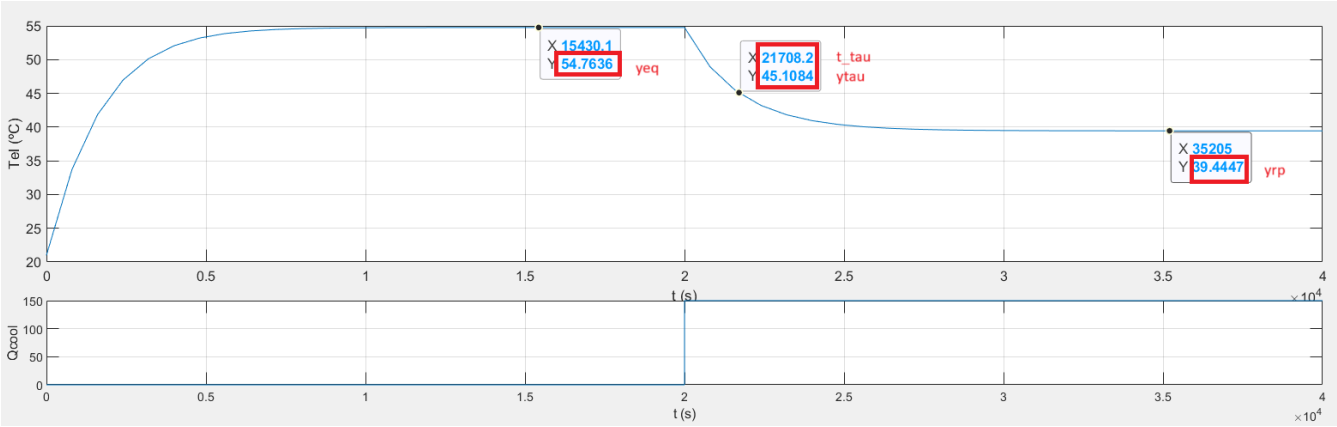

Figura 37. Temperatura del electrolizador ante escalón a la entrada

Observando la respuesta del sistema se puede aproximar por uno de primer orden con ganancia estática negativa.

A partir del experimento, se pueden sacar varios datos de interés que servirán para calcular el modelo:

- ***Yeq* = 54.76 ºC**. Valor de equilibrio de la salida.
- *Yrp* = 39.44 ºC. Valor de la salida en régimen permanente tras el escalón. Toma un valor por debajo del límite ideal de operación de temperatura, pero no es crucial en este caso ya que ahora solo nos interesa la respuesta del sistema al aumentar la refrigeración (*Qcool*).
- *Ytau* = 45.1084 ºC. Valor de la salida al 63% del régimen transitorio.
- *T_tau* = 21708 s. Instante de tiempo en el cual la salida vale *Ytau*.

Al ser un modelo de primer orden, tendrá la siguiente forma:

$$G(s) = \frac{Kest}{\tau s + 1}$$

Ecuación 26. Expresión de la función de transferencia de sistema de primer orden

Donde *Kest* será la ganacia estática y τ la constante de tiempo. Para calcular estos parámetros se hará uso de las siguientes fórmulas de Teoría de Control:

$$\boldsymbol{Kest} = \frac{Yrp - Yeq}{escalón} = \frac{39.44 - 54.76}{150} = \mathbf{0.1021}\ ºC/usc$$

Ecuación 27. Cálculo de la ganancia estática

$$\boldsymbol{\tau} = T\_tau - T\_escalón = 21708 - 20000 = \mathbf{1708}\ s$$

Ecuación 28. Cálculo de la constante de tiempo

Por tanto, el modelo con el que se trabajará será el siguiente:

$$G(s) = -\frac{0.1021}{1708\,s+1}$$

Ecuación 29. Función de transferencia del modelo

Se puede comprobar en la Figura 38 que el modelo es una buena aproximación del sistema real, ya que al realizar el mismo experimento coinciden los tiempos y valores de *Yeq*, *Yrp* e *Ytau* del sistema con el modelo.

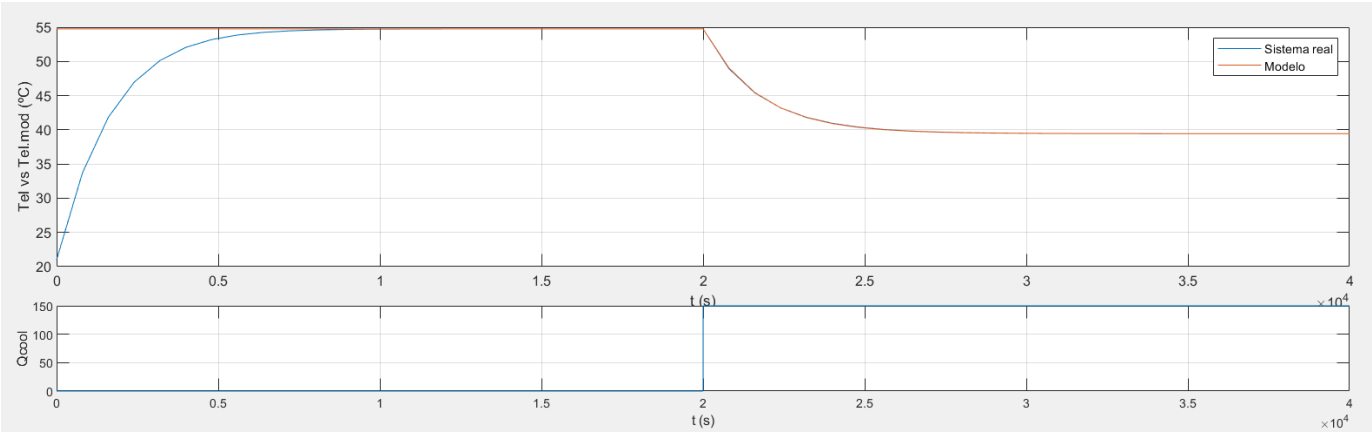

Figura 38. Comparación del modelo con el sistema real

5.2.2 Diseño del controlador

Para diseñar el controlador se hace uso de la G(s) previamente calculada y de un script de Matlab llamado '*DibujaBodeGba.m*'. Este programa permite introducir los datos de la G(s) y visualizar su respuesta frecuencial en bucle abierto, GBA(s), de manera que se pueda diseñar el controlador añadiendo los parámetros necesarios y visualizando gráficamente la forma resultante de la GBA(s). Como la ganancia estática del sistema es negativa, se trabajará sin signo para facilitar los cálculos del controlador y evitar márgenes de fase negativos; y cuando esté diseñado se le añadirá el signo menos correspondiente.

$$GBA(s) = G(s)*C(s)$$

Ecuación 30. Expresión de GBA(s)

```
%%%%%%%%%%%%%%%%%%%%%%%% CONTROLADOR %%%%%%%%%%%%%%%%
numC=[1 1];   % Numerador del controlador
denC=[1 0];   % Denominador del controlador
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%% DATOS DEL SISTEMA %%%%%%%%%%%%%%%%%%%%

% DEFINICIÓN DE FUNCIÓN DE TRANSFERENCIA TEÓRICA
num=0.1021;    % Numerador
den=[1708 1];  % Denominador

% Rango de frecuencias para dibujar el bode teórico

w_min=0.000001; % Frecuencia mínima en rad/s
w_max= 100; % Frecuencia máxima en rad/s
```

```
% Vector de frecuencias entre w_min y w_max con 100 datos logarítmicamente
equiespaciados
w_teorica=logspace(log10(w_min), log10(w_max), 100);

% CÁLCULO DE Gba
numGba_t= conv(numC,num);
denGba_t= conv(denC,den);
[gGba_t, fGba_t]=bode(numGba_t,denGba_t,w_teorica);


% DIBUJO DE GRÁFICAS
figure;  % Nueva gráfica;
subplot(2,1,1); % Dividida en dos y dibujando sobre la parte de arriba

% Dibujo de la ganancia en decibelios:
%   semilogx: igual que la función "plot", pero el eje horizontal lo representa en
escala logarítmica
%   log10 :  logaritmo en base 10 (para pasar a decibelios)
semilogx(w_teorica,20*log10(gGba_t),'b',[min(w_min) max(w_max)],[0 0],'k');
xlabel('frecuencia (rad/s)');
ylabel('ganancia (dB)');
legend('Gba Teórica');
grid;
% Seleccionando la parte de abajo de la grçafica para dibujar
subplot(2,1,2);
% Dibujo de la ganancia fase en grados:
semilogx(w_teorica,fGba_t-360,'b');
xlabel('frecuencia (rad/s)');
ylabel('fase (º)');
legend('Gba Teórica');
grid;
```

Código 3. Script de Matlab 'DibujaBodeGba.m'

En el Código 3, 'num' y 'den' representan el numerador y denominador del modelo G(s), mientras que 'numC' y 'denC' representan el numerador y denominador del controlador C(s). Inicialmente se partirá de un controlador base $C(s) = \frac{1}{1}$.

Según las especificaciones de diseño, el tiempo de subida, tsbc, debe ser aproximadamente 3 veces la constante de tiempo del modelo para que se controle a la velocidadad del propio sistema (Ec. 31).

$$tsbc = 3 * \tau = 3 * 1708 = 5124 \text{ s} \approx 5000\text{s}$$

Ecuación 31. Cálculo del tiempo de subida en bucle cerrado

Por tanto, la frecuencia de corte será:

$$wc = \frac{\pi}{tsbc} = \frac{\pi}{5000} = 6.2832*10^{-4} \text{ rad/s}$$

Ecuación 32. Cálculo de la frecuencia de corte

En primer lugar, se parte de la GBA(s) del sistema y de un controlador C(s) = $\frac{1}{1}$. Se ejecuta el script (Cod. 3) para visualizar el resultado (Fig. 39).

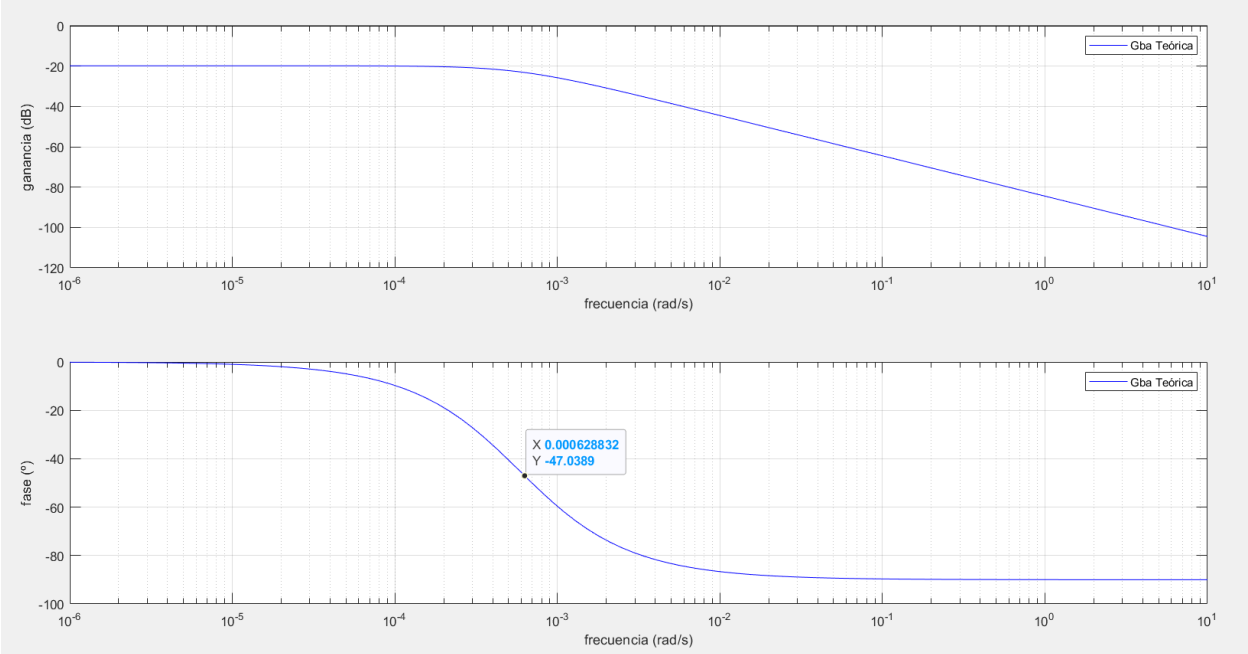

Figura 39. GBA(s) con C(s) = 1/1

Como no queremos errores en régimen permanente se añade un integrador y como no queremos sobreoscilación se tiene un margen de fase deseado, 'Mfdes', de 90º.

De la Figura 33 se puede obtener el margen de fase actual, 'Mfact':

$$Mfact = 180º - 47.0389º = 132.9611º$$

Ecuación 33. Cálculo del margen de fase actual

Para lograr el margen de fase deseado, el controlador debe aportar una fase, 'γ':

$$\gamma = Mfdes - Mfact + 5º = 90º - 132.9611º + 5º = -37.9611º$$

Ecuación 34. Cálculo del aporte de fase

Como γ < -35, se opta por diseñar un **PI** en vez de un PID. Al tratarse de un PID el controlador tendrá un solo cero y tendrá la siguiente forma:

$$C(s) = -\frac{k*(\tau c\, s+1)}{s}$$

Ecuación 35. Expresión del controlador PI

Donde 'k' es la ganancia, 'τc' es la constante de tiempo del controlador y la 's' en el denominador es el integrador previamente añadido.

La constante de tiempo de puede calcular con la siguiente ecuación:

$$\tau c = \frac{1}{wc} * \tan(90º + \gamma) = \frac{1}{0.00062832} * \tan(90º - 37.9611º) = 776.9194$$

Ecuación 36. Cálculo de la constante de tiempo del controlador

También se decide añadir un polo a alta frecuencia, 'τaf', que actuará como filtro de ruido, mejorará la realzabilidad del controlador y evitará ampliación en frecuencias no deseadas (Ec. 37).

$$\tau af = \frac{1}{10*wc} = \frac{1}{10*0.00062832} = 159.1549$$

Ecuación 37. Cálculo del polo a alta frecuencia

Se vuelve a ejecutar el script (Cod. 3), pero esta vez añadiendo el integrador, el cero del controlador y el polo a alta frecuencia (Fig. 40).

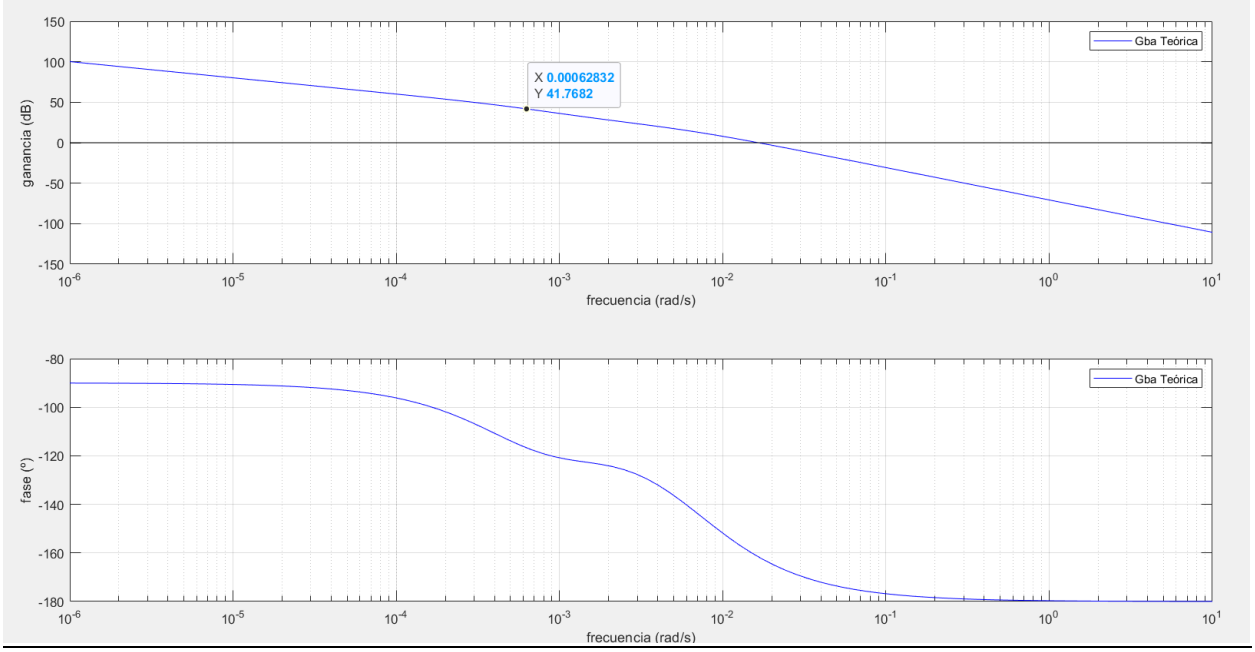

Figura 40. GBA(s) con C(s) = (776.9194s+1)/(159.1549s+1)s

La ganancia se puede ajustar, observando de la figura 40 la ganancia para la frecuencia de corte. En este caso, 41.7682 db. Por tanto, la ganancia, k, será:

$$k = 10^{-\frac{41.7682}{20}} = 0.0082$$

Ecuación 38. Cálculo de la ganancia k

Se vuelve a ejecutar el script (Cod. 3) añadiendo esta ganancia, para comprobar los resultados (Fig. 41):

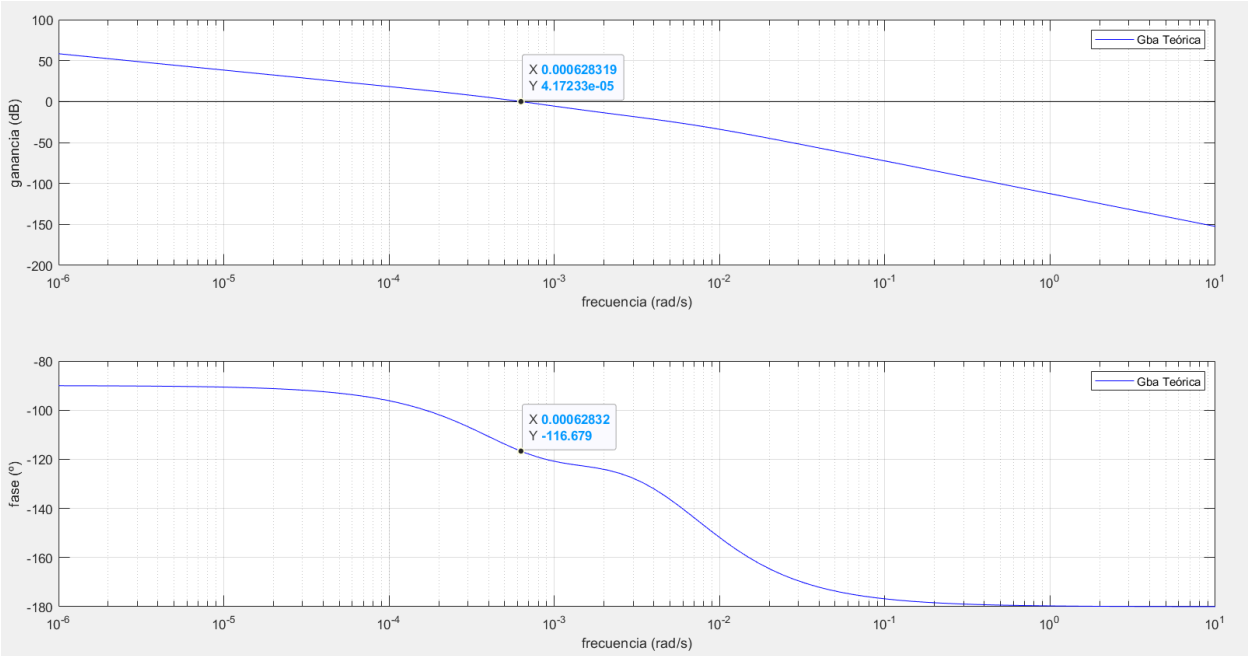

Figura 41. GBA(s) con C(s) = 0.0082(776.9194s+1)/(159.1549s+1)s

Se puede comprobar en la figura 41, que la ganancia a la frecuencia de corte es prácticamente cero y se tiene una fase constante de aproximadamente -90º hasta la frecuencia de corte, por lo que teóricamente el controlador debe funcionar correctamente como se desea.

Por último, se añade el signo correspondiente al controlador, quedando definido como

$$\mathbf{C(s)} = -0.0082\,\frac{(776.9194\,s+1)}{(159.1549\,s+1)\,s}$$

Ecuación 39. Expresión final del controlador PI

Antes de simular se realiza un prefiltrado a la referencia, Cr(s), para evitar sobreoscilaciones y picos al inicio, donde el controlador puede reaccionar de forma exagerada a un cambio brusco en la referencia:

$$\mathrm{Cr(s)} = \frac{1}{\frac{tsbc}{3}s+1} = \frac{1}{\frac{5000}{3}s+1} = \frac{1}{1666.67\,s+1}$$

Ecuación 40. Expresión del prefiltro

En la figura 42 se puede observar como se programó una simulación de 50000s para visualizar como responde el sistema controlado ante cambios de referencia y rechazo de perturbaciones:

- Figura 42,1. Se añade el controlador previamente calculado (Ec. 39).
- Figura 42,2. Se añade el prefiltro previamente calculado (Ec. 40).
- Figura 42,3. Se añade un bloque 'Switch', que se activa a los 2500s, permitiendo al sistema alcanzar su punto de equilibrio.
- Figura 42,4. Se realimenta negativamente la salida.
- Figura 42,5. Se programa el cambio de referencia, que inicialmente vale 54.76ºC: a los 10000s se le da un escalón de -14.96ºC.

- Figura 42,6. Se programa un cambio de la perturbación *I_el*, que inicialmente vale 30A: a los 25000s se le da un escalón de 5A.
- Figura 42,7. Se programa un cambio de la perturbación *Tamb*, que inicialmente vale 25ºC: a los 40000 se le da un escalón de 5ºC.
- Figura 42,8. Por último, se añade un bloque 'Saturation' con límite inferior en 0ºC y límite superior en 300ºC.

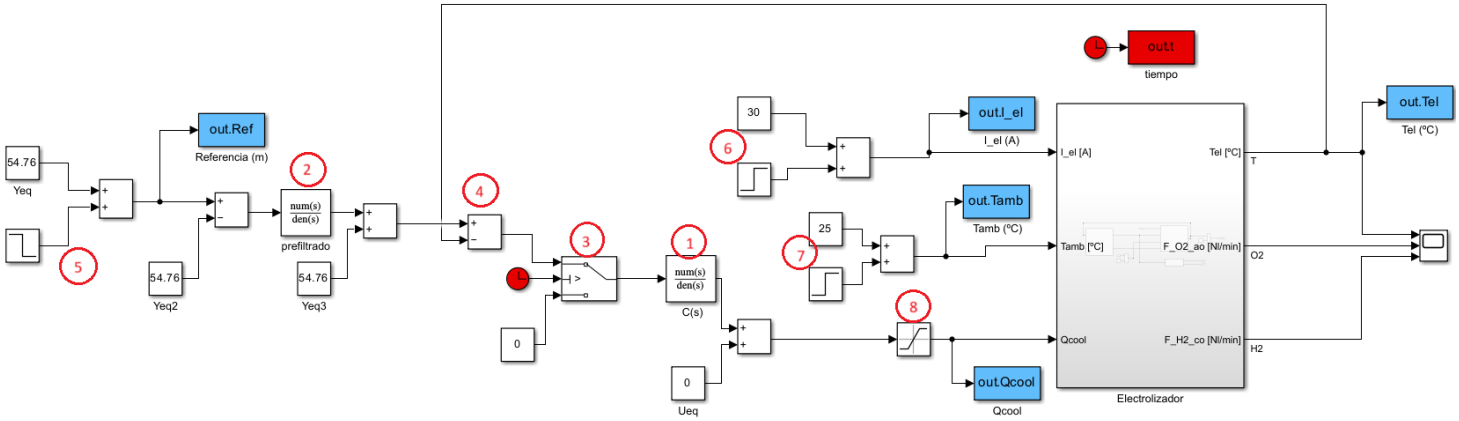

Figura 42. Simulación programada en Simulink

Tras realizar la simulación se obtienen los siguientes resultados (Fig. 43):

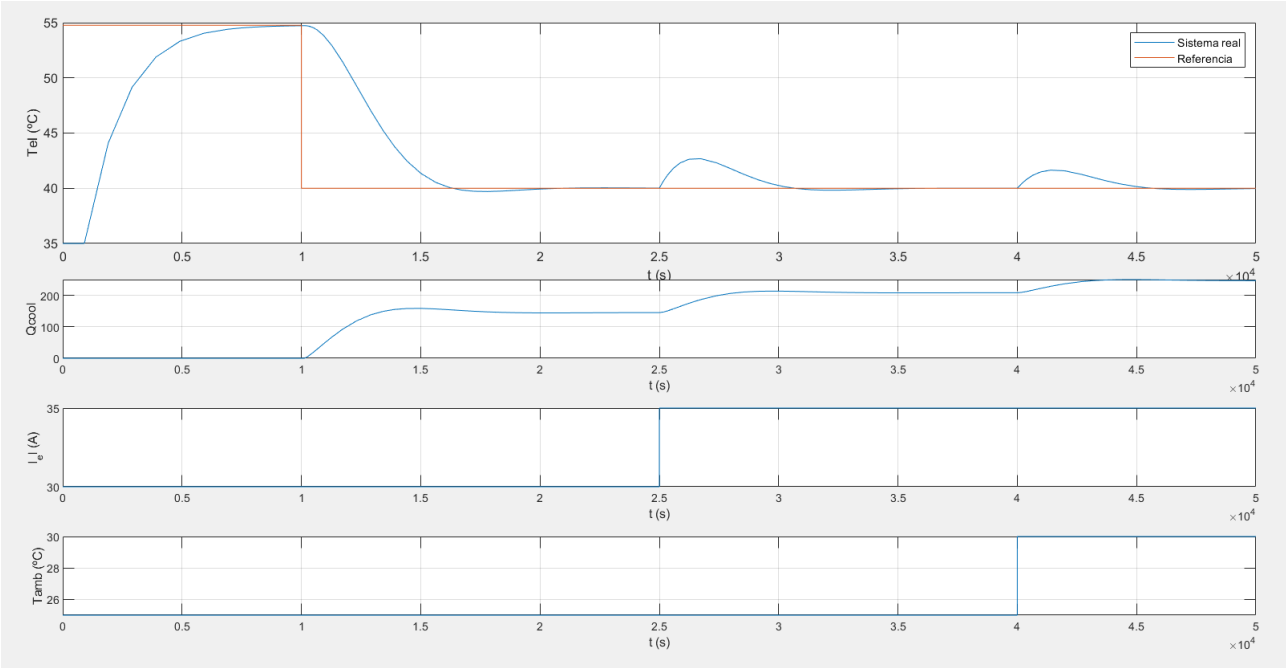

Figura 43. Resultados de la simulación en Matlab

Se puede apreciar que el sistema alcanza la referencia y rechaza las perturbaciones en el tiempo establecido (5000s), sin sobreoscilaciones, sin errores en régimen permanente y sin que la señal de control sature; por lo que se puede dar este experimento como válido y se puede considerar que el controlador esta bien diseñado.

5.3 Control de la temperatura desde EAE

Una vez diseñado el controlador en Matlab, se pretende poner en práctica la comunicación mediante OPC UA entre Simulink y EAE. El objetivo es que el control se realice desde EAE, y Matlab solo sea un intermediario, actuando como cliente que solo envía y recibe datos de EAE para simular la planta en Simulink. El procedimiento a seguir será el siguiente:

1. Simulink escribe sobre las variables *Tel* y *Tref* .
2. EAE implementa el PI discreto con los datos recibidos y calcula *Qcool*.
3. Simulink lee Qcool y se la da como entrada al electrolizador.

5.3.1 Configuración en Matlab

Como se ha explicado, Matlab solo actuará como intermediario, enviando y recibiendo datos. Para establecer la comunicación OPC UA entre aplicaiones es necesario realizar una configuración similar a la del capítulo 5.1, con la diferencia que los datos se enviarán y recibirán desde Simulink.

5.3.1.1 Sample time

La acción de control se aplicará de forma discreta con un mantenedor de orden cero (zoh). Para implementar esto, se establece el sample time de Simulink a 0.001 segundos para obtener una alta precisión y resolución temporal; y se realiza la escritura y lectura cada 10s. De esta forma, se consigue que la dinámica de control sea más lenta que la dinámica del sistema, como ocurriría en la realidad, ya que al tratarse de la temperatura es un proceso que evoluciona lentamente.

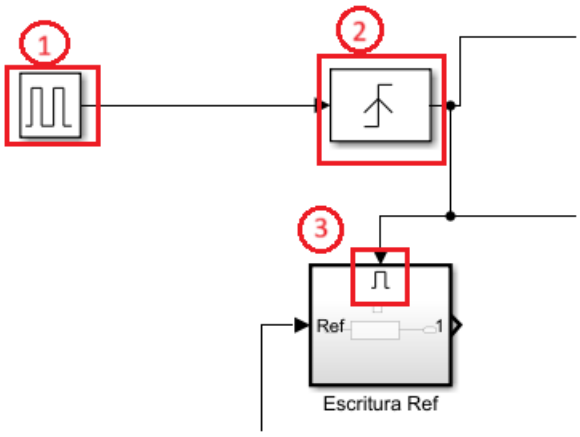

Figura 44. Señal de habilitación de los bloques de lectura y escritura

Por otra parte, para que la escritura y lectura se haga cada 10 s, los bloques de escritura y lectura se han anidado dentro de *Subsystems* con señales de 'Enable' (Fig. 44,3). De esta forma, las funciones se ejecutarán únicamente cuando 'Enable' esté a nivel alto. Esta señal se activará cuando se produzca un flanco de subida (Fig. 44,2) en el generador de pulsos creado (Fig. 44,1). Este tren de pulsos tiene un periodo de 0.01 s y una amplitud de pulso del 1%, por lo que estará a nivel alto durante 0.1s cada 10s.

Dado que el sample time de Simulink y EAE se ha fijado en 0.001 s, y la señal de habilitación ('Enable') permanece en nivel alto durante 0.1 s cada 10 s (según el generador de pulsos configurado), se concluye que dicha señal estará activa durante 100 ciclos de simulación. Esto asegura que los bloques de lectura y escritura se activen brevemente dentro de cada ventana de control, tal como se busca simular un control discreto lento sobre una planta de dinámica más rápida.

5.3.1.2 Funciones de lectura y escritura

Para leer y escribir desde Simulink, se han creado tres funciones distintas y un script de conexión incial, particularizando el script mostrado en el capítulo 5.1 (Cód. 2).

En primer lugar, es necesario ejecutar el script '*conexion.m*', que inicializa la conexión:

```
%%%% Inicialización %%%%
global uaClient

% Acceder a los parámetros definidos en la máscara
serverName = 'EcoStruxure';

% Obtener información del servidor
serverList = opcuaserverinfo('localhost');
hsInfo = findDescription(serverList, serverName);

% Crear cliente y conectar
uaClient = opcua(hsInfo);
connect(uaClient);
uaClient
```

Código 4. Script de Matlab 'conexion.m'

El cliente 'uaClient' se crea como variable global para que las funciones de escritua y lectura tengan acceso al mismo, y no sea necesario crear un cliente cada vez que se llame a alguna de ellas. Esto se ha hecho con la finalidad de acelerar las simulaciones.

Al ejecutar este script, si todo está orden deberá aparecer un mensaje como el de la Figura 34 en la ventana de comandos de Matlab, asegurando que se ha podido establecer la conexión.

A continuación, se muestra la función de escritura de '*Tref*':

```
function y = writeRef(u)
    persistent uaClient nodoActual

    % Inicialización de uaClient y nodo
    if isempty(uaClient)
        % Inicializa el cliente OPC UA
        uaClient = opcua('localhost', 4840);
        connect(uaClient);

        % Navegar jerárquicamente
        nodePath = {'drawwork', 'IThis', 'Tref'};
        nodoActual = uaClient.Namespace;
        for i = 1:length(nodePath)
            nodoActual = findNodeByName(nodoActual, nodePath{i});
            if isempty(nodoActual)
                error("No se encontró el nodo '%s' en la ruta.", nodePath{i});
            end
        end
    end

    % Valor a escribir
    nuevoValor = u;

    % Escribir en el nodo
    writeValue(uaClient, nodoActual, nuevoValor);
```

```
    % Devolver el valor como confirmación
    y = nuevoValor;

    fprintf("Valor '%s' escrito correctamente en la variable '%s'.\n", ...
        string(nuevoValor), 'Tref');
end
```

Código 5. Función de Matlab 'writeRef.m'

Por otra parte, la escritura sobre la variable '*Tel*' se hace con la siguiente función:

```
function y = writeTemp(u)
    persistent uaClient nodoActual

    % Inicialización de uaClient y nodo
    if isempty(uaClient)
        % Inicializa el cliente OPC UA
        uaClient = opcua('localhost', 4840);
        connect(uaClient);

        % Navegar jerárquicamente
        nodePath = {'drawwork', 'IThis', 'Tel'};
        nodoActual = uaClient.Namespace;
        for i = 1:length(nodePath)
            nodoActual = findNodeByName(nodoActual, nodePath{i});
            if isempty(nodoActual)
                error("No se encontró el nodo '%s' en la ruta.", nodePath{i});
            end
        end
    end

    % Valor a escribir
    nuevoValor = u;

    % Escribir en el nodo
    writeValue(uaClient, nodoActual, nuevoValor);

    % Devolver el valor como confirmación
    y = nuevoValor;

    fprintf("Valor '%s' escrito correctamente en la variable '%s'.\n", ...
        string(nuevoValor), 'Tel');
end
```

Código 6. Función de Matlab 'writeTemp.m'

Por último, la lectura de la variable '*Qcool*' se realiza ejecutando la siguiente función:

```
 function y = readQcool(u)
    persistent uaClient nodoActual

    % Inicialización de uaClient y nodo
    if isempty(uaClient)
        % Inicializa el cliente OPC UA
        uaClient = opcua('localhost', 4840); % ajusta IP/puerto
```

```
        connect(uaClient);

        % Navegar jerárquicamente
        nodePath = {'drawwork', 'IThis', 'Qcool'};
        nodoActual = uaClient.Namespace;
        for i = 1:length(nodePath)
            nodoActual = findNodeByName(nodoActual, nodePath{i});
            if isempty(nodoActual)
                error("No se encontró el nodo '%s' en la ruta.", nodePath{i});
            end
        end
    end

    % Leer valor
    Q = readValue(uaClient, nodoActual);
    y = double(Q);

    fprintf("Valor '%s' leído correctamente en la variable '%s'.\n", ...
        string(y), 'Qcool');
end
```

Código 7. Función de Matlab 'readQcool.m'

Sin embargo, al realizar pruebas, esta filosofía daba constantemente el siguiente error por el uso de variables globales en las funciones:

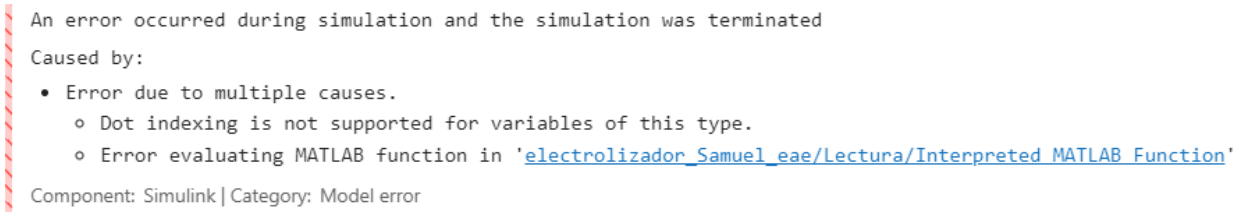

Figura 45. Error por el uso de variables globales en las funciones

Este error se consiguió solucionar con el uso de variables de tipo 'persistent' dentro de las funciones en lugar de las globales. Esta decisión se fundamenta en criterios de robustez, encapsulación y mantenimiento del código. Una variable 'persistent' mantiene su valor entre llamadas consecutivas a la función, pero su ámbito se limita exclusivamente a dicha función. Esto garantiza que la conexión con el servidor OPC UA y la referencia a los nodos se inicialicen una única vez y se conserven para posteriores ejecuciones, evitando así la sobrecarga de reconexiones innecesarias. Al mismo tiempo, este enfoque aísla la gestión de la conexión del resto del código, previniendo errores derivados de modificaciones externas o de la dependencia del orden de ejecución del programa principal, como podría ocurrir con el uso de variables global. Para sistemas más complejos, que usan muchas funciones distintas para acceder al servidor OPC UA esta solución si podría provocar inefiecia o retardos, pero en este caso, como se emplean tres funciones con un único 'persistent uaClient' dentro de cada una no supondría ningún problema.

Para llamar a estas funciones, se utilizan los bloques 'Interpreted MatLab Function'. Cabe destacar que este tipo de bloques no permite recibir más de una entrada. Es por ello por lo que no se ha conseguido diseñar una función general de escritura a la que se pueda pasar como parámetros de entrada la ruta del nodo y el valor a escribir.

Los *Subsystems* donde se encuentran los respectivos bloques 'Interpreted MatLab Function' son como los que se muestran en la Figura 44,3. Internamente se ven de esta forma:

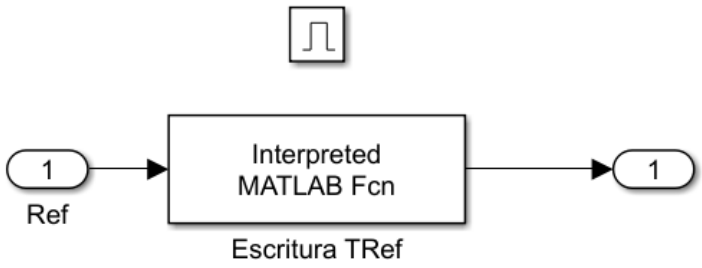

Figura 46. Subsystem 'Escritura Ref'

Con la idea de realizar el mismo experimento que se muestra en la Figura 42 y comparar los resultados con los mostrados en las Figura 43, se realiza el siguiente montaje en Simulink:

- Se añaden los *Subsystems* de lectura y escritura, mostrados en la Figura 46; y los bloques mostrados en la Figura 44.
- Los cambios de referencia y perturbaciones se siguen realizando en Simulink.
- Se mantiene el prefiltrado de la referencia en Simulink por simplicidad.
- Se añaden 'Scopes' y 'Displays' para visualizar la simulación en tiempo real.

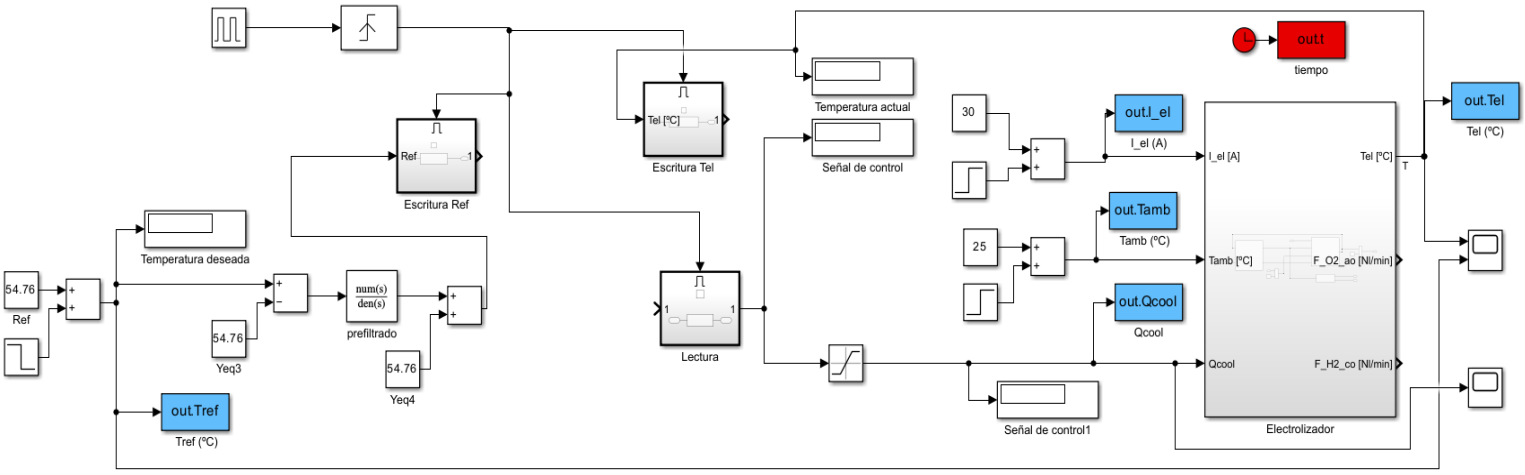

Figura 47. Diagrama de bloques de Simulink para la conexión con EAE

5.3.2 Diseño del PI desde EAE

La lógica a seguir para la implementación del PI en EAE se ha diseñado según la ecuación de control del un PID (Ec. 25), manteniendo a cero el valor del parámetro *kd*.

A continuación, se muestra el diagrama de bloques realizado, tomando como base el ejemplo del capítulo 5.1; así como una explicación detallada de cada bloque:

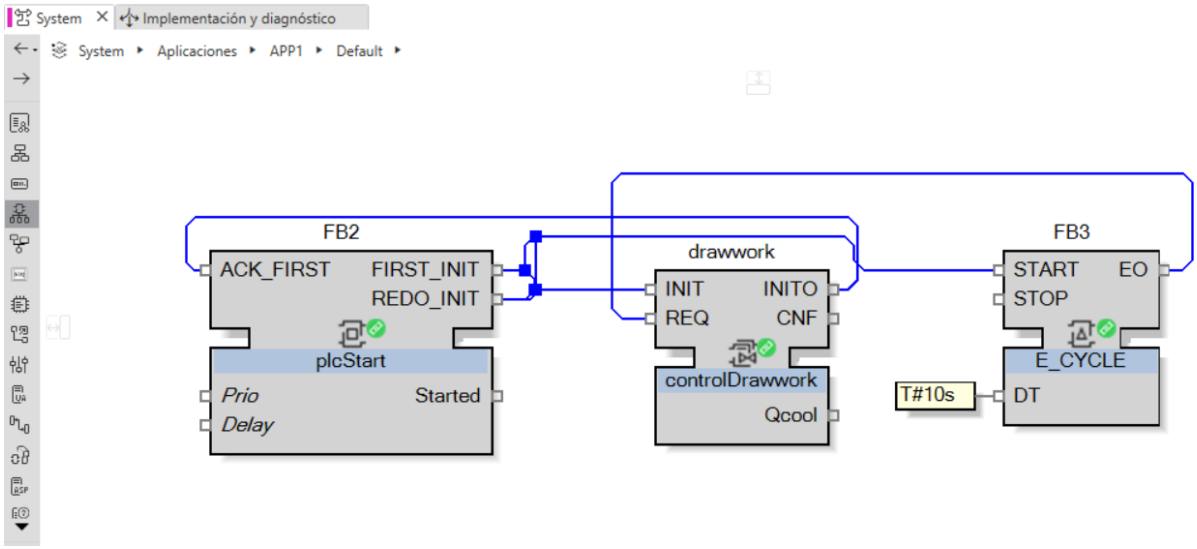

Figura 48. Red de bloques de la aplicación

Con respecto a la Figura 29, se añade un bloque 'E_CYCLE', con el objetivo de generar el evento REQ cada 10 segundos. Esto se hace con la idea de ahorrar coste computacional, una mejor adaptación a la velocidad real del proceso y evitar saturar el controlador; ya que al tratarse de un proceso térmico no se requiere actuar o leer datos muy rápido.

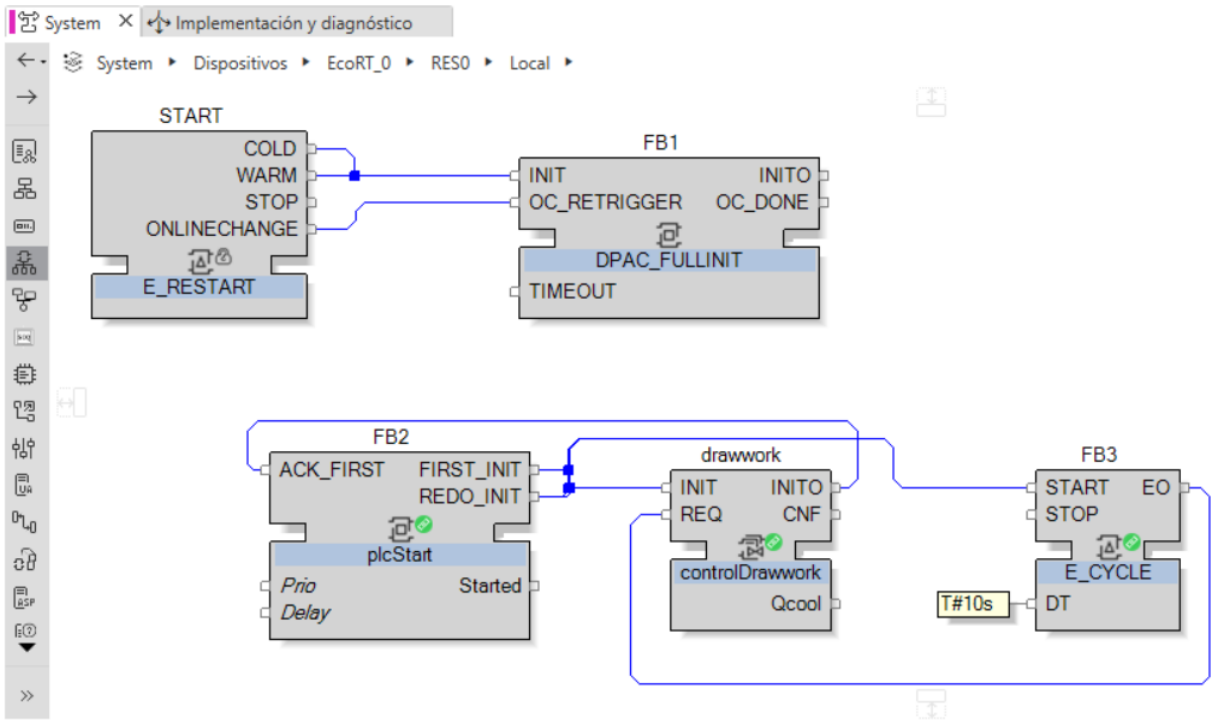

Figura 49. Red de bloques del recurso

Los bloques de la aplicación (Fig. 48) se mapean al recurso 'EcoRT_0.RES0', al igual que se hacía en el ejemplo (Fig. 30).

El bloque 'drawwork' cumplirá el mismo papel que el bloque 'prueba' (Fig. 26) del ejemplo, será donde se desarrollará toda la lógica del controlador.

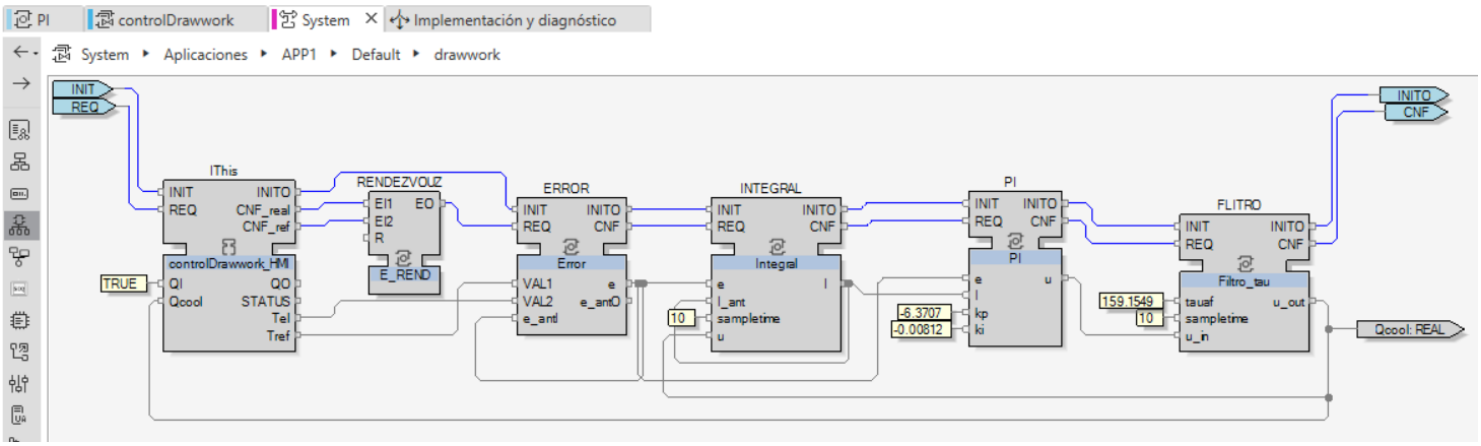

Figura 50. Red de bloques del CAT 'drawwork'

A continuación, se explicará el funcionamiento de cada uno de los bloques de la Figura 50, en sus respectivos subcapítulos.

5.3.2.1 Bloque IThis

Como ya se explicó anteriormente, las variables que se deseen compartir deben encontrarse dentro de un bloque 'IThis'. EAE recibe los valores de '*Tel*' y '*Tref*' y devuelve el valor de la señal de control '*Qcool*'; variables de Matlab (Fig. 47). En EAE por simplicidad se les han dado el mismo nombre a cada variable. Las dos primeras son salidas del bloque, para permitir la escritura, y la última es entrada, para que solo pueda leerse. Esto evita que el cliente (Simulink) modifique su valor por error. Es importante recordar que hay que exponer las variables en la pestaña *Servidor OPCUA* dentro del bloque (Fig. 51), dándoles los accesos necesarios, como se hizo en el ejemplo.

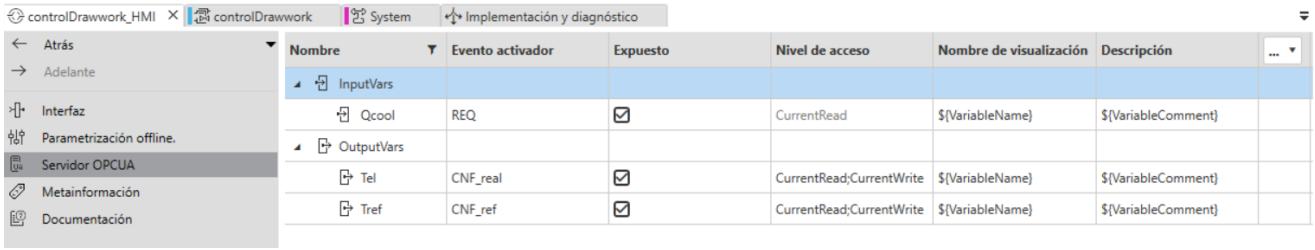

Figura 51. Configuración OPC UA de las variables compartidas en 'controlDrawwork_HMI'

En este bloque se configuran dos eventos CNF; 'CNF_real', que se asocia a la variable '*Tel*' y 'CNF_ref', que se asocia a la variable '*Tref*'. Se configura de esta forma debido a que la lectura de ambas variables se produce en el mismo ciclo de reloj, generando dos eventos simultaneos.

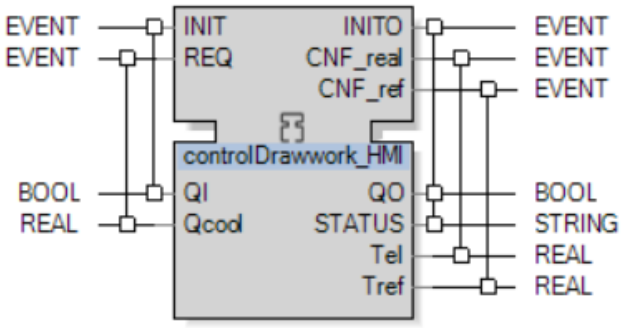

Figura 52. Datos y eventos del bloque 'controlDrawwork_HMI'

5.3.2.2 Bloque RENDEZVOUZ

Este bloque converge los dos eventos 'CNF' del anterior bloque. Lo hace generando un nuevo evento de salida 'EO' únicamente cuando ambos eventos de entrada 'EI1' y 'EI2' se producen.

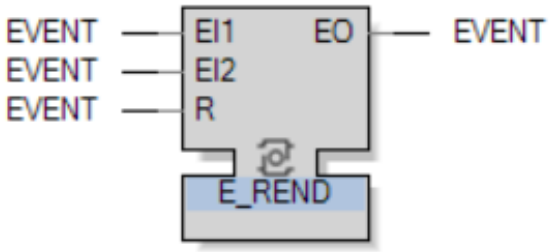

Figura 53. Datos y eventos del bloque 'E_REND'

5.3.2.3 Bloque ERROR

El error se calcula como la diferencia entre la temperatura comandada '*Tref*' y la temperatura real '*Tel*'. Para esta simple operación, se ha creado un BFB que toma como entradas estos valores y devuelve el error '*e*' (Fig. 50). Además, se ha creado una nueva entrada llamada '*e_antI*' en la que se copia el error calculado a la salida, y una nueva salida '*e_antO*' para sacar también este valor. Esto se hace en caso de que se quiera guardar el valor del error anterior para usarlo en el término derivativo de un PID, que no es el caso.

También se ha creado una variable interna de tipo bool, '*iniciar*', que se inicializa a FALSE. Esta variable se utilizará para "activar el control" una vez que se alcance por primera vez a la referencia.

Nombre	Tipo	Tam...	Valor ini...	Con	Atrib
EventInputs					
INIT					<none>
REQ				VAL1, VAL2, e_...	<none>
<nueva interfaz>					
EventOutputs					
INITO					<none>
CNF				e, e_antO	<none>
<nueva interfaz>					
InputVars					
VAL1	REAL				<none>
VAL2	REAL				<none>
e_antI	REAL				<none>
<nueva variable>					
OutputVars					
e	REAL				<none>
e_antO	REAL				<none>
<nueva variable>					
InternalVars					
iniciar	BOOL		FALSE		<none>
<nueva variable>					

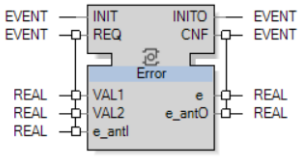

Figura 54. Datos y eventos del bloque 'Error'

La secuencia de estados es la siguiente:

1. INIT: se inicializa el ciclo.
2. RESTA: ejecuta el algoritmo 'Resta' que calcula el nuevo error y guarda el error anterior. Al terminar, activa el evento de salida 'CNF'.

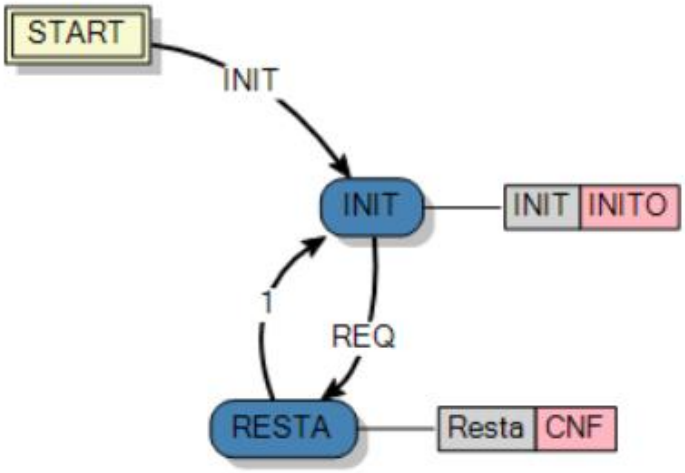

Figura 55. Secuencia de estados del bloque 'Error'

El algoritmo (Cód. 8) calcula el resultado del error cuando la variable 'iniciar' esta a TRUE, es decir, una vez se haya alncanzado por primera vez la referencia. Antes de llegar a esta condición se fuerza el error a cero para evitar que se acumule error innecesario en el bloque 'Integral' y ralentice la respuesta del sistema. Esto provoca, según la Ecuación 25, que la salida del PI sea cero hasta que se de dicha condición.

De esta manera, se podría decir que no se empieza a controlar[4] hasta que la temperatura de referencia alcance por primera vez la temperatura de referencia.

La variable de entrada '*e_antI*' está conectada a la variable de salida '*e*' (Fig. 50), por lo que se copiará en '*e_antI*' el valor que tenga '*e*' cada vez que termine ejecución del algoritmo.

[4] Realmente si se está controlando pero como la señal de control se mantiene a cero no tiene ningún efecto

```
ALGORITHM Resta IN ST:
(* Add your comment (as per IEC 61131-3) here

*)
;
IF (VAL1<>0) AND (VAL2<>0) AND (ABS(VAL2-VAL1)<=0.1) THEN // Cuando se alcance por primera vez la referencia
   iniciar:=TRUE;
END_IF;

// Se fuerza que el error sea 0 hasta que el no se haya alcanzado aún la referencia
IF iniciar=TRUE THEN
   e:=VAL1-VAL2;
ELSE
   e:=0;
END_IF;

e_antO:=e_antI;
END_ALGORITHM
```

Código 8. Algoritmo 'Resta' del bloque 'Error'

5.3.2.4 Bloque INTEGRAL

En este bloque se calcula el término integral '*I*'. Del mismo modo, se hace usando un BFB que toma como entradas el error '*e*', la integral anterior '*I_ant*', el periodo de muestreo '*sampletime*' y el valor de la señal de control '*u*'.

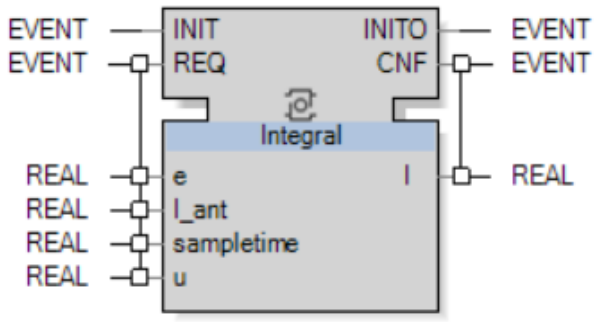

Figura 56. Datos y eventos del bloque 'Integral'

La secuencia de estados es la siguiente:

1. INIT: se inicializa el ciclo.
2. ESPERA: tras inicializar, se entra directamente en este estado que únicamente espera a que llegue un evento de entrada 'REQ' con nuevos datos.
3. CALCULO: cuando se produce el evento REQ, se ejecuta el algoritmo 'Calculo'. Al finalizar, se activa el evento de salida 'CNF' para que continue la cadena.

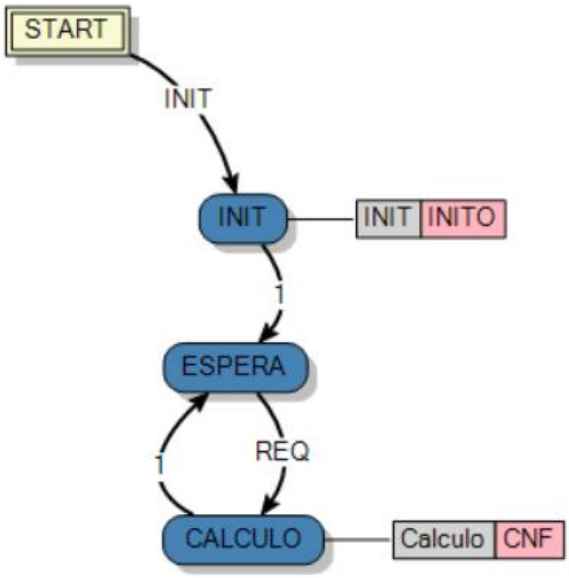

Figura 57. Secuencia de estados del bloque 'Integral'

El algoritmo 'Calculo', como su propio nombre indica, calcula la integral '*I*' sumando el error acumulado.

También se implementa control anti-windup para evitar que se siga acumulando error cuando la señal de control ya está saturada (Capítulo 4.3.3). De esta manera se evita "inflar" el término integral inncesariamente, ya que esto provocaría un retardo considerable o incluso la intestabilidad del sistema.

Al igual que en el bloque 'Error', la entrada '*I_ant*' está conectada a la salida '*I*', por lo que se copiará su valor al terminar el algoritmo.

```
ALGORITHM Calculo IN ST:
(* Add your comment (as per IEC 61131-3) here

*)
;
IF u>=300 AND e<0 THEN     // Si se alcanza el límite superior
   I:=I_ant;
ELSIF u<=0 AND e>0 THEN // Si se alcanza el límite inferior
   I:=I_ant;
ELSE                                 // Si no satura
   I:=I_ant+(e*sampletime);
END_IF;
END_ALGORITHM
```

Código 9. Algoritmo 'Calculo' del bloque 'Integral'

5.3.2.5 Bloque PI

En este bloque se calcula la acción de control. También se hace usando un BFB que tiene como entradas el error '*e*', el término integral '*I*' y los parámetros del PI, '*kd*' y '*ki*', del controlador calculado en el capítulo 5.2.2 (Ec. 39). Sin embargo este controlador no puede intrudirse tal y como está en el programa por lo que es necesario adaptarlo. De la Ecuación 39 se obtendrán los parámetros *kd* y *ki*; y el filtro a alta frecuencia se implementará en el siguiente bloque por separado.

Un PI puede expresarse de la siguiente forma:

$$C(s) = kp\frac{(Ti\,s+1)}{Ti}$$

Ecuación 41. Expresión de un PI

Igualando esta expresión (Ec. 41) a la expresión obtenida anteriormente para el controlador (Ec. 39), sin tener en cuenta el polo a alta frecuencia, se pueden extraer los parámetros:

$$Ti = 776.9194$$
$$\boldsymbol{kp} = -0.0082 \cdot 776.9194 = \boldsymbol{-6.3707}$$
$$\boldsymbol{ki} = \frac{kp}{Ti} = \boldsymbol{-0.00812}$$

Ecuación 42. Cálculo de los parámetros del PI

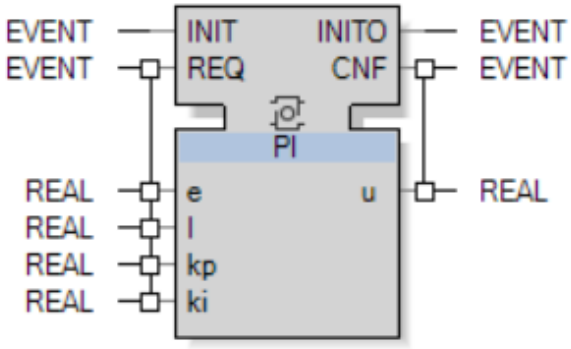

Figura 58. Datos y eventos del bloque 'PI'

Donde '*ki*' y '*kd*' son constantes con los valores calculados (Ec. 42).

La secuencia de estados es la siguiente:

1. INIT: se inicializa el ciclo.
2. PID: al terminar el cálculo del término '*I*' se activa el evento de entrada 'REQ' y comienza el algoritmo 'Pid'. Al finalizar, se activa el evento de salida 'CNF'.

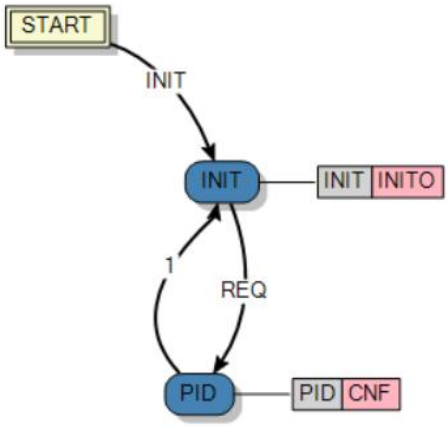

Figura 59. Secuencia de estados del bloque 'PI'

En el 'Pid' se calcula la acción de control '*u*' con la ecuación expuesta al principio del capítulo (Ec. 25), despredicando el parámetro *kd*, ya que se trata de un PI y no un PID.

```
ALGORITHM PID IN ST:
(* Add your comment (as per IEC 61131-3) here

*)
;
u:=kp*e+ki*I;
END_ALGORITHM
```

Código 10. Algoritmo `Pid' del bloque 'PI'

5.3.2.6 Bloque FILTRO

Por último, en este bloque se implementa el filtro a alta frecuencia. Este bloque recibe como entradas la acción de control calculada por el bloque anterior '*u_in*', el tiempo de muestreo '*sampletime*' y el valor de la constante de tiempo del polo alta frecuencia del contrlador (Ec. 39) '*tau_af*'. La salida de este bloque '*u*_out' es finalmente la señal de control que se leerá desde Matlab.

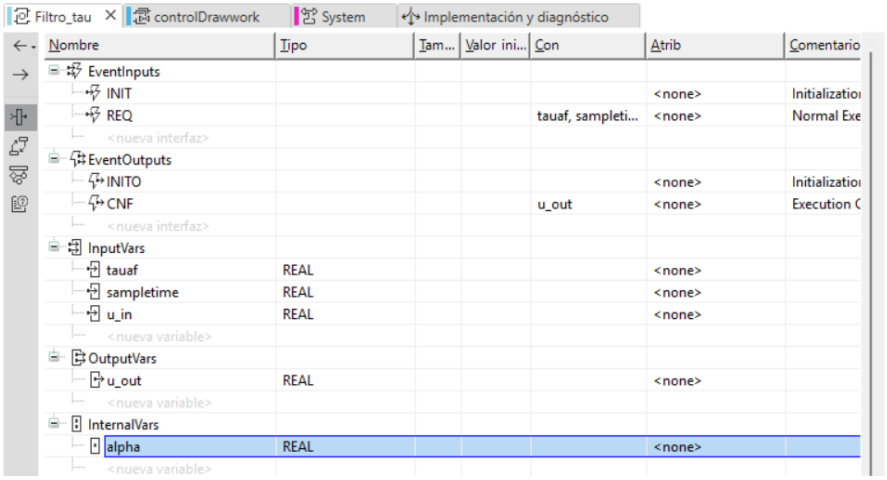

Figura 60. Datos y eventos del bloque 'Filtro_tau'

La secuencia de estados es la misma que en bloques anteriores.

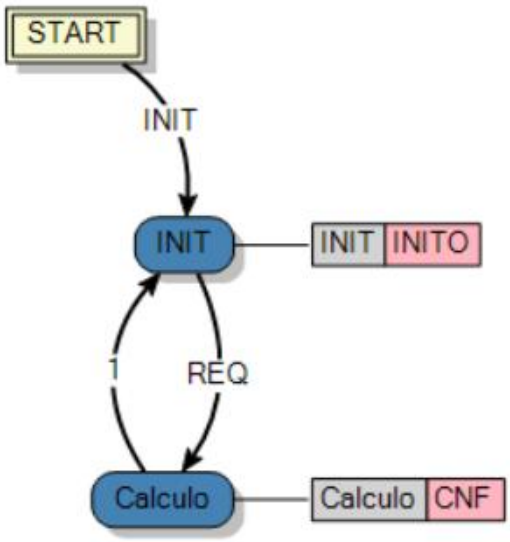

Figura 61. Secuencia de estados del bloque 'Filtro_tau'

El algoritmo 'Calculo' calcula el valor de la acción de control filtrada por el polo a alta frecuencia siguiendo la siguiente ecuación:

$$u[k] = \alpha \cdot u[k] + (1 - \alpha) \cdot u[k-1]$$

$$\alpha = \frac{Ts}{\tau_{af} + Ts}$$

Ecuación 43. Cálculo del filtro a alta frecuencia

También límita salida en el rango de operación para evitar saturación.

```
ALGORITHM Calculo IN ST:
(* Add your comment (as per IEC 61131-3) here

*)
alpha:=sampletime/(tauaf+sampletime);
u_out:=alpha*u_in+(1.0-alpha)*u_out;
IF u_out>=300 THEN
   u_out:=300;
ELSIF u_out<=0 THEN
   u_out:=0;
END_IF;
END_ALGORITHM
```

Código 11. Algoritmo 'Calculo' del bloque 'Filtro_tau'

5.3.3 Realización de simulaciones

Una vez se tienen configurados ambos entornos, el procedimiento a seguir para realizar las simulaciones es el siguiente:

1. Se ejecuta el programa de EAE. Los iconos del campo 'Estado', de la vetana 'Implementación y diagnóstico deberán estar tal y como se mostró en la Figura 31.

2. En Matlab, se ejecuta en primer lugar el script ‘conexión.m’, para establecer la conexión con el servidor de EAE. Se deberá tener un mensaje en la ventana de comandos como el de la Figura 34, indicando que se ha establecido con éxito la conexión.
3. En Simulink, después de establecer la duración de la simulación, se deberá pulsar el botón ‘Run’ situado en la parte superior de la interfaz, en la ventana ‘SIMULATION’. Es importante recordar que para que los bloques ‘Interpreted Matlab Function’ reconozcan las funciones de lectura y escritura, sus respectivos scripts debene estar situados en el mismo directorio que el archivo del modelo de Simulink ‘electrolizador_Samuel_eae.slx’.
4. Como paso adicional, se pueden visualizar los resultados en tiempo de simulación mediante los ‘scopes’ y ‘displays’ de Simulink, o con la función ‘Inspeccionar’ de EAE (Fig. 36).
5. Por último, uns vez finalizada la simulación, se pueden graficar los resultados de la misma, con los comandos ‘plot’ y ‘subplot’ de Matlab, obteniendo gráficas como la de la Figura 43.

6 RESULTADOS Y ANÁLISIS

En este capítulo se muestran los reultados de las simulaciones realizadas controlando el sistema desde EAE. Se compararán con las simulaciones del sistema controlado desde Matlab y se realizarán simulaciones más enfocadas en un caso real, buscando mejorar la eficiencia del controlador en la medida de lo posible.

Para empezar, se realiza la msima simulación que la descrita al final del Capítulo 5.2, en la Figura 42.

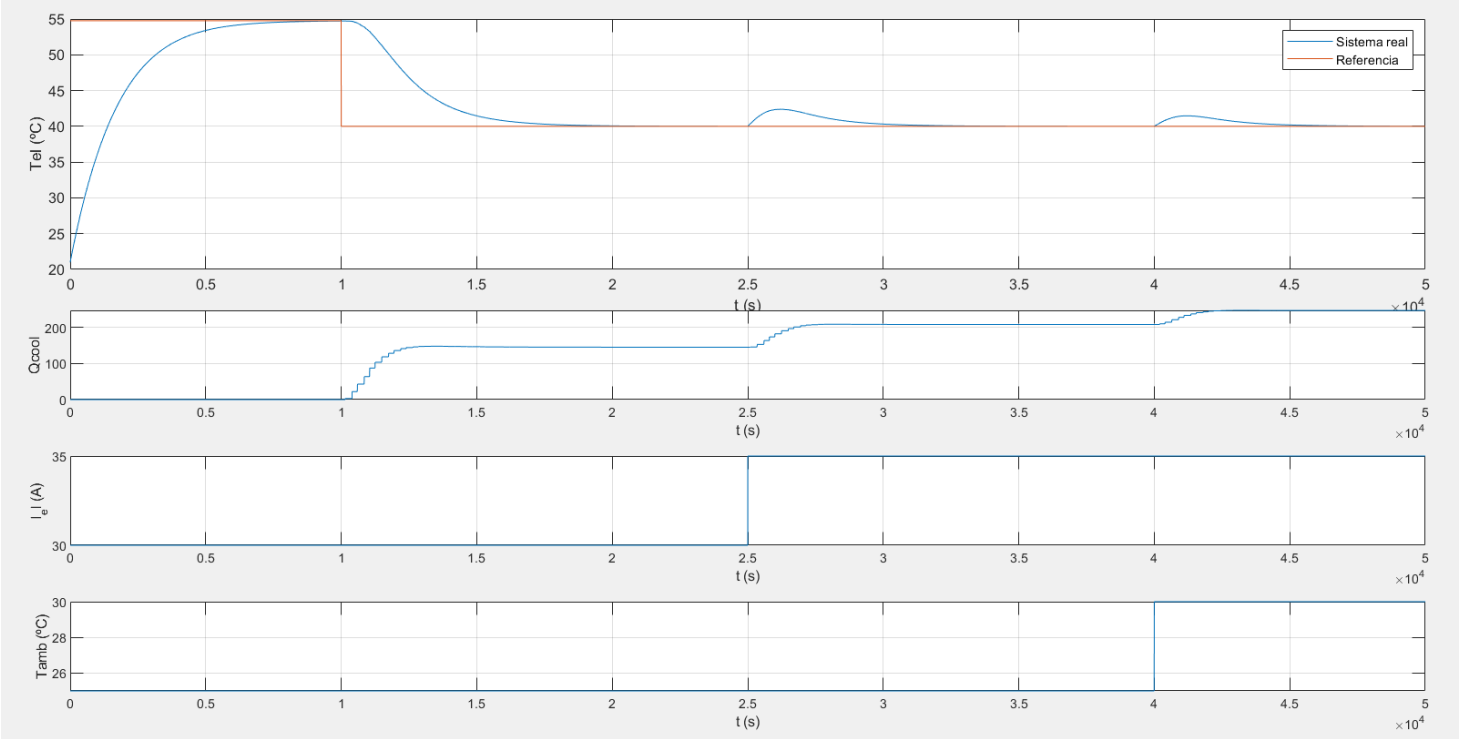

Figura 62. Resultados de la simulación 1 en EAE

Comparando ambas gráficas, se puede apreciar un enorme parecido entre ambas simulaciones, cumpliendo con las especificaciones de control en todo momento; por lo que se puede afirmar que el controlador ha sido implementado con éxito en EAE.

No obstante, se decidió ajustar los parámetros del PI, buscando la máxima eficiencia posible. Se aumentaron los parámetros *kp* y *ki*, buscando una respuesta más rápida; evitando sobreoscilación, saturación, y por supuesto, inestabilidad. Los nuevos valores para estos parámetros serán: ***kp* = -12.5** y ***ki* = -0.02**.

Tras realizar el mismo experimento con los nuevos parámetros se obtienen los siguientes resultados:

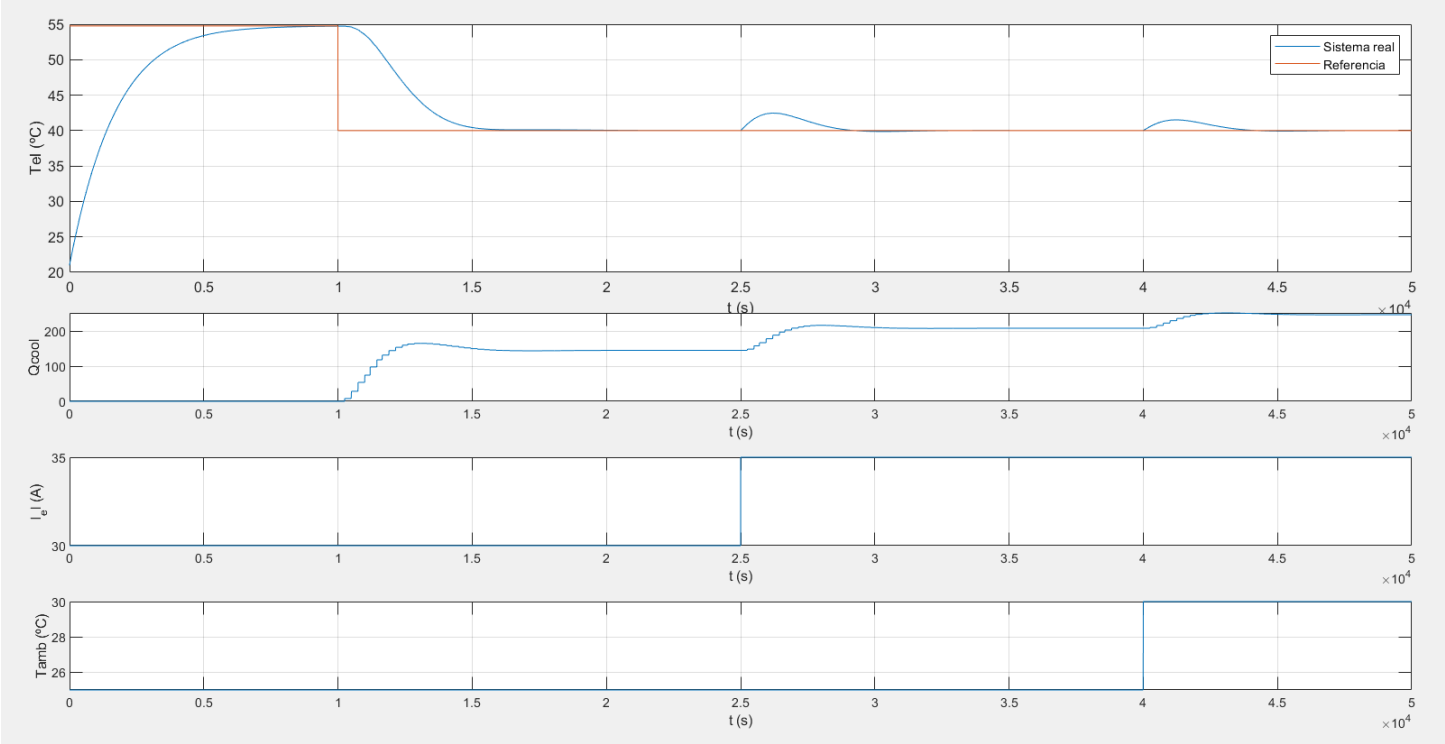

Figura 63. Resultados de la simulación 2 en EAE

Comparando con los resultados obtenidos anteriormente (Fig. 62), se puede apreciar una mejora en la velocidad de respuesta tanto en el seguimiento de la referencia como en el rechazo de perturbaciones.

Se realizaron otras simulaciones para ver como respondía el sistema a más cambios de referencia y perturbaciones (Fig. 64), y para forzar la saturación de la señal de controlar y comprobar el funcionamiento del anti-windup (Fig. 65).

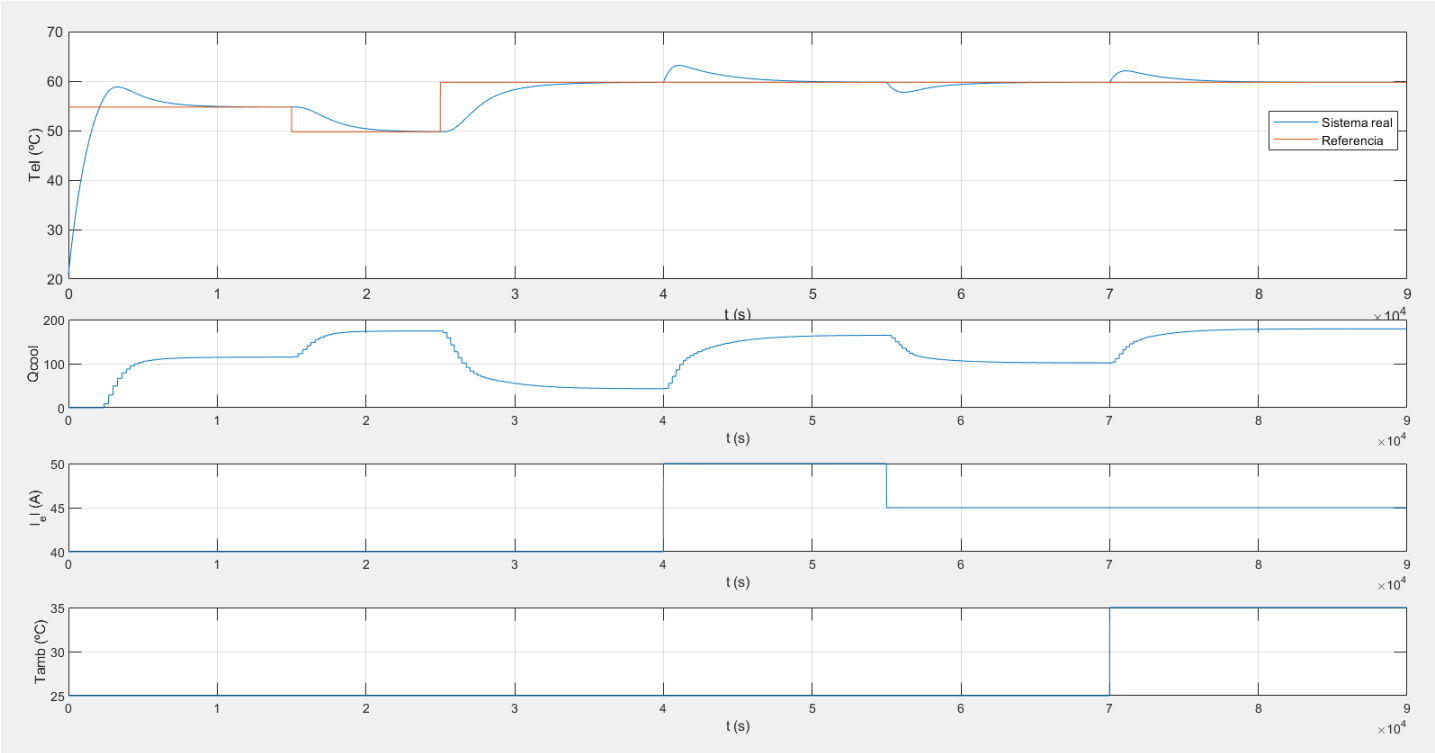

Figura 64. Resultados de la simulación 3 en EAE

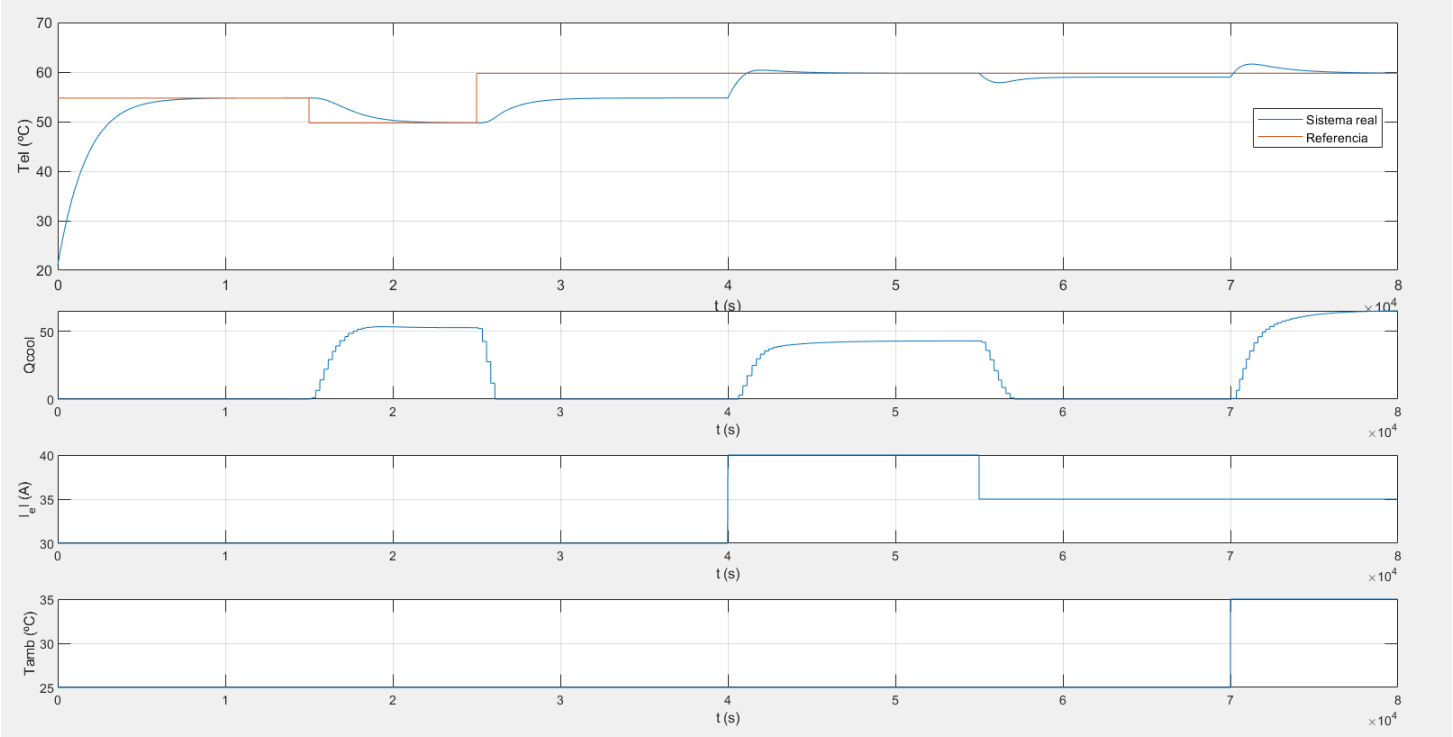

Figura 65. Resultados de la simulación 4 en EAE

Por último, se realizó una simulación con varias variaciones en la perturbación de intensidad de entrada del electrolizador (Fig. 67), simulando caso real.

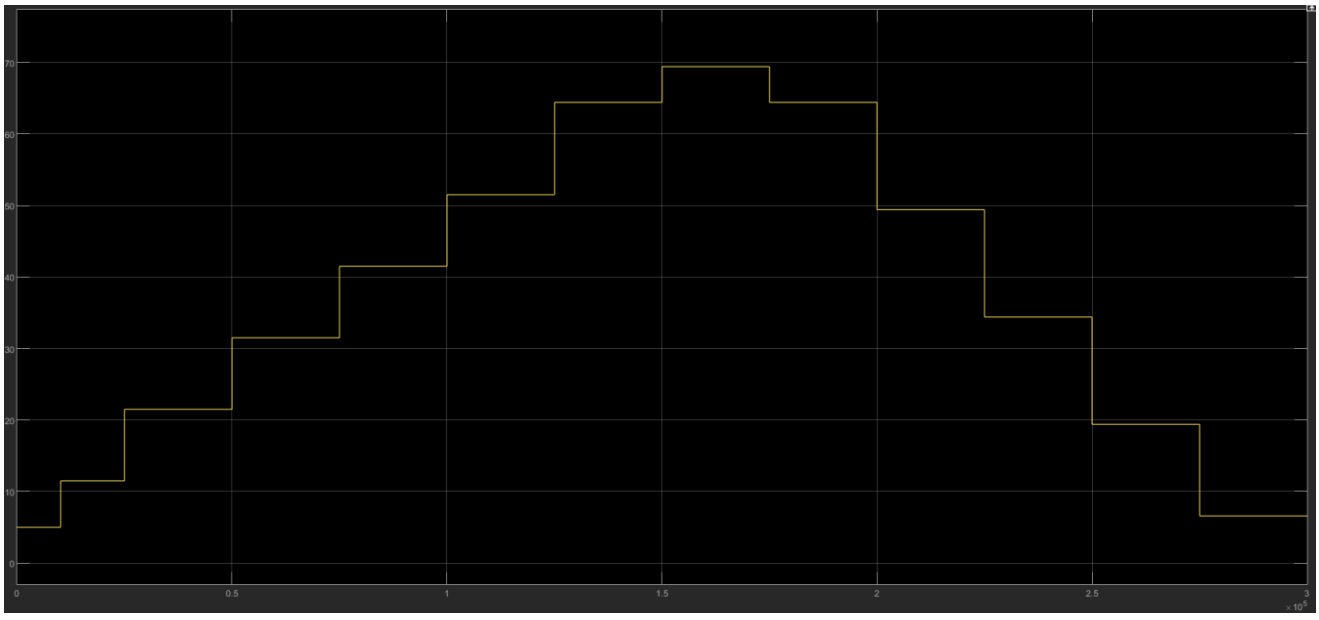

Figura 66. Valores de la perturbación *I_el* en la simulación 5

Se realizaron algunos cambios de referencia para evitar instantes en los que la señal de control satura. Esta saturación es provocada por valores extremos de la perturbación y por la capacidad del sistema de refrigeración, por lo que es totalmente natural y no se considera un error de diseño.

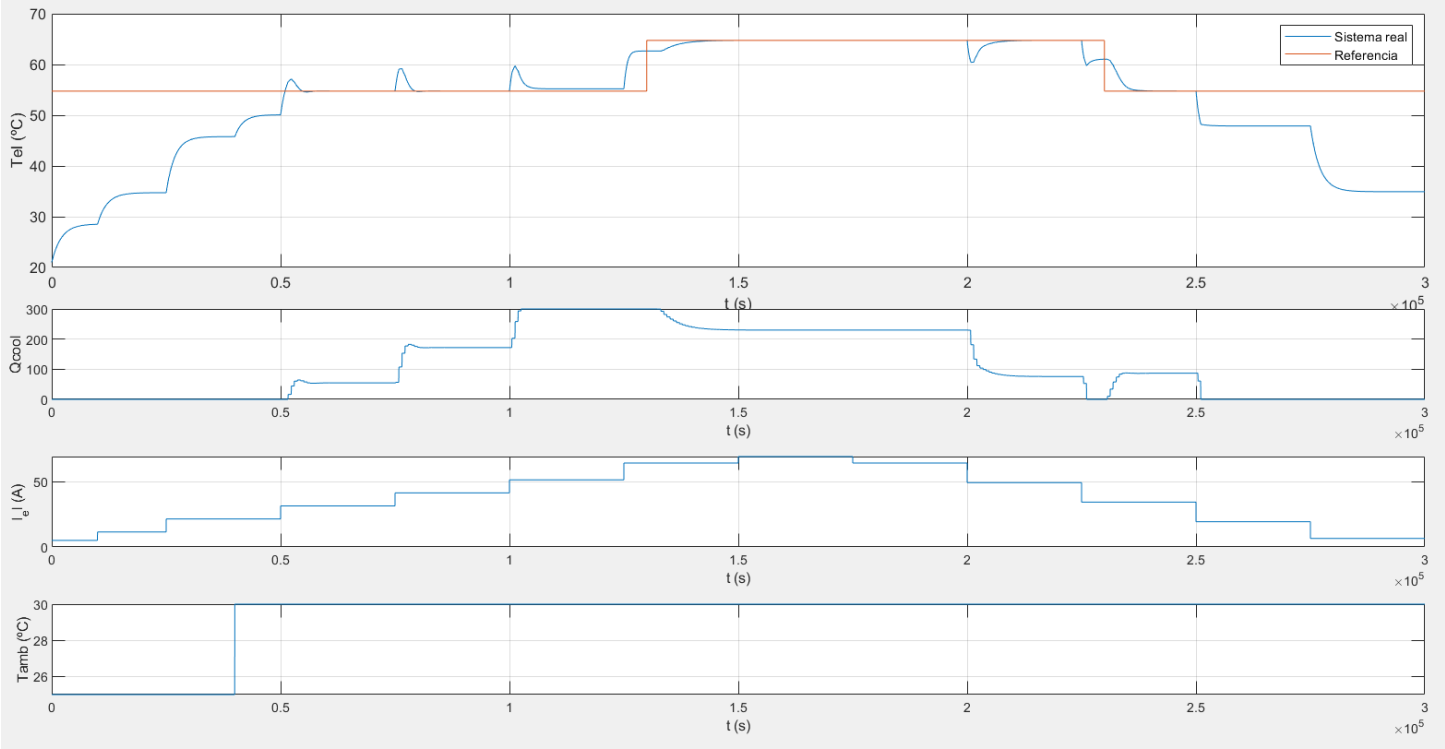

Figura 67. Resultados de la simulación 5 en EAE

Tras la realización de diversas simulaciones, se puede concluir que el objetivo del proyecto ha sido alcanzado satisfactoriamente. Se logró controlar la temperatura del sistema de manera eficiente, manteniéndola dentro de los límites permitidos por la capacidad de refrigeración disponible. Además, los tiempos de respuesta obtenidos resultaron adecuados para un proceso térmico industrial. Estos resultados validan la viabilidad y eficacia de la norma IEC 61499 para su aplicación en el control de sistemas industriales.

7 CONCLUSIONES Y TRABAJOS FUTUROS

En el presente proyecto se ha desarrollado un sistema de control para la temperatura de un electrolizador basado en la norma IEC 61499, aprovechando su enfoque orientado a eventos y su capacidad para la implementación de sistemas distribuidos. La arquitectura propuesta ha permitido validar el funcionamiento del controlador en un entorno de simulación, demostrando su potencial para gestionar de forma modular y escalable procesos industriales relacionados con la producción de hidrógeno.

Los resultados obtenidos evidencian que la norma IEC 61499 ofrece una solución flexible para la integración y coordinación de múltiples dispositivos, permitiendo una mayor adaptabilidad frente a cambios en la topología del sistema. No obstante, la validación se ha limitado a un entorno virtual, por lo que su implementación en un contexto real podría presentar nuevos desafíos técnicos y de integración.

En cuanto a trabajos futuros, se plantean las siguientes líneas de desarrollo:

- **Implementación en planta real**: Migrar el controlador desde el entorno de simulación a un electrolizador físico, evaluando su rendimiento, fiabilidad y robustez frente a condiciones operativas reales.
- **Control de flujos de hidrógeno y oxígeno**: Incorporar estrategias avanzadas de control para regular con precisión los caudales de hidrógeno y oxígeno, optimizando la eficiencia energética y la seguridad del proceso.
- **Seguridad en sistemas distribuidos bajo IEC 61499**: Profundizar en el análisis e implementación de mecanismos de seguridad que permitan prevenir accesos no autorizados, fallos de comunicación o comportamientos anómalos, considerando tanto la ciberseguridad como la seguridad funcional.

En resumen, el trabajo desarrollado constituye una base sólida para la aplicación de la norma IEC 61499 en el control de electrolizadores, y abre la puerta a futuras investigaciones orientadas a su implementación industrial segura y eficiente.

REFERENCIAS

[1] V. Vyatkin, «The IEC 61499 standart and it semantics,» *IEEE Industrial Electronics Magazine,* vol. 3, nº 4, pp. 40-48, 2009.

[2] A. Zoitl y R. Lewis, Modelling control systems using IEC 61499, vol. 2, Londrés: Control Engineering Series 95, The Institution of Electrical Engineers, 2014.

[3] IEA, «Global Hydrogen Review,» 2022. [En línea]. Available: https://www.iea.org/reports/global-hydrogen-review-2022.

[4] A. Buttler y H. Splietthof, «Current status of water electrolysis for energy storage, grid balancing and sector coupling via power-to-gas and power-to-liquids: A review,» *Renewable and Sustainable Energy Reviews,* vol. 82, pp. 2440-2454, 2018.

[5] RAE, «Diccionario de la lengua española | Electrólisis,» [En línea]. Available: https://dle.rae.es/electr%C3%B3lisis?m=form. [Último acceso: 29 Junio 2025].

[6] P. Ramírez Pons, Automatización de un electrolizador industrial, Sevilla: Universidad de Sevilla | Trabajo Fin de Máster, 2024.

[7] J. M. Martín Arcos y D. M. F. Santos, «The hydrogen color spectrum: Techno-economic analysis of the available technologies for hydrogen production,» *Gases,* vol. 3, nº 1, p. 25, 2023.

[8] Hydrogen Europe, «The Colors of Hydrogen,» [En línea]. Available: https://hydrogeneurope.eu/in-a-nutshell/. [Último acceso: 29 Junio 2025].

[9] M. Yu, K. Wang y H. Vredenburg, «Insights into low-carbon hydrogen production methods: Green, blue and aqua hydrogen,» *International Journal of Hydrogen Energy,* vol. 46, pp. 21261-21273, 15 Junio 2021.

[10] K. Zeng y D. Zhang, «Recent progress in alkaline water electrolysis for hydrogen production and applications,» *Progress in Energy and Combustion Science,* vol. 36, nº 3, pp. 307-326, Junio 2010.

[11] F. Barbir, «PEM electrolysis for production of hydrogen from renewable energy sources,» *Solar Energy,* vol. 78, nº 5, pp. 661-669, Mayo 2005.

[12] M. Laguna-Bercero, «Recent advances in high temperature electrolysis using solid oxide fuel cells: A review,» *Journal of Power Sources,* vol. 203, pp. 4-16, 1 Abril 2012.

[13] L. Qihao et al., «Anion Exchange Membrane Water Electrolysis: The Future of Green Hydrogen,» *The Journal of Physical Chemistry C,* vol. 127, nº 17, 28 Marzo 2023.

[14] F. L. Jiménez Sáez, Evaluación técnica y económica del uso de hidrógeno verde en aplicaciones para la

industria y desplazamiento de combustible fósil, Universidad de Chile | Tesis Pregrado, 2020.

[15] IEA, «The Future of Hydrogen: Seizing today's opportunities,» 2019. [En línea]. Available: https://www.iea.org/reports/the-future-of-hydrogen.

[16] IRENA, «Green Hydrogen for Industry: A Guide to Policy Making,» 2020. [En línea]. Available: https://www.irena.org/publications/2022/Mar/Green-Hydrogen-for-Industry.

[17] U. E. Jiménez-Ocampo et al., «Feedback control strategy for optimizing biohydrogen production from organic solid waste in a discontinuous process,» *International Journal of Hydrogen Energy,* vol. 46, nº 72, 19 Octubre 2021.

[18] P. Antsaklis y J. Bailleul, «Special Issue on Technology of Networked Control Systems,» *Proceedings of the IEEE,* vol. 95, nº 1, pp. 5-8, 2007.

[19] S. Guellouez et al., «Designing Efficient Reconfigurable Control Systems Using IEC61499 and Symbolic Model Checking,» *IEEE Transactions on Automation Science and Engineering,* vol. 16, nº 3, pp. 1110-1124.

[20] Schneider Electric, «EcoStruxure Automation Expert: Introduction to IEC61499».

[21] E. Monroy Cruz, L. R. García Carrillo y L. A. Cruz Salazar, «Comparación de estándares IEC 61131-3 e IEC 61499 para implementar sistemas de control distribuido,» XXVIII Congreso Internacional Anual de la SOMIM, 2022.

[22] Schneider Electric, «EcoStruxure Automation Expert: Basic Function Blocks».

[23] Schneider Electric, «EcoStruxure Automation Expert: Composite Function Blocks».

[24] M. V. García Sánchez, Metodologías para el diseño de sistemas de control y automatización industrial basados en IEC 61499, Bilbao: Universidad del País Vasco | Tesis doctoral, 2018.

[25] Schneider Electric, «Schneider Electric presenta EcoStruxure Automation Expert, el primer sistema de automatización industrial centrado en el software,» Interempresas, 2022. [En línea]. Available: https://www.interempresas.net/Robotica/Articulos/397741-Schneider-Electric-presenta-EcoStruxure-Automation-Expert-primer-sistema-automatizacion.html. [Último acceso: 18 Julio 2025].

[26] J. Prieto Navarro, Control de sistema industrial mediante comunicación OPC UA entre EcoStruxure Automation Expert y SIMULINK, Sevilla: Universidad de Sevilla | Trabajo de Fin de Grado, 2025.

[27] M. Álvarez, «Una introducción a OPC UA,» 2023. [En línea]. Available: https://www.masoutodev.com/introduccion-a-opc-ua/. [Último acceso: 18 julio 2025].

[28] Paessler The Monitoring Experts, «IT Explained: OPC UA,» [En línea]. Available: https://www.paessler.com/es/it-explained/opc-ua. [Último acceso: 18 Julio 2025].

[29] L. H. Yoong, P. S. Roop, Z. E. Bhatti y M. M. Y. Kuo, Model-Driven Design Using IEC 61499. A Synchronous Approach for Embedded and Automation Systems, 1ª ed., Springer International Publishing, 2015.

[30] M. Mora, C. Bordons, E. López, M. A. Ridao, F. Isorna y J. J. Caparrós, «Development and experimental validation of the dynamic model of an electrolyser for its integration with renewable generation,» 2023

IEEE International Conference on Energy Technologies for Future Grids, Wollongong, Australia, 2023.

[31] Universal Automation. https://universalautomation.org

GLOSARIO[5]

AEM	Anion Exchange Membrane	10
BFB	Basic Function Block	13
CAT	Component Automation Type	23
CFB	Composite Function Block	18
EAE	EcoStruxure Automation Expert	2
ECC	Execution Control Chart	14
FB	Function Block	13
GEI	Gases de Efecto Invernadero	5
HMI	Human-Machine Interface	20
IEA	International Energy Agency	2
IEC	International Electrotechnical Commission	1
IT/OT	Information Technologies/Operational Technologies	20
MQTT	Message Queuing Telemetry Transport	20
OPC UA	Open Platform Communications Unified Architecture	2
PAS	Publicity Available Specification	1
PEM	Proton Exchange Membrane	2
PID	Proportional Integral Derivative	42
PKI	Public Key Infrastructure	21
PLC	Programmable Logic Controller	1
SCADA	Supervisory Control And Data Acquisition	23
SIFB	Service Interface Function Block	18
SMR	Steam Methane Reforming	4
SOA	Service Oriented Arquitecture	21
SOEC	Solid Oxide Electrolyzer Cell	9

[5] Orden alfabético, con la página de primera aparición