UNIVERSITAT JAUME I

Grado en Ingeniería Informática

Trabajo Final de Grado

Aplicación de control de un sistema de clasificación por visión artificial

Autor:
Rafael Pallarés Palos

Tutor académico:
Sergio Barrachina Mir

Fecha de lectura: 28 de Junio de 2018
Curso académico 2017/2018

A mi familia, por ser siempre el apoyo necesario

A mis compañeros de estudios, por formar un grupo de amigos maravilloso

A Euroelettra Ingeniería, por todas las facilidades

A Esther, por ser la compañera ideal

Resumen

El proyecto presentado en este documento consiste en una aplicación de control y gestión de una máquina de clasificación de envases plásticos, cuyo objetivo es, mediante visión artificial y pesaje, descartar o aceptar los envases plásticos recién fabricados. El proyecto se enfoca en el diseño e implementación de la aplicación de control de esta máquina, que permitirá controlar su estado y funciones, así como el diseño e implementación de la arquitectura de red utilizada para intercomunicar los distintos elementos del sistema. Este proyecto ha sido implementado en el lenguaje de programación *C#* utilizando el entorno de desarrollo *Visual Studio*, apoyándose en el gestor de bases de datos *SQL Server* y en el protocolo de comunicación con autómatas programables *Omron* llamado *FINS*, sobre los cuales se ha tenido que realizar un aprendizaje preliminar. El resultado final se encuentra en producción en la fabrica del cliente que solicitó el proyecto.

Palabras clave

Adquisición de datos y control, Aplicación de control, Control industrial, PLC

Keywords

Data acquisition and control, Control application, Industrial control, PLC

Índice general

Índice de figuras

Capítulo 1

Introducción

Índice del capítulo

Esta memoria presenta el trabajo de desarrollo y pruebas de la aplicación de control de una máquina de control de calidad de envases plásticos, realizada en la empresa *Euroelettra ingeniería S.L*, así como de la configuración de red para teleasistencia. El proyecto realizado permite a los operarios y técnicos de la planta de producción controlar los parámetros, estados y estadísticas de la máquina de control de calidad.

Durante el primer capítulo, se desarrolla la introducción y presentación del proyecto, así como el contexto técnico del proyecto. En el segundo capítulo se describirá de forma detallada el proyecto, describiendo también los elementos de hardware que forman la máquina completa, la estructura de red utilizada y demás aspectos técnicos. En el tercer capítulo se expone la planificación del proyecto, siguiendo la metodología del diagrama de *Gantt*, así como la planificación del proyecto y las estimaciones de los recursos. En el cuarto capítulo, se hablará sobre el análisis y diseño del sistema, explicando mediante diagramas de clases y casos de uso, la fase de diseño de la aplicación de control. En el quinto capítulo se comentarán los diferentes detalles de implementación y las pruebas llevadas a cabo tanto en el taller como en la fábrica del cliente. Para finalizar, en el sexto capítulo, se exponen las conclusiones extraídas del proyecto.

1.1. Contexto y motivación del proyecto

La industria en los últimos años ha visto multiplicada su producción y demanda de forma exponencial. Es por esto que cada día es más importante para las empresas dedicadas a esta rama, la automatización y mecanización de sus tareas, para así evitar posibles peligros para el personal y aumentar su productividad. Por tanto es fundamental la existencia de empresas e iniciativas que dediquen su actividad a desarrollar, producir y mantener soluciones de automatización de maquinaria y de procesos industriales.

Euroelettra Ingeniería S.L, con sede en Vila-Real, Castelló, se dedica desde su nacimiento al desarrollo y mantenimiento de sistemas integrales de automatización.

En este contexto, uno de los clientes de la empresa requirió un sistema de control de calidad para los envases plásticos a la salida de la extrusora encargada de su fabricación, que controlase errores en la fabricación como manchas, deformaciones y variaciones de peso.

Además de la aplicación de control y los demás elementos informáticos en los que se centra este documento, el equipo entregado al cliente consta de diferentes brazos articulados, sensores de peso y un sistema de visión artificial, que puestos en conjunto cumplen con la tarea encomendada.

1.2. Objetivos del proyecto

El objetivo principal de este proyecto es el de desarrollar una aplicación de control que permita a los operarios y técnicos de la planta, gestionar los siguientes aspectos de la máquina de clasificación:

- Control del estado de la máquina y sus diferentes modos de funcionamiento.
- Visualización y modificación de los parámetros del sistema.
- Acceso al software de control de la visión artificial.
- Visualización de las estadísticas de producción.
- Gestión del estado del sistema de teleasistencia.

Capítulo 2

Descripción del proyecto

Índice del capítulo

En este capítulo se detalla el contexto del proyecto, así como una pequeña presentación de la empresa donde se ha realizado el trabajo, la descripción del proyecto y las tecnologías utilizadas, así como una pequeña puesta en contexto de las tecnologías explicadas.

2.1. Euroelettra Ingeniería S.L

Euroelettra Ingeniería, S.L, cuyo logo se muestra en la figura 2.1, ofrece soluciones completas en los procesos de automatización industrial que comienza con la identificación de la necesidad del cliente y el diseño del proyecto, y se completan con el montaje en las instalaciones y la puesta en funcionamiento de los sistemas y maquinarias de supervisión, gestión y control de procesos industriales o de sistemas de pesaje. La empresa se hace cargo de todo el proceso y entrega los proyectos llave en mano, lo que supone un grado de comodidad y rapidez importante para los clientes, ya que no tienen que contactar con diferentes proveedores para completar la instalación.

Figura 2.1: Logo Euroelettra

La empresa está certificada según la norma *ISO 9001* para la gestión interna de calidad, del año 2008. Dicha normativa certifica que la empresa cumple con los requisitos de gestión de

calidad interna [1]. El procedimiento para asegurar la gestión de calidad interna seguido es el siguiente:

1. Propuesta del sistema:
   a) Presentación y definición de necesidades por parte el cliente.
   b) Recogida de datos, planos, especificaciones técnicas, etc.
   c) Diseño del proyecto y su valoración económica.
   d) Presentación del diseño y soluciones, junto con la propuesta económica al cliente. Si el cliente acepta, el proyecto pasa a la fase de desarrollo.
2. Desarrollo o ejecución del proyecto y su control:
   a) Distribución de la documentación.
   b) Realización de los planos y esquemas eléctricos.
   c) Planificación de compras y producción.
   d) Construcción de las máquinas, cuadros de control, la instalación y montaje desarrollo del software, instrumentación y sistemas de dosificación.
   e) Pruebas del sistema desarrollado en nuestras instalaciones con la presencia del cliente.
3. Puesta en marcha del proyecto en las instalaciones del cliente.
4. Validación del proyecto: una vez finalizada la puesta en marcha el proyecto se somete a la validación por parte del cliente.
5. Entrega de documentación y manuales.
6. Servicio de asistencia técnica (S.A.T) y mantenimiento del sistema desarrollado, tanto hardware, como software.

2.2. Entorno tecnológico

En este apartado se describen las diferentes tecnologías que se han utilizado para el desarrollo del nuevo sistema. Entre ellas encontramos lenguajes de programación en entornos *Windows* como *C#*, el sistema gestor de bases de datos *SQL Server Express* y el entorno de desarrollo *Visual Studio 2015*.

También se incluye una breve introducción al protocolo *FINS* [2] para la comunicación con autómatas *Omron*, pero puesto que para implementar la comunicación se emplea la biblioteca de comunicación propietaria de *Euroelettra Ingeniería*, no se entra en detalles específicos, ya que no serían de utilidad para este documento.

2.2.1. Desarrollo de aplicación de escritorio

La mayor parte del tiempo empleado en el proyecto se ha centrado en el desarrollo de la aplicación de control, por ello en este apartado se introducen el lenguaje de programación y el entorno de desarrollo empleados durante esta fase del proyecto.

C#

Dado que todos los proyectos modernos que implementa *Euroelettra Ingeniería* se desarrollan en el lenguaje de programación C#, este desarrollo ha seguido esa linea, por compatibilidad con las bibliotecas utilizadas, tanto de desarrollo propio como desarrolladas por terceras partes. Este es un lenguaje de programación orientado a objetos desarrollado por *Microsoft* como parte de la plataforma *.NET* [3]. Desarrollado como lenguaje de propósito general, con revisión estricta de tipos de datos, recolección de basura automática y que genera aplicaciones ligeras en cuanto a uso de memoria y procesador.

Este lenguaje de programación está basado en los ya populares lenguajes *C* y *C++* y es el usado en la gran parte del desarrollo de este proyecto.

Visual Studio 2015

Este es el *IDE* o entorno de desarrollo de *Microsoft*, se encuentra disponible en varias plataformas y se enfoca a gran cantidad de lenguajes de programación, tratando de ser universal, abarcando el desarrollo de aplicaciones para escritorio, web y móvil.

En este proyecto en concreto y dado que la gran mayoría de tecnologías y *frameworks* utilizados se basan en la plataforma *.NET* y en el entorno de *Windows*, elegir este IDE para el desarrollo, es la decisión adecuada ya que facilitará las pruebas, la depuración y la implantación.

2.2.2. Procesamiento por lotes

Dado que la máquina que describe este documento, en algunas ocasiones carecerá de dispositivos de entrada (como ratón y teclado), se añaden una serie de botones en el cuadro de mandos para que los operarios de la planta de producción sean capaces de controlar algunas de las funciones, como el apagado del PC o el inicio retrasado de la aplicación.

Batch

Las funciones auxiliares mencionadas, se han implementado en *Batch* [4] , el lenguaje de *scripting* para sistemas *DOS*. Este lenguaje permite ejecutar comandos por lotes que pueden incluir sentencias condicionales y bucles, sin necesidad de compilar el programa ni instalar ningún interprete o software adicional.

2.2.3. Sistema gestor de bases de datos

La aplicación de control requiere de una base de datos para realizar el almacenamiento de:

- Históricos de alarmas.
- Históricos de acceso de usuarios.
- Parámetros de configuración de la aplicación y de la comunicación con el autómata.
- Gestión de accesos de usuarios.

Es por todo ello que se implementa una base de datos con el sistema de gestión de bases de datos *SQL Server* de Microsoft, principalmente porque la empresa donde se ha desarrollado el proyecto implementa todos sus trabajos en este sistema. Esto proporciona una mayor flexibilidad a la hora de escalar el proyecto en un futuro o facilidad a la hora de que otro compañero de la empresa pueda realizar una asistencia o una modificación en ausencia del desarrollador principal del proyecto. *SQL Server* debe ejecutarse como proceso separado en el equipo en el que desee instalarse la aplicación desarrollada y requiere de claves de acceso, con posibilidad de segmentar los permisos, eliminando así problemas de seguridad relacionados con los usuarios finales de la aplicación.

La versión empleada durante el desarrollo del proyecto ha sido *SQL Server Express 2014* [5] dado que es la versión para la que la empresa en la que se ha desarrollado el trabajo ofrece soporte. *Euroelettra Ingeniería* trata siempre de permanecer en una versión estable y testada de las herramientas que utiliza en producción, es por eso que la versión 2017 de este sistema de gestión de bases de datos no se ha utilizado en este proyecto.

2.2.4. Teleasistencia

Una de las funcionalidades más demandadas por los clientes de *Euroelettra Ingeniería* es la teleasistencia, ya que esto permite solucionar problemas del día a día sin que los técnicos y desarrolladores deban desplazarse al lugar de instalación. En el caso de este sistema, la teleasistencia permitiría además de solucionar problemas en la parte de la aplicación de control descrita en este documento, abordar problemas y asistencia en el autómata y en la cámara de visión artificial, ya que como se verá en subsiguientes capítulos de este documento, los tres dispositivos se encuentran interconectados.

El sistema de teleasistencia además, requiere de un nivel de seguridad alto, para prevenir entradas no autorizadas o filtración de información sensible para el cliente y para la propia empresa.

Es por eso que la teleasistencia se implementa mediante una conexión VPN, en la que el equipo en el que se ejecuta la aplicación de control funciona como servidor y el equipo remoto de los técnicos de *Euroelettra Ingeneria* actúa como cliente, como se puede ver en la figura 2.2.

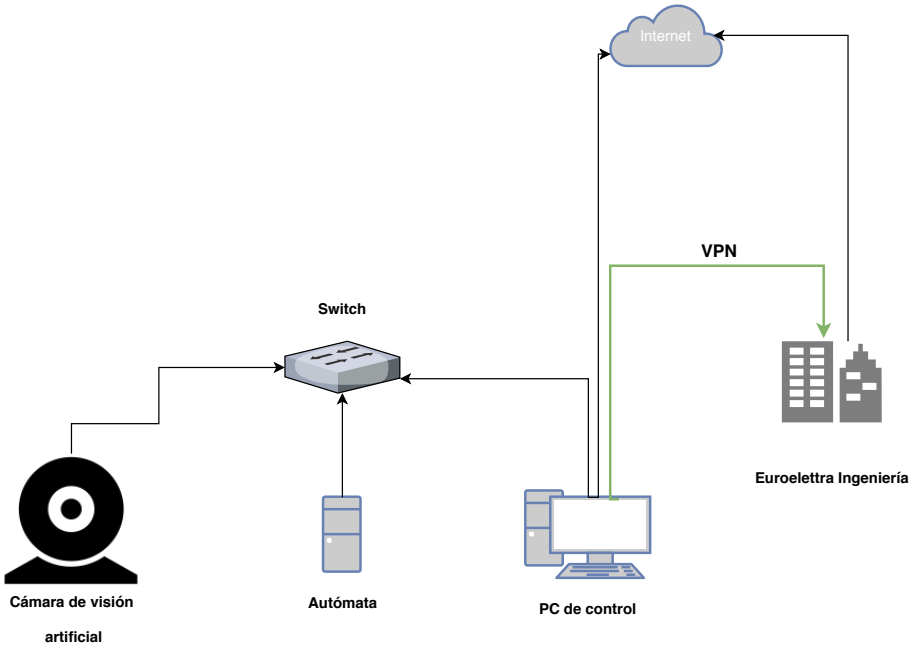


Figura 2.2: Esquema de teleasistencia

Capítulo 3

Planificación del proyecto

Índice del capítulo

En este capítulo se presenta la planificación que se ha aplicado para realizar este proyecto, empezando por la metodología empleada para su desarrollo. A continuación se describen las tareas en las que se ha dividido el proyecto y su estimación temporal. Se incluyen sendos diagramas de *Gantt* con el fin de ilustrar la planificación inicial y su seguimiento.

También se incluye un análisis del coste de los recursos del proyecto de desarrollo de la aplicación, el hardware y licencias de software incluidas en el PC de control y los elementos de electrónica de red. Se dejan fuera de este análisis los elementos del producto que no se abordan en este documento, como las partes de programación de autómatas, mecanización, pesaje y robótica.

3.1. Metodología

El proyecto, se ha desarrollado principalmente en un entorno *Windows*, sobre el que se ha implementado la aplicación y los elementos necesarios. Para llevar a cabo el trabajo de desarrollo, se ha empleado una metodología de planificación predictiva. La metodología de planificación predictiva realiza una descomposición del trabajo en tareas, según las necesidades del proyecto. Algunas de las tareas dependen de sus predecesoras para poder ser llevadas a cabo. Otras en cambio son independientes, con lo que estas últimas, se podrían haber llevado a cabo en un orden diferente o de forma paralela, sin interferir entre ellas.

Dado que el desarrollo del proyecto ha sido llevado a cabo por un solo programador, se han

ido abordando las tareas de forma secuencial, iniciando unas al terminar las anteriores. Esto es, se ha comenzado con una etapa de análisis, seguida del diseño y se ha terminado con la implementación y las pruebas.

3.2. Planificación

Tras varias reuniones mantenidas entre los técnicos de *Euroelettra Ingeniería* y el cliente, para definir los requisitos de la aplicación y sus necesidades, se planifica el trabajo. Dado que el proyecto contenía ciertas áreas en las que la empresa no era experta y requirieron más tiempo de investigación y desarrollo del esperado, las estimaciones temporales iniciales han variado considerablemente con la ejecución final del proyecto.

En la figura 3.1 se muestra un diagrama de estructura de descomposición del trabajo (*EDT*), desglosando las actividades del proyecto.

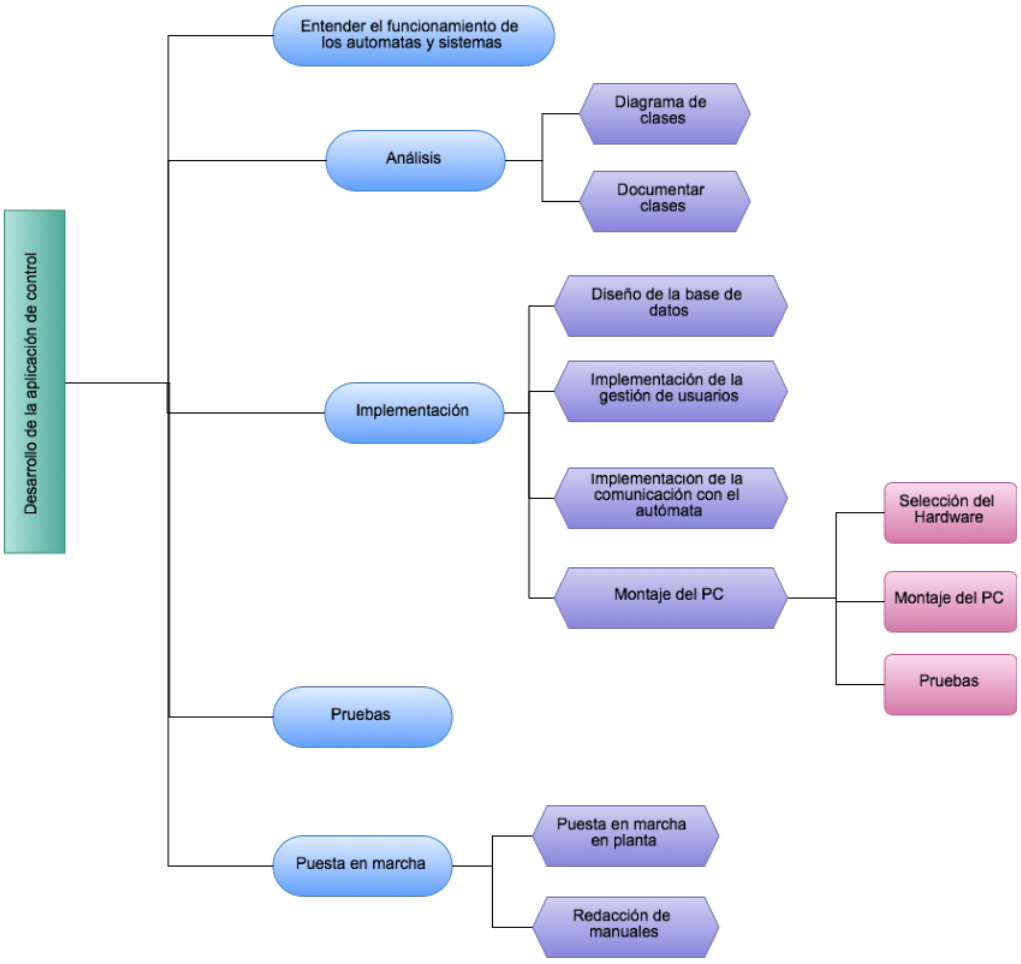


Figura 3.1: Diagrama de desglose de las actividades del proyecto

3.2.1. Diagrama de Gantt

Con el objeto de estimar los costes temporales de las actividades de desarrollo del proyecto, se ha tenido en cuenta que el trabajador no estaba dedicado por completo a este proyecto, ya que otras averías o proyectos paralelos iban apareciendo durante su desarrollo. La estimación mostrada se elaboró utilizando la técnica de analogía de proyectos, apoyándose en reuniones con el cliente y compañeros experimentados en las herramientas y lenguajes de programación.

Finalmente el desglose temporal de las tareas a realizar fue el siguiente:

- Entender el funcionamiento de los autómatas y diferentes sistemas (*3 días*)
- Definición de requisitos (*2 días*)
  - Diagrama de casos de uso y documentar requisitos de datos (*1 día*)
  - Definir requisitos tecnológicos (*1 día*)
- Análisis (*2 días*)
  - Diagrama de clases (*1 día*)
  - Documentar clases (*1 día*)
- Desarrollo del proyecto (*20 días*)
  - Diseño y creación de la base de datos (*3 días*)
  - Desarrollo de la aplicación (*9 días*)
  - Implementación de la comunicación con el autómata (*4 días*)
  - Implementación de la gestión de usuarios (*1 día*)
  - Diseño del PC (*3 días*)
    - Selección del hardware (*1 día*)
    - Montaje y puesta en marcha del PC (*1 día*)
    - Pruebas del PC (*1 día*)
- Pruebas (*3 días*)
- Puesta en marcha (*4 días*)
  - Puesta en marcha en planta (*2 días*)
  - Redacción de los manuales (*2 días*)
  - **Entrega final**

El diagrama de Gantt en el que se refleja la distribución temporal de estas tareas, siguiendo la planificación inicialmente estimada, se puede ver en la figura 3.2.

3.3. Estimación de recursos y costes del proyecto

En el cuadro 3.1 se desglosan los costes de los recursos del proyecto descrito en este documento, incluyendo las licencias de software en los casos en los que se han requerido. En el cuadro 3.2, se desglosan los costes de hardware del PC sobre el que se ejecuta el producto final, en la tabla. Cabe destacar que dado que el documento trata simplemente del desarrollo de la aplicación de control y el diseño e implementación de la red interna, tan solo los elementos relacionados con dichas fases del proyecto, se tendrán en cuenta en este apartado. Quedan por tanto fuera los elementos mecánicos, brazos robóticos, autómatas, cables y demás elementos de hardware que conforman el producto final.

Recurso	Coste unitario	Horas	Coste
Programador	20€/Hora	160	3.200€
Diseñador	30€/Hora	40	1.200€
Analista	45€/Hora	16	720€
Ordenador personal de desarrollo	850€	–	28,33€
Total			5.148,33€

Cuadro 3.1: Recursos y costes correspondientes al proyecto

Recurso	Coste	Unidades	Coste total
Licencia Windows 7	100€	1	100€
Licencia SQL Server Express	0€	1	0€
Caja PC	80€	1	80€
Placa base	70€	1	70€
Memoria RAM 4GB	44€	2	88€
Procesador	85€	1	85€
Disco duro SSD	120€	1	120€
Fuente de alimentación	45€	1	45€
Switch 4 puertos ethernet	25€	1	25€
Monitor empotrable	320€	1	320€
Total			934€

Cuadro 3.2: Recursos de hardware y software correspondientes al proyecto

Remarcar que el ordenador personal de desarrollo continuará siendo utilizado en otros proyectos en la empresa y que por tanto se aplica el coeficiente de amortización que según la agencia tributaria española [6], es de hasta seis años. Dado que este equipo trabajará en ocasiones en condiciones de polvo o humedad en las visitas a las fábricas o clientes, se aplica la amortización en cinco años. Es por eso que el coste real del equipo fue de 850€ pero en la tabla se refleja el coste asociado exclusivamente a este proyecto teniendo en cuenta un periodo de amortización de cinco años y una duración aproximada del proyecto de dos meses.

También quedan fuera los recursos humanos invertidos en la programación del autómata que utiliza el proyecto, ya que esta tarea ha sido llevada a cabo por el departamento de programación de autómatas de *Eurolettra Ingeniería*. Por otro lado, el sistema de visión artificial proporcionado por una empresa externa también queda fuera del alcance del proyecto y por consiguiente, de este desglose de costes.

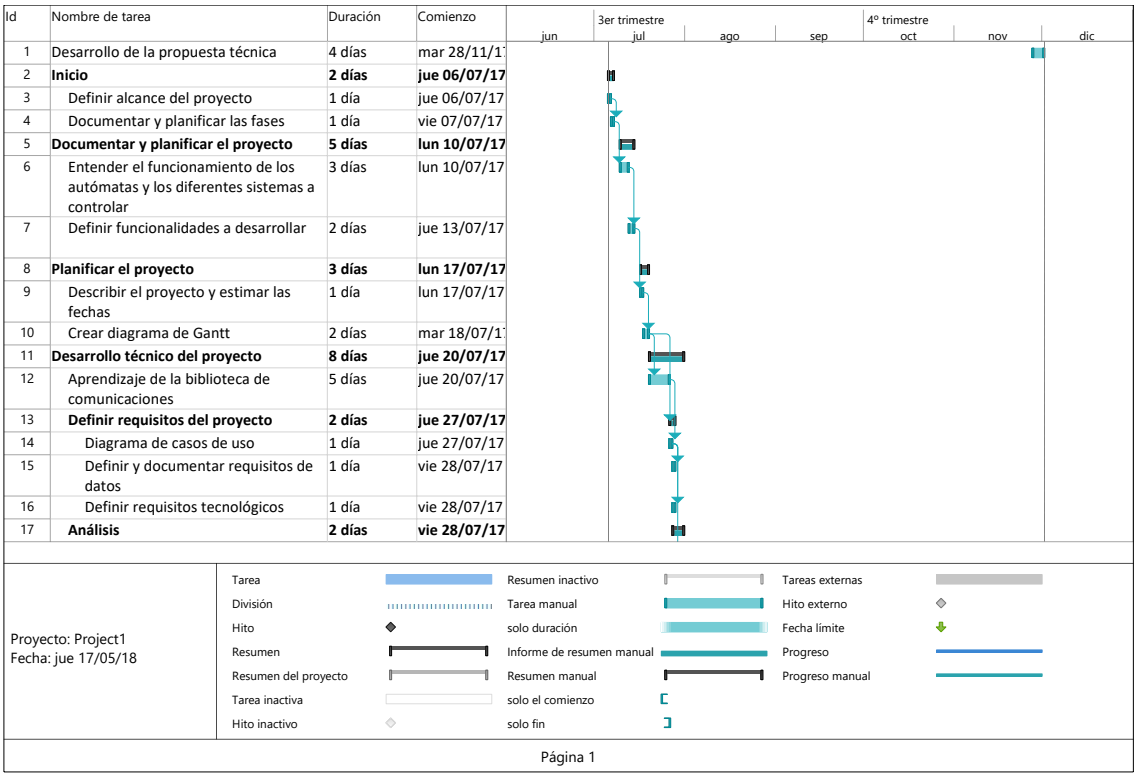
3.4. Seguimiento de la planificación del proyecto

Durante el desarrollo del proyecto, se ha ido realizando un seguimiento para verificar que los plazos y duraciones estipuladas en los apartados anteriores se cumplían. La estimación inicial expuesta en el apartado 3.2.1 se basa en las estimaciones a partir de proyectos anteriores del mismo calibre, pero dada la falta de experiencia del desarrollador de la aplicación con algunos de los elementos del proyecto, tales como los autómatas y el desarrollo en *C#*, han hecho que algunas de las etapas se retrasen.

El apartado de *Implementación de la comunicación con el autómata* se ha alargado cuatro días, debido a los motivos expuestos anteriormente. También la implementación de la gestión de usuarios se alargó en un día, dado el desconocimiento del desarrollador de la biblioteca utilizada para tal fin.

Por último, en el apartado del desarrollo, se denota una demora de un día adicional en el diseño de la base de datos, para adecuarla al funcionamiento con las bibliotecas de comunicación explicadas en el capítulo 5.1.

Por otra parte, las pruebas en el taller también sufrieron una amplia dilatación en el tiempo, debiéndose esto en gran parte a la inexperiencia con la comunicación con los autómatas. El diagrama de Gantt correspondiente a la ejecución del proyecto se muestra en la figura 3.3.



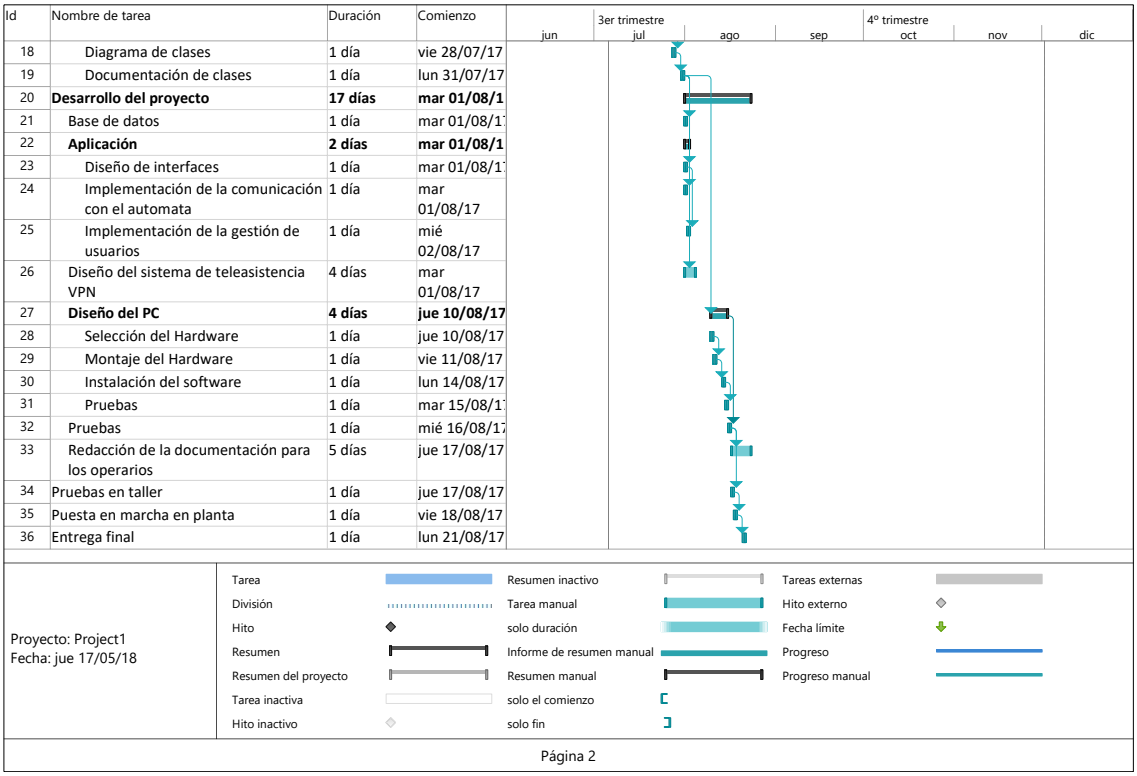
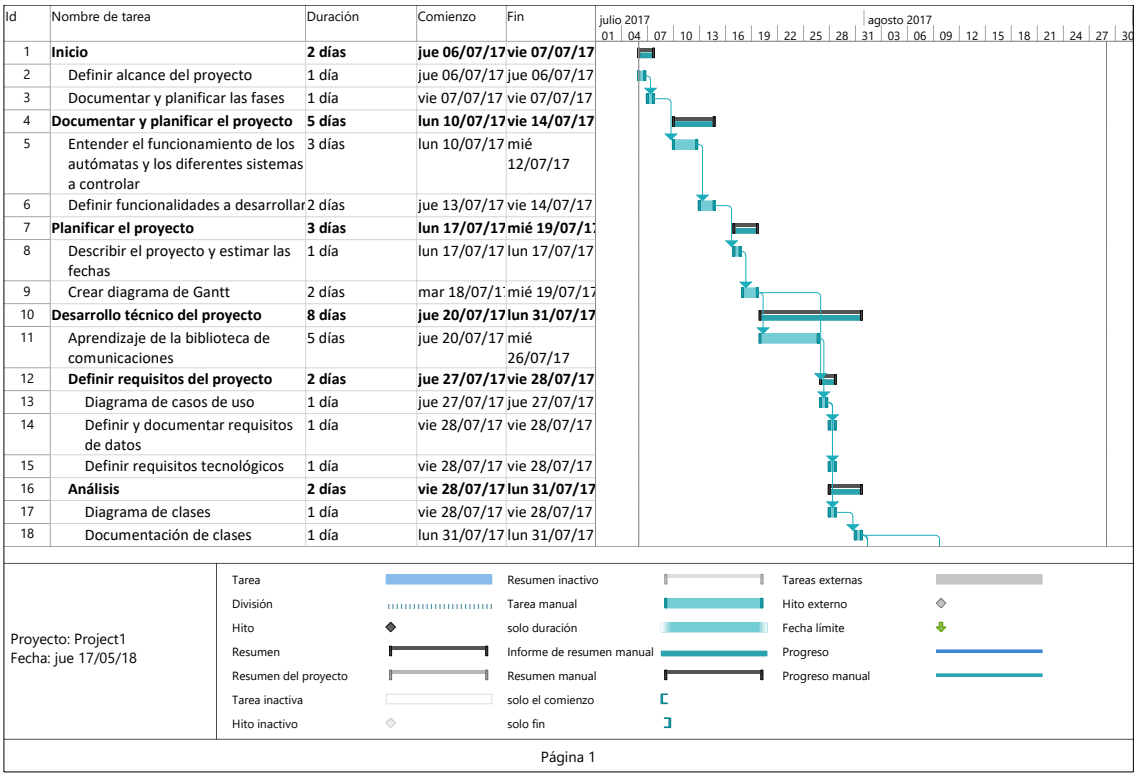
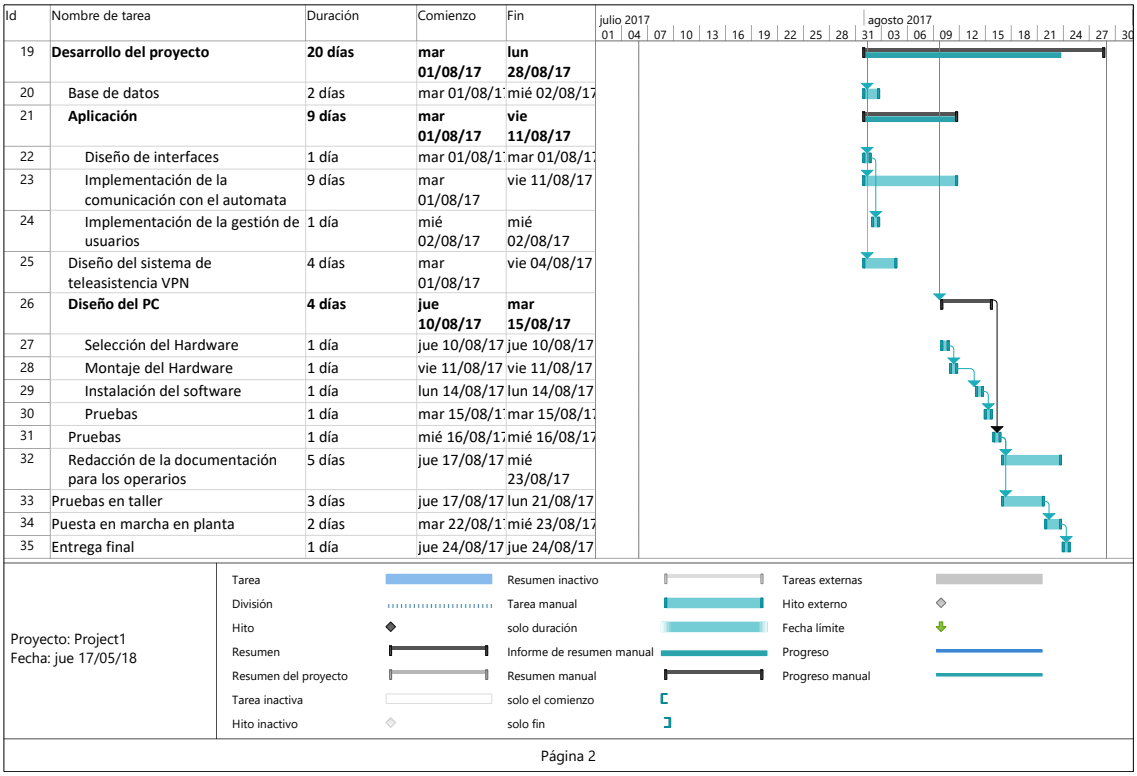


Figura 3.2: Diagrama de Gantt inicial



Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Inicio**	**2 días**	**jue 06/07/17**	**vie 07/07/17**
2	Definir alcance del proyecto	1 día	jue 06/07/17	jue 06/07/17
3	Documentar y planificar las fases	1 día	vie 07/07/17	vie 07/07/17
4	**Documentar y planificar el proyecto**	**5 días**	**lun 10/07/17**	**vie 14/07/17**
5	Entender el funcionamiento de los autómatas y los diferentes sistemas a controlar	3 días	lun 10/07/17	mié 12/07/17
6	Definir funcionalidades a desarrollar	2 días	jue 13/07/17	vie 14/07/17
7	**Planificar el proyecto**	**3 días**	**lun 17/07/17**	**mié 19/07/17**
8	Describir el proyecto y estimar las fechas	1 día	lun 17/07/17	lun 17/07/17
9	Crear diagrama de Gantt	2 días	mar 18/07/17	mié 19/07/17
10	**Desarrollo técnico del proyecto**	**8 días**	**jue 20/07/17**	**lun 31/07/17**
11	Aprendizaje de la biblioteca de comunicaciones	5 días	jue 20/07/17	mié 26/07/17
12	**Definir requisitos del proyecto**	**2 días**	**jue 27/07/17**	**vie 28/07/17**
13	Diagrama de casos de uso	1 día	jue 27/07/17	jue 27/07/17
14	Definir y documentar requisitos de datos	1 día	vie 28/07/17	vie 28/07/17
15	Definir requisitos tecnológicos	1 día	vie 28/07/17	vie 28/07/17
16	**Análisis**	**2 días**	**vie 28/07/17**	**lun 31/07/17**
17	Diagrama de clases	1 día	vie 28/07/17	vie 28/07/17
18	Documentación de clases	1 día	lun 31/07/17	lun 31/07/17

Página 1



Id	Nombre de tarea	Duración	Comienzo	Fin
19	**Desarrollo del proyecto**	**20 días**	**mar 01/08/17**	**lun 28/08/17**
20	Base de datos	2 días	mar 01/08/17	mié 02/08/17
21	**Aplicación**	**9 días**	**mar 01/08/17**	**vie 11/08/17**
22	Diseño de interfaces	1 día	mar 01/08/17	mar 01/08/17
23	Implementación de la comunicación con el automata	9 días	mar 01/08/17	vie 11/08/17
24	Implementación de la gestión de usuarios	1 día	mié 02/08/17	mié 02/08/17
25	Diseño del sistema de teleasistencia VPN	4 días	mar 01/08/17	vie 04/08/17
26	**Diseño del PC**	**4 días**	**jue 10/08/17**	**mar 15/08/17**
27	Selección del Hardware	1 día	jue 10/08/17	jue 10/08/17
28	Montaje del Hardware	1 día	vie 11/08/17	vie 11/08/17
29	Instalación del software	1 día	lun 14/08/17	lun 14/08/17
30	Pruebas	1 día	mar 15/08/17	mar 15/08/17
31	Pruebas	1 día	mié 16/08/17	mié 16/08/17
32	Redacción de la documentación para los operarios	5 días	jue 17/08/17	mié 23/08/17
33	Pruebas en taller	3 días	jue 17/08/17	lun 21/08/17
34	Puesta en marcha en planta	2 días	mar 22/08/17	mié 23/08/17
35	Entrega final	1 día	jue 24/08/17	jue 24/08/17

Página 2

Figura 3.3: Diagrama de Gantt de seguimiento de la planificación

Capítulo 4

Análisis y diseño del sistema

Índice del capítulo

En este capítulo se expone la fase inicial del proyecto, detallando el análisis de necesidades y casos de uso. Posteriormente y apoyándose en este análisis se muestra el diagrama de clases seguido del diseño de la base de datos, para finalmente explicar el diseño de las interfaces.

4.1. Especificación de casos de uso

En la especificación de casos de uso se recopila el diagrama de casos de uso que describe y explica la funcionalidad de los diferentes componentes de la aplicación.

El objetivo principal de esta documentación fue la de fijar objetivos de funcionalidad que debía satisfacer el proyecto, teniendo en cuenta que durante la etapa de implementación y pruebas, pudieran surgir otras funcionalidades o que las proyectadas se modificasen. Para obtener el documento que se muestra, en primer lugar se analizaron las diferentes áreas de actuación de los diferentes usuarios que iban a tener relación con el programa y posteriormente se consensuaron con el cliente y el director del proyecto en la empresa.

En la figura 4.1 se puede ver el diagrama de casos de uso de la aplicación de control descrita en este documento.

Los casos de uso se acompañan también con las plantillas que los detallan.

Descripción del caso de uso	
Identificador	CU.01
Nombre	Operaciones de marcha y paro
Versión	1.0
Autor	Rafael Pallarés Palos
Fuentes	Euroelettra Ingeniería S.L
Descripción	El sistema debe permitir a los operarios de planta tener acceso a las funciones de paro y marcha de la máquina
Alcance	Ciclo de uso de la aplicación en modo de operario
Nivel	Tarea principal
Actor principal	Operario
Actores secundarios	-
Precondición	El sistema tiene corriente y no hay alarmas en el mismo
Condición final con éxito	El usuario es capaz de dar las ordenes de paro , marcha y puesta a cero de la máquina
Condición final con fracaso	Los comandos de paro y marcha no se ejecutan correctamente
Trigger	El usuario debe iniciar o parar una producción
Secuencia normal	**Acción**
1	El usuario ha arrancado el equipo y la aplicación se muestra en pantalla sin alarmas
2	El usuario presiona uno de los botones de actuación
3	El sistema envía la orden al autómata
Exepciones	**Excepción 1**
1	El sistema tiene alarmas
2	El usuario deberá verificar el estado físico de la máquina
3	El usuario deberá aceptar las alarmas en el sistema
4	Reintentar acción
Frecuencia esperada	Varias veces al día
Importancia	Vital
Prioridad	Alta
Comentarios	Puesto que este requisito está diseñado para que el operario pueda detener el funcionamiento de la máquina en cualquier momento, la aplicación deberá ser capaz de realizar estas acciones siempre

Cuadro 4.1: CU.01- Operaciones de marcha y paro

Descripción del caso de uso	
Identificador	CU.02
Nombre	Puesta a cero
Versión	1.0
Autor	Rafael Pallarés Palos
Fuentes	Euroelettra Ingeniería S.L
Descripción	El sistema debe permitir a los operarios de planta poner a cero los contadores y con ello restablecer la posición de la máquina
Alcance	Ciclo de uso de la aplicación en modo de operario
Nivel	Tarea principal
Actor principal	Operario
Actores secundarios	-
Precondición	El sistema tiene corriente y no hay alarmas en el mismo
Condición final con éxito	El usuario es capaz de dar la orden de puesta a cero al sistema
Condición final con fracaso	El comando de puesta a cero no se ejecuta correctamente
Trigger	El usuario quiere poner a cero los contadores y restablecer la posición de la máquina
Secuencia normal	**Acción**
1	El usuario ha arrancado el equipo y la aplicación se muestra en pantalla sin alarmas
2	El usuario presiona el botón de puesta a cero
3	El sistema envía la orden al autómata
Exepciones	**Excepción 1**
1	El sistema tiene alarmas
2	El usuario deberá verificar el estado físico de la máquina
3	El usuario deberá aceptar las alarmas en el sistema
4	Reintentar acción
Frecuencia esperada	Varias veces a la semana
Importancia	Alta
Prioridad	Alta
Comentarios	Puesto que para cada cambio de modelo de producción se deberán reiniciar la máquina y los contadores, esta acción está permitida a los usuarios no supervisores

Cuadro 4.2: CU.02 - Puesta a cero

Descripción del caso de uso	
Identificador	CU.03
Nombre	Configuración de parámetros
Versión	1.0
Autor	Rafael Pallarés Palos
Fuentes	Euroelettra Ingeniería S.L
Descripción	El sistema debe permitir al supervisor modificar los parámetros de configuración de la máquina
Alcance	Ciclo de uso de la aplicación en modo de supervisor
Nivel	Tarea principal
Actor principal	Supervisor
Actores secundarios	-
Precondición	El usuario ha iniciado sesión como supervisor
Condición final con éxito	El usuario es capaz de cambiar los parámetros de la máquina
Condición final con fracaso	Los parámetros no se pueden cambiar
Trigger	El usuario quiere cambiar los parámetros de configuración de la máquina
Secuencia normal	**Acción**
1	El usuario ha iniciado sesión como administrador
2	El usuario selecciona en el menú herramientas la opción Parámetros o pulsa la tecla F3 del teclado
3	El sistema muestra la lista de parámetros actualmente en uso
4	El usuario modifica los parámetros, que se envían al autómata
Exepciones	**Excepción 1**
1	El usuario no ha iniciado sesión como supervisor
2	El usuario deberá iniciar sesión como supervisor
3	El usuario deberá volver a intentar la acción
Exepciones	**Excepción 2**
1	El sistema tiene problemas para comunicar con el autómata
2	El usuario deberá comprobar las conexiones del sistema
3	El usuario deberá volver a intentar la acción, si el problema persiste deberá contactar con el soporte técnico
Frecuencia esperada	Baja
Importancia	Alta
Prioridad	Alta
Comentarios	–

Cuadro 4.3: CU.03 - Configuración de parámetros

Descripción del caso de uso	
Identificador	CU.04
Nombre	Visualización de estado
Versión	1.0
Autor	Rafael Pallarés Palos
Fuentes	Euroelettra Ingeniería S.L
Descripción	El sistema debe permitir al supervisor visualizar el estado de los componentes hardware de la máquina
Alcance	Ciclo de uso de la aplicación en modo de supervisor
Nivel	Tarea ocasional
Actor principal	Supervisor
Actores secundarios	-
Precondición	El usuario ha iniciado sesión como supervisor
Condición final con éxito	El usuario es capaz de visualizar el estado de los componentes hardware de la máquina mediante la interfaz
Condición final con fracaso	No se puede visualizar el estado
Trigger	El usuario quiere visualizar el estado de la máquina
Secuencia normal	**Acción**
1	El usuario se ha identificado como supervisor
2	El usuario selecciona en el menú herramientas la opción visualización o pulsa la tecla V del teclado
3	El sistema muestra la ventana de visualización
Exepciones	**Excepción 1**
1	El usuario no ha iniciado sesión como supervisor
2	El usuario deberá iniciar sesión como supervisor
3	El usuario deberá volver a intentar la acción
Frecuencia esperada	Esporádica
Importancia	Normal
Prioridad	Baja
Comentarios	–

Cuadro 4.4: CU.04 - Visualización de estado

Descripción del caso de uso	
Identificador	CU.05
Nombre	Habilitar teleasistencia
Versión	1.0
Autor	Rafael Pallarés Palos
Fuentes	Euroelettra Ingeniería S.L
Descripción	El supervisor debe ser capaz de habilitar la teleasistencia
Alcance	Durante el periodo que dure la asistencia remota
Nivel	Tarea ocasional
Actor principal	Supervisor
Actores secundarios	-
Precondición	El usuario ha iniciado sesión como supervisor
Condición final con éxito	El usuario es capaz de habilitar la teleasistencia
Condición final con fracaso	El usuario no puede habilitar la asistencia remota
Trigger	El usuario necesita asistencia remota
Secuencia normal	**Acción**
1	El usuario ha arrancado el equipo y la aplicación se muestra en pantalla
2	El usuario selecciona en el menú Herramientas la opción teleasistencia o presiona la tecla T del teclado
3	El sistema establece la conexión y comprueba la IP
4	El sistema muestra la IP de conexión que el usuario deberá facilitar a los técnicos
Exepciones	**Excepción 1**
1	El usuario no ha iniciado sesión como supervisor
2	El usuario deberá iniciar sesión como supervisor
3	El usuario deberá volver a intentar la acción
Exepciones	**Excepción 2**
1	El usuario ha iniciado sesión como supervisor
2	El sistema muestra el aviso de que no hay salida a internet
3	El usuario deberá comprobar la conexión a internet de la planta de producción
4	Reintentar la acción
Frecuencia esperada	Esporádica
Importancia	Normal
Prioridad	Baja
Comentarios	–

Cuadro 4.5: CU.05 - Habilitar teleasistencia

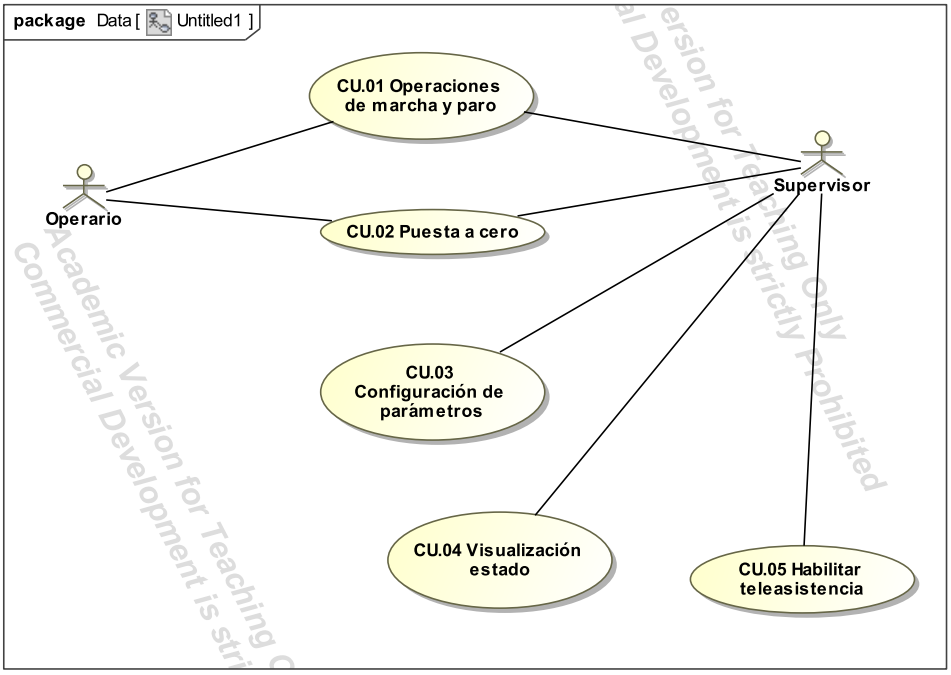


Figura 4.1: Diagrama de casos de uso

4.2. Diagrama de clases

A continuación se presenta un estudio sobre el diagrama de clases, basado en los requisitos definidos para el proyecto. En esta fase, se definen las clases que existen en el sistema que se ha desarrollado y la relación entre ellas si la hay. La figura 4.2 muestra el diagrama de clases y sus relaciones.

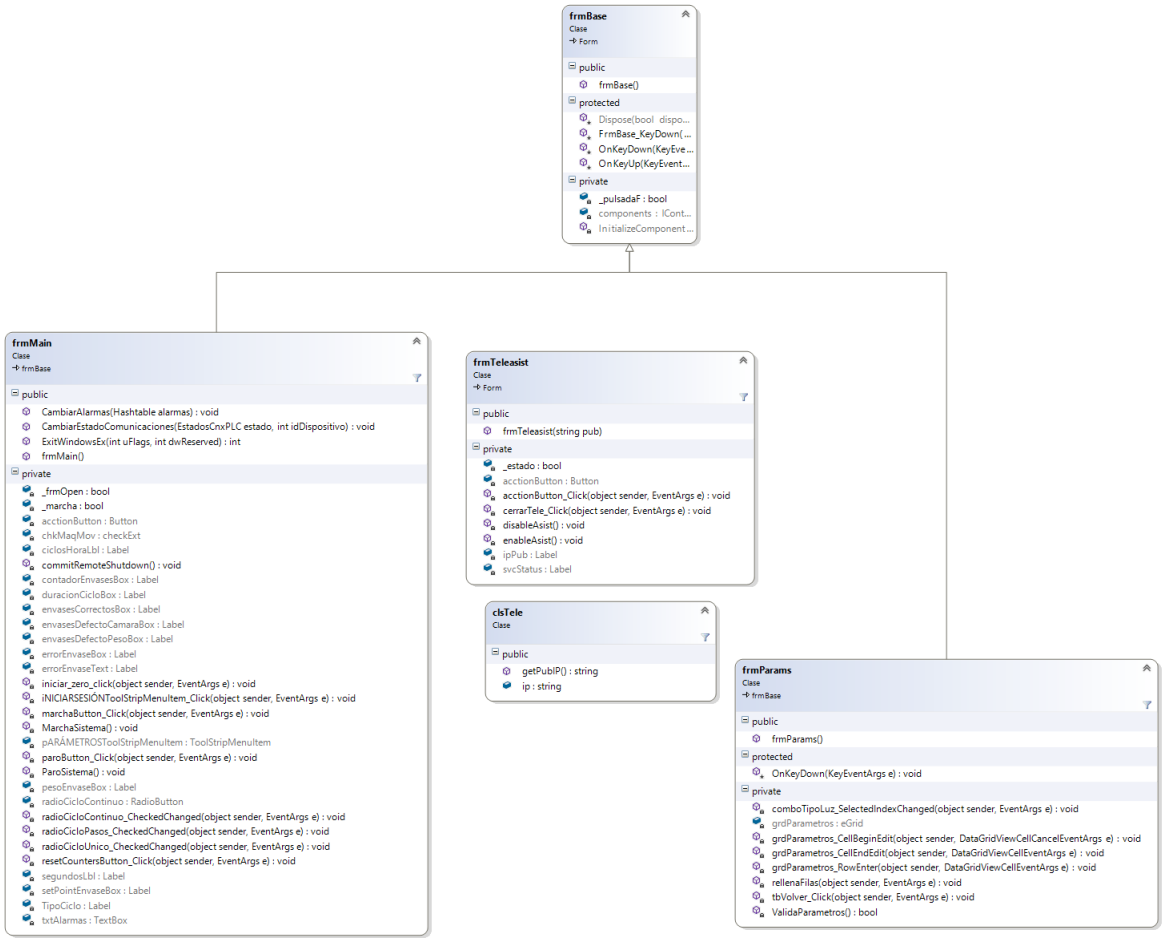


Figura 4.2: Diagrama de clases del proyecto, en el que se muestran las relaciones entre la superclase `frmBase` y sus descendientes `frmMain y frmParams`, así como las clases `clsTele y frmTeleasist` utilizadas en la teleasistencia. Las clases se muestran con más detalle en las figuras 4.3 4.4 4.5 y 4.6

- **frmBase**: Superclase de la que descenderán el formulario principal `frmMain` y el formulario `frmParams`.Ver figura 4.3.
- **frmMain**: Clase del formulario principal de la aplicación, descendiente de `frmBase`. Contiene los métodos de utilidad de la aplicación como por ejemplo `MarchaSistema()` o `CommitRemoteShutdown()`.Ver figura 4.4.
- **frmParams**: Clase del formulario de gestión de parámetros, descendiente de `frmBase`. Contiene los métodos de gestión de una tabla como `grd_Parametros_cellEndEdit()`,

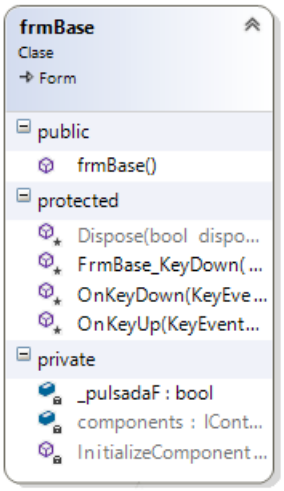


Figura 4.3: Diagrama de la clase `frmBase`

método en el que se ejecutará el envío al autómata de los parámetros. O métodos de comodidad como `ValidaParametros()`, al que se llamará para verificar que los parámetros introducidos, tienen el formato correcto antes de ser enviados al PLC. Ver figura 4.5.

- **frmTeleasist**: Clase de gestión del formulario de teleasistencia. Con métodos para la activación y desactivación de la teleasistencia como `enableAsist()`. Ver figura 4.6.
- **clsTele**: Clase de gestión del servicio de teleasistencia. Con métodos para la obtención de la IP pública del equipo.

4.3. Diseño de la base de datos

En este apartado se presenta el diseño de la base de datos. Por política de empresa y para seguir con el estilo de las otras aplicaciones desarrolladas en *Euroelettra Ingeniería*, se ha diseñado una estructura de bases de datos dividida en 3 bases de datos separadas. Esto facilita la gestión de copias de seguridad además de asegurar la compatibilidad con otros proyectos de la empresa y así asegurar que otro compañero que no ha desarrollado este proyecto, puede hacerse cargo de una reparación.

Las bases de datos se dividen en tres áreas funcionales:

- Histórico: Donde se almacenarán los datos de históricos y registros de inicio de sesión y alarmas.
- Permisos: En la que se almacenarán todo los datos de configuración del sistema de gestión de usuarios, tales como las contraseñas y los grupos de permiso.
- Proceso: Donde se almacenan los datos de configuración del programa. Datos como los parámetros de conexión con el autómata y controles a ocultar o mostrar.

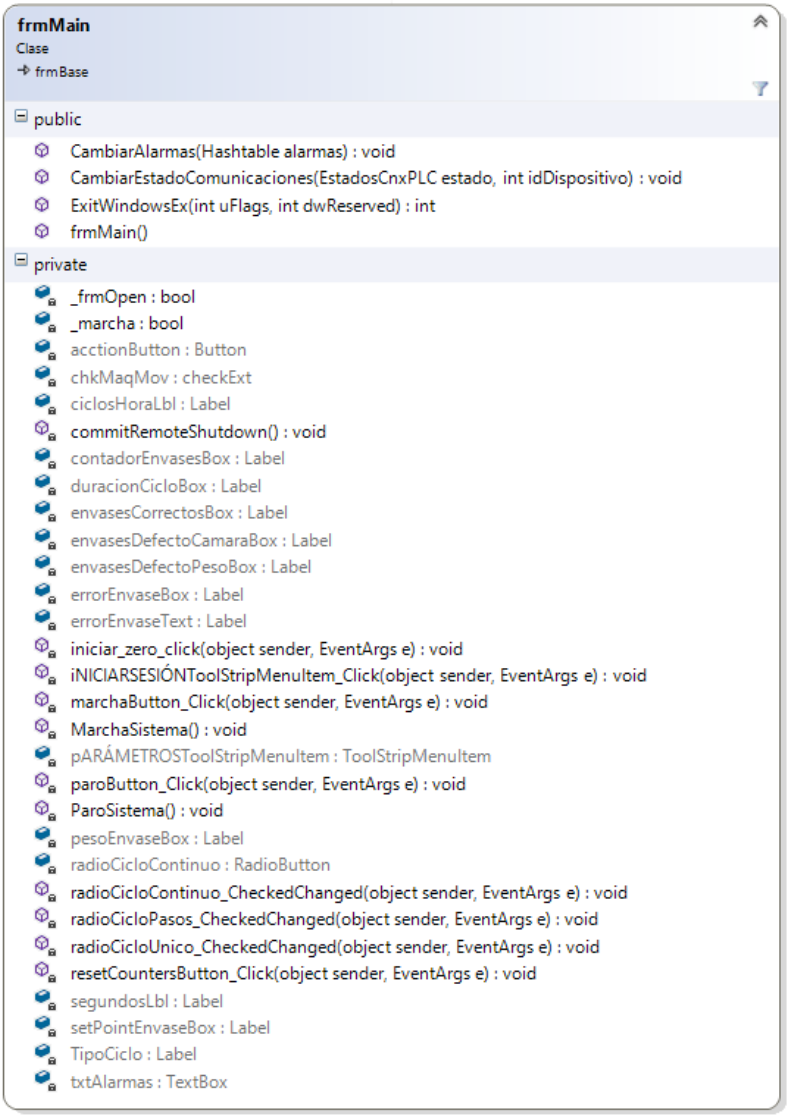


Figura 4.4: Diagrama de la clase `frmMain`

4.3.1. Histórico

En esta base de datos se encuentran dos tablas que se pueden ver en la figura 4.7. En primer lugar la tabla *HistoricoAlarmas* en la que se almacenarán todas las alarmas que ha generado el sistema junto con su hora de inicio, hora de aceptación y hora de fin, su duración y la descripción. En segundo lugar se encuentra la tabla *HistoricoEventos*, tabla en la que se almacenarán todos los eventos generados por la aplicación, como eventos de paro y marcha.

4.3.2. Permisos

La base de datos "permisos", almacena información referente a la gestión de usuarios y sus permisos para realizar acciones en la aplicación. Sobre esta base de datos se apoya el sistema

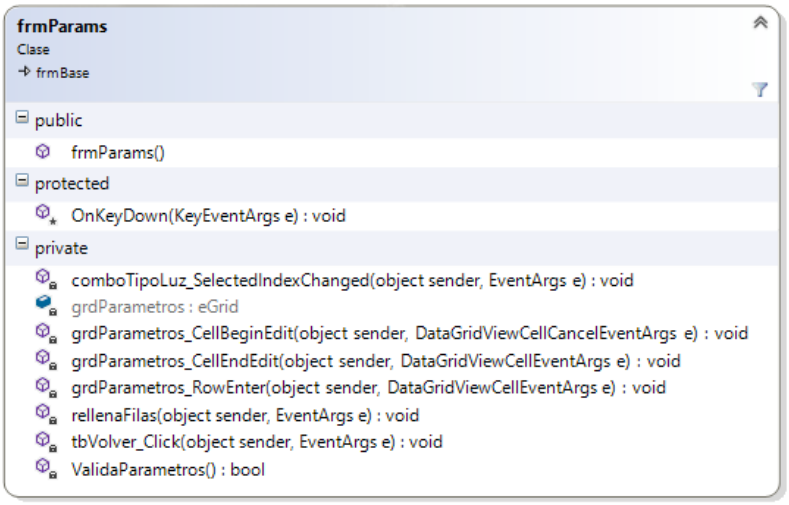


Figura 4.5: Diagrama de la clase `frmParams`

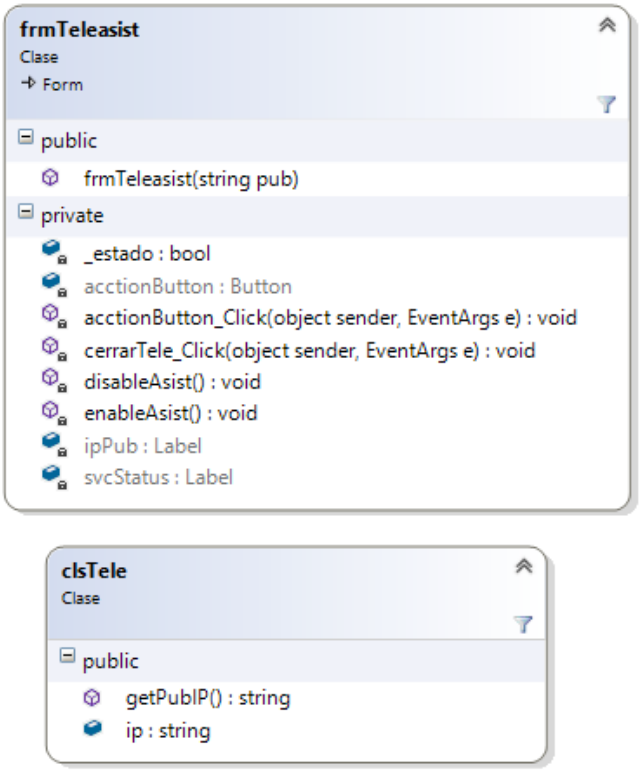


Figura 4.6: Diagrama de la clase `frmTele` y `clsTele`

de gestión de usuarios explicado en el apartado 5.6. En la figura 4.8 se muestra la distribución de las tablas en esta base de datos. En primer lugar se puede ver la tabla Permisos, en la que se almacenará un identificador de acceso único para cada nombre de usuario, también se almacena

Figura 4.7: Esquema de la base de datos Histórico

un valor booleano para marcar si el usuario es un grupo y un valor booleano para marcar si el usuario tiene el acceso denegado.

En segundo lugar, se encuentra la tabla Relaciones, utilizada para enlazar los usuarios con los grupos a los que pertenecen, que se definen en la cuarta tabla llamada Grupos. La mera existencia de esta tabla, indica que este sistema se ha heredado sin actualizar.

En tercer lugar se puede ver la tabla Usuarios, encargada de almacenar el `hash` de las contraseñas y el nombre de los usuarios. A continuación se describe la tabla Registro, encargada de llevar un log de accesos, donde se almacenará cada vez que un usuario ha iniciado sesión.

Por último se puede ver la tabla Accesos, encargada de definir las diferentes acciones disponibles en la aplicación y relacionarlos con los usuarios definidos en la tabla usuarios.

4.3.3. Proceso

Por último se encuentra la base de datos "proceso", dedicada a almacenar datos de configuración de la aplicación y de la biblioteca de comunicaciones. En la figura 4.9 se puede apreciar la estructura de tablas de esta base de datos.

En primer lugar se puede ver la tabla Alarmas, en la que se definen todas las alarmas disponibles en el sistema, su dirección de memoria en el autómata y su descripción. Esta tabla es utilizada por la biblioteca de comunicaciones para mostrar alarmas en el sistema cuando el PLC activa una señal en la dirección de memoria indicada en alguna de las filas de esta tabla.

En segundo lugar encontramos las tablas Puertos y Dispositivos, que deberán ir siempre en conjunción, ya que son las encargadas de almacenar la configuración de comunicación con los autómatas. Se definen valores como la IP del autómata, el puerto de comunicación, el tipo de PLC, el protocolo de comunicaciones o el modo de comunicación, entre otros. En el caso

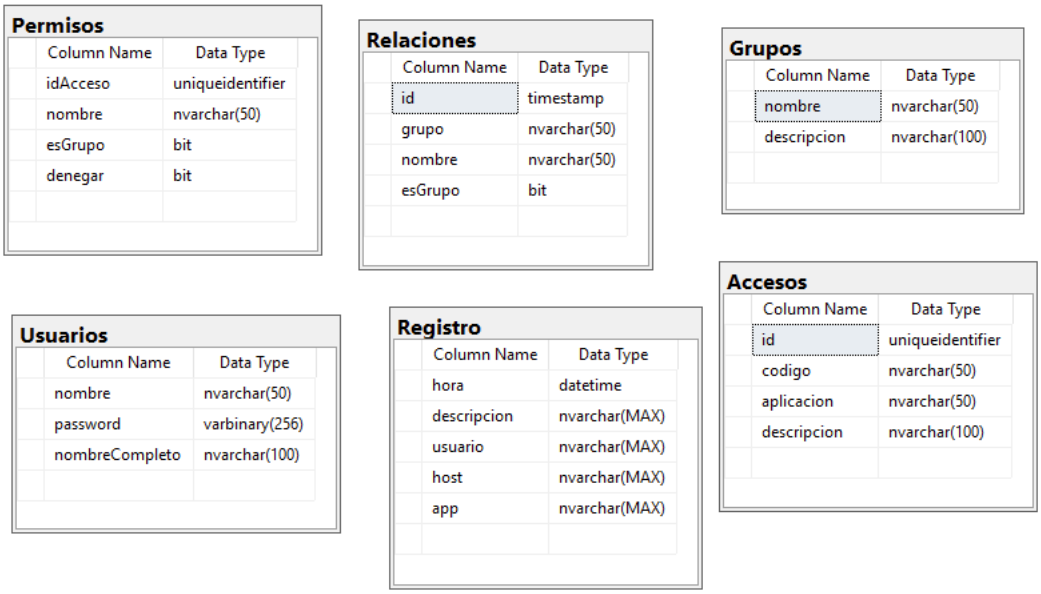


Figura 4.8: Esquema de la base de datos Permisos

del proyecto que ocupa este documento, solo se está comunicando con un autómata, pero para preservar la compatibilidad con el resto de aplicaciones de la empresa, se ha conservado esta estructura.

A continuación se encuentra la tabla controles, que se encarga de indicar a la biblioteca de comunicaciones qué controles ocultar o mostrar según se activen o desactiven respectivamente, las señales en las direcciones de memoria especificadas en esta tabla.

Por último, la tabla Tipos Datos, almacena los tipos de datos disponibles en la aplicación, como pueden ser *INT32, BIT, REAL32* y demás.

4.4. Diseño de la interfaz

Las restricciones impuestas por el cliente a la hora de diseñar la interfaz fueron una fuerte influencia ya que muchas de ellas propiciaban un rediseño completo. La pantalla en la que se muestra la aplicación final en producción tiene una resolución de 800x600 píxeles, por lo que el tamaño de la ventana de la aplicación y sus cuadros de diálogo no podían superar ese tamaño. Además, durante el desarrollo y pruebas, se comprobó que tan solo unos pocos botones de acción, como son el de paro, marcha o cambio de ciclo, eran los más utilizados y debían ser más grandes. Por estos motivos, el *boceto* presentado en la figura 4.10 dista mucho del producto real ya que fue realizado en fases muy tempranas del desarrollo, en las que no se habían fijado estos condicionantes.

Por otra parte, señalar que muchas de las funciones de la aplicación han sido asignadas a combinaciones de teclas o atajos de teclado, para facilitar su uso en producción.

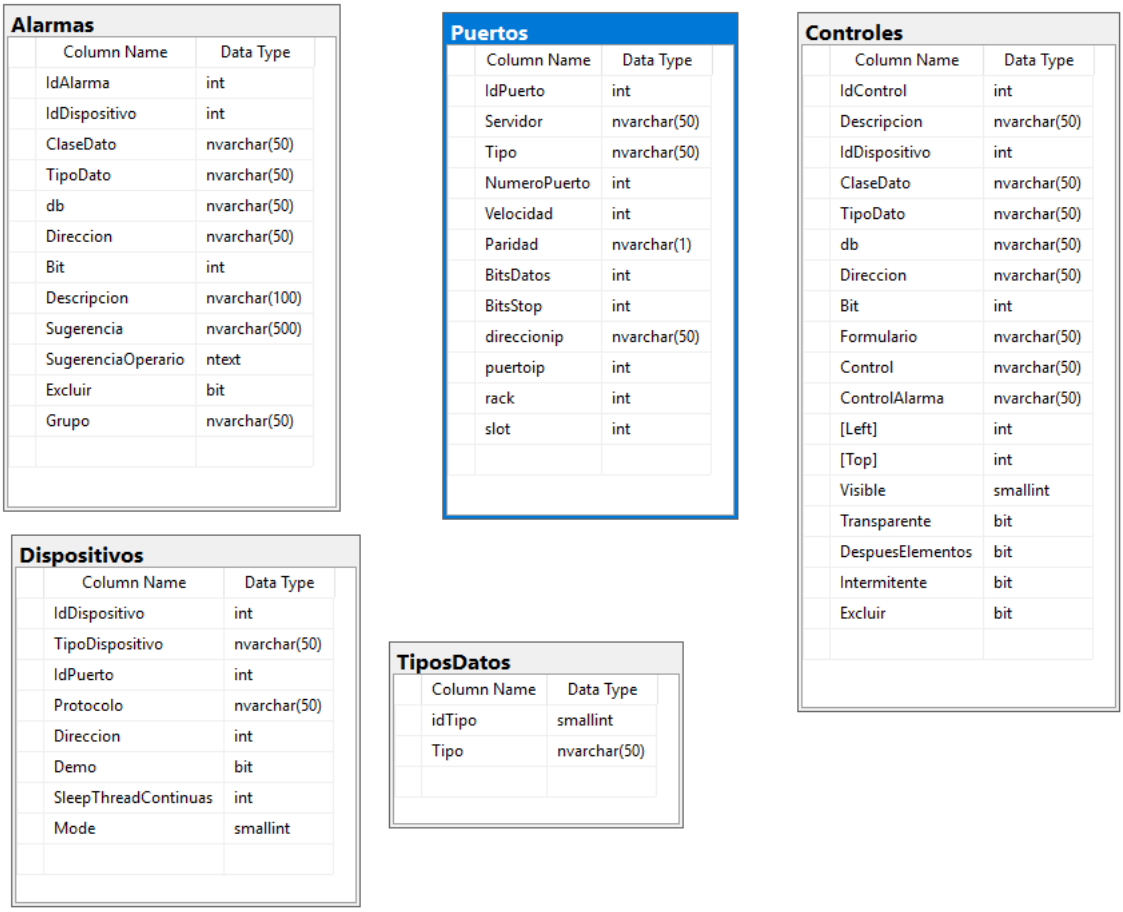


Figura 4.9: Esquema de la base de datos Proceso

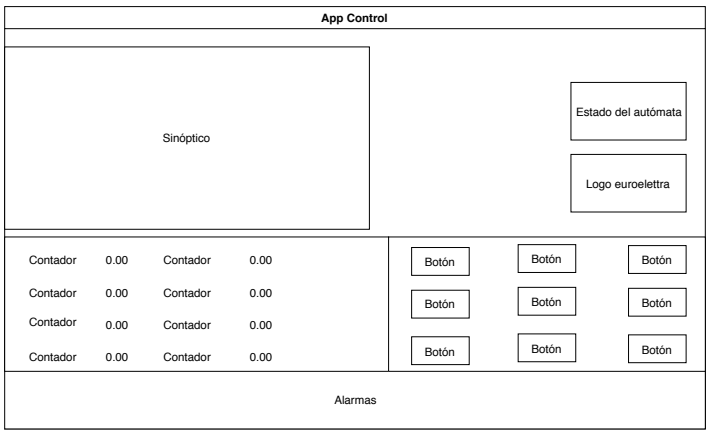


Figura 4.10: Boceto de la aplicación inicial

En la figura 4.11 se puede apreciar el aspecto final de la aplicación en producción así como en la figura 4.12 se puede apreciar la interfaz de teleasistencia como cuadro de diálogo por encima de la ventana principal.

En el cuadro de diálogo de teleasistencia, cabe destacar que se añadió un resaltado en amarillo para que los números de la dirección IP fueran fácilmente visibles por los operarios, dadas las malas condiciones de iluminación de la planta de producción, que dificultaban la identificación de algunos números.

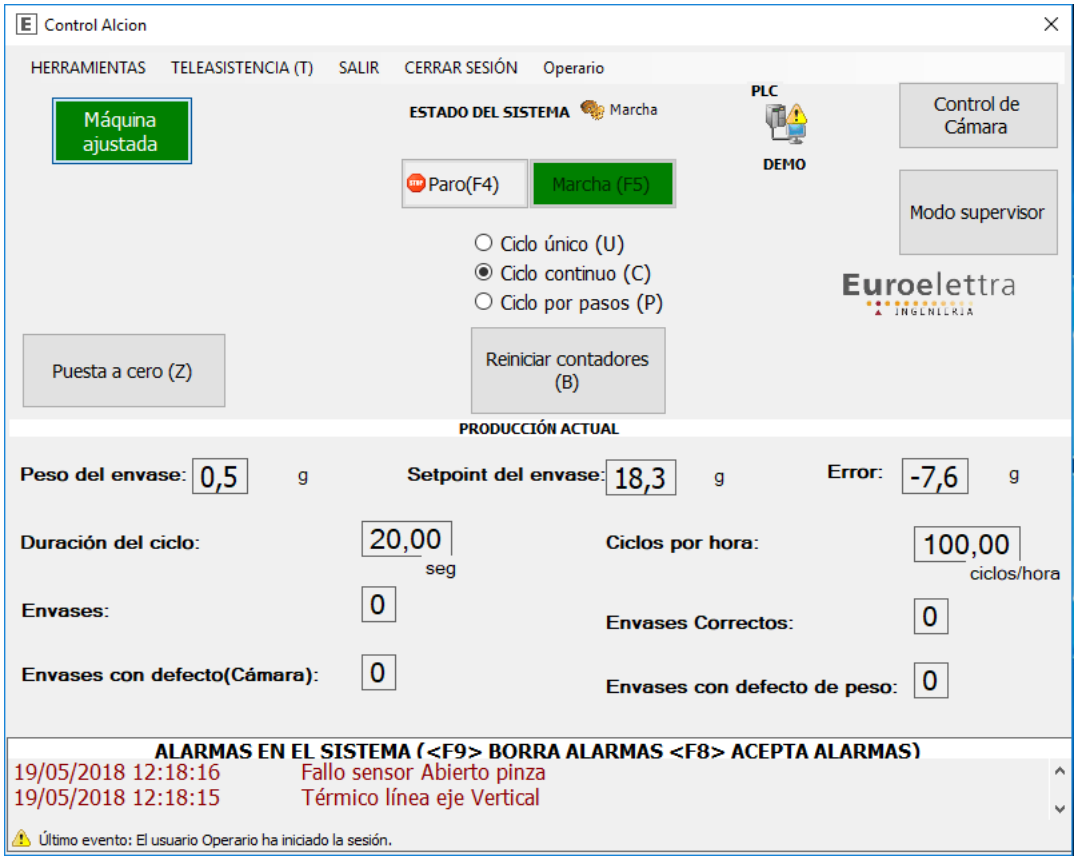


Figura 4.11: Aspecto final de la aplicación

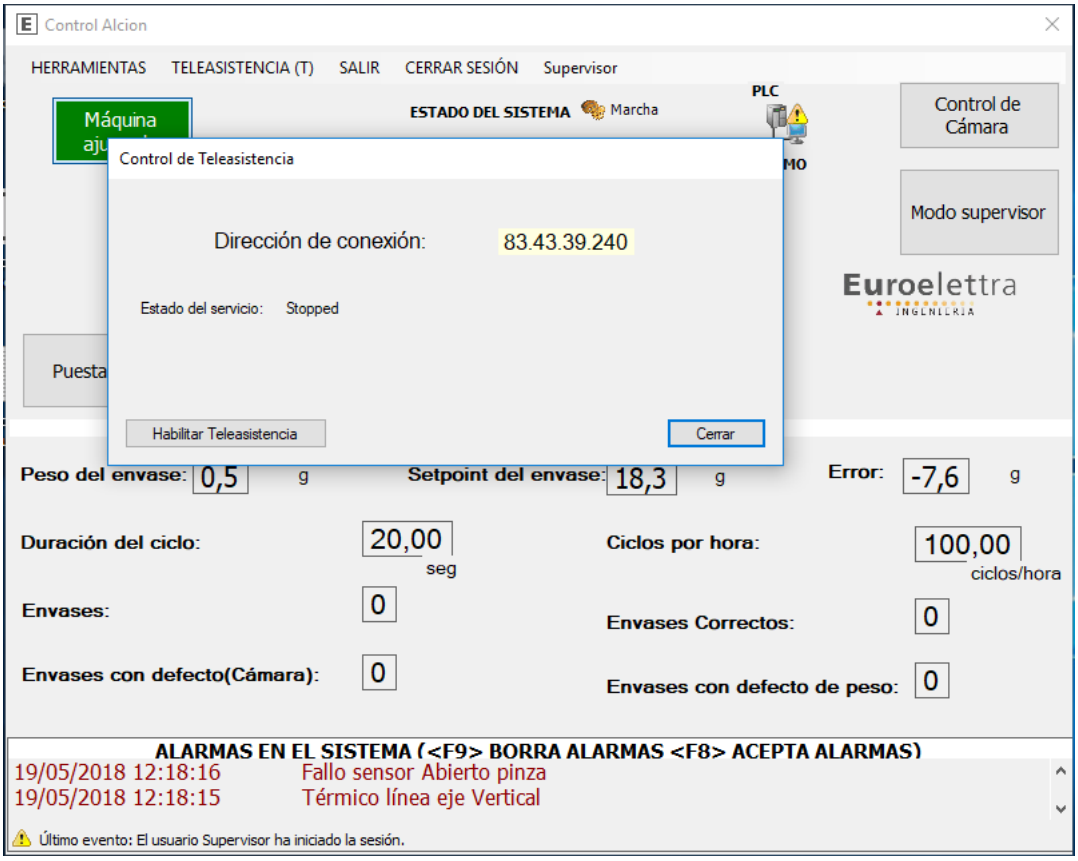


Figura 4.12: Aspecto final de la aplicación con un cuadro de diálogo abierto

Capítulo 5

Implementación y pruebas

Índice del capítulo

En este capítulo se detalla el desarrollo del proyecto, abordando la implementación de la comunicación con el autómata, utilizando la biblioteca desarrollada por *Euroelettra Ingenería*, para conseguir la funcionalidad de lecturas continuas, que permita leer o escribir los datos necesarios en el autómata así como mantener los datos actualizados en pantalla.

También se detalla la implementación del sistema de envío de parámetros al autómata, para enviar y leer los parámetros de configuración de la máquina.

Se continúa con las funciones de apagado remoto para poder apagar el equipo mediante una señal del autómata y el arranque retrasado, que permitirá lanzar la aplicación en el arranque del PC solo cuando el sistema esté listo.

Se continúa con la implementación de la teleasistencia, que facilitará a los técnicos la modificación o reparación a distancia, mediante un túnel VPN. Se expone a continuación la gestión de usuarios, también utilizando la biblioteca propia de la empresa.

Se cierra el capítulo con un apartado donde se detallan las pruebas que se realizaron en el taller para comprobar la funcionalidad de la aplicación.

5.1. Comunicación con el autómata

La biblioteca de comunicaciones de *Euroelettra Ingenería* realiza las funciones de lectura y escritura al autómata, abstrayendo al programador los detalles más complejos de la comunicación, como podrían ser la construcción de tramas Ethernet, el control del tamaño de los paquetes y demás elementos de bajo nivel de la comunicación. La biblioteca proporciona métodos *delegados* a los que se invocará cuando se realicen cambios de estado, o de datos. Como primer ejemplo el listado 5.1 muestra el delegado `_CambioEstadoComunicaciones`, que será invocado cuando algún autómata cambie su disponibilidad.

Listado 5.1: Comunicación y cambio de estado de las comunicaciones

```
private void _engine_CambioEstadoComunicaciones(object sender,
    CambioEstadoComunicacionesEventArgs e)
  {
    EstadosCnxPLC estado = EstadosCnxPLC.CnxError; //Se define conexión con error por
        defecto
     if (e.Comunica) //Si el autómata comunica
      {
        estado = EstadosCnxPLC.CnxOk;
     }
        if (App.FrmMain != null)
            App.FrmMain.Invoke(new CambiarEstadoComunicacionesInvoker(App.FrmMain.
                CambiarEstadoComunicaciones), new object[] { estado, e.IdDispositivo }); //
                Se realiza la acción correspondiente
      }
```

En el caso del fragmento de código mostrado, se utiliza el método `invoke` estándar de *.NET*, para invocar al formulario principal y notificar del cambio de estado de comunicaciones.

Otros delegados como `onCambioAlarmas` u `onCambioDato`, están disponibles pero no se expondrán para reducir la extensión de este documento.

Configuración de la biblioteca

Para realizar sus funciones, la biblioteca, deberá tener datos de configuración, tales como la IP de los autómatas con los que comunicar, puerto al que comunicar y demás datos de la conexión. Estos datos se configuran a través de la base de datos expuesta en el apartado 4.3, en la que se puede ver una base de datos llamada *Proceso* que almacenará los datos de configuración.

Las tablas *Dispositivos* y *Puertos* definen los parámetros de configuración de la conexión con los autómatas. La figura 5.1, muestra la configuración del único autómata utilizado en este proyecto. Cabe destacar que muchos de los parámetros que estas tablas son capaces de almacenar, no se utilizan en este caso y están reservados para configuraciones con muchos autómatas o con otros protocolos de comunicación.

	IdDispositivo	TipoDispositivo	IdPuerto	Protocolo	Direccion	Demo	SleepThreadC...	Mode
▶	0	PLCOmron	1	FINS	1	True	1	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

	IdPuerto	Servidor	Tipo	NumeroPuerto	BitsDatos	BitsStop	direccionip	puertoip	rack	slot
	1	NULL	UDP	1	NULL	NULL	192.168.0.100	9600	NULL	NULL
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 5.1: Tablas Dispositivos y Puertos y de configuración de la comunicación con el autómata

Ocultación de controles

Otra de las funcionalidades que proporciona la biblioteca es la función de ocultación y mostrado de los controles, que mediante la configuración en la tabla *Controles*, de la base de datos *Proceso*, es capaz de mostrar u ocultar elementos de la interfaz en función de los datos leídos del autómata.

Esta característica facilita la activación y desactivación de botones y pilotos de visualización. En la figura 5.2, se puede ver un fragmento de la tabla çontrolesçorrespondiente al formulario de visualización explicado en el apartado 5.3.

	IdControl	Descripcion	IdDispositivo	ClaseDato	TipoDato	db	Direccion	Bit	Formulario	Control	ControlAlarma
▶	1	check maquina...	0	DM	BIT16	0	1006	0	NULL	chkMaquinaAj...	0
	2	SVinres1	0	DM	BIT16	0	1000	0	frmVisu	SVinres1	0
	3	SVinson1	0	DM	BIT16	0	1000	1	frmVisu	SVinson1	0
	4	SVinst11	0	DM	BIT16	0	1000	2	frmVisu	SVinst11	0
	5	SVindi01	0	DM	BIT16	0	1000	3	fmVisu	SVindi01	0
	6	SVindi11	0	DM	BIT16	0	1000	4	frmVisu	SVindi11	0
	7	SVoutalm1	0	DM	BIT16	0	1000	8	frmVisu	SVoutalm1	0
	8	SVoutinp	0	DM	BIT16	0	1000	9	frmVisu	SVoutinp1	0
	9	SVoutrd1	0	DM	BIT16	0	1000	10	frmVisu	SVoutrd1	0
	10	SVoutzp1	0	DM	BIT16	0	1000	11	frmVisu	SVoutzp1	0
	11	SHinres1	0	DM	BIT16	0	1001	0	frmVisu	SHinres1	0
	12	SHinson1	0	DM	BIT16	0	1001	1	frmVisu	SHinson1	0
	13	SHinst11	0	DM	BIT16	0	1001	2	frmVisu	SHinst11	0
	14	SHindi01	0	DM	BIT16	0	1001	3	frmVisu	SHindi01	0
	15	SHindi11	0	DM	BIT16	0	1001	4	frmVisu	SHindi11	0
	16	SHoutalm1	0	DM	BIT16	0	1001	8	frmVisu	SHoutalm1	0
	17	SHoutinp1	0	DM	BIT16	0	1001	9	frmVisu	SHoutinp1	0
	18	SHoutrd1	0	DM	BIT16	0	1001	10	frmVisu	SHoutrd1	0
	19	SHoutzp1	0	DM	BIT16	0	1001	11	frmVisu	SHoutzp1	0
	20	Ginres1	0	DM	BIT16	0	1002	0	frmVisu	Ginres1	0
	21	Ginson1	0	DM	BIT16	0	1002	1	frmVisu	Ginson1	0
	22	Ginsetup1	0	DM	BIT16	0	1002	2	frmVisu	Ginsetup1	0
	23	Gindri1	0	DM	BIT16	0	1002	3	frmVisu	Gindri1	0
	24	Ginin01	0	DM	BIT16	0	1002	4	frmVisu	Ginin01	0
	25	Ginin11	0	DM	BIT16	0	1002	5	frmVisu	Ginin11	0
	26	Goutalarm1	0	DM	BIT16	0	1002	8	frmVisu	Goutalarm1	0
	27	Goutestop1	0	DM	BIT16	0	1002	9	frmVisu	Goutestop1	0
	28	Goutsvre1	0	DM	BIT16	0	1002	10	frmVisu	Goutsvre1	0
	29	Goutseton1	0	DM	BIT16	0	1002	11	frmVisu	Goutseton1	0
	30	Goutbusy1	0	DM	BIT16	0	1002	12	frmVisu	Goutbusy1	0
	33	DEstart1	0	DM	BIT16	0	1003	0	frmVisu	DEstart1	0
	34	DEinsel01	0	DM	BIT16	0	1003	1	frmVisu	DEinsel01	0

Figura 5.2: Fragmento de la tabla de configuración de ocultación de controles

Envío de comandos

Una de las partes más importantes de la funcionalidad de la aplicación de control, reside en la capacidad de enviar comandos de acción al autómata. Estos comandos indican al PLC las

órdenes de paro, marcha, reinicio, puesta a cero y demás operaciones de funcionamiento.

La implementación del envío de dichos comandos es similar para todos, por lo que se expone uno de ellos en el listado 5.2, en el que se explica la forma de lanzar un comando de paro al autómata. Suponiéndose que la dirección 1151 es la asignada para registrar el comando de paro y el valor 1, como valor de activación de esa señal.

Listado 5.2: Envío de comando de paro al autómata

```
private void EnviarComandoPLC(int direccionComando, int valorComando)
        {
            PLCOmron plc = App.Coms.PLC[0]; //El sistema solo tiene un PLC

            plc.EscribirBCD16(PLCOmron.EnumArea.DM, 0, direccioncomando, new int[] {
                valorComando }); //Se utiliza la biblioteca para escribir un bit en la dirección del
                 comando.
        }

EnviarComandoPLC(1151,1);
```

5.2. Lectura y envío de parámetros

La posibilidad de cambiar los parámetros de ajuste de la máquina era uno de los requisitos del cliente al inicio del proyecto ya que esto le permitía cambiar tamaños de envase, peso y ajustes en las cintas transportadoras, entre otros parámetros.

Todos los parámetros a ajustar se encuentran almacenados en el *PLC* ya que será el elemento ejecutor de las acciones con los parámetros programados. Es por esto que este apartado se expone inmediatamente después del apartado 5.1 en el que se explica la implementación de la comunicación con el autómata, ya que cada modificación en los parámetros deberá ser transferida al autómata.

El envío de parámetros se gestiona desde el formulario `frmParams`, con el cual se realizará la actualización de parámetros.

Es importante mencionar que como requisito de control de calidad interno, se requiere que cada parámetro introducido sea inmediatamente transferido al autómata, sin necesidad de botones de guardado o aceptación.

Para realizar la configuración de los parámetros en la máquina se define un proceso con tres fases. En primer lugar será necesario recuperar los valores de los parámetros actualmente en uso, en segundo lugar será necesario visualizar los valores en el formato correcto, para permitir su modificación por parte del operario, en tercer lugar, se procederá al envío de los valores actualizados al autómata.

En primer lugar y desde la ventana principal de la aplicación se abrirá el panel de gestión de parámetros. Este panel se abrirá invocando a la función `ShowDialog()` estándar del sistema,

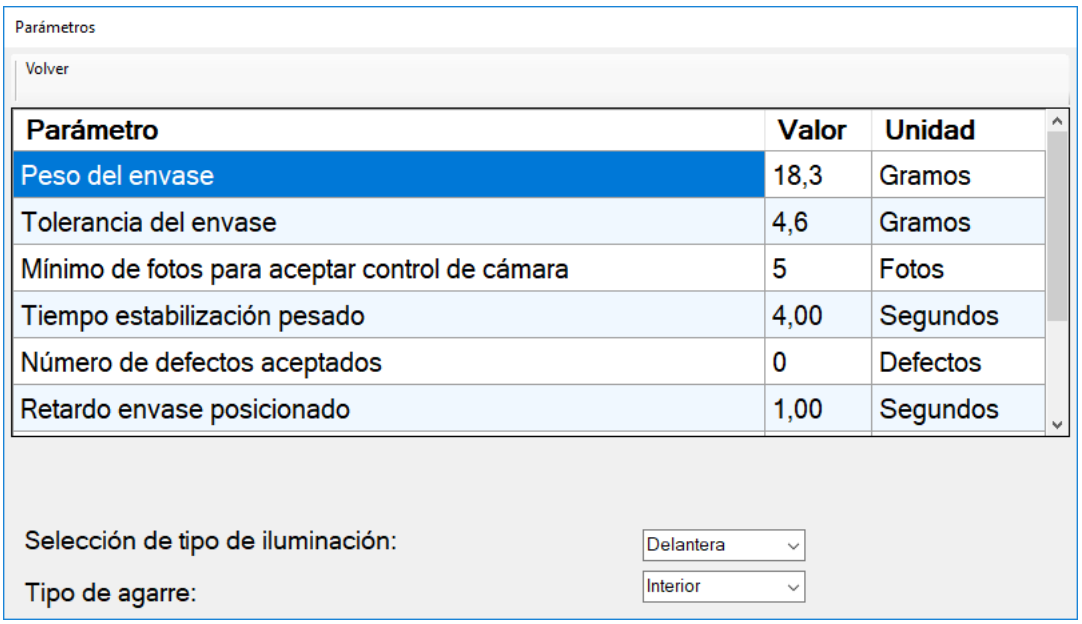


Figura 5.3: Formulario de gestión de parámetros

tras haber realizado la creación del objeto `frmParams` como se ve en el listado 5.3.

Listado 5.3: Apertura de diálogo de gestión de parámetros

```
private void pARÁMETROSToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (ControlUsuarios.CheckPermiso("SupervisorAlcion", "")) //Comprobación permisos
                de usuario
            {
                frmParams parame = new frmParams(); //Creación del formulario

                if (!_frmOpen) //comprobación de que no hay ningún cuadro de diálogo abierto
                {
                    parame.FormClosed += otherFormClosing; //Se asigna un manejador de
                        eventos para controlar cuando el diálogo se cierre poder devolver la
                        variable "_frmOpen" a false
                    _frmOpen = true; //Marcado de que hay un formulario abierto
                    parame.TopMost = true; //Forzar que el formulario de gestión de parámetros
                         esté como ventana superior y por tanto visible
                    parame.Show(); //Llamada estándar del sistema para la apertura de cuadros
                        de diálogo
                }
                else //ya hay un cuadro de diálogo abierto, no se abrirá otro para evitar
                    corrupción de datos
                {
                    parame.TopMost = false;
                    Mensaje.MostrarAviso("Hay otro cuadro de diálogo abierto, cierrelo antes
                        de poder abrir otros");
                }

            }
        }
```

En ese momento el formulario `frmParams` inicia su ejecución realizando una primera lectura de los datos almacenados en el *PLC* y dándoles el formato adecuado para su visualización y posterior modificación si es necesario. El listado 5.4 muestra cómo se realiza la lectura de los parámetros. Por motivos de extensión, no se añade el método completo ya que la lectura de los dos primeros parámetros ilustra suficientemente el propósito del método `rellenaFilas`.

Listado 5.4: Lectura y rellenado de filas de los parámetros

```
public enum direccionesParametros //Enumerador para facilitar los accesos a las direcciones de
    memoria del PLC
   {
       PesoEnvase=11010,
       ToleranciaEnvase=11012,
       MinimoFotosAceptar=11020,
       TiempoEnvaseSeleccionDestino=11021,
       DefectosAceptados=11023,
       RetardoEnvasePosicionado=11413,
       RetardoMarchaCinta=11414,
       TiempoEstabilidadBascula=11415,
       PesoEstabilidadBascula=11500,
       TipoIluminacion=11355,
       TipoAgarre = 11025

   }


private void rellenaFilas(object sender, EventArgs e)
        {
            double dato = 0.0;
            //Tolerancia del envase
            DireccionMemoriaPLCOmron direccion = new DireccionMemoriaPLCOmron(0,
                PLCOmron.EnumArea.DM, (int)direccionesParametros.ToleranciaEnvase); //
                Definición de la dirección de memoria en la que se encuentra el dato a recuperar.
            LecturaPLCOmron lect = new LecturaPLCOmron(0,direccion); //Definición del
                objeto lecturaPLC para realizar la lectura
            DatoPLCOmron dat = new DatoPLCOmron(0, direccion, Dispositivo.EnumTipoDato
                .REAL32); //Inicialización del objeto donde se almacenará el dato leido
            lect.NuevoDato(dat); //Asignación del dat en el que se almacenará el resultado de la
                lectura lect
            lect.Leer(); //función de lectura
            double.TryParse(dat.Valor.ToString(), out dato); //Función para dar formato al dato
                recibido
            dato = Math.Round(dato, 1); //Redondeo a un decimal
            grdParametros.Rows.Add("Tolerancia del envase", "Gramos", dato.ToString("#,#0.0
                "), (int)direccionesParametros.ToleranciaEnvase); //Función estándar del sistema
                 para añadir una fila a la rejilla de parámetros con su formato 0.00.

            //Tolerancia mínimo fotos aceptar control cámara
            direccion = new DireccionMemoriaPLCOmron(0, PLCOmron.EnumArea.DM, (int)
                direccionesParametros.MinimoFotosAceptar);
            lect = new LecturaPLCOmron(0, direccion);
            dat = new DatoPLCOmron(0, direccion, Dispositivo.EnumTipoDato.INT16);
            lect.NuevoDato(dat);
            lect.Leer();
            grdParametros.Rows.Add("Minimo de fotos para aceptar control de cámara", "Fotos
```

```
            ", dat.Valor, (int)direccionesParametros.MinimoFotosAceptar);
}
```

Como se puede apreciar en el código mostrado en el listado 5.4, todas las filas de la rejilla de parámetros tienen una columna invisible correspondiente a la dirección del dato, que se utilizará en la función de modificación del dato, función que se presenta en el listado 5.5.

En primer lugar, cabe destacar que para conseguir la función de actualización directa sin necesidad de botones de aceptación, se enlaza el evento `CellEndEdit` estándar del sistema, al método `grdParametros_CellBeginEdit`. Este método será llamado por el sistema cada vez que el usuario termine de realizar una modificación en una fila.

En consecuencia, este método realiza la comprobación del tipo de datos, la transformación al tipo de datos necesario y su escritura.

Listado 5.5: Comprobación del tipo de dato y escritura en el autómata

```
private void grdParametros_CellEndEdit(object sender, DataGridViewCellEventArgs e)
{
        int changedDir = (int)grdParametros.Rows[e.RowIndex].Cells[3].Value; //recuperación de
            la dirección del dato editado desde la columna invisible
            double changedValue = 0.00; //inicialización del objeto en el que se almacenará el
                dato cambiado
            string valorEnCadena = (string)grdParametros.Rows[e.RowIndex].Cells[2].Value; //
                inicialización del dato que se guardará en formato texto
            if (valorEnCadena.Contains(".")) //Prevenir puntos en lugar de comas para tener los
                valores coherentes
            {
                valorEnCadena=valorEnCadena.Replace(".", ",");
            }
            double.TryParse(valorEnCadena, out changedValue); //conversión de cadena a
                flotante y almacenamiento en el objeto changedValue

            DireccionMemoriaPLCOmron direccion = new DireccionMemoriaPLCOmron(0,
                PLCOmron.EnumArea.DM, changedDir); // Definición de la dirección de
                memoria donde se encuentra el datoa escribir
            DatoPLCOmron dat = null; //inicialización en vacío del dato a escribir


            switch (changedDir) //filtrado de los datos según su driección
            {
                case 11010:
                case 11012:
                case 11500:
                    dat = new DatoPLCOmron(0, direccion, Dispositivo.EnumTipoDato.REAL32
                        );
                    grdParametros.Rows[e.RowIndex].Cells[2].Value = changedValue.ToString("
                        #,#0.0");
                    break;
                case 11020:
                case 11023:
                    dat = new DatoPLCOmron(0, direccion, Dispositivo.EnumTipoDato.INT16);
                    string datoRec = (string)grdParametros.Rows[e.RowIndex].Cells[2].Value;
                    if (datoRec.Contains(","))
```

```
            {
                Mensajes.Mensaje.MostrarAviso("Este parámetro es entero, se truncará
                    el dato.");
            }
            int dato = (int)changedValue;
            changedValue = dato;
            grdParametros.Rows[e.RowIndex].Cells[2].Value = dato;
            break;
        case 11021:
        case 11413:
        case 11414:
        case 11415:
            dat = new DatoPLCOmron(0, direccion, Dispositivo.EnumTipoDato.INT16);
            grdParametros.Rows[e.RowIndex].Cells[2].Value = changedValue.ToString("
                #,##0.00");
            changedValue = changedValue * 100;
            break;
        default:
            dat = new DatoPLCOmron(0, direccion, Dispositivo.EnumTipoDato.INT16);
            break;
    }


    dat.Valor = changedValue;
    dat.Escribir(); //función de escritura en el plc
    }
```

Como se puede ver en el código, el sistema filtra los datos según su dirección de memoria, ya que en diferentes direcciones de memoria se encuentran diferentes tipos de datos (enteros, flotantes, etc) y habrá que aplicar diferentes tipos de transformaciones o comprobaciones según su tipo de datos.

5.3. Ventana de visualización

Enfocado al mantenimiento de los técnicos de *Eurolettra Ingeniería*, se requiere una ventana de visualización de parámetros en la que la aplicación de control muestra el estado de todos los actuadores y demás elementos del sistema.

La ventana de visualización hace las veces de visor de estado de todos los elementos físicos y actuadores del sistema. Esto permite a los técnicos de *Euroelettra Ingeniería*, la rápida detección de errores o problemas en el hardware del equipo.

Su implementación está basada en el sistema de ocultación y mostrado de controles descrito el apartado 5.1, es por eso que no se utiliza código adicional para manejar los cambios. En su lugar en tiempo de diseño, mediante el diseñador de interfaces de *Windows Forms*, se añaden dos controles por elemento a visualizar. El primer control, será el encargado de mostrarse en caso de que el dato a visualizar esté apagado; estará posicionado en la parte inferior en la pila de elementos. El segundo, será el encargado de mostrarse en caso de que éste esté encendido, quedando este elemento en la parte frontal de la pila de elementos.

Siguiendo esta estructura, simplemente ocultando o mostrando el segundo control, encargado de visualizar el estado encendido, se consigue un efecto de etiquetas *ON / OFF* como se puede apreciar en la figura 5.4.

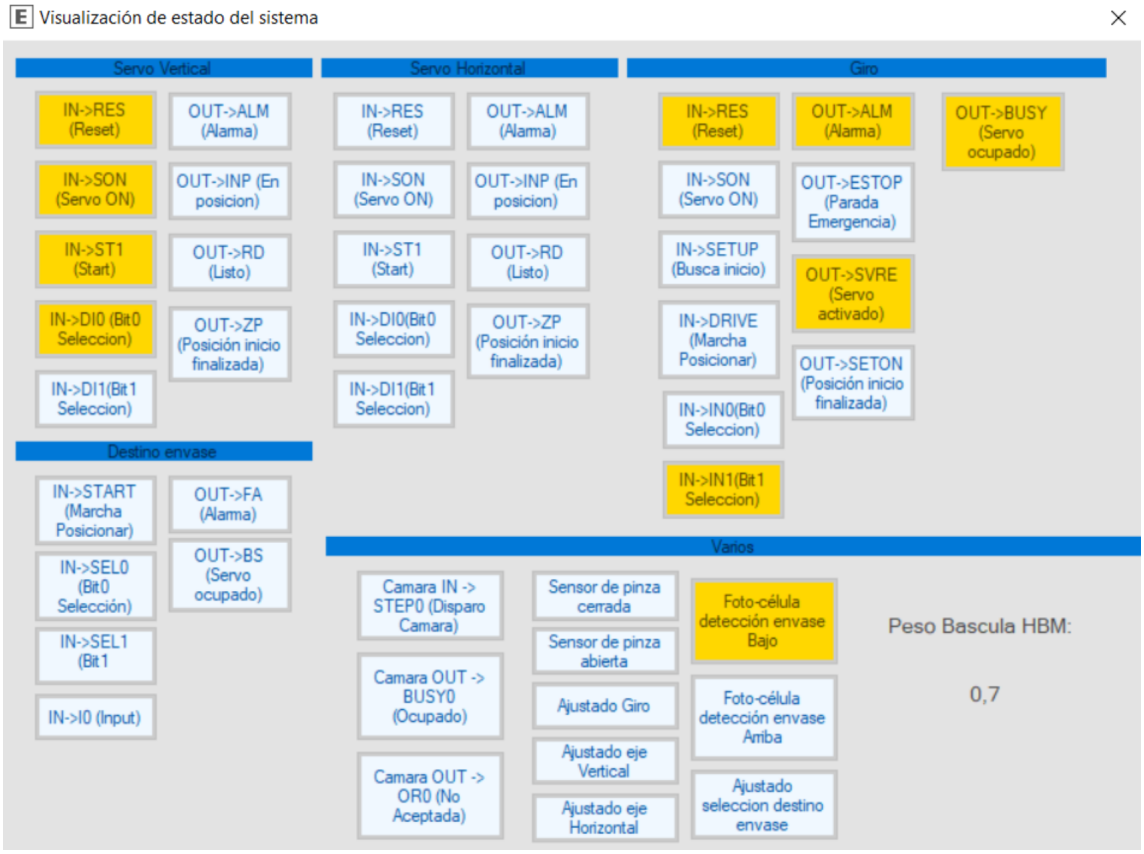


Figura 5.4: Ventana de visualización de estado

5.4. Apagado remoto y arranque automático

El cliente requiere para el equipo de producción unas altas medidas de seguridad, para evitar que los operarios puedan realizar modificaciones en los parámetros o funcionamiento, sin el consentimiento de un supervisor, pero sean capaces de llevar a cabo las operaciones básicas de funcionamiento de la máquina. Es por ello que las siguientes medidas se implementan:

- Un sistema de gestión de usuarios, analizado en el apartado 5.6.
- La eliminación de los periféricos de entrada, tales como ratón y teclado del cuadro de mandos, que se sustituyen por una botonera de funciones.
- La restricción del entorno de escritorio de *Windows*.

Arranque automático

Dadas estas restricciones, es necesario implementar formas para arrancar la aplicación tras un corte de tensión y formas para apagar el PC de control, de forma controlada antes de quitar la corriente a la máquina.

Para estas dos funciones, se optó por implementar *scripts* en lenguaje *batch*, ya que permiten lanzar comandos de sistema de forma ordenada y simple. En el listado 5.6, se expone el código necesario para realizar el arranque de la aplicación de forma retardada. Esta función es necesaria para permitir que tanto el autómata como la cámara de visión artificial arranquen antes que la aplicación de control y no se generen conflictos de comunicación.

Listado 5.6: Inicio retrasado de la aplicación de control

```
@@echo off
echo ********Iniciando servicios***************
timeout 55
start C:\Euroelettra\Alcion\Alcion.exe
exit
```

Apagado automático

Para implementar el apagado automático, el autómata se equipó con un pulsador de apagado, que activa una dirección de memoria al ser pulsado. La aplicación de control lee de forma continua mediante *polling* esta dirección de memoria y lanza el script de apagado cuando se activa. El listado 5.7, muestra las líneas que realizan en apagado controlado del PC tras haber detectado la señal desde el autómata.

Listado 5.7: Script de apagado

```
@@echo off
echo ********Apagando, por favor espere***************
shutdown.exe /f
```

5.5. Teleasistencia

El sistema de teleasistencia, permite a los técnicos de *Euroelettra Ingeniería* resolver incidencias o realizar modificaciones en el sistema, sin necesidad de desplazarse al lugar de trabajo de la máquina. Además no solo se requiere acceso remoto al PC, también a los otros elementos que conforman el sistema, como son el autómata y el equipo de visión artificial. Por otra parte la seguridad de la conexión debe garantizarse, sin que esto afecte al rendimiento y confiabilidad del sistema de teleasistencia. El cliente también requiere tener control total sobre la posibilidad de acceder a la teleasistencia, por tanto se debe implementar una interfaz que permita activar o desactivar el servicio. La interfaz también será necesaria para mostrar la dirección IP pública a la que los técnicos deberán conectarse para realizar la asistencia.

Arquitectura de red

Para tal fin, se diseña una red *Ethernet* local que interconectará a los tres equipos, utilizando el PC como *puerta de enlace* para la salida a internet.

En la figura 5.5, se puede apreciar la arquitectura de red del sistema, con el PC equipado con dos tarjetas de red, la primera dedicada a la conexión con la red local y la segunda utilizada para la conexión de salida a internet proporcionada por el cliente.

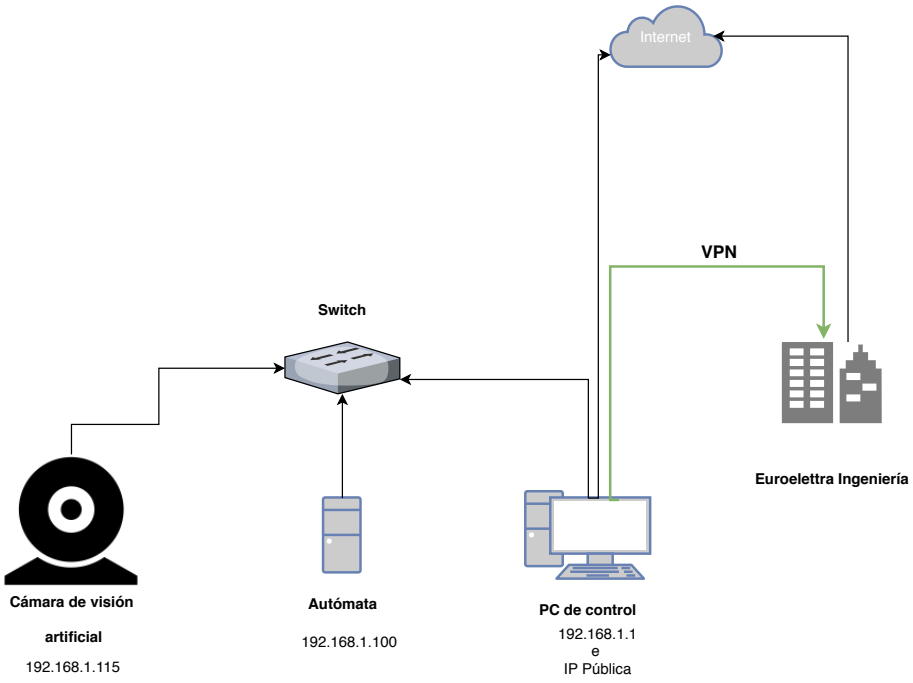


Figura 5.5: Arquitectura de red del sistema

Se configuran los tres equipos en la misma subred local *192.168.1.0/24*, de forma que se puedan comunicar entre ellos.

Servidor VPN PPTP

Para conseguir la funcionalidad de servidor VPN, sin hardware adicional, se emplea el servicio llamado *conexión entrante* que proporciona el sistema operativo *Windows 7* [7]. Dicho sistema crea un servidor VPN PPTP, utilizando como credenciales de autenticación las cuentas de usuario ya existentes en el sistema y asigna direcciones IP dentro de la subred local explicada el apartado 5.5.

Interfaz de usuario

Con la arquitectura mostrada en la figura 5.5, resta implementar una interfaz de usuario, que permita a los operarios activar el sistema de teleasistencia y facilitar la dirección IP a la

que los técnicos deberán conectarse.

El listado 5.8 muestra el código con el que la aplicación consigue la IP pública del equipo.

Listado 5.8: Obtención de IP pública

```
public static string getPubIP()
        {
            try
            {
                string pubip = new WebClient().DownloadString("http://icanhazip.com");
                if (!ip.Equals(pubip))
                    ip = pubip;
                return pubip;
            }
            catch (Exception e)
            {

                Mensajes.Mensaje.MostrarError("No hay conexión a internet!");
            }
            return "--";
        }
```

Para garantizar la seguridad de la máquina, se deshabilita por defecto el servicio del sistema operativo, encargado de gestionar las conexiones VPN llamado `RemoteAccess`

El código 5.9 muestra los dos métodos utilizados para activar y desactivar el servidor VPN.

Listado 5.9: Métodos de habilitación y deshabilitación de la teleasistencia

```
        private void enableAsist()
        {

            ServiceController sc = new ServiceController("RemoteAccess");
            switch (sc.Status)
            {
                case ServiceControllerStatus.Running:
                    _estado = true;
                    acctionButton.Text = "Deshabilitar Teleasistencia";
                    return;
                case ServiceControllerStatus.Stopped:
                    try
                    {

                        sc.Start();
                        sc.WaitForStatus(ServiceControllerStatus.Running,new TimeSpan(0,0,20)
                             );
                        acctionButton.Text = "Deshabilitar Teleasistencia";
                        _estado = true;
                    }
                    catch (Exception)
                    {
                        Mensajes.Mensaje.MostrarError("No se pudo iniciar el servicio de
                            acceso remoto, contacte con asistencia");
                        _estado = false;
```

```
                acctionButton.Text = "Habilitar Teleasistencia";
            }

            break;
        default:
            Mensajes.Mensaje.MostrarError("Error iniciando el servicio de acceso
                remoto!");
            return;

    }
    svcStatus.Text = sc.Status.ToString();

}
private void disableAsist()
{
    _estado = false;
    ServiceController sc = new ServiceController("RemoteAccess");
    switch (sc.Status)
    {
        case ServiceControllerStatus.Stopped:
            _estado = false;
            acctionButton.Text = "Habilitar Teleasistencia";
            return;
        case ServiceControllerStatus.Running:
            try
            {

                sc.Stop();
                sc.WaitForStatus(ServiceControllerStatus.Stopped, new TimeSpan(0, 0,
                    20));
                acctionButton.Text = "Habilitar Teleasistencia";
            }
            catch (Exception)
            {
                Mensajes.Mensaje.MostrarError("No se pudo pararar el servicio de
                    acceso remoto, contacte con asistencia");
                _estado = false;
                acctionButton.Text = "Deshabilitar Teleasistencia";
            }

            break;
        default:
            Mensajes.Mensaje.MostrarError("Error parando el servicio de acceso remoto
                !");
            return;

    }
    svcStatus.Text = sc.Status.ToString();
}
```

5.6. Gestión de usuarios

La gestión de usuarios basa su funcionamiento en la base de datos `Permisos`, que almacenará los usuarios registrados junto con sus contraseñas y sus permisos. La tabla `Accesos`, permite definir diferentes niveles de acceso, que posteriormente mediante la tabla `Permisos`, se relacionarán con los nombres de usuario, para asignar diferentes permisos a diferentes usuarios.

Para realizar el inicio de sesión, se invoca al método `LogIn` de la clase `ControlUsuarios`, con los parámetros nombre de usuario y contraseña, como se muestra en el listado 5.10. Este método devolverá un booleano indicando si se ha llevado a cabo la autenticación de forma satisfactoria.

Listado 5.10: Inicio de sesión como administrador

```
ControlUsuarios.LogIn("supervisor", "supervisor");
```

Durante la ejecución de la aplicación, la clase estática `ControlUsuarios`, almacena el nombre de usuario que ha iniciado la sesión y permitirá mediante el método `CheckPermisos`, realizar comprobaciones sobre si el usuario que tiene la sesión iniciada tiene permisos para ejecutar la acción con su nivel de acceso.

El método `CheckPermisos`, devuelve un valor booleano, indicando si el usuario tiene o no permisos para realizar esa acción.

En el listado 5.11 se expone la forma de verificar si el usuario tiene permisos para abrir el cuadro de diálogo de parámetros, que requiere nivel *supervisor*, antes de realizarla.

Listado 5.11: Ejemplo de comprobación de permisos

```
if (ControlUsuarios.CheckPermiso("SupervisorAlcion", ""))
            {
                frmParams parame = new frmParams();

                if (!_frmOpen)
                {
                    parame.FormClosed += otherFormClosing;
                    _frmOpen = true;
                    parame.TopMost = true;
                    parame.Show();
                }
```

5.7. Pruebas

Tras finalizar la implementación de la aplicación y para comprobar el correcto funcionamiento de todas las características, se llevaron a cabo una serie de pruebas tanto en el taller de la empresa, como en la planta de producción del cliente. La verificación del producto se realizó en dos fases. En primer lugar, las pruebas en el taller en la que se verificó que la comunicación con los distintos elementos era correcta. En una segunda fase de pruebas, en la planta del cliente,

el producto fue probado con envases y condiciones de producción reales. Durante el desarrollo del proyecto se fueron realizando pruebas de verificación, para constatar que los métodos y el código desarrollado eran correctos. Así que se podría decir que gran parte del proyecto ha sido probado mediante pruebas unitarias.

Pruebas en el taller

Estas pruebas se realizaron en conjunto con los compañeros de programación de autómatas, quienes conectados al PLC, verificaban que las lecturas y escrituras que proporcionaba la aplicación coincidían con los valores almacenados en el autómata. Realizando acciones de uso normal de la máquina, como acciones de paro, marcha y cambio de parámetros, se verificaba que los datos y señales llegasen al autómata de forma correcta.

Otra parte de la verificación en el taller fue la conectividad VPN, que se realizó utilizando un modem USB, para proporcionar la salida a internet necesaria al PC de control. Posteriormente desde uno de los equipos conectados a la red corporativa de *Euroelettra Ingeniería*, se establecieron varias conexiones al servidor VPN explicado en el apartado 5.5.

Para finalizar las pruebas internas, se pidió al cliente que facilitara muestras de envases similares a los que se iban a utilizar en producción y se repitieron las pruebas anteriores incluyendo ya los envases y su análisis.

Pruebas en la planta de producción

Una vez realizado el montaje e instalación de la máquina en la planta de producción del cliente, se realizaron de nuevo las pruebas de comunicación entre los componentes. Encontrando en este punto ciertos problemas de comunicación, debidos a un error en el montaje que fue subsanado rápidamente tras ser detectado.

Para finalizar se realizaron varias pruebas de conexión remota desde las instalaciones de la empresa, para verificar el correcto funcionamiento del servidor de teleasistencia.

La fase de pruebas quedó finalizada tras un periodo de dos días en el que se mantuvo a la máquina funcionando sin parar, para verificar su estabilidad.

Capítulo 6

Conclusión

Los objetivos definidos en la propuesta técnica han sido satisfechos ya que la aplicación de control junto con la máquina completa, se encuentran funcionando en la planta de producción del cliente que la encargó. A excepción del apartado de *configuración del sistema de visión artificial*, que al venir dado por una empresa externa no fue necesario. Se considera, por tanto, que todos los puntos expuestos en la propuesta técnica se han satisfecho.

Durante el desarrollo del proyecto se ha podido conocer de primera mano el mundo de la automatización industrial, abordando temas como la comunicación con autómatas *Omron* [2]. Por otra parte, el desarrollo se ha realizado en *Windows*, un sistema operativo sobre el que tenía un gran desconocimiento en la parte de desarrollo ya que ni en la etapa de enseñanza universitaria ni de aprendizaje autodidacta, lo había abordado.

Este desconocimiento del sistema y sus marcos de trabajo, ha propiciado que la velocidad de desarrollo haya sido menor y que las dificultades se hayan visto aumentadas, llegando a encontrar en ocasiones problemas para avanzar en el proyecto. También tuve que gestionar durante el desarrollo del proyecto, problemas con las actualizaciones y compatibilidad de Windows que en ocasiones me llevaban a perder un día de trabajo.

El completo desconocimiento del mundo de los autómatas programables y su funcionamiento también hizo mella en mi productividad ya que en muchas ocasiones cometía errores debido a la inexperiencia y dificultad de los mismos. Sin embargo el resultado final deja claro que se han solucionado los problemas y se ha conseguido un buen producto.

La redacción de esta memoria técnica también ha presentado retos para mí, ya que no estoy acostumbrado a trabajar en LaTex y he consumido mucho tiempo solucionando errores y detalles que posiblemente con otro editor ya conocido no hubieran aparecido. Finalmente y gracias a la ayuda del tutor, he conseguido utilizar de forma competente el sistema, lo cual ha hecho que me guste y esté pensando en proponer a la empresa implantarlo como nuestro formato para los manuales y documentación.

Por otra parte, es pertinente valorar la experiencia en la empresa *Euroelettra Ingeniería S.L*, en la que se ha creado el proyecto. No tengo más que palabras de agradecimiento para los com-

pañeros que han sido extremadamente comprensivos, ayudando en todo momento y facilitando el aprendizaje. El estilo de trabajo de la empresa fomenta el compañerismo, encontrando ocasiones en las que varios compañeros se preocupaban por solucionar problemas o dudas surgidas, incluso sin ser este su proyecto.

Me gustaría remarcar también mi aportación a la metodología de trabajo de la empresa, la cual no incluía un sistema de control de versiones de código a mi llegada. Tras unos meses de trabajo allí, implanté un servidor Git que en la actualidad está siendo utilizado por mis compañeros.

Como nota negativa, indicar que la fuerte dependencia de la empresa en productos de Microsoft, hace que todos los sistemas y productos de software que se desarrollan, deban estar atados a esta plataforma. Personalmente opino que no es el mejor sistema operativo para ejecutar programas que gestionen procesos críticos, dada su filosofía privativa y su filosofía de licencias restrictivas, además de por su conocida inestabilidad e inseguridad. No parece que la empresa esté dispuesta a cambiar de dirección en este aspecto y opino que es una mala decisión.

Bibliografía

[1] International Organization for Standarization. Normativa ISO 9001 para el control de calidad interno. https://www.iso.org/iso-9001-quality-management.html. Fecha de último acceso: 24-5-2018.

[2] Omron industrial. Biblioteca de comunicaciones con autómatas programábles Omron. https://industrial.omron.us/en/media/Sysmac_NJ_FINS_TechnicalGuide_en_201205_W518-E1-01_tcm849-109257.pdf. Fecha de último acceso: 22-5-2018.

[3] Microsoft. Plataforma de desarrollo para sistemas de Microsoft. https://www.microsoft.com/net/. Fecha de último acceso: 22-5-2018.

[4] Microsoft. Lenguaje de scripting para sistemas DOS. https://msdn.microsoft.com/en-us/library/cc722477.aspx. Fecha de último acceso: 23-5-2018.

[5] Microsoft. Sistema gestor de bases de datos de Microsoft. https://www.microsoft.com/es-es/download/details.aspx?id=42299. Fecha de último acceso: 20-5-2018.

[6] Agencia tributaria española. Tabla de coeficientes de amortización. https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml. Fecha de último acceso: 7-6-2018.

[7] Microsoft. Documentación de conexiones entrantes en Windows 7 para la creación de un servidor VPN PPTP. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc753616(v=ws.10). Fecha de último acceso: 24-5-2018.

Apéndice A

Manual de usuario de la aplicación

Se reproduce a continuación el manual de usuario redactado tras la finalización del proyecto y que se entregó al cliente como parte de la documentación del sistema de clasificación por visión artificial.

Control de calidad de botellas cilíndricas

ALCION

Manual del Usuario

1. PRÓLOGO

Este manual presenta una explicación completa de todas las funciones del programa de Control de calidad de botellas **cilíndricas.**

Este programa está desarrollado en el entorno de programación Windows con el fin de ofrecer un programa fácil e intuitivo incluso a los usuarios menos expertos gracias a su desarrollo totalmente gráfico, presentado en diversas ventanas de trabajo según las funciones a desempeñar.

Se presenta una única pantalla principal en la que se muestra el estado del sistema así como los controles necesarios para el funcionamiento del sistema.

Se dispone además de una serie de menús contextuales para editar parámetros o activar la Teleasistencia (En caso de estar disponible).

2. CARACTERÍSTICAS GENERALES

El programa está realizado con herramientas de desarrollo muy difundidas en el mercado mundial como son **Microsoft Visual Studio y Microsoft SQL Server**, funcionando bajo el entorno **Microsoft Windows**, lo cual les permite ser completamente compatibles con cualquier aplicación de **Microsoft**[1] y altamente conectables a sistemas informáticos del cliente para que, a través de una conexión en red poder realizar la supervisión y programación de la instalación desde terminales remotos.

Este tipo de programas puede conectarse con los PLC mas difundidos en el mercado. Además pueden comunicarse con varios reguladores de control.

Los comandos son fácilmente seleccionados por las teclas de función (F1, F2, etc.) o pulsando la letra subrayada del comando que se desee accionar.

El programa dispone de una ventana en la que se muestran los datos dentro de celdas, para moverse entre las celdas se utilizan las flechas " ↑ ↓ " y las flechas " ← → ".

Para confirmar operaciones o datos se utilizará la tecla Enter " ↵ ".

El programa permite además el uso del Mouse o Ratón, por tanto todos los botones o selecciones pueden realizarse con el ratón pulsando el botón derecho del mismo, lo que denominaremos "hacer clic" en las indicaciones de este manual.

Se entiende por tanto que las denominaciones "pulsar la tecla XX o hacer clic sobre la tecla XX provocan el mismo resultado, dejamos a la **libre elección del usuario** el uso del ratón o del teclado según comodidad o hábitos del mismo.

[1] Microsoft Visual Basic, Microsoft SQL Server y Microsoft Windows son marcas registradas de Microsoft Corporation USA.

3. ÍNDICE

4. DESCRIPCIÓN GENERAL DEL SISTEMA

En su funcionamiento normal el ordenador presenta una imagen similar a la siguiente.

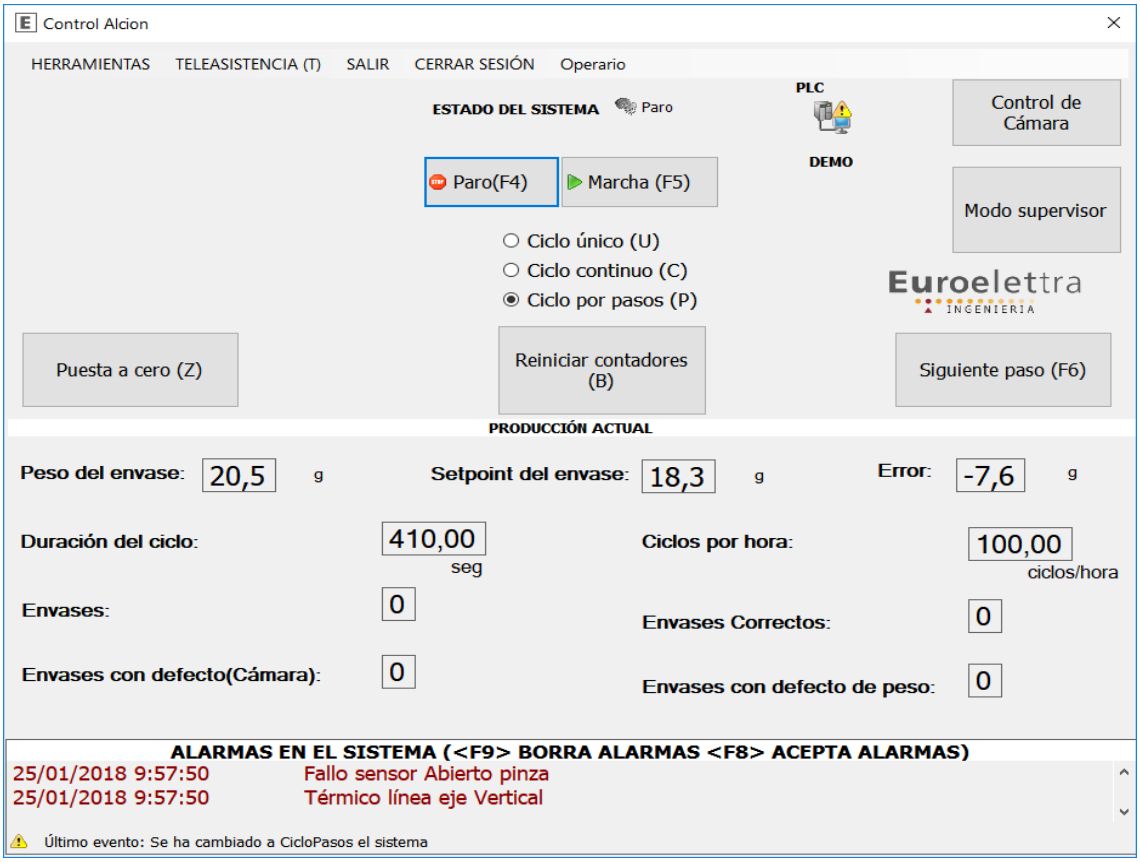


Ilustración 1: Ventana inicial de la aplicación

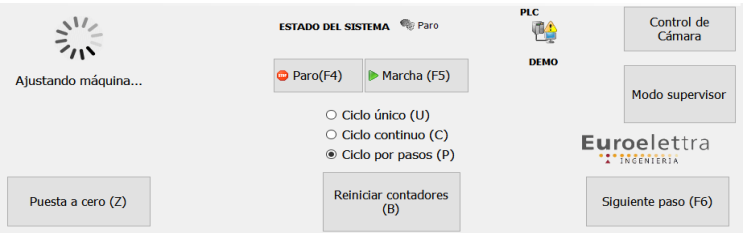
Dividimos la interfaz principal en tres secciones diferenciadas por sus cabeceras:

- Estado del sistema : Que representa el estado y modo de funcionamiento del sistema.
- Producción actual: Que informa de los datos a tiempo real de la producción actual
- Alarmas en el sistema: Que informa de las diferentes alarmas y errores producidos en el sistema.

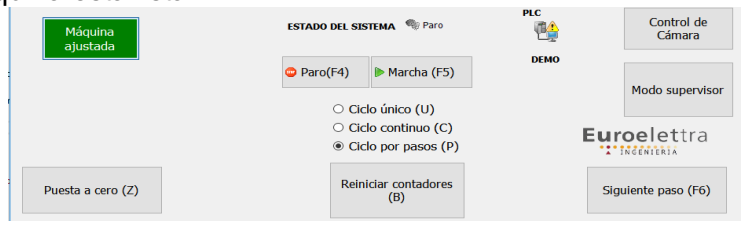
5. ESTADO DEL SISTEMA

La sección de estado del sistema contiene :

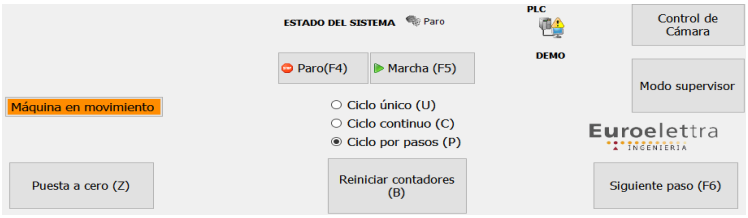
- Indicadores del estado de la máquina:
  - **Ajustando** : La máquina está posicionando todos sus elementos para iniciar el funcionamiento normal



  - **Maquina ajustada:** El posicionamiento de los elementos ha finalizado y la máquina está lista.



  - **Máquina en movimiento:** Durante el funcionamiento normal de la máquina se señaliza cualquier movimiento de la misma mediante el piloto de movimiento.



- Controles del estado de la máquina:
  - **Botón de puesta a cero:** Inicia el proceso de puesta a cero de la máquina, que permitirá iniciar el funcionamiento de la misma.

    NOTA: Tras finalizar el proceso de puesta a cero, la máquina quedará en **PARO.**
  - **Selector de modo de funcionamiento**: El sistema es capaz de ejecutar tres modos de funcionamiento:
    - Ciclo único: El sistema realizará un ciclo de análisis y volverá a su posición inicial
    - Ciclo continuo: El sistema ejecutará ciclos de forma ininterrumpida.
    - Ciclo por pasos: Para propósitos de ajuste del sistema, este modo permite ejecutar el ciclo por pasos. Para ejecutar el siguiente paso de cada ciclo, presione el botón “Siguiente paso” o pulse la barra espaciadora.

NOTA: El botón “Siguiente paso”, desaparece si no se ha seleccionado el modo de ciclo por pasos.

- **Interruptor de paro/marcha**
- **Botón de acceso al control de cámara:** Este botón lanzará la aplicación de control de visión artificial y se conectará al controlador de la cámara
- **Botón de cambio a modo supervisor:** Dado que la aplicación se ejecuta en modo restringido, para ejecutar tareas de mantenimiento, se deberá seleccionar el botón de cambio a modo supervisor.
  - Tras seleccionar el botón, el sistema requerirá la contraseña de **supervisor**
  - Tras la satisfactoria inserción de la contraseña de supervisor, el sistema cierra la sesión restringida y permite acceder a la sesión de administrador.

6. PRODUCCIÓN ACTUAL

En el apartado producción actual, se muestran los diferentes datos referentes a la producción en curso con los siguientes datos:

- Peso del envase: Que refleja el peso del último envase que se ha procesado
- Setpoint del envase: Que refleja el peso que debería tener el envase para ser considerado correcto
- Error: Diferencia entre el peso del envase y el *setpoint*.
  - Nota: Este recuadro aparecerá en rojo o verde, dependiendo de si el error está dentro de los parámetros de tolerancia.
- Duración del ciclo: Medido en segundos, este indicador refleja la duración de un ciclo completo.
- Ciclos por hora: Ciclos que se realizan por hora.
- Envases: Número total de envases que se han procesado
- Envases correctos: Número total de envases correctos
- Envases con defecto (Cámara): Número de envases marcados como incorrectos por defecto visual.
- envases con defecto de peso: Número de envases marcados como incorrectos por defecto de peso

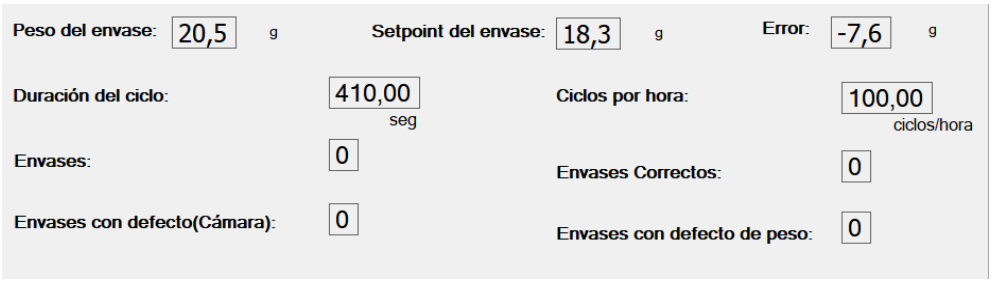


Ilustración 2: Sección producción actual

El botón “Reiniciar contadores”, reinicia los contadores de envases, ciclos por hora, duración de ciclo, envases correctos y envases con defecto.

7. ALARMAS EN EL SISTEMA

En la sección de alarmas en el sistema, se muestran todas las alarmas que el sistema genera a tiempo real.

Para borrar las alarmas, pulse la tecla “F9” y reinicie el funcionamiento de la máquina. Para aceptar las alarmas, pulse la tecla “F8” y la sirena parará.

Ilustración 3: Sección de alarmas en el sistema

Nota: Tras una alarma, la máquina pasa a **estado de paro y se requiere la puesta a cero** y el cambio a estado de marcha, para reiniciar su funcionamiento.

8. SEGURIDAD Y GESTIÓN DE USUARIOS

El sistema está provisto de control de acceso granular a usuarios, pudiendo dividir las zonas accesibles y accionables de la aplicación.
Se dividen tres niveles de acceso:

- **Invitado**: Ningún control está disponible para su accionamiento, las visualizaciones, contadores y estado del sistema, continúan actualizándose.
  - Toda la interfaz aparece deshabilitada.

- **Operario**: Los elementos de control del sistema están disponibles para su accionamiento (Borrado de contadores, cambio de tipo de ciclo, paro,marcha, cero), pero los elementos de parametrización y ajustes avanzados no están disponibles.
  - **La aplicación se inicia POR DEFECTO en este nivel de acceso.**

- **Supervisor**: Todos los elementos tanto de control como de ajustes avanzados están disponibles.

El usuario supervisor, puede crear, eliminar y modificar usuarios a los que se le asignarán los niveles de acceso. A continuación se detalla el proceso de creación de usuarios y asignación de nivel de accesos.
Para abrir el panel de gestión de usuarios, inicie sesión como supervisor y a continuación seleccione "Administrar usuarios" en el menú herramientas o pulse la tecla **R.**

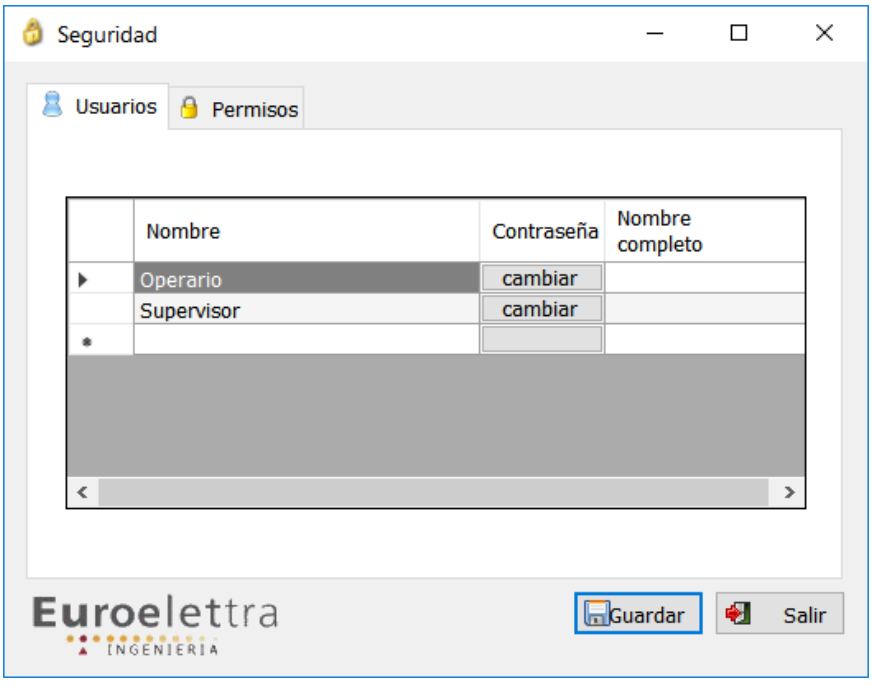


Ilustración 4: Ventana de administración de usuarios

La ventana de gestión de usuarios se divide en dos pestañas:

- Usuarios: Donde se permite la creación y eliminación de usuarios y el cambio de contraseñas de los mismos.
- Permisos: Donde se permite asignar a los usuarios los niveles de acceso.

Para crear un usuario nuevo:

1. Seleccione la fila vacía y escriba el nombre de usuario deseado
2. Seleccione el botón "Cambiar" del campo "contraseña"
3. En una ventana emergente, se le solicita que introduzca dos veces la nueva contraseña deseada.
4. Opcionalmente puede añadir un nombre completo al usuario
5. **Seleccione el botón guardar en la esquina inferior derecha, para hacer efectivos los cambios.**

   Para añadir un nivel de acceso a un usuario:

1. Seleccione la pestaña permisos de la ventana de gestión de usuarios
2. Seleccione el desplegable vacío del campo "Acceso" y elija uno de los dos niveles de acceso.
3. Seleccione el desplegable vacío del campo "Nombre" y seleccione el usuario deseado al que otorgarle permisos.
   - El permiso *AlcionApp* se corresponde con el nivel de acceso de operario
   - El permiso *SupervisorAlcion* se corresponde con el nivel de acceso de supervisor.
   - El supervisor **DEBERÁ** tener asignados ambos permisos para tener acceso completo.
4. **Seleccione el botón guardar en la esquina inferior derecha, para hacer efectivos los cambios.**

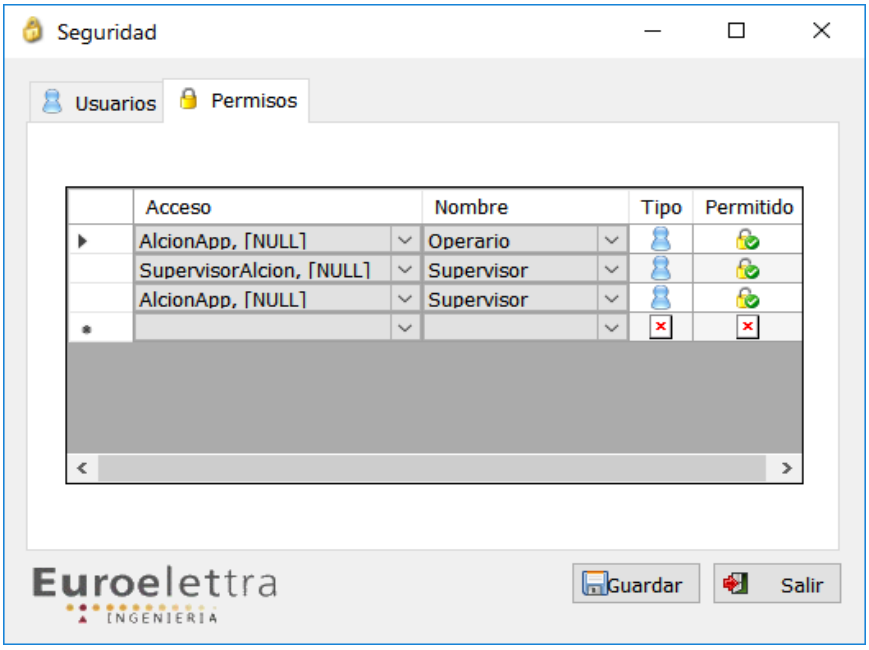


Ilustración 5: Ventana de administración de permisos

Nota: La aplicación se suministra con dos usuarios por defecto

- Usuario S*upervisor* con contraseña *supervisor* → con permisos de acceso de supervisor.
- *Usuario Operario* con contraseña *operario* → Con permisos de acceso de operario.

9. BARRA DE MENÚ

La parte superior de la ventana contiene una barra de menú con las diferentes funciones avanzadas descritas a continuación:

Herramientas

Visualización	**V**
Parámetros	**F3**
Administrar usuarios	**R**
Aceptar alarmas	**F8**
Borrar alarmas	**F9**

Teleasistencia **T**
Salir

9.1. Menú Herramientas

9.1.1. Visualización

El panel de visualización está enfocado a labores de mantenimiento del sistema y muestra el estado de los diferentes equipos que lo componen (actuadores, motores, sensores…)

Para acceder al panel de visualización seleccione el menú "Herramientas" y a continuación "Visualización" o pulse la letra V en el teclado del sistema.

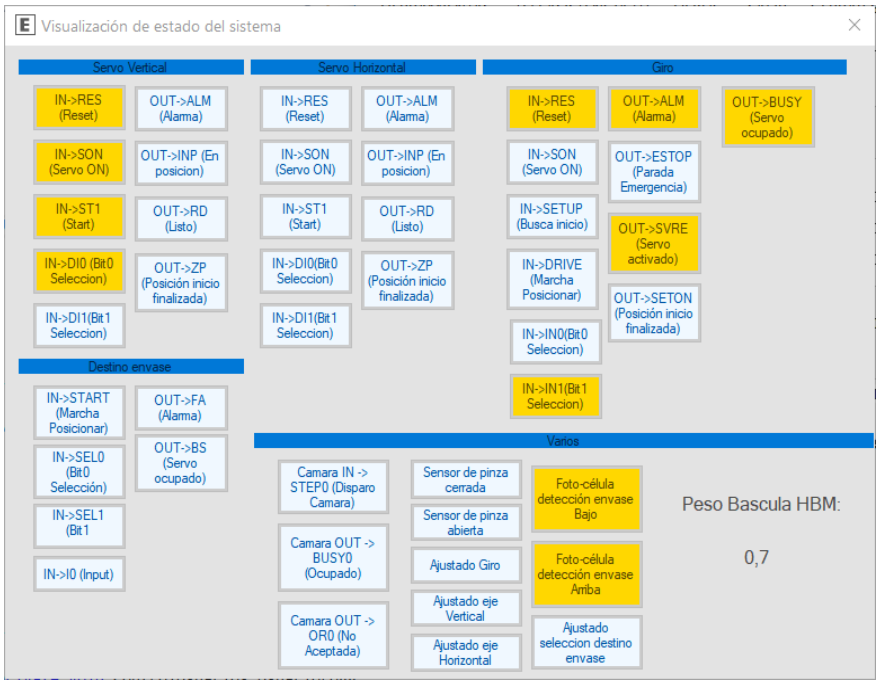


Ilustración 6: Ventana de visualización de estado del sistema

9.1.2. Parámetros

La ventana de parámetros permite ajustar los parámetros de funcionamiento del sistema.

ATENCIÓN: No modifique ninguno de estos parámetros sin conocimiento del mismo, puesto que podría afectar al funcionamiento del dispositivo.

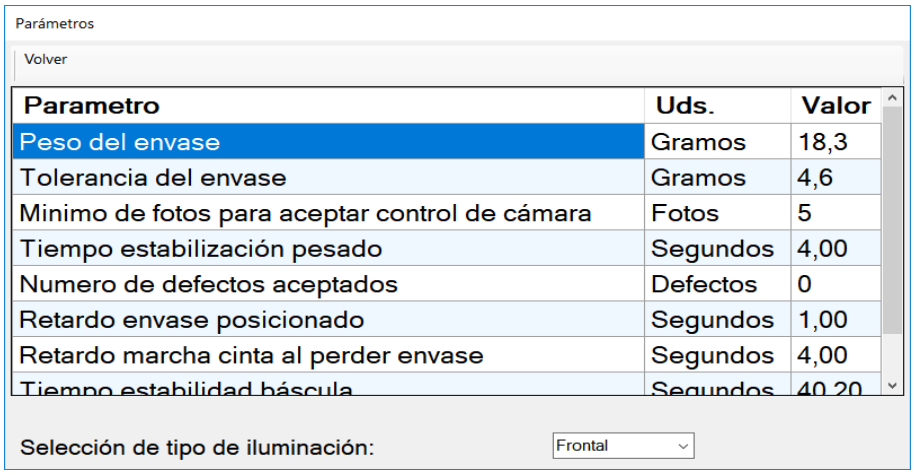


Ilustración 7: Ventana de parámetros

La ventana de parámetros permite ajustar los siguientes parámetros:

- Peso el envase: Peso ideal del envase que se está procesando
- Tolerancia del envase: Tolerancia (+-) en gramos para dar como correcto un envase.
- Mínimo de fotos para aceptar control de cámara: Número de fotos mínimo que debe de tomar la cámara para aceptar un envase.
- Tiempo de estabilización de pesado: Tiempo en segundos, que debe esperar el sensor de peso para estabilizarse.
- Número de defectos aceptados: Número de defectos que el control de cámara acepta.
- Retardo de envase posicionado: Tiempo en segundos que el control de cinta espera para posicionar el envase.
- Retardo de marcha en la cinta al perder un envase: Tiempo en segundos que la cinta espera tras la pérdida de un envase.
- Tiempo estabilidad Báscula: Tiempo en segundos que el control de peso espera para estabilizar el envase al pesarlo.
- Peso estabilidad báscula: Peso en gramos que el control de peso tolerará al realizar el pesado.
- Selección de tipo de iluminación: Selección del tipo de iluminación "backlight" o "Frontlight".

9.2. Menú Teleasistencia

Pulsando la tecla **T** o seleccionando la opción del menú se activa la ventana de Teleasistencia.

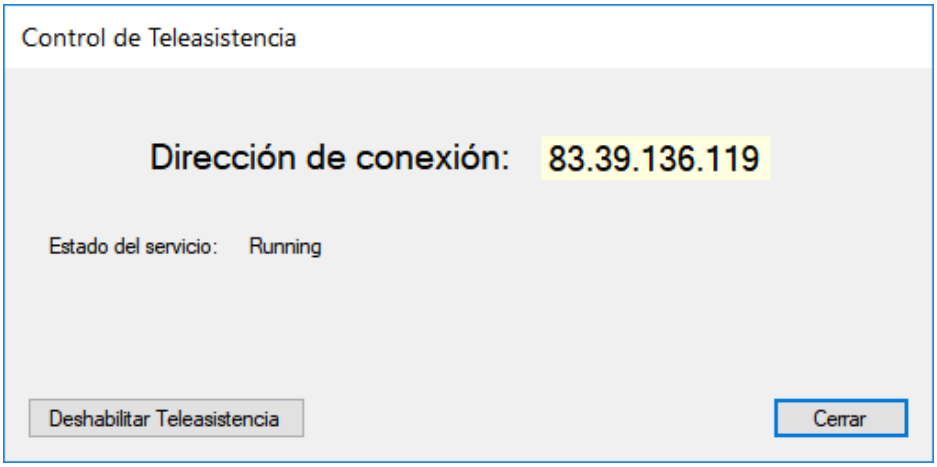


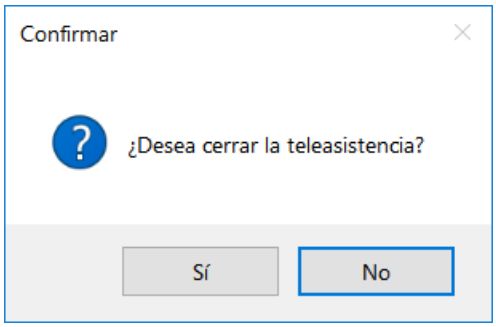
Ilustración 8: Ventana de control de Teleasistencia

Desde esta ventana se controla el estado de la Teleasistencia y se consulta la *dirección de conexión*.

Para garantizar la seguridad del sistema, la Teleasistencia funciona EXCLUSIVAMENTE bajo demanda y si se cumplen los siguientes requisitos:

1. El sistema tiene salida a Internet
2. El operario ha habilitado el servicio de Teleasistencia
3. El operario ha facilitado al asistente la *dirección de conexión*

Al pulsar sobre el botón "Cerrar" se deshabilita la teleasistencia de forma automática.



Un cuadro de diálogo para confirmar la acción aparece antes de deshabilitar la Teleasistencia.

9.3. Menú Salir

Seleccionando el menú salir o bien cerrando la ventana de la aplicación, se muestra el menú salir mostrado a continuación:

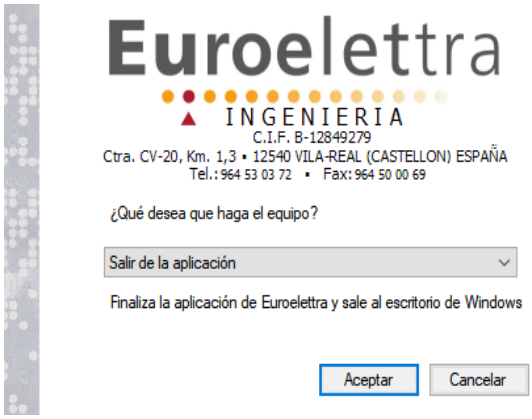


Ilustración 9: Menú Salir

Para salir de a la aplicación, seleccione “Salir de la aplicación” y luego “Aceptar” en el menú salir.

NOTA: Al salir de la aplicación los servicios de **teleasistencia** se deshabilitan.

9.4. Menú de sesión

El menú de gestión de sesión toma dos versiones:

- **Sesión iniciada**
  - En este modo el menú de gestión de sesión muestra un botón para cerrar la sesión y el nombre del usuario actualmente iniciado en el sistema.

- **Usuario invitado**
  - En este modo el menú de gestión de sesión muestra un botón para iniciar sesión.