CREACIÓN Y DESARROLLO DE SOFTWARE PARA CÁMARAS INTELIGENTES EN EL CAMPO DE LA AUTOMÁTICA INDUSTRIAL

INTERTRONIC

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSIDAD POLITECNICA DE VALENCIA

Alumno: Joan Orti Navarro

Tutor: Luis Ignacio Gracia Calandín

Julio 2017

Dedicado a mis padres, porque todo el esfuerzo que invertisteis en mi sirvió para poder apreciar el vuestro. Gracias de corazón.

Tabla de contenido del TFG

INTRODUCCIÓN

1.1 Cámaras inteligentes en la industria

Es un hecho que la automatización en la industria cada vez cuenta con más presencia, llegando a ser un pilar fundamental. Es inconcebible hoy en día pensar en una industria con una alta producción sin apenas automatizar. Líneas de mecanizado, plantas de envasado, de montaje… Son solo unos cuantos ejemplos en los que la automática desempeña un papel fundamental, no solo en el ahorro de tiempo si no en el consecuente ahorro económico.

Dentro de la automatización industrial hay desde los más simples sensores inductivos, capaces de detectar piezas metálicas a cortas distancias, a los robots más sofisticados, como los brazos 6R de ABB o KUKA. No obstante, nuestro objeto de estudio se centrará más en el campo de los sensores, más concretamente en los sensores de visión. Antes se ha hablado de los sensores inductivos como un elemento “base” en la automatización. Básicamente devuelven una salida, 1 o 0, dependiendo de si detectan respectivamente presencia de pieza metálica o no. Los sensores de barrera tienen un comportamiento similar, solo que en este caso proyecta un haz de luz sobre una longitud predeterminada, y en el momento en el que se corta ese haz de luz cambian el valor de su salida de 0 a 1 o al revés. No obstante y como se puede observar, estos sensores solo son capaces de distinguir si una pieza u objeto pasa por delante suyo, marcando con un 1 o un 0 su presencia o ausencia. En el caso de querer ir más allá, es decir, no solo detectar una pieza sino ver su forma, su holografía, sus adhesivos en el caso de que tuviera… Evidentemente, esta información no se puede reducir a unos y ceros. Es por eso que entran en juego las cámaras de visión inteligente.

Véase una cinta de transporte de una industria de patatas fritas en sus etapas finales. A falta del empaquetado y su posterior distribución, hace falta comprobar que todas las bolsas que pasan por esa cinta tengan unas características comunes de acabado y forma, ya sean códigos de barras, tamaño del paquete, forma del logotipo… Puede venir a la cabeza el típico empleado que se encuentra de pie frente a la cinta comprobando manualmente si todas las bolsas están en buenas condiciones para su posterior envasado y venta. Es posible que dicho trabajador haya estado ejerciendo ese puesto toda su vida, agudizando su habilidad de detectar pequeños defectos en los paquetes. No obstante se ha de tener en cuenta varias consideraciones: Los paquetes pueden variar de tamaño, forma, color con bastante frecuencia, según la estrategia de Marketing que adopte la empresa; los defectos pueden ser tan diminutos que sea casi inapreciable para el ojo humano; y por último pero no menos importante, el fallo humano, ya sea por un error involuntario o por un deterioro de la visión. Todos estos hechos no dejan en muy buen lugar al operario aunque este sea muy bueno. Por eso muchas industrias, por no decir la mayoría con grandes tiradas, prefieren adoptar un método más rápido, eficaz y barato a largo plazo. Es aquí donde entran en escena las cámaras inteligentes, capaces de detectar a una alta velocidad los fallos i defectos de los productos de una cinta, en este caso las bolsas de patatas fritas. La salida de estas cámaras se reduce a un 1 o un 0, pero a diferencia de los sensores anteriormente mencionados, esta salida está condicionada a diferentes parámetros y no solo a uno. Es el equivalente al operador lógico AND: Solo se cumple cuando todos los parámetros incluidos son verdaderos. Así pues la salida de la cámara será 1 cuando, por ejemplo, se cumplen los criterios de medida, forma y decoración.

Sin embargo, existe un inconveniente que el operario que vigila la cinta no tiene. Y es que se necesita saber manejar ciertos programas y comandos para poder aprovechar al 100% la capacidad que tiene la cámara. Es por eso que los fabricantes de estas cámaras inteligentes cada vez se empeñan más en crear un software sencillo, intuitivo y de fácil ejecución. Desafortunadamente no siempre se consigue y muchas veces las empresas han de contactar con el fabricante para resolver sus dudas. Este es uno de los temas a tratar en este informe, por lo que se profundizará más adelante. Otro inconveniente de estas cámaras es la amortización de las mismas ya que para tiradas pequeñas, a diferencia de las grandes, se necesitan muchas más horas de trabajo, además de más producto, para amortizarla por completo, lo que hace que este tipo de empresas se replanteen el comprar una. En cuanto al mantenimiento, al ser un sensor bastante caro y complicado, se hace indispensable devolverlo al fabricante o a la empresa especializada para su evaluación y reparación, lo que supone tener parada la maquinaria mucho tiempo sin sacar ningún rendimiento.

A pesar de que pueden parecer desalentadoras las desventajas expuestas, si finalmente se consigue dominar el software necesario es una herramienta que incrementará tanto la calidad como el rendimiento, repercutiendo enormemente en las ventas del mismo. Es más, hoy en día hay muchas empresas que se dedican en parte a la configuración de los trabajos de estas cámaras, como Intertronic, para que el cliente solo tenga que preocuparse de seleccionar un trabajo u otro. Además, también se ofrece soporte técnico en caso de que se produzca un fallo a nivel de software, lo que hace que las empresas se decanten cada vez más por adquirir este tipo de dispositivos.

A modo de resumen, hoy en día existe la posibilidad de automatizar el control de calidad de los productos en la misma industria gracias a las cámaras inteligentes, que pese a que son algo difíciles de manejar sin tener un cierto conocimiento de visión, ahorran tanto tiempo como dinero, en beneficio de aumentar la calidad del producto distribuido.

1.2 Visión artificial

Hecha ya una pequeña introducción del campo de aplicación a tratar, conviene que antes de continuar se haga una pequeña introducción a la llamada visión artificial. La visión artificial es procedimiento de adquisición de imágenes, sin contacto y mediante sistemas ópticos, donde se realiza el análisis automático de las mismas, o dicho de otra manera, es el método para extraer cierta información para controlar un proceso o actividad.

Estos sistemas de visión han ido evolucionando tanto tecnológicamente como en la propia filosofía del sistema de visión, haciendo que hoy en día sean un elemento estándar en el análisis de procesos. Han dado pie a que en los procesos industriales se produzca una inspección continua del 100% de los productos bajo unos criterios constantes en tiempo real, analizando los errores del mismo producto que pueden ir desde el aspecto, hasta la forma y el color.

La visión artificial abarca muchos sectores dentro de la industria como pueden ser la automoción, la alimentación, el sector farmacéutico, la electrónica, la robótica, envase y embalaje… Y cuyos usos se extienden a tareas tales como el transporte o guiado de vehículos, identificación facial, inspección de madera, aplicaciones de tráfico, sistemas 3D y ocio y entretenimiento entre otros.

Si bien hoy en día, tal y como se ha comentado anteriormente, son un elemento estándar en los procesos industriales y una necesidad referente a la calidad del producto y la reducción de costes, no hace mucho eran una apuesta por la innovación y por la calidad en los procesos de fabricación. Si bien ahora no es tanta la innovación a la hora de crear estas cámaras, sí que sigue existiendo a nivel de exigencia, es decir, un tratamiento de datos más veloz. En muchos casos, la visión viene impuesta por el cliente final.

Hace unos años, el usuario de un sistema de visión era un experto en la materia. Tanto los sistemas de visión como sus aplicaciones estaban orientadas a un perfil muy técnico y requería de amplios conocimientos de hardware y del proceso a controlar. Además, las prestaciones que ofrecían los componentes de las cámaras eran bastante limitadas para un coste muy alto, lo que hacía que muchas industrias dudaran de su adquisición. Actualmente, como se mencionaba con anterioridad, los softwares de visión se desarrollan con el fin de que un usuario con simples conocimientos de visión pueda manipular las cámaras a su gusto, dependiendo del proceso a controlar. Esto sumado a que las prestaciones tanto del software como del hardware van en aumento, hace que los sistemas se vayan compactando y reduciendo de precio, haciendo que muchas situaciones que en el pasado resultaran difíciles de resolver por las limitaciones existentes queden resueltas.

Componentes

Antes se ha mencionado el hardware de la cámara y su evolución con el tiempo. Pero, ¿qué es exactamente el hardware? Obviamente, una cámara de visión no solo cuenta con el dispositivo en sí. Este puede estar compuesto por diferentes componentes que pueden ir desde luces, ópticas, cables de conexión, software o accesorios.

Las luces no son ni más ni menos que dispositivos de iluminación que, en el caso de que las cámaras no dispongan de flash, hacen que la lectura sea más nítida para un posterior análisis aumentando el contraste entre el fondo y el objeto o forma a leer. La iluminación no tiene porqué ser solo de color blanco, sino que dependiendo de cada necesidad puede ser de un color u otro, e incluso puede instalarse luz ultravioleta para comprobar marcas invisibles a simple vista.

La óptica, como en toda cámara que se precie, es una lente que proyecta la luz del exterior formando una imagen en el sensor. Principalmente, en el campo de las cámaras de visión artificial existen 2 tipos de ópticas: Las telecéntricas y las pericéntricas. Las ópticas telecéntricas son aquellas que hacen que la imagen, siempre que el objeto se encuentre a una cierta distancia llamada profundidad de campo, permanezca siempre con unas dimensiones invariables. Esto es, que los rayos solo entran en el objetivo siguiendo un recorrido casi paralelo al eje óptico, provocando así el efecto antes comentado. Además, no existen errores de perspectiva y crea muy poca distorsión. Las ópticas pericéntricas, cada vez menos utilizadas, permiten ver una perspectiva lateral desde una visión cenital. Una de las ventajas que presenta este tipo de ópticas es que no es necesaria la utilización de espejos, por lo que se simplifican, además de permitirnos ver las imágenes como si de un gran angular se tratara. Se adelanta que el modelo con el que usualmente trabajaremos, la VS-06 de Di-soric, dispone de una versión con óptica acoplada a la cámara,

aunque no es el caso de todas las cámaras ya que obviamente es más cómodo y sencillo contar con un autoenfoque que tener que regular cada vez los objetivos.

La cámara por supuesto es la columna vertebral de nuestro sistema de visión. Es la encargada de capturar la imagen proyectada en un elemento sensor y transferirla a un sistema electrónico externo como un PC. Hoy en día se invierte cada vez más en la mejora de las cámaras, ya sea aumentando la resolución de la misma o la velocidad de captura, e incluso creando dispositivos capaces de realizar procesos fuera del espectro visible (UV).

Otro apartado muchas veces menospreciado, pero no por ello deja de ser importante, es el cableado. Es común que un comportamiento anómalo de la cámara, véase que el PC no la reconozca o simplemente que no se encienda, se deba a un mal funcionamiento en el cableado. Es por ello que los cables que interconectan nuestra cámara y el PC o autómata deben estar resguardados y protegidos, sobretodo en entornos industriales donde fácilmente pueden dañarse, parando la producción hasta la reparación del problema.

Una de las partes más importantes, sin la cual nada más que se tiene un dispositivo sin capacidad de análisis, es el Software. El software de visión se compone de una serie de herramientas tanto de pre proceso como de proceso que analizan la imagen y extraen información de la misma según los algoritmos en los que estén basadas dichas herramientas. Como avanzábamos, cada vez más se busca un software más intuitivo y fácil de manejar, por lo que cualquiera con un mínimo conocimiento de visión y del software puede abarcar casi cualquier problema. Normalmente el software de edición lo proporciona el fabricante y suele ser específico de cada modelo de cámara, aunque no suele diferir mucho entre modelos. También es posible gestionar ciertos parámetros de la cámara, más o menos simples, haciendo conexiones locales entre cámara y ordenador como Telnet, FTP o puerto Serie RS-232 a través de su IP.

Por último, para ciertas aplicaciones es conveniente contar con ciertos accesorios, tales como filtros, sujeciones (trípodes, brazos articulados…), posicionadores, carcasas…

Campos de aplicación

En automoción, uno de los sectores con más demanda de cámaras de visión, la visión artificial cumple bastantes funciones. En lo que se refiere a soldaduras, se utiliza para detectar tanto discontinuidades como poros en el cordón de la soldadura, mediante una iluminación led envolvente y difusa. Otra aplicación puede ser la verificación del encendido de los testigos luminosos en el salpicadero o la calibración de la aguja del cuentakilómetros y cuentarrevoluciones, a través de la iluminación de los propios testigos.

En alimentación resulta ser uno de los procesos clave en el control de calidad. Por ejemplo, el control y sellado de bandejas de yogures, embutidos… donde se examina que la unión esté bien conseguida además de que no haya restos de producto, examinado exhaustivamente con iluminación ultravioleta. Otra aplicación más común en el sector de la alimentación es la de inspección del embalaje. Se comprueba que el envase está bien impreso y centrado y que contiene todos los dibujos e imágenes requeridos. Por último la verificación de la colocación de los elementos en un recipiente, como puede ser unas porciones de queso en un embalaje redondo.

Ya no solo en alimentación sino en las industrias que se dedican al envasado de productos, resulta crucial descartar los envases en mal estado. En botellas de vidrio se realiza un control de la boca de la botella en busca de posibles roturas que hayan podido producirse en el proceso de envasado. También se controla tanto la cantidad de producto como el posicionamiento del mismo dentro de los envases, tal y como ocurre en las cestas de botellas de cerveza, donde se debe ver que el pack está completo además de que todas las botellas estén en su lugar correspondiente.

Otro sector donde está muy demandada la visión artificial es farmacia. Se usa para analizar los blísteres de medicamentos en busca de pastillas extraviadas mediante láser y led lineal y también en la lectura de Datamatrix, los cuales son códigos en 2 dimensiones asociados a cada medicamento y que la cámara debe de des encriptar y asociar con un valor preestablecido dentro del software de edición. Este último se inspecciona mediante fluorescente ultravioleta.

Para finalizar con los sectores más importantes, está el de la electrónica. Cuenta con 2 aplicaciones mayoritarias. Una es la verificación de condensadores electrolíticos, donde se debe comprobar que el cátodo está bien posicionado, o de lo contrario causaría un fallo en el sistema. La segunda es el control del pineado, donde se comprueba que los pines estén alineados y posicionados a la distancia correcta.

Nuevas Tecnologías

Además de las aplicaciones que se han mencionado con anterioridad, en el campo de la visión artificial cada vez se exploran más posibilidades, como los sistemas en 3D, las cámaras Time-of-flight o los sistemas Hiperespectrales.

Los sistemas 3D son un método de iluminación láser proyectada de forma controlada con el objetivo de extraer información dimensional de un objeto. Es ideal para aplicaciones de reconstrucción en 3D, ingeniería inversa, medición o posicionado.

Las cámaras Time-of-flight, utilizan una tecnología similar a las anteriores, en tanto en cuanto escanea la morfología de un objeto con el fin de obtener imágenes calibradas en 3D en tiempo real. Estos sistemas están adaptados para trabajar con un cierto rango de distancias y ángulo de visión, lo cual para ciertas aplicaciones resulta un factor limitante.

En los sistemas hiperespectrales, como su propio nombre dice, es la asociación de una longitud de onda de la transmisión / reflexión / emisión a cada uno de los píxeles de la imagen. Así por ejemplo, guiándonos solo con la emisión de color del objeto, se puede identificar de qué material está hecho, muy útil en el reciclaje donde es posible encontrar diferentes materiales acumulados en el mismo montón.

MEMORIA

2.1 Problema a resolver

En el cada vez más demandado sector de la visión artificial, se hace imprescindible tener una cierta base de conocimientos acerca de cómo manipular las cámaras inteligentes, incluso a la hora de hacer simples cambios de trabajos en las cámaras. Esto unido a que las cámaras, hasta hace bien poco, no disponían de una memoria interna lo suficientemente grande como para albergar un alto número de trabajos, hace que un elevado número de empresas tengan bastantes problemas a la hora de gestionar y trasferir los trabajos de la cámara, ya que para ello deben acceder a través del software que proporciona el fabricante y desde ahí eliminar, reemplazar o bien descargar nuevas aplicaciones. Esto supone instalar en los PC's conectados a la cámara el correspondiente software de visión, lo que supone emplear mucha memoria del disco duro en el programa para solo emplear un 5% de su capacidad. La otra opción sería utilizar tanto Telnet como FTP para la gestión de los trabajos, pero resulta muy complejo y largo para un operario sin conocimientos de la materia. En resumen, cada vez más empresas demandan un software ligero y fácil de instalar, que sea capaz de resolver de forma efectiva la problemática de la gestión de proyectos de la cámara y que pueda ser utilizado por cualquier usuario/operario.

2.2 Solución a desarrollar

Desde el departamento de soporte técnico de Intertronic, se ha pensado en desarrollar una pantalla que sirva de interfaz entre usuario y cámara para la gestión de proyectos, fácil de usar e instalar y que emplee un lenguaje de programación que permita su ejecución en cualquier PC, independientemente del sistema operativo que se use. La solución deberá adecuarse a las cámaras que suministra Intertronic a sus clientes, en concreto a las que presentan más limitación de memoria. Se deberá buscar un método de transferencia entre cámara y PC rápido y fácil de acceder, independientemente del terminal que se posea y por último que sea capaz de visualizar la tarea que se esté llevando a cabo en ese momento.

Dado que Di-soric está trabajando paralelamente en una pantalla para la gestión de trabajos, la que se diseñe en este proyecto debe tener la posibilidad de personalización que no puede tener Di-soric, no en el sentido de la adquisición de imágenes ni estadísticas, sino en la parte estética. Esto puede suponer que el cliente se plantee seriamente utilizar una pantalla completamente personalizable a la hora de incorporar botones o diseños de interfaz, en lugar de la estándar proporcionada por Di-soric con presumiblemente menos opciones que la primera. No se pretende crear una competencia con la empresa suministradora de cámaras, sino realizar un producto diseñado exclusivamente para cada cliente, con el que se sienta identificado a través del diseño personalizado.

2.3 Elección de las herramientas de trabajo

2.3.1 Elección de las cámaras

Una cuestión importante a la hora de realizar el proyecto es la elección de las cámaras. No todas las cámaras que hay en el mercado se comportan igual, ni tienen las mismas características, incluyendo el software de edición. Aunque este último se diseñe para que al operario le resulte fácil con solo unos pocos conocimientos de visión, tal y como se comentó con anterioridad, siempre hay diferencias entre marcas e incluso entre diferentes modelos, tanto en complejidad como en aplicaciones. Por poner un ejemplo, las cámaras de la marca Keyence presentan un software de los más intuitivos del mercado, aunque sus cámaras no presenten tantas herramientas como la competencia. En el otro lado está Microscan o Di-soric, los cuales comercializan cámaras con muchas herramientas pero disponen de un software menos intuitivo y asequible de cara al usuario.

No obstante, el suministrador principal tanto de sistemas de visión como de accesorios para los mismos de Intertronic es Di-soric, por lo que la elección de la marca queda solventada, a falta de saber los modelos.

Di-soric

Desde hace 30 años, el grupo Di-soric se especializa en el desarrollo y fabricación de sensores para la automatización industrial. Con sus innovaciones a lo largo de estos años, han ampliado su catálogo con una extensa gama de productos como iluminación LED de alta calidad, sistemas de visión artificial y sistemas de identificación. Esta empresa familiar cuenta con más de 200 personas en su plantel entre la sede central, el centro de desarrollo y la producción, que se encuentran en Alemania. Entre sus clientes se incluyen pequeñas y medianas empresas, así como fabricantes de automóviles. Distribuyen a todo el mundo a través de personal propio y distribuidores experimentados, como es el caso de Intertronic.

VS-06: Detalles de la cámara

El primer modelo que se baraja es la VS-06. Es una cámara compacta que proporciona gran capacidad de visión en un formato relativamente pequeño y un software intuitivo. Con una completa integración de comunicaciones I/O hace que sea fácil de incorporar en prácticamente cualquier aplicación de visión artificial. El autofoco de lente líquida y el zoom óptico modular permiten a la VS-06 inspeccionar objetos a distancias de 33 mm a 2 m. Al pulsar el botón AutoVision en la parte posterior de la cámara se realiza un autofoco dinámico. Cuando un objeto está centrado en el campo de visión y se presiona el botón AutoVision, la cámara ajusta automáticamente la distancia y establece parámetros internos para optimizar las capturas de imagen. El software AutoVision, diseñado para su uso con el VS-06, proporciona una interfaz intuitiva, una configuración paso a paso y una biblioteca de presets que permiten una fácil instalación e implementación. Para aplicaciones de visión más complejas, El sistema se puede actualizar de AutoVision a Visionscape.

La VS-06 cuenta con dos modelos, uno estándar que cuenta con la morfología de una cámara de visión normal y corriente y otra con óptica intercambiable según la aplicación que se desee implementar, llamado C-Mount en adelante.

Foto 1: Cámara VS-06 compacta (izquierda) y VS-06 con montura C (derecha).

La cámara dispone de las siguientes características:

- Resoluciones disponibles: SXGA (1280 x 960), WVGA (752 x 480) y WUXGA (2048 x 1088, C-Mount solamente)
- El primer sistema de visión del mundo con autofoco de lentes líquidas
- Iluminación integrada
- Ethernet integrado
- Opciones de programación flexibles para aplicaciones personalizadas
- Botón AutoVision para la orientación automática, calibración y activación
- Configuración simplificada con el software AutoVision
- Totalmente escalable con Visionscape
- Las aplicaciones pueden ser porteadas a Visionscape

Este modelo en concreto se utiliza para la verificación de montaje en automoción, identificación de piezas, posicionamiento de etiquetas, verificación de contenido, verificación e identificación de ensamblajes electrónicos y empaque de semiconductores e inspección de componentes entre otras aplicaciones.

Cableado

Como se puede observar, existen dos entradas en la cámara:

- Alimentación: Se realiza a través de un conector métrica 12 de 12 pines, generalmente conectado a un transformador VSID-PS-24V-ES, que reduce la tensión de red a 24V, la tensión de alimentación de la cámara.

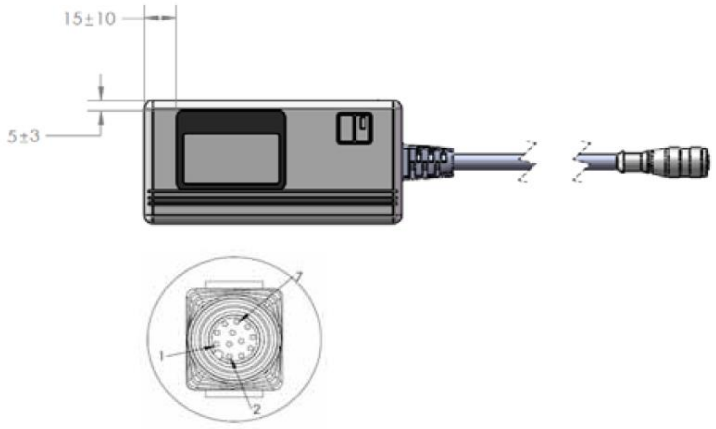


Foto 2: Transformador VSID-PS-24V-ES de 24V con conector M12 de 12 pines.

- Ethernet: Es el cable encargado de la transmisión de datos bidireccional cámara-PC. En este caso es un cable VKHM-Z-5/RJ45 métrica 12 de 8 pines unido a un conector RJ45 en su extremo.

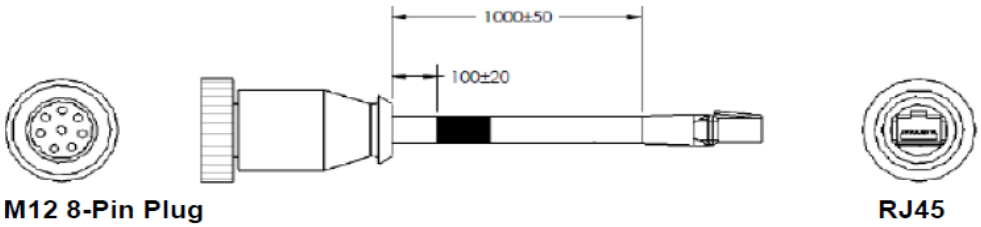


Foto 3: Cable VKHM-Z-5/RJ45 con conectores RJ45 y M12 de 8 pines.

Problemática con las VS-06

A pesar de que cuenta con múltiples herramientas para elaborar proyectos de muchos tipos, esta cámara tiene una limitación muy grande: La memoria interna. Con anterioridad se ha hablado de que muchas cámaras del mercado carecen de una amplia memoria, por lo que resulta impráctico tener que cambiar constantemente de trabajos para un mismo proceso a través del software de edición. Pues bien, esta cámara solo dispone de 6 MB de memoria interna, por lo que si se cargan trabajos relativamente pesados, con multitud de herramientas utilizadas, hace que se agote muy rápidamente el espacio disponible. De esta manera se perfilan dos maneras de gestionar los trabajos de la cámara: O bien por el software de edición o bien por TCP/IP.

La primera manera supone que en el PC que se vaya a controlar la cámara se instale AutoVision única y exclusivamente para transferir trabajos, ya que la parte de edición y análisis se suele llevar a cabo en otro departamento. Esto supone dos cosas: Que solo se va a utilizar un pequeño porcentaje de la capacidad total del software (solo transferir trabajos), y que puede haber riesgo de que el operario modifique de forma inintencionada tanto los trabajos albergados en la cámara como los del PC, como también la misma configuración de la cámara. Además, la instalación de este software supone que el sistema operativo del PC o del hmi deba ser Windows. Esto a priori no supone un grave problema ya que la mayoría de los ordenadores corren bajo Windows, pero si en su lugar se deseara utilizar Linux por ahorrar espacio en el procesador, sería imposible ejecutar el programa.

La segunda opción, la transferencia por TCP/IP, no necesita del uso del software como ocurría en el otro caso. Es más, se puede realizar en cualquier PC independientemente de la capacidad que tenga el mismo. El principal inconveniente es la necesidad de conocer y entender cómo funciona Telnet y ftp además de ciertos comandos que ni siquiera aparecen dentro del manual de la cámara. Cierto es que son mecánicos y repetitivos, pero al tratarse de una larga serie de comandos, aparece un factor crucial en la industria: El tiempo. En un proceso de traspaso y carga de un trabajo se pueden llegar a perder más de 30 segundos debido a su algo atrasado procesador. Si a eso se le suma que se deben introducir manualmente todos los comandos, puede llegar a suponer una pérdida de más de 1 minuto, imperdonable en procesos de análisis rápido (más de 800 PPM).

¿Por qué elegir esta cámara?

Se ha visto que esta cámara cuenta con unas cuantas limitaciones que pueden suponer un inconveniente para ciertos procesos en los que el tiempo es crucial. No obstante esta cámara es una buena candidata para utilizar el software que se plantea, ya que debido a su reducida memoria se hace necesario disponer de una vía rápida tanto para transferir como para cargarle trabajos y dado que se pretende evitar utilizar AutoVision, resulta ideal. Bien es cierto que seguirá siendo bastante lenta a la hora de cargar los trabajos y transferirlos, aunque presumiblemente el tiempo se reducirá a la mitad gracias a la rápida ejecución de los comandos adecuados mediante programación.

CS-50: Detalles de la cámara

Foto 4: Cámara CS-50.

El otro modelo que se plantea usar es el sensor de visión CS-50. La sucesora de la VS-06 supone un gran salto de prestaciones, en cuanto a la memoria y al procesador, en un reducido tamaño. Esta cámara dispone de las características que se muestran a continuación:

- Máximo rendimiento con hasta 2520 inspecciones por minuto
- Exportación de imágenes vía FTP
- LED internos, filtro, campo de polarización intercambiable
- Múltiples detecciones y mediciones
- Profinet / Ethernet- / TCP-IP / RS 232
- Software multilingüe e intuitivo
- Detección de forma y presencia, contador de posicionamiento y lógica
- Autofoco sin desgaste integrado
- Dos longitudes focales en una unidad, conmutables mediante clic-zoom
- Diseño compacto y bajo peso
- Número prácticamente ilimitado de puestos de trabajo posibles

Ventajas de la CS-50 frente a la VS-06

Si se compara con la anterior cámara, en principio es sustancialmente mejor: Incluye transferencia de imágenes vía FTP, bastante útil si se quiere analizar el problema que pueda haber en el proceso, un diseño compacto que hace que la cámara pueda instalarse en casi cualquier sitio y lo más importante, se pueden almacenar casi infinitos trabajos. Esta última característica supone que el problema que existía con las VS-06 ya no existe, pudiéndose cargar dentro de la cámara multitud de trabajos de una sola vez sin necesidad de recurrir al software de edición ni al traspaso TCP/IP más adelante por cambios en la producción.

Si además se tiene en cuenta que el procesador que lleva incorporado la CS-50 es mucho más rápido que su predecesora, hace que el tiempo que se tarda en traspasar el trabajo y cargarlo en la cámara se reduzca de más de 30 segundos a menos 10, aunque llega a reducirse a apenas tres segundos ya que desaparece la necesidad de traspasar los trabajos en el mismo instante, es decir 10 veces más rápido.

Desventaja de la CS-50: Las herramientas

Hasta ahora la CS-50 implica un salto de calidad muy grande en comparación con el anterior modelo, tanto en velocidad y tamaño como en capacidad. Pero tal drástica reducción del tamaño de la cámara implica una desventaja que para ciertas aplicaciones puede ser crucial: Las herramientas de edición. Así como el software de la VS-06 cuenta con un amplio abanico de herramientas de análisis a nuestra disposición, en este modelo más nuevo solo se encuentran 5 herramientas, lo cual limita su uso para aplicaciones más complejas como lectura de códigos o de texto, o comparación de cadenas (de estas herramientas se hablará más adelante).

Por lo tanto, es una cámara que ejecuta aplicaciones de manera muy rápida, aunque claro está que su principal hándicap es que esas aplicaciones no tienen una gran complejidad, lo que resulta una gran desventaja en la industria ya que cada vez es más estricto el control de calidad de los procesos.

¿Cuál elegir?

Vistas las principales ventajas y desventajas que poseen las dos cámaras es momento de decidir en qué cámara volcar nuestra aplicación: bien en la VS-06 con una gran cantidad de herramientas de análisis de trabajos pero con una memoria reducida y un procesador más desfasado, o bien la CS-50 con una alta velocidad de procesamiento y mayor memoria pero con muchas menos herramientas.

En principio parece obvio que la cámara que más necesita un software de apoyo es la VS-06, tanto para traspasar como para cargar trabajos a la cámara. No obstante, y dado que es un modelo que progresivamente se irá sustituyendo por la CS-50, es conveniente que se cree también un software intuitivo y fácil para la misma, ya que a pesar de que la carga de trabajos sea muy rápida, sigue siendo engorroso hacerlo vía Telnet.

Partiendo de esta premisa, se pasaría de crear un software para solo una cámara a un software para las dos, lo que supone una ventaja que no se había barajado con anterioridad: En el caso de que varias cámaras de diferentes modelos estén integradas en una red PROFINET, se podría controlar todas ellas a través de la misma pantalla, cambiando únicamente la IP de las cámaras.

En resumidas cuentas, el software a crear debe “maquillar” las carencias de las dos cámaras al máximo. En el caso de la VS-06 debe poderse traspasar trabajos del PC a la cámara y cargar trabajos en el menor tiempo posible, mientras que en la CS-50 se debería crear una interfaz únicamente para la carga y cambio de trabajos. Con todo ello se podría considerar que las necesidades de cada cámara quedan cubiertas.

2.3.2 Elección del entorno de programación

Resulta lógico pensar que esta es sino la más importante, una de las partes más importantes de todo el proyecto. De este paso depende que el software que se diseñe funcione correctamente y se ejecute lo más rápidamente posible, es decir, que esté bien optimizado. Como bien se ha dicho en el apartado de las cámaras, si se quiere emplear la VS-06 es prioritario reducir el tiempo de ejecución al máximo para perder el menor tiempo posible transfiriendo y cargando trabajos. Es por ello que se ha de elegir el entorno de programación con sumo cuidado.

¿Qué leguaje escoger?

Hoy en día hay numerosos lenguajes de programación tales como C++, Java, MATLAB o Visual Basic, además de todos los softwares de programación específicos de cada dispositivo como es el caso de ABB y el programa Rapid para sus robots, o los diagramas de contactos y el GRAFCET en TIA Portal de Siemens por ejemplo. Sin embargo para la aplicación que se desea realizar hay que tener en cuenta diferentes factores:

- Debe de ser compatible con otros diferentes sistemas operativos, lo que supondría que independientemente del sistema operativo elegido para controlar las cámaras funcionará de la misma manera sin ser necesario ningún cambio de entorno ni de lenguaje
- Debe de emplearse un lenguaje de programación de nivel medio o alto, en el cual los algoritmos se expresan de una manera más lógica y fácil para el programador, al contrario que los lenguajes que interactúan directamente con el hardware.
- Debe tener la posibilidad de construir ventanas gráficas, ya que obviamente es la manera más intuitiva posible con la que el operario se puede comunicar con la cámara.

Con todas estas consideraciones, se ha concluido junto al departamento de soporte técnico de Intertronic que la aplicación se desarrolle en Mono, más concretamente con el editor MonoDevelop.

Mono

Mono es una plataforma de desarrollo de código abierto basada en .NET Framework, que permite a los desarrolladores crear aplicaciones multiplataforma con una productividad mejorada para el desarrollo. La implementación .NET de Mono se basa en las normas ECMA (European Computer Manufacturers Association) para C # y la Common Language Infrastructure.

Foto 5: Logo de Mono Project

Soportado previamente por Novell, Xamarin y ahora Microsoft y la Fundación .NET, el proyecto Mono tiene una comunidad activa y entusiasta. Mono incluye tanto las herramientas de desarrollador como la infraestructura necesaria para ejecutar aplicaciones cliente y servidor .NET.

Hay varios componentes que componen Mono:

- **Compilador de C #:** El compilador del C # de Mono es característica completa para C # 1.0, 2.0, 3.0, 4.0 y 5.0 (ECMA).
- **Mono Runtime:** El tiempo de ejecución implementa la infraestructura de lenguaje común (CLI) de ECMA. El tiempo de ejecución proporciona un compilador Just-in-Time (JIT), un compilador Ahead-of-Time (AOT), un cargador de biblioteca, el recolector de basura, un sistema de subprocesamiento y la funcionalidad de interoperabilidad.
- **Biblioteca de clases de .NET Framework:** La plataforma Mono proporciona un conjunto completo de clases que proporcionan una base sólida para crear aplicaciones. Estas clases son compatibles con las clases .NET Framework de Microsoft.
- **Mono Class Library:** Mono también ofrece muchas clases que van por encima y más allá de la Biblioteca de clases base proporcionada por Microsoft. Estos proporcionan funcionalidad adicional que son útiles, especialmente en la construcción de aplicaciones Linux. Algunos ejemplos son clases para Gtk +, archivos Zip, LDAP, OpenGL, Cairo, POSIX, etc.

Tal y como se comentaba en el anterior apartado, uno de los requisitos indispensables era que el software que se cree fuera capaz de correr en diferentes sistemas operativos sin necesidad de modificarlo a posteriori. Pues bien Mono C# es capaz de correr tanto en Linux, OS X, BSD y Microsoft Windows, incluyendo x86, x86-64, ARM, s390, PowerPC y mucho más. Además de esta importante característica cuenta con bastantes más:

- **Multi lenguaje**: Se puede desarrolar en C # 4.0 (incluyendo LINQ y dinámico), VB 8, Java, Python, Ruby, Eiffel, F #, Oxygene, y otros lenguajes.
- **API compatible de Microsoft:** Ejecute aplicaciones ASP.NET, ADO.NET, Silverlight y Windows.Forms sin recompilación
- **Open Source, Software Libre:** Mono Runtime, los compiladores y las bibliotecas se distribuyen utilizando la licencia del MIT.
- **Cobertura integral de tecnología:** Enlaces e implementaciones gestionadas de muchas bibliotecas y protocolos populares.

Con todo lo expuesto con anterioridad, la utilización de Mono supone unas enormes ventajas a la hora de programar, tales como:

- **Popularidad:** Construido sobre el éxito de .NET, hay millones de desarrolladores que tienen experiencia en la creación de aplicaciones en C #. También hay decenas de miles de libros, sitios web, tutoriales y código fuente de ejemplo para ayudar con cualquier problema imaginable.
- **Programación de nivel superior:** Todos los idiomas Mono se benefician de muchas características del tiempo de ejecución, como gestión de memoria automática, reflexión, genéricos y enhebrado. Estas características le permiten concentrarse en escribir su aplicación en lugar de escribir el código de infraestructura del sistema.

- **Biblioteca de clases base:** tener una biblioteca de clases completa proporciona miles de clases incorporadas para aumentar la productividad. ¿Necesita código de socket? No hay necesidad de escribir uno propio ya que está integrado en la plataforma.
- **Cross Platform:** Mono está construido para ser plataforma cruzada. Mono se ejecuta en Linux, Microsoft Windows, Mac OS X, BSD y Sun Solaris, Nintendo Wii, Sony PlayStation 3, Apple iPhone y Android. También se ejecuta en x86, x86-64, IA64, PowerPC, SPARC (32), ARM, Alpha, s390, s390x (32 y 64 bits) y más. Desarrollar su aplicación con Mono le permite ejecutar en casi cualquier computadora en existencia.
- **Common Language Runtime (CLR):** El CLR le permite elegir el lenguaje de programación que más le conviene trabajar, y puede interoperar con código escrito en cualquier otro lenguaje CLR. Por ejemplo, puede escribir una clase en C #, heredar de ella en VB.Net y usarla en Eiffel. Puede elegir escribir código en Mono en una variedad de lenguajes de programación.

Resumidamente, Mono no solo cumple las expectativas que se tenía acerca del software de programación, sino que además las supera. La elección del lenguaje de programación según las exigencias del usuario, la ejecución de las aplicaciones .NET sin tener que recompilar o la popularidad que ha adquirido la programación en C# concretamente, hace que este sea la plataforma ideal a utilizar.

Xamarin Studio (Monodevelop) y el desarrollo de ventanas gráficas

Una vez vistas las ventajas que supone utilizar Mono como la plataforma de desarrollo de la aplicación, el objetivo es conocer ahora la herramienta de programación. En este caso se trata de Xamarin Studio (antes conocido como Monodevelop), cuya función no solo se limita a las aplicaciones de consola o a las ventanas gráficas, sino que se ha potenciado en estos últimos años como una herramienta para el desarrollo web en HTML y aplicaciones móviles, con la ventaja que supone crear un software multiplataforma, tanto para IOS como para Windows o Android, en un lenguaje de programación tan extendido como el C#. La implicación de este hecho es que cualquiera con ciertos conocimientos de C# es capaz de crear aplicaciones móviles de 0 y lanzarlas al mercado de cada plataforma. No obstante, al ser solo la parte de diseño gráfico de ventanas la que atañe a la aplicación que se pretende desarrollar, se dejará la parte de programación móvil en un segundo plano.

Foto 6: Logo de Xamarin Studio

2.3.3 Trasferencia de archivos y ejecución de comandos: Telnet y puerto Serie

No es ningún misterio que la transferencia de datos entre cámara y PC es fundamental para esta aplicación a la hora de manejar trabajos bidireccionalmente, lo que hace que la buena comunicación entre cámara y PC sea indispensable. Pero además de la transferencia de trabajos en las dos direcciones, se hace necesario poder obtener información del estado de la cámara, manejar los trabajos o simplemente iniciar o parar una inspección, por lo que obviamente no solo es cuestión de pasar trabajos de un lado a otro.

Hoy en día el tratamiento y procesamiento de la información debe ser rápido, fiable y en el menor espacio posible. Las memorias USB son un gran ejemplo: Cada vez procesan más rápido la información y son más pequeños con la misma memoria. Además solo es necesario conectarlo al dispositivo correspondiente para poder tener acceso a la misma información. No obstante y dado que se trabaja en un ámbito industrial, se hace necesario tener un equipo resistente y que garantice que la información llegue a su destino, por lo que las memorias USB no serían la mejor opción ya que cuentan con componentes electrónicos pequeños que pueden dañarse con facilidad en según qué ambientes. Se necesita una transmisión directa (vía cable) a ser posible, con conectores robustos y resistente a ambientes duros. Es por eso que, a pesar de que sea la única opción disponible en las cámaras, se utilizará un cable Ethernet con conector RJ45 en un extremo y un conector de métrica 12 de 8 pines en el otro (conector de la cámara).

Esta medida solo deja dos posibilidades, o bien la transferencia de archivos y comunicación a través de Ethernet IP, la cual es la solución más utilizada nivel industrial, o bien a través de puerto serie que, a pesar de ser una tecnología un poco desfasada, realiza una transferencia de archivos extremadamente rápida. En un primer momento parece lógico utilizar Ethernet IP, ya que como se ha comentado en un principio es la solución más utilizada en la industria y proporciona una comunicación fiable y bastante rápida. Sin embargo, la problemática resulta cuando ciertos comandos críticos (incluidos dentro del manual que proporciona Di-soric con la VS-06), precisan una conexión puerto serie RS-232 para ejecutarse. Es el caso del comando “jobdownload”, cuya función es transferir un trabajo a la cámara, esencial para la aplicación. Lo que hace inevitable que para que la aplicación funcione se tenga que recurrir al puerto serie.

Puerto Serie

Un puerto serie o puerto en serie es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, donde la información es transmitida bit a bit, enviando un solo bit a la vez.

En tecnologías básicas, es una interfaz física de comunicación en serie a través de la cual se transfiere información mandando o recibiendo un bit. A lo largo de la mayor parte de la historia de las computadoras, la transferencia de datos a través de los puertos de serie ha sido generalizada. Se ha usado y sigue usándose para conectar las computadoras a dispositivos como terminales o módems. Los ratones, teclados, y otros periféricos también se conectaban de esta forma.

Mientras que otras interfaces (como Ethernet y USB) mandaban datos como un flujo en serie, el término "puerto serie" normalmente identifica el hardware más o menos conforme al estándar RS-232, diseñado para interactuar con un módem o con un dispositivo de comunicación similar.

Inconvenientes del puerto serie

En la mayoría de los periféricos en serie, la interfaz USB ha reemplazado al puerto serie por ser más rápida. La mayor parte de las computadoras están conectados a dispositivos externos a través de USB y, a menudo, ni siquiera llegan a tener un puerto serie. Se elimina para reducir los costes y se considera que es un puerto heredado y obsoleto. Sin embargo, los puertos serie todavía se encuentran en sistemas de automatización industrial y algunos productos industriales y de consumo. Es por eso que las cámaras de Di-soric cuentan con ciertos comandos que solo se pueden ejecutar a través de RS-232, en parte por la rapidez de transmisión que se contaba con anterioridad. No obstante esta transmisión se seguiría haciendo mediante el cable Ethernet, ya que la cámara puede tolerar los comandos serie, pero lo más seguro es que el propio ordenador no cuente siquiera con el mencionado puerto. Por lo tanto, se sigue siendo una dificultad transmitir las órdenes sin contar con un conversor Puerto Serie-USB, que afortunadamente tiene disponible Di-soric.

Transferencia vía Telnet: Posible

Cuando el proyecto completo se iba a centrar completamente en el traspaso de archivos mediante puerto serie, navegando por distintos foros que recogían la problemática de estas cámaras se encontró la solución definitiva sin la necesidad de utilizar el puerto serie. Lo único necesario era una conexión FTP y otra conexión Telnet. Esto supone una gran diferencia en cuanto a facilidad de traspaso, ya que toda la problemática que suponía conectar un acople USB para el puerto serie y la conversión necesaria para poder utilizar cable Ethernet con la cámara desaparecen. En definitiva, no iba a hacer falta ningún intermediario entre cámara y PC.

Básicamente este procedimiento consiste en crear primeramente mediante Telnet, un drive temporal suficientemente grande como para almacenar el archivo que se desee traspasar a la cámara, para luego en una sesión FTP traspasarlo al directorio que se haya creado en Telnet, con las pertinentes indicaciones previas antes de la transferencia del archivo. Finalmente cuando el archivo se ha traspasado al directorio interno, mediante un comando que se verá a continuación, se le indicará al archivo que se descomprima, se almacenen sus componentes en el directorio correspondiente y que se almacene en el slot que el usuario le indique.

No obstante y antes de mostrar cual es la resolución del problema, se comentaran ciertos aspectos básicos de las conexiones FTP y Telnet.

Telnet

Telnet (Telecommunication Network) es el nombre de un protocolo de red que permite acceder a otra máquina para manejarla remotamente como si se estuviera sentado delante de ella. También es el nombre del programa informático que implementa el cliente. Para que la conexión funcione, como en todos los servicios de Internet, la máquina a la que se acceda debe tener un programa especial que reciba y gestione las conexiones, en este caso el prompt de comandos de Windows, Linux… El puerto que se utiliza generalmente es el 23, aunque ya se adelanta que en la aplicación se utilizarán más puertos.

Telnet sólo sirve para acceder en modo terminal, es decir, sin gráficos, pero es una herramienta muy útil para arreglar fallos a distancia, sin necesidad de estar físicamente en el

mismo sitio que la máquina que los tenga. También se usa para consultar datos a distancia, como datos personales en máquinas accesibles por red, información bibliográfica, etc.

Para iniciar una sesión con un intérprete de comandos de otro ordenador, puede emplear el comando Telnet seguido del nombre o la dirección IP de la máquina en la que desea trabajar, por ejemplo si desea conectarse a la máquina "modem1" se deberá teclear "Telnet modem1", y para conectarse con la dirección IP "192.168.0.1" deberá utilizar "Telnet 192.168.0.1".

Una vez conectado, se podrá ingresar el nombre de usuario y contraseña remoto (si los hay), para iniciar una sesión en modo texto a modo de consola virtual. El único inconveniente que puede tener el uso de Telnet, es que la actividad que se realice puede ser vista por otros usuarios si tienen acceso al canal por el que se transmite la información, haciendo que este no sea un método seguro de transmisión de datos. No obstante y al ser utilizado en redes locales, su uso no debería implicar ningún problema para el sistema completo.

FTP

FTP (siglas en inglés de File Transfer Protocol, 'Protocolo de Transferencia de Archivos') en informática, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

El servicio FTP es ofrecido por la capa de aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y el 21, en el caso de nuestra aplicación el 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor y/o apropiarse de los archivos transferidos.

Para solucionar este problema son de gran utilidad aplicaciones como SCP y SFTP, incluidas en el paquete SSH (Secure Shell en inglés), que permiten transferir archivos pero cifrando todo el tráfico.

FTP tiene dos tipos de denominaciones, o bien se es servidor o bien se es cliente FTP. Por un lado, como su propio nombre indica, el servidor FTP es un programa especial que se ejecuta en un equipo servidor normalmente conectado a Internet (aunque puede estar conectado a otros tipos de redes, LAN, MAN, etc.). Su función es permitir el intercambio de datos entre diferentes servidores/ordenadores. Las aplicaciones más comunes de los servidores FTP suelen ser el alojamiento web, en el que sus clientes utilizan el servicio para subir sus páginas web y sus archivos correspondientes; o como servidor de backup (copia de seguridad) de los archivos importantes que pueda tener una empresa. Para ello, existen protocolos de comunicación FTP para que los datos se transmitan cifrados, como el SFTP (Secure File Transfer Protocol).

En cuanto al cliente FTP, es un programa que se instala en el ordenador del usuario, y que emplea el protocolo FTP para conectarse a un servidor FTP y transferir archivos, ya sea para descargarlos o para subirlos. Para utilizar un cliente FTP, se necesita conocer el nombre del

archivo, el ordenador en que reside (servidor, en el caso de descarga de archivos), el ordenador al que se quiere transferir el archivo (en caso de querer subirlo al servidor), y la carpeta en la que se encuentra. Algunos clientes de FTP básicos en modo consola vienen integrados en los sistemas operativos, incluyendo Microsoft Windows, DOS, GNU/Linux y Unix. Si se desea tener privilegios de acceso a cualquier parte del sistema de archivos del servidor FTP, de modificación de archivos existentes, y de posibilidad de subir nuestros propios archivos, generalmente se suele realizar mediante una cuenta de usuario. Para las cámaras de la aplicación se utiliza la autenticación "login" y "password".

¿Cómo transferir archivos desde el PC a la cámara?

Una vez definidos los aspectos básicos de cada una de las conexiones remotas expuestas anteriormente, se procederá a realizar un ejemplo manualmente con el fin de que quede claro el procedimiento que más tarde se automatizará en la aplicación. Es importante recalcar que este ejemplo se va a realizar bajo Windows 7, por lo que los menús pueden ser diferentes con respecto a otras versiones de Windows u otros sistemas operativos en la primera parte, aunque el procedimiento es exactamente el mismo. También advertir que según el antivirus que el usuario tenga instalado en el PC, puede dar problemas a la hora de realizar el traspaso, por lo que es recomendable cambiar los permisos de para la red local con la cámara o bien desactivarlo (no recomendable si el equipo dispone de conexión a internet).

El primer paso es activar Telnet en el equipo para poder activar la conexión remota. Para ello se deberá ir al apartado de programas en el panel de control, y dentro de programas y características clicar a la izquierda el menú de "Activar o desactivar las características de Windows". Una vez dentro se desplegará el menú de características (Foto1).

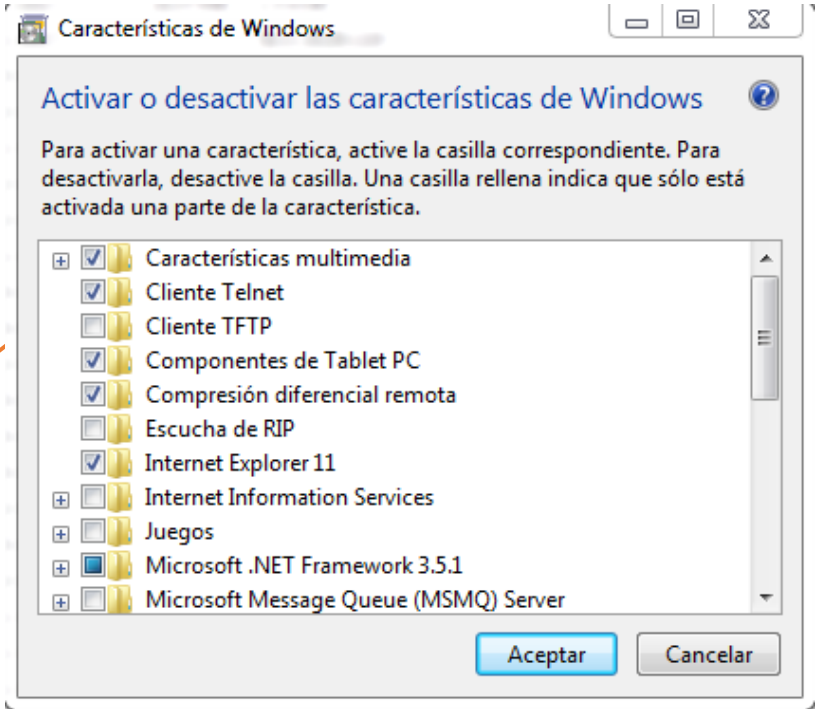


Foto 7: Activación del cliente Telnet en el equipo

A continuación se deberá marcar la casilla de cliente Telnet para poder activar el mismo, tal y como se indica en la foto, para luego apretar el botón de aceptar en la parte de abajo. Este proceso puede tardar unos segundos en realizarse.

Siguiendo con el proceso, se abrirá una ventana de comandos de Windows (simplemente escribiendo en la barra de búsqueda de inicio cmd y clicando sobre el resultado) y se procederá a iniciar una nueva sesión Telnet con la IP de la cámara que se vaya a utilizar, en este caso la 192.168.0.11.

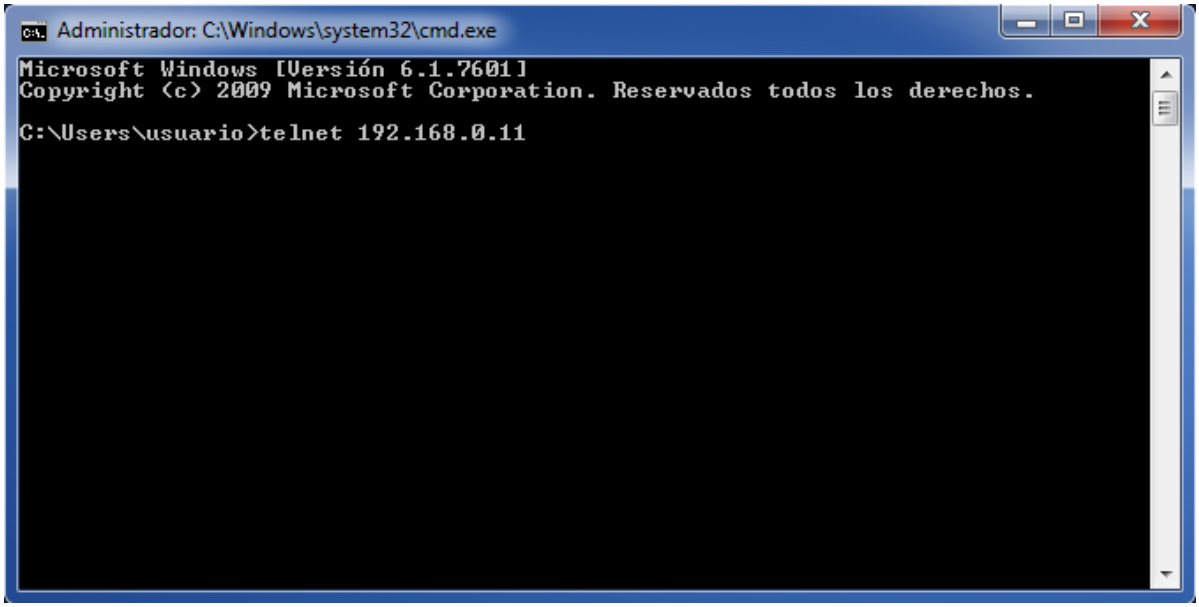


Foto 8: Comando para iniciar la conexión Telnet.

Llegado a este punto pueden haber dos situaciones: Que la sesión Telnet esté protegida con usuario y contraseña, lo cual se mostrará en pantalla y el usuario tendrá que escribir los credenciales, o bien que no esté protegido y se entre directamente a la sesión Telnet. En este caso la cámara no está protegida, luego se puede entrar sin problemas.

Ya en Telnet, el usuario deberá escribir el comando <CreateBINDrive (1024*400)> y pulsar la tecla intro .Este comando como se adelantaba al principio, crea un disco temporal dentro del directorio principal de la cámara con el fin de pasarlo a la memoria "flash" de la misma. El producto numérico corresponde al tamaño del trabajo a traspasar indicado en KB. Normalmente si no es un trabajo excesivamente pesado, el archivo rondará los 400 KB, por lo que normalmente se trabaja de unos 500 a 600 KB. En este caso el archivo del trabajo es muy pequeño, entorno a los 250 KB, por lo que con 400 será más que suficiente. Si el proceso ha resultado un éxito, devolverá un 1 como valor.

Para el siguiente paso se deberá abrir nuevamente otra ventana de comandos dejando abierta la anterior, ya que se volverá a usar más tarde. En este caso lo que se hará es iniciar una sesión ftp con la IP de la cámara, para indicar los parámetros de la transferencia además de la

realización de la misma. Una vez introducido el comando se pedirá unos credenciales como ocurría con Telnet, solo que en este caso son los mismos para todas las cámaras: “target” para el usuario y “password” para la contraseña, lo cual facilitará mucho más la automatización del proceso posteriormente.

Foto 9: Valor de retorno del comando “CreateBINDrive”.

Dentro de la sesión FTP se utilizarán varias instrucciones. La primera, <binary>, consiste en indicarle al sistema que el modo de transferencia va a ser en binario, lo que quiere decir que transmite todos los ocho bits por byte y, por lo tanto, proporciona menos posibilidades de un error de transmisión, fundamental para transmitir archivos distintos de los archivos ASCII. La segunda a ejecutar es <lcd>, la cual cambia el directorio de trabajo del PC al que se le indique, como en este caso en el que el archivo a transferir se encuentra dentro del disco C, por lo que habrá que introducir <lcd C:\>. Si se ha cambiado de directorio en el PC de usuario, ahora habrá que cambiar de directorio en la máquina remota (la cámara) en la tercera instrucción. Para ello se necesitará el comando <cd> para establecer un directorio remoto de trabajo, en este caso la instrucción completa sería <cd /bind0>, donde bind0 es el directorio temporal que se ha creado mediante Telnet en la anterior ventana, y donde se mandarán los trabajos de la cámara antes de almacenarlos en los slots de la misma. Y por último la instrucción más importante sería <send> que, como su nombre en inglés indica, envía el archivo de la dirección del PC que se le haya indicado a la carpeta del directorio remoto seleccionada. El trabajo que se pretende traspasar se llama “pieza_grande.avz” y se encuentra en el disco C de nuestro PC. Cabe hacer un inciso en esta parte, ya que para poder realizar este proceso el trabajo generado mediante AutoVision debe ser guardarse en formato .avz, un formato comprimido que incluye el trabajo más las herramientas que se han utilizado en el mismo, para su correcto funcionamiento. Por lo tanto el comando a escribir es <send pieza_grande.avz>.

Una vez introducido todos estos comandos, si la transferencia se ha realizado con éxito aparecerá un mensaje con la cantidad de datos enviados y el tiempo que ha llevado enviarlos

junto con la velocidad de transferencia. Ya ejecutados los comandos se podrá salir de la sesión FTP o bien con el comando <quit> o bien cerrando la ventana directamente.

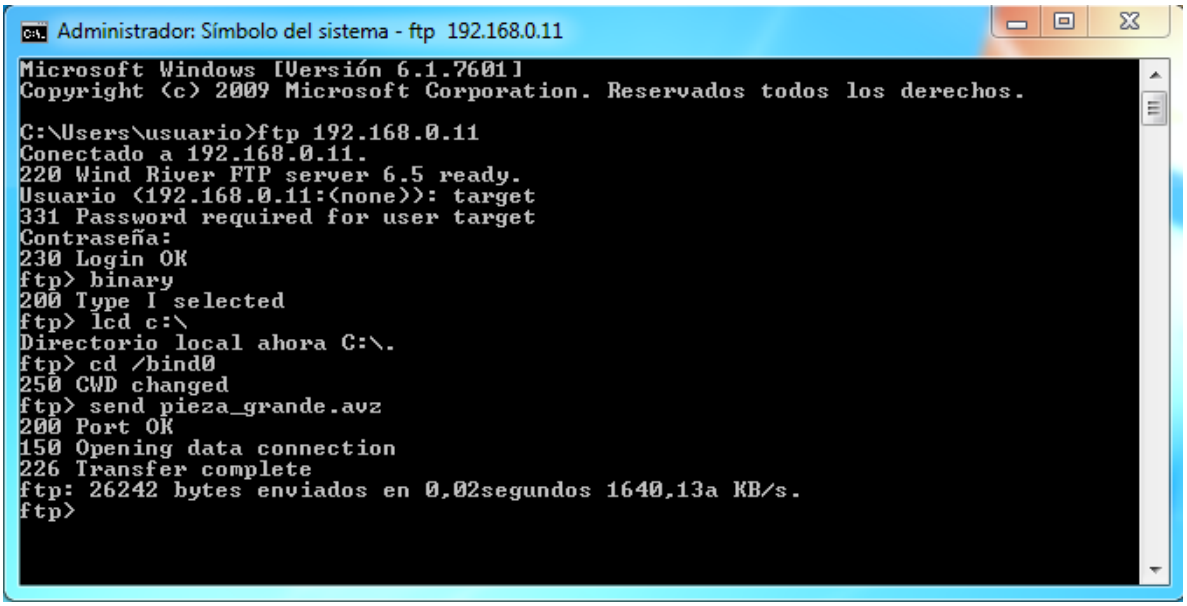


Foto 10: Comandos FTP ejecutados.

El último paso a seguir para completar la transferencia se llevará a cabo en la ventana de la sesión Telnet que se ha dejado abierta en apartados anteriores. En este último paso se deberá utilizar el comando <FlashAvz> con el cual se indicará tanto la dirección del archivo del trabajo como el slot al que se quiera mover este. Este comando descomprime el archivo .avz que se había transferido en el FTP y lo transfiere al slot deseado. En este caso se indicará que el trabajo se cargue en el slot 2, por ejemplo. Así pues el comando a introducir será <FlashAvz "/bind0/pieza_grande.avz",2>. La ejecución de esta instrucción puede llevar bastante tiempo en el caso de las VS-06 ya que el procesador es bastante desfasado, como se ha insistido varias veces, y puede tardar entorno a los 20 segundos en ejecutarse. Si en efecto se ha completado el traspaso al slot, la pantalla devolverá el valor 1 (Foto 11).

No obstante, en el caso de querer cerciorarse de que el trabajo se ha traspasado con éxito, se puede acceder a otra sesión Telnet en el puerto 49211. Desde este puerto se pueden ejecutar, controlar y examinar los trabajos de dentro de la cámara, modificar las variables del trabajo e incluso controlar las luces del dispositivo. Entrar a este puerto es tan sencillo como repetir el paso visto en la Foto 2 solo que incluyendo a continuación de la instrucción el puerto 49211. Una vez dentro, se deberá pulsar las teclas intro y escape hasta que salga el símbolo de un corazón latente. El comando para visualizar el contenido de la cámara es <jobinfo>, el cual proporciona el nombre de cada trabajo incluido en cada slot si lo hubiera. Se puede observar que si se ejecuta el comando aparece que en el slot 2 se encuentra el trabajo pieza_grande.avp, por lo que se ha transferido correctamente (Foto 12).

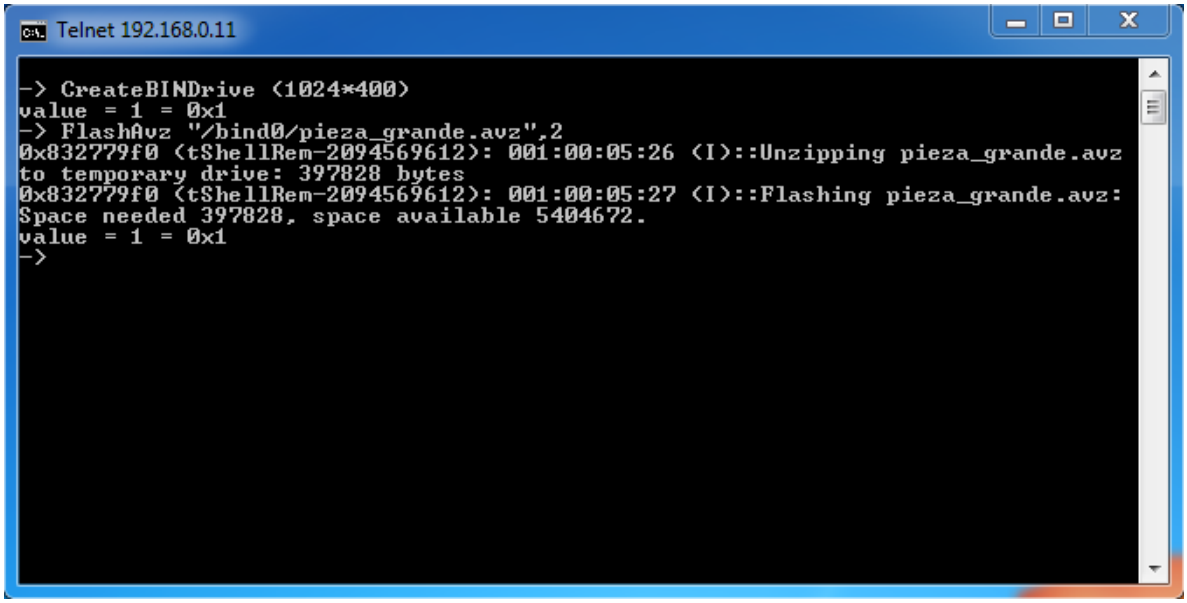


Foto 11: Resultado del traspaso a la cámara.

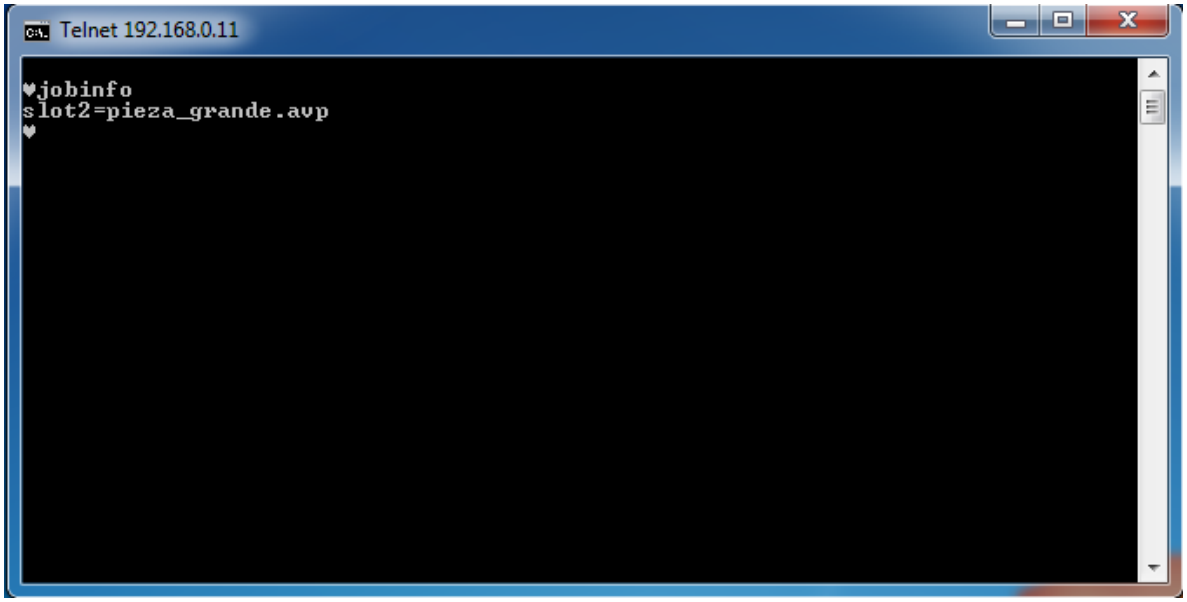


Foto 12: Comprobación del traspaso en Telnet en el puerto 49211.

2.4 Diseño de las interfaces

Si la intención del proyecto desde un primer momento ha sido la rápida gestión de los trabajos incluidos dentro de la cámara, el diseño que tenga la misma pantalla es fundamental a la hora de agilizar el proceso. Tal y como ocurre con los sistemas operativos, si no hubiera una interfaz gráfica que sirviera de capa entre el sistema y el usuario, muy pocos serían capaces de manejar un ordenador debido a su tremenda complejidad. No obstante y por suerte, los ordenadores modernos cuentan con esta "capa" que, si bien pude diferir entre fabricantes de sistemas operativos, intenta hacer que las funciones del ordenador sean más intuitivas.

No es ningún misterio que para el control de procesos muchas veces es necesario, sino imprescindible, una interfaz gráfica con la que el operario sea capaz tanto de visualizar como de interactuar con el proceso. Así por ejemplo, en materia de pantallas Táctiles Siemens ha desarrollado un importante abanico de productos para implantarlos paralelamente en el proceso mediante TIA Portal. Además las opciones que implementa el mismo programa para su personalización, ya sea a través de imágenes o formas y polígonos, lo hacen una opción realmente interesante a la hora de crear interfaces intuitivas incluso para un operario poco experimentado.

Foto 13: PC Industrial de 12".

Sin embargo en la aplicación para la cámara no queda más remedio que descartar la opción de realizarlo a través de TIA portal, ya que es necesario realizarla a través de comandos Telnet y FTP como ya se ha visto, y también porque es necesario contar con una programación detrás basada en diagrama de contactos en un PLC. En su lugar, se ha pensado en realizarla en un PC táctil industrial de pantalla de 12" con Windows 7 integrado. Esto facilita más las cosas debido a que el software se va a programar en Windows para Windows, aunque como se avanzó, si el cliente desea utilizar otra pantalla con un sistema operativo distinto a Windows no habría ningún inconveniente.

2.4.1 Prototipo 1: Boceto (02/03/2017)

Desde soporte técnico de Intertronic en un principio se pensaba en una interfaz gráfica como la que aparece a continuación:

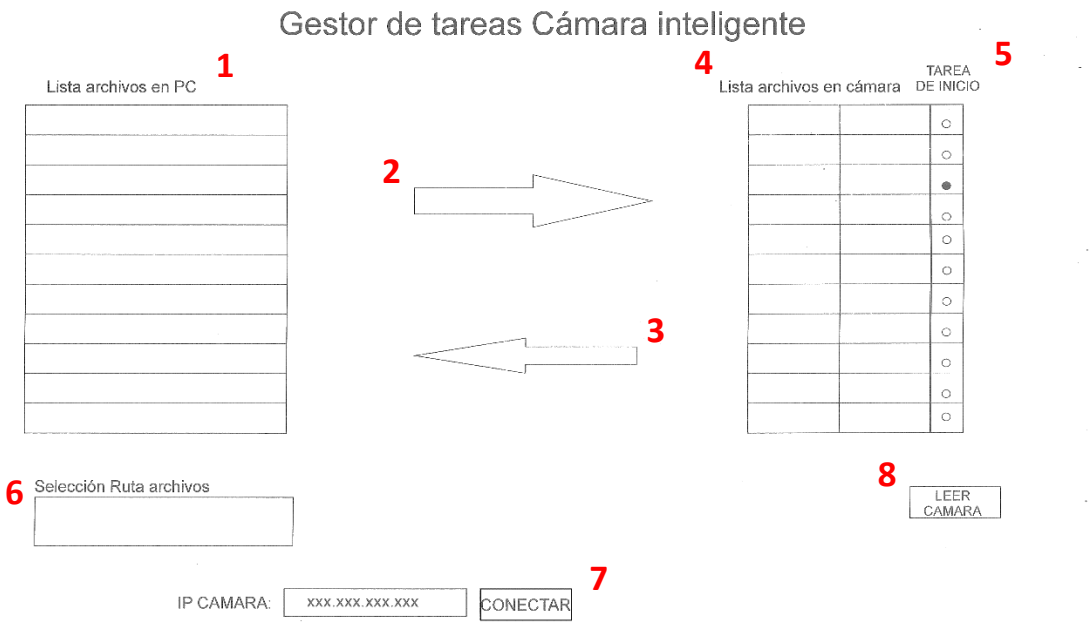


Foto 14: Boceto incial suministrado por Intertronic.

Como se puede apreciar, la intención en todo momento es crear una pantalla sencilla con apartados lo suficientemente claros como para que sea intuitiva de cara al operario. Así por ejemplo, se puede deducir que la flecha hacia la derecha indica que se va a traspasar del PC a la cámara, mientras que la flecha de derecha a izquierda indica que se va a traspasar de la cámara al PC. No obstante se hará una descripción de cada una de las partes de la interfaz a continuación:

1. En esta lista se encuentran todos los trabajos compatibles de la cámara, los cuales a su vez están almacenados dentro de una carpeta que el usuario tendrá que seleccionar manualmente en el apartado 6.
2. Como se avanzaba brevemente, para traspasar los trabajos almacenados dentro de la ruta especificada por el usuario se pulsará el botón/flecha con dirección a la cámara, previamente especificado el trabajo y el slot en el que se va a guardar.
3. Sigue siendo la misma premisa que el apartado anterior, solo que se invierte el sentido de la transferencia de la cámara a la carpeta deseada. Cabe decir que la carpeta suele ser la misma que la que se utiliza para los trabajos que se quieren traspasar a la cámara, aunque no necesariamente.
4. Esta es la lista que recoge todos los trabajos que están dentro de la cámara. Visualiza los trabajos incluidos en cada slots además de los slots que están libres. En principio como es una lista dinámica, es decir que varía al traspasar trabajos, es importante que el operario actualice la lista cada vez que realiza un traspaso, tal y como se verá en el paso 8. En esta

lista además el operario podrá seleccionar el slot que se desee para ejecutar el trabajo incluido en él.

5. Un aspecto que no se ha comentado de las cámaras es el concepto de trabajo de inicio. Como su propio nombre indica, es el trabajo que se pone en marcha cada vez que se inicia (enciende) la cámara. Desde AutoVision, el usuario puede seleccionar el trabajo que desee hacer para hacerlo trabajo de arranque, aunque obviamente en la interfaz se emplearán comandos Telnet, en este caso "jobboot". El usuario clicará el botón correspondiente a cada slot para hacerlo trabajo de arranque para la próxima vez que se arranque la cámara.
6. Tal como se avanzaba en el apartado 1, en esta entrada es donde se deberá introducir manualmente la ruta que incluya los archivos de la cámara a traspasar a la cámara, o bien donde se quiera almacenar los archivos que se recuperen de la cámara.
7. Este paso es fundamental para la comunicación de la cámara con el PC, ya que sin este no puede existir comunicación. El usuario deberá escribir la IP de la cámara, la cual se puede consultar en AutoVision por si existe alguna duda, y darle al botón de conectar. El programa deberá hacer una conexión vía Telnet con la cámara y comprobar que esta es correcta. Obviamente, el programa se deberá desarrollar condicionado siempre a la correcta introducción de la IP, ya que de lo contrario no podrá siquiera funcionar.
8. Por último un botón relevante a la hora de ver en tiempo real los trabajos incluidos en la cámara es el de actualizar que, como ya se ha hablado con anterioridad realiza un chequeo de los slots de la cámara incluyendo las modificaciones que se hayan podido hacer en ellos.

Pese a ser un diseño bastante rudimentario para una interfaz gráfica, resulta infinitamente más cómodo trabajar a través de ella que mediante comandos Telnet. Todo hay que decir que el diseño que se ha propuesto no tiene porqué ser exactamente el mismo que el que se vaya a utilizar finalmente. Sin ir más lejos, hay ciertos aspectos que se deberían optimizar de cara al cliente final, como por ejemplo el acceso a la aplicación. Este aspecto según qué cliente lo demande es interesante, ya que si el HMI va a estar en un entorno industrial en el cual no solo el operario que controla la cámara tiene acceso a ella, puede resultar peligroso que accidentalmente se cambie el trabajo correcto por uno equívoco, arruinando la producción del día.

Cabe decir que este diseño corresponde a la etapa previa de la programación, por lo que todas las consideraciones que fueron surgiendo con el paso del tiempo y que mejoraron la funcionalidad de la pantalla, se expondrán a medida que se vaya avanzando en el proyecto. No obstante, este boceto dio pie a crear una interfaz gráfica bastante parecida, por no decir idéntica.

2.4.2 Prototipo 2: Primer diseño (03/03/2017-16/03/2017)

Para esta segunda fase se pasó a realizar un primer diseño de la pantalla que serviría de apoyo en posteriores diseños. Se podría decir que era bastante fiel al boceto que se proporcionó, aunque con ciertas mejoras y consideraciones.

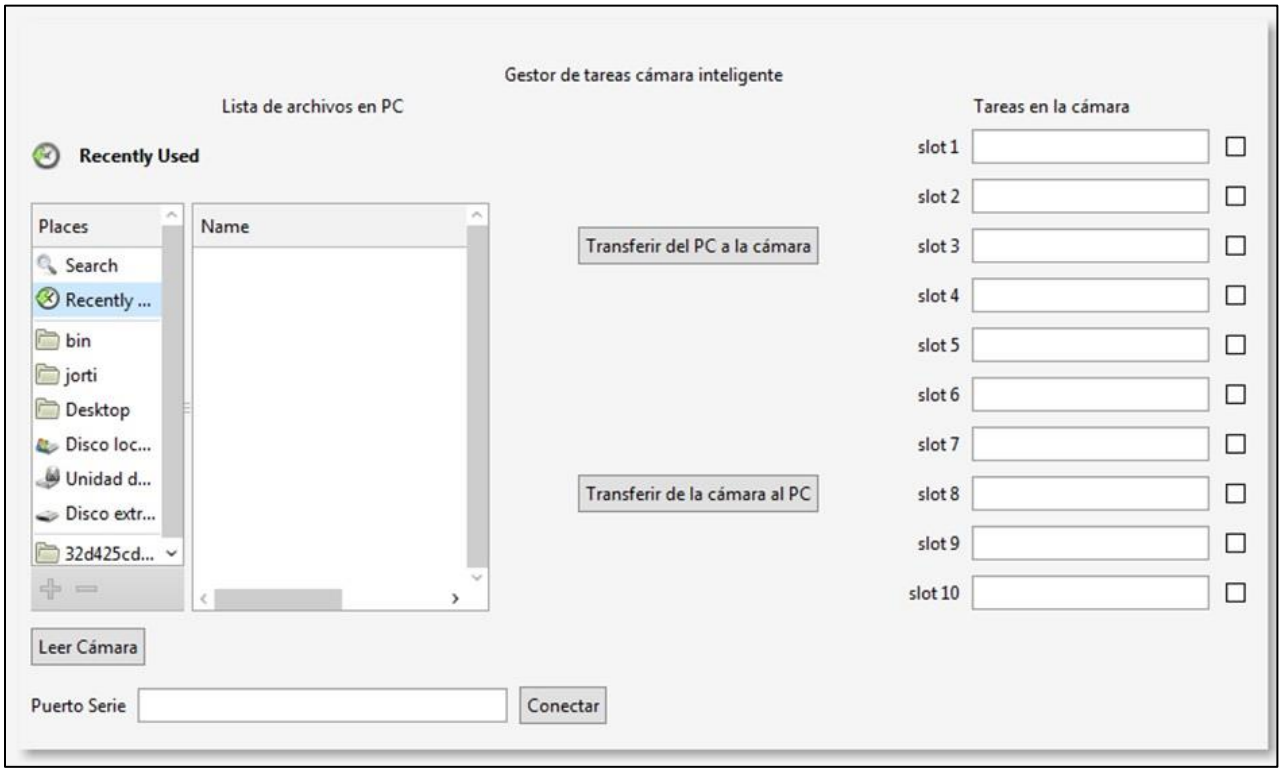


Foto 15: Pantalla de gestión de trabajos con puerto serie.

En esta primera pantalla es apreciable la gran similitud que tiene con el boceto que se proporcionó en primera instancia. Las dos diferencias más importantes que se pueden apreciar son en la búsqueda de trabajos dentro del PC y en la conexión con la cámara.

Se decantó por poner un buscador al estilo explorador de Windows en vez de una lista por la simple razón de que es mucho más visual este estilo. De la otra manera se debía introducir manualmente la ruta de los archivos, lo cual era muy engorroso y poco intuitivo para el usuario, requiriéndose además un recuadro extra donde introducir la ruta.

El segundo cambio es la introducción manual del puerto serie. Como se ha mencionado en etapas anteriores, la transferencia de trabajos del PC a la cámara resultaba imposible hacerlo mediante Telnet, por lo que se tenía que utilizar el puerto serie. En el recuadro indicado para ello se debía introducir el nombre del puerto (COM1, COM2, COM3…) y a continuación clicar en conectar. No obstante también era necesaria la conexión Ethernet a la hora de cambiar de trabajos o recopilar información acerca de ellos.

No obstante esta versión nunca se llegó a probar, ya que se encontró la solución al problema de la transferencia a través del método que ya se ha explicado, por lo que la idea del puerto serie

se desechó. Ahora solo se necesitaba realizar conexiones Telnet y FTP para llevar a cabo el proceso, lo que llevó a la creación de una nueva pantalla.

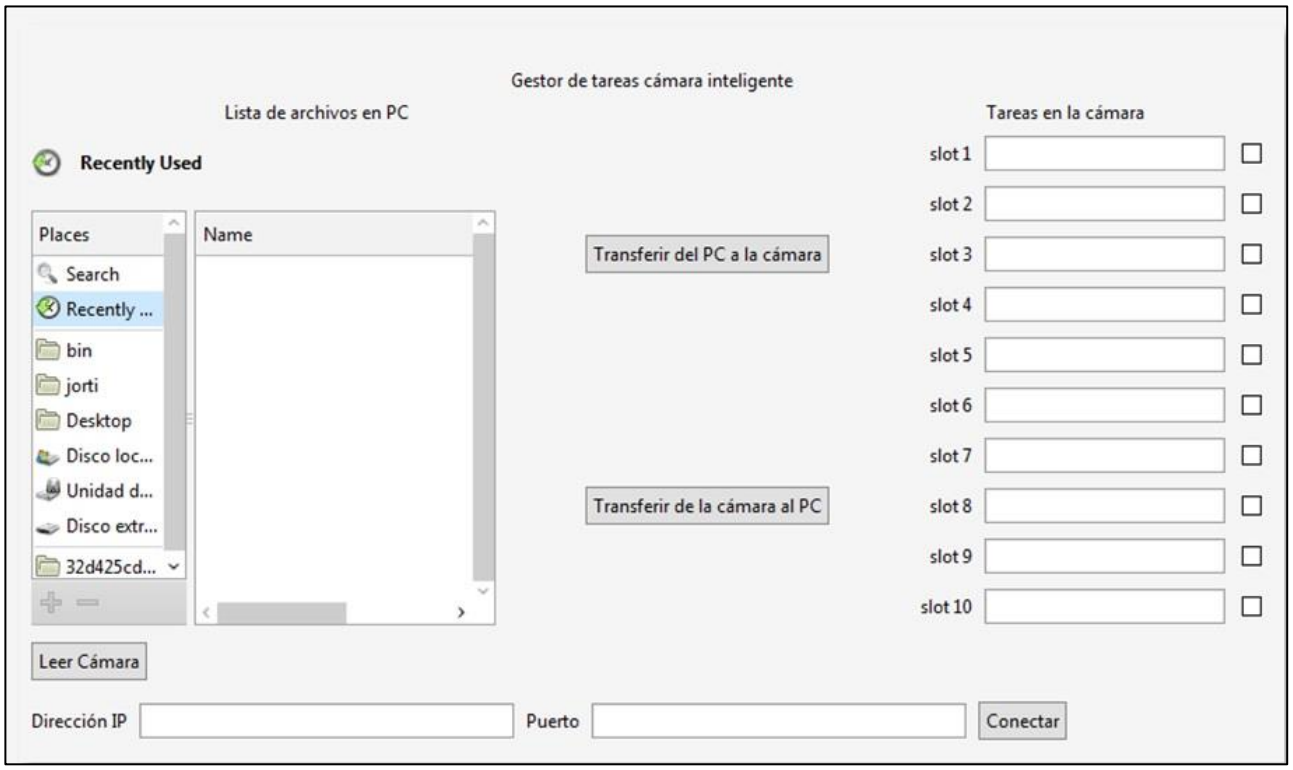


Foto 16: Pantalla de gestión de trabajos con Telnet.

En este caso el cambio en el diseño es simplemente en la introducción de la IP de la cámara y el puerto al que se va a conectar, dependiendo de la acción que se desee ejecutar. Para el traspaso de la cámara se utilizará el puerto 23 que es el que se utiliza por defecto, por lo que tampoco sería necesario escribirlo. Para la recopilación de información de los trabajos incluidos dentro de la cámara y para cambiar de trabajos en ejecución, se utilizará el puerto 49211.

No obstante, como se puede deducir, esta pantalla no pasó de ser un simple diseño ya que nunca se llegó a programar enteramente. Si bien la ejecución de los comandos Telnet y FTP manualmente era relativamente sencilla pero larga, a la hora de programar en C# es bastante complicado, tanto por las estructuras utilizadas como por las clases que se definen. Es por ello que es recomendable utilizar “forms” o librerías ya hechas que se puedan integrar dentro del proyecto y que resuelvan el problema de la complejidad de las conexiones remotas.

En resumidas cuentas, en esta fase del proyecto se tenía claro que la aplicación se iba a volcar en la transferencia vía Telnet y FTP, aunque no se sabía exactamente el modo a emplear para resolver ese requisito, y por lo tanto el resto de la programación (botones, entradas y toggles) queda pendiente de este paso previo. Además, como se adelantó en el anterior apartado, sería necesario implementar algunas pantallas auxiliares con el fin de mejorar la navegación por la interfaz.

2.4.3 Prototipo 3: Primer diseño funcional (17/03/2017-10/04/2017)

Es en este segundo diseño cuando se empieza a perfilar lo que va a ser la interfaz gráfica que se desea. Se incluyen diversas pantallas auxiliares, 5 en concreto, de las cuales se hablará en detalle en el modelo definitivo. A continuación se muestra la nueva pantalla:

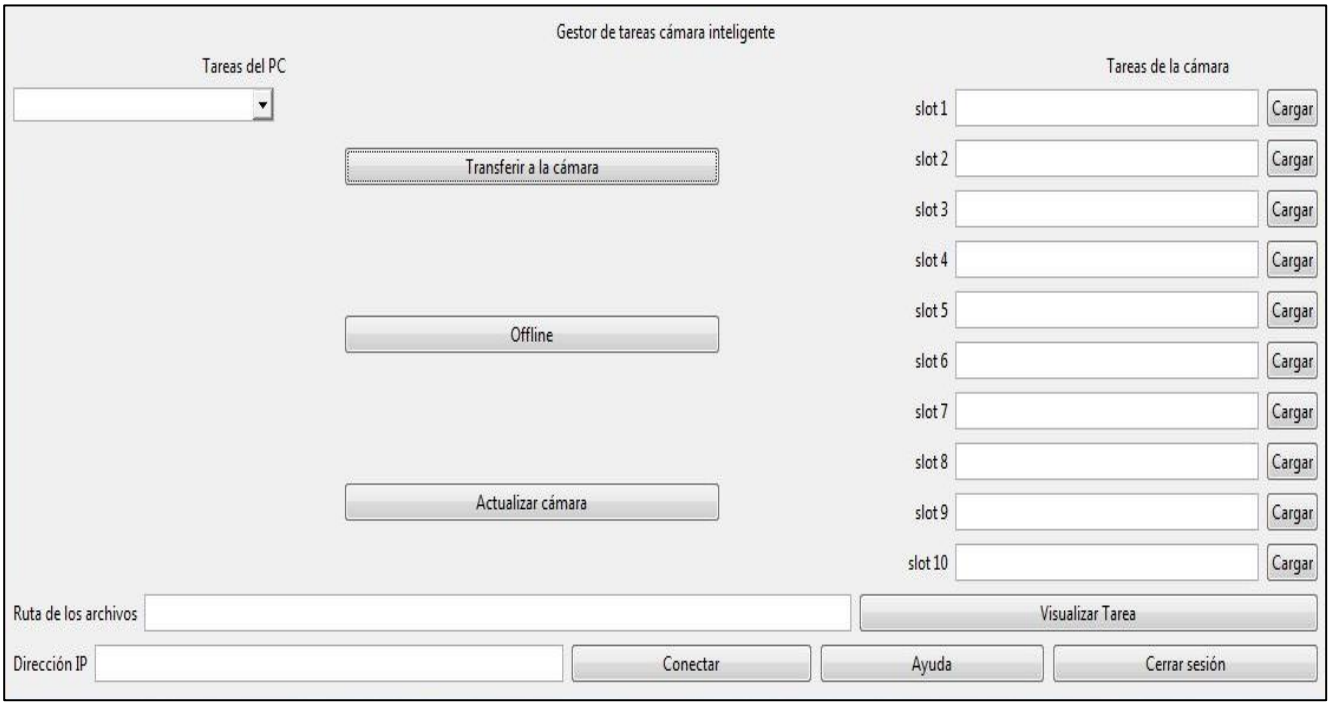


Foto 17: Primer diseño funcional de la pantalla de gestión de trabajos.

A simple vista se observan cambios profundos en la pantalla, tanto en botones como en funcionalidades. Seguidamente se comentará de izquierda a derecha y de arriba abajo las nuevas características de esta pantalla. Primeramente se encuentra un desplegable donde se intuye que están incluidas los trabajos de la cámara almacenados dentro del PC. El usuario deberá desplegarlo y seleccionar el trabajo que desee, para poder traspasarlo a la cámara a continuación. Esto de cara al operario resulta muy útil, ya que se pueden ver de una todos los archivos almacenados en la carpeta que este indique más abajo. Por otro lado puede suponer un problema que haya demasiados trabajos almacenados en una sola carpeta, ya que puede ser muy difícil de localizar el trabajo deseado. Como no, este entre otros debe ser un aspecto a tener en cuenta en futuros prototipos del software.

Pasando ahora a los botones que se encuentran en el centro de la pantalla, se empezará por uno de los más importantes del proyecto: El botón de transferir a la cámara. Anteriormente se comentó que la complejidad del uso de Telnet en C# hace que sea bastante recomendable que se recurra a librerías o “forms” de terceros. Pues bien, finalmente se encontró una librería que casaba perfectamente con las necesidades del proyecto, fácil de integrar y de usar. Se trata de la librería Minimalistic Telnet, creada por Tom Janssens, fundador de Virtual Sales Lab, que se encuentra en el anexo. Esta librería reduce la conexión Telnet a través de C# a una instrucción en la que se indica la IP del dispositivo y el puerto al que se desee conectarse, y a ciertos comandos

para mandar las ordenes a la cámara y recibir su respuesta. Antes de seguir, es importante mencionar que no se puede acceder a ninguna función de la pantalla, excepto al botón de ayuda, sin haber conectado primeramente con la cámara a través de su IP, por lo que como los demás botones, la función de transferir a la cámara necesita de una conexión previa. Continuando con la explicación, una vez conectado con la cámara y apretado el botón se desplegará una pantalla (Foto 2) en la cual se tendrá que introducir el slot al que transferir el trabajo previamente seleccionado.

Foto 18: Pantalla intermedia para el traspaso de trabajos.

Una vez elegido el slot deseado, se deberá pulsar aceptar para iniciar la transferencia. Sobra decir que esta no se podrá realizar a menos que se haya seleccionado antes el trabajo en el desplegable de la esquina superior izquierda, y se haya introducido un número de slot del 1 al 10. De lo contrario saldrá un desplegable avisando acerca de uno de estos dos problemas. Si se ha introducido todo correctamente, se ejecutaran los comandos Telnet y FTP tal y como se expone en el apartado x, en un tiempo aproximado de unos 20 segundos, que se mostrará en la barra de progreso que haya en la parte inferior de la pantalla. Una vez finalizado la pantalla auxiliar se cerrará y se volverá a la interfaz original.

El botón offline que aparece justo debajo del botón transferir a la cámara, no tiene más misterio que el de parar las inspecciones de la cámara. Es un botón que, a pesar de no estar en el diseño original de la pantalla, aporta comodidad al operario a la hora de realizar cualquier tarea dentro de la interfaz, sin tener que ejecutar ningún comando en Telnet.

El siguiente botón es Actualizar cámara, cuya función es actualizar la lista de trabajos que se encuentran en la parte derecha. Este botón es realmente necesario para saber en cada momento cuales son los trabajos que se encuentran en cada slot, sobre todo al iniciar la aplicación y al realizar una transferencia de trabajo. Es importante recalcar que cada vez que se ejecuta la aplicación se actualiza la lista de trabajos en la cámara, por lo que no es necesario apretar el botón. Sin embargo, para los casos en los que se ha realizado una transferencia de trabajo, se le recuerda al usuario que antes de ejecutar un trabajo de la cámara actualice la lista por posibles incoherencias de los trabajos mostrados en la ella.

Yendo hacia la derecha se encuentra la lista de los trabajos con sus correspondientes botones de carga a su lado. En este diseño se ha procurado que se identifique claramente que, si se pulsa el botón de carga que se encuentra al lado del recuadro con el nombre del trabajo, este trabajo sea el que se cargue obviamente. Como se avanzaba, el proceso de carga suele tardar unos 10

segundos en la situación más desfavorable, por lo que para hacerlo de manera más gráfica se utilizará una barra de progreso que indique al usuario cuando se ha cargado el trabajo por completo. Además, para que no resulte en ninguna confusión, se ha implementado en la programación que se haga un cambio de color en el recuadro del trabajo que se haya cargado, siendo blanco el color por defecto cuando el trabajo no se está ejecutando, y verde cuando sí.

Aunque no se haya mencionado con anterioridad, en el diseño solo aparecen unos 10 slot. El motivo de este número es que a partir de 11 slots o más, la velocidad de procesamiento de la cámara cae en picado, por lo que es conveniente que se limite su número a solo 10 para que no afecte a los tiempos calculados dentro de la programación, es decir, los 20 segundos en el caso de la transferencia de trabajos y los 10 segundos para su carga. Tampoco supondría ningún problema a la hora de ejecutar trabajos, ya que en unos 30 segundo se puede pasar uno del PC a la cámara, que además es el cometido de la misma.

Pasando ahora a la ruta de los archivos en la parte inferior izquierda, es notable la sustitución del buscador estilo Windows, más intuitivo, por una entrada de texto donde el usuario deberá escribir la ruta en la que se encuentra el trabajo y pulsar conectar al mismo tiempo que introduce la IP de la cámara. Una vez conectado, en el desplegable de los trabajos del PC aparecerán los trabajos que se encuentran en esa ruta, o bien aparecerá un mensaje diciendo que el directorio no existe. Es obvio que con respecto al diseño que se había realizado en un principio es un paso atrás, ya que se pierde en sencillez y en capacidad visual. Se pasa de poder ver en tiempo real donde están los trabajos a través de iconos a una simple entrada con nada de lo mencionado anteriormente, y que está condicionada a los fallos que pueda tener el usuario a la hora de escribir la ruta. No obstante, hay una razón que obliga a tener que usar la entrada de datos manual. Y es que a la hora de utilizar los comandos FTP para traspasar trabajos se necesita escribir la ruta manualmente, por lo que de momento hace que esa sea la única solución por el momento, aun sabiendo que no debería ser una solución definitiva. También hay que tener en cuenta que el primer diseño solo se utilizó como referencia o boceto, por lo que no era representativo del prototipo final, al menos en esta fase.

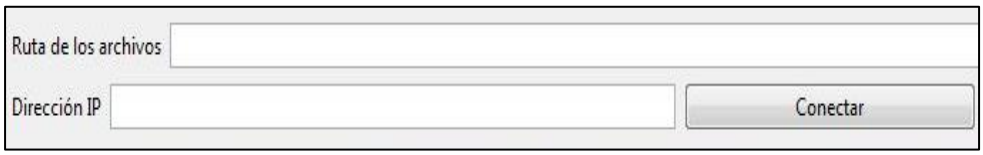


Foto 19: Detalle de la búsqueda de trabajos en el PC.

Siguiendo con el tema de la conexión con la cámara, más abajo se encuentra el botón de conectar justo a la derecha de la entrada de texto la dirección IP. Tan simple como introducir la IP de la cámara junto con la ruta de los archivos como se explicó en el segundo párrafo y pulsar conectar. Si se ha llegado a conectar con la cámara, el usuario recibirá un mensaje diciendo que la conexión ha resultado exitosa. De lo contrario, el mensaje que obtendrá será que no se ha encontrado la IP introducida. Ya se ha mencionado anteriormente que la cámara debe de estar conectada para que se puedan acceder a las funcionalidades de la pantalla, por obvias razones. También decir que no es preciso introducir la ruta de los archivos a la hora de conectar con la cámara, pero sí lo es si el propósito es transferir trabajos a la cámara.

Justo a continuación de la ruta de los archivos se encuentra el botón visualizar Tarea, el cual redirige al visor web en el cual se puede observar la tarea sin necesidad de abrir AutoVision, tal y como se describió en los apartados anteriores. El uso de Cloud Link es suficiente para la aplicación, ya que lo que se pretende en todo momento es que se use como un gestor de trabajos y no como un editor, ya que esa tarea concierne al departamento de soporte técnico, bien de la empresa misma o del distribuidor de las cámaras como puede ser Intertronic. Esta pantalla del Cloud Link sería interesante que para un futuro se pudiese modificar el "layout" para personalizarlo según el cliente desee.

Para finalizar con los elementos de la pantalla, en el margen inferior derecho se encuentran los botones de Ayuda y Cerrar sesión. El primero es un botón que genera una pantalla en la que se leen unas breves instrucciones de cómo utilizar la pantalla en caso de que el operario tenga sus dudas. La función del botón cerrar sesión es lógica: Cierra la pantalla una vez el operario acaba de realizar el trabajo.

Problema con la transferencia de archivos al PC

Si hay algo que se echa en falta en la pantalla con respecto al boceto que planteó en un primer momento Intertronic, es la transferencia de la cámara al PC. En un diseño previo, se incluyó el botón en la pantalla para poder ejecutar la orden, pero se obtenían unos resultados bastante aleatorios.

El trabajo se realizaba con éxito con los trabajos que previamente se habían incluido en la memoria Flash temporal, es decir, los que se habían traspasado del PC a la cámara. Sin embargo esta memoria temporal desaparecía cuando la cámara se apagaba, naturalmente por el hecho de ser temporal, eliminando así todos los trabajos que se podían haber incluido en ella. Esto supone un problema ya que la única razón por la cual se pensó en utilizar el botón fue para recuperar los datos que habían almacenados dentro de la cámara, bien para archivarlos o modificarlos. Al solo poder acceder a los trabajos que se acababan de pasar del PC a la cámara, esta función no servía para nada porque los trabajos que se introducían en la memoria flash ya estaban en el PC.

Posteriormente haciendo una conexión FTP con la cámara, dentro del directorio Jobs se encontraron los trabajos almacenados en la cámara, lo que en un principio parecía resolver el problema. Sin embargo existía un inconveniente, y es que los trabajos que se obtenían de ese directorio no eran en formato .avz, como ocurría con los trabajos que se transmitían a la cámara. En este caso eran en formato .avp, con el nombre cambiado según el slot en el que se encontraban. Esto es debido a que cuando se realiza el proceso de transferencia a la cámara, con el comando "FlashAvz" que se utilizaba en la sesión Telnet, se descomprime el archivo .avz, quedando así un .avp más las herramientas incluidas dentro del mismo trabajo (contornos, contadores de blobs, OCR…). Si se tiene en cuenta que normalmente el operario querrá tener todos los archivos en el mismo formato y en la misma carpeta, hace que no haya más remedio que descartar la transferencia inversa. La solución que plantea Intertronic para el problema es que antes de poner a funcionar una cámara, se recopilen mediante AutoVision todos los trabajos que haya dentro de la cámara y se guarden en el formato adecuado si se quieren volver a usar posteriormente.

2.4.4 Prototipo 4: Diseño final (11/04/2017-12/05/2017)

Si bien el proyecto que se tenía a principios de abril era funcional y realizaba la gestión de trabajos perfectamente, sí que es cierto que era un diseño que distaba mucho de ser una versión final. Para empezar, no contaba con una pantalla de inicio de sesión, lo cual como se explicaba en el boceto, podía suponer un gran problema si en la planta había más de una persona que pudiera modificar la pantalla accidentalmente; no era suficientemente intuitiva para el operario, ya que por ejemplo se debía introducir la ruta manualmente y era difícil la búsqueda de los trabajos en el desplegable que se había implementado; y por último, solo estaba hecho para el sistema de visión VS-06 que, a pesar de tener una amplia gama de productos, no era ni mucho menos el sensor de visión más vendido por Intertronic. En este caso, dada su simpleza, velocidad de procesamiento y memoria interna entre otras muchas características, hacen que sea mucho más fácil de vender un sensor CS-50 que la ya casi obsoleta VS-06.

Obviamente, si se quería ser competitivo en el mercado la mejor solución pasaba por crear un software híbrido, capaz de trabajar a la vez con una VS-06 y con una CS-50. Esto es un factor clave para el cliente ya que los dos softwares son incompatibles en el mismo PC, haciendo que solo se pueda tener una versión en un mismo PC. Si en una misma instalación se tienen los dos modelos de Di-soric, cada cámara se debe de controlar con un ordenador distinto o bien con una máquina virtual en el mismo PC. En el caso de la transferencia de trabajos no es ningún problema, puesto que se realiza a través de Telnet y no de AutoVision, siendo en este caso totalmente compatibles las dos cámaras. Por esta razón es imperativo que se pueda manejar en la interfaz las dos cámaras.

Problema con las CS-50: La transferencia de archivos

Desde que se propuso hacer esta aplicación, la intención siempre ha sido enfocarla de cara a las cámaras VS-06, las cuales su factor limitante es la memoria. Al disponer de solo 6 MB se hace imprescindible un gestor rápido y eficiente. Sin embargo, y como se ha expuesto en apartados anteriores, las cámaras CS-50 no tienen esa problemática: Cuentan con una memoria interna que roza los 1,4 GB, lo que hace que se puedan almacenar en muchos casos infinitos trabajos.

No obstante sí que se propuso en un primer momento que se pudieran traspasar trabajos del PC a la cámara como sucedía con su antecesora. La idea era realizar el mismo proceso que en la VS-06 en la misma pantalla, lo que implica que no sería necesaria una pantalla extra. Sin embargo, las transferencias que se hacían hacia la cámara funcionaban de manera intermitente. Había veces que los trabajos se situaban en el slot especificado, pero otras muchas no se traspasaban correctamente. Tras múltiples intentos durante días, finalmente se decidió contactar con Di-soric para averiguar una posible solución. Sin embargo la solución que proporcionaron desde Alemania no respondía a la solución para el problema que se había planteado, sino que más bien propusieron una vía más lógica.

La solución que aportaron fue que al tener 1,4 GB de memoria, la CS-50 no precisaba del traspaso de trabajos que se realizaba en la VS-06, por lo que carecía de sentido que se llevara a cabo con la nueva cámara. Desde Intertronic se estudió el caso llegando a la conclusión de que, si bien se necesitaban muchos trabajos para un proceso, se podrían incluir todos en la cámara. No obstante, al disponer de tanta memoria, se hace necesario un buscador de trabajos dentro de la cámara. La idea para este buscador sería uno basado en la estructura del explorador de archivos

de Windows, con un desplegable con todos los trabajos de la cámara, una entrada para filtrar el trabajo por el nombre y un cuadro de texto en el que aparezca el trabajo que se haya seleccionado y si se está ejecutando o no.

A continuación se va a hacer una exposición de lo que es el modelo definitivo de la pantalla, tanto de las ventanas gráficas como de la programación que se ha utilizado en cada caso. También recalcar que hay dos modelos de pantalla para la CS-50: Uno correspondiente a un primer diseño en una fase más primeriza, y otro con el que se ha lanzado finalmente la versión 1.0 del software. En este caso se expondrán las dos interfaces y se compararán entre sí.

Pantalla de introducción: Ingreso de nombre, contraseña e IP de la cámara

Esta pantalla de inicio es una característica que se lleva buscando desde hace bastante tiempo, como se recalca en todo el proyecto. Si bien los motivos de su implementación son de sobra conocidos, no ha sido hasta esta fase final cuando se ha implementado. La estructura de la misma es la siguiente:

Foto 20: Pantalla de inicio de sesión.

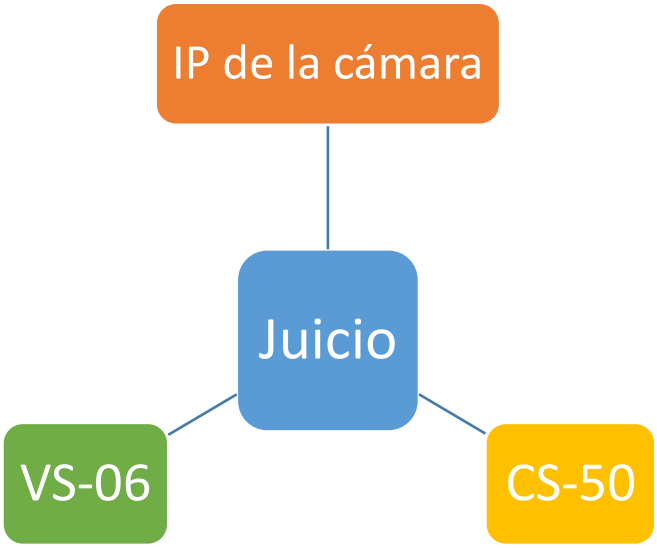
Como se puede apreciar, esta pantalla tiene todos los elementos típicos de una pantalla de inicio normal y corriente, salvo la IP de la cámara propia de este modelo. Su funcionamiento no tiene ningún misterio más allá de que se deba introducir todos los campos correctamente. En este caso se ha decidido implementar dos modos de funcionamiento: Operario y Demo.

Empezando con el modo demo, el nombre ya implica que es un modo de no mucha seguridad, en el cual se necesitan menos privilegios que para el modo operario (gestión de los trabajos). En este modo se abre en el explorador la página correspondiente al HMI online de la cámara, llamado CloudLink. Además de ello se despliega una pantalla de pequeño tamaño con dos

botones: Un botón de Online para iniciar las inspecciones y otro offline para pararlas. Resulta de mucha utilidad en el caso de que haya un problema en la línea y se desee parar las inspecciones para un análisis más detallado por parte del operario. Este modo es el indicado si se ha terminado de editar los trabajos que se encuentran en la cámara y se desea tener una visión más directa del trabajo que se esté ejecutando, por lo que en condiciones normales, este será el modo que esté más tiempo en marcha durante la producción.

Siguiendo con el modo operario, o modo edición como ya adelantábamos, no es ni más ni menos que la interfaz con la que se ha estado trabajando desde el principio, es decir, la gestión de trabajos. Obviamente será un modo con mayor seguridad que el modo demo, por las funciones críticas que desempeña la pantalla en este caso.

Al ser una pantalla totalmente nueva, implica que sea una novedad para el proyecto en sí misma. No obstante hay un elemento que antes estaba en la pantalla de edición y que finalmente se decidió colocar en esta: La IP de la cámara. El principal motivo por el que se decidió migrar esta entrada a la pantalla de inicio es la introducción de la nueva cámara en la aplicación. El hecho de que las dos cámaras utilicen una forma de trabajar distinta, hace que sea imperativo realizar un juicio previo para saber el modelo que se está utilizando. Por ello mediante comandos Telnet, la aplicación podrá ver cuál es la cámara correspondiente a la IP que se ha introducido, llevando de esa manera a una interfaz u otra.



Más allá de lo que supone la IP de la cámara en esta pantalla, los elementos de la misma son simples y corriente: Una entrada para el usuario que en este caso será “operario” o “demo”; una entrada para la contraseña, una para cada modo; la entrada de la IP de la cámara como se ha comentado en el párrafo anterior; un botón de aceptar que sirve para realizar una comprobación del usuario, la contraseña y la IP, para posteriormente entrar en la pantalla correspondiente al modo; un botón de cerrar para salir de la aplicación y finalmente un botón de ayuda que despliega

una ventana que informa de que pasos hay que seguir en cada momento, útil para usuarios primerizos en la aplicación.

La programación de esta pantalla es bastante sencilla, ya que su función a desarrollar no es tan compleja. Primeramente, tras declarar las variables tipo cadena para las entradas de usuario, contraseña e IP, se procede a codificar la acción del botón aceptar, el cual va a realizar el juicio y la selección de una pantalla u otra dependiendo del modo. En primera instancia comprueba que el usuario y la contraseña son correctos dentro de un condicional "if": Si el usuario y la contraseña son "Demo" y "1234" respectivamente, se entrará en el modo demo; si por el contrario se introduce "operario" y "987." se entrará en el modo edición. Obviamente si no se ha introducido ni usuario ni contraseña, deberá aparecer un mensaje advirtiendo sobre ello. Cabe decir que en todo momento se pueden cambiar las contraseñas de uno y otro modo por unas de mayor o menor seguridad, dependiendo del cliente.

```
if (contraseña == "987." & usuario == "operario")
else if (contraseña == "1234" & usuario == "demo")
else if (usuario == "" || contraseña == "")
```

Dentro de cada condicional se procederá a comprobar que la IP de la cámara está escrita, y si lo está comprobar que existe esa IP. Para ello se utilizará el comando ping en el cmd de Windows, iniciando un nuevo proceso con la librería "System.Diagnostics". El comando a utilizar es el comando "ping", el cual envía paquetes de datos a la cámara comprobando que los recibe y mostrando una estadística donde aparecen los paquetes perdidos y los enviados correctamente. Se procurará que el proceso no aparezca por pantalla, sino que actúe en segundo plano, por lo que habrá que deshabilitar la visibilidad del cmd. Los datos que se vayan recibiendo del cmd se irán incluyendo dentro de un archivo de texto, que más tarde se leerá para comprobar que efectivamente se ha conectado con éxito a la cámara. Leyendo ese archivo línea por línea, el indicativo de que se ha conectado bien es que aparezca la frase "Respuesta desde (IP de la cámara)", por lo que si se encuentra dentro del archivo la pantalla mostrará un mensaje con la información de que se ha conectado satisfactoriamente. De lo contrario desplegará el mensaje de que no se ha encontrado la IP. Mencionar que todos los mensajes que aparecen por pantalla se llevan a cabo a través del comando "MessageBox.Show" de la librería "System.Windows.Forms", que hace que las ventanas que aparecen tengan una apariencia típica de Windows.

```
MessageBox.Show("Rellene todos los campos.", "Error",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
```

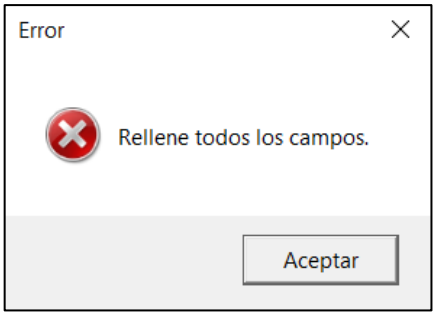


Foto 21: Mensaje de error suministrado por la pantalla.

Pantalla de juicio: Usuario y contraseña Telnet

Antes se ha hablado del juicio que hacía la aplicación para saber si la IP que había introducido el usuario era la IP de una CS-50, o bien la IP de una VS-06. Técnicamente de eso se ocupa la pantalla de la que se va a hablar a continuación. Se trata de la pantalla previa a las interfaces de gestión de trabajo de ambas cámaras, en la cual se debe introducir el usuario y la contraseña Telnet. Estos credenciales, si bien en las VS-06 no era necesario introducirlos (aunque sí que se podían implementar), sí que lo son en las CS-50, haciendo que la conexión Telnet sea más segura. Esto deja un problema a resolver del cual se pueden sacar dos soluciones. El hecho de que se necesite un usuario y una contraseña para poder entrar en Telnet deja la posibilidad abierta a que se pueda averiguar por esa vía el modelo de la cámara que esté conectada, gracias al comando "BP-Dump()".

Foto 22:Pantalla de comprobación de credenciales Telnet.

Como es apreciable, es un diseño muy simple de características muy similares a la pantalla de inicio. Las únicas diferencia que puede haber entre ambas es que en esta pantalla no es necesario introducir la IP de la cámara en cuestión. En el usuario Telnet habrá que introducir lógicamente el usuario, que por defecto para las cámaras de Di-soric es "target", y en la contraseña Telnet la contraseña que si no se modifica por el usuario será "password". En el caso de pulsar aceptar, se comprobará que se han introducido los credenciales, y si estos son correctos o erróneos. Si de hecho son correctos, se pasará a la ventana correspondiente de edición de trabajos. En caso de apretar el botón cancelar se volverá atrás a la pantalla de inicio.

Hablando de la programación de la interfaz, difiere en muy poco con la primera que se ha comentado. Las estructuras a emplear son las mismas con pequeñas diferencias. Nada más aparecer la pantalla se despliega un mensaje como el que se ha visto al estilo "Windows.Forms", en el que se indica al usuario que debe introducir el usuario Telnet además de la contraseña. En el caso de que no tenga contraseña, se debe dejar ambas entradas de texto en blanco, aunque en realidad no supondría un problema que se introdujeran ya que no afectarían a los comandos realizados con posterioridad. Así pues, una vez introducidos los credenciales y pulsado el botón aceptar, se entrará dentro de tres condicionales: Uno es el correspondiente a la presencia de usuario y contraseña escritos por el operario, otro si no se ha escrito ni usuario ni contraseña y otro si se ha escrito en un campo pero no en otro.

```
if (Usuario != "" && Contraseña != "")
else if (Usuario == "" && Contraseña == "")
else if (Usuario == "" && Contraseña != "" || Usuario != "" &&
Contraseña == "")
```

Si el usuario ha escrito los credenciales, primeramente hace una conexión Telnet mediante la librería "Minimalistic Telnet", de la cual se habló brevemente en el segundo prototipo desarrollado, conectando con la IP introducida en la pantalla de inicio en el puerto 23 (puerto Telnet por defecto). Siguiendo con la introducción de comandos, en este caso tanto el usuario como la contraseña, se comprueba que son correctos gracias a la captura del texto mostrado por el "Shell" en un archivo .txt, el cual mostrará un "login incorrect" si no es correcto y un cursor si es correcto. Obviamente en el caso de que sea incorrecto se mostrará un mensaje que dé muestra de ello, mientras que en el caso de que sea correcto se introducirá el comando "BP_Dump()" en el que se muestra el modelo de la cámara. Seguidamente, se volverá a capturar el texto resultado del comando dentro de un archivo de texto y, leyendo línea a línea, se podrá por fin conseguir el modelo de la cámara, dando pie a una pantalla u otra.

```
TelnetConnection tc2 = new TelnetConnection(MainWindow.IP, 23);
Console.WriteLine(tc2.Read());
tc2.WriteLine(Usuario);
Console.WriteLine(tc2.Read());
tc2.WriteLine(Contraseña);
Console.WriteLine(tc2.Read());
tc2.WriteLine("BP_Dump()");
```

En el caso de que no se haya introducido los credenciales, se realizará directamente la conexión Telnet con la IP tal y como se ha explicado en el anterior párrafo. El proceso de comprobación, en lugar de ser antes de ejecutar el comando, se realiza después, con lo que se compara los tipos de respuesta que proceden. Si en el archivo de texto se lee "CS-50", se irá a la pantalla correspondiente, al igual que si se lee "VS-06". En el caso de que no se lea ni una cosa ni la otra, solo puede significar que la cámara necesita un Login, por lo que como en la mayor parte de los casos de la aplicación, se mostrará un mensaje diciendo que necesita de Login.

Por último, y el caso que se debe evitar en todo momento es que se haya escrito el usuario y no la contraseña y viceversa. Obviamente es una situación que no se puede dar, ya que o está protegido o no lo está. En este caso vuelve a aparecer un mensaje indicando que se escriba en los dos campos o que se dejen los dos en blanco.

Con los datos obtenidos previamente de la pantalla de inicio y los obtenidos en esta pantalla, la aplicación dirige al operario al modo de edición adecuado. Esto supone una gran ventaja en cuanto a simplicidad, ya que el operario no tiene que seleccionar en ningún momento el modelo de la cámara, sino que la aplicación lo sabe automáticamente. Puede que a simple vista, la pantalla que se acaba de exponer sea inútil en el sentido de que se usa siempre el mismo Login. No obstante el cliente en todo momento puede cambiar la contraseña Telnet, por lo que se antoja imposible que se suprima esta pantalla.

Pantalla de edición de la VS-06: Mejoras en el modelo

Antes en el prototipo 2 se mostró una pantalla de gestión de trabajos para la VS-06 bastante completa. Si bien realizaba las labores de traspasar y cargar trabajos a la perfección, no estaba suficientemente pulido como para aceptarlo como un diseño final. La modificación de esta pantalla ha ido encaminada a hacerla mucho más intuitiva para el usuario, al fin y al cabo el que la va a utilizar más a menudo. Si cada vez que el operario desee cambiar de carpeta del PC para elegir un trabajo que traspasar a la cámara tiene que escribir la ruta completa en un cajetín, hace que aumente exponencialmente el tiempo que está parada la línea, y en consecuencia la empresa pierde dinero. Otro de los puntos a tratar era la búsqueda de trabajos en el caso de que la carpeta contenedora de los mismos esté llena, por ejemplo con unos 200 trabajos. Lo que se pretende es que el usuario, en lugar de tener que introducir manualmente un nombre que puede que no sea el correcto, pueda visualizar todos los trabajos incluidos en el PC de manera mucho más cómoda a través de una ventana.

Es posible que la ventana en sí no haya sufrido cambios drásticos como se pudiera esperar, ya que el 90% de los elementos que aparecen ya estaban en el prototipo 2. Sin embargo ese 10% de cambio puede suponer un cambio sustancial en el tiempo que se tarda en la gestión de trabajos, traducido a un ahorro considerable de dinero, en tanto en cuanto la empresa produce más tiempo.

Sin más preámbulos, a continuación se mostrará el aspecto actual de la pantalla de gestión de trabajos de la VS-06, además de unas pantallas complementarias para orientar de una mejor manera al usuario.

Como ya se había adelantado, los cambios que más se pueden apreciar en la pantalla son la

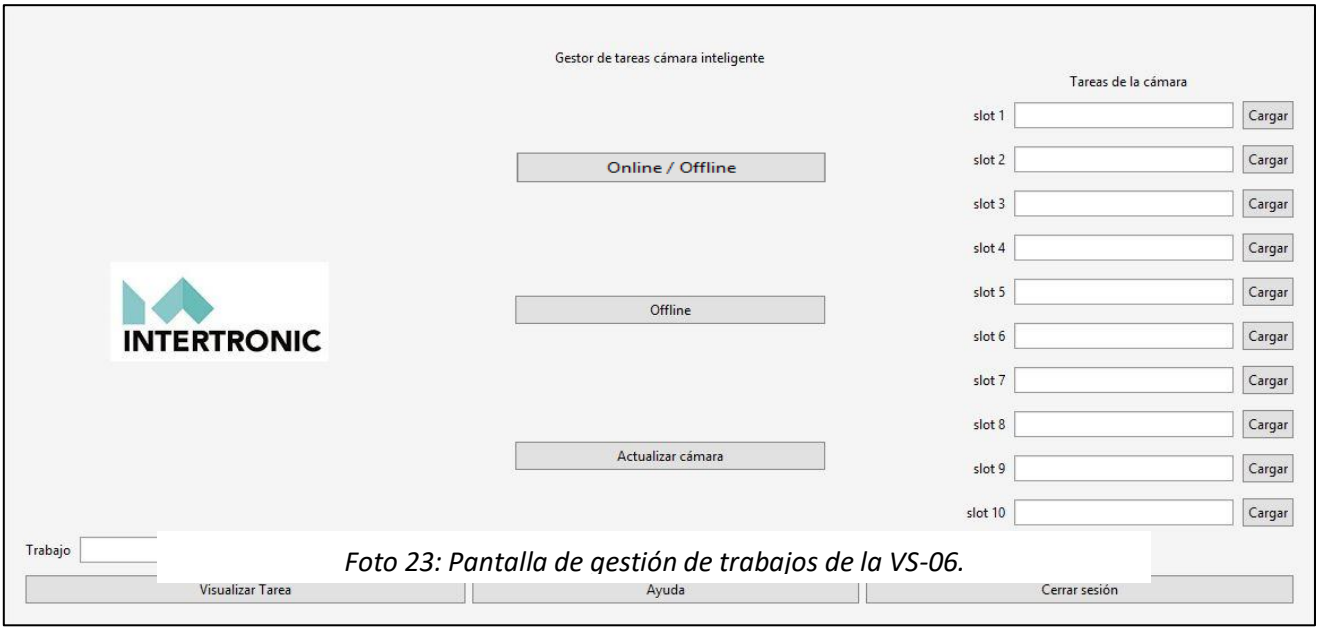


Foto 23: Pantalla de gestión de trabajos de la VS-06.

desaparición del desplegable en la parte superior izquierda de la pantalla, la inclusión del logo de Intertronic y la modificación del recuadro donde el usuario tenía que introducir la ruta de los

archivos. El resto, los botones y los recuadros donde se encuentran almacenados los trabajos son iguales que en el prototipo anterior, por lo que solo se comentaran las novedades que trae la interfaz.

Primera y principal, la búsqueda de trabajos en el PC, la cual está ahora situada en la parte inferior izquierda de la pantalla. El diseño se ha inspirado en los buscadores más conocidos, donde al lado de la caja de búsqueda se encuentra un botón con unos puntos suspensivos el cual abre un explorador similar al de Windows.

En él se pude observar a la izquierda una serie de carpetas que van desde los sitios vistos

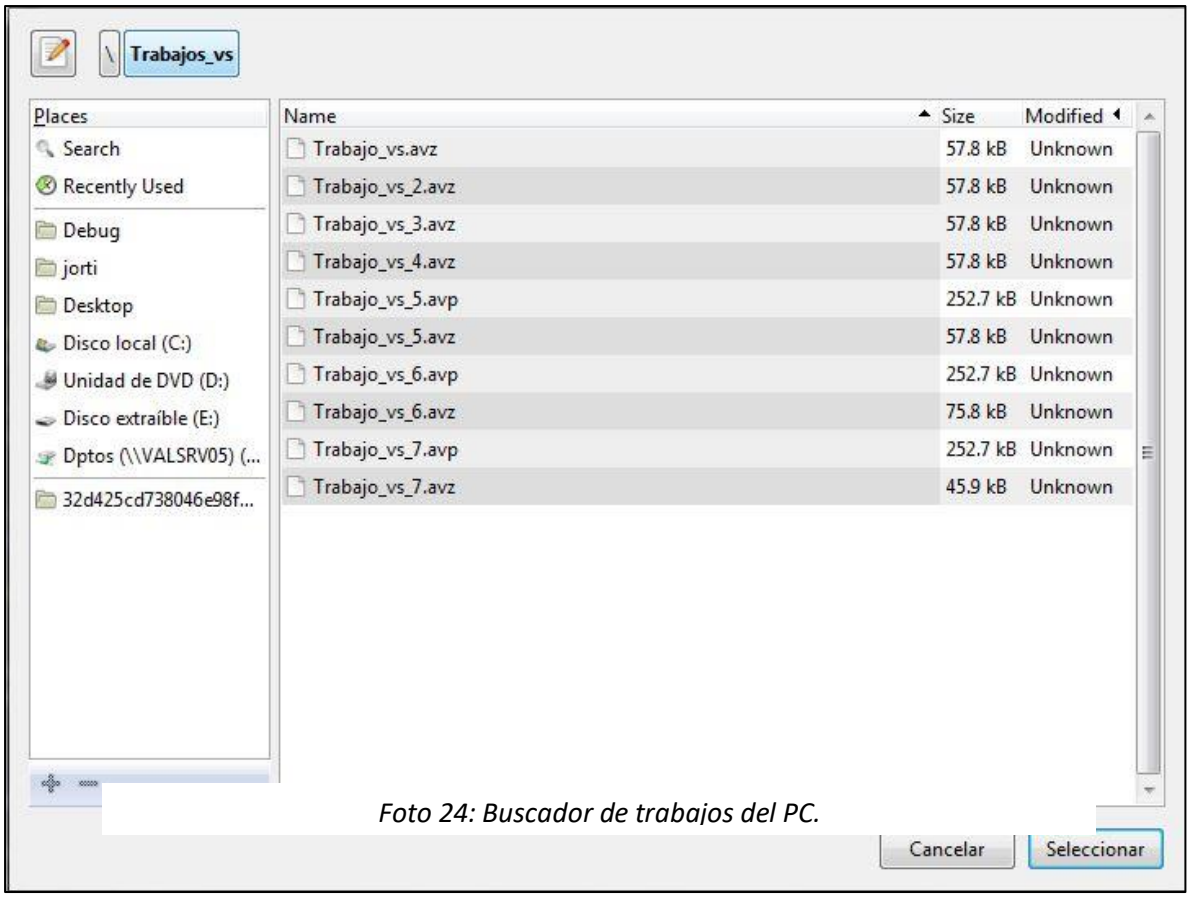


Foto 24: Buscador de trabajos del PC.

recientemente, hasta los discos que haya en ese momento en el ordenador. En este caso se encuentra en una carpeta que contiene los trabajos de la VS-06, tanto ".avp" como ".avz". Esto supone un inconveniente, ya que solo los trabajos que contengan la extensión ".avz" serán los válidos, por lo que hay que impedir que de alguna manera el usuario pueda seleccionar un trabajo con un dominio distinto, como se explicará más adelante en la programación. Esto lleva a otro problema: Los trabajos que se generan en el AutoVision tienen la misma extensión siempre independientemente del modelo de cámara escogido, por lo que desde el menú que se está enseñando es imposible distinguir con qué modelo se ha creado. Es por eso que es altamente recomendable que se separen los trabajos en carpetas según modelos, ya que de esa manera no habrá confusión a la hora de seleccionar el trabajo para cada modelo de cámara. De lo contrario, si

no coincide el modelo de la cámara con el modelo con el cual fue creado el trabajo, la cámara dejará de funcionar correctamente, siendo necesario resetearla por completo, borrando todos los trabajos incluidos dentro de ella. Obviamente, si solo se dispone de un modelo de cámara no será necesario que se almacenen en carpetas diferentes, a no ser que sea preferencia del cliente.

Volviendo al menú de selección de trabajos, se tiene que indicar al usuario que un trabajo no tiene el formato adecuado a través de un mensaje, como viene siendo habitual en todo el proyecto, tal y como se muestra a continuación.

Foto 25: Mensaje de error al elegir un formato de archivo erróneo.

Si por el contrario el trabajo es el correcto, se cerrará el menú y en la ruta del archivo seleccionado aparecerá en la barra de búsqueda, la cual se podrá utilizar cómodamente más tarde a la hora de realizar la programación.

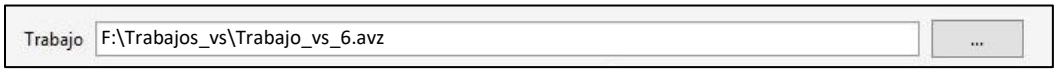


Foto 26: Muestra de la ruta de la cámara al seleccionar el trabajo.

Otro cambio que puede que resulte inapreciable pero supone un cambio importante, es el del botón de Online/Offline. Anteriormente solo se podía parar el trabajo cuando el mismo estaba en ejecución, lo cual era bastante cómodo si se pretendía interrumpir la inspección de manera rápida en caso de parada breve. Sin embargo entrañaba un problema, y es que solo hacía la parada de las inspecciones y no la reanudación, por lo que para volver a iniciarlas hacía falta volver a cargar el mismo trabajo otra vez, con la consecuente gran pérdida de tiempo. Es por ello que se implementó la función de Online para el mismo botón, por lo que solo es necesario apretarlo otra vez en el caso de que se quieran reanudar las inspecciones. Además, el sistema es suficientemente inteligente para saber en qué momento se necesita hacer un Online o un Offline.

Por último, una característica que se ha implementado en este último modelo es la posibilidad de hacer trabajo de inicio el trabajo que se esté ejecutando. Para las cámaras de Disoric existe la posibilidad de seleccionar un trabajo que se ejecutará cuando se arranque la cámara, llamado “boot job”. Esta opción se puede activar o bien desde el propio AutoVision o a través de un comando Telnet. Obviamente, de querer utilizar esta opción se haría a través de Telnet, que es lo que finalmente se ha realizado. La cuestión era donde podía seleccionar el usuario el trabajo de arranque. Se llegó a la conclusión de que una vez acabado de gestionar todos

los trabajos y al apretar cerrar sesión, sería lógico preguntar en ese momento si el trabajo que se encuentra en ejecución se quiere establecer como trabajo de arranque. Por ello en ese momento se desplegará un mensaje preguntando si se desea hacer "boot job" el trabajo que se está ejecutando, de la forma que aparece a continuación.

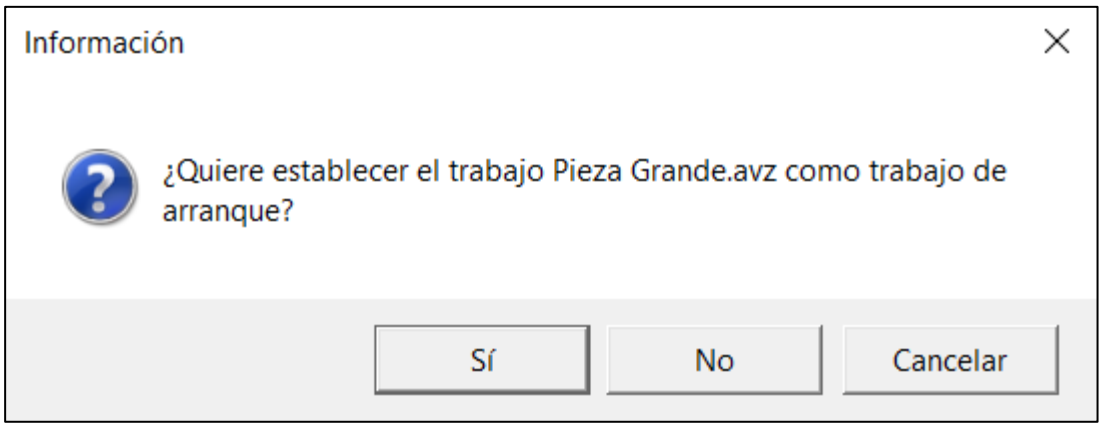


Foto 27: Mensaje para establecer el trabajo de arranque.

En el caso de que el usuario pulse "Sí", se establecerá como trabajo de arranque, mientras que si se pulsa "No", no lo hará. En el caso de que se apriete cancelar, solamente cerrará el mensaje continuando dentro de la interfaz de gestión de trabajos.

Pantalla intermedia: Port del PC a la cámara

Si bien es cierto que en los casos anteriores se ha comentado la programación de la pantalla antes de pasar a la siguiente, en este caso resulta imprescindible que se comente antes la pantalla intermedia de traspaso de los trabajos del PC a la cámara, para luego explicar toda la programación que conlleva las dos pantallas, casadas en cierta manera. Aquí abajo se muestra la misma pantalla, la cual se comentó con anterioridad, pero que ha sufrido ciertos cambios al igual que la principal.

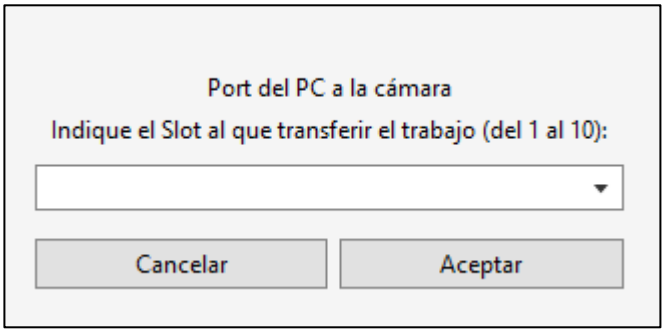


Foto 28: Pantalla selector de slots para el port a la cámara.

Como bien se aprecia en la nueva pantalla, se ha sustituido el cuadro de texto en el que se tenía que introducir el Slot por un desplegable en el que salen los números del 1 al 10, por lo que la selección del mismo ahora no puede implicar ningún error de introducción de caracteres.

También se ha suprimido la barra de carga que se encontraba en la parte inferior, ya que esta se ha movido a una independiente, la cual se comentará posteriormente para ver sus usos.

Obviamente, al pulsar aceptar se procederá al traspaso del trabajo seleccionado, siempre que se haya seleccionado un slot y el mismo trabajo. De lo contrario mostrará el mensaje dando parte de ello. Por su parte el botón cancelar cerrará la ventana como era de esperar.

Una vez explicada brevemente esta pantalla intermedia, se procederá a comentar la programación de la que es la pantalla más compleja del proyecto. En esta ocasión se hará un recorrido por lo que el usuario tiene que hacer para traspasar un trabajo y ejecutarlo, además de comentar los botones adicionales que pueda haber.

Primeramente, cuando se construye la pantalla, simula un clic del botón actualizar, para mostrar los trabajos que hay dentro de la cámara. La programación de este botón se mostrará más adelante ya que es muy importante. El usuario ahora para realizar un traspaso de trabajo debe introducir la ruta del archivo manualmente o bien apretar el botón de los puntos suspensivos para buscarlo. Este botón hace que se despliege el buscador antes mencionado, gracias a una función GTK llamada “FileChooserDialog”. Se puede modificar tanto el título de la pantalla como la función de los botones de la parte inferior.

```
Gtk.FileChooserDialog filechooser = new Gtk.FileChooserDialog(
"Seleccione el trabajo", this, FileChooserAction.Open,"Cancelar",
ResponseType.Cancel, "Seleccionar", ResponseType.Accept);
```

Una vez pulsado el botón “Aceptar”, lo que se realiza es un análisis de si el trabajo que se ha seleccionado es correcto o no. Para ello se observa la extensión que tiene el archivo, indicando que si contiene “.avz”, es un trabajo correcto y por tanto se puede traspasar a la cámara, mientras que si no lo contiene salta el mensaje de error visto con anterioridad y se vuelve a mostrar el buscador. Este proceso se realiza con dos condicionales “if”, incluidos a su vez dentro de otro condicional, en el caso de que se haya pulsado aceptar. Por defecto, si se pulsa cancelar, el buscador se cierra automáticamente.

Cuando se ha seleccionado el trabajo, el siguiente paso es apretar el botón de “Transferir a la cámara”, el cual desplegará la pantalla intermedia que se acaba de comentar. Eso sí, no se podrá abrir a menos que no se esté ejecutando ningún trabajo, ya que puede causar problemas. Esto se sabe fácilmente gracias a las variables utilizadas de marcadores para saber si se encuentra Online o no. Una vez dentro de la pantalla se procede, se procede a poblar el desplegable con los números del 1 al 10. El Slot se seleccionará cuando se pulse sobre uno de los números que aparecen en el desplegable. Si está todo en orden, tanto trabajo como Slot, al pulsar aceptar se harán dos condicionales “if”, en el caso de que la cámara tenga contraseña o no para acceder a Telnet.

```
if (Window_8.Usuario=="" & Window_8.Contraseña=="")
else if (Window_8.Usuario != "" & Window_8.Contraseña != "")
```

El método en las dos resulta el mismo, a excepción de que en una se ha de introducir usuario y contraseña, mientras que en el otro no. Al ser el mismo proceso en esencia, se mostrará el que necesita de credenciales por ser el más completo. Primero se establece una conexión Telnet con el puerto 23, mediante el ya conocido comando "TelnetConnection". Se comprueba que se ha conectado con un condicional "if", y se pasa a ejecutar una serie de comandos, empezando por pasarle al prompt el usuario y la contraseña introducidos en la segunda pantalla del proyecto con las instrucciones "WriteLine".

```
tc150.WriteLine(prompt);
```

Es en ese momento cuando se le pasa la instrucción "CreateBINDrive" al prompt, tal y como se vio en los métodos de transferencia. Una vez pasada esa orden, lo que se hace a continuación es crear dos archivos ejecutables: un batch file y un ftp file. En el primero se escribe una línea que el sistema interpretará como que se va a iniciar una transferencia ftp, mientras que en el segundo se inicia la misma indicando la IP de la cámara, los credenciales necesarios, y todos los pasos a seguir mostrados en la transferencia ftp en su correspondiente apartado. Ya ejecutados los dos archivos, se procede a escribir el comando "FlashAvz" en la misma sesión Telnet que se abrió con anterioridad, de esta manera:

```
tc150.WriteLine("FlashAvz \"/bind0/" + Trabajo + "\"," + Slot);
```

Es en este momento cuando aparece la barra de transferencia que antes estaba en la ventana intermedia. Esta barra tiene 3 usos: Uno para la transferencia de los trabajos de la VS-06, otro para ejecutar los mismos y otro para ejecutar los de la CS-50. En este caso, como en muchos otros se utilizará varios condicionales, dependiendo de si es una fase u otra, aunque su estructura sea la misma siempre cambiando el tiempo que está activa. Para el caso que se está comentando ahora mismo, se indicara en el "main" que se active cuando se acabe de realizar el comando que se ha expuesto arriba. Para una mejor visualización de cómo funciona la función que se ha elaborado, se mostrará a continuación.

```
const int msDelay = 200;
            const int lowerBound = 0;
            const int upperBound = 100;

            progressbar1.Adjustment.Lower = lowerBound;
            progressbar1.Adjustment.Upper = upperBound;
            for (int pct = lowerBound; pct <= upperBound; pct++)
            {
                progressbar1.Adjustment.Value = pct;
                Main.IterationDo(false);
                progressbar1.Text = (pct + "%");
                Thread.Sleep(msDelay);
            }
```

Se ve por ejemplo la definición del intervalo en milésimas de segundo, 200 ms, y los límites inferior y superior, 0 y 100. Esto quiere decir que si se pretende conseguir un tiempo de espera de 20 segundos, cada tanto por ciento que se incremente la barra, deben de pasar los 200 ms antes mencionados (200x100=20.000 ms=20 s). Para ello se dispone un bucle "for" que incremente de 200 ms en 200 ms hasta llegar a 100. Finalmente cuando se acabe de rellenar la barra, esta desaparece automáticamente, lo que supone que el trabajo ya está traspasado.

A continuación el usuario pulsará actualizar para poder ver que efectivamente el trabajo ha sido traspasado de manera satisfactoria. Este botón borra primeramente los nombres de los trabajos que había dentro de las cajas de texto de cada slot, para posteriormente realizar una conexión Telnet en el puerto 49211. Se manda el comando "jobinfo" para recibir los nombres de los trabajos de la cámara, y almacenarlos dentro de un archivo de texto para mayor comodidad, como ya se ha hecho tantas veces. Una vez rellenado el txt, se procederá a leer línea por línea los trabajos y el número de slot de cada uno. Por ejemplo, si en una línea aparece slot=1, quiere decir que el el trabajo se encuentra en el slot 1. Así se va introduciendo nuevamente el nombre de los trabajos en la casilla correspondiente. Además se modifica el nombre de cada trabajo, suprimiendo "Slot=x" y cambiando la terminación ".avp" por ".avz".

```
if (line.Contains("slot1="))
                    {
                        this.Slot1.Text = "";
                        line = line.Replace("avp", "avz");
                        line = line.Replace("slot1=", "");
                        this.Slot1.Text = line;
                    }
```

Para finalizar con la carga del trabajo, se procede a pulsar el botón de carga adyacente al Slot que se desee ejecutar. Primero comprueba que efectivamente la lista de trabajos está actualizada, por lo que si no lo está no se podrá ejecutar ningún trabajo. Si está actualizada, cambia por defecto el color de cada casilla por un color blanco, para hacer entender que no se está ejecutando ningún trabajo. Se pasa entonces a conectar vía Telnet con el puerto 49211 y se ejecuta el siguiente comando para cargar el trabajo de ese slot.

```
tc.WriteLine("jobload -slot=5 -r");
```

Siguiendo el proceso, se vuelve a mostrar la barra de carga anteriormente comentada, solo que esta vez serán unos 10 segundos. Cuando la barra llega a su fin, se apreciará que el trabajo ha cambiado de color a verde, para mostrar que efectivamente se está ejecutando.

En el caso de que se quiera volver a parar la inspección, solo se tiene que volver a pulsar el botón "Online/Offline", el cual cambiará el color de todos lo recuadros de los slots a blanco, ejecutándose a su vez el comando offline en Telnet. En caso de volver a apretar el botón, según el marcador que se haya definido en a la hora de cargar el trabajo (del 1 al 10), se hará un "switch" para saber el recuadro del slot que debe de cambiar de color y ejecutar un "Online".

La opción de cerrar sesión, es muy importante, a la hora de hacer el “bootjob”. Teniendo en cuenta el mensaje que se mostró en la sección anterior, si resulta ser “sí” la contestación se ejecutará un comando Telnet de nuevo haciendo un “bootjob”. El trabajo que se deba hacer de arranque se verá con los mismos marcadores utilizados en el párrafo anterior, por lo que de esa manera el mensaje será personalizado para cada trabajo. Al igual que con el botón “Online/Offline", todo se realizará con un “switch”.

Para el botón Visualizar, se ejecutará un nuevo proceso con el que se visualizará el trabajo que esté ejecutando en el web server de la siguiente manera.

```
Process.Start("http://" + MainWindow.IP + "/app/hmi/index.html");
```

Finalmente, el botón ayuda, mostrará un detallado mensaje informando de cómo se debe traspasar un trabajo del PC a la cámara: "Para traspasar un trabajo a la cámara, seleccione en la parte superior la ruta de los archivos alojados en el PC, pulse "Transferir del PC a la cámara", seleccione el slot al que le quiera pasar el trabajo y pulse aceptar. Para ejecutar el trabajo de un slot, pulse el botón ejecutar a la derecha del slot deseado.".

Pantalla de edición de la CS-50: primer modelo y su versión mejorada

Como ocurre con la mayor parte del proyecto, esta es una parte nueva diseñada de 0, ajustada a las necesidades del mercado. Como se ha repetido en varias ocasiones, el sensor CS-50 es el destinado a dominar el mercado, en cuanto a sensores de visión de Di-soric se refiere. En un formato muy reducido, supera en casi todos los sentidos a su predecesora, la VS-06, por lo que era impensable no hacer una aplicación dedicada también a esta cámara.

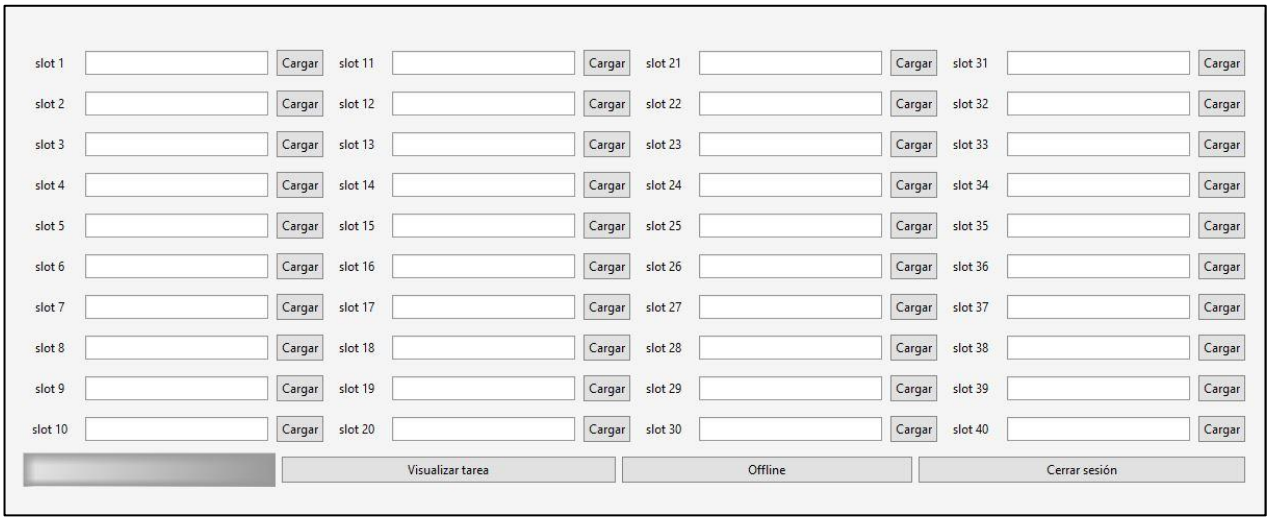


Foto 29: Prototipo 1 de la pantalla de gestión de la CS-50.

Habiéndose expuesto los problemas que da la cámara a la hora de recibir archivos del PC, resulta lógico adaptar estas necesidades a la pantalla que se pretenda crear. Sin archivos que pasar a la cámara, solamente sería necesario una lista de todos los trabajos incluidos dentro de ella y los botones necesarios para cargar los trabajos y visualizarlos a través del web server. En

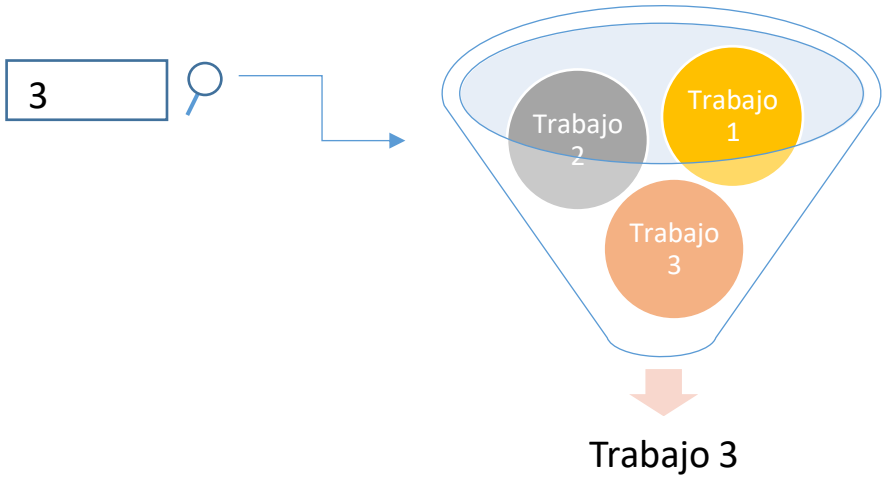
principio no es más que adaptar la pantalla de la VS-06 a la CS-50, suprimiendo el traspaso a la cámara. Es el motivo por el cual la primera pantalla diseñada tiene ese aspecto.

A primera vista parece una extensión de la VS-06, y en realidad es lo que es. Se ha multiplicado por 4 el número de Slot, hasta llegar a los 40. El principio de funcionamiento tampoco difiere: Antes de visualizarse la pantalla hace un chequeo de todos los trabajos que se encuentran dentro de la cámara a través de Telnet en el puerto 49211 y con la instrucción "jobinfo" se captura en un archivo de texto todos los trabajos guardados. Para cargar los trabajos, solo hace falta apretar el botón cargar al lado del trabajo correspondiente para ponerlo en ejecución. Se distingue el que está en ejecución de los demás por el hecho de que cambia el color de la caja, de blanco a verde. En el caso de querer visualizar la tarea a través de CloudLink, se habilita un botón de visualizar tarea que abre el explorador y para cerrar la sesión de edición el usuario solo tendrá que apretar el botón con esa misma función escrita.

Ninguna diferencia hasta ahora con los comandos utilizados en la VS-06, por lo que la programación de esta pantalla es prácticamente calcada a la de la antigua. Dependiendo del número de slot que tenga cada trabajo se almacenará en un cajetín u otro, cosa que se ve muy fácilmente gracias al archivo de texto en el que se ha guardado el resultado del trabajo "jobinfo".

Inconveniente del modelo 1

No obstante y el motivo por el cual este diseño se abandonó fue por la gran limitación que tenía. Puede que 40 trabajos sean suficientes para algunas industrias, ya que en el caso más favorable, la empresa cuenta con 3 o 4 trabajos almacenados. Sin embargo, hay empresas que les ocurre todo lo contrario: Necesitan tantos trabajos que puede que solo con 40 no sean suficientes, incluso con 100. Siendo esta la situación más desfavorable, no es aceptable que la pantalla solo tenga esos 40 slots quedando en terreno de nadie, es decir, demasiado para unos y muy poco para otros. Es por ello que se precisa una solución flexible para cada caso, que no muestre 35 slots vacíos, ni que no muestre todos los almacenados. Además, para este último caso sería necesario implementar un buscador de trabajos, para que el operario no tenga que perder demasiado tiempo en busca del trabajo que necesita. Es aquí donde entra en juego el explorador tipo Windows que se comentó con anterioridad: Una barra de búsqueda y un desplegable con los resultados de la misma, para una búsqueda mucho más fácil.



El resultado es un espacio mucho más despejado, adaptado en cada momento a las necesidades de cada industria, sin el vacío que se crea cuando no se trabaja con muchos proyectos, ni la saturación que implica un gran número de trabajos.

Modelo 2: la mejora

Si bien la anterior pantalla era un calco de la distribución que se había hecho con los trabajos de la VS-06, en esta poco o nada tiene que ver con aquella. Con la introducción del buscador ya no era necesario todo eso espacio que ocupaban los 40 trabajos. En cambio, se ha reducido considerablemente las medidas de la pantalla en detrimento de más modernidad y funcionalidad, como se puede apreciar en la imagen.

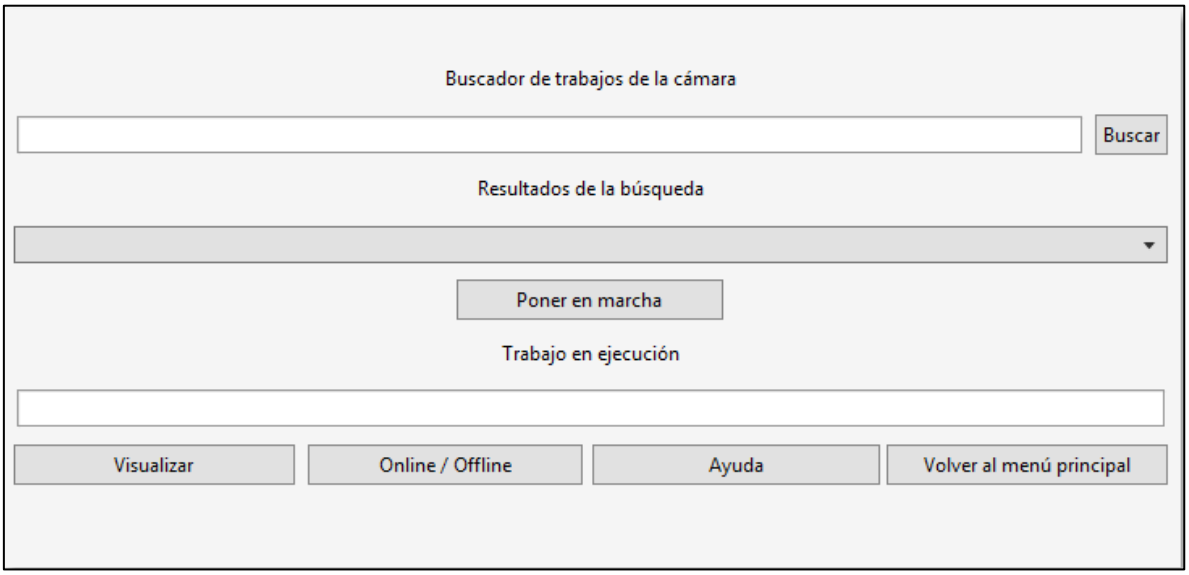


Foto 30: Pantalla definitiva para la gestión de la CS-50.

Pasando a comentar un poco el diseño, en primer lugar como indica la etiqueta de la parte superior, está el buscador de trabajos. Es un entrada editable en la cual el usuario deberá introducir un término de búsqueda, y la cámara hará un barrido por todos los trabajos que tiene dentro en busca de alguna coincidencia. Si no se encuentra ninguna coincidencia, se mostrará un mensaje anunciando la ausencia de coincidencias. En el caso de que exista alguna coincidencia, aparecerá dentro del desplegable de la parte de abajo los trabajos relacionados con la búsqueda. Si por algún casual se aprieta el botón buscar sin haber introducido ningún criterio de búsqueda, en el desplegable aparecerán todos los trabajos de la cámara, tal y como sucede nada más abrir la aplicación.

Dentro del desplegable pueden darse 3 situaciones. La primera es que no haya ningún trabajo, debido a que el criterio de búsqueda introducido no da ningún resultado, mostrándose el mensaje que se avanzaba en el anterior párrafo. La segunda es que se encuentren todos los trabajos de la cámara, o bien porque se acaba de iniciar la aplicación, o bien porque no se ha introducido ningún filtro cuando se ha apretado buscar. Por último, cuando se ha introducido un filtro válido, aparecerán todos los trabajos que lleven ese filtro en su nombre. La forma de mostrar los trabajos para mayor claridad es la siguiente: SLOT X→nombre del trabajo, siendo X el slot en el

que se encuentra. El usuario debe seleccionar uno para que se pueda poner en marcha en el siguiente paso.

Debajo de la barra de búsqueda de proyectos se encuentra el botón poner en marcha. Su propio nombre delata su función. Una vez seleccionado del desplegable el trabajo deseado, este es el botón indicado para ejecutarlo. No obstante, si el usuario no ha seleccionado ningún trabajo se mostrará otro mensaje como muestra de ello. Cuando se pone en marcha, el trabajo aparecerá resaltado en verde en el cuadro de texto con la etiqueta "Trabajo en ejecución", quedándose de color blanco cuando se va a offline la ejecución.

En cuanto a los botones que aparecen en la parte inferior ya se han tratado con anterioridad. El botón Visualizar tarea abre el explorador con el HMI online que proporciona Di-soric para poder ver los trabajos en vivo. El botón Offline y Online ponen en marcha o paran el trabajo que se encuentra en "Trabajo en ejecución". El botón ayuda despliega una serie de pasos que el usuario primerizo puede seguir para guiarse por la aplicación, mientras que el botón volver al menú principal devuelve al operario a la pantalla de inicio de sesión.

Pasando ahora a la programación, resulta bastante sencillo exponerla comparado con la gran cantidad de funciones que tenía la anterior pantalla. Para empezar, se realiza una conexión Telnet con el comando que ya se ha visto de la librería "Minimalistic Telnet", para luego mandarle un comando "jobinfo" y guardar la respuesta del comando en un txt, un método para nada desconocido. Para este caso se ha implementado una función que comprueba si hay trabajos dentro de la cámara, leer, y si no los hay mostrar que no los hay con un mensaje de "System.Windows. Forms". En el caso de que se encuentren trabajos, modifica los nombres de la manera que se ha indicado anteriormente (Ejemplo: SLOT 1→Tapas_San_Miguel).

El siguiente paso sería programar el botón buscar. Hay dos posibilidades definidas por condicionales "if" y "else". El primer criterio corresponde con la ausencia de un criterio de búsqueda, por lo que primero borrará el contenido del desplegable, para luego repoblarlo.

```
if (campo == "")
{
  for (s = i; s >= 0; s--)
  combobox1.RemoveText(s);
  leer();
}
```

Para el "else", correspondiente con la segunda condicional, se repoblará el desplegable con los trabajos que contengan el criterio de búsqueda. Primeramente se eliminarán los trabajos que estén dentro del desplegable, como se ha indicado en la programación de arriba, para luego leer el archivo txt en el que se ha almacenado los nombres de los trabajos. Todos aquellos trabajos que tengan en su nombre la palabra de filtro se introducirán dentro del combo box. Si no coincide con ningún trabajo entonces muestra el mensaje "No hay trabajos coincidentes".

```
if (line.Contains(campo))
                    {
```

```
            line = line.Replace(".avp", "");
            combobox1.AppendText(line);
            c = 1;
            i++;
        }
```

Una vez terminado de poblar el desplegable, es el turno del botón de poner en marcha. Tal y como sucedía en la VS-06, se hará mediante el comando "jobload", aunque en este caso habrá que realizar otros pasos previos. Para empezar hay que obtener el número del slot en el que se encuentra el trabajo. Para ello se hará una modificación en el nombre del trabajo que había en el desplegable hasta dejarlo en el estado inicial, a excepción de la palabra "slot", la cual será eliminada. La idea es hacer una separación de cadena a través del símbolo "=" que hay entre el nombre del trabajo y el número del slot, ahora que no está la palabra slot, con el fin de incluir el número en el comando.

```
Slot = Slot.Replace("  -->  ","=");
Slot = Slot.Replace("SLOT ", "slot");
Slot = Slot.Replace("slot", "");
string[] palabras = Slot.Split(caracter);
foreach (string x in palabras)
      {
          slt = x;
          break;
      }
tc.WriteLine("jobload -slot=" + slt + " -r");
```

La consecuencia de la separación de la cadena es que también se obtiene el nombre del trabajo que se está ejecutando, por lo que puede ser mostrado dentro del recuadro "Trabajo en ejecución", matando dos pájaros de un tiro.

Finalmente, comentada ya toda la programación de peso, lo que resta tiene mucha similitud con lo que se ha visto de la VS-06. Al ser unos botones iguales, la programación será prácticamente un calco a la vista en el diseño anterior. Empezando con el primer botón que se encuentra a la izquierda, el botón de Visualizar, simplemente se inicia el proceso de búsqueda, abriéndose el explorador predeterminado con la dirección del HMI de la cámara.

```
Process.Start("http://" + MainWindow.IP + "/app/hmi/index.html");
```

El botón "Online/Offline" no es para nada desconocido. Efectuará un Offline cuando el trabajo indicado esté en marcha, es decir cuando esté el recuadro de color verde, y hará un Online cuando esté parado. Eso sí, saltará un mensaje de advertencia cuando no haya ningún trabajo cargado para avisar al operario. Cuando el "marcador" inicio vale 1, quiere decir que se acaba de abrir la pantalla y que obviamente no puede haber ningún trabajo pendiente de ejecutar. Si este marcador vale 0 entonces es porque ya se ha cargado un trabajo. Para tener una cierta referencia del modo en el que está, se habilita otro marcador, "cargado", el cual valdrá 1 cuando esté online

y por lo tanto se ejecutará un offline apretando el botón, o 0 cuando esté offline y se ejecutará un online al apretar el botón.

Para el botón ayuda se mostrará un mensaje tal que así con el "System.Windows.Forms":

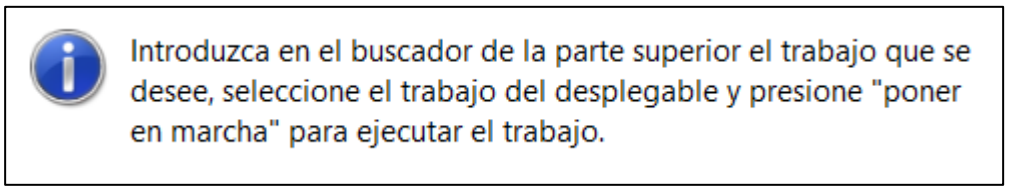


Foto 31: Mensaje desplegado tras pulsar el botón ayuda.

Para acabar, el botón cerrar sesión que, como sucedía con el botón de la VS-06, da la posibilidad al usuario de hacer el trabajo en ejecución trabajo de arranque (boot job). El comando es exactamente el mismo, incluido dentro de un mensaje de "System.Windows.Forms". Existen 3 condicionales, uno para una respuesta afirmativa, otro para una negativa y otra para cancelar. La diferencia entre la negativa y cancelar es que con la primera se vuelve al menú principal y con la otra solamente se cierra el mensaje. A pesar de ser la misma programación para ambas pantallas, a continuación se muestra la llevada a cabo en este caso:

```
if (result == DialogResult.Yes)
                {
                    tc.WriteLine("jobboot -slot=" + slt);
                    this.Hide();
                    MainWindow winx = new MainWindow();
                    winx.Show();
                    tc.WriteLine("exit");
                }
else if (result == DialogResult.No)
                {
                    this.Hide();
                    MainWindow winx = new MainWindow();
                    winx.Show();
                    tc.WriteLine("exit");
                }
```

Pantalla de Demostración: Visualización del trabajo

Por último y pese a ser la pantalla con la menor programación, sin duda es la que más tiempo va a estar en ejecución, ya que es muy útil a la hora de visualizar solo el trabajo que esté en marcha. Los dos botones de los que dispone hacen que la programación sea bastante sencilla en comparación con las que se ha expuesto con anterioridad. Básicamente es el control Online/Offline que se introdujo en las pantallas de gestión de trabajos de la CS-50 y la VS-06, solo que en este caso están mostrados de forma separada. La imagen que se muestra de fondo es la correspondiente al HMI de Di-soric que se encuentra en el explorador, el cual también se mostraba apretando el botón visualizar de las dos pantallas de edición. En definitiva, esta pantalla

no implementa ninguna función nueva, pero su papel es clave a la hora de visualizar sin peligro el trabajo que se esté ejecutando.

Foto 32: Botones de control en el modo demo.

He aquí la programación de los dos botones que, como se ha comentado anteriormente, es bastante simple.

```
protected void OnOnlineClicked(object sender, EventArgs e)
{
TelnetConnection tc1 = new TelnetConnection(MainWindow.IP, 49211);
tc1.WriteLine("esc\n");
tc1.WriteLine("online");
}

protected void OnOfflineClicked(object sender, EventArgs e)
{
TelnetConnection tc1 = new TelnetConnection(MainWindow.IP, 49211);
tc1.WriteLine("esc\n");
tc1.WriteLine("offline");
}
```

Simplemente realiza dos conexiones Telnet en el puerto 49211, con un “Online” para el caso de arriba y un “Offline” para el caso de abajo.

2.5 Conclusión

Dada la necesidad que había en un principio de poder gestionar los trabajos de la VS-06 de Di-soric, tanto traspasarlos del PC a la cámara como de la cámara al PC, se podría decir que solo se ha cumplido el 50%. No obstante, no sería justo decir que ambas partes, traspaso del PC a la cámara y viceversa, tengan la misma importancia ni el mismo grado de utilidad. Si bien es muy necesario poder traspasar los trabajos del ordenador al sistema de visión, el proceso inverso es muy poco frecuente e inusual, por lo que no resulta descabellado poder hacerlo a través de AutoVision.

Si a este hecho se le suma la posibilidad de poder incluir en el proyecto la CS-50, en un entorno en el que pueden convivir ambas cámaras, se puede decir que las aspiraciones que tenía el proyecto se han superadas con creces. No solo se abastece la necesidad que había desde el principio con la VS-06, sino que se extiende además al sensor de visión más vendido de Di-soric, la CS-50. Si se añade que corre en cualquier sistema operativo y en cualquier PC, el resultado es un software flexible, multiherramienta y muy intuitivo. A pesar de ser un proyecto que parte de cero y sin haber ninguna referencia en el mercado, se puede decir que el proyecto es todo un éxito, superando incluso las expectativas del mismo.

Vías de mejora en un futuro

La novedad que implica este producto, produce cierta incerteza en cuanto a saber exactamente cuál va a ser la reacción del público, al no haber nada parecido en la marca Di-soric. En este sentido la prueba de mercado es crucial para poder saber en qué aspectos mejorar, qué se ha de implementar que puede resultar útil y qué se puede eliminar. Bien es cierto que a simple vista no es necesario ningún cambio drástico, aunque a nivel de programación sí que puede optimizarse más de cara a una futura revisión. Estos pueden ser uno de los aspectos a mejorar:

- Automatizar en la pantalla de la VS-06 la actualización de los trabajos que se muestran, sin necesidad de apretar el botón actualizar.
- Personalizar el Cloud Link de cada cámara a gusto del cliente para una mayor presencia e identificación de la empresa.
- Eliminar el script que aparece cada vez que sale cada vez que se realiza un traspaso a la cámara, ya que hoy por hoy no lo hace en segundo plano.

PRESUPUESTO

Puesto que se trata de un proyecto que se realiza con varias cámaras, el coste total del desarrollo de la aplicación es bastante relativo, ya que depende de la cámara o cámaras que use el cliente, la aplicación es más o menos cara. En este caso se ha decidido hacer 3 presupuestos en base a los sensores más asequibles que tiene Di-soric en estos momentos, los cuales cuentan con una resolución VGA. Todo sea dicho, que se presupuesten estas cámaras no quiere decir que la aplicación solo funcione con ellas, al contrario, funciona con todos los modelos de la VS-06 y la CS-50 sin distinción. A continuación una exposición de los presupuestos para cada cámara.

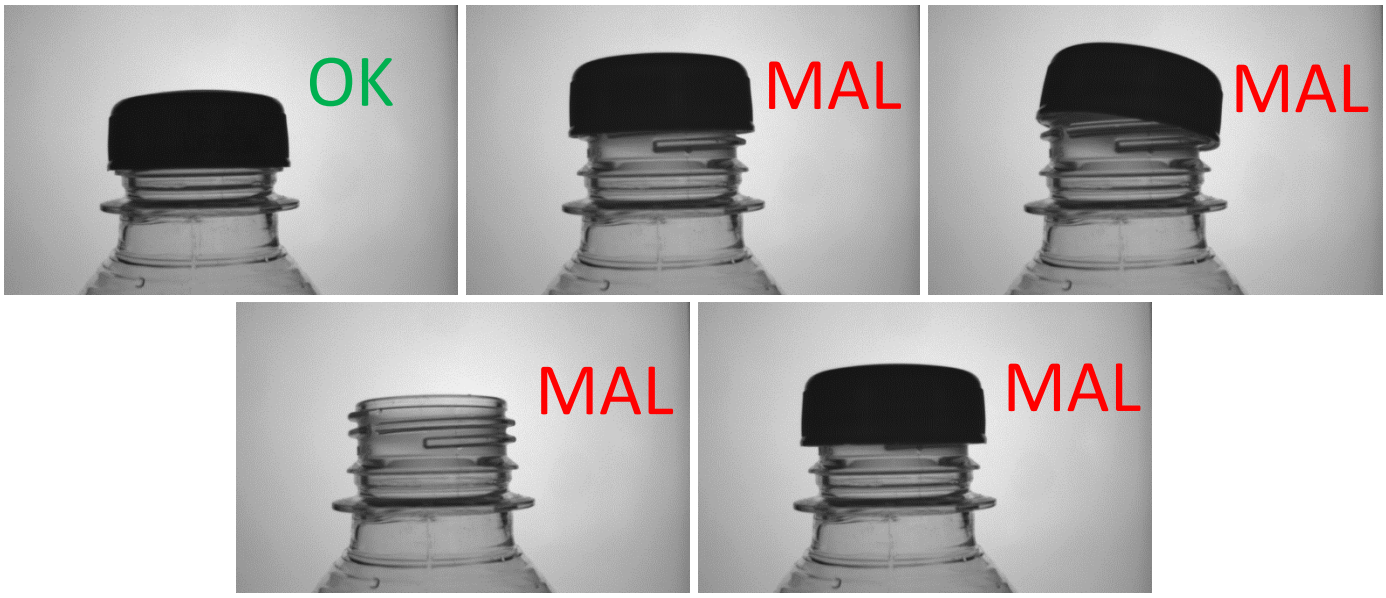
Artículo	Precio
CS-50	
Sensor de visión CS-50-BM2-2-ES-G5	1.239,09
Fuente de alimentación VSID-PS-24V-ES	181,00
Cable Ethernet VKHM-Z-1/RJ45	116,36
PC Industrial táctil 12"	875,00
Software AutoVision	——
Software Xamarin Studio	——
Programación e instalación	500,00
Total	**2.911,45**
VS-06 C-MOUNT VGA CON ÓPTICA	
Sistema de visión montura-C VS-06-BM2-00-ES	3.693,75
Óptica 6mm O-C1-S-06-14	105,41
Fuente de alimentación VSID-PS-24V-ES	181,00
Cable Ethernet VKHM-Z-1/RJ45	116,36
PC Industrial táctil 12"	875,00
Software AutoVision	——
Software Xamarin Studio	——
Programación e instalación	500,00
Total	**5.471,51**
VS-06 COMPACTA VGA	
Sistema de visión compacto VS-06-BM2-30-ES	3.919,62
Fuente de alimentación VSID-PS-24V-ES	181,00
Cable Ethernet VKHM-Z-1/RJ45	116,36
PC Industrial táctil 12"	875,00
Software AutoVision	——
Software Xamarin Studio	——
Programación e instalación	500,00
Total	**5.591,97**

ANEXO 1

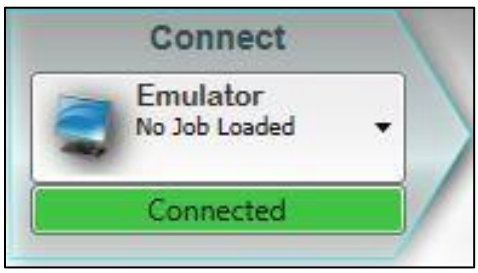
REALIZACIÓN DE UN PROYECTO TIPO CON EL SOFTWARE DE VISIÓN AUTOVISION

En este anexo se va a profundizar en el uso del software de edición y creación de trabajos para las cámaras de Di-soric, AutoVision, a través de un ejemplo práctico. No se puede entender completamente las cámaras sin antes saber cómo funciona este software, pilar de todos los proyectos de visión artificial. Antes de comenzar, decir que se ha realizado con el software AutoVision específico para la VS-06, en concreto la versión 3.0. Las imágenes se tomaron previamente con la cámara antes de empezar el ensayo, por lo que el paso de adquisición de imágenes se suprime dado que es bastante obvio, pasándose a trabajar en simulación.

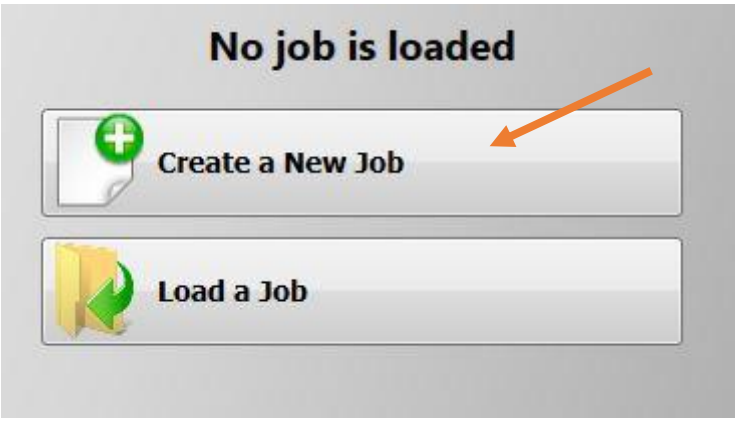
Se ha hecho un ensayo para detectar la presencia y el correcto posicionamiento de un tapón roscado en una botella de gaseosa. Para ello se han tomado 5 imágenes, de las cuales una es correcta y el resto defectuosas.



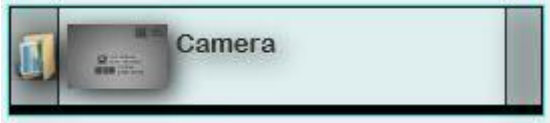
Para la prueba se ha utilizado un backlight para que resalte mucho más el tapón y se pueda ver mejor la posición y presencia del mismo. En primer lugar, nada más abrir Autovision, se seleccionará en el desplegable que aparece en la parte superior izquierda de la pantalla el simulador, como aparece en la imagen.



Posteriormente en el centro de la pantalla aparecerán dos botones, uno para crear un nuevo trabajo y otro para cargar uno ya hecho. Como obviamente no hay ningún trabajo creado, se procederá a crear uno nuevo.



La primera pantalla que se despliega es la correspondiente a la pestaña imágenes, pero como ya se han adquirido previamente, se pasará a la pestaña edición donde se podrán seleccionar las mismas. Una vez dentro de la pantalla edición, se puede apreciar justo debajo de donde se encuentra la pestaña de conectar “Connect”, un recuadro como el siguiente:

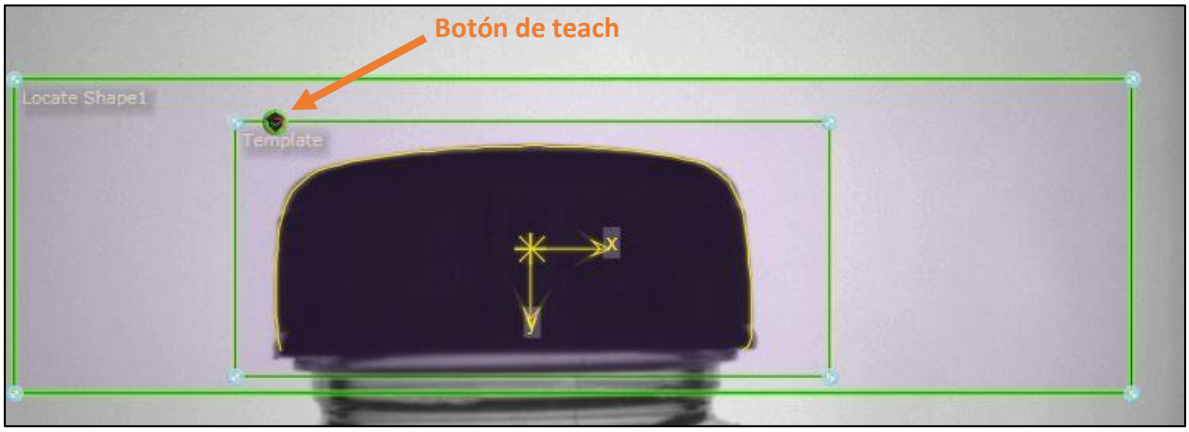


Clicando en el icono de la carpeta que aparece en la parte izquierda se podrá seleccionar la carpeta donde se encuentran las imágenes, y así poder pasar a editarlas en el modo edición. Una vez cargadas las fotos, en el área de edición aparecerá una de las imágenes. Se podrá pasar a la siguiente simplemente clicando el botón de la cámara que aparece dentro del área.

A continuación se van a exponer las herramientas que se pretenden usar para la aplicación:

De izquierda a derecha se encuentran la herramienta localizador, ideal para localizar formas y para referenciar otras herramientas a esa posición, en el caso de que la pieza se mueva. La siguiente es la herramienta de presencia, la cual en base a un número de píxeles que se encuentren en un área juzgará si se encuentra o no un elemento definido por el usuario. Por último la herramienta de medida, que simplemente efectúa una medida en píxeles y juzga si es una longitud que se encuentra en el rango de tolerancias o no.

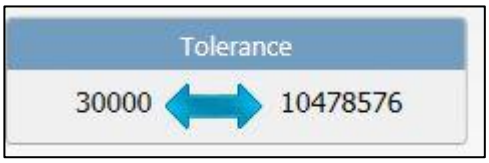
Vistas de forma breve, a continuación se elegirá la que se considera como imagen buena para empezar a crear el trabajo. Antes se ha hablado de la herramienta de localización como una herramienta a la que referenciar otras, por si la pieza no está siempre en el mismo sitio o rota. Es recomendable que se elija una parte de la imagen irrepetible para que siempre sea capaz de detectar la pieza de manera fiable. En este caso se seleccionará el tapón ya que es una parte única e irrepetible de la botella.



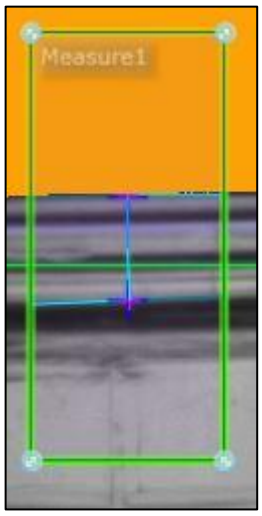
Esta herramienta consta de dos áreas, una dentro de la otra. La exterior es el llamado ROI, que delimita el área de búsqueda de la forma, la cual se obtiene haciendo un “teach” sobre el área del tapón. En este ejemplo se puede observar como capta perfectamente la curvatura del tapón, y como el ROI se extiende en ambos lados por si la botella se mueve lateralmente.

Habiendo aprendido la forma del tapón, el siguiente paso será detectar la presencia del mismo. Con la herramienta de presencia se seleccionará un área que incluya dentro el tapón para poder ver si está o no. Por defecto, todas las herramientas que vayan después de la herramienta de localización quedarán referenciadas a esta última, por lo que no hace falta modificar nada más. Una vez seleccionada el área donde se encuentra el tapón, se ajustará los límites de detección de píxeles en el histograma que aparece en la parte derecha de la pantalla. En este caso el límite

superior se ha establecido en 45, siendo 0 el negro absoluto y 255 el blanco más puro. Inmediatamente debajo se encuentra la tolerancia. Esta se ajustará dependiendo de la cantidad de píxeles que encuentre la herramienta, con lo que se ajustará el límite inferior en base a los píxeles que encuentra la herramienta en cada imagen con presencia de tapón. De entre todas las imágenes el caso más desfavorable son 34.000 píxeles, por lo que se establecerá un límite inferior de 30.000, suficiente para detectar su presencia. Para terminar con la herramienta, se deberá pulsar el símbolo de giro que aparece en la parte superior izquierda del margen derecho, para habilitar la rotación del área de detección en caso de que el tapón no esté completamente recto.

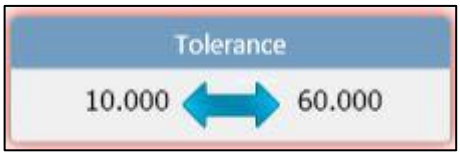


Ya para terminar con la edición del trabajo, se utilizará la herramienta de medida. Para ello se selccionará un área que vaya desde casi la parte superior del tapón hasta la rebaba de plástico del cuello de la botella. Esta última será la que se utilice como referencia, ya que la parte que se moverá será la parte inferior del tapón. Con todo se pretende llegar a un resultado como este:

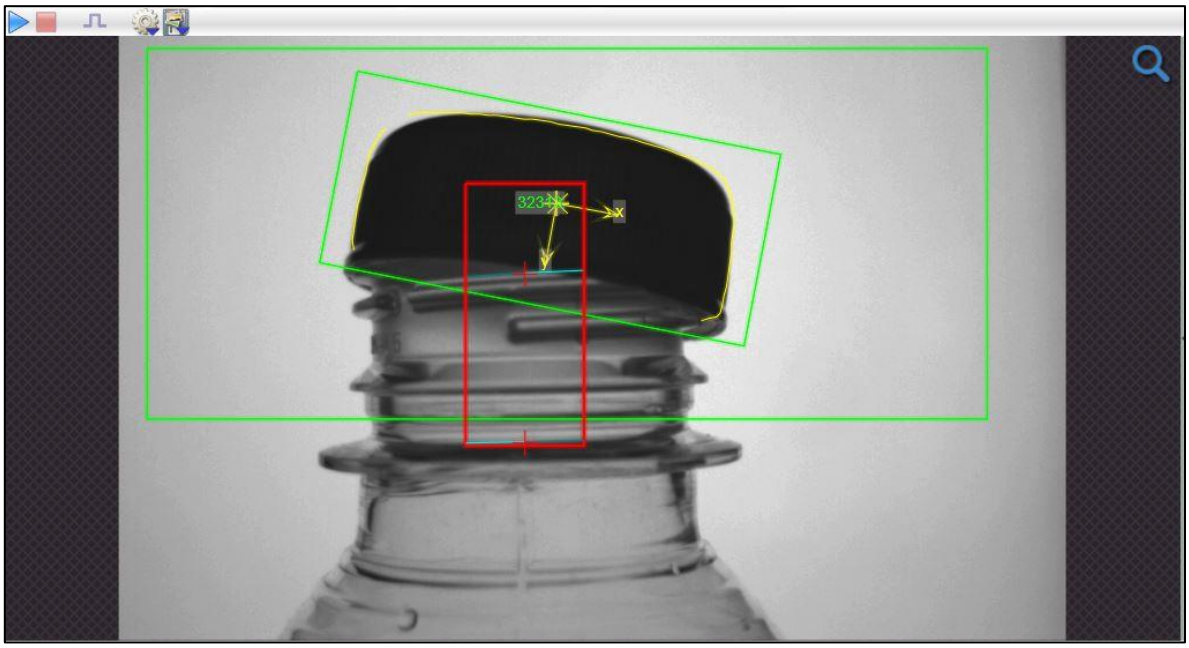


Los pasos que se han seguido para obtener la distancia que se muestra en la imagen son primero seleccionar la polaridad de los bordes en el margen derecho de la pantalla: Si son de blanco a oscuro o de oscuro a blanco. En este caso como lo que se quiere ver es un área blanca que se encuentra entre el tapón oscuro y la rebaba que también lo es, se elegirán los bordes de oscuro a blanco.

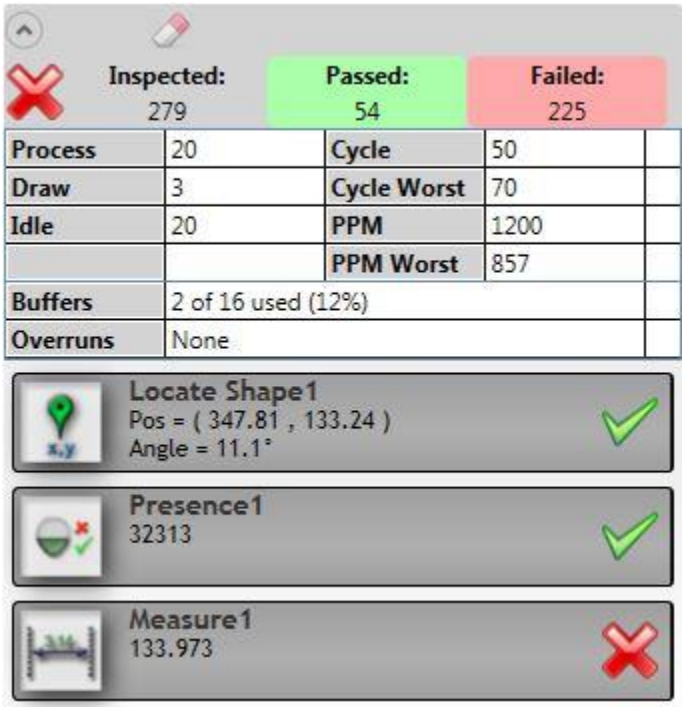
Con esta opción los parámetros que vienen por defecto deben ser suficientes para detectar los dos bordes. Por último la tolerancia, la cual para el caso correcto da una distancia de 55 píxeles, por lo que el límite superior se ajustará a unos 60 para descartar las botellas erróneas.



Finalizado el proceso de edición del trabajo y viendo que detecta correctamente los elementos de cada imagen, se procede ahora a poner en RUN el proyecto. Con ello se obtiene un carrusel de imágenes en el que la cámara (en este caso el simulador) hace una inspección foto por foto y enviando un veredicto. Si se cumplen las tres condiciones propuestas la salida será buena. Si falla una de las tres o más, dará mala la salida. He aquí un ejemplo del veredicto que hace de una imagen de las que se habían tomado:



Se observa que se ha detectado la forma del tapón, que hay presencia del mismo, pero la distancia entre tapón y rebaba no es la correcta, por lo tanto dará como mala la imagen de manera acertada. Como se ve a la derecha, hay un panel de estadísticas en el cual se observa el número de inspecciones, las correctas, las erróneas, las inspecciones por minuto… Lo cual es muy útil a la hora de analizar que un proceso se ejecuta de manera correcta. Más abajo se encuentran los diferentes resultados de las inspecciones para cada herramienta. Se muestran cuales han fallado y cuales han dado un resultado positivo. Así en este caso aparece que las herramientas de presencia y localización son correctas, pero la medida al ser superior a 60, 133 como se puede apreciar en la imagen que se encuentra a continuación, la inspección resulta como mala. En este caso y en esta imagen captada, 1 de cada 5 inspecciones resulta buena, concordando con las características de las imágenes que se han seleccionado en un primer momento, siendo así una inspección ideal.

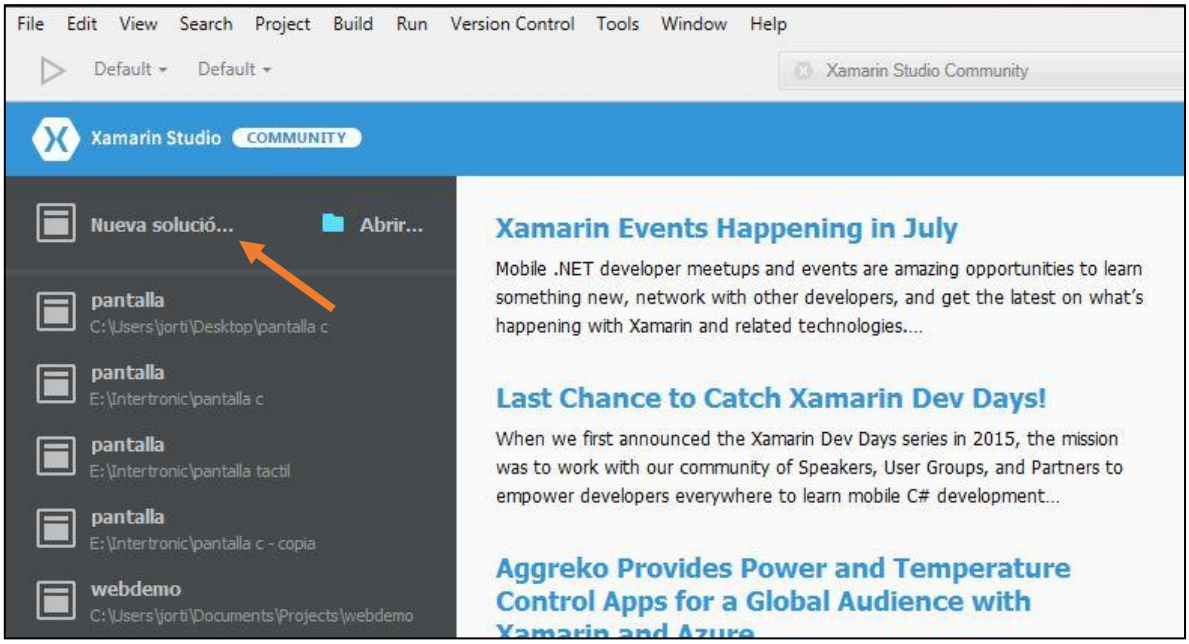

Inspected:
279
Passed:
54
Failed:
225
Process
20
Cycle
50
Draw
3
Cycle Worst
70
Idle
20
PPM
1200
PPM Worst
857
Buffers
2 of 16 used (12%)
Overruns
None
Locate Shape1
Pos = (347.81 , 133.24)
Angle = 11.1°
Presence1
32313
Measure1
133.973

ANEXO 2

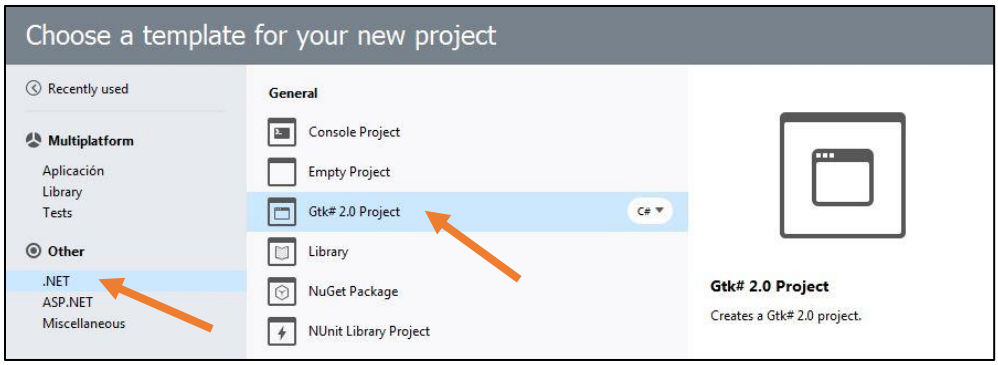
REALIZACIÓN DE UN PROYECTO DE VENTANA BÁSICO CON XAMARIN STUDIO

A la hora de mostrar cómo se realiza un proyecto en Xamarin Studio de una ventana gráfica, se procederá a realizar una en el mismo programa, comentando a su vez los diferentes detalles que el usuario se puede encontrar en el transcurso de su creación. En este caso se va a crear un simple botón con el que se desplegará el mensaje “Hola Mundo”.

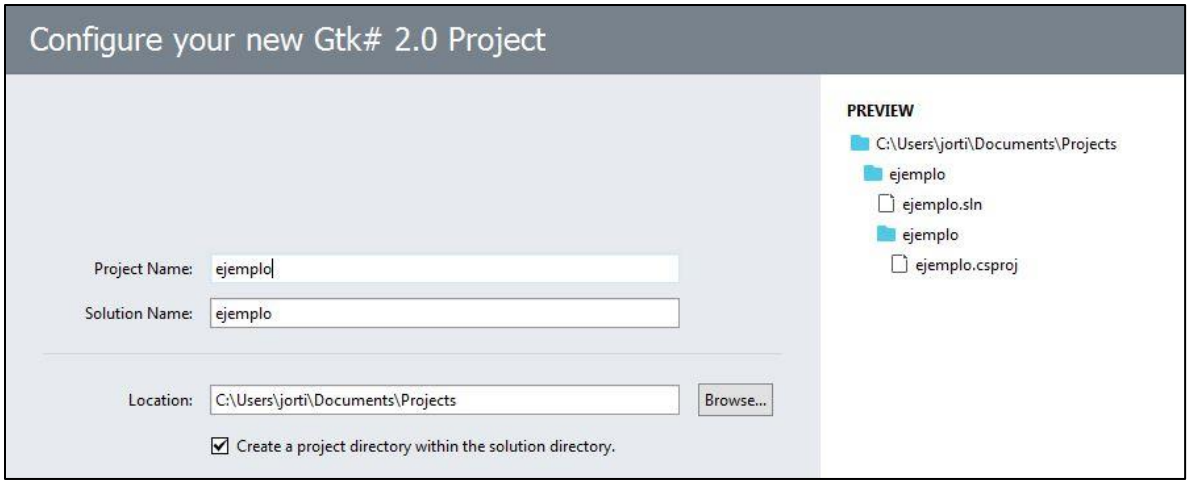
Nada más abrir el programa aparecerá la pantalla de inicio, en la cual habrá que dirigirse al apartado de nueva solución que se encuentra en la parte superior izquierda.



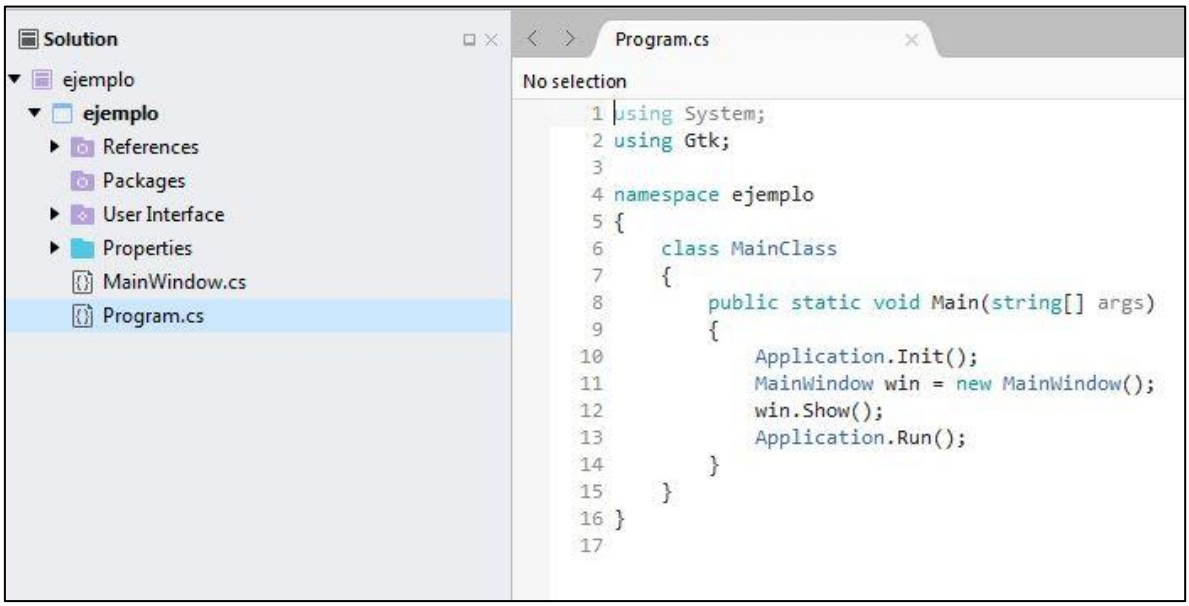
Una vez clicado el botón aparecerá un menú de selección del tipo de proyecto que se desea crear. Como se ha dicho anteriormente, en Xamarin se pueden crear multitud de clases de trabajos, desde el básico proyecto de consola, pasando por ventanas gráficas, librerías para otros proyectos, aplicaciones web, o incluso proyectos vacíos. Obviamente en este caso se trabajará con la aplicación de ventana gráfica, para lo cual se en el apartado .NET en la lista de la izquierda se seleccionará la opción GTK# 2.0 Project (Foto 2) y se clicará “Next”.



A continuación el programa pedirá al usuario el nombre del nuevo proyecto a crear, el sitio donde guardarlo y un esquemático donde aparece un resumen del proyecto a crear. En este caso se ha nombrado al archivo "Ejemplo", guardándose en la carpeta en la que se archivan los proyectos por defecto. Para finalizar la configuración del proyecto se clicará en "Create".



Llegados a este punto, el usuario se encontrará con un proyecto con dos pestañas con cierta programación en ellas.

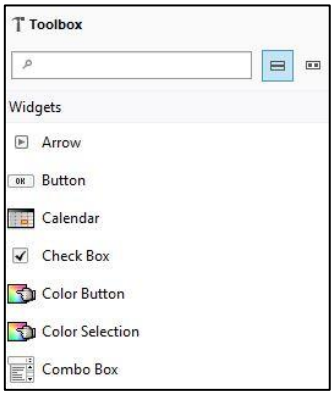


Por defecto el nombre de cada una de estas pestañas es "Program" y "Main Window". El form "Program" es por así decirlo un código maestro, donde la aplicación se inicializa y llama a los diferentes forms o ventanas gráficas en este caso. Resulta imprescindible a la hora de ejecutar el programa y por lo tanto está presente en todos los proyectos que se creen en Xamarin. Por otra

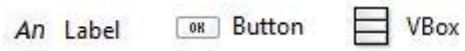
parte “Main Window” es el nombre que se le da a la ventana gráfica que el programa crea por defecto. Es en este form donde el usuario podrá crear, editar y modificar la ventana gráfica. Si no se modifica su contenido, cuando se ejecute el proyecto solo se verá una pantalla vacía, la cual solo se podrá cerrar finalizando al mismo tiempo su ejecución. Pero sin adelantar acontecimientos, a continuación se procederá a crear la ventana gráfica.

Cabe decir que en la ventana “Main Window” el usuario dispone de dos modos de edición seleccionables en el margen inferior izquierdo en la pantalla: “Source” para editar el código y “Designer” en el cual se puede editar la ventana. Primeramente, al ser la parte más sencilla e intuitiva, se empezará por el diseño gráfico.

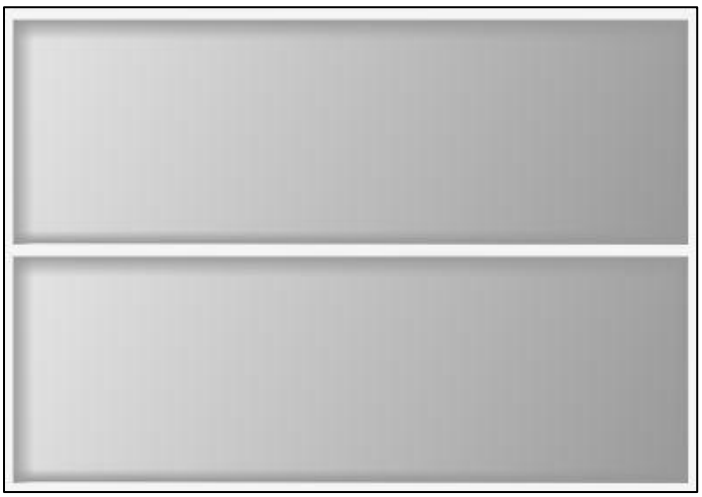
Una vez dentro se puede apreciar un recuadro en gris sin contenido alguno en el centro de la pantalla. Este será el espacio de trabajo donde se incluirá todos los elementos de la ventana gráfica que se desee. En la parte derecha se visualiza una barra vertical en la cual hay diversas pestañas, de las cuales 2 son relevantes para este caso: “Toolbox” y “Properties”. Como su propio nombre indica, la pestaña “Toolbox” es donde se encuentran las herramientas con las cuales se puede trabajar, desde botones a desplegables, pasando por entradas de texto, buscadores, títulos…



En el caso que se pretende abordar, solo van a ser necesarias dos herramientas: Un botón y un título o label. No obstante se va a necesitar antes dividir la pantalla en dos secciones horizontales, una para cada herramienta. Esto se consigue con la herramienta VBox, que se encuentra casi al final del desplegable.

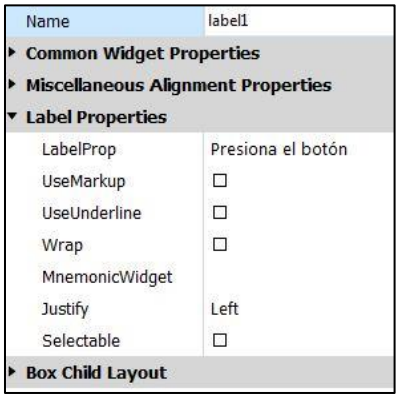


Para dividir la pantalla en 2 se tendrá que arrastrar esta última herramienta a la ventana de color gris. Una vez soltada la herramienta, se puede observar que se ha dividido la pantalla en 3 secciones horizontales. Como para este ejemplo se necesitan solo 2, se procederá a borrar uno de los tres recuadros que hay ahora pulsando el botón derecho del ratón y luego en “delete”.

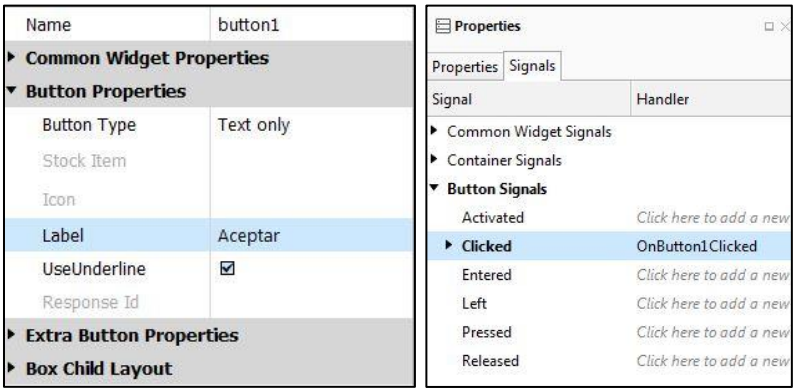


Una vez dispuestas las dos secciones horizontales, obviamente se introducirán las herramientas de botón y título. Para ello se seguirá el mismo proceso que con la herramienta VBox, arrastrándolos hasta la sección que se desee. En este caso se pondrá el título en la parte superior mientras que el botón irá en la parte de abajo.

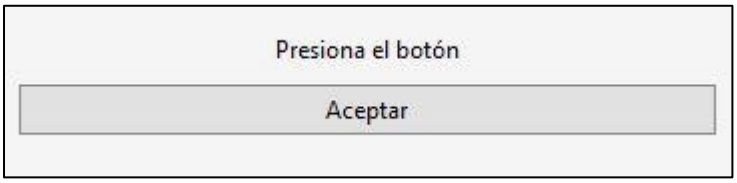
Una vez terminado, se procede ahora con la personalización de las dos herramientas. En el caso del Título, se seleccionará primeramente para después acceder a la pestaña de “Properties” en el margen de la derecha. Dentro de “Properties”, las opciones a modificar están dentro de “Label properties”, en concreto “Labelprop” para modificar el nombre. En este caso se introducirá el nombre “Presiona el botón”.



Para el botón, se tienen que tener en cuenta otros parámetros, como la respuesta del botón a un clic. Antes se modificará de la misma manera que en el título el nombre, dentro del apartado "label" en "Button properties". Se cambiará por el título "Aceptar" dentro del botón.



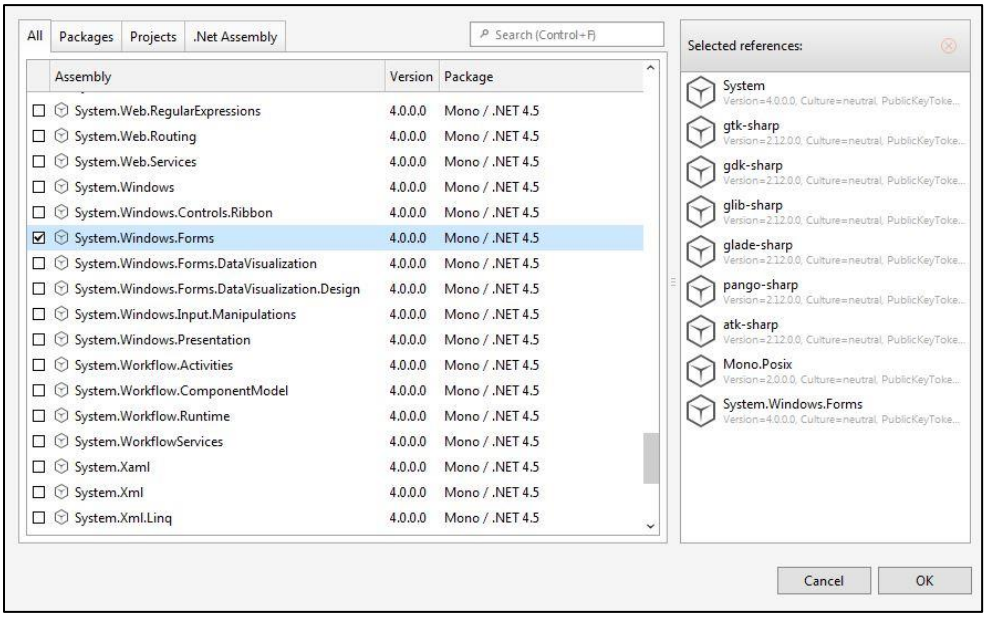
En cuanto a la respuesta del botón cuando se clica, se irá esta vez dentro de la pestaña "Properties" a "Signals" en la parte superior. Dentro de los diversos apartados se desplegará el apartado "Button signals", en el cual se seleccionará la opción "Clicked". Cuando se seleccione aparecerá inmediatamente a su derecha "OnButton1Clicked", lo que supone que ya está implementado. De esta manera se habría finalizado el diseño gráfico, consiguiendo un modelo tal que así:



El siguiente paso es la edición del código fuente. Para ello habrá que dirigirse a la pestaña "Source" en la parte inferior izquierda. Se puede ver como ahora hay una función llamada "OnButton1Clicked" al final del form. Es aquí donde habrá que indicar que es lo que debe suceder cuando se aprieta el botón, en este caso desplegar el mensaje "Hola Mundo".

```
protected void OnButton1Clicked(object sender, EventArgs e)
{
}
```

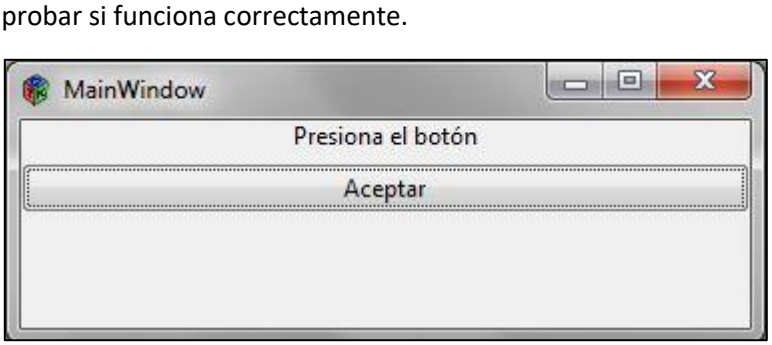
Primero hay que introducir una nueva librería llamada "System.Windows.Forms", la cual es muy utilizada en entornos de Windows. Para incluirla habrá que escribir al principio del Form using "System.Windows.Forms;", y clicar con el botón derecho del ratón en referencias en el esquema de la solución que se encuentra en el margen izquierdo. Se selecciona la opción "edit references" y una vez dentro se marca la librería "System.Windows.Forms" y se clica en "OK".



Finalizado este proceso, a continuación se procede a cumplimentar la función "OnButton1Clicked" con el siguiente código:

```
MessageBox.Show("Hola Mundo", "Aviso", MessageBoxButtons.OK,
                MessageBoxIcon.Asterisk);
```

Con este mensaje se pretende que además de mostrarse le mensaje, además lleve como título en la ventana "Aviso", un botón de "OK" en la parte de abajo y un dibujo de una exclamación. Así pues, con este último paso, el Proyecto estaría acabado por lo que solo haría falta probar si funciona correctamente.



En efecto se ha conseguido que se muestre el mensaje de "Hola mundo cuando se pulsa el botón "Aceptar".

ANEXO 3

COMANDOS TELNET Y RS-232 DE LAS CÁMARAS (PÁGINAS 161 - 167 DEL MANUAL DE LA VS-06)

Esta sección proporciona descripciones de los comandos serie que se pueden enviar a la cámara a través del puerto TCP (Telnet), Terminal AutoVision o HyperTerminal.

SET {tagname}{value}

Sets value of a global tag.

The tagname must correspond to one of the supported tags within the device.

The value can contain spaces.

The command is terminated by a carriage return and/or line feed character.

The value can be a list of comma-separated items to set a sequence of tags:

Send **SET int1 1, 2, 3** to set int1 = 1, int2 = 2, int3 = 3.

The AVP service allows setting of step and datum information from the job tree using forward slash '/' in the symbolic name path. **SET avp/insp1/snapshot1/acq1/gain 2.0** paths are not case-sensitive and do not need to be fully qualified if unique.

SET avp/acq1/gain 2.0 will set the same gain value if there is only one acquire.

Control tags in the AVP service such as **START**, **STOP**, and **TRIGGER** act as momentary switches. **SET avp.start 1** is equivalent to the **ONLINE** command. **avp.start** will reset immediately and always read as **0**.

Success Return: On success will return **!OK** followed by an echo of the command. For example:

!OK SET matchstring1 ABCD

Fail Return: On failure will return **!ERROR** followed by the reason for the failure. For example:

!ERROR Tag matchstring66 not found

GET {tagname}

Gets value of a global tag.

The tagname must correspond to one of the supported tags within the device.

The command is terminated by a carriage return and/or line feed character.

Include an index to get a single value from an array such as **GET int1**. If the index is omitted, the full array of values will be returned in a commaseparated list of values.

Send **Get {tagname}** to get the value of a tag within the global data service. To get the value of a tag within another service, prefix the tagname with the service name. For example, a **GET {service.tagname}** command such as **GET eip.input** for the EIP input assembly.

The AVP service allows retrieval of step and datum information from the job tree using forward slash '/' in the symbolic name path. **GET avp/insp1/snapshot1/status** paths are not case-sensitive and do not need to be fully qualified if unique.

GET avp/snapshot1/status will return the same result if there is only one inspection.

When issued against a step, **GET avp/snapshot1** will return the values for all datums.

Success Return: On success will return the value stored in the tag. For example:

ABCD

Fail Return: On failure will return **!ERROR** followed by the reason for the failure. For example:

!ERROR Tag matchstring66 not found

INFO {service.tagname or service}

Gets information about a tag or service.

INFO with no arguments gets a list of services.

INFO {service} gets a list of tags in that service.

INFO {service.tagname} gets attributes of the tag as well as a list of subtags.

The AVP service allows retrieval of step and datum information from the job tree using forward slash '/' in the symbolic name path.**INFOavp/insp1/snapshot1/status** paths are not case-sensitive and do not need to be fully qualified if unique.

INFO avp/snapshot1/status will return the same result if there is only one inspection.

When issued against a step, INFO avp/snapshot1 returns properties of the step, a list of child datums, and a list of child steps. Child steps are indicated by a trailing forward slash.

GETIMAGE {-transfer=ymodem} {-type=failed}{-format=[jpg|png]}

{-quality=n} {-inspection=n} {woi=l,t,r,b}

Initiates serial transfer of inspection image.

-**transfer=ymodem** is not currently optional - only Ymodem protocol is supported.

-**type=failed** to retrieve the last failed image. If omitted, the current image is returned.

-**format=[jpg|png]** specifies the format of the image. If omitted, the image format is JPG.

-**quality=n** specifies a JPG compression quality of n less than or equal to 100. The default quality is **80** if not specified.

-**inspection=n** specifies the inspection from which to retrieve an image. The image will be from the first snapshot within that inspection. If not specified, the image will be from the first inspection that does contain a snapshot.

woi=left,top,right,bottom specifies a rectangular area of the image to be included in the output image. If omitted, the full image buffer is returned.

ONLINE

Starts all inspections.

OFFLINE

Stops all inspections.

TRIGGER {inspection index}

Triggers inspection. If {inspection index} is omitted, inspection 1 is triggered.

VT (Virtual Trigger) Command

Triggers an inspection by pulsing a Virtual I/O point. For example:

VT 1

will return pulse **VIO1**. The inspection will run if it is configured to use **VIO 1** as a trigger.

Syntax: VT [VIO Index]

• If specified, the VIO index must be in the allowed range for Virtual I/O points within Visionscape. The virtual I/O line will be set high then low.

• If VIO Index is not specified, VIO1 is assumed

Success Return: Nothing is echoed on success of the VT command.

Fail Return: Return **!ERROR** followed by the reason for the failure. For example:

!ERROR No such trigger

JOBSAVE [-slot=]n

Save job to slot n.

JOBLOAD [-slot=n][-r]

Load job from slot n.

-r = Start inspections.

JOBDELETE [-slot=n]

Delete job in slot n.

JOBINFO [[-slot=]n][-v]

Get job summary or info about slot n.

-v = Verbose. This option shows the amount of space that would be freed if the job were deleted. It also lists the total disk space and free disk space.

JOBBOOT {-slot=n}

Set bootup job slot n.

JOBDOWNLOAD [-transfer=]{YMODEM}

Download .avz job packaged via transfer method.

JOBDELETE -all

Delete all jobs in job slots.

Important: Does not delete the current job loaded in camera memory.

GET SYSTEM.JOBSLOT

Retrieve the slot of the current job. Note that the current job in the camera can be loaded from a job slot or the PC. If it isn't loaded from a job slot then this command will return **-1**.

ANEXO 4

DATASHEETS DE LA CS-50 Y LAS VS-06

Vision Sensor CS 50
Vision Sensor CS 50

- **Höchste Performance mit bis zu 2520 Prüfungen / Minute**
- **Bildexport über FTP**
- **Beleuchtung, Filter und Polarisator wechselbar**
- **Gleichzeitig mehrfache Merkmalserkennung und -messung möglich**
- **Profinet / Ethernet- / TCP-IP / RS 232**
- **Mehrsprachige, intuitive Software**
- **Teileerkennung, Anwesenheit, Messung, Positionieren, Zählen und Logik**

- **Highest performance with up to 2520 inspections per minute**
- **Image export via FTP**
- **Internal LEDs, filter, polarizer field exchangeable**
- **Multiple detections and measurements**
- **Profinet / Ethernet- / TCP-IP / RS 232**
- **Multi-language and intuitive software**
- **Shape- and presence detection, positioning, counting & logic**

Technische Daten (typ.)	Technical data (typ.)	**bei** / at **+20°C, 24 V DC**
Anzahl Ein-/Ausgänge	Number of in / output	1/3
Betriebsspannung	Service voltage	4,75 ... 30 V DC
Eigenstromaufnahme	Internal power consumption	150 mA, (24 V DC)
Bedientasten	Control buttons	1
Schaltausgang	Switching output	max. 100 mA pro Ausgang, opto-isoliert (Strombegrenzung kundenseitig) /
		per output, opto-isolated (current limited by user)
Externer Trigger-Eingang	External trigger input	off = < 1,0 V (Eingang EIN / Input ON)
Inspektionsaufgaben	Inspection functions	Finden, Zählen, Anwesenheit, Messen, Logik
		Presence, Counting, Finding, Measurement, Logic
Ausstattung	Configuration	Interner Trigger / Internal trigger
		Externer Triggereingang / External trigger input
		PC-Softwareeinrichtung / PC Software Setup
		Logik für benutzerdefinierte Ausgänge / Logic for user-defined outputs
		Höchste Auflösung / Highest resolution (640 x 480 Pixel Global shutter)
Beleuchtung wechselbar	Lighting changeable	ja, (rot / red, weiss / white, blau / blue, IR / infrared)
Umgebungstemperatur	Ambient temperature	0 ... +40 °C
Schutzart	Protection class	IP 65, IP 67
Schutzklasse	Protection degree	III, Betrieb an Schutzkleinspannung / Operation on protective low voltage
Gehäusematerial	Casing material	Aluminium schwarz / black
Gewicht	Weight	68 g
Bildausgabe	Image	sw / VGA
Anzahl Jobs	Number of jobs	Unbegrenzt / unlimited (1,4 GB)
		1-255 bei Betrieb mit ProfiNet / in operation with ProfiNet
Belichtungszeit	Shutter time	66 ... 58.825 µsec
Vibrations- / Schockfestigkeit	Vibration / Shock resistance	55 Hz-Sinus / 2.000 Hz-Random / 50 g

VS-06-BM2-30-ES
Smart camera

- Integrated with di-soric VS AutoVision software
- Industry proven machine vision tools
- Fully integrated with processor, lens and illumination
- Liquid lens autofocus
- Integrated Ethernet network
- Scaleable system - extended toolset available

TECHNICAL INFORMATION (typ.)	+20°C, 24V DC
Size	40.5 x 57.6 x 96.3 mm (Dimensions)
Emitted light	Red light laser, 655 nm, (for positioning)
Laser class	1 (IEC 60825-1)
Service voltage	5 ... 28 V DC
Internal power consumption	max. 170 mA, (24 V DC)
Resolution	752 x 480 pixel
Trigger input	4,5 ... 28 V DC, 13 mA (24 V DC)
Outputs 1, 2 and 3	1 ...28 V DC, <100 mA at 24 V DC (current limited by user)
Interface	Ethernet, RS 232
Laser power	564 µW
Optics	30° optics
Ambient temperature	0 ... +45 °C
Protection class	IP 65
Connection	Ethernet M12, 8-poled, I/O-Cable M12, 12-poled
Sensor size	1/3"
Protocols	Point-to-Point, Point-to-Point w/XON/XOFF, TCP / IP, Ethernet / IP, Profinet
Weight	280 g
Image sensor	CMOS (Global shutter)
Focusing	Liquid-lens Autofocus
Frame rate	60 images/second (max.)

VS-06-BM2-00-ES
Cámara inteligente

- Software VS AutoVision de di-soric integrado
- Herramientas de visión probadas industrialmente
- Versión de montaje C
- Conectividad Ethernet integrada
- Sistema escalable - software avanzado disponible

CE

TECHNICAL INFORMATION (typ.)	+20°C, 24V DC
Tamaño	102,3 x 57,6 x 40,5 mm (dimensiones de la carcasa)
Tensión de alimentación	5 ... 28 V DC
Consumo propio	105 mA, (24 V DC)
Resolución	752 x 480 pixel
Entrada de trigger	4,5 ... 28 V DC, 13 mA (24 V DC)
Salidas 1, 2 y 3	1 ...28 V DC, <100 mA a 24 V DC (limitación de corriente por el usuario)
Interface	Ethernet, RS 232
Temperatura ambiente	0 ... +50 °C
Técnica de protección	IP 65, IP 67, (operando con VSID-LP-C-xx)
Conexión	Ethernet M12, 8-polos, cable I/O M12, 12-polos
Tamaño del sensor	1/3"
Protocolos	Punto a punto, Punto a punto w/XON/XOFF, TCP / IP, Ethernet / IP, Profinet
Peso	320 g
Sensor de imagen	CMOS (Global shutter)
Focusing	C-Mount, manual
Capturas por segundo	60 imágenes/segundo (máx.)

ANEXO 5

Librería C# Minimalistic Telnet

```
using System;
using System.Net.Sockets;
using System.Text;
using System.Threading;

namespace MinimalisticTelnet
{
    public class TelnetConnection : IDisposable
    {
        private TcpClient tcpSocket;
        private int TimeoutMs = 100;

        public bool IsConnected
        {
            get
            {
                return tcpSocket.Connected;
            }
        }

        public TelnetConnection(string hostname, int port)
        {
            tcpSocket = new TcpClient(hostname, port);
        }

        ~TelnetConnection()
        {
            Dispose(false);
        }

        public string Login(string username, string password,
        int loginTimeoutMs)
        {
            int oldTimeoutMs = TimeoutMs;
            TimeoutMs = loginTimeoutMs;

            string s = Read();
            if (!s.TrimEnd().EndsWith(":"))
            {
                throw new Exception("Failed to connect : no login prompt");
            }

            WriteLine(username);

            s += Read();
            if (!s.TrimEnd().EndsWith(":"))
            {
                throw new Exception("Failed to connect :
                no password prompt");
            }

            WriteLine(password);
```

```
        s += Read();

        TimeoutMs = oldTimeoutMs;

        return s;
    }

    public void WriteLine(string cmd)
    {
        Write(cmd + "\n");
    }

    public void Write(string cmd)
    {
        if (!tcpSocket.Connected)
        {
            return;
        }

        byte[] buf = ASCIIEncoding.ASCII.GetBytes(cmd.Replace("\0xFF",
        "\0xFF\0xFF"));
        tcpSocket.GetStream().Write(buf, 0, buf.Length);
    }

    public string Read()
    {
        if (!tcpSocket.Connected)
        {
            return null;
        }

        var sb = new StringBuilder();
        do
        {
            ParseTelnet(sb);
            Thread.Sleep(TimeoutMs);

        } while (tcpSocket.Available > 0);

        return sb.ToString();
    }

    private void ParseTelnet(StringBuilder sb)
    {
        while (tcpSocket.Available > 0)
        {
            int input = tcpSocket.GetStream().ReadByte();
            switch (input)
            {
                case -1:
                    break;
```

```
                    case (int)Verbs.Iac:
                        // interpret as command
                        int inputVerb = tcpSocket.GetStream().ReadByte();
                        if (inputVerb == -1)
                        {
                            break;
                        }

                        switch (inputVerb)
                        {
                            case (int)Verbs.Iac:
                                sb.Append(inputVerb);
                                break;

                            case (int)Verbs.Do:
                            case (int)Verbs.Dont:
                            case (int)Verbs.Will:
                            case (int)Verbs.Wont:

                               int inputoption =
                               tcpSocket.GetStream().ReadByte();
                                if (inputoption == -1)
                                {
                                    break;
                                }

                           tcpSocket.GetStream().WriteByte((byte)Verbs.Iac);

                                if (inputoption == (int)Options.Sga)
                                {
                                tcpSocket.GetStream().WriteByte(inputVerb ==
 (int)Verbs.Do ? (byte)Verbs.Will : (byte)Verbs.Do);
                                }
                                else
                                {
                                tcpSocket.GetStream().WriteByte(inputVerb ==
 (int)Verbs.Do ? (byte)Verbs.Wont : (byte)Verbs.Dont);
                                }

                                tcpSocket.GetStream().WriteByte((byte)inputo
ption);
                                break;
                        }

                        break;

                    default:
                        sb.Append((char)input);
                        break;
                }
            }
```

```
        }

        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }

        protected virtual void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (tcpSocket != null)
                {
                    ((IDisposable)tcpSocket).Dispose();
                    tcpSocket = null;
                }
            }
        }


        #region Private Enums

        enum Verbs
        {
            Will = 251,
            Wont = 252,
            Do = 253,
            Dont = 254,
            Iac = 255
        }

        enum Options
        {
            Sga = 3
        }

        #endregion
    }
}
```