Universidad de Oviedo

Memoria del Trabajo Fin de Máster realizado por

OSCAR ANDRÉS BRAÑA

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

Diseño y programación del sistema de control de una turbina mediante programación orientada a objetos

Julio 2016

Tutor académico
Felipe Mateos Martín
Tutor de empresa
Jose Luis Pérez Quintas
Empresa
ITRESA – Ingeniería e Informática Industrial de Asturias S.L.

ÍNDICE DE LA MEMORIA

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

MEMORIA

1 INTRODUCCIÓN

1.1 Enunciado

El objetivo principal del presente proyecto es el diseño, montaje e implementación de un sistema de control y supervisión de una turbina generadora de energía eléctrica que funciona mediante tecnología undimotriz. Este tipo de tecnología permite la obtención de electricidad a partir de energía mecánica generada por el movimiento de las olas del mar, por lo que es considerada una forma de energía renovable

A grandes rasgos, el sistema realizará dos tareas principales: recoger y procesar la información procedente de la turbina y permitir la supervisión y el control de todo el sistema. Así pues, se pueden diferenciar dos partes principales dentro del sistema: la parte de adquisición y procesamiento de datos, en adelante *DSP,* y la parte de supervisión y control del sistema, en adelante *HMI*.

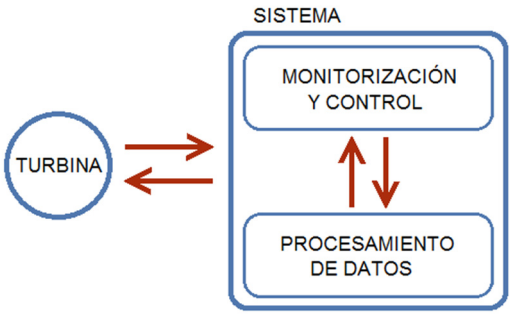


Imagen 1. Esquema general del sistema.

Las señales recibidas desde la turbina serán convenientemente adaptadas para poder ser procesadas por el sistema. Además, el sistema permitirá a los operarios de la instalación la monitorización de los datos adquiridos y un control de las funciones básicas de la turbina.

Con todo ello, a lo largo del presente documento se tratarán temas que se consideran clave en proyectos de desarrollo de software, como son la definición de objetivos y especificaciones, el proceso de modelado del sistema y la creación de los programas e interfaces que lo componen.

1.2 Alcance

Ante la variedad de opciones a la hora de afrontar el proyecto, se realizará un estudio en el que se valorarán las alternativas en materia de arquitectura de control, comunicaciones y estructuración del software, seleccionando la que mejor se adapte a las especificaciones y tecnología disponibles.

Como se ha dicho anteriormente, el sistema estará formado por dos partes, DSP y HMI, que cumplirán dos tareas bien diferenciadas. De cara a obtener un sistema, en el que la robustez, la sencillez y la eficiencia estén lo más presentes posible, se valorarán una serie de opciones, a nivel de arquitectura y comunicaciones, con el único fin de comprobar cuál es la óptima tanto desde el punto de vista práctico como económico.

Una vez establecida la opción más apropiada, se instalarán y configurarán todos los equipos necesarios para llevar a cabo la implementación del sistema, así como el diseño y creación de los distintos programas necesarios para el control y supervisión de la instalación.

La parte de diseño y programación de los diferentes programas que van a formar parte del sistema final constituye la parte fundamental del presente proyecto, por lo que, con el fin de aplicar los conocimientos adquiridos, se pretende conseguir un código claro y fácilmente depurable, así como una interfaz gráfica amigable y sencilla de manejar desde el punto de vista del usuario final.

La integración y la correcta puesta en marcha de todos los componentes del sistema se considera trascendental, ya que permite comprobar si lo diseñado sobre el papel se puede llevar a la práctica. Además, constituye un paso más en la formación adquirida, ya que permite llevar a la realidad proyectos que en la etapa de estudios se quedaban en meras simulaciones.

Finalmente se propondrán posibles ampliaciones que, si bien no constituyen objeto del presente proyecto, sí merecen estar recogidas en este documento con el fin de proporcionar ideas para el desarrollo de futuros proyectos similares.

2 ESPECIFICACIONES

Establecer cuáles son los objetivos que se pretenden cumplir, el conocer cuáles son las alternativas que pueden conducir a una solución acerada y el organizar temporalmente los trabajos a realizar son tareas que es conveniente realizar previamente a cualquier proyecto. El propósito de estas tareas no es otro que conseguir un ahorro importante en el tiempo empleado en la subsanación de errores que con dicha planificación no se hubieran producido. Se trata, pues, de un punto clave en el que dedicar una parte importante del tiempo de trabajo.

2.1 Especificaciones generales

Como especificaciones generales se consideran las siguientes:

- Se ha de realizar un registro de los datos recibidos de la turbina en una base de datos.
- El usuario debe poder visualizar curvas de tendencia de cualquiera de los parámetros de los que se registran datos.
- Las alarmas producidas en la turbina han de ser fácilmente comprobables en cualquier momento.
- Se impondrá la sencillez y usabilidad de los interfaces de usuario desarrollados.

2.2 Especificaciones técnicas

En cuanto al sistema de adquisición de datos:

- Las diferentes señales de la turbina se enviarán al cuadro de control para su tratamiento. El equipo de procesamiento instalado en el cuadro generará una trama cada 100 milisegundos con los datos correspondientes, que se enviará a través del puerto serie al equipo de tratamiento de tramas.
- De cara al diseño del sistema debe tenerse en cuenta que la sala de control y el emplazamiento de la turbina se encuentran separadas.

En cuanto al sistema de supervisión y control:

- El sistema debe ser sencillo, de fácil manejo e intuitivo para el usuario.
- De cara a la elaboración de históricos de datos, debe permitirse la supervisión de varias curvas de valores al mismo tiempo de datos pasados.
- Cuando se produzca una alarma o un cambio en el estado de funcionamiento de la turbina el sistema mostrará un aviso.
- Se podrá realizar un sencillo control sobre la turbina mediante el envío de una serie de comandos que se detallan en la descripción del protocolo de comunicaciones.

2.3 Protocolos de comunicaciones

En este apartado se definen los protocolos de comunicación para los diferentes elementos del sistema. Para facilitar el desarrollo, toda comunicación seguirá el mismo patrón: una trama con caracteres de control de inicio y fin. Estos protocolos vienen dados como especificaciones ya que dependen de los equipos adaptadores de señal instalados en el cuadro de control.

Los equipos de adaptación enviarán la información procesada a la unidad central de adaptación, la cual se comunicará con el equipo de tratamiento de datos vía comunicación serie mediante cable *RS232*. La velocidad de transmisión será asíncrona de 9600 baudios, 8 bits, sin paridad y con 1 bit de stop. No se realizará control de flujo a través del hardware.

Para más información sobre el conexionado de estos equipos con los elementos de la turbina se pueden consultar los esquemas eléctricos correspondientes.

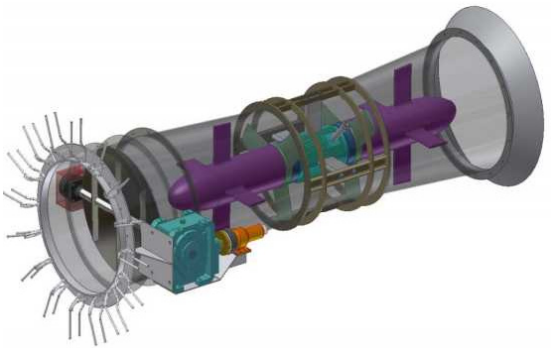


Imagen 2. Esquema de la turbina.

Comunicación turbina -> DSP

El *DSP* recibirá tramas de datos con una frecuencia de 10 Hz. La longitud de esta trama es de 28 bytes, más la longitud de las cabeceras de inicio y fin de trama que se utilicen, haciendo un total de 32 bytes. Los bytes 0, 1, 31 y 32 se corresponden con los siguientes caracteres de control de inicio y fin de trama:

- **DLE**: Data-Link-Escape (0001 0000≡10h).
- **STX**: Start-of-TeXt (0000 0010≡02h).
- **ETX**: End-of-TeXt. (0000 0011≡03h).

Byte	Descripción
0	DEL -> 10h
1	STX -> 02h
2	Velocidad generador (Parte alta)
3	Velocidad generador (Parte baja)
4	Presión en la cámara (Parte alta)
5	Presión en la cámara (Parte baja)

6	Posición del cilindro (Parte alta)
7	Posición del cilindro (Parte baja)
8	Potencia (Parte alta)
9	Potencia (Parte baja)
10	Vibraciones (Parte alta
11	Vibraciones (Parte baja)
12	Corriente iu (Parte alta)
13	Corriente iu (Parte baja)
14	Corriente iv (Parte alta)
15	Corriente iv (Parte baja)
16	Tensión VDC (Parte alta)
17	Tensión VDC (Parte baja)
18	Temperatura 1 (Parte alta)
19	Temperatura 1 (Parte baja)
20	Temperatura 2 (Parte alta)
21	Temperatura 2 (Parte baja)
22	Comando de posición de válvula (Parte alta)
23	Comando de posición de válvula (Parte baja)
24	Referencia de velocidad de giro máxima (Parte alta)
25	Referencia de velocidad de giro máxima (Parte baja)
26	Error 1
27	Error 2
28	Error 3
29	Estado
30	Reservado
31	DLE -> 10h
32	ETX -> 03h

Tabla 1. Trama de datos.

Los bytes del 2 al 25 son palabras de 2 bytes en formato entero sin signo. El resto de bytes, del 26 al 30, son palabras de un solo byte, donde cada bit de cada palabra tiene la siguiente codificación:

Palabra de estado:

Bit	**Descripción**
0	Velocidad limitada
1	Posición intermedia de la válvula; Activado (1) / Desactivado (0)
2	Estado de funcionamiento; Parado (1) / Funcionamiento normal (0)
3	Alarma; Estado de alarma (1) / Funcionamiento OK (0)
4	Estado del motor del cilindro; ON (1) / OFF (0)
5	Posición del cilindro; Extendido (1) / Recogido (0)
6	Electroimán;)1) / OFF (0)
7	Estado de la turbina; Girando (1) / Parada (0)

Tabla 2. Palabra de estado.

Palabra de error 1:

Bit	Descripción
0	Lectura anómala del sensor iu
1	Lectura anómala del sensor iv
2	Lectura anómala del sensor ir
3	Lectura anómala del sensor is
4	Lectura anómala del sensor vdc
5	Lectura anómala del sensor de vibraciones
6	Lectura anómala del sensor de posición del cilindro
7	Lectura anómala del sensor de presión

Tabla 3. Palabra de error 1.

Palabra de error 2:

Bit	Descripción
0	Lectura anómala del sensor de temperatura 1
1	Lectura anómala del sensor de temperatura 2
2	Sobrecorriente en fase u
3	Sobrecorriente en fase v
4	Sobrecorriente en fase r
5	Sobrecorriente en fase s
6	Vibración excesiva
7	Presión en la cámara excesiva

Tabla 4. Palabra de error 2.

Palabra de error 3:

Bit	Descripción
0	Velocidad de la turbina excesiva
1	Sobretensión en el bus DC
2	Fallo en el encoder
3	Exceso de temperatura 1
4	Exceso de temperatura 2
5	-
6	-
7	-

Tabla 5. Palabra de error 3.

Comunicación DSP -> turbina

Se podrán enviar órdenes en forma de comandos a la turbina en cualquier momento. Estas órdenes tendrán una longitud de 3 bytes más con los caracteres de control, haciendo un total de 7 bytes.

En el primer byte se especifica el tipo de orden. En el segundo y tercer bites se mandará la parte alta y la parte baja, respectivamente, de la referencia correspondiente a la orden en

formato entero sin signo de 16 bits. En caso de que la orden no lleve asociado ninguna referencia, los bytes 2 y 3 se rellenarán con ceros.

0	1	2	3	4	5	6
DLE	**STX**	Orden	Referencia (parte alta)	Referencia (parte baja)	**DLE**	**ETX**

Tabla 6. Formato de trama HMI -> DSP.

Órdenes previstas:

- Funcionamiento normal = 0x00
- Parar turbina = 0xFF
- Arrancar turbina = 0xAA
- Abrir válvula = 0xF0
- Velocidad de giro máxima = 0x0A + referencia
- Posición de la válvula = 0xA0 + referencia

3 VALORACIÓN Y ESTUDIO DE ALTERNATIVAS

Con el fin de dar cumplimiento a las especificaciones pedidas, se han valorado una serie de opciones, incluyendo o no los equipos proporcionados, que pueden ser más o menos válidas en función de los costes totales y la complejidad y funcionalidad del sistema. El fin de esta comparativa no es más que llegar a la solución que mejor aproveche los recursos y sea lo más eficiente posible desde el punto de vista económico y funcional.

3.1.1 Alternativas en la arquitectura del sistema

- Un solo equipo para la comunicación con la turbina y para la instalación del interface HMI.
- Dos equipos independientes. En uno correrá el software del DSP y en el otro el interface HMI.

La utilización de un solo equipo en el sistema simplifica la instalación sustancialmente, pues con uno solo no sería necesario el planteamiento de alternativas de modos de comunicación, ya que todos los programas de control y supervisión se encontrarían en el mismo equipo. De igual manera, el reducir el número de equipos repercute directamente sobre los costes de la instalación. Por otro lado, el dejar en manos del mismo equipo las labores de adquisición, control y supervisión, puede ocasionar problemas de cara al mantenimiento o al mismo tratamiento de los datos. Las propias tareas del software HMI podrían ralentizar las operaciones de adquisición de datos, pudiendo llegar a perderse información sobre el estado actual de la turbina.

Imagen 3. Arquitectura con un solo equipo.

El separar la parte de adquisición y procesamiento de la parte de supervisión hace que estos problemas de solapamiento de recursos no existan, ya que el software correspondiente a cada una de las partes corre en diferentes equipos. Además, con esta división de tereas, se podría incluir más de un equipo HMI en el sistema, en forma de clientes del equipo DSP. Por otro lado, el coste de un equipo para el HMI, un PC de sobremesa, es relativamente bajo en comparación con la pérdida de datos relativos a información del estado y errores en la turbina.

Imagen 4. Arquitectura con dos equipos.

La realización de tareas de mantenimiento y actualizaciones del software, se simplifican considerablemente dividiendo el sistema en dos equipos. Sin embargo, el sistema se hace más complejo y debe establecerse un modo de comunicación entre ambos equipos que permita el funcionamiento del sistema de una forma fluida y sin errores. Además, debe establecerse cómo van a sincronizarse ambos equipos y preverse cómo va a actuar el sistema, por ejemplo, ante una pérdida en la comunicación entre ambos equipos o ante la desconexión de uno de ellos del sistema.

3.1.2 Alternativas de arquitectura de bases de datos

- Una única base de datos instalada en el equipo DSP.
- Una única base de datos instalada en el equipo HMI.
- Utilización de un equipo independiente como servidor de bases de datos.
- Dos bases de datos instaladas en un equipo.
- Dos bases de datos independientes, instaladas en dos equipos.

Al igual que en las alternativas planteadas en cuanto a la arquitectura del sistema, el instalar una única base de datos en el sistema lo simplifica de forma considerable, ya que toda la información necesaria se encontraría en un único sitio y no existirían problemas de datos duplicados o de sincronización. Cabe decir que se hablará en todo momento de bases de datos relacionales.

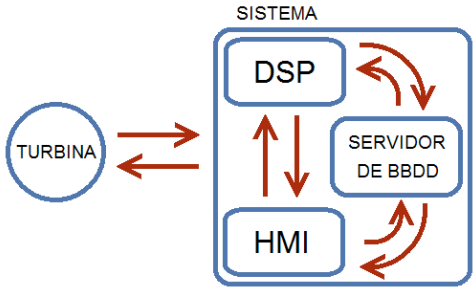


Imagen 5. Arquitectura con base de datos independiente.

Sin embargo, en este tipo de sistema, el disponer de una sola base de datos puede ser un problema si se producen fallos, tales como un borrado accidental de la misma o un fallo en el equipo que la contiene, con la consiguiente pérdida de datos anteriores y nuevos.

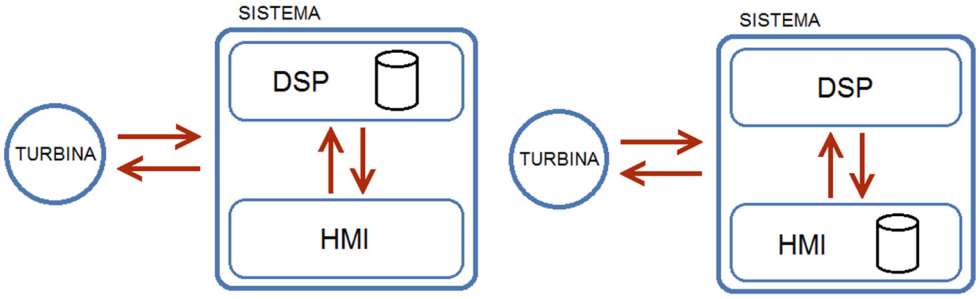


Imagen 6. Arquitecturas con una base de datos integrada.

El disponer de dos bases de datos permite tener un respaldo de los datos obtenidos hasta el momento, ya que ante repentinas pérdidas de información en una de las bases de datos se dispone de la otra como respaldo. Por ello, la sincronización entre bases de datos es crítica. No deben duplicarse datos al realizar los respaldos de información y debe elegirse un período de sincronización coherente con el sistema. Por otro lado, el instalar dos bases de datos duplica el espacio requerido para el almacenamiento.

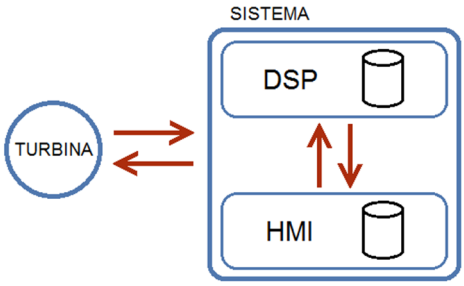


Imagen 7. Arquitectura con dos bases de datos integradas.

Instalando ambas bases de datos en el mismo equipo permitiría disponer de un único equipo en el sistema, pero no solucionaría el problema de pérdida completa de información ante fallos en dicho equipo. Sin embargo, instalando ambas bases de datos en equipos diferentes se reduciría a la mitad el riesgo de una pérdida completa de datos, únicamente se perdería información en el espacio entre sincronizaciones. De igual manera, en este caso se incrementa la complejidad en la configuración de los equipos y aparece la necesidad de decidir el protocolo y el modo de comunicación entre ambas bases de datos.

3.1.3 Alternativas de comunicaciones

- Red local inalámbrica.
- Red local cableada.

En cuanto a la forma de comunicación entre los posibles equipos que se pueden incluir en el sistema, existen distintas formas de crear una red local que establezca una comunicación entre todos ellos.

Es posible crear una red local inalámbrica en la que no sean necesarios cables que conecten los equipos. Esto reduciría la complejidad del sistema en lo que al montaje se refiere, ya que es posible instalar los equipos en cualquier parte dentro del alcance de la red. Aunque los rangos de distancias en los que trabajan estos equipos son cada vez mayores, tienen sus limitaciones. La gran desventaja de la conexión inalámbrica es su velocidad y fiabilidad. La transmisión sin medio físico hace que existan fluctuaciones en la transmisión de paquetes de datos, pudiendo llegar a perderse algunos de ellos. Además, los equipos necesarios para crear una comunicación de este tipo suelen necesitar una configuración compleja y tener un coste elevado.

Por otro lado, la red cableada aprovecha todo el ancho de banda disponible, pudiendo llegar a altas velocidades en la transmisión de datos. Además, la red cableada aporta mayor

seguridad al sistema ya que, al ser una red cerrada, solo es posible el acceso a la misma mediante conexión directa. Sin embargo, la instalación de una red de este tipo puede complicarse si las distancias son muy largas, siendo necesario instalar repetidores que amplifiquen la señal cada cierta distancia.

Comparativa velocidad redes

Interfaz	Velocidad	Alcance
Ethernet 10 Base T	10 Mbps – 1.25 MBps	100 m (sin repetidores)
Ethernet 100 Base T	100 Mbps – 12.5 MBps	
Ethernet 1000 Base T (Gigabit)	1000 Mbps – 125 MBps	
Ethernet 10000 Base T (10 Gigabit)	10000 Mbps – 1.25 GBps	
Wireless 802.11b	11 Mbps – 1.3 MBps	30 m
Wireless 802.11g	54 Mbps – 6.75 MBps	15 m
Wireless 802.11n	300 Mbps – 37.5 MBps 600 Mbps – 75 MBps	70 m

Tabla 7. Comparativa velocidad en redes.

3.1.4 Alternativas de lenguajes de programación

Teniendo en cuenta que los programas de control van a correr sobre un sistema operativo Windows, en cuanto a los lenguajes de programación existen infinidad de alternativas que permiten crear interfaces HMI de mayor o menor complejidad:

- Lenguajes de más bajo nivel, como el C puro.
- Lenguajes de más alto nivel que permiten la Programación Orientada a Objetos (POO), como el C++, C# y Visual Basic
- Lenguajes orientados al cálculo como MATLAB, Oracle o LabView.

Todos ellos tienen sus ventajas e inconvenientes en cuanto a versatilidad, complejidad de sintaxis, librerías, documentación…, por lo que, en principio, cualquiera de ellos podría servir para el desarrollo de los programas que son propósito del presente proyecto.

3.2 Especificaciones de la solución adoptada

Teniendo en cuenta los argumentos expuestos en el apartado anterior, se ha llegado a las siguientes conclusiones:

- En cuanto a la arquitectura del sistema, se optará por instalar **dos equipos independientes**. Se ha tomado esta decisión teniendo en cuenta la versatilidad y robustez de aporta el mantener separadas la parte de DSP de la de HMI. Además, la posibilidad de ampliar el sistema añadiendo más estaciones HMI hace que esta sea la opción más conveniente.

- En cuanto a arquitectura de bases de datos, se ha decidido disponer de **dos bases de datos, una en cada equipo**. Esta decisión se toma en base al punto anterior ya que, disponiendo de dos equipos independientes, lo más razonable es separar también las bases de datos y sincronizarlas cada cierto tiempo, obteniendo así un respaldo de los datos en un equipo independiente.
- En cuanto al modo de comunicación entre los dos equipos, se establecerá una **red local cableada**. Atendiendo a la mayor seguridad frente a intrusiones y fiabilidad ante pérdidas de paquetes de datos de la misma, teniendo en cuenta las distancias que deben salvarse y, por la sencillez de su instalación y su bajo coste respecto a los equipos necesarios para crear la red inalámbrica, se ha optado por esta alternativa.
- En cuanto al lenguaje para el desarrollo de los programas de control, se ha optado por el **C#** en ambos. Este lenguaje soporta el desarrollo de aplicaciones utilizando el **framework .NET de Microsoft**, lo que permite la portabilidad sin problemas entre distintos equipos con distintos sistemas operativos de Microsoft y, en algunos casos, incluso de otras compañías. Además, se dispone de una licencia de pago de una librería con controles para gráficas que permitirán el desarrollo de un interface HMI más intuitivo y amigable utilizando el framework .NET. Este punto puede haber sido determinante a la hora de la elección del lenguaje de programación para el desarrollo de los programas. Por otro lado, y no menos importante, es la gran cantidad de ayuda y documentación disponible sobre este lenguaje y las librerías propias de Microsoft.

Con todo ello, el diseño de la solución final adoptada para el sistema se puede esquematizar como sigue:

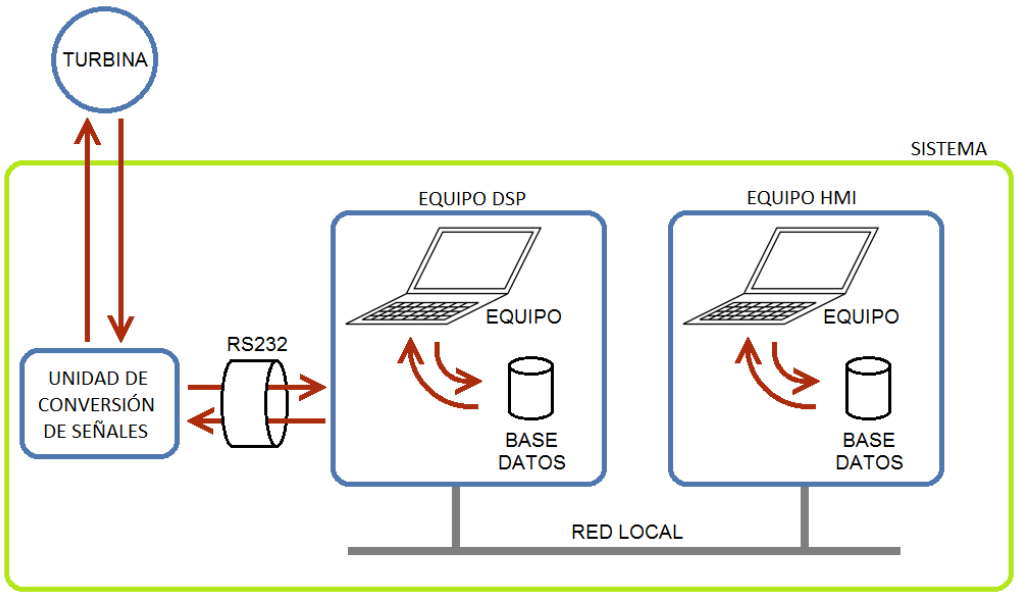


Imagen 8. Esquema general de la solución adoptada.

4 METODOLOGÍA Y ORGANIZACIÓN TEMPORAL DEL PROYECTO

En cuanto a la organización del proyecto, se han establecido una serie de hitos o fases de trabajo, recogidos en la siguiente tabla, teniendo en cuenta una jornada laboral de 8 horas y 5 días semanales:

Fase introducción	
Estudios previos sobre programación orientada a objetos y librerías	15 días
Selección y pedido de los materiales necesarios	3 días
Preparación de los equipos componentes del sistema	2 días
Instalación de las bases de datos	1 día
Fase DSP	
Creación de un protocolo propio para la comunicación entre la turbina y el DSP	1 día
Programación de una pequeña aplicación para la simulación de la turbina	3 días
Diseño y programación de la aplicación para la adquisición y procesamiento de datos	30 días
Fase HMI	
Creación de un protocolo propio para la comunicación entre el DSP y el HMI	1 día
Diseño y programación del interfaz HMI	30 días
Fase pruebas	
Instalación de los programas en los equipos correspondientes	2 días
Cableado provisional y pruebas en ITRESA	5 días
Instalación y puesta en marcha in situ	2 días
Fase documentación	
Memoria e ingeniería	20 días
Presupuestos	2 días
Esquemas eléctricos	3 días
Total días	120 días

Tabla 8. Organización temporal del proyecto.

5 BASE TEÓRICA PARA EL DISEÑO

En base a las especificaciones definidas y a las soluciones adoptadas para llevar a cabo el proyecto, se expone a continuación la base teórica necesaria para ayudar al seguimiento de las explicaciones sucesivas.

5.1 Programación Orientada a Objetos (POO)

La programación orientada a objetos (*POO*) es una filosofía de programación, con una teoría y metodología propias, que pretende abordar la creación de programas de una forma diferente a la que plantea la programación estructurada.

La POO plantea la descomposición del problema global en *objetos,* es decir, el programador debe intentar simular al máximo, desde un punto de vista conceptual, el escenario real del sistema que va a programar. Aparece entonces el concepto **objeto** como elemento o entidad que posee una serie de propiedades o **atributos** y es capaz de realizar un conjunto de acciones o **métodos**.

A modo de ejemplo: Un alumno, con su edad, DNI y nota media, puede estudiar, ir a clase y realizar exámenes; y un profesor, con su edad, DNI y departamento, puede dar clase, evaluar y tutorar. De esta manera, y siguiendo el paradigma de la *POO,* tendríamos el objeto *alumno*. Su *edad*, *DNI* y *nota media* serían sus atributos, y *estudiar*, *ir a clase* y *realizar exámenes* serían sus métodos. Por otro lado, tendríamos el objeto *profesor,* con sus atributos *edad*, *DNI* y *departamento,* y con sus métodos *dar clase, evaluar* y *tutorar*.

Como se puede observar, se ha identificado un objeto con un elemento que se diferencia por sus características y acciones. Por tanto, un objeto es la representación de un concepto, a nivel de programación, y que contiene toda la información necesaria para utilizarlo dentro del programa. También puede observarse como los objetos pueden estar relacionados entre sí. En el caso del ejemplo que nos ocupa, el profesor puede tutorar al alumno y a su vez el alumno puede acudir a las clases propuestas por el profesor.

Debido a que en el mundo real pueden existir varios objetos de un mismo tipo, de un modo más técnico se habla de los objetos como **clases**. De esta manera, y volviendo al ejemplo, todos los profesores de una universidad se podrían considerar dentro de la clase *profesor*, aun teniendo diferente edad, DNI o departamento. De un modo análogo, se podría hablar de la clase *alumno*. Puede decirse entonces, que una clase es una plantilla que define las características y acciones que un conjunto de elementos de un mismo tipo.

Con todo ello, a modo de resumen, se podría decir que los objetivos que persigue la Programación Orientada a Objetos se resumen en:

- **Abstracción:** proceso mental de extracción de las características esenciales de un elemento, tales como sus características y acciones.

- **Encapsulación:** proceso por el que se protegen las características esenciales de un elemento de cara a los demás.
- **Jerarquización:** proceso de estructuración por el que se produce la organización del conjunto total de elementos en grados de relación de unos con otros.
- **Modularidad:** proceso de descomposición de un sistema en un conjunto de elementos independientes con significado propio.

Como tal, la POO surgió a principios de los noventa y se fue desarrollando y evolucionando hasta tal punto que en la actualidad existen multitud de lenguajes basados en dicho paradigma. Entre ellos cabría destacar C++, C#, Java, JavaScript, Python, Visual Basic, Scala, Fortran, etc.

5.2 Lenguaje Unificado de Modelado (UML)

UML, acrónimo de *Unified Modeling Languaje*, es un lenguaje gráfico estándar para el modelado y desarrollo de sistemas. Permite la encapsulación de problemas complejos afrontándolos de forma más simplificada, por lo que por sus similitudes es la metodología de modelado más apropiada para el diseño de sistemas mediante programación orientada a objetos.

El punto de partida es la declaración del problema a resolver. Esta declaración, que proporciona una visión conceptual del sistema propuesto, se va definiendo con mayor precisión a lo largo del proceso de modelado mediante los tres siguientes pasos:

- **Modelado de casos de uso**: identificación de cómo se va a usar el sistema y lo que debe hacer como respuesta a ese uso. Esta información se presenta en forma de diagrama de casos de uso, en el que intervienen una serie de actores (quienes realizan las acciones) sobre unos determinados escenarios (lo que hace el sistema ante esa acción).
- **Modelado de clases**: identifica cada clase perteneciente al sistema, sus atributos y sus relaciones con otras clases. Esta información se representa en forma de diagramas de clases.
- **Modelado dinámico**: determina las acciones realizadas por cada clase y sobre qué clase o clases se realizan. Esta información se representa en forma de diagramas de comportamiento.

Al tratarse de un lenguaje estándar de modelado orientado a objetos, esto permite la traducción a cualquier lenguaje de este tipo, por lo que modelado y diseño pueden realizarse en fases diferentes, teniendo siempre en cuenta las posibles particularidades de cada lenguaje a la hora de llevar a código el modelo diseñado.

Así pues, el UML es un lenguaje para especificar modelos precisos, inequívocos y completos; para construir un sistema, a partir del mismo modelo, mediante diversos lenguajes

de programación; y para documentar de forma detallada toda la arquitectura del sistema, así como su funcionamiento.

Un modelo definido mediante UML consta de una serie diagramas que se podrían clasificar en dos tipos diferentes:

- Los **diagramas de estructura** muestran la estructura estática del sistema, la organización de las clases y las relaciones entre objetos. Dentro de este grupo se encuentran los *diagramas de clases* y los *diagramas de objetos,* entre otros.
- Los **diagramas de comportamiento** muestran la evolución dinámica del sistema. Capturan operaciones y cambios de estado internos de los elementos del sistema. Dentro de este grupo se encuentran los *diagramas de casos de uso*, los *diagramas de actividad* y los *diagramas de interacción*, entre otros.

La inclusión de todos los diagramas dentro del modelo no es obligatoria, pero con un mayor número de diagramas, la definición del sistema será más precisa. A continuación se describen los diagramas más utilizados y que más contribuyen a una definición precisa del sistema.

5.2.1 Diagrama de casos de uso

Representa las relaciones del sistema con su entorno. Muestra las necesidades que debe satisfacer el sistema en función de lo que se le demande en cada momento (caso de uso). Mediante este diagrama puede definirse con claridad los requerimientos funcionales del sistema de cara a las necesidades que plantea el cliente. De esta manera pueden definirse algunas clases, así como algunas de sus características.

Cada caso de uso responde a las acciones que se realizan sobre él mediante reacciones. Las acciones son desencadenadas por los actores del sistema. La reacción a dicha acción la recibe otro actor, que puede ser o no ser el mismo que la desencadenó.

En la *Imagen 9* puede verse un ejemplo de diagrama de casos de uso.

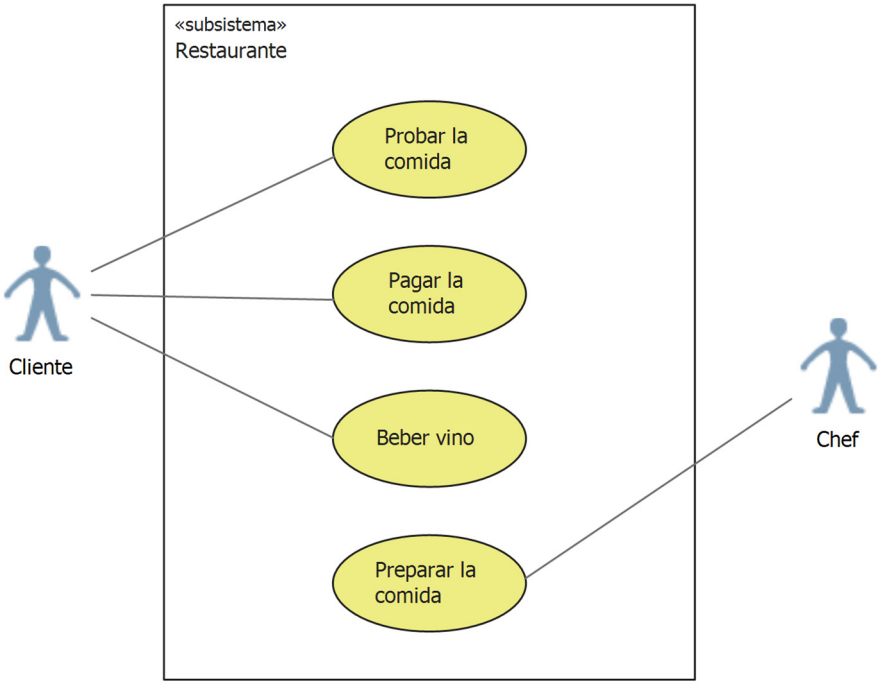


Imagen 9. Ejemplo de diagrama de casos de uso.

En él el sistema se representa en el rectángulo central. Los actores son elementos, usuarios u objetos, externos al sistema pero que interactúan con él de diferentes maneras. De esta forma se definen los límites en las relaciones de cada actor con el sistema.

Si se desea entrar más en detalle se puede disgregar cada caso de uso en pequeños subcasos que lo describan con mayor precisión. Por otro lado, este diagrama suele ir acompañado de una descripción más detallada de cada uno de los casos de uso, lo que permite comprender con mayor claridad la interacción entre actores y casos de uso.

5.2.2 Diagrama de clases

Realiza una descripción estática del sistema mostrando las clases del mismo con sus relaciones estructurales y de herencia. Cada clase posee sus atributos y sus métodos, correspondiéndose los primeros con las propiedades de la clase y los segundos con las operaciones que puede realizar. Como puede verse, la similitud con la POO es evidente desde el principio. Además, en estos diagramas se muestran las relaciones, ya sean de herencia o de uso, entre todas las clases mediante el uso de distintos tipos de flechas.

La *Imagen 10* muestra un ejemplo de diagrama de clases.

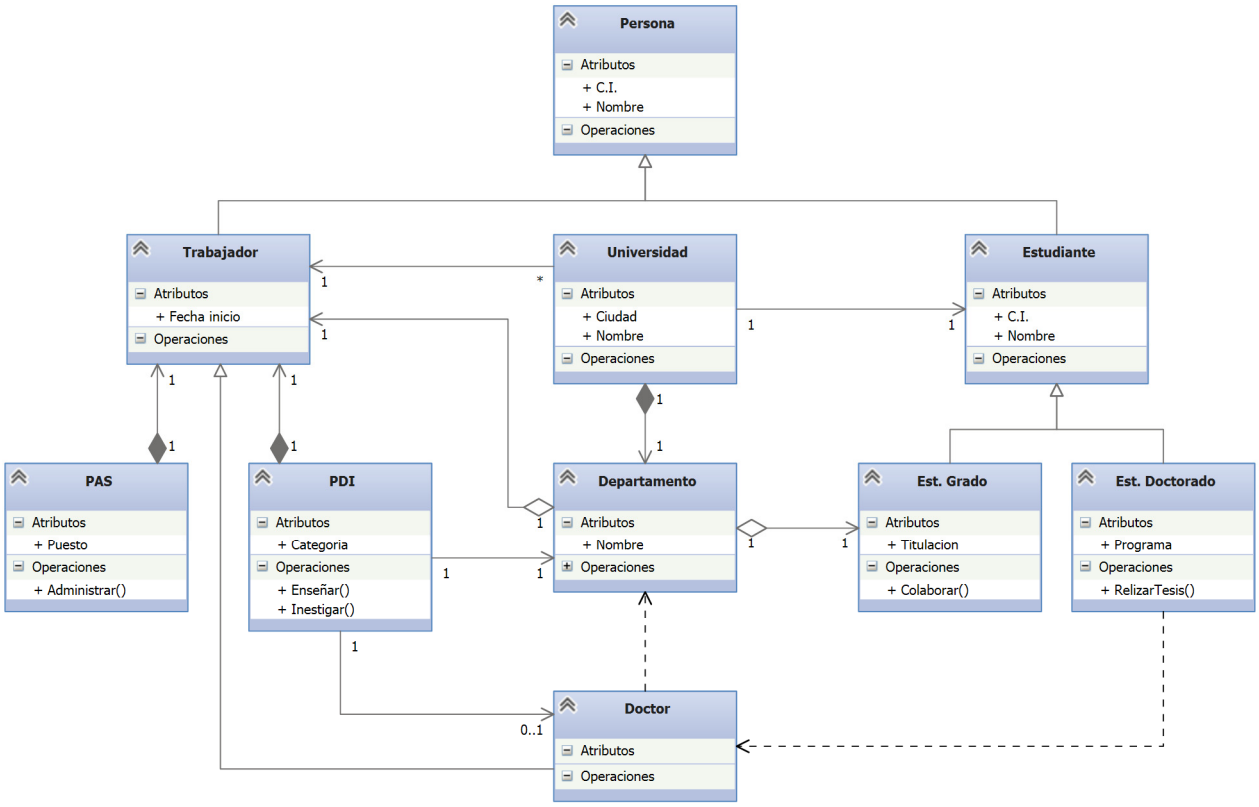
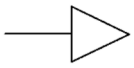
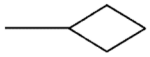
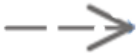


Imagen 10. Ejemplo de diagrama de clases.

En él pueden verse diferentes tipos de flechas que se corresponden con diferentes relaciones entre clases.

	Indica relación de **herencia** entre clases. Las clases inferiores heredan sus atributos y métodos de la clase inmediatamente superior jerárquicamente.
	Indica relación de **asociación** entre clases. El mero hecho de que dos clases interactúen entre sí marca una relación de asociación.
	Indica relación de **agregación** entre clases. La agregación es un tipo de asociación. Una clase agregada puede tener varias clases componentes, pero puede existir, aunque no existan todas o parte de ellas.
	Indica relación de **composición** entre dos clases. Es un caso particular de agregación en el que la clase agregada no puede existir sin sus clases componentes.
	Indica relación de **dependencia** entre clases. Una de las clases usa a la otra, bien sea mediante una instancia o recibiéndola como parámetro de un método.

Con este tipo de diagrama se pretende definir de forma precisa con el cliente los detalles del sistema. Es un diagrama que permanece en constante cambio a lo largo del diseño e incluso

durante el desarrollo del código, ya que debido a las particularidades de cada lenguaje pueden introducirse pequeños cambios. Además, en las sucesivas reuniones con el cliente suelen redefinirse aspectos importantes del sistema, por lo que suelen introducirse, eliminarse y modificarse clases.

5.2.3 Diagrama de objetos

Muestra las diferentes instancias de cada clase, llamadas *objetos*, y las relaciones que existen entre cada uno de ellos. Cada objeto tiene sus correspondientes atributos y operaciones que comparte con el resto de instancias de la misma clase. Un diagrama de clases puede tener infinitos diagramas de objetos.

Los *atributos* representan alguna propiedad del objeto, y por tanto son inherentes a él. Así pues, diferentes objetos de una misma clase tienen los mismos atributos, pero éstos pueden tener diferentes valores.

Las *operaciones* representan funciones que son realizadas por o sobre el objeto. Es decir, es lo que el objeto puede hacer. Cada objeto posee los atributos propios de la clase a la que pertenece, pero pueden o no tener diferentes valores. Además, cada objeto puede realizar todas las operaciones asignadas a la clase a la que pertenece.

5.2.4 Diagrama de estados

Recoge el estado en el que se encuentra el sistema en un momento determinado, así como la secuencia de estados que se está siguiendo.

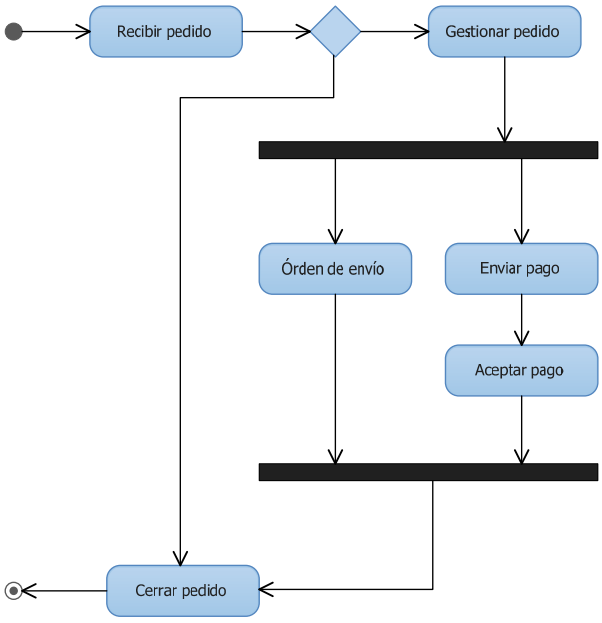


Imagen 11. Ejemplo de diagrama de actividad.

Pueden existir tantos diagramas como actividades haya en el sistema. En cada uno se representa la sucesión de acciones que se dan desde el inicio hasta el fin de la acción. Por

tratarse de un diagrama de alto nivel, en cada acción únicamente se indica con una frase descriptiva qué hace el sistema.

En caso de existir secuencias de acciones diferentes en función de una condición, se plantea un *nodo de decisión*, en el que la decisión de qué secuencia seguir debe ser unívoca. Por otro lado, secuencias de datos alternativas pueden confluir en los llamados *nodos de reunión*, aunque también puede darse el caso de que cada secuencia alternativa tenga su propio final.

Por otro lado, en este tipo de diagrama también se contempla la ejecución en paralelo de varias acciones y la invocación de otra serie de actividades desde una misma.

5.2.5 Diagrama de interacción

También llamado diagrama de secuencias, muestran las relaciones existentes entre objetos, de forma secuencial, durante el desarrollo de una actividad. Contiene detalles de la implementación ante diferentes escenarios, así como los mensajes y eventos intercambiados entre los objetos que componen el sistema.

Por *mensaje* se entiende cualquier comunicación entre objetos que implique intercambio de información. Lo más habitual es que un mensaje se refiera a un evento en el que un objeto informe a otros de que algo ha ocurrido y cómo deben reaccionar ante el mismo.

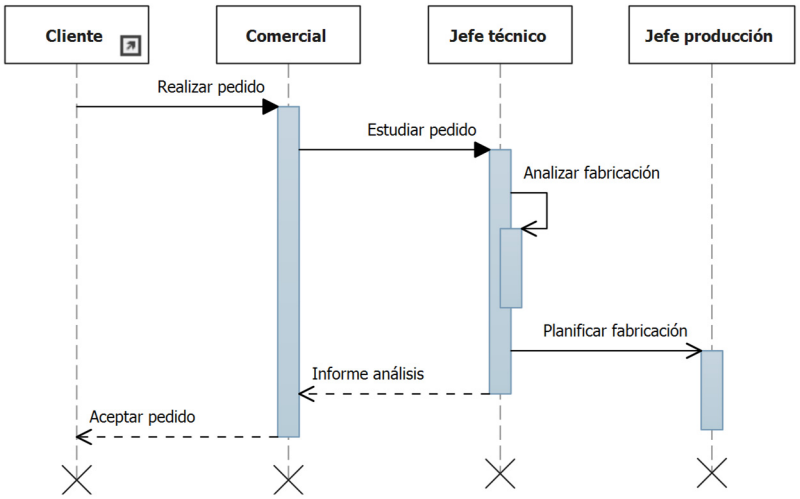


Imagen 12. Ejemplo de diagrama de interacción.

En este diagrama se representa todo intercambio de mensajes entre los objetos que participan en dicha actividad. Además, la representación es secuencial, por lo que, de esta manera, también se conoce cuál es el orden de paso de dichos mensajes.

5.3 Comunicación serie

La comunicación serie es un protocolo común de comunicación entre dispositivos. También es muy utilizada para la adquisición y transmisión de datos y, por dispositivos de instrumentación, para la transmisión de la información que proporcionan.

En la comunicación serie se envían y reciben bytes de información bit a bit, mientras que en la comunicación en paralelo se transmiten bytes completos de cada vez. Por tanto, la comunicación serie es más lenta que la paralelo, pero es un método más sencillo y puede alcanzar mayores distancias. En la comunicación en paralelo las distancias entre dispositivos no deben ser superiores a 20 metros, mientras que en la comunicación serie estas distancias aumentan hasta los 1200 metros.

Normalmente, la comunicación serie se emplea para transmitir datos en formato ASCII simple. Para ello se utilizan como mínimo 3 líneas de transmisión. Debido a que la transmisión es asincrónica, es posible enviar datos por una línea mientras se reciben datos por la otra. El resto de líneas disponibles se utilizan para realizar intercambio de pulsos de sincronización, pero no son siempre necesarias.

Las características más importantes de la comunicación serie son la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar es necesario que sus características sean iguales.

- **Velocidad de transmisión *(baud rate)*:** Indica el número de bits por segundo que se transfieren. Se mide en baudios.
- **Bits de datos:** Se refiere a la cantidad de bits de la transmisión. El número de bits que se envía depende del tipo de información que se transfiere.
- **Bits de parada:** Indica el fin de la transmisión de un paquete de datos. Los valores típicos son 1, 1.5 o 2 bits. Además, dan un margen de tolerancia para la sincronización en caso de que esté prevista.
- **Paridad:** Permite verificar si hay errores en la transmisión. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. También está la opción de no usar paridad.
- **Control de flujo:** Se utiliza para verificar la calidad en la transmisión de datos. Puede realizarse vía software o hardware.

La norma RS232 es una de las utilizadas en la comunicación serie vía PC. Aunque actualmente está ampliamente superada por la transmisión serie a través de USB, son numerosos los dispositivos que todavía la utilizan. La norma RS232 marca los aspectos fundamentales en la comunicación entre equipos de tipo *Terminal de Datos (DTE)*, por ejemplo, un PC, y equipos de tipo *Comunicador de Datos (DCE)*, por ejemplo, un ratón.

La norma RS232 también prevé la comunicación entre dos *DTE*, como es el caso del presente proyecto, en el que el envío y recepción de datos se realizará entre dos PCs. Este es un aspecto importante de cara a la definición del cable que se utilizará para la transmisión. Para la

conexión entre dos PCs, que enviarán y recibirán información de forma asíncrona, la norma recomienda el uso de conectores tipo DB9 en los que sus terminales seguirán el siguiente patrón de conexión.

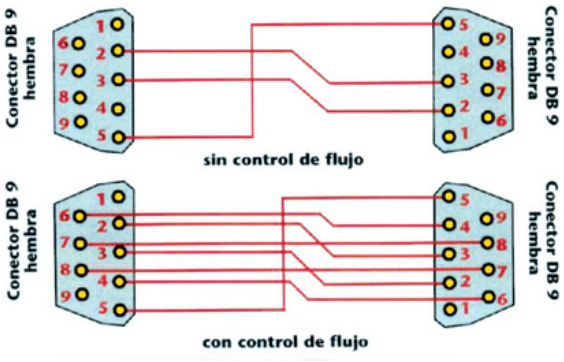


Imagen 13. Esquemas de terminales para comunicación serie.

El significado de cada pin de conexión se muestra en la siguiente tabla.

Terminal	Nombre	Símbolo RS232	Descripción
1	CD	CF	Detección de portadora
2	RXD	BB	Recepción de datos
3	TXD	BA	Transmisión de datos
4	DTR	CD	Terminal de datos preparado
5	GND	AB	Señal de tierra
6	DSR	CC	Dispositivo preparado
7	RTS	CA	Petición de envío
8	CTS	CB	Preparado para transmitir
9	RI	CE	Indicación de llamada entrante

Tabla 9. Pines del conector DB9 en comunicación serie.

De esta manera, por ejemplo, el terminal de recepción de datos de un conector recibirá la información enviada desde el terminal de envío del otro conector y, de forma análoga, para el resto de terminales. Tal y como se ha definido en las especificaciones técnicas, no se realizará control del flujo de datos, por lo que bastará con realizar el conexionado de los terminales de envío, recepción y tierra de la forma que indica el gráfico. El resultado se muestra en la siguiente imagen.

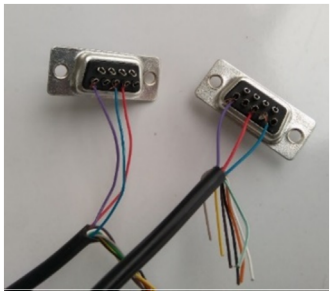


Imagen 14. Conexionado de los terminales.

6 DESCRIPCIÓN GENERAL DEL SISTEMA

Como ya se ha ido definiendo a lo largo del documento, el sistema estará formado por dos partes claramente diferenciables: la parte de adquisición y procesamiento de datos *(DSP)* y la parte de supervisión y control *(HMI)*.

Todos los elementos del *DSP* se encuentran dispuestos en una envolvente a la que llegarán todas las señales procedentes de la sensórica instalada en la turbina.

La electrónica instalada en el armario realizará las conversiones analógico-digitales necesarias para que las señales, procedentes de la turbina, puedan ser interpretadas por el ordenador tipo *fanless* dispuesto a tal efecto. Este equipo recibirá los datos vía comunicación serie y se encargará de la parte de tratamiento y procesamiento de la información de la turbina. Se trata de un ordenador de tipo industrial sin refrigeración forzada diseñado para soportar condiciones extremas de temperatura y vibraciones.

Imagen 15. Armario de control.

La parte destinada a la supervisión y control, *HMI,* constará únicamente de un PC de sobremesa con los dispositivos periféricos necesarios (teclado ratón, pantalla…). Para establecer la comunicación entre el *DSP* y el *HMI* se creará una red local cableada, de tipo ethernet industrial, a la que se podrán conectar ambos equipos.

6.1 Funcionalidades del equipo DSP

Este equipo se encargará de toda la parte de tratamiento y procesamiento de la información procedente de la turbina en forma de trama de datos. A grandes rasgos, el equipo recibirá la trama de datos a través del puerto serie, extraerá la información de la misma y la almacenará en la base de datos instalada localmente. Además, si la trama informa de algún estado de error en la turbina, se informará inmediatamente a la parte de supervisión *(HMI)*. Por otro lado, si desde el *HMI* se envía algún comando de control, el *DSP* lo procesará y lo enviará a través del puerto serie.

Con todo ello, las tareas mínimas que debe realizar el equipo *DSP* son las siguientes:

- Procesamiento y tratamiento de los paquetes de datos recibidos a través del puerto serie.
- Extracción de la información de la trama de datos recibida.
- Almacenamiento de la información en la base de datos local.
- Chequeo de las palabras de error y estado de la trama recibida.
- Envío de las palabras de error y estado al *HMI*, cuando proceda.
- Envío de los comandos de control a través del puerto serie, cuando proceda.
- Registro de eventos en el sistema.

6.2 Funcionalidades del equipo HMI

El equipo *HMI* permitirá al personal de la instalación monitorizar y controlar ciertos parámetros de la turbina. A grandes rasgos, la aplicación permitirá la visualización de las variables medidas en la turbina, en forma de curvas de tendencia. Además, se seguirá un registro de las alarmas detectadas y se permitirá controlar ciertos parámetros de la turbina a los usuarios cualificados.

Con todo ello, las tareas mínimas que debe realizar el equipo *HMI* son las siguientes:

- Permitir al usuario la visualización, en forma de curvas de tendencia, de los datos recogidos por el *DSP,* en el rango de tiempos que desee.
- Servir de respaldo para la base de datos del *DSP.*
- Registro histórico de alarmas y cambios en el estado de funcionamiento de la turbina.
- Envío de comandos al *DSP* para el control de la turbina.

7 MODELADO Y DISEÑO DEL SOFTWARE

Para el diseño de los programas de control participantes en el sistema se ha decidido utilizar la metodología UML. En este apartado se muestran los diferentes diagramas, junto con sus explicaciones, necesarios para la definición del sistema.

El código, en lenguaje C#, de cada programa de control se adjunta en los anexos para su consulta. No obstante, es importante recalcar que el uso de dicha metodología permite la traducción de los diagramas creados a casi cualquier lenguaje basado en programación orientada a objetos, teniendo en cuenta las particularidades sintácticas y estructurales de cada uno.

En este apartado se incluyen los diagramas referentes al modelado de las dos aplicaciones presentes en el sistema: el *DSP* y el *HMI*. Conviene recordar que se encontrarán corriendo en dos equipos diferentes, pero que, a pesar de ser independientes, se encuentran conectadas entre sí a través de una red local, mediante la cual se intercambian mensajes y datos. Conviene señalar también que existen una serie de elementos, externos al sistema, que participan activamente en el mismo, como son las bases de datos, el registro de configuración de Windows, etc.

Se comienza mostrando los diagramas correspondientes con el diseño estático (diagramas de casos de uso y de clases) para, a continuación, mostrar los de comportamiento (diagramas de secuencias y de estado). No se incluyen los diagramas de objetos, pues las clases definidas únicamente tendrán una instancia de sí mismas, por lo que los diagramas de clases y estados serán idénticos en ambas aplicaciones.

7.1 Aplicación DSP

7.1.1 Diagrama de casos de uso

A partir de las especificaciones del DSP se pueden extraer cada una de las posibles interacciones entre los actores principales y la aplicación. Como actores se han definido todos aquellos elementos externos al sistema que interactúan con el mismo:

- **Usuario:** realiza acciones de forma directa sobre la aplicación en forma de eventos según sus necesidades.
- **Turbina:** envía o recibe datos en forma de trama binaria.
- **Aplicación HMI:** permite el control y monitorización del sistema de forma remota.
- **Base de datos local:** almacena los datos deseados y permite su consulta. Se encuentra instalada en el mismo equipo que la aplicación.
- **Pantalla:** sirve como interface visual entre el usuario y la aplicación.

Así pues, se han definido los siguientes casos de uso, algunos de los cuales incluyen subcasos de uso que pueden o no ejecutarse en función del cumplimiento de ciertas condiciones:

- Iniciar aplicación.
  - Abrir puerto serie.
  - Abrir conexión con la base de datos.
- Recibir comandos.
  - Procesar comandos.
  - Envío de comandos a la turbina.
- Cerrar aplicación.
  - Cerrar puerto serie.
  - Cerrar conexión con la base de datos.
  - Terminar servidor.
- Recibir trama de datos.
  - Procesar trama de datos recibida.
  - Insertar trama recibida en base de datos.
- Envío de errores al HMI.
- Mostrar mensajes por pantalla.

El diagrama de casos de uso se muestra en la *Imagen 16*.

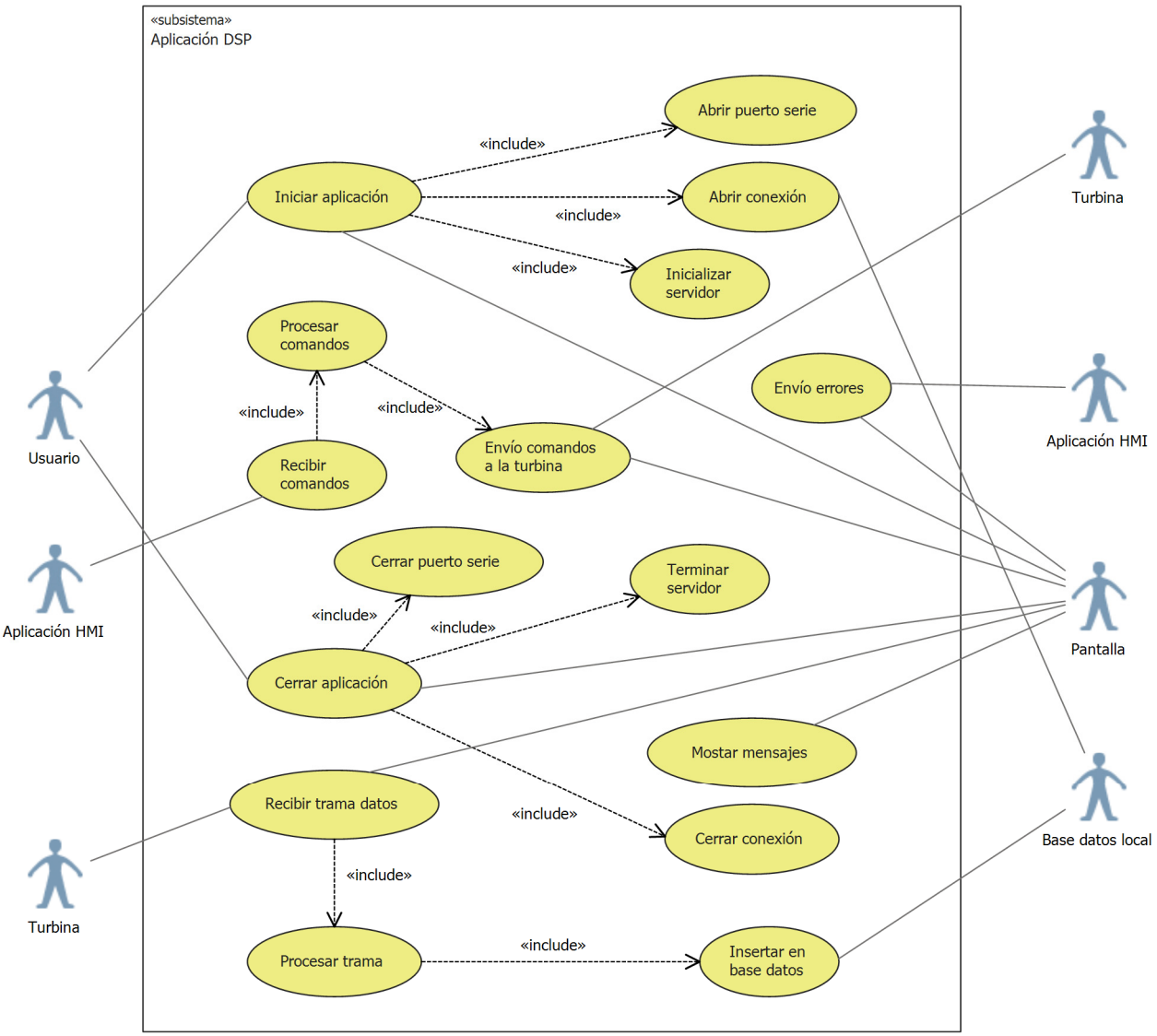


Imagen 16. Diagrama de casos de uso aplicación DSP.

A continuación, se muestran cada uno de los casos de uso recogidos en el diagrama, detallando a los participantes en cada uno de ellos, los objetivos que se pretenden conseguir en cada caso de uso y los requisitos previos necesarios para conseguirlos.

7.1.1.1 Iniciar aplicación

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

Precondiciones

- El usuario ha adquirido la aplicación y dispone del hardware necesario para iniciarla.
- El hardware es compatible con la aplicación.
- El usuario decide arrancar la aplicación en su dispositivo.

Garantía de éxito

El usuario consigue arrancar la aplicación y mostrarla por pantalla.

Escenario principal de éxito

i. El usuario arranca el hardware.
ii. El usuario inicia la aplicación.
iii. Aparece el interfaz gráfico de la aplicación en la pantalla.
iv. El puerto serie se abre correctamente.
v. La conexión con el servidor de estados se inicializa correctamente.

7.1.1.2 Abrir puerto serie

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de al menos un puerto serie habilitado.
- Los parámetros, referentes al puerto serie, definidos en la aplicación son correctos y coherentes.

Garantía de éxito

El puerto serie se ha inicializado y abierto con éxito.

Escenario principal de éxito

i. El puerto serie se inicializa con los parámetros configurados.
ii. Se intenta abrir el puerto serie.
iii. El puerto serie se abre correctamente.

7.1.1.3 Inicializar servidor de comandos

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de al menos una tarjeta de red habilitada. Los parámetros, referentes a la tarjeta de red, definidos en la aplicación son correctos y coherentes.
- El equipo se encuentra conectado a la red local del sistema.

Garantía de éxito

El servidor de comandos se ha inicializado con éxito. Es posible la recepción de datos desde la aplicación remota a través de la red local del sistema.

Escenario principal de éxito

i. El servidor de comandos se inicializa correctamente con los parámetros configurados.
ii. Comienza la escucha de conexiones entrantes.

7.1.1.4 Abrir conexión con la base de datos

Actores participantes y objetivos

- **Base de datos local**: permite el almacenamiento y consulta de datos. Se encuentra instalada en el mismo equipo que la aplicación.

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de un motor de bases de datos instalado.
- Los parámetros, referentes a la base de datos, definidos en la aplicación son correctos y coherentes.
- El servicio correspondiente al motor de bases de datos se encuentra iniciado.

Garantía de éxito

La conexión con la base de datos se ha realizado correctamente. Se pueden introducir datos y realizar consultas a la base de datos.

Escenario principal de éxito

i. Se establece la conexión con la base de datos.
ii. La conexión es correcta.

7.1.1.5 Recibir trama de datos

Actores participantes y objetivos

- **Turbina**: envía diferentes señales al armario de control. Estas señales se tratan para enviarlas, vía comunicación serie, al ordenador de procesamiento en forma de trama de datos.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de al menos un puerto serie habilitado.
- El puerto serie se ha inicializado y abierto con éxito.
- El equipo dispone de un motor de bases de datos instalado.

- Los parámetros, referentes a la base de datos, definidos en la aplicación son correctos y coherentes.
- El servicio correspondiente al motor de bases de datos se encuentra iniciado.

Garantía de éxito

Se reciben datos vía puerto serie. Los datos son correctos y completos. Los datos se introducen correctamente en la base de datos. Cada cierto número de adquisiciones se informa por pantalla de que se están recibiendo datos.

Escenario principal de éxito

i. Se reciben tramas de datos por el puerto serie.
ii. La trama se procesa y es correcta.
iii. Los datos son introducidos en la base de datos local de forma satisfactoria.

7.1.1.6 Procesar trama de datos recibida

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de al menos un puerto serie habilitado.
- El puerto serie se ha inicializado y abierto con éxito.
- Se ha recibido una trama de datos en el puerto serie.

Garantía de éxito

La trama recibida es correcta.

Escenario principal de éxito

i. Se recibe una trama de datos en el puerto serie.
ii. Se procesa la trama por si ésta se hubiera enviado en varios paquetes.
iii. Se comprueban las palabras de control de la trama para determinar si ésta es correcta.
iv. Se descodifican los datos de la turbina.

7.1.1.7 Insertar trama recibida en base de datos

Actores participantes y objetivos

- **Base de datos local**: permite el almacenamiento y consulta de datos. Se encuentra instalada en el mismo equipo que la aplicación.

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de un motor de bases de datos instalado.

- Los parámetros, referentes a la base de datos, definidos en la aplicación son correctos y coherentes.
- El servicio correspondiente al motor de bases de datos se encuentra iniciado.
- Se ha descodificado una trama de datos recibida en el puerto serie.

Garantía de éxito

Los datos de la trama recibida se introducen de forma correcta en la base de dato local.

Escenario principal de éxito

i. Se tienen los datos de la turbina referentes a una trama.
ii. La conexión con la base de datos se ha establecido satisfactoriamente.
iii. Los datos se introducen de forma satisfactoria en la base de datos local.

7.1.1.8 Envío de errores al HMI

Actores participantes y objetivos

- **Aplicación HMI**: permite la monitorización de la turbina (alarmas estado general), así como el envío de comandos para el control de la misma.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

Precondiciones

- La aplicación DSP se ha iniciado correctamente.
- La aplicación HMI se ha iniciado correctamente.
- El equipo dispone de al menos una tarjeta de red habilitada. Los parámetros, referentes a la tarjeta de red, definidos en la aplicación son correctos y coherentes.
- El equipo se encuentra conectado a la red local del sistema.
- La tarjeta se ha inicializado de forma correcta.
- Se ha descodificado una trama de datos recibida en el puerto serie.
- El valor de las palabras de error y estado de la turbina se han modificado de la adquisición anterior a la actual.

Garantía de éxito

Las nuevas palabras de error y estado de la turbina se han enviado al HMI.

Escenario principal de éxito

i. Se comparan las palabras de error y estado de la turbina actuales con las de la adquisición anterior. Son diferentes.
ii. Se realiza una petición de conexión al equipo remoto (servidor de estados).
iii. La conexión es correcta.
iv. El envío de las nuevas palabras de error y estado de la turbina se ha realizado de forma satisfactoria.

7.1.1.9 Recibir comandos

Actores participantes y objetivos

- **Aplicación HMI**: permite la monitorización de la turbina (alarmas estado general), así como el envío de comandos para el control de la misma.

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de al menos una tarjeta de red habilitada. Los parámetros, referentes a la tarjeta de red, definidos en la aplicación son correctos y coherentes.
- El equipo se encuentra conectado a la red local del sistema.
- El servidor de comandos se ha inicializado de forma correcta.

Garantía de éxito

Se ha recibido un comando de la aplicación remota de forma correcta.

Escenario principal de éxito

i. El servidor de comandos se encuentra escuchando conexiones entrantes.
ii. Un equipo remoto realiza una solicitud de conexión.
iii. El equipo está dentro del rango. La conexión se acepta.
iv. Se recibe el comando enviado desde la aplicación remota.

7.1.1.10 Procesar comandos

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de al menos una tarjeta de red habilitada. Los parámetros, referentes a la tarjeta de red, definidos en la aplicación son correctos y coherentes.
- El equipo se encuentra conectado a la red local del sistema.
- La tarjeta se ha inicializado de forma correcta.
- Se ha recibido un comando de un equipo remoto.

Garantía de éxito

El comando recibido es correcto y se ha interpretado su significado.

Escenario principal de éxito

i. Se recibe un comando de un equipo remoto en forma de trama.
ii. Se comprueban las palabras de control de la trama para determinar si ésta es correcta.

iii. Se descodifica el comando recibido.

7.1.1.11 Envío de comandos a la turbina

Actores participantes y objetivos

- **Turbina**: envía diferentes señales al armario de control. Estas señales se tratan para enviarlas, vía comunicación serie, al ordenador de procesamiento en forma de trama de datos.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de al menos un puerto serie habilitado.
- El puerto serie se ha inicializado y abierto con éxito.
- Se ha descodificado un comando recibido de un equipo remoto.

Garantía de éxito

El comando se envía correctamente a la turbina.

Escenario principal de éxito

i. Se tiene un comando descodificado.
ii. Se codifica el comando en forma de trama para su envío a través del puerto serie.
iii. El comando se envía a través del puerto serie con éxito.
iv. Se muestra el comando enviado por pantalla.

7.1.1.12 Mostrar mensajes por pantalla

Actores participantes y objetivos

- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

Precondiciones

- El usuario ha iniciado la aplicación.
- Existe algún mensaje que mostrar.

Garantía de éxito

El mensaje se muestra correctamente en el interface.

Escenario principal de éxito

i. Se produce un evento en la aplicación.
ii. El mensaje se muestra de forma correcta en el interface.

7.1.1.13 Cerrar aplicación

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

Precondiciones

- La aplicación se encuentra iniciada.
- El servidor de comandos se encuentra escuchando conexiones entrantes.
- El puerto serie está abierto.
- La conexión con la base de datos local es correcta.

Garantía de éxito

La aplicación se cierra y se deja de mostrar por pantalla.

Escenario principal de éxito

i. El usuario decide cerrar la aplicación de forma voluntaria.
ii. El servidor de comandos deja de escuchar conexiones entrantes.
iii. El puerto serie se cierra.
iv. La conexión con la base de datos se termina.
v. Se deja de mostrar la aplicación por pantalla.

7.1.1.14 Dejar de escuchar conexiones entrantes

Precondiciones

- El usuario ha decidido cerrar la aplicación

Garantía de éxito

El servidor de comandos deja de escuchar conexiones entrantes.

Escenario principal de éxito

i. El usuario decide cerrar la aplicación de forma voluntaria.
ii. Se dejan de escuchar conexiones entrantes a través de la tarjeta de red.

7.1.1.15 Cerrar puerto serie

Precondiciones

- El usuario ha decidido cerrar la aplicación

Garantía de éxito

El puerto serie se cierra correctamente.

Escenario principal de éxito

i. El usuario decide cerrar la aplicación de forma voluntaria.
ii. El puerto serie se cierra con éxito.

7.1.1.16 Cerrar conexión con la base de datos

Actores participantes y objetivos

- **Base de datos local**: permite el almacenamiento y consulta de datos. Se encuentra instalada en el mismo equipo que la aplicación.

Precondiciones

- El usuario ha decidido cerrar la aplicación

Garantía de éxito

La conexión con la base de datos local se termina correctamente.

Escenario principal de éxito

i. El usuario decide cerrar la aplicación de forma voluntaria.
ii. La conexión con la base de datos local se termina correctamente.

7.1.2 Diagrama de clases

A través del diagrama de clases se describe la estructura estática del sistema en términos de clases y de relaciones entre las mismas. Se ha pretendido encapsular al máximo el programa, dentro de unos límites razonables, creando una clase por cada interacción usuario-sistema. Además, se han creado algunas clases necesarias para poder llevar a cabo algunos métodos de algunas de las clases. De esta manera, como ya se ha dicho, cada clase tendrá una única instancia de sí misma, o lo que es lo mismo, solo existirá un objeto asociado a cada una de las clases. Por ello el diagrama de objetos será análogo al de clases.

Con el fin de facilitar la comprensión del diagrama, las clases se muestran sin sus atributos ni métodos. Únicamente se muestran las relaciones existentes entre todas ellas. No obstante, en los anexos se pueden observar cada una de las clases con sus respectivos atributos y métodos.

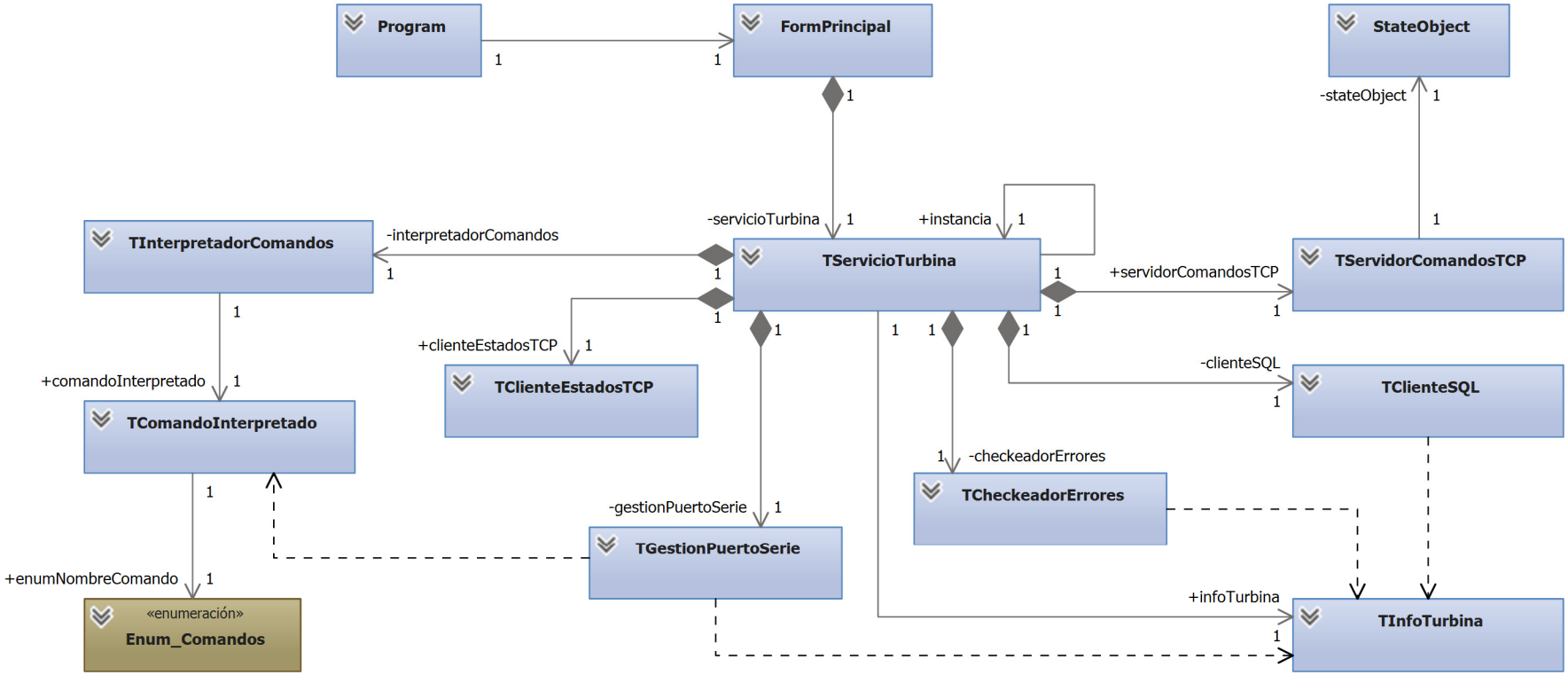


Imagen 17. Diagrama de clases aplicación DSP.

La estructura del programa es la siguiente:

- La clase ***Program*** es la clase principal de la aplicación. En ella se inicializa el formulario principal junto con su correspondiente hilo.
- La clase ***FormPrincipal*** inicializa y gestiona todos los eventos referentes al interface con el usuario. Además, permite mostrar los mensajes de eventos por pantalla.
- La clase ***TServicioTurbina*** gestiona todo el modelo de negocio de la aplicación, o lo que es lo mismo, gestiona todos los eventos y pasos de parámetros entre las diferentes clases del programa. Como se observa en el diagrama de clases, se podría decir que es la clase central del programa. Todas las clases que interactúan con algún actor del sistema se crean y se destruyen con esta clase.
- La clase ***TClienteEstadosTCP*** realiza todas las tareas referentes al envío de las alarmas y estado de la turbina cuando se detecta un cambio en los mismos de una adquisición a otra. El envío se realiza a través de la red local mediante protocolo TCP/IP por lo que, para que el envío sea correcto, debe haber un servidor de estados escuchando en la aplicación remota. Al finalizar el envío, bien haya sido de forma satisfactoria o no, se informa al servicio.
- La clase ***StateObject*** ayuda a la escucha asincrónica del servidor de comandos.
- La clase ***TServidorComandos*** realiza todas las tareas referentes a la recepción de comandos procedentes de la aplicación HMI. Una vez inicializado, el servidor se encuentra escuchando conexiones entrantes. Cuando recibe una y la acepta informa al servicio de que la recepción se ha completado con éxito y pasa a interpretar el comando haciendo uso de las tres siguientes clases.
- La clase ***TInterpretadorComandos*** descodifica e interpreta el comando, recibido en el servidor de comandos en forma de trama de bytes, haciendo uso de las dos siguientes clases.
- La clase ***TComandoInterpretado*** interpreta el comando recibido a uno de los posibles que se esperan haciendo uso de la siguiente clase. Una vez conocido el comando se informa al servicio.
- La clase ***Enum_Comandos*** contiene una enumeración con la traducción (byte-texto) de todos los posibles comandos que se esperan.
- La clase ***TGestionPuertoSerie*** realiza todas las tareas relacionadas con el puerto serie. Inicializa y cierra el puerto según convenga y realiza todas las tareas relacionadas con el tratamiento de las tramas recibidas; desde la recepción de todos los paquetes de datos de la trama, hasta la extracción de la información presente en la misma. Al finalizar el procesamiento de una trama recibida informa al servicio.
- La clase ***TCheckeadorErrores*** comprueba si las palabras correspondientes a las alarmas y el estado de la turbina se han modificado de una adquisición a otra. En tal caso informa al servicio.
- La clase ***TInfoTurbina*** almacena la información descodificada de la trama recibida en la adquisición actual.

- La clase ***TClienteSQL*** realiza todas las tareas relacionadas con la interacción con la base de datos local; desde la inicialización y cierre de la conexión con la base de datos, según convenga, hasta la inserción de la información de la turbina de la adquisición actual.

7.1.3 Diagramas de secuencias

En este apartado se muestran los diagramas de secuencias correspondientes con los casos de uso anteriormente descritos. Cabe destacar que no se muestra un diagrama por cada caso de uso, sino que dentro de uno mismo pueden incluirse varios por tener algún tipo de relación. Otro aspecto reseñable es que, para el diseño de sistemas mediante esta metodología, solo suelen realizarse los diagramas de secuencias para los escenarios principales de éxito, ya que el incluir todos los diagramas posibles, en caso de no encontrarse en escenario de éxito, implicaría una cantidad ingente de diagramas que aportarían poco al propio diseño. De esta manera, los diagramas de secuencias incluyen los siguientes casos de uso:

1) Iniciar aplicación:
   a) *Iniciar aplicación.*
   b) *Abrir puerto serie.*
   c) *Iniciar servidor de comandos.*
   d) *Abrir conexión con la base de datos local.*
2) Recibir trama de datos:
   a) *Recibir trama de datos.*
   b) *Procesar trama.*
   c) *Insertar datos en base de datos.*
3) Envío de alarmas y estado al HMI.
   a) *Envío de alarmas y estado al HMI.*
4) Recibir comandos del HMI.
   a) *Recibir comandos del HMI.*
   b) *Procesar comandos.*
   c) *Envío de comandos a la turbina.*
5) Cerrar aplicación.
   a) *Cerrar aplicación.*
   b) *Cerrar conexión con la base de datos local.*
   c) *Cerrar puerto serie.*
   d) *Dejar de escuchar.*

Por otro lado, la aparición del caso de uso *Mostrar mensajes,* es común a muchos otros casos de uso.

7.1.3.1 Iniciar aplicación

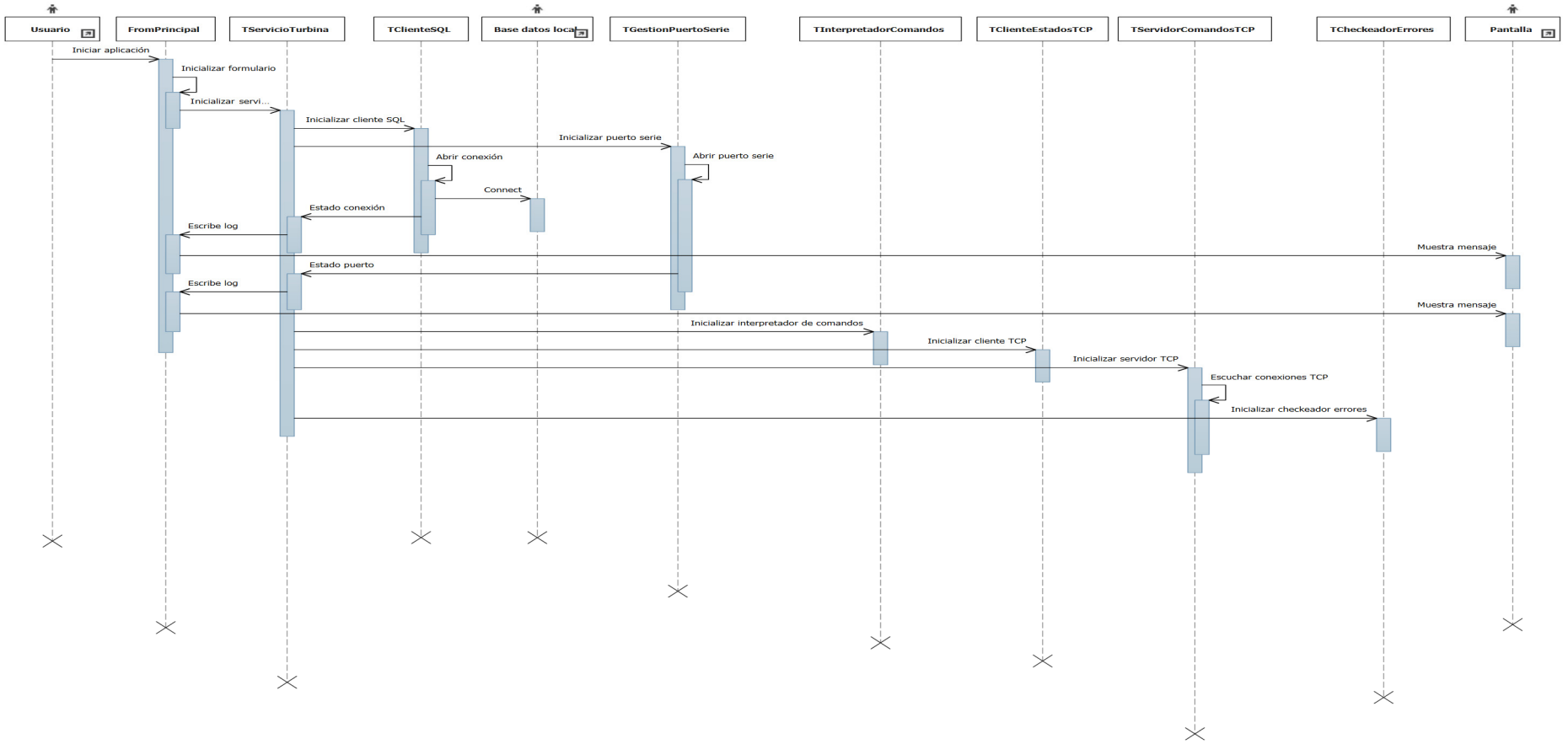


Imagen 18. Diagrama de secuencias DSP. Iniciar aplicación.

Cuando el ***usuario*** inicia la aplicación, el ***formulario principal*** inicializa todos sus componentes y el ***servicio***. Por su parte, el servicio inicializa el ***puerto serie,*** la conexión con la ***base de datos local*** y el ***servidor de comandos***, así como sus respectivas clases auxiliares, lo que permite la recepción de tramas de datos y de comandos remotos respectivamente. Al completarse cada inicialización se muestra un mensaje a través del formulario.

7.1.3.1 Recibir trama de datos

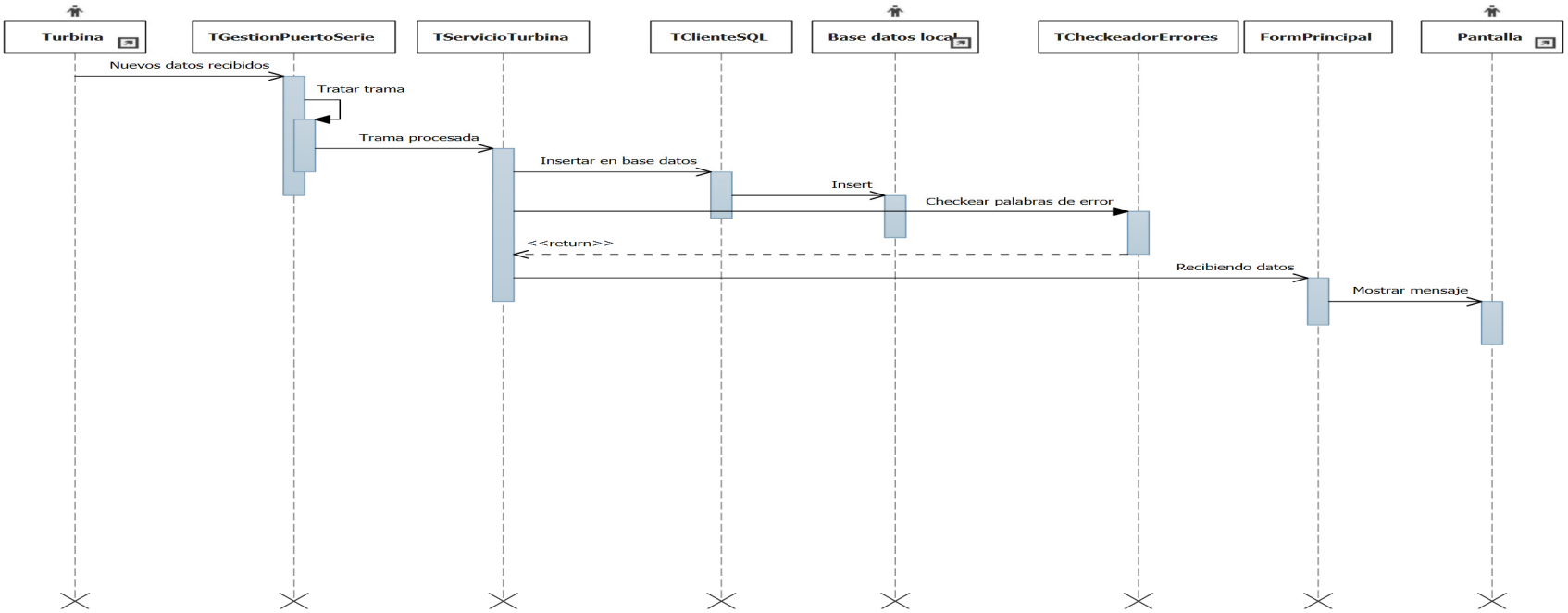


Imagen 19. Diagrama de secuencias DSP. Recibir trama de datos.

Cuando se recibe una trama desde la ***turbina*** en el ***puerto serie,*** la clase encargada de su gestión la trata e interpreta para extraer la información relativa a la turbina. Si la trama recibida es correcta se introduce la información en la ***base de datos local***, tras lo cual se informa al servicio. Finalmente, se ***chequean las palabras de error y estado*** por si se hubiera producido algún cambio de una adquisición a otra; en tal caso se informa al servicio. Cada cierto número de capturas se muestra un mensaje informativo en el ***formulario principal***.

7.1.3.2 Envío de alarmas y estado al HMI

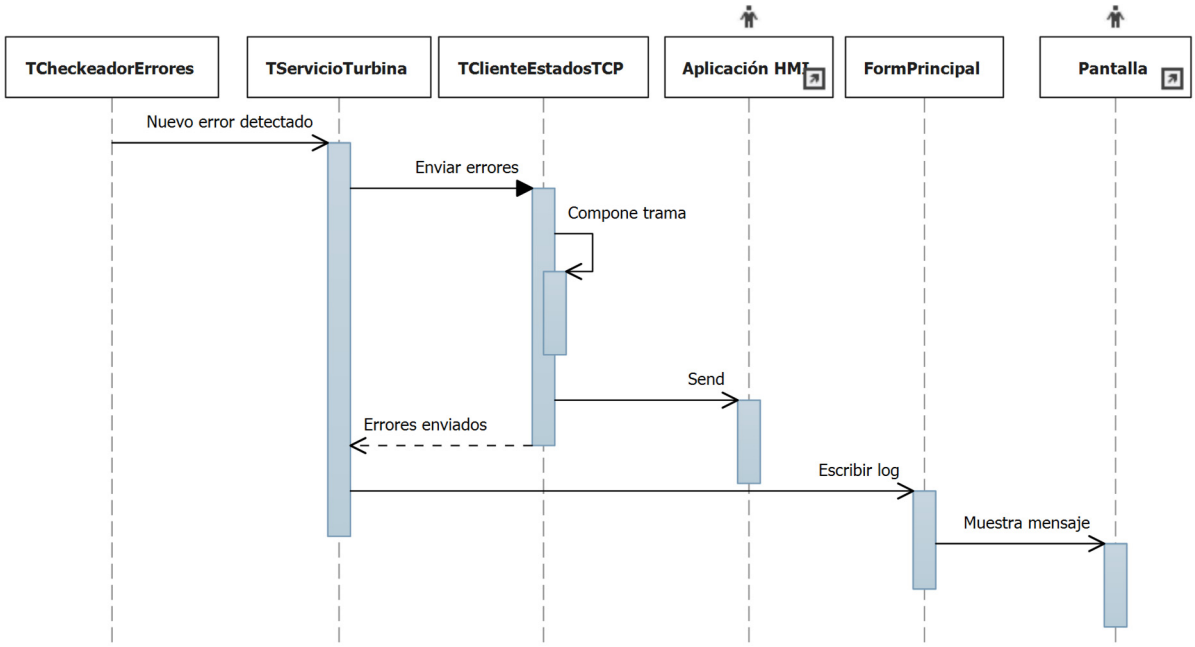


Imagen 20. Diagrama de secuencias DSP. Envío de alarmas y estado al HMI.

Si se detecta una variación en las palabras de estado o error se informa al ***servicio,*** que procede al envío de las mismas a la ***aplicación remota*** a través de un **cliente TCP**, el cual compone una trama para el envío. Tras el envío se muestra un mensaje en el ***formulario principal***.

7.1.3.3 Recibir comandos del HMI

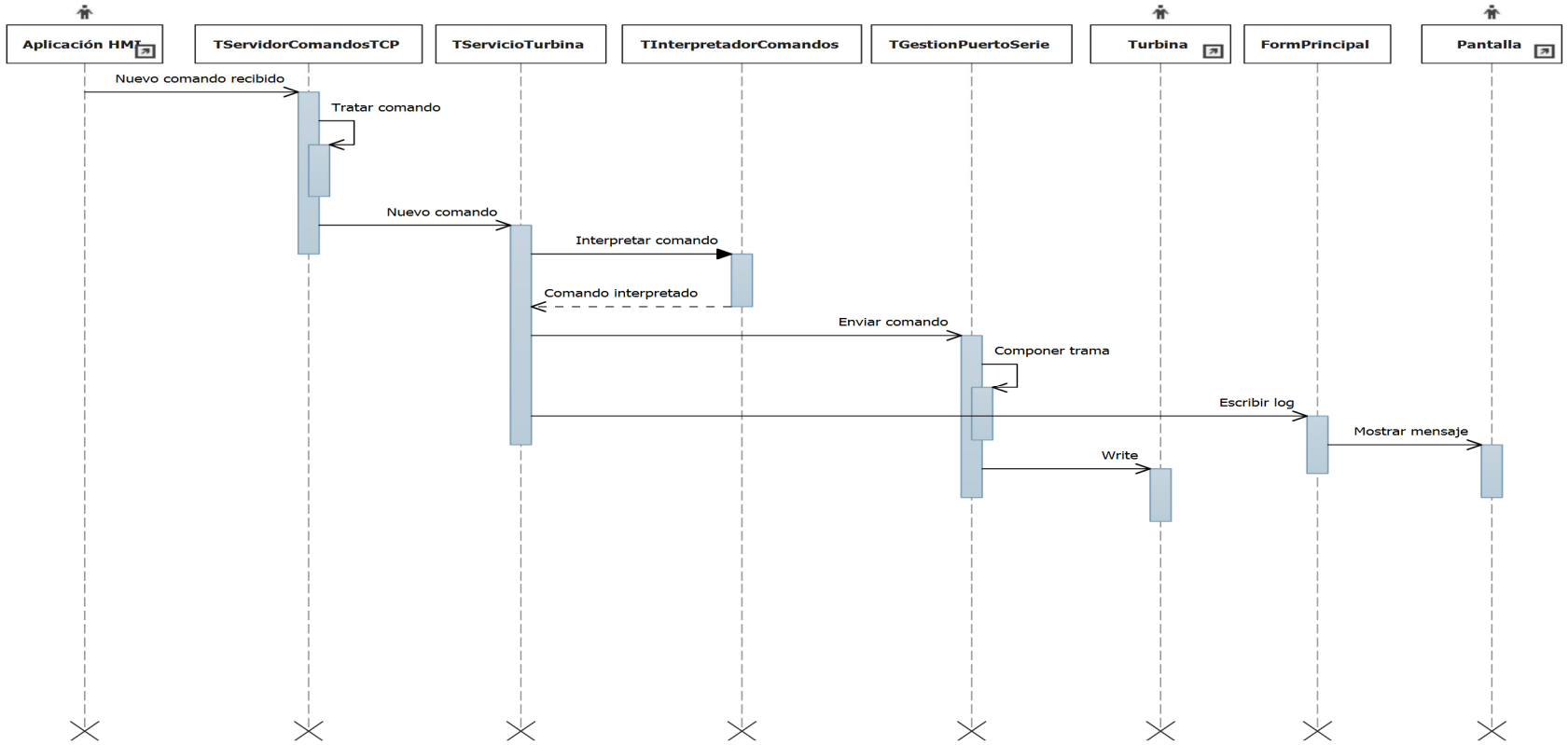


Imagen 21. Diagrama de secuencias DSP. Recibir comandos del HMI.

Cuando el ***servidor de comandos*** recibe un comando de la ***aplicación remota***, informa al ***servicio,*** el cual da la orden al ***interpretador de comandos*** para que lo devuelva interpretado. Finalmente, el servicio da la orden para que la trama se envíe a la ***turbina*** a través del ***puerto serie***. Si el comando se ha enviado con éxito se muestra un mensaje en el ***formulario principal***.

7.1.3.1 Cerrar aplicación

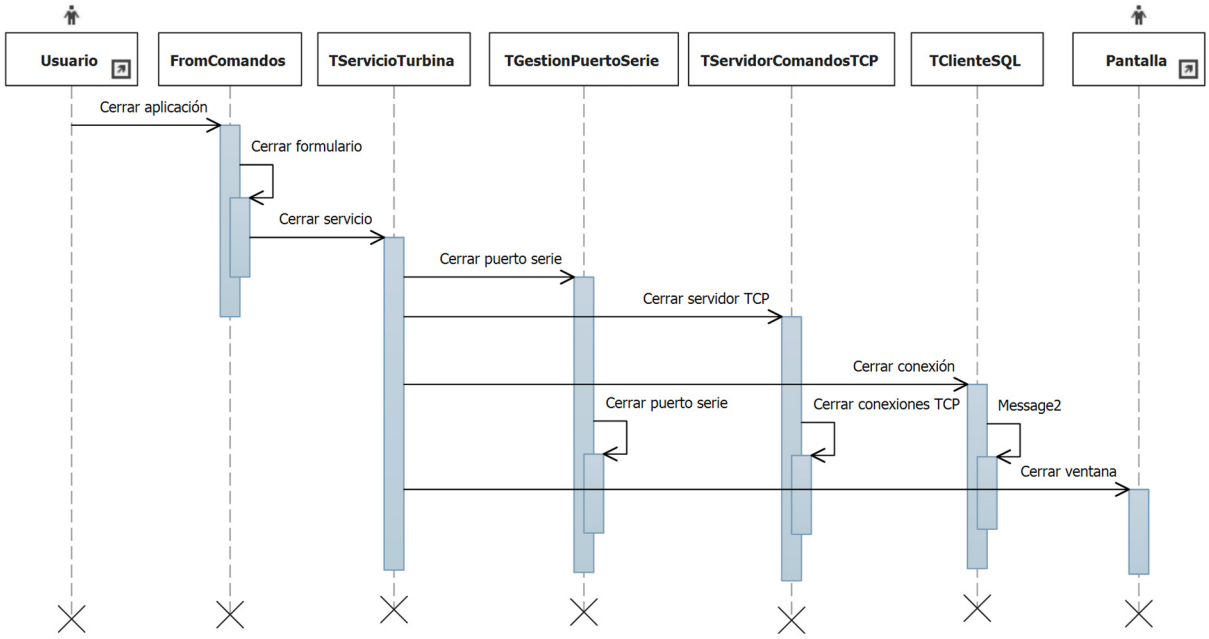


Imagen 22. Diagrama de secuencias DSP. Cerrar aplicación.

Cuando el ***usuario*** decide cerrar la aplicación, pulsa el botón correspondiente en el ***formulario principal***. Acto seguido, el ***servicio*** cierra la conexión con el ***motor SQL***, termina el ***servidor de comandos*** y cierra el ***puerto serie***.

7.1.4 Diagrama de estados

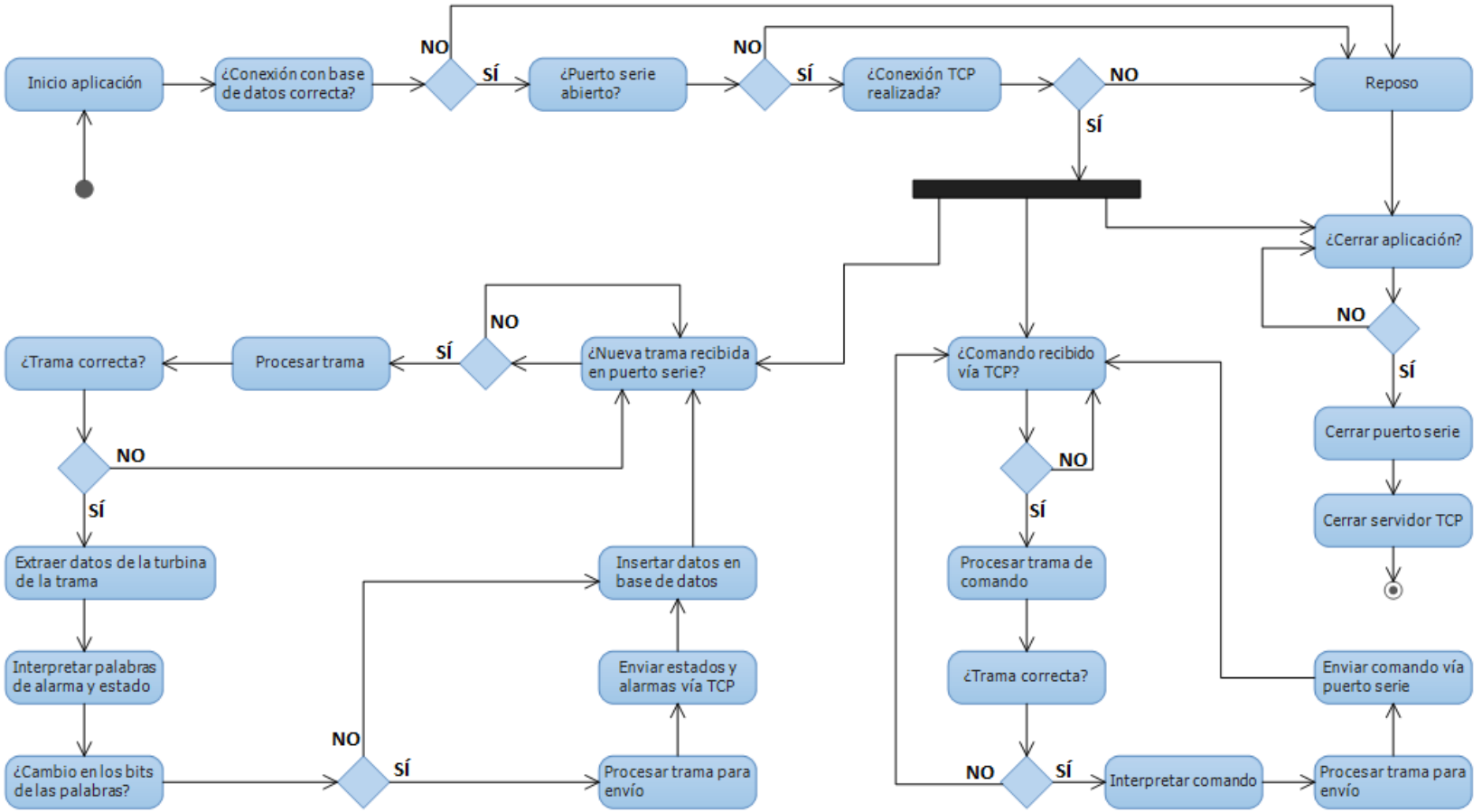


Imagen 23.Diagrama de estados aplicación DSP.

7.2 Aplicación HMI

7.2.1 Diagrama de casos de uso

A partir de las especificaciones del HMI se pueden extraer cada una de las posibles interacciones entre los actores principales y la aplicación. Como actores se han definido todos aquellos elementos externos al sistema que interactúan con el mismo:

- **Usuario:** realiza acciones de forma directa sobre la aplicación en forma de eventos según sus necesidades.
- **Aplicación DSP:** recibe datos de la turbina y comandos para el control de la misma.
- **Base de datos local:** almacena los datos deseados y permite su consulta. Se encuentra instalada en el mismo equipo que la aplicación.
- **Base de datos remota:** almacena los datos deseados y permite su consulta. Se encuentra instalada en un equipo diferente al de la aplicación.
- **Registro de Windows:** almacena datos de configuración de las aplicaciones instaladas en el equipo.
- **Pantalla:** sirve como interface visual entre el usuario y la aplicación.

Así pues, se han definido los siguientes casos de uso, algunos de los cuales incluyen subcasos de uso que pueden o no ejecutarse en función de si se cumplen ciertas condiciones:

- Iniciar aplicación.
  - Sincronizar bases de datos.
  - Abrir registro de Windows.
- Modificar configuración de las curvas.
- Mostrar gráficos.
- Exportar datos a Excel.
- Visualizar alarmas.
- Visualizar estado de la turbina.
- Enviar comandos a la turbina.
- Recibir alarmas y estado de la turbina.
- Cerrar aplicación.
  - Terminar servidor.

El diagrama de casos de uso se muestra en la *Imagen 24*.

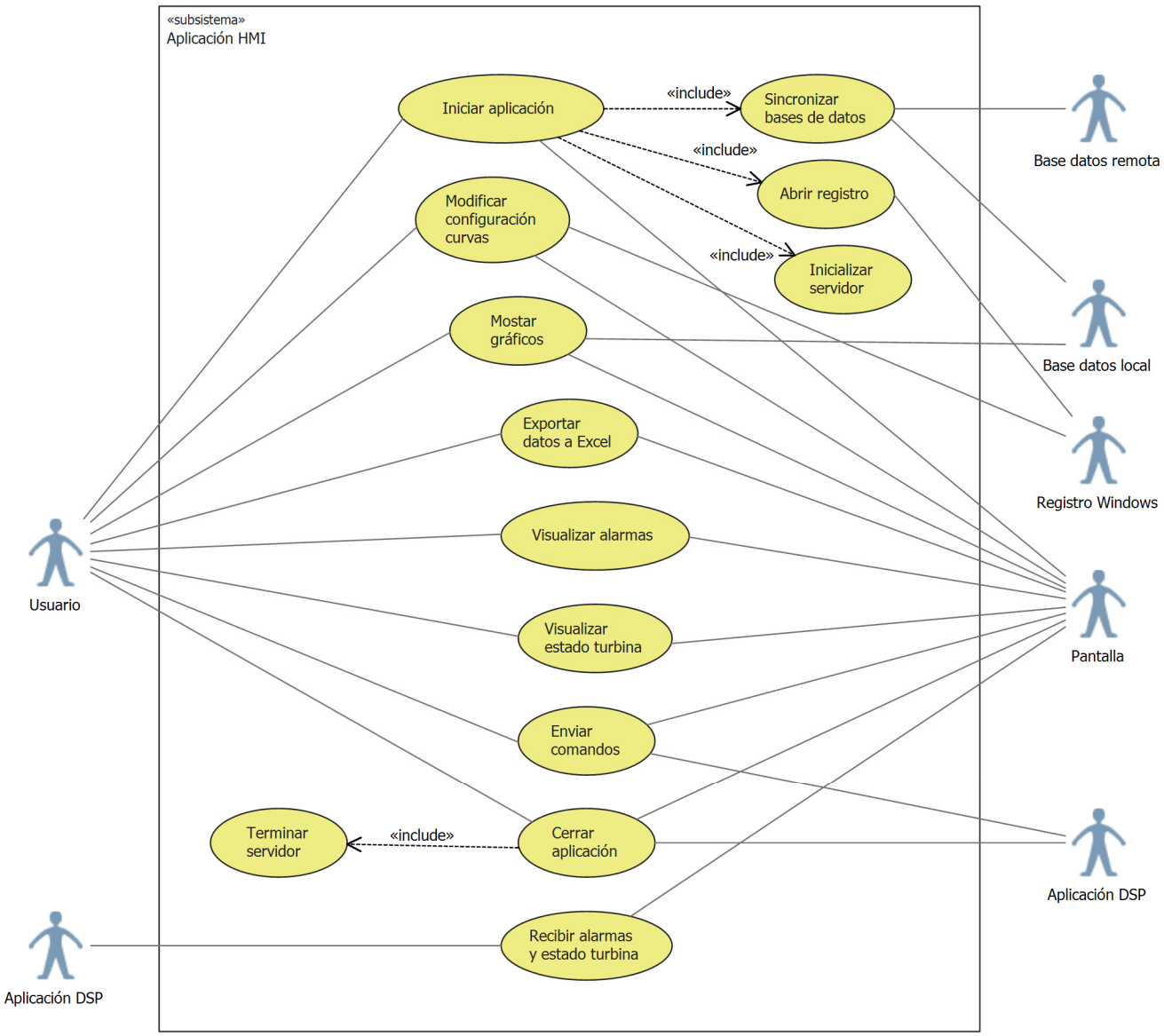


Imagen 24. Diagrama de casos de uso aplicación HMI.

A continuación se muestran cada uno de los casos de uso recogidos en el diagrama, detallando a los participantes en cada uno de ellos, los objetivos que se pretenden conseguir en cada caso de uso y los requisitos previos necesarios para conseguirlos.

7.2.1.1 Iniciar aplicación

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

- **Base de datos local**: permite el almacenamiento y consulta de datos. Se encuentra instalada en el mismo equipo que la aplicación.
- **Base de datos remota**: permite el almacenamiento y consulta de datos de forma remota. Se encuentra instalada en un equipo remoto conectado a la red.
- **Registro de Windows**: es una base de datos intrínseca al sistema operativo en la que se guardan parámetros de configuración de los diferentes programas y aplicaciones instalados en el equipo.

Precondiciones

- El usuario ha adquirido la aplicación y dispone del hardware necesario para iniciarla.
- El hardware es compatible con la aplicación.
- El usuario decide arrancar la aplicación en su dispositivo.

Garantía de éxito

El usuario consigue arrancar la aplicación y mostrarla por pantalla.

Escenario principal de éxito

i. El usuario arranca el hardware.
ii. El usuario inicia la aplicación.
iii. El servidor de estados se inicializa correctamente.
iv. La lectura inicial del registro de Windows se consigue satisfactoriamente.
v. Las conexiones con las bases de datos local y remota se realizan correctamente.
vi. Aparece el interfaz gráfico de la aplicación en la pantalla.

7.2.1.2 Sincronizar bases de datos

Actores participantes y objetivos

- **Base de datos local**: permite el almacenamiento y consulta de datos. Se encuentra instalada en el mismo equipo que la aplicación.
- **Base de datos remota**: permite el almacenamiento y consulta de datos de forma remota. Se encuentra instalada en un equipo remoto conectado a la red.

Precondiciones

- El usuario ha adquirido la aplicación y dispone del hardware necesario para iniciarla.
- El hardware es compatible con la aplicación.
- El equipo dispone de un motor de bases de datos instalado.
- Los parámetros, referentes a las bases de datos, definidos en la aplicación son correctos y coherentes.
- El servicio correspondiente al motor de bases de datos se encuentra iniciado.

Garantía de éxito

La sincronización ente bases de datos se completa correctamente.

Escenario principal de éxito

i. Se consulta la fecha del último dato presente en la base de datos local.
ii. Se consultan las filas de datos de la base de datos remota con fecha mayor que la consultada en la base de datos local.
iii. Se hace un volcado de estos datos desde la base de datos remota a la local.

7.2.1.3 Abrir registro de Windows

Actores participantes y objetivos

- **Registro de Windows**: es una base de datos intrínseca al sistema operativo en la que se guardan parámetros de configuración de los diferentes programas y aplicaciones instalados en el equipo.

Precondiciones

- El usuario ha adquirido la aplicación y dispone del hardware necesario para iniciarla.
- El hardware es compatible con la aplicación.

Garantía de éxito

La lectura del registro es correcta y los datos leídos son coherentes.

Escenario principal de éxito

i. Se comprueba la existencia del directorio del registro de la aplicación, si no se crea.
ii. Se comprueba la existencia de los distintos subdirectorios del registro de la aplicación, si no se crean.
iii. Se leen los valores de todos los directorios del registro de la turbina.
iv. Los valores leídos se aplican a la configuración actual de la aplicación.

7.2.1.4 Inicializar servidor de estados

Precondiciones

- El usuario ha iniciado la aplicación.
- El equipo dispone de al menos una tarjeta de red habilitada. Los parámetros, referentes a la tarjeta de red, definidos en la aplicación son correctos y coherentes.
- El equipo se encuentra conectado a la red local del sistema.

Garantía de éxito

El servidor de estados se ha inicializado con éxito. Es posible la recepción de datos desde la aplicación remota a través de la red local del sistema.

Escenario principal de éxito

i. El servidor de estados se inicializa correctamente con los parámetros configurados.
ii. Comienza la escucha de conexiones entrantes.

7.2.1.5 Modificar configuración de las curvas

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.
- **Registro de Windows**: es una base de datos intrínseca al sistema operativo en la que se guardan parámetros de configuración de los diferentes programas y aplicaciones instalados en el equipo.

Precondiciones

- La aplicación se ha iniciado correctamente.
- La ruta de la aplicación en el registro de Windows se ha leído o creado correctamente.
- El usuario ha decidido modificar la configuración de las curvas a mostrar.
- El usuario ha pulsado el botón que muestra la ventana en la que se pueden realizar estas modificaciones.

Garantía de éxito

El usuario modifica los parámetros deseados con éxito.

Escenario principal de éxito

i. La ventana de configuración de curvas se abre correctamente cuando el usuario desea.
ii. El usuario realiza las modificaciones que desea.
iii. El usuario guarda las modificaciones realizadas.
iv. La nueva configuración se almacena correctamente en el registro.

7.2.1.6 Mostrar gráfico

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

- **Base de datos local**: permite el almacenamiento y consulta de datos. Se encuentra instalada en el mismo equipo que la aplicación.

Precondiciones

- La aplicación se ha iniciado correctamente.
- El servicio correspondiente al motor de bases de datos se encuentra iniciado.
- El usuario ha decidido mostrar un gráfico de datos.
- El usuario ha pulsado el botón que muestra la ventana para crear un nuevo gráfico.

Garantía de éxito

El usuario muestra las curvas de datos deseadas en el rango de tiempos seleccionado.

Escenario principal de éxito

i. La ventana para mostrar un nuevo gráfico se abre correctamente cuando el usuario desea.
ii. El usuario selecciona las curvas que quiere visualizar en el rango de tiempos deseado.
iii. El usuario actualiza el gráfico.
iv. Se leen los datos actuales de la configuración de curvas.
v. Se leen los datos a mostrar de la base de datos local.
vi. Se convierten los datos a una tabla que puede ser interpretada en forma de gráfico.

7.2.1.7 Exportar datos a Excel

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

Precondiciones

- La aplicación se ha iniciado correctamente.
- Microsoft Excel se encuentra instalado en el equipo.
- El usuario ha mostrado un gráfico de datos.
- El usuario ha decidido exportar el gráfico mostrado a un archivo Excel.
- El usuario ha pulsado el botón que inicia la exportación a Excel.

Garantía de éxito

Los gráficos mostrados en la aplicación se representan en una hoja de Excel de forma individual y los datos correspondientes a las mismas se presentan en forma de tabla.

Escenario principal de éxito

i. El usuario pulsa el botón para iniciar la exportación.
ii. El usuario selecciona la ruta donde desea guardar el archivo resultante de la exportación.
iii. Se lee la tabla de datos asociada a los gráficos mostrados.
iv. Se presenta cada curva de forma individual y la tabla de datos en el archivo Excel.

7.2.1.8 Visualizar alarmas

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

Precondiciones

- La aplicación se ha iniciado correctamente.
- El usuario ha decidido mostrar la ventana de registro de alarmas.

Garantía de éxito

La ventana de registro de alarmas de la turbina se muestra correctamente.

Escenario principal de éxito

i. El usuario pulsa el botón para mostrar la ventana de registro de alarmas.
ii. La ventana se muestra correctamente con los valores actuales.

7.2.1.9 Visualizar estado de la turbina

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.

Precondiciones

- La aplicación se ha iniciado correctamente.
- El usuario ha decidido mostrar la ventana de estado de la turbina.

Garantía de éxito

La ventana de estado de la turbina se muestra correctamente.

Escenario principal de éxito

i. El usuario pulsa el botón para mostrar la ventana de estado de la turbina.
ii. La ventana se muestra correctamente con los valores actuales.

7.2.1.10 Enviar comandos

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.
- **Aplicación DSP**: procesa los datos recibidos desde la turbina y envía a la misma los comandos pertinentes demandados desde la aplicación HMI.

Precondiciones

- La aplicación HMI se ha iniciado correctamente.
- La aplicación DSP se ha iniciado correctamente.
- El usuario ha decidido enviar un comando de control a la turbina.
- El usuario ha pulsado el botón para enviar el comando correspondiente.

Garantía de éxito

El comando se envía a la turbina de forma correcta.

Escenario principal de éxito

i. La ventana para el envío de comandos de control se abre correctamente cuando el usuario desea.
ii. El usuario parametriza el comando a enviar.
iii. El usuario envía el comando pulsando el botón correspondiente.

7.2.1.11 Recibir alarmas y estado de la turbina

Actores participantes y objetivos

- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.
- **Aplicación DSP**: procesa los datos recibidos desde la turbina y envía a la misma los comandos pertinentes demandados desde la aplicación HMI.

Precondiciones

- La aplicación se ha iniciado correctamente.
- El servidor de estados se encuentra escuchando conexiones entrantes.
- Se ha producido un cambio en el estado de la turbina o en alguna de sus posibles alarmas.

Garantía de éxito

Las nuevas alarmas y el nuevo estado de la turbina se reciben correctamente. Las ventanas de alarmas y estado de la turbina muestran una alerta.

Escenario principal de éxito

i. El servidor de estados recibe una petición de conexión entrante con las nuevas alarmas y estado de la turbina.
ii. La conexión se acepta.
iii. La trama se descodifica correctamente.
iv. Las nuevas alarmas y estado de la turbina son interpretados.
v. Se muestra una alerta en las ventanas de alarmas y estado de la turbina.

7.2.1.12 Cerrar aplicación

Actores participantes y objetivos

- **Usuario**: desea observar de una manera clara el comportamiento de la aplicación y hacer uso de las diferentes opciones para las que está destinada.
- **Pantalla**: muestra la ventana de la aplicación con sus diferentes elementos.
- **Base de datos local**: permite el almacenamiento y consulta de datos. Se encuentra instalada en el mismo equipo que la aplicación.
- **Base de datos remota**: permite el almacenamiento y consulta de datos de forma remota. Se encuentra instalada en un equipo remoto conectado a la red.

Precondiciones

- La aplicación se encuentra iniciada.
- El usuario ha decidido cerrar la aplicación voluntariamente.

Garantía de éxito

La aplicación se cierra correctamente y se deja de mostrar por pantalla.

Escenario principal de éxito

i. El usuario pulsa el botón de cierre de la aplicación.
ii. El servidor de estados deja de escuchar conexiones entrantes.
iii. Se cierran las conexiones con las bases de datos local y remota.
iv. La aplicación deja de mostrarse por pantalla.

7.2.2 Diagrama de clases

A través del diagrama de clases se describe la estructura estática del sistema en términos de clases y de relaciones entre las mismas. Se ha pretendido encapsular al máximo el programa, dentro de unos límites razonables, creando una clase por cada interacción usuario-sistema. Además, se han creado algunas clases necesarias para poder llevar a cabo algunos métodos de algunas de las clases. De esta manera, al igual que para el DSP, cada clase tendrá una única instancia de sí misma, o lo que es lo mismo, solo existirá un objeto asociado a cada una de las clases. Por ello, el diagrama de objetos será análogo al de clases.

Con el fin de facilitar la comprensión del diagrama, las clases se muestran sin sus atributos ni métodos. Únicamente se muestran las relaciones existentes entre todas ellas. No obstante, en los anexos se pueden observar cada una de las clases con sus respectivos atributos y métodos.

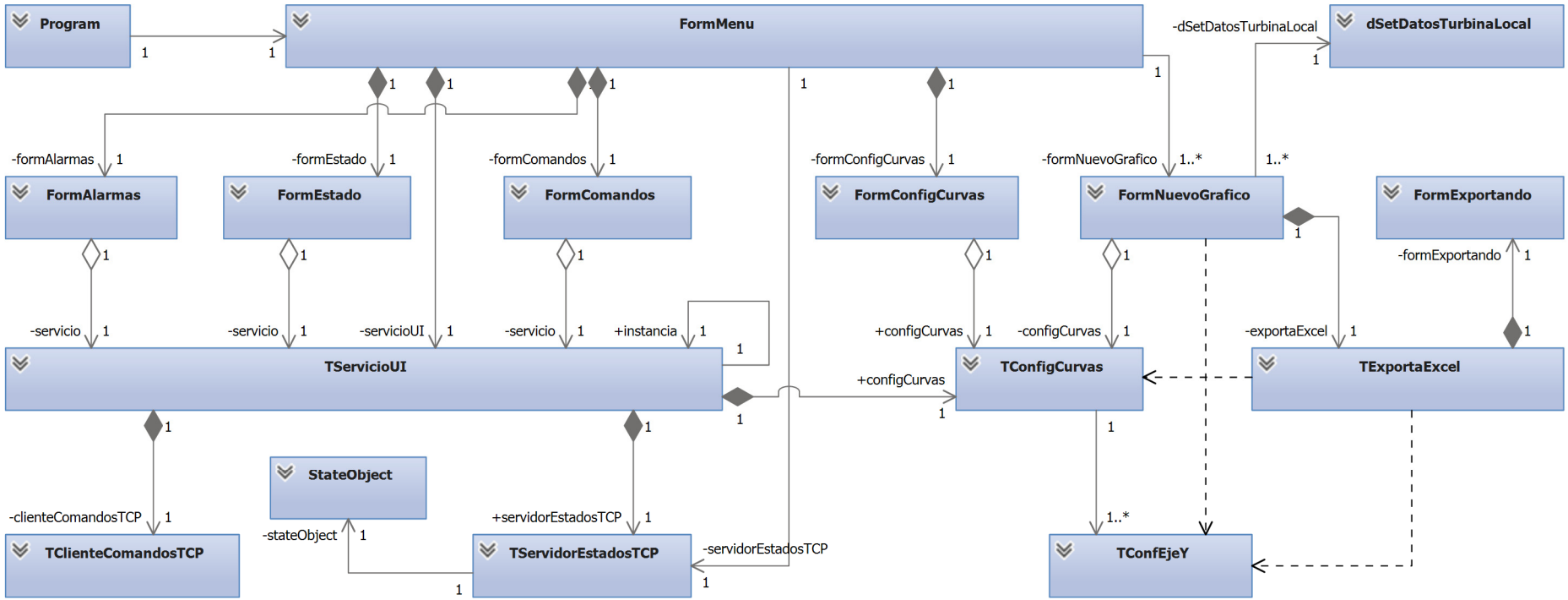


Imagen 25. Diagrama de clases aplicación HMI.

La estructura del programa es la siguiente:

- La clase ***Program*** es la clase principal de la aplicación. En ella se inicializa el formulario principal junto con su correspondiente hilo.
- La clase ***FormMenu*** realiza todas las tareas referentes al interface de usuario. Actúa como menú principal de la aplicación y, por tanto, desde ella se permite el acceso al resto de formularios de la aplicación. Por tanto, es la encargada de inicializar dichos formularios y de lanzar los hilos correspondientes a los mismos, así como de cancelarlos cuando se cierra la aplicación.
- La clase ***TServicioUI*** gestiona todo el modelo de negocio de la aplicación, o lo que es lo mismo, gestiona todos los eventos y pasos de parámetros entre las diferentes clases del programa. Como se observa en el diagrama de clases, se podría decir que es la clase central del programa. Todas las clases que interactúan con algún actor del sistema se crean y se destruyen con esta clase. Gestiona, también todos los aspectos relativos a la sincronización entre las bases de datos local y remota.
- La clase ***FormAlarmas*** se corresponde con el formulario que muestra las alarmas actuales existentes en la turbina. Todos los eventos relacionados con esta clase son gestionados por el servicio.
- La clase ***FormEstado*** se corresponde con el formulario que muestra el estado actual de la turbina. Todos los eventos relacionados con esta clase son gestionados por el servicio.
- La clase ***FormComandos*** se corresponde con el formulario en el que se pueden configurar los comandos que se enviarán a la turbina. Todos los eventos relacionados con esta clase son gestionados por el servicio.
- La clase ***FormConfigCurvas*** se corresponde con el formulario en el que se puede modificar la configuración de las curvas que se mostrarán en los gráficos. Todos los eventos relacionados con esta clase son gestionados por el servicio.
- La clase ***TConfigCurvas*** almacena la información relativa a la configuración actual de las curvas, rangos de representación y colores, con la ayuda de la siguiente clase. Realiza todas las tareas de interacción con el registro de Windows.
- La clase ***TConfEjeY*** almacena la información relativa al rango de representación de las diferentes curvas.
- La clase ***FormNuevoGrafico*** se corresponde con el formulario que permite mostrar nuevas curvas de datos de un rango de tiempos determinado. Realiza todas las tareas relacionadas con las consultas a la base datos local con la ayuda de la siguiente clase.
- La clase ***dSetDatosTurbinaLocal*** almacena los resultados de la consulta realizada para mostrar un nuevo gráfico.
- La clase ***FormExportando*** se corresponde con el formulario que informa del progreso de la exportación a Excel.

- La clase ***TExportaExcel*** realiza todas las tareas relacionadas con la exportación de datos a Excel. Crea el hilo correspondiente a la exportación, cuando ésta se inicia, y lo cancela cuando termina. Se en carga de realizar todas las tareas relacionadas con la presentación de los datos en formato Excel.
- La clase ***TServidorEstadosTCP*** realiza todas las tareas referentes a la recepción de las alarmas y el estado de la turbina procedentes de la aplicación DSP. Una vez inicializado, el servidor se encuentra escuchando conexiones entrantes. Cuando recibe una y la acepta informa al servicio de que la recepción se ha completado con éxito.
- La clase ***StateObject*** ayuda a la escucha asincrónica del servidor de estados.
- La clase ***TClienteComandosTCP*** realiza todas las tareas referentes al envío de comandos a la turbina cuando el usuario así lo ha considerado. El envío se realiza a través de la red local mediante protocolo TCP/IP por lo que, para que el envío sea correcto, debe haber un servidor de comandos escuchando en la aplicación remota. Al finalizar el envío, bien haya sido de forma satisfactoria o no, se informa al servicio.

7.2.3 Diagramas de secuencias

Al igual que para el DSP, se muestran los diagramas de secuencias correspondientes con los casos de uso anteriormente descritos. Cabe destacar que no se muestra un diagrama por cada caso de uso, sino que dentro de uno mismo pueden incluirse varios por tener algún tipo de relación. Como ya se ha dicho, por claridad solo se muestran los diagramas de secuencias teniendo en cuenta los escenarios principales de éxito.

7.2.3.1 Iniciar aplicación

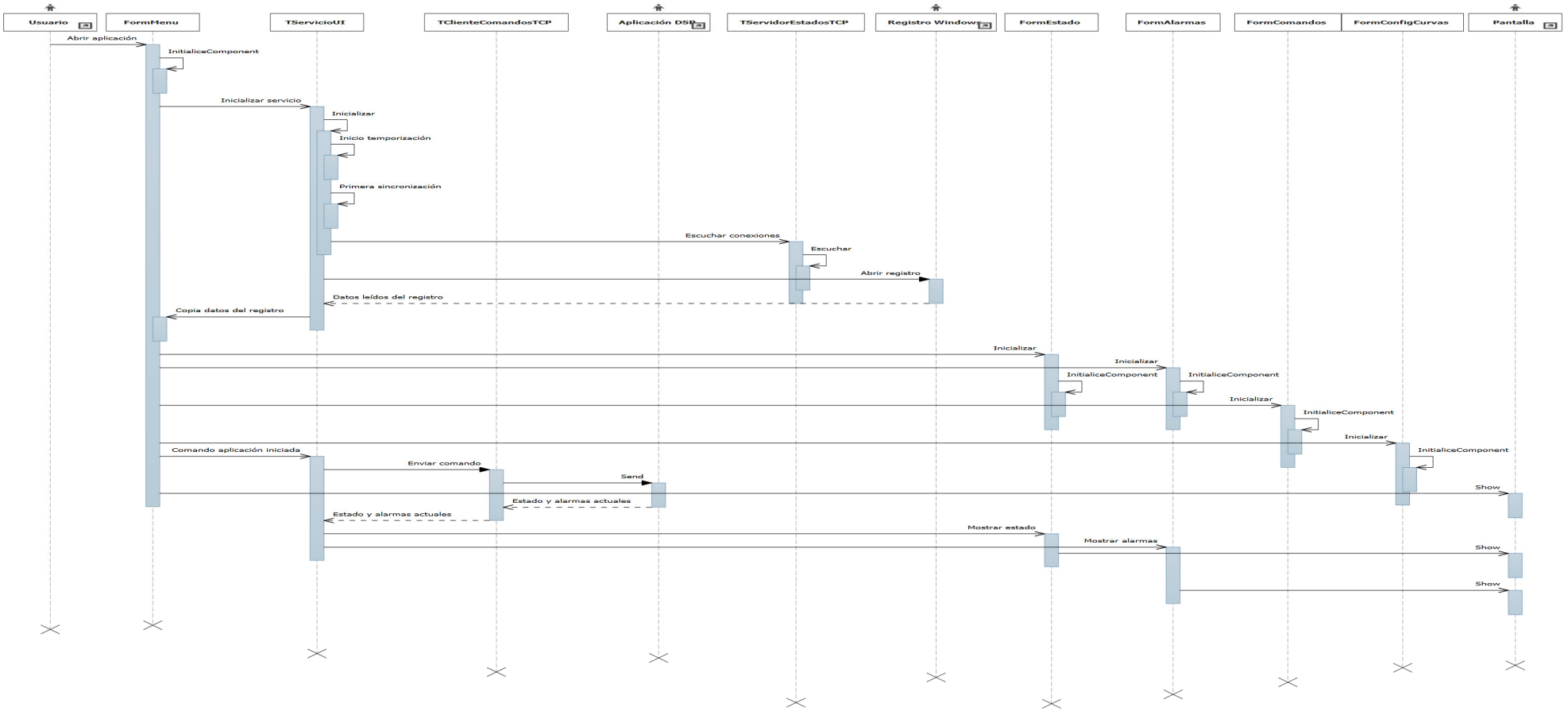


Imagen 26. Diagrama de secuencias HMI. Iniciar aplicación.

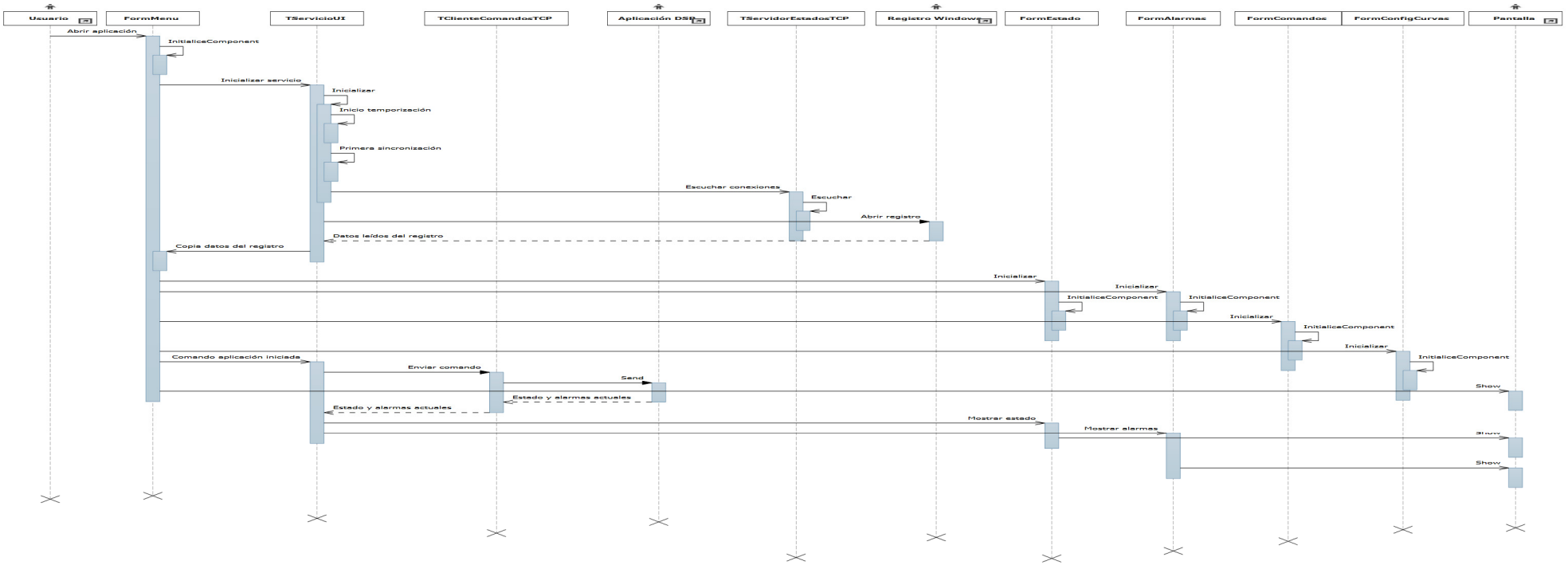


Imagen 27. Diagrama de secuencias HMI. Iniciar aplicación (continuación).

Cuando el ***usuario*** decide iniciar la aplicación se inicializa el ***formulario menú***, que será el principal. Esta clase inicializa el ***servicio*** de la aplicación, el cual sigue la misma filosofía que el descrito para el DSP. En la inicialización del servicio se realiza la primera sincronización entre las ***bases de datos local y remota*** y se inicia la temporización para futuras sincronizaciones. Posteriormente, se inicializa el ***servidor de estados*** y se hace una lectura de la ruta de la aplicación en el ***registro de Windows***. A continuación, se inicializan los ***formularios de estado*** de la turbina, ***de alarmas***, ***de envío de comandos*** y ***de configuración de curvas***, ya que de ellos solo va a existir una instancia. Finalmente se ***envía un comando*** a la ***aplicación DSP*** que informe de que la aplicación HMI se ha iniciado correctamente.

7.2.3.1 Sincronizar bases de datos

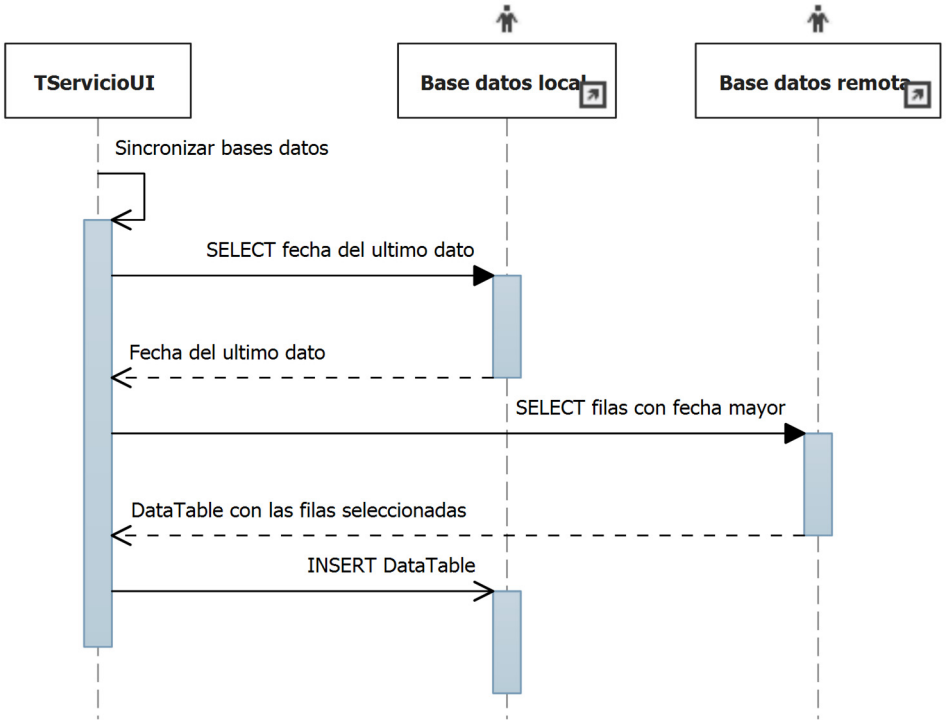


Imagen 28. Diagrama de secuencias HMI. Sincronizar bases de datos.

Cuando el ***servicio*** lo indica, se realiza una sincronización entre las ***bases de datos local y remota***. Se realiza una consulta a la base de datos local que devuelva la fecha del último dato. A continuación, se realiza una consulta a la base de datos remota que devuelva todas las filas con fecha mayor que la almacenada. Finalmente, se insertan dichas filas en la base de datos local.

7.2.3.2 Modificar configuración curvas

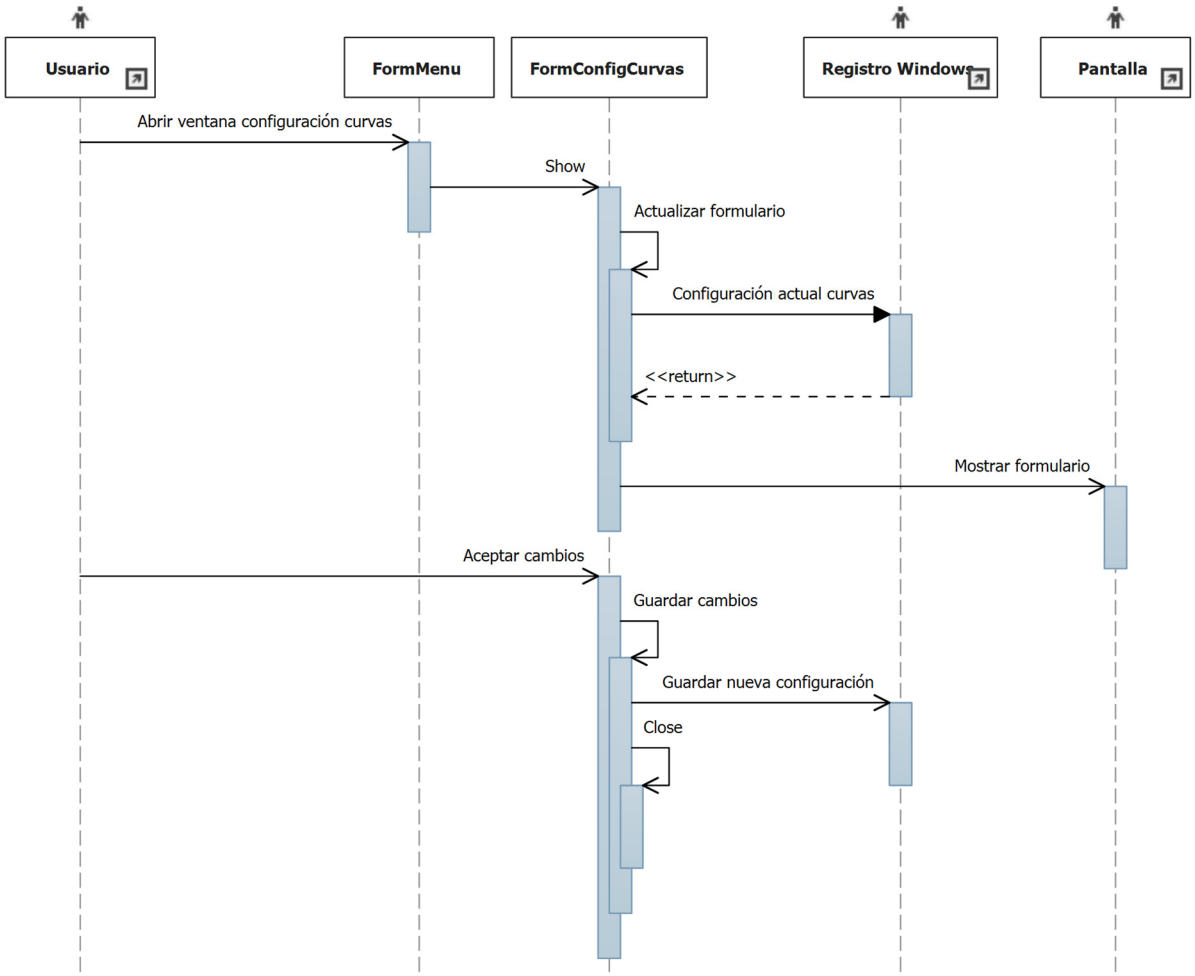


Imagen 29. Diagrama de secuencias HMI. Modificar configuración de curvas.

Cuando el ***usuario*** decide abrir el ***formulario de configuración de curvas,*** pulsa el botón correspondiente en el ***formulario menú***. Este formulario ya se ha inicializado en el proceso de apertura de la aplicación, por lo que directamente pasa a actualizarse su contenido con los parámetros de configuración guardados en el ***registro de Windows***.

Cuando el usuario desea guardar los cambios deseados pulsa el botón correspondiente en el formulario y comienza la escritura de los nuevos parámetros en la ruta de la aplicación en el registro de Windows para, finalmente, cerrar el formulario.

7.2.3.3 Mostrar gráfico

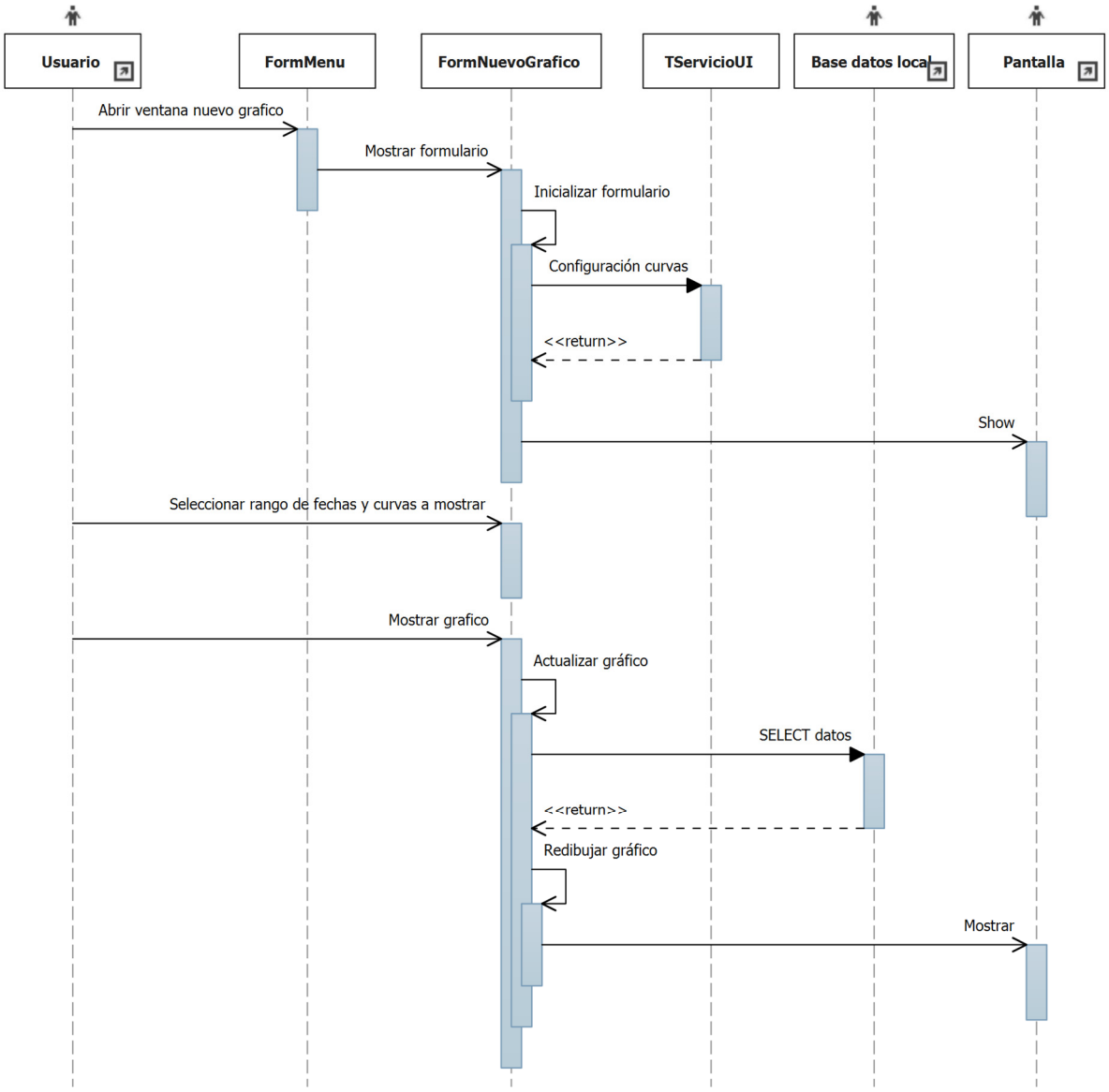


Imagen 30. Diagrama de secuencias HMI. Mostrar gráfico.

Cuando el ***usuario*** decide mostrar una nueva curva con los datos existentes, pulsa el botón correspondiente en el ***formulario menú***, que abre el ***formulario de nuevo gráfico***. Como se van a poder tener varios formularios de este tipo abiertos al mismo tiempo, cada uno será una nueva instancia que deberá inicializarse. En la inicialización se pide al ***servicio*** la configuración actual de las curvas.

Antes de mostrar las curvas, el usuario debe seleccionar un rango coherente de fechas y las curvas de las que desea visualizar datos.

Cuando el usuario decide mostrar los datos seleccionados, pulsa el botón correspondiente en el formulario. Se realiza entonces una consulta a la ***base de datos local***, que devuelve las filas correspondientes con los datos seleccionados. Finalmente se actualiza el gráfico y se muestra por ***pantalla***.

7.2.3.4 Exportar gráficas a Excel

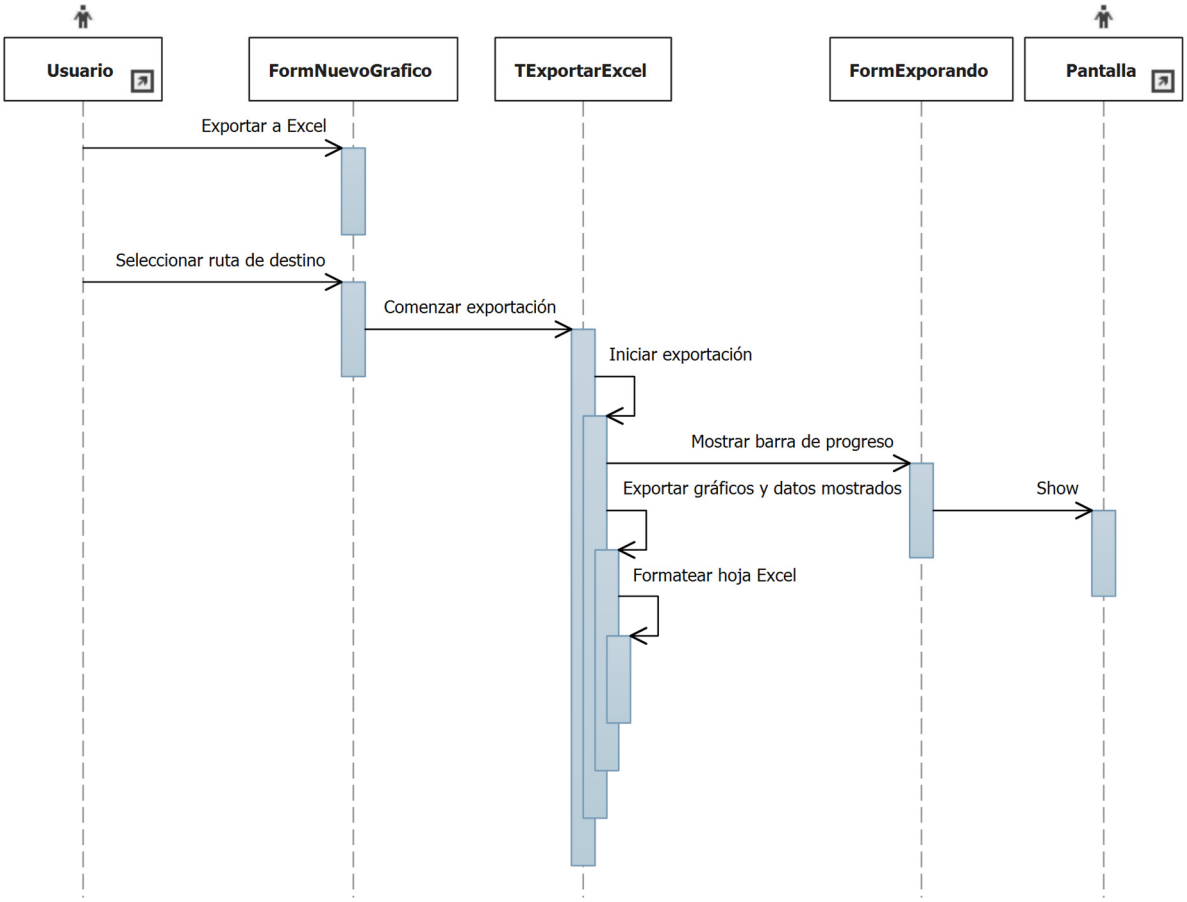


Imagen 31. Diagrama de secuencias HMI. Exportar gráficas a Excel.

Cuando el ***usuario*** decide exportar a un archivo Excel los datos que se están visualizando en el ***formulario del gráfico***, pulsa el botón correspondiente para, a continuación, seleccionar la ruta en la que se guardará el archivo. Comienza entonces el tratamiento de los datos para la ***exportación a Excel***, cuyo estado se sigue a través del ***formulario de progreso***.

7.2.3.5 Recibir alarmas y estado

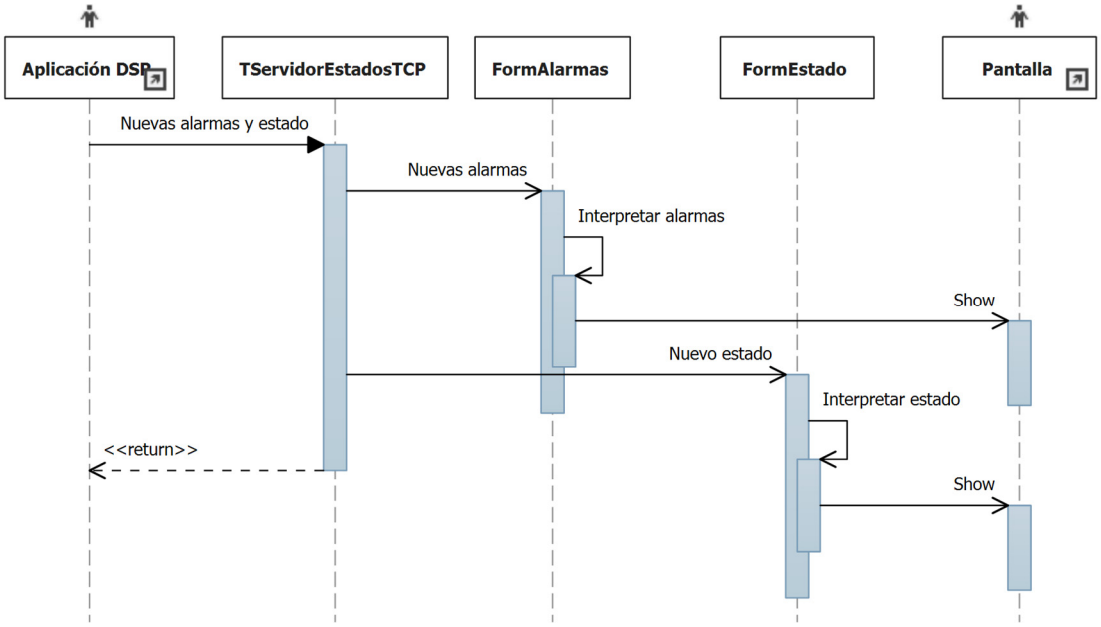


Imagen 32. Diagrama de secuencias HMI. Recibir alarmas y estado.

En el momento en que el ***servidor de estados*** recibe nuevas alarmas y estado de la turbina desde la ***aplicación DSP***, las interpreta y las muestra en los ***formularios de alarmas y estado*** según corresponda.

7.2.3.6 Visualizar alarmas

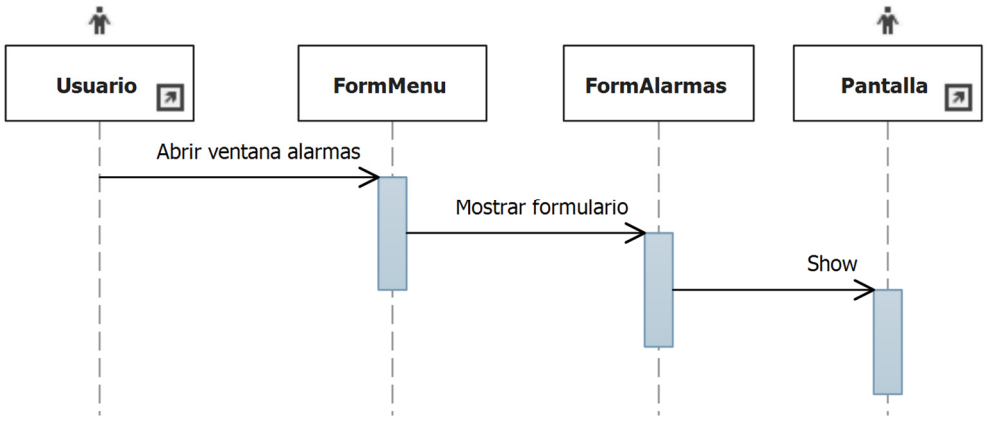


Imagen 33. Diagrama de secuencias HMI. Visualizar alarmas.

Cuando el ***usuario*** decide abrir el ***formulario de alarmas***, pulsa el botón correspondiente en ***formulario menú*** para que se muestre por ***pantalla***.

7.2.3.7 Visualizar estado turbina

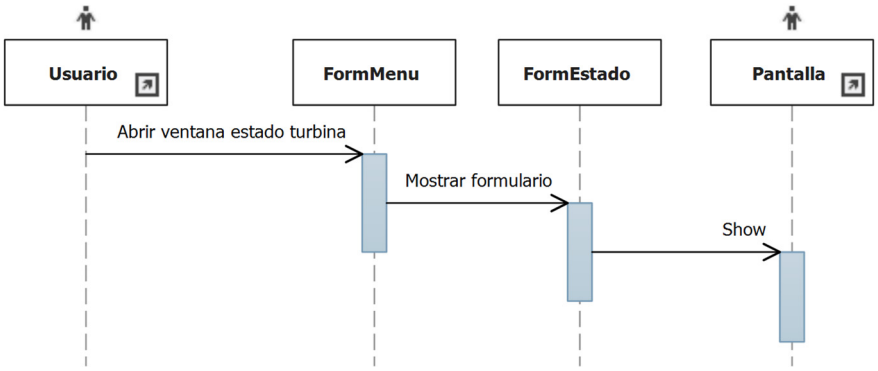


Imagen 34. Diagrama de secuencias HMI. Visualizar estado de la turbina.

Cuando el ***usuario*** decide abrir el ***formulario de estado*** de la turbina, pulsa el botón correspondiente en ***formulario menú*** para que se muestre por ***pantalla***.

7.2.3.1 Enviar comandos al DSP

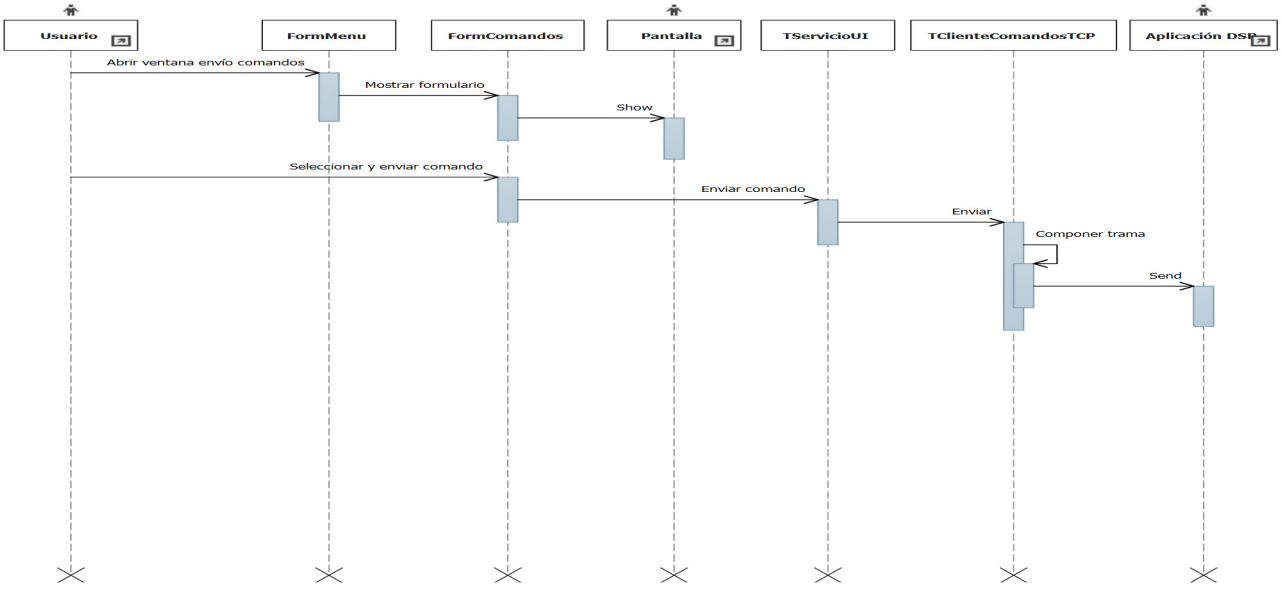


Imagen 35. Diagrama de secuencias HMI. Enviar comandos al DSP.

Cuando el ***usuario*** desea enviar un comando a la ***aplicación DSP***, pulsa el botón correspondiente en el ***formulario menú*** para abrir el ***formulario de envío de comandos***.

Cuando el usuario ha parametrizado el comando que desea enviar, pulsa el botón correspondiente de envío. Se informa al ***servicio*** para que lo envíe a la aplicación DSP a través del ***cliente de comandos TCP***.

7.2.3.1 Cerrar aplicación

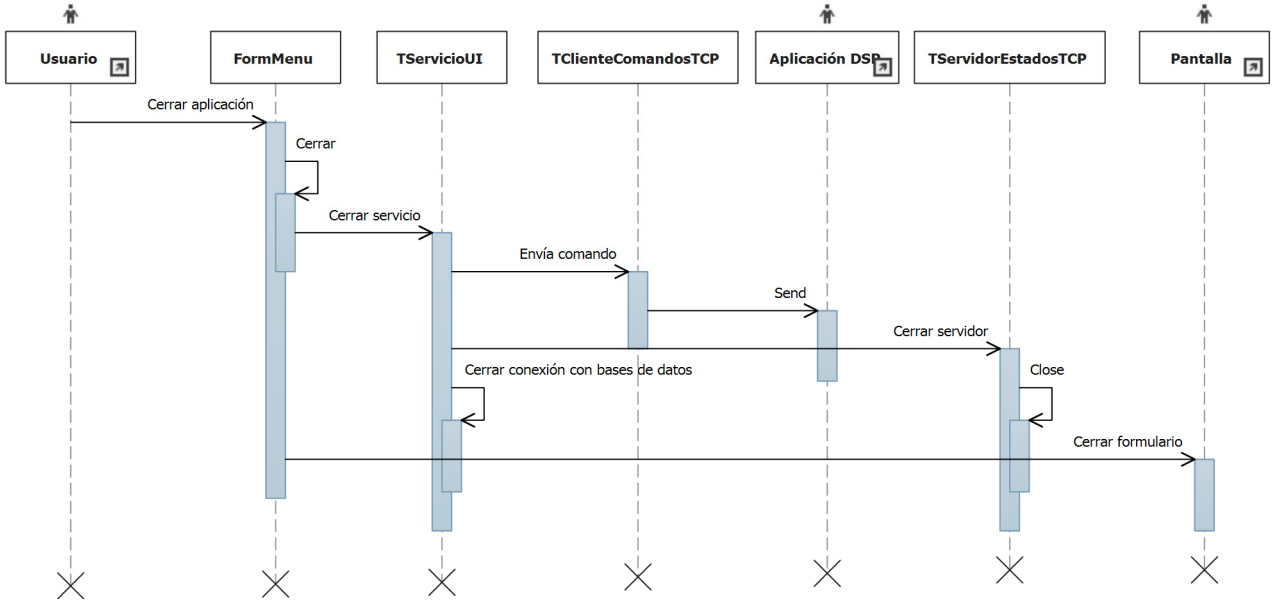


Imagen 36. Diagrama de secuencias HMI. Cerrar aplicación.

Cuando el ***usuario*** decide cerrar la aplicación, lo hará a través del ***formulario menú***. Este formulario cerrará el resto de formularios que estén abiertos e informará al ***servicio*** de que debe cerrar las conexiones con las bases de datos, enviar un comando informativo a la ***aplicación DSP*** y terminar las escuchas del ***servidor de estados***.

7.2.4 Diagramas de estados

7.2.4.1 Diagrama de estados general

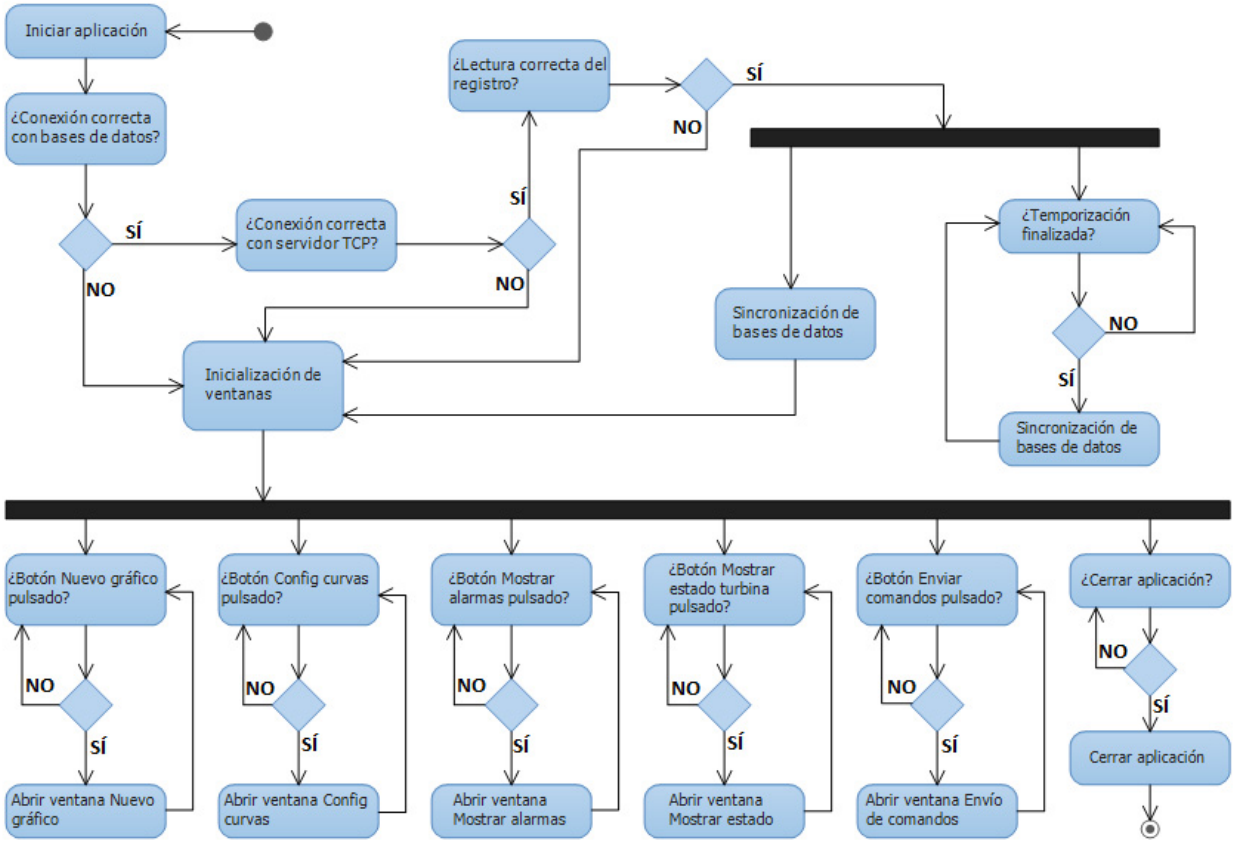


Imagen 37. Diagrama de estados HMI. General.

7.2.4.2 Abrir ventana de visualización de alarmas

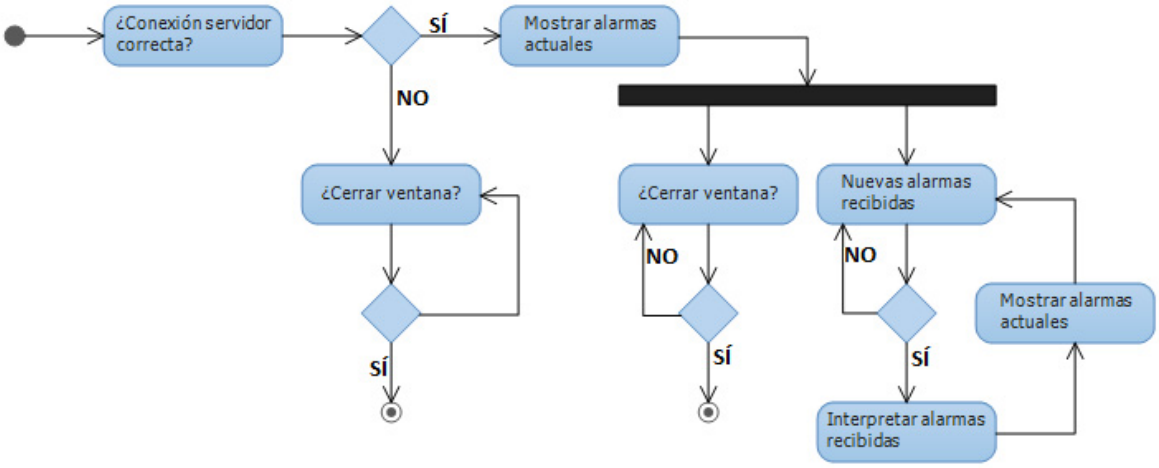


Imagen 38. Diagrama de estados HMI. Abrir ventana de visualización de alarmas.

7.2.4.3 Abrir ventana configuración de curvas

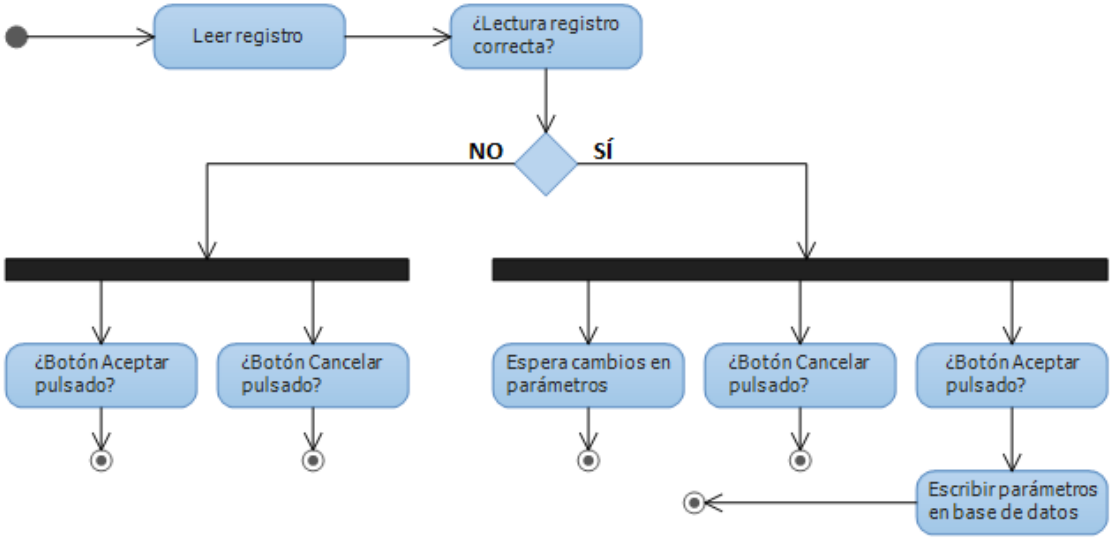


Imagen 39. Diagrama de estados HMI. Abrir ventana de configuración de curvas.

7.2.4.4 Abrir ventana de envío de comandos

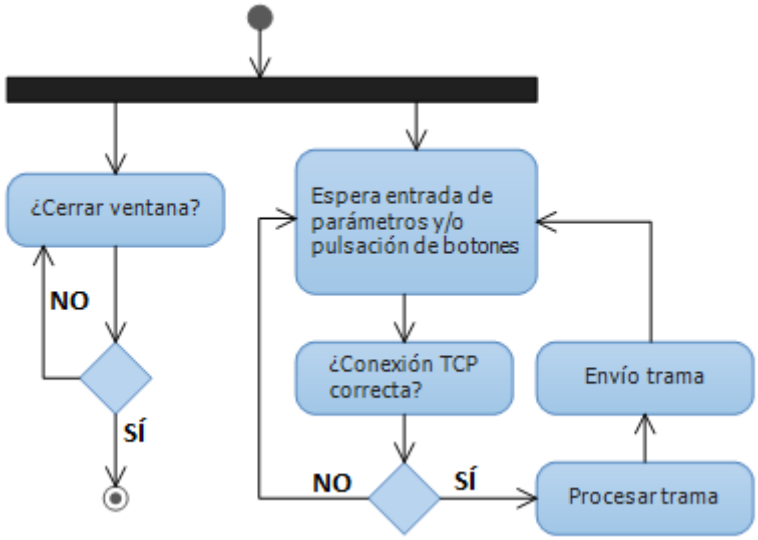


Imagen 40. Diagrama de estados HMI. Abrir ventana de envío de comandos.

7.2.4.5 Abrir ventana de visualización del estado de la turbina

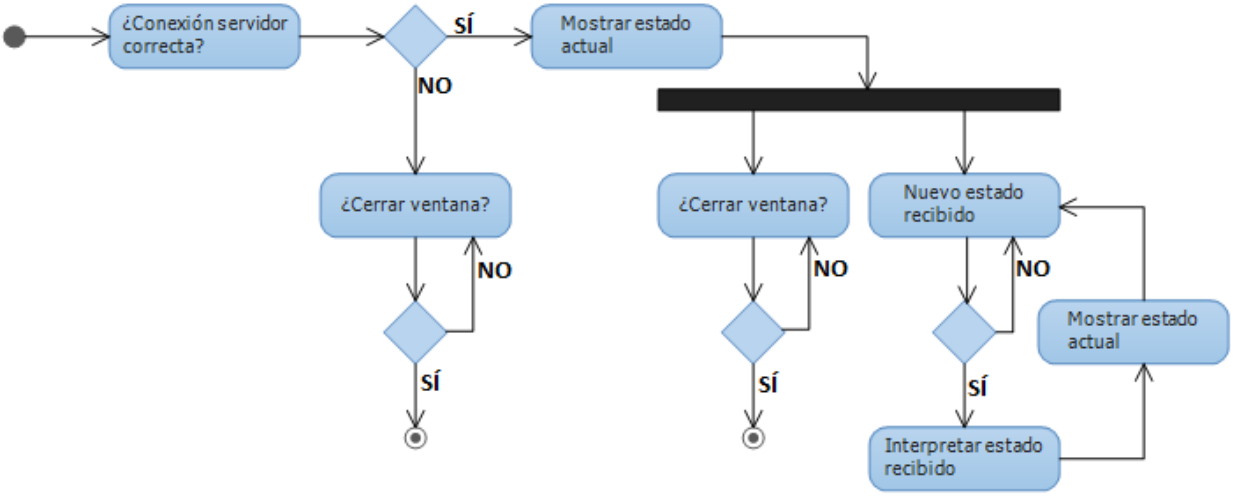


Imagen 41. Diagrama de estados HMI. Abrir ventana de visualización estado turbina.

7.2.4.1 Sincronización de bases de datos

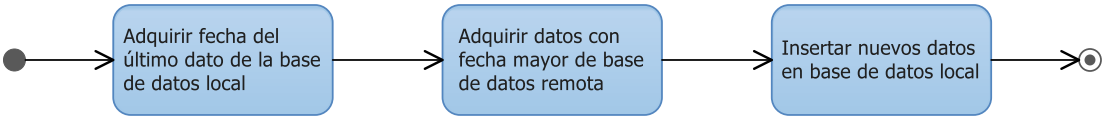


Imagen 42. Diagrama de estados HMI. Sincronización de bases de datos.

7.2.4.2 Abrir ventana para generar nuevo gráfico

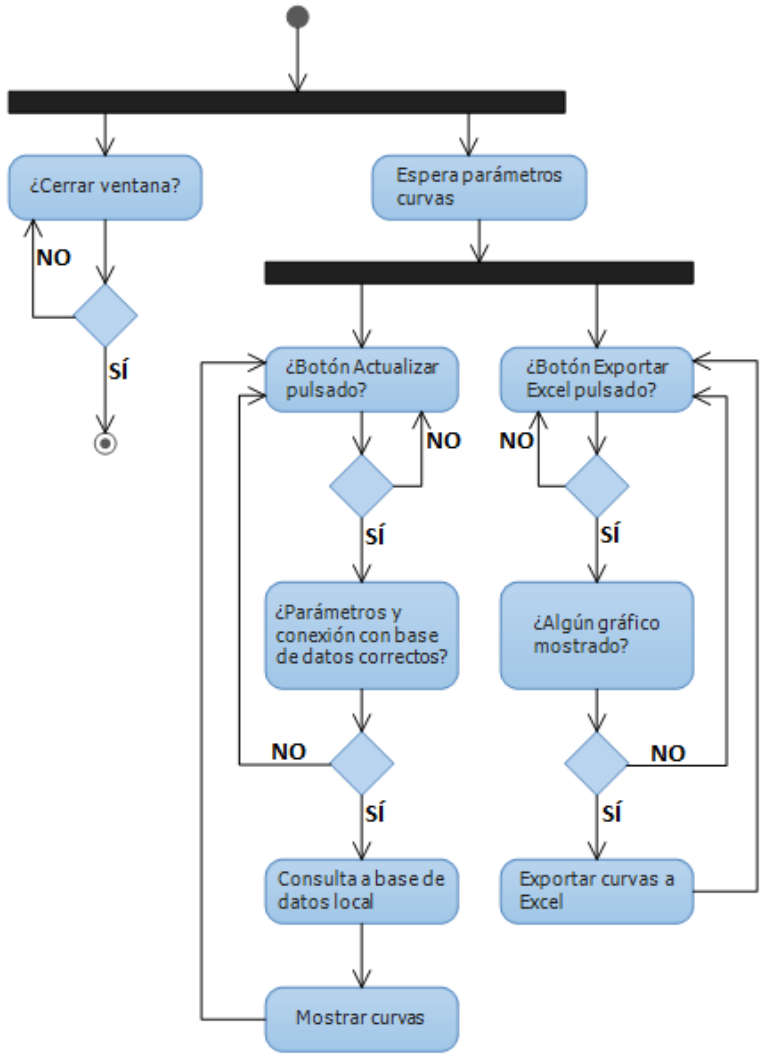


Imagen 43. Diagrama de estados HMI. Abrir ventana de generación de gráficos.

8 PROCEDIMIENTO DE PUESTA EN MARCHA

En este apartado se muestran los pasos y configuraciones que hay que realizar en cada uno de los equipos para poner en funcionamiento todo el sistema. El proceso es muy similar para ambos equipos, aunque cada uno presenta sus particularidades.

8.1 Equipo DSP

8.1.1 Configuración del motor de bases de datos

1) Es necesario instalar el motor de bases de datos Microsoft SQL Server 2012 o superior en el equipo.
2) El equipo en sí mismo actuará como servidor de bases de datos con las siguientes características o similares.

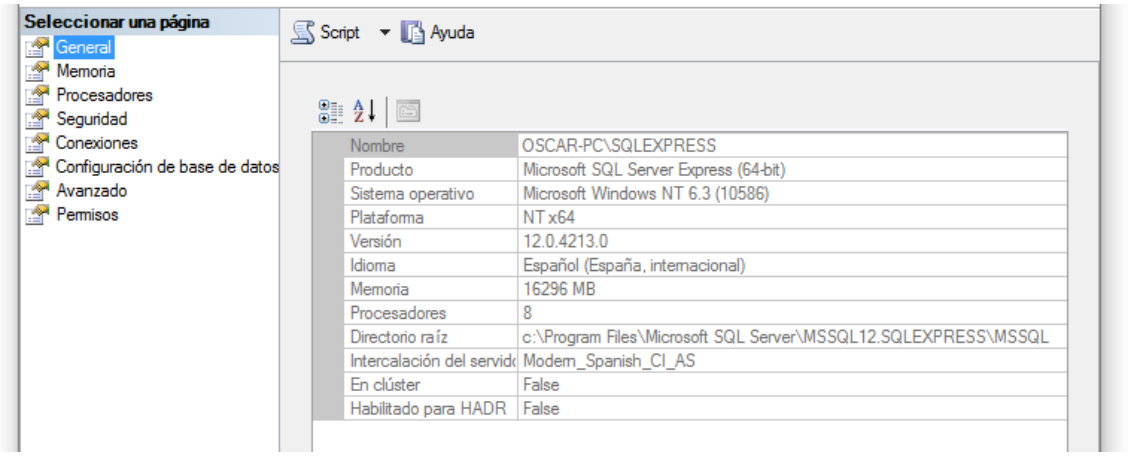


Imagen 44. Propiedades del servidor de bases de datos del DSP (I).

3) Como aspectos clave del servidor de bases de datos cabe destacar el modo de conexión y los permisos de acceso al mismo. Se pueden establecer dos formas de acceso al servidor de bases de datos: mediante usuario y contraseña propios de SQL o a través del usuario y contraseña de inicio de sesión de Windows.

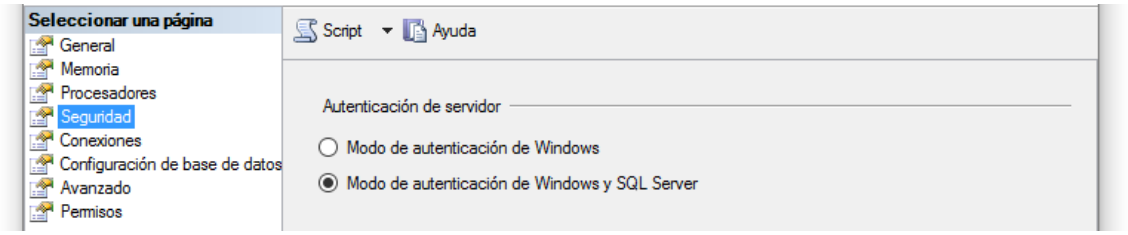


Imagen 45. Propiedades del servidor de bases de datos del DSP (II).

4) En este caso se permite el acceso de ambas maneras por si más adelante se decide crear más usuarios. Como se ve en la siguiente imagen, se ha dado permiso de conexión al servidor al usuario *administrador* y al usuario del equipo actual.

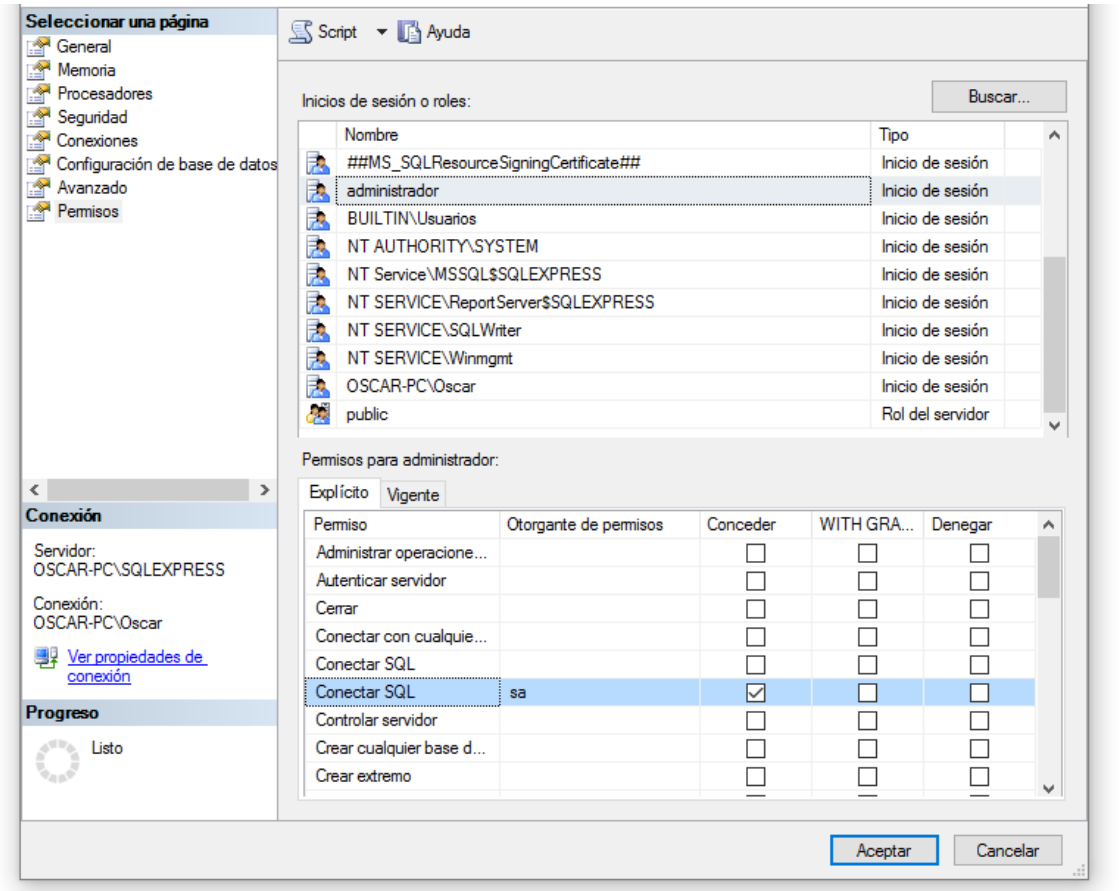


Imagen 46. Propiedades del servidor de bases de datos del DSP (III).

5) A continuación, se signan contraseñas y privilegios de seguridad a cada usuario. Por defecto se otorgan privilegios de administrador del sistema a ambos usuarios *(sysadmin)*.

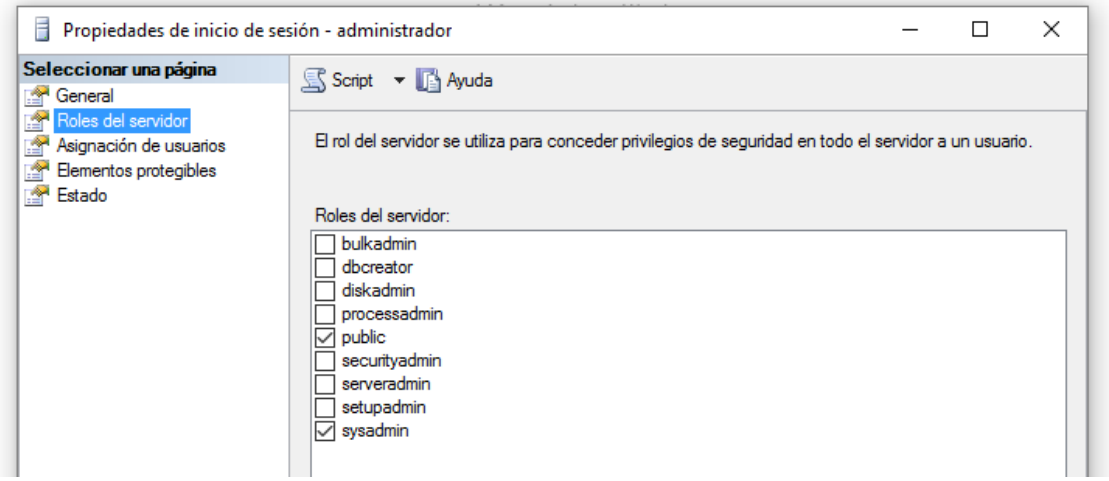


Imagen 47. Propiedades de inicio de sesión al servidor de bases de datos del DSP.

6) Dentro del servidor configurado se crea la base de datos *Turbina* con las siguientes características.

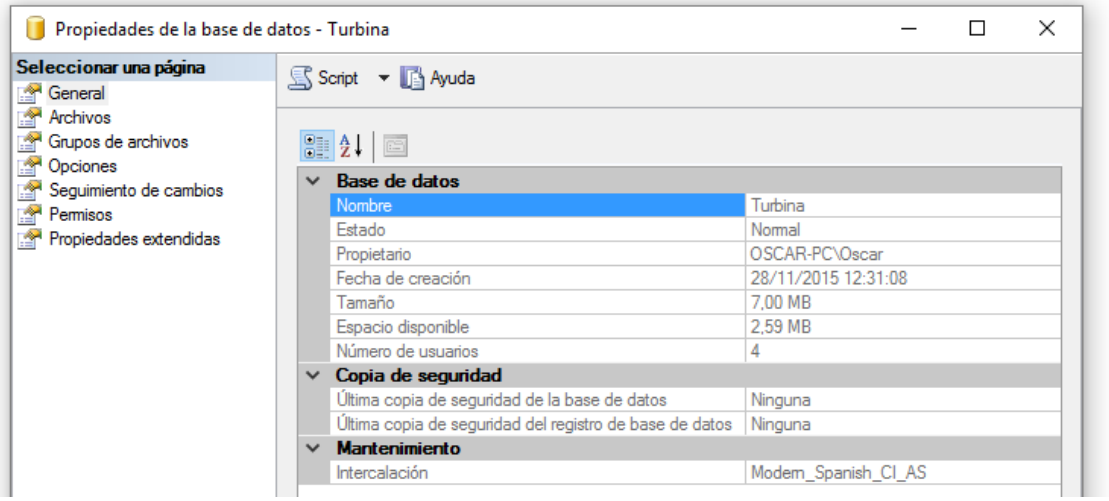


Imagen 48. Propiedades de la base de datos "Turbina" del DSP.

7) Finalmente, se crea la tabla que almacenará la información de la turbina.

```
CREATE TABLE [dbo].[tablaTurbina](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [dtFecha] [datetime] NOT NULL,
    [velGenerador] [real] NULL,
    [presionCamara] [real] NULL,
    [posicionCilindro] [real] NULL,
    [potencia] [real] NULL,
    [vibraciones] [real] NULL,
    [corrienteIU] [real] NULL,
    [corrienteIV] [real] NULL,
    [tensionVDC] [real] NULL,
    [temperatura1] [real] NULL,
    [temperatura2] [real] NULL,
    [comandoPosValvula] [real] NULL,
    [refVelocidadGiroMax] [real] NULL,
    [bError1] [binary](50) NULL,
    [bError2] [binary](50) NULL,
    [bError3] [binary](50) NULL,
    [bEstado] [binary](50) NULL,
 CONSTRAINT [PK_tablaTurbina] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
) ON [PRIMARY]

GO
```

Imagen 49. Diseño de la tablaTurbina.

Así pues, el esquema del servidor de bases de datos en el equipo DSP será el siguiente:

- Servidor: *nombreEQUIPO\nombreSERVIDOR*
  - Base de datos: *Turbina*
    - Tabla: *tablaTurbina*

Por otro lado, es necesario habilitar el protocolo de comunicaciones *TCP/IP* para que el servidor de bases de datos permita el acceso desde otros equipos de la red.

Para ello, iniciar el *SQL Server Configuration Manager* y en la configuración de red del servidor habilitar el protocolo TCP/IP y definir el puerto de escucha, tal y como se muestra en la siguiente imagen.

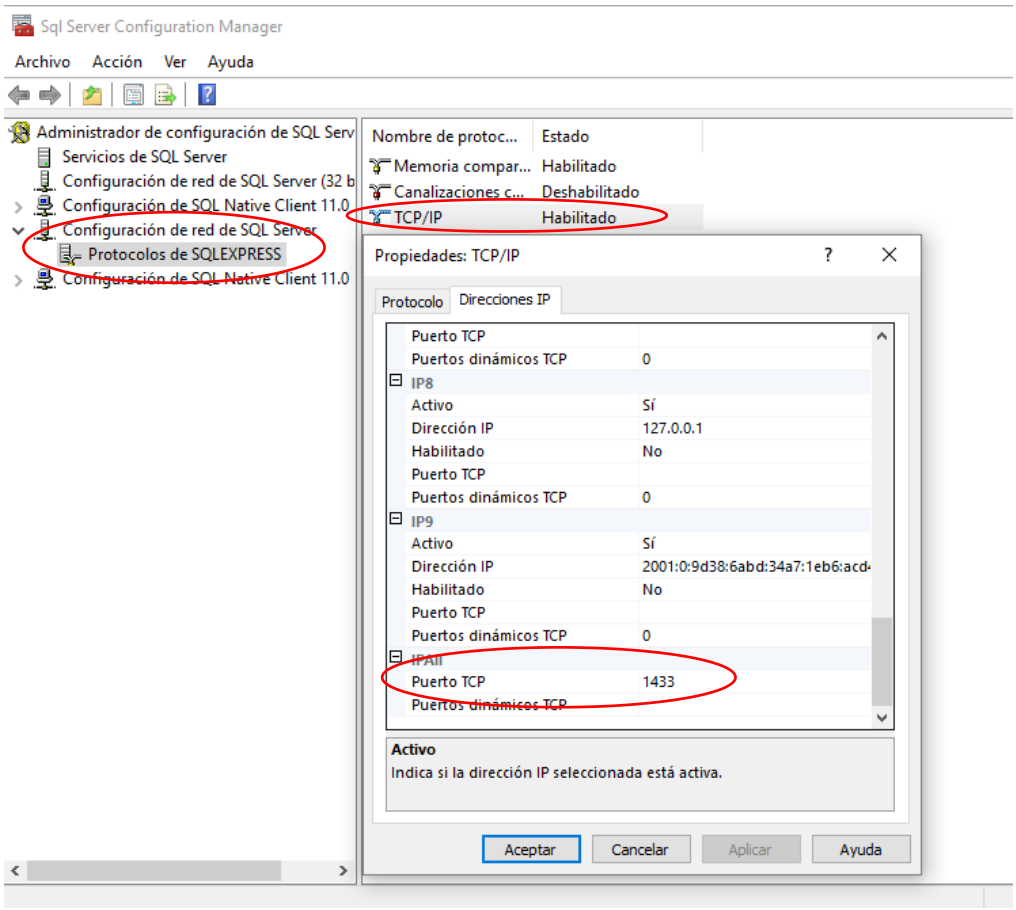


Imagen 50. Propiedades TCP/IP de la base de datos "Turbina".

8.1.2 Configuración del puerto serie

En el menú *Hardware y sonido* del *Panel de control* de Windows, iniciar el *Administrador de dispositivos* para configurar el puerto serie con los parámetros definidos en las especificaciones técnicas. Debe tenerse en cuenta el número de puerto que se va a utilizar, ya que es un parámetro a configurar a posteriori en la aplicación.

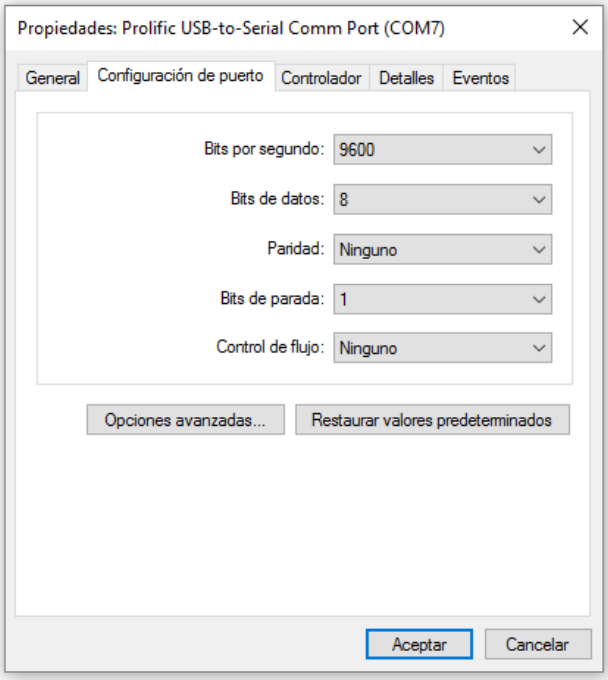


Imagen 51. Configuración del puerto serie del DSP.

8.1.3 Configuración de la tarjeta de red

En el menú *Redes e Internet* del *Panel de control* de Windows, iniciar el *Centro de redes y recursos compartidos* para *Cambiar la configuración del adaptador*. Seleccionar las *Propiedades* de la tarjeta de red para asignar los parámetros correspondientes. Para que ambos equipos sean visibles entre sí, deben asignarse *IPs* coherentes con la máscara de subred.

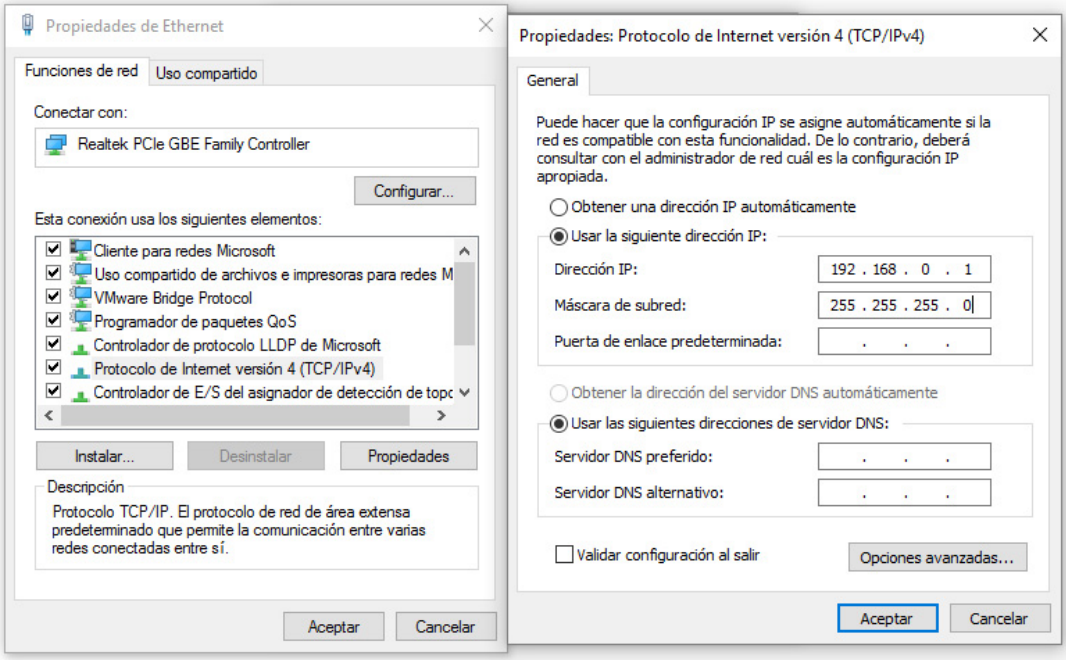


Imagen 52. Configuración de la tarjeta de red del equipo DSP.

De esta manera se asignará una *IPv4* fija a la tarjeta de red del equipo para que pueda ser reconocida dentro de la red local del sistema. Debe tenerse presente la *IP* asignada al equipo, ya que es un parámetro a configurar a posteriori en la aplicación.

8.1.4 Configuración de la aplicación

En la carpeta de la aplicación abrir el archivo *ComunicacionTurbina.exe.config*. Este archivo es un fichero de código *XAML,* que permite la asignación de ciertos valores a variables del programa de la aplicación sin necesidad de una compilación posterior. Dentro del archivo buscar el contenedor *<ComunicacionTurbina.Properties.Settings>* en el que se encuentran dichas variables, de las que debe comprobarse su valor:

- *BBDD_Local_ConnString*: Data Source=localhost\DSPTURBINA;Initial Catalog=Turbina;Trusted_Connection=Yes
- *ipServidorComandos*: 192.168.0.1
- *ipServidorEstados*: 192.168.0.2
- *iPuertoBaudRate*: 9600
- *iPuertoDataBits*: 8

- *iPuertoTCP*_Comandos: 33334
- *iPuertoTCP*_Estados: 33333
- *rutaLog*: C:\LogTurbina\logTurbina.txt
- *sCOMPuerto*: COM1
- *sNombreTablaTurbina*: tablaTurbina

8.1.5 Otras configuraciones

Debe crearse la carpeta y el archivo *.txt* para el log en la ruta definida. Para que la aplicación pueda escribir en dicho archivo debe eliminarse la opción de *Solo lectura* de la carpeta contenedora.

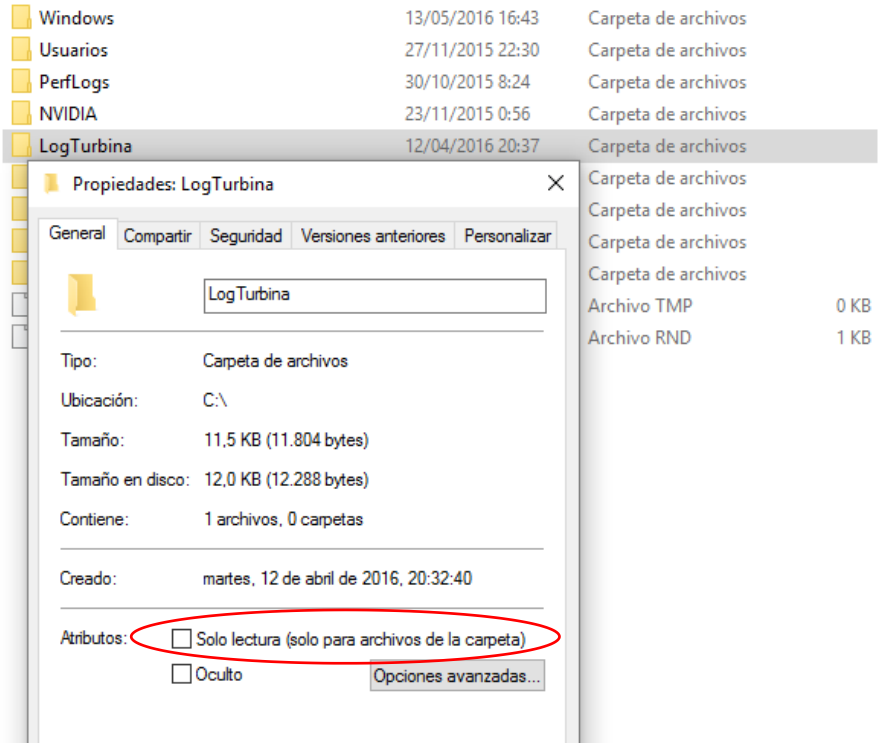


Imagen 53. Propiedades de la carpeta de archivo del log del DSP.

Finalmente, para permitir la integración total del equipo dentro de la red local, desactivar el *Firewall de Windows* en *Panel de control**Sistema y seguridad**Firewall de Windows.*

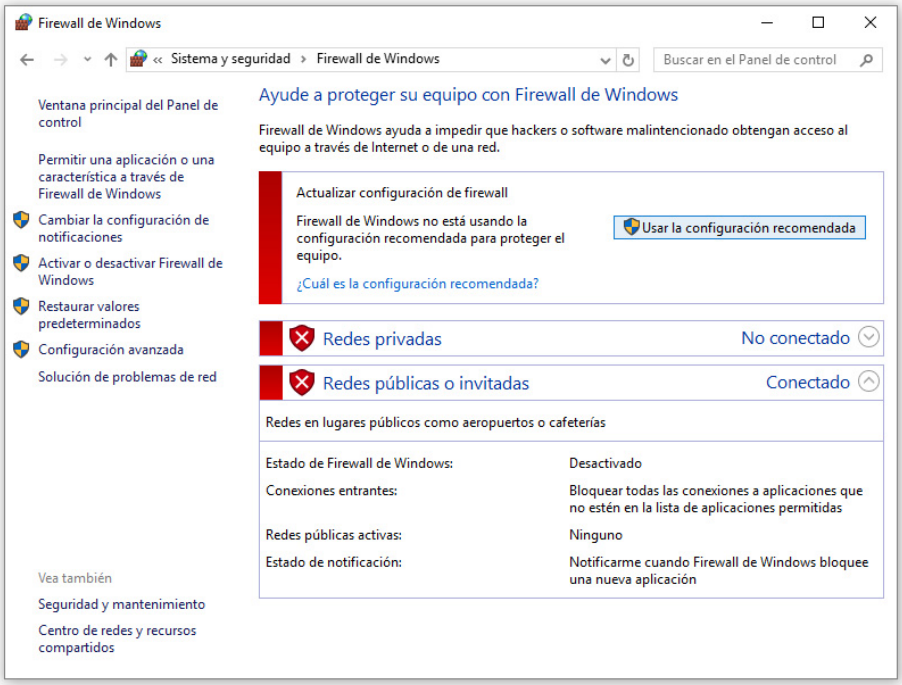


Imagen 54. Desactivación del Firewall de Windows.

8.2 Equipo HMI

8.2.1 Configuración del motor de bases de datos

1) Las configuraciones previas son idénticas a las realizadas para el equipo DSP.
2) Dentro del servidor configurado se crea la base de datos *Graficas* con las siguientes características.

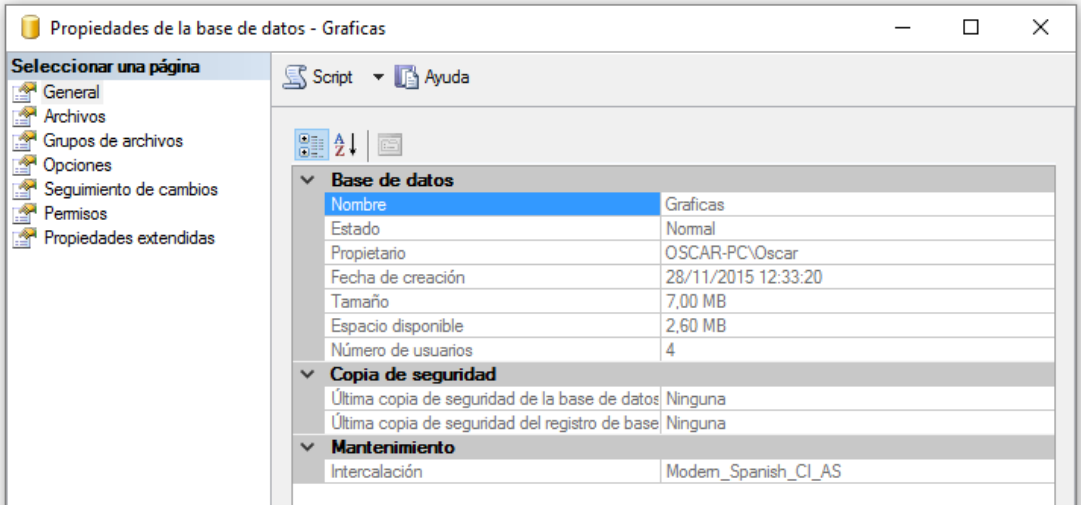


Imagen 55. Propiedades de la base de datos "Graficas" del HMI.

3) Finalmente se crea la tabla que almacenará los datos para mostrar las gráficas.

```
CREATE TABLE [dbo].[tablaGraficas](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [dtFecha] [datetime] NOT NULL,
    [velGenerador] [real] NULL,
    [presionCamara] [real] NULL,
    [posicionCilindro] [real] NULL,
    [potencia] [real] NULL,
    [vibraciones] [real] NULL,
    [corrienteIU] [real] NULL,
    [corrienteIV] [real] NULL,
    [tensionVDC] [real] NULL,
    [temperatura1] [real] NULL,
    [temperatura2] [real] NULL,
    [comandoPosValvula] [real] NULL,
    [refVelocidadGiroMax] [real] NULL,
    [bError1] [binary](50) NULL,
    [bError2] [binary](50) NULL,
    [bError3] [binary](50) NULL,
    [bEstado] [binary](50) NULL,
 CONSTRAINT [PK_tablaGraficas] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
) ON [PRIMARY]

GO
```

Imagen 56. Diseño de la tablaGraficas.

Así pues, el esquema del servidor de bases de datos en el equipo DSP será el siguiente:

- Servidor: *nombreEQUIPO\nombreSERVIDOR*
  - Base de datos: *Graficas*
    - Tabla: *tablaGraficas*

Es necesario habilitar el protocolo de comunicaciones *TCP/IP* para que el servidor de bases de datos permita el acceso desde otros equipos de la red. Para ello debe procederse de la misma manera descrita para el equipo DSP.

8.2.2 Configuración de la tarjeta de red

En el menú *Redes e Internet* del *Panel de control* de Windows, iniciar el *Centro de redes y recursos compartidos* para *Cambiar la configuración del adaptador*. Seleccionar las *Propiedades* de la tarjeta de red para asignar los parámetros correspondientes. Para que ambos equipos sean visibles entre sí, deben asignarse *IPs* coherentes con la máscara de subred.

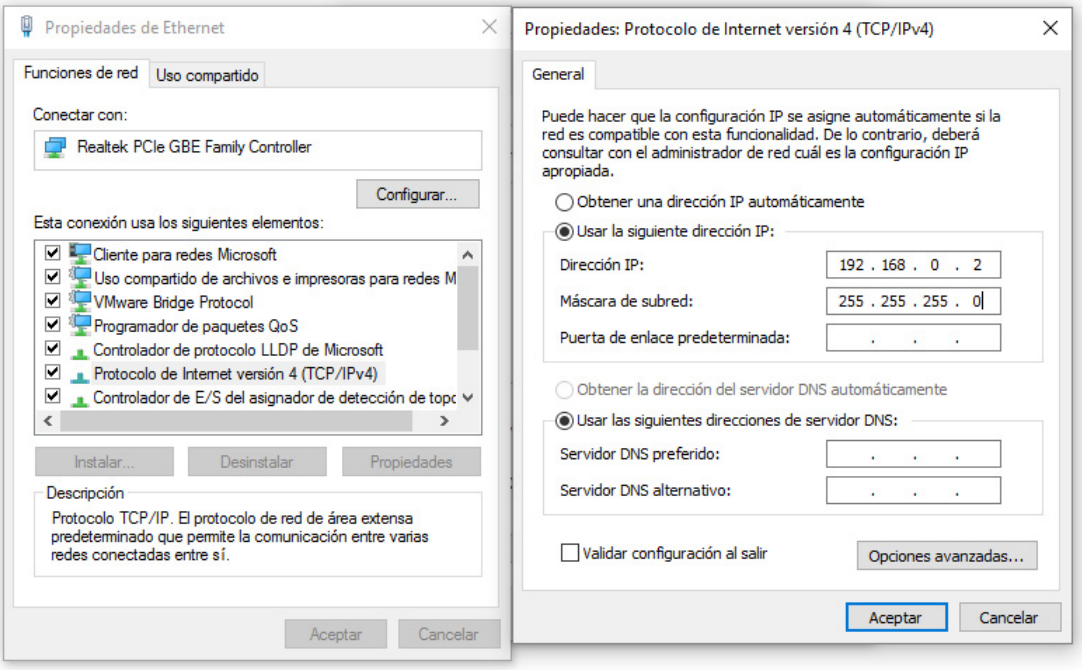


Imagen 57. Configuración de la tarjeta de red del equipo HMI.

De esta manera se asignará una *IPv4* fija a la tarjeta de red del equipo para que pueda ser reconocida dentro de la red local del sistema. Debe tenerse presente la *IP* asignada al equipo, ya que es un parámetro a configurar a posteriori en la aplicación.

8.2.3 Configuración de la aplicación

Se procederá de forma análoga a la descrita para el equipo DSP. En la carpeta de la aplicación abrir el archivo *ClienteGraficas.exe.config*. Dentro del mismo, buscar el contenedor *<ClienteGraficas.Properties.Settings>* en el que se encuentran las variables de las que se debe comprobar su valor:

- *cadenaLocal*: Data Source=localhost\HMITURBINA;Initial Catalog=Graficas;Trusted_Connection=Yes
- *BBDD_Remota_ConnString*: Data Source=PCTURBINA\DSPTURBINA;Initial Catalog=Turbina;Trusted_Connection=Yes
- *ipServidorComandos*: 192.168.0.1
- *ipServidorEstados*: 192.168.0.2
- *iPuertoTCP*_Comandos: 33334
- *iPuertoTCP*_Estados: 33333
- *rutaLogEstados*: C:\LogsHMI\logEstados.txt
- *rutaLogAlarmas*: C:\LogsHMI\logAlarmas.txt
- *sNombreTablaRemota*: tablaTurbina
- *sNombreTablaLocal*: tablaGraficas

8.2.4 Otras configuraciones

De forma análoga a lo realizado en el equipo DSP, debe crearse la carpeta y el archivo *.txt* para los logs en las rutas definidas. De igual manera, para que la aplicación pueda escribir en dichos archivos, debe eliminarse la opción de *Solo lectura* de las carpetas contenedoras.

Finalmente, para permitir la integración total del equipo dentro de la red local, desactivar el *Firewall de Windows* en *Panel de control**Sistema y seguridad**Firewall de Windows.*

9 MANUAL DE USUARIO

En esta sección se describen los programas del sistema de cara al usuario final. Se muestran las ventanas existentes en ambos programas y se explican los elementos de cada una de ellas. Con ello se pretende que el usuario conozca las opciones que ofrece el sistema y que pueden dar respuesta a sus necesidades, al tiempo que sepa reaccionar ante posibles fallos o errores en el sistema.

Como ya se ha dicho, el sistema está formado por dos aplicaciones diferentes: la referida a la adquisición y procesamiento de los datos procedentes de la turbina (*DSP*) y la referida a la supervisión de dichos datos (*HMI*).

Para facilitar el desarrollo de las aplicaciones que constituyen el sistema, se ha creado un simulador que permite crear las tramas que generaría la turbina para, posteriormente, enviarlas al *DSP.* Este simulador no forma parte del sistema, pero se incluye en este manual con el fin de que un posible usuario final (técnico de mantenimiento) pueda utilizarla para encontrar errores en el sistema.

9.1 Aplicación DSP

Es la parte del sistema encargada de la adquisición y procesamiento de los datos procedentes de la turbina. Está constituida por una única ventana meramente informativa de cara al usuario, ya que no permite ninguna interacción.

9.1.1 Log de eventos de la turbina

Es la ventana principal y única del *DSP.* En ella se recogen eventos y errores relacionados con la adquisición y el procesamiento de los datos procedentes de la turbina, datos importantes (alarmas y estado de la turbina) recibidos, eventos y errores en las comunicaciones y acciones procedentes del HMI.

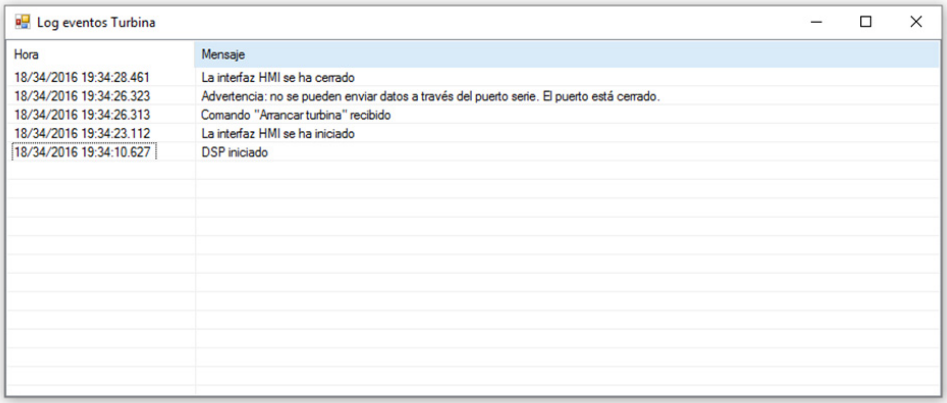


Imagen 58. Aplicación DSP. Ventana principal.

Como respaldo del log, cada mensaje recibido se guarda en un fichero *.txt* en la ruta *C:\LogTurbina* con el nombre *logTurbina.txt.* De esta manera quedan registrados todos los eventos y mensajes recibidos por el *DSP,* aunque se reinicie el programa. No obstante, es importante destacar que no se registrarán eventos ni mensajes estando la aplicación cerrada.

Así pues, los tipos de mensajes recogidos en el log se pueden resumir en los siguientes:

- DSP iniciado.
- DSP cerrado.
- HMI iniciado.
- HMI cerrado.
- Estado de la conexión con el servidor de estados.
- Errores y estados enviados al HMI.
- Errores y estados no enviados al HMI.
- Estado de la conexión con la base de datos local.
- Estado y errores del puerto serie.
- Trama errónea recibida de la turbina.
- Tipo de comando y valor recibido del HMI.
- Errores y estado de la turbina modificados.
- Recibiendo datos de la turbina vía puerto serie.

9.2 Aplicación HMI

Constituye la parte de supervisión del sistema. Permite visualizar los datos obtenidos de la turbina en forma de curvas de tendencia, así como las alarmas y el estado de funcionamiento de la misma.

9.2.1 Menú principal

Al iniciar la aplicación se abre la ventana que muestra el menú principal. Es la ventana principal de la aplicación y permanece siempre abierta. Consiste en una serie de botones que abren nuevas ventanas en las que se pueden realizar diferentes tareas de configuración, control y supervisión de la turbina.

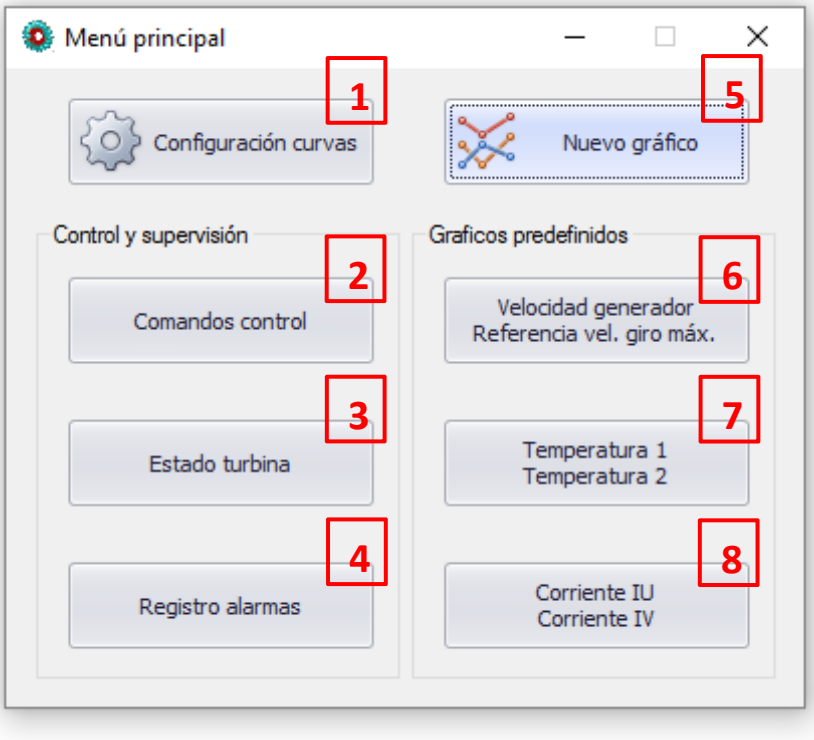


Imagen 59. Aplicación HMI. Menú principal.

Según la referencia de números de la imagen, cada uno de los botones del menú principal tiene las siguientes funciones:

1. **Configuración curvas**: abre la ventana de configuración de curvas.
2. **Comandos control**: abre la ventana que permite enviar comandos a la turbina.
3. **Estado turbina**: abre la ventana que registra y muestra el estado actual de la turbina.
4. **Registro alarmas**: abre la ventana que registra y muestra las alarmas de la turbina.
5. **Nuevo gráfico**: abre la ventana que permite configurar y mostrar las curvas deseadas en un rango de fechas determinado.
6. **Velocidad generador vs. Referencia velocidad giro máxima**: abre la ventana que permite configurar y mostrar las curvas de velocidad actual del generador y de la referencia máxima de velocidad de giro en un rango de fechas determinado.
7. **Temperatura 1 vs. Temperatura 2**: abre la ventana que permite configurar y mostrar las curvas de Temperatura 1 y Temperatura 2 en un rango de fechas determinado.
8. **Corriente IU vs. Corriente IV**: abre la ventana que permite configurar y mostrar las curvas de en un rango de fechas determinado.

Por tratarse de la ventana principal, durante el funcionamiento de la aplicación permanece siempre abierta, en segundo plano o minimizada, y al cerrarla se cierran el resto de ventanas abiertas de la aplicación. Mientras la aplicación se encuentra abierta, se realizan copias de respaldo de la base de datos del *DSP* a la local del *HMI* cada cierto intervalo de tiempo. Además, cada vez que se abre la aplicación se comprueban las diferencias entre ambas bases de datos y se realiza una copia de respaldo de los datos que no se encuentran en el equipo del *HMI.* Por ello, si la aplicación permanece cerrada durante un tiempo, cuando se vuelva a abrir, los datos que no estén en la base de datos del *HMI* se copiarán automáticamente.

9.2.2 Configuración de curvas y ejes

En esta ventana se pueden modificar los colores y los rangos de representación de todas las curvas de datos. Estos valores se almacenan en el equipo local, por lo que la aplicación puede cerrarse sin perder la información.

Si se tiene abierta esta ventana no se podrá manipular el resto de ventanas de la aplicación.

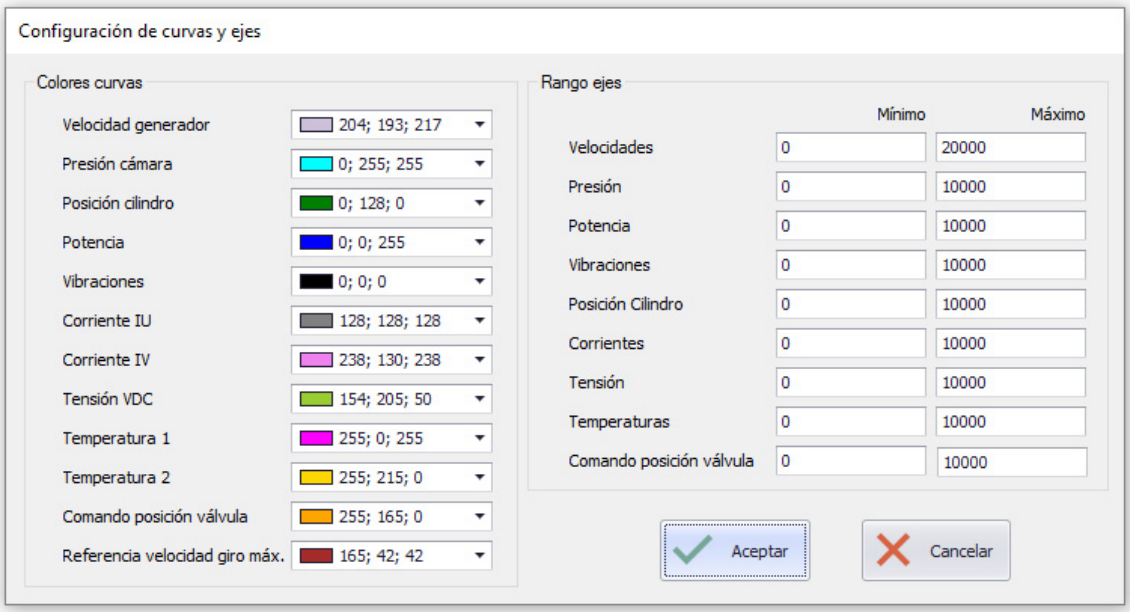


Imagen 60. Aplicación HMI. Ventana de configuración de curvas y ejes.

Para cerrar la ventana y aceptar los cambios realizados, pulsar *Aceptar*. Para cerrarla descartando los cambios, pulsar *Cancelar*.

9.2.3 Envío de comandos de control

Desde esta ventana (Imagen 61) se pueden enviar comandos para el control de la turbina. Mediante ellos se pueden realizar las siguientes acciones:

1. Arrancar la turbina.
2. Parar la turbina.

3. Limitar la velocidad máxima de giro.
4. Establecer el funcionamiento normal.
5. Abrir la válvula.
6. Cerrar la válvula.
7. Posicionar el vástago de la válvula en una posición determinada.

Para activar un comando basta con pulsar sobre el botón correspondiente. Si se tratara de un comando de consignación bastaría con asignar el valor deseado en el campo de texto correspondiente y posteriormente pulsar *Enviar consigna*. El valor de consigna puede aumentarse o disminuirse haciendo uso de las flechas, o introducirse directamente escribiendo el valor en el campo de texto.

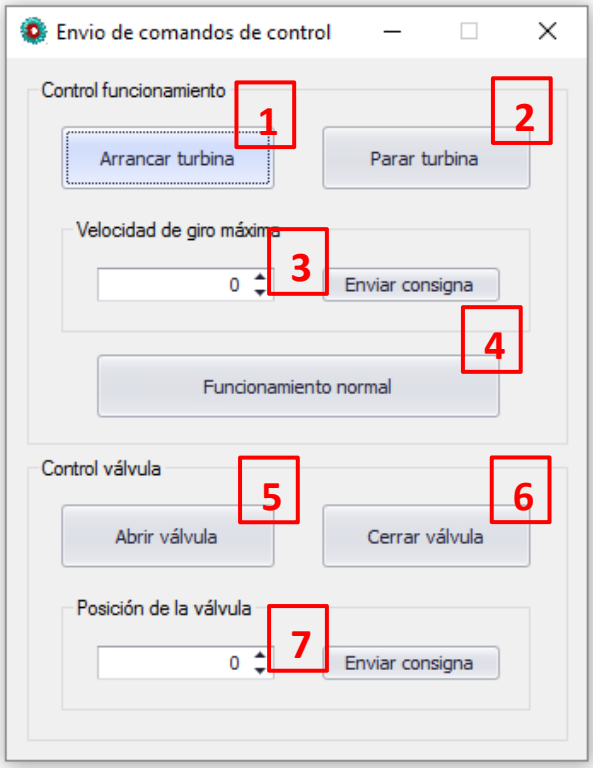


Imagen 61. Aplicación HMI. Ventana de envío de comandos.

Esta ventana puede permanecer abierta o minimizada mientras se manipulan otras. Si, una vez abierta, se pulsa el botón en el menú para volver a abrirla, la ventana pasará al frente y se restablecerá su posición original.

9.2.4 Estado general de la turbina

Esta es una ventana de alerta o de consulta. En caso de producirse algún cambio en el estado de la turbina, esta ventana se abre y se muestra instantáneamente. En caso de ya estar abierta, al pulsarse el botón correspondiente del menú, o al detectarse algún cambio, pasará al frente permitiendo chequear el nuevo estado en el que se encuentra la turbina.

En la parte izquierda se muestra el valor actual de los parámetros referentes al estado de la turbina:

- **Velocidad limitada**: Sí/No.
- **Posición intermedia de la válvula**: Activado/Desactivado.
- **Estado de funcionamiento**: Parado/Funcionamiento normal.
- **Estado alarma**: En alarma/Funcionamiento OK.
- **Estado del motor del cilindro**: ON/OFF.
- **Posición del cilindro**: Extendido/Recogido.
- **Electroimán**: ON/OFF.
- **Estado de la turbina**: Girando/Parada.

En el caso de que la turbina esté en alarma se mostrará un icono de alerta ( ) al lado del texto.

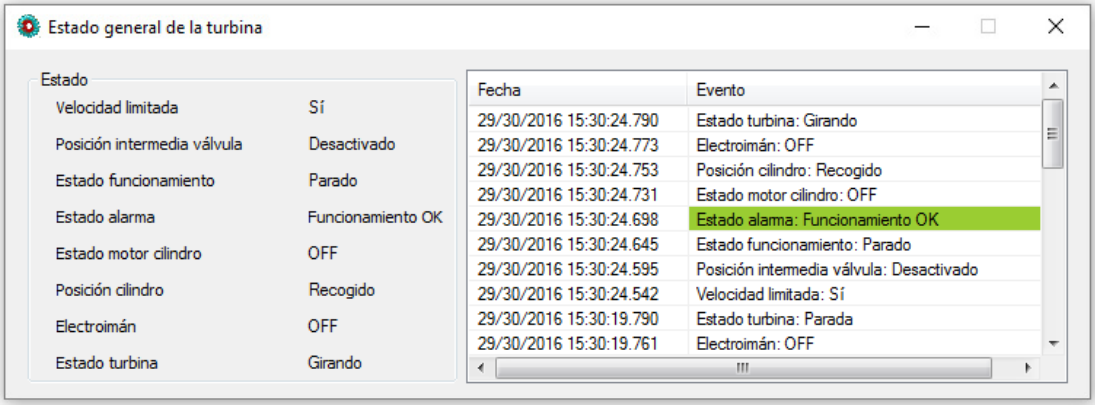


Imagen 62. Aplicación HMI. Ventana de visualización del estado de la turbina.

En la parte derecha existe un pequeño registro de los estados de la turbina cada vez que se produce un cambio en alguno de ellos, siendo los más recientes los que aparecen en la parte superior. De esta manera se pueden conocer cambios pasados en el estado de la turbina. Existe un respaldo de este registro en forma de archivo *.txt* que se encuentra en la ruta *C:\LogsHMI* con el nombre *logEstados.txt.*

9.2.5 Registro de alarmas

Esta ventana es muy similar a la descrita en el apartado anterior. Es una ventana de alerta o de consulta. En caso de detectarse alguna alarma en la turbina, se abre y se muestra instantáneamente. En caso de ya estar abierta, al pulsarse el botón correspondiente del menú, o al detectarse alguna alarma nueva, pasará al frente.

En la parte izquierda de la ventana se muestra el valor actual de las alarmas clasificadas por tipos:

- **Errores de en los sensores**:
  - Lectura anómala del sensor IU.
  - Lectura anómala del sensor IV.
  - Lectura anómala del sensor IR.
  - Lectura anómala del sensor IS.
  - Lectura anómala del sensor VDC.
  - Lectura anómala del sensor de vibraciones.
  - Lectura anómala del sensor de posición del cilindro.
  - Lectura anómala del sensor de presión.
  - Lectura anómala en el sensor de temperatura 1.
  - Lectura anómala en el sensor de temperatura 2.
- **Errores en las fases**:
  - Sobrecorriente en la fase U.
  - Sobrecorriente en la fase V.
  - Sobrecorriente en la fase R.
  - Sobrecorriente en la fase S.
- **Errores de carácter general**:
  - Vibración excesiva.
  - Presión excesiva en la cámara.
  - Velocidad de la turbina excesiva.
  - Sobretensión en el bus DC.
  - Fallo en el encoder.
  - Exceso de temperatura 1.
  - Exceso de temperatura 2.

En caso de que se haya detectado algún error se mostrará un icono indicativo (✕) en la línea correspondiente. En caso contrario se mostrará el icono de estado correcto (✓).

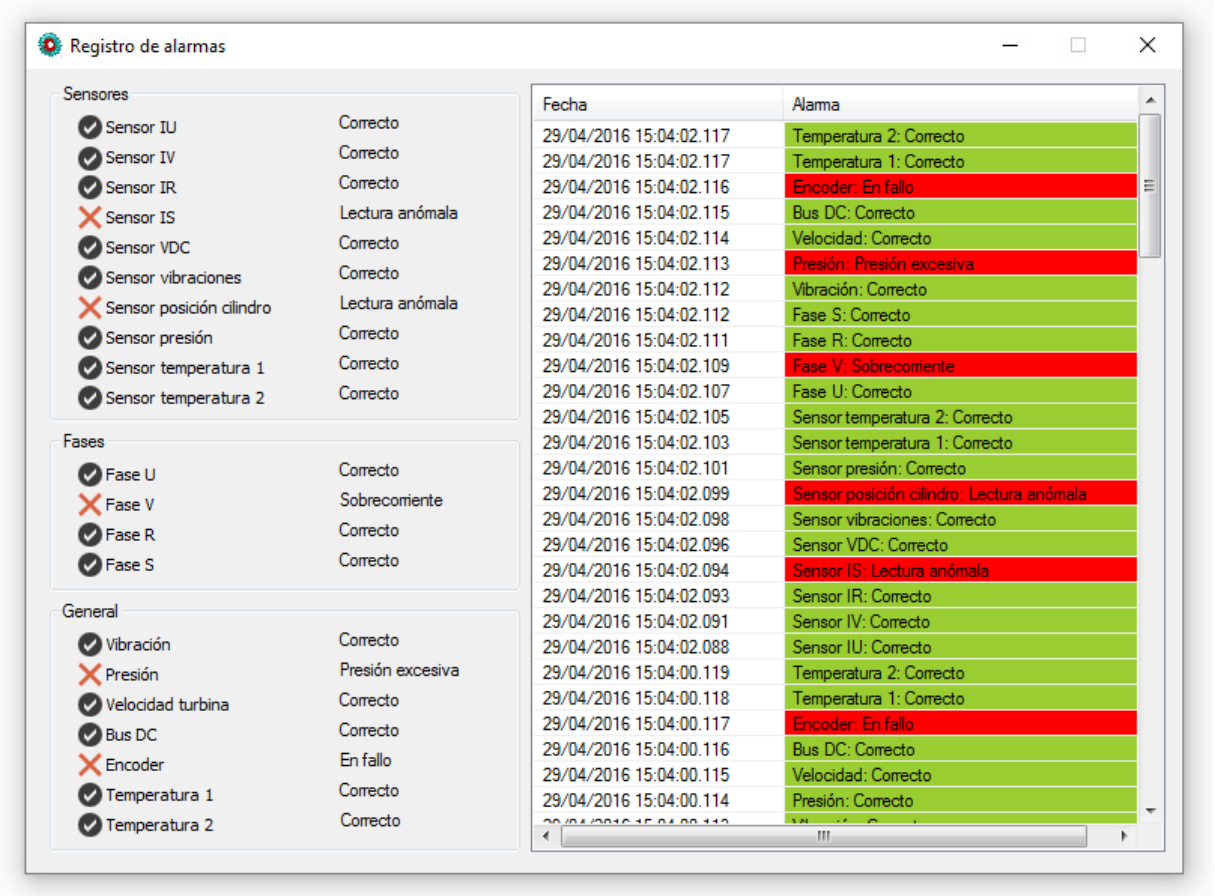


Imagen 63. Aplicación HMI. Ventana de visualización del registro de alarmas.

En la parte derecha existe un pequeño registro de errores de la turbina. Cada vez que se produce un cambio, bien sea de estado correcto a incorrecto o a la inversa, se muestra en el registro, siendo los cambios más recientes los mostrados en la parte superior. De esta manera se pueden conocer errores pasados en producidos en la turbina. Existe un respaldo de este registro en forma de archivo *.txt* que se encuentra en la ruta *C:\LogsHMI* con el nombre *logAlarmas.txt.*

9.2.6 Nuevo gráfico

Es la ventana principal de monitorización de datos. Desde ella se pueden representar las curvas de los parámetros disponibles en un rango de fechas deseado. Se pueden abrir tantas ventanas de este tipo como se deseen, en las cuales se pueden representar gráficos totalmente distintos.

Se puede representar más de una curva en el mismo gráfico. Los colores y los rangos de los ejes se configuran en la ventana de configuración de curvas y ejes. Cada unidad de ingeniería tiene su eje asociado, pero el color de cada curva es independiente.

La parte superior de la pantalla constituye el área de representación y la banda inferior el área de configuración.

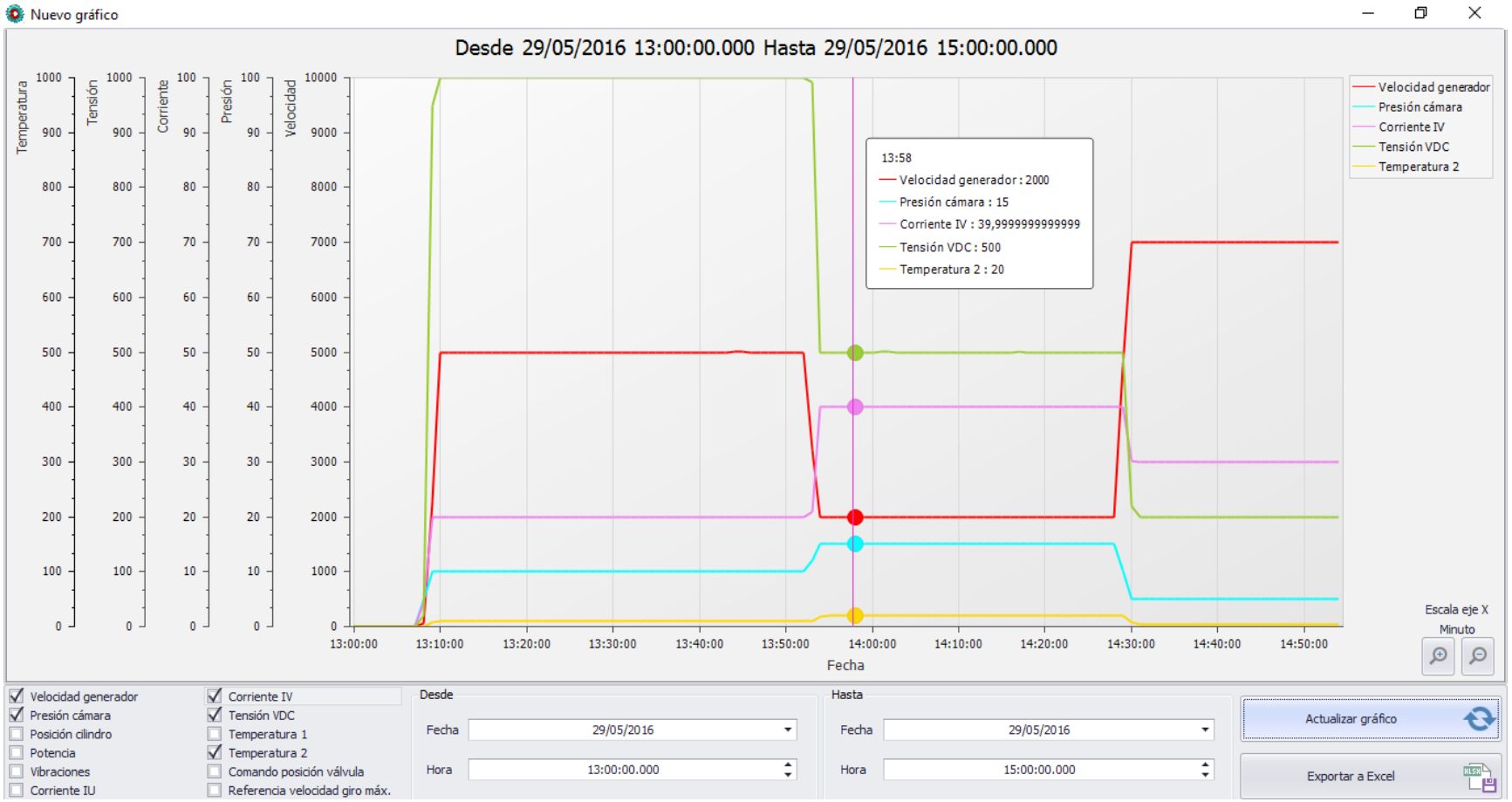


Imagen 64. Aplicación HMI. Ventana de generación de gráficos.

El proceso para representar una o más curvas en el área de representación es el siguiente:

1. Marcar las curvas deseadas en la lista.
2. Seleccionar el rango de fechas y horas del que se desean mostrar los datos. La especificación de las horas permite una precisión de milésimas de segundo. Las fechas se pueden escribir directamente o bien se pueden seleccionar de los calendarios desplegables.

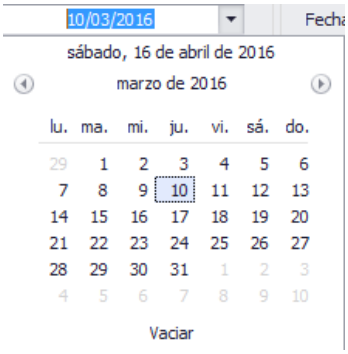


Imagen 65. Aplicación HMI. Selección de rango de fechas para mostrar un gráfico.

3. Pulsar el botón *Actualizar gráfico*.

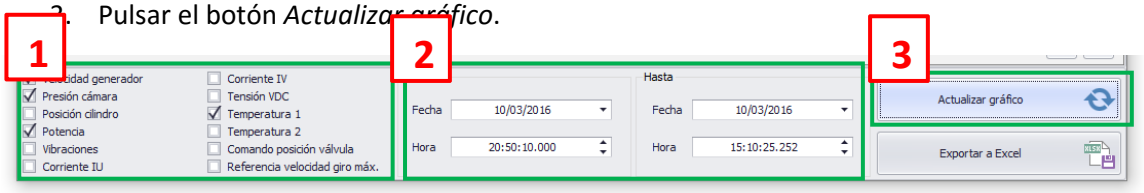


Imagen 66. Aplicación HMI. Configuración de un nuevo gráfico.

La ventana toma el nombre de las curvas seleccionadas, y la leyenda y los ejes se actualizan. La leyenda muestra los colores asociados a cada curva y los ejes toman los rangos especificados en la ventana de configuración.

Si dentro del rango de tiempos seleccionado existe un periodo sin datos, se mostrará una interpolación de los datos en dicho período.

Cada vez que se desee mostrar datos en otros rangos de tiempo o se modifiquen las curvas a mostrar, se debe pulsar el botón *Actualizar gráfico* para que la aplicación realice la consulta pertinente a la base de datos.

Mediante los botones de zoom se puede aumentar la precisión de representación del eje de tiempos.

El botón *Exportar a Excel* permite transferir los datos mostrados en el área de representación a un fichero *.xlsx.* Cada una de las curvas seleccionadas y los datos recogidos para ese rango de tiempos se exportarán de forma individual a hojas distintas del fichero.

Una vez pulsado el botón se mostrará una ventana informativa de qué curvas se van a exportar y en qué rango de tiempos. A continuación, debe seleccionarse la ruta en la que se desea guardar el fichero *.xlsx*.

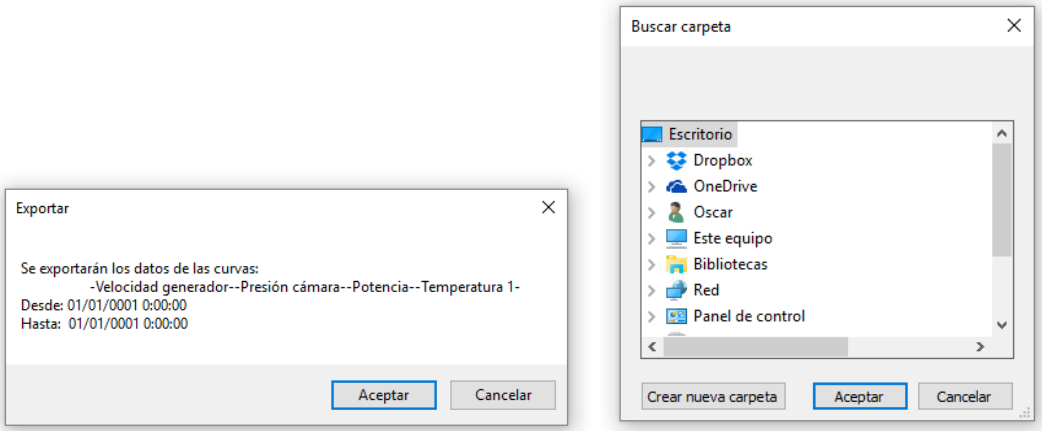


Imagen 67. Aplicación HMI. Exportación de datos a Excel.

Una vez seleccionada, comienza la exportación de los datos. Este proceso se realiza en segundo plano, por lo que mientras tanto pueden realizarse otras tareas con la aplicación.

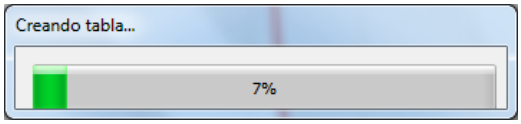


Imagen 68. Barra de progreso de la exportación.

9.2.7 Gráficos predeterminados

Estas ventanas crean gráficos de grupos de curvas predeterminados en un rango de tiempos seleccionable. Son ventanas exactamente iguales a la descrita en el apartado anterior, con sus mismas características, solo que están restringidas a unas curvas predefinidas.

De cada gráfico predeterminado se pueden abrir tantas ventanas como se deseen, incluso con rangos de tiempos diferentes. Además, se han definido tres tipos diferentes de gráficos predeterminados.

1. **Velocidad del generador vs. Referencia velocidad giro máxima.**

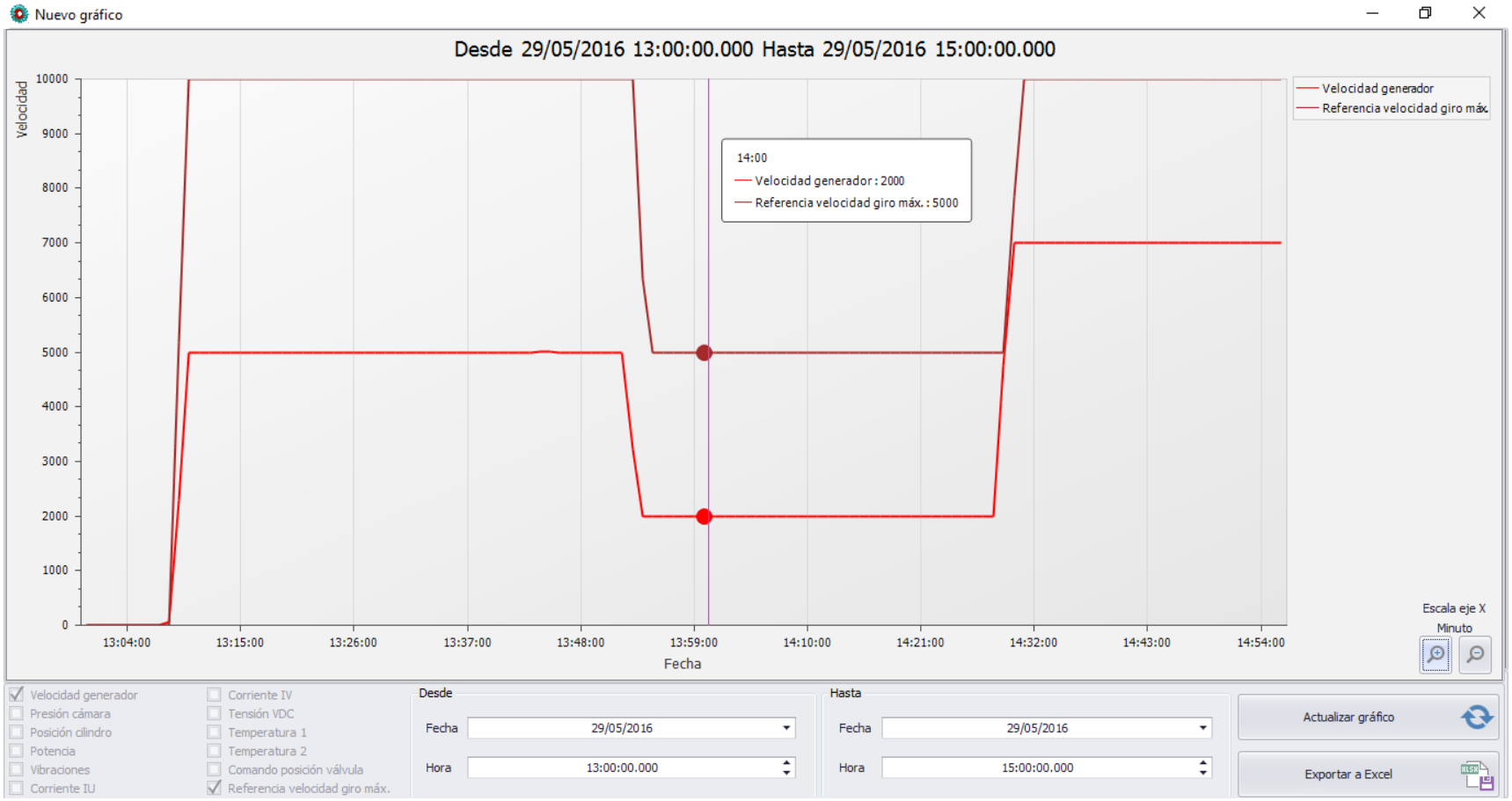


Imagen 69. Aplicación HMI. Gráfico predeterminado Vel. generador vs. Ref. velocidad giro máx.

2. Temperatura 1 vs. Temperatura 2.

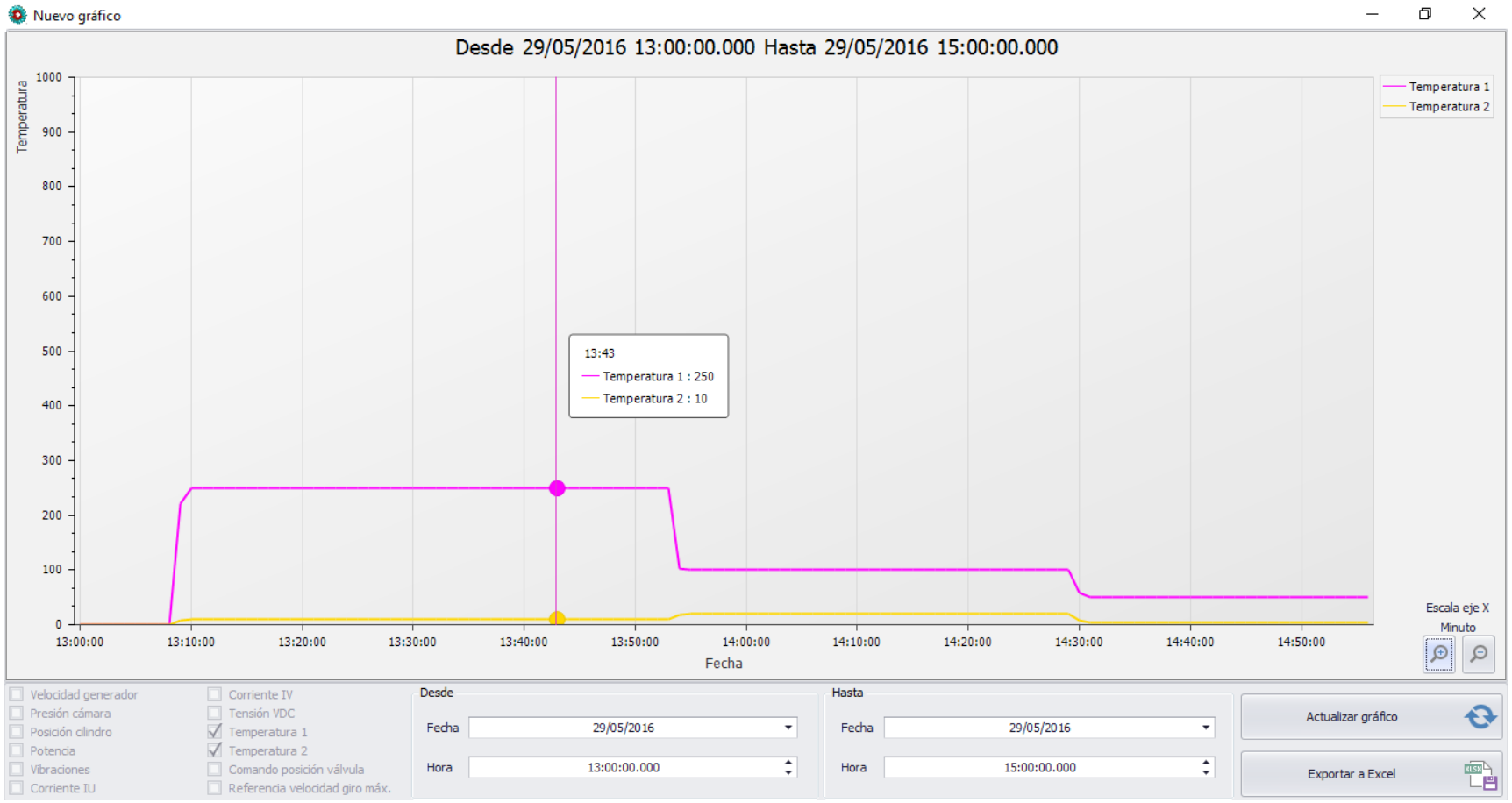


Imagen 70. Aplicación HMI. Gráfico predeterminado Temperatura 1 vs. Temperatura 2.

3. Corriente IU vs. Corriente IV.

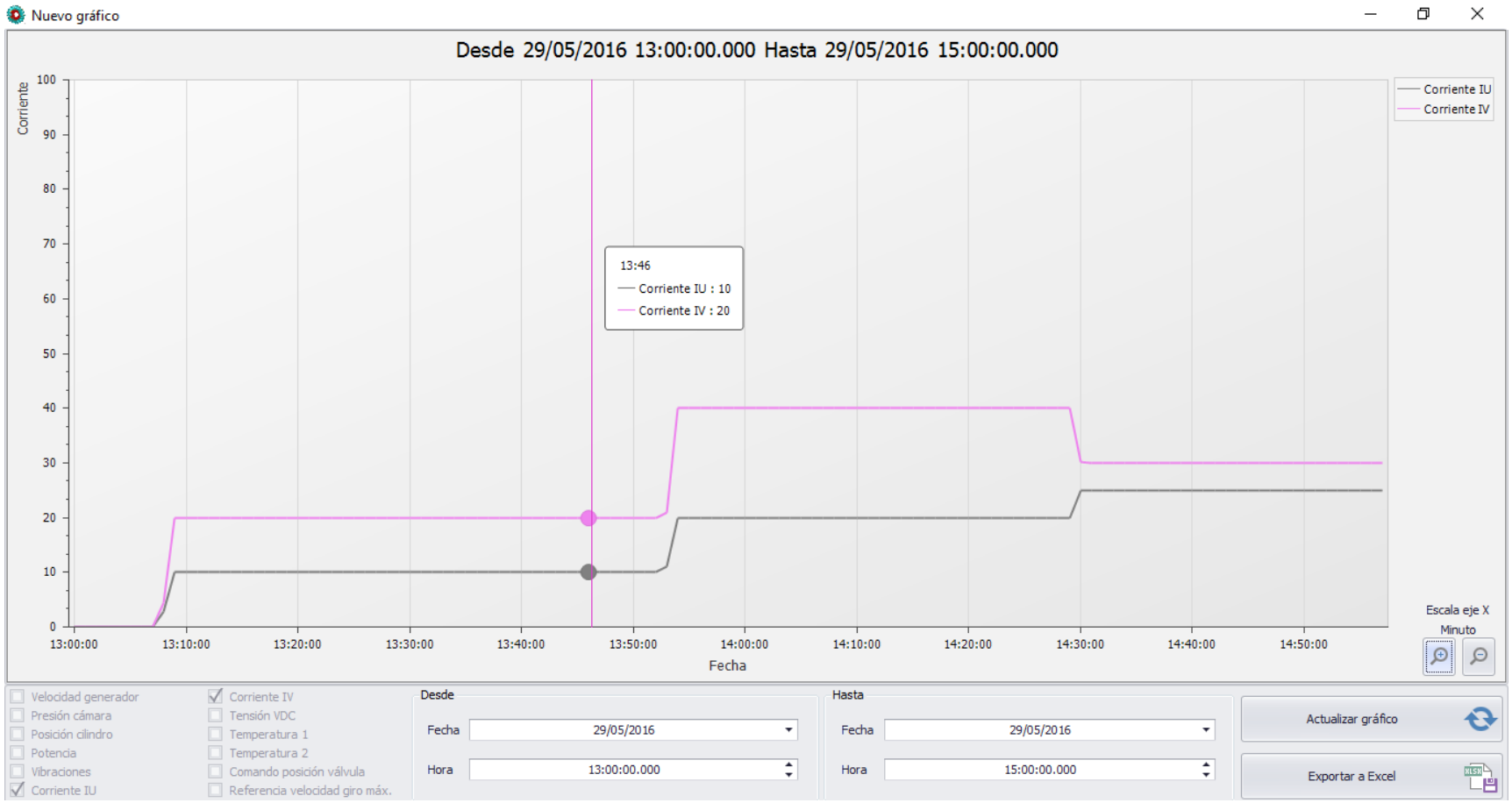


Imagen 71. Aplicación HMI. Gráfico predeterminado Corriente IU vs. Corriente IV.

Mediante la definición de estos gráficos predeterminados se pretende dar agilidad en la visualización de datos referentes a propiedades importantes en la turbina.

Como se ven en las imágenes de cada uno de ellos, poseen las mismas opciones que la ventana de *Nuevo gráfico* (zoom, selección de fechas, botones de actualización y exportación a Excel), pero sin la posibilidad de modificar las curvas a mostrar.

9.2.8 Mensajes de error

Problema	Chequeo/solución
No se puede conectar con el cliente de estados	Las direcciones IP de los equipos no coinciden con las configuradas.
Comando no enviado. Imposible conectar al servidor de comandos	
No se puede cerrar el servidor de estados	El servidor de estados ya está cerrado.
Error al establecer conexión con las bases de datos	El servicio SQL Server no está iniciado. La base de datos o la tabla no existen. No se tienen permisos suficientes para el acceso a la base de datos.
Error al consultar la base de datos local	
Error al consultar la base de datos remota	
Error al actualizar el gráfico con datos de la BBDD	
Error al abrir el registro	El registro especificado no existe. No se tienen los permisos suficientes para abrir/leer/escribir el registro.
No se ha podido actualizar el registro	
Error al leer del registro	
Error en la exportación a Excel	No se tienen los permisos suficientes para la creación de archivos. Los datos cargados son incorrectos.

Tabla 10. Aplicación HMI. Mensajes de error.

9.3 Simulador de la turbina

Este programa no forma parte del sistema, ya que no interviene ni en la parte del *DSP* ni del *HMI*. Se trata meramente de un simulador utilizado para la realización de pruebas en ausencia de la turbina real. Como la turbina envía sus datos al *DSP* en forma de trama de datos, mediante el simulador se pueden recrear todas las posibles tramas que se podrían enviar al *DSP*.

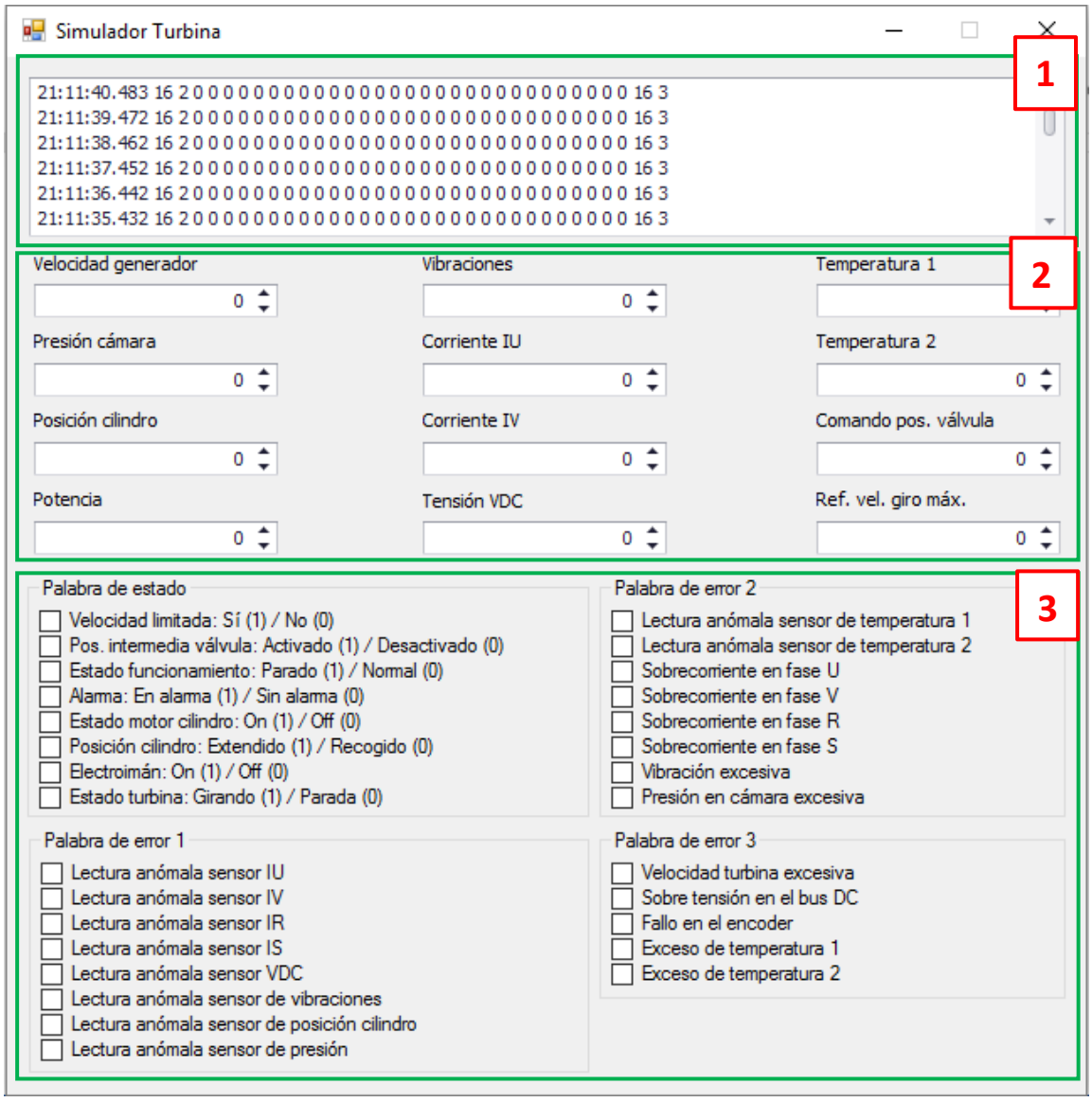


Imagen 72. Simulador Turbina. Ventana principal.

La ventana está dividida en tres zonas:

1. **Zona de visualización de las tramas enviadas**: muestra las tramas enviadas al *DSP* y el instante de tiempo en el que se enviaron. De esta manera puede comprobarse si la trama enviada es la correcta y si el período de envío es también el correcto.
2. **Zona de parámetros numéricos**: permite dar valor a parámetros de la turbina tales como velocidades, corrientes, temperaturas… Para dar un valor numérico a un campo basta con introducirlo a través del teclado o hacer uso de los botones anexos para aumentar o disminuir el valor.

3. **Zona de bits de estado y errores:** permite activar o desactivar bits de las palabras de estado y error de la turbina. Para activar o desactivar un bit basta con pulsar sobre la casilla correspondiente.

Problema	Chequeo/solución
No se puede abrir el puerto serie	Los parámetros configurados para el puerto no son los correctos. El puerto configurado no existe. El puerto ya está abierto/cerrado y no se puede abrir/cerrar.
No se puede leer el puerto serie	
No se puede escribir el puerto serie	

Tabla 11. Simulador Turbina. Mensajes de error.

10 DISCUSIÓN, CONCLUSIONES Y POSIBLES AMPLIACIONES

El objeto de este apartado es realizar un análisis crítico sobre el trabajo realizado en el proyecto. Se pretende analizar las ventajas y desventajas de los métodos utilizados a lo largo del mismo con el fin de reflexionar sobre la idoneidad de los mismos.

10.1 Discusión del método

Tal y como queda presente a lo largo de este documento, el desarrollo del proyecto gira en torno al modelado y programación de los programas que componen el sistema. Para la parte del modelado se ha utilizado el lenguaje UML y para la programación se ha optado por utilizar el paradigma de la programación orientada a objetos (POO).

Si bien es conocida la utilidad del UML a la hora de definir el alcance y las especificaciones de un proyecto de desarrollo de software, también es sabido que su utilidad radica en su versatilidad. Es decir, el desarrollador utilizará las opciones que brinda el UML como mejor le convenga para modelar el sistema. En otras palabras, el obtener un modelado UML totalmente completo en el que se incluyan todos y cada uno de los detalles del sistema no implica necesariamente que se trate de un modelado mejor. En ocasiones es conveniente omitir ciertos elementos del modelado que sí formarán parte del sistema final. De esta forma, todos los elementos que forman parte del interface de usuario podrían haberse omitido del modelado del presente sistema, obteniéndose unos diagramas más sencillos y claros de cara a la comprensión.

El modelado de cualquier sistema debe realizarse de forma previa a la programación del mismo. No obstante, esto no evita que se realicen modificaciones sobre el modelado inicial, ya que es muy difícil tener en cuenta todos los detalles del sistema desde un principio. Además, las particularidades del lenguaje de programación que se va a utilizar pueden invitar a modificar ciertas partes del modelado. Por ello, de cara a optimizar el método, parece útil realizar un modelado inicial completo para, posteriormente, realizar las sucesivas iteraciones del mismo de forma paralela a la implementación del sistema final.

Por otro lado, la implementación del sistema valiéndose del paradigma de programación orientada a objetos proporciona una versatilidad a la que los lenguajes de texto estructurado no llegan. El nivel de abstracción y encapsulación que se alcanza con el concepto de clase es tal que la reutilización de código para proyectos similares, y la depuración, pueden ser realmente eficientes. No obstante, debe establecerse un límite razonable para no caer en un nivel de abstracción excesivo en el que el código esté tan fragmentado que resulte prácticamente incomprensible.

10.2 Conclusiones

El uso del UML como herramienta principal para modelar todo el sistema ha constituido una gran ventaja a la hora de definir los elementos que forman parte de los programas de control. Como esta metodología de modelado está especialmente orientada a la programación con objetos, el utilizarla junto con un lenguaje de programación de este tipo ha resultado una combinación muy eficiente de cara al desarrollo completo del proyecto.

Se han encontrado especialmente ventajosas las propiedades de la programación orientada a objetos. La encapsulación del código en forma de clases que se asemejen a elementos reales, que realicen unas tareas propias y que tengan sus propiedades específicas ha facilitado en gran medida la creación de los programas de control del sistema. No obstante, el llegar a un equilibrio entre dichas propiedades y el desarrollo de un código no demasiado abstracto ha constituido un gran reto, ya que es el propio desarrollado el que establece el límite.

Por otro lado, se ha comprobado que el UML sirve para modelar un sistema sea cual sea el lenguaje que se va a utilizar para la programación. También se ha observado la gran cantidad de diagramas que se realizan, iteración tras iteración, a lo largo de todo el proceso de modelado. No obstante, ha quedado más que demostrada la utilidad de esta metodología a la hora de definir y concretar las tareas que debe realizar el sistema, así como de todas las relaciones entre los elementos del mismo.

10.3 Posibles ampliaciones

El presente proyecto no está exento de posibles ampliaciones y mejoras. A continuación, se proponen algunas de ellas:

- Implementación de un control de usuarios en la aplicación *HMI*, de tal manera que se mantenga un registro de qué usuario ha realizado qué acciones y cuándo las ha realizado.
- Visualización en tiempo real de la variación de los parámetros de la turbina. Establecimiento de un tiempo de actualización óptimo para evitar la sobrecarga de la red.
- Comprobación de la escalabilidad del sistema para permitir más estaciones de control y monitorización.
- Ampliación de la seguridad de la red local del sistema o integración del mismo en la red local de toda la planta.
- Habilitación de un túnel VPN para acceso remoto al sistema desde fuera de la red local.

11 BIBLIOGRAFÍA

- Stevens, P. (2002). *Utilización de UML en Ingeniería del Software con Objetos y Componentes*. Madrid: Pearson educación, S.A.
- Riordan, R. (2002). *Microsoft ADO.NET Step by Step*. Washington: Microsoft Press.
- Sharp, J. (2005). *Microsoft Visual C# 2005 Step by Step*. Washington: Microsoft Press.
- Ritcher, J. (2006). *CLR via C#, Second Edition*. Washington: Microsoft Press.