Universidad Politécnica de Cartagena

etsit escuela técnica superior de ingeniería de telecomunicación

Desarrollo de un software para la gestión y procesamiento de capturas de fotos en un sistema de fotografía digital.

Alumno: Francisco Álvarez Rodríguez.

Tutor: Francisco Miguel Monzó Sánchez.

ÍNDICE

1. **INTRODUCCION** Página 1

1.1. Descripción general del proyecto Página 2

1.2. Descripción específica de mí parte del proyecto Página 3

2. **DESCRIPCION GENERAL DEL SISTEMA** Página 4

2.1. Esquema del sistema Páginas 4-6

2.2. Cámara canon Eos Páginas 7-8

2.3. Terminal Página 9

2.3.1. Ordenador Página 9

2.3.2. Placa de Control Página 10

2.3.3. Validador de monedas Páginas 10-11

2.3.4. Monitor táctil Páginas 12-13

2.3.5. Impresora Páginas 14-15

2.4. Monitor exterior Páginas 16-17

2.5. Software Página 18

3. **APLICACIÓN SENSOR. Descripción y funcionamiento** Páginas 19-22

4. **APLICACIÓN SIMULADOR. Descripción y funcionamiento** Páginas 23-26

5. **APLICACIÓN SHOWRUN. Descripción y funcionamiento** Páginas 27-30

6. **APLICACIÓN MONTADOR. Descripción y funcionamiento** Páginas 31-36

7. **COMUNICACIÓN ENTRE APLICACIONES** Páginas 37-38

8. **DIFICULTADES DURANTE EL DESARROLLO Y COMO SE SALVARON** Páginas 39-40

9. **SOFTWARE ¿MÁS O MENOS INTELIGENTE?** Página 41

10. **APLICACIÓN PRÁCTICA** Páginas 42-48

11. **CONCLUSIÓN** Página 49

1. INTRODUCCIÓN

El propósito de este proyecto es crear una aplicación capaz de automatizar la gestión de unas fotos que son generadas por una cámara fotográfica, posicionarlas en sus directorios correspondientes y ponerlas al servicio de otra aplicación ya implementada.

Nuestra aplicación está adaptada a un software ya existente, que se encarga de, estando en sus directorios correspondientes, mostrar las fotografías en una pantalla táctil de un terminal, además de la gestión de impresión y álbum.

¿Cuál sería la funcionalidad de este sistema?

El sistema en conjunto está pensado para funcionar en una atracción de feria, donde una cámara fotográfica recoge las instantáneas de los usuarios durante su viaje, y las muestra en tiempo real en un monitor, para después, a la finalización del mismo, ponerlas a su servicio para la impresión, previo pago.

El objetivo de este proyecto, además de su desarrollo software a partir de unas especificaciones, es el de desarrollarlo de la forma más eficiente. También me aportará la experiencia de ser partícipe en todas las fases de un proyecto software, excepto en la toma de requisitos. Tener una visión global, y de cómo encajan las piezas en el desarrollo de un proyecto software, o de cualquier proyecto, es fundamental para el buen funcionamiento del mismo.

1.1. DESCRIPCIÓN GENERAL DEL PROYECTO

El sistema lo podemos dividir en dos partes:

- La parte hardware
- La parte software.

La parte hardware está formada por un terminal, un monitor, una cámara fotográfica y un sensor (catadióptrico). El terminal a su vez lo forma: una pantalla táctil, una placa de control, un validador de monedas, un PC, y una impresora.

El software es un conjunto de aplicaciones encargadas de la automatización y la gestión de las fotografías, que describiremos con más detalle en los sucesivos capítulos.

Este sistema está pensado para prestar un servicio al usuario de una atracción de feria, por ejemplo, y a la vez, proporcionarle el servicio fotográfico mientras está disfrutando de la atracción.

1.2. DESCRIPCIÓN ESPECÍFICA DE MI PARTE DEL PROYECTO

El ingrediente a este completo sistema, que es de mi aporte, es de la parte del software.

Mi trabajo consistió en dar una solución para la gestión de unas fotografías que generaba una cámara fotográfica, y ponerlas a disposición de otra aplicación que forma parte del sistema.

Para ello, tenía que gestionar cada foto de manera individual en función de en qué momento se tomara y teniendo en cuenta su orden.

Para la realización del software, he tenido que crear otras aplicaciones auxiliares.

Estas aplicaciones simulan elementos hardware del sistema, y me permiten sobre todo, realizar pruebas y una evaluación global del funcionamiento del sistema.

La aplicación 'montador' es la aplicación principal de mi programación.

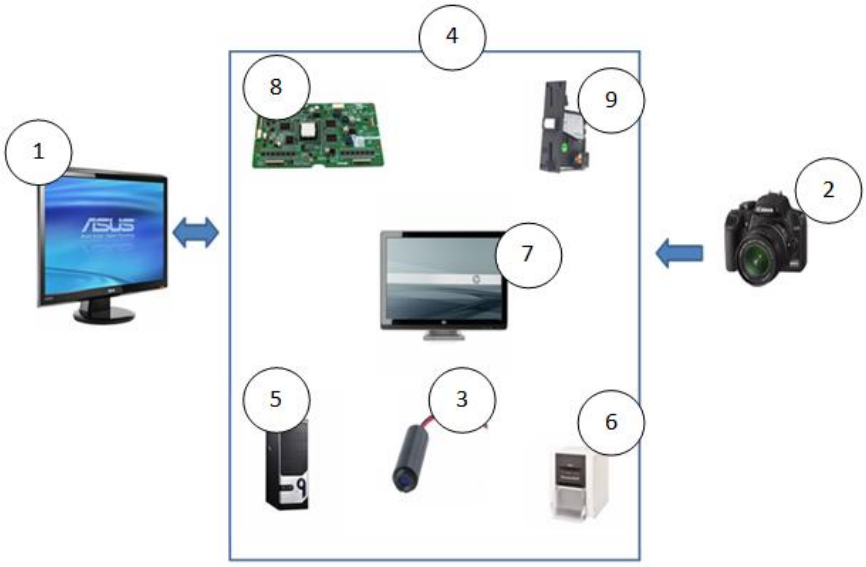
Las aplicaciones auxiliares, aunque no por ello menos importantes, reciben el nombre de: 'sensor', 'simulador' y 'showrun', que en capítulos siguientes serán detallas tanto en su funcionamiento individual, como en el funcionamiento global y sincronización.

2. DESCRIPCIÓN GENERAL DEL SISTEMA

A continuación vamos a describir de manera detallada todo el sistema, tanto la parte software como el hardware, y también añadiremos un esquema para una mejor comprensión.

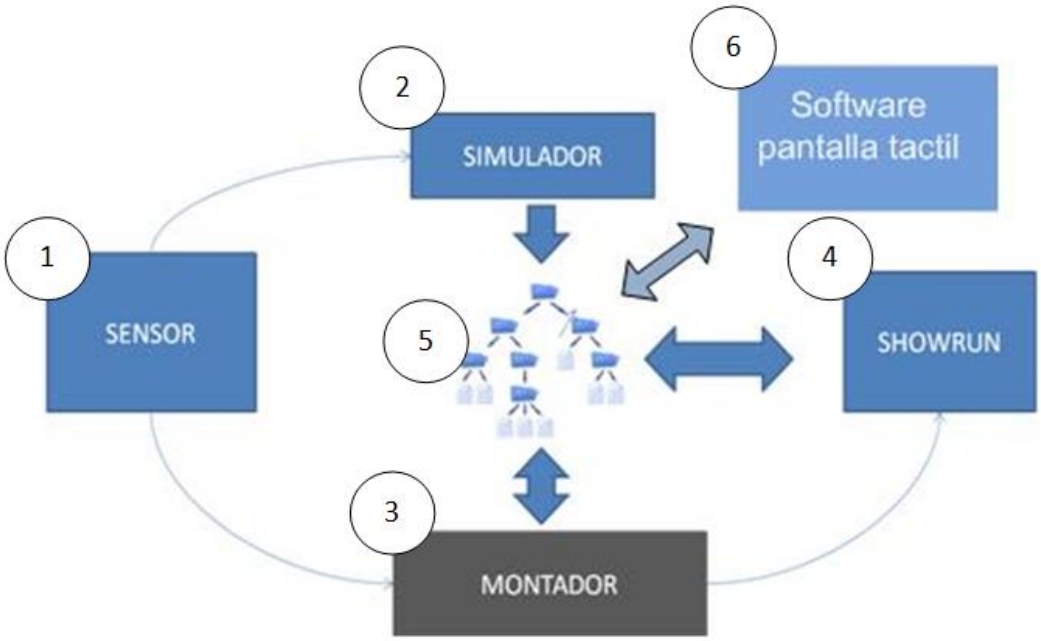
2.1. ESQUEMA DEL SISTEMA

- Esquema hardware:



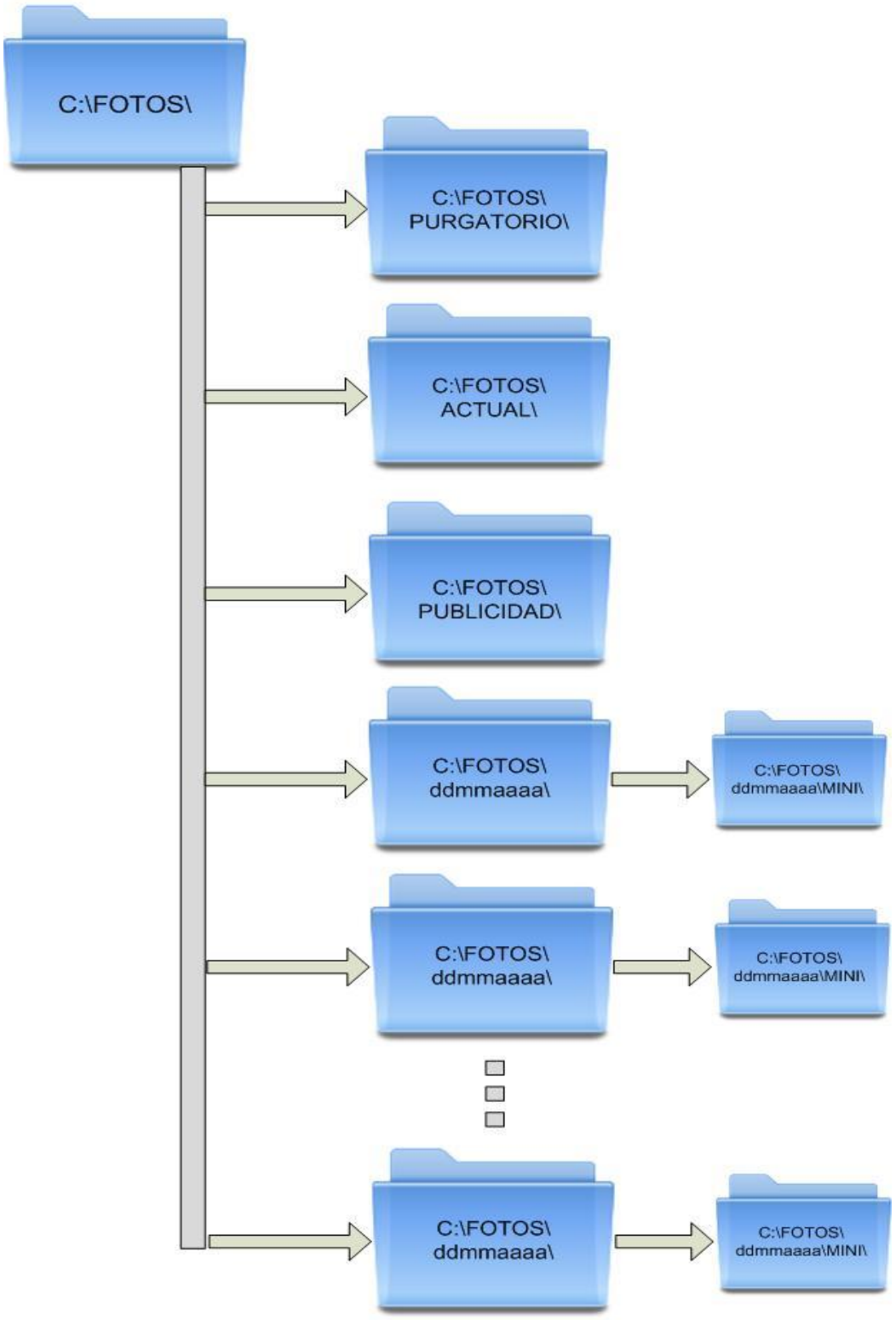
1. Monitor externo.
2. Cámara fotográfica.
3. Sensor.
4. Terminal.
5. Ordenador.
6. Impresora.
7. Pantalla táctil.
8. Placa de control.
9. Validador de monedas.

- Esquema software:



1. Aplicación sensor.
2. Aplicación simulador.
3. Aplicación montador.
4. Aplicación showrun.
5. Árbol de directorios.
6. Software previo que gestiona la fotografías del árbol de directorios.

- Jerarquía de directorios:



2.2. CÁMARA CANON EOS

Las características más notables de la cámara son:

- Sensor CMOS de 10,1 Megapíxeles
- Hasta 3 fps
- AF de área amplia en 7 puntos
- Sistema Integrado de Limpieza EOS “EOS Integrated Cleaning System”
- Pantalla LCD de 2,5” (6,35 cm), con Modo de Visión en Directo “Live View Mode”
- Procesador DIGIC III
- Ranura para tarjetas SD/SDHC
- Procesamiento Estilo de Imagen “Picture Style”
- Compacta y ligera
- Compatible con los objetivos EF/EF-S y los flashes Speedlite EX

De todas ellas tenemos que destacar las que la hace la más adecuada para nuestro propósito:

Sensor CMOS de 10,1 Megapíxeles

Un sensor CMOS de 10,1 Megapíxeles permite captar suficiente detalle para imprimir las fotos hasta tamaño A4, incluso recortando la imagen. La tecnología de los sensores CMOS de Canon garantiza imágenes nítidas y claras, incluso cuando se fotografía en situaciones con poca luz.

Hasta 3 fotogramas por segundo

Para poder conseguir fotos espectaculares de acción, la EOS 1000D es capaz de captar imágenes JPEG grandes en serie, a una velocidad de hasta 3 fotogramas por segundo y sin límite en cuanto al número, hasta que se llene la tarjeta de memoria. Con una tarjeta de 2 GB se podrían llegar a captar hasta 514 imágenes sin interrupción.

AF de área amplia en 7 puntos

El Sistema Autofoco de área amplia utiliza 7 puntos de enfoque separados, para poder enfocar con precisión los sujetos que se desplazan a gran velocidad. El punto de enfoque central permite incluso enfocar en situaciones con poca luz o sobre sujetos con contraste bajo.

Sistema Integrado de Limpieza EOS "EOS Integrated Cleaning System"

El sistema integrado de limpieza del polvo de Canon previene de tres formas distintas que las imágenes resulten afectadas por el polvo: reduciendo el polvo que se genera dentro de la cámara, desprendiendo el polvo que pudiera haberse depositado en el sensor, cada vez que se conecta y desconecta la cámara y, por último, localizando las motas de polvo que aún pudieran permanecer en el sensor para su posterior eliminación mediante el software Digital Photo Professional incluido de serie.

Todas estas caracterizas hacen que la cámara canon Eos 1000D sea la candidata adecuada para formar parte del sistema integral.

2.3. TERMINAL

El terminal es la parte más 'voluminosa' del sistema, la podemos ver como una 'caja negra' con los siguientes subsistemas comunicados entre sí.

2.3.1. ORDENADOR

La función de nuestro ordenador es contener y ejecutar el software de la aplicación.

Hoy en día, la evolución y prestaciones de un ordenador personal cada vez son más avanzadas y a precios más competitivos, es por ello, que un ordenador personal de gama media, y sin entrar en el terrenos de las 'marcas', sería válido para la ejecución de nuestro propósito.

Cuando 'hablamos' de un ordenador personal, no lo identificamos con un ordenador de sobremesa, hoy en día un ordenador portátil, es más que nunca un ordenador personal, además este tipo de computadores tienen cada vez precios más asequibles, y la flexibilidad que te proporciona su portabilidad.

2.3.2. PLACA DE CONTROL

La placa de control es la coordinadora de todo el sistema, encargada de comunicar todos los elementos hardware entre sí.

2.3.3. VALIDADOR DE MONEDAS

El validador de monedas es la parte del sistema que se encarga de recoger las monedas que introduce el usuario y validar su autenticidad.
Y en primera instancia, es el encargado de dar la orden de impresión.

El proceso de medición y reconocimiento de monedas es uno de los más sofisticados del sistema como se explica a continuación:

Las características de las monedas introducidas se registran mediante tres sensores de medición dispuestos uno tras otro. Los parámetros de medición son, entre otros, diámetro, espesor, aleación y para ciertas monedas también las características de acuñamiento.

Los valores de medición obtenidos, se comparan con las bandas de aceptación almacenadas en el interior de los 12 canales. En el caso de una coincidencia total con un canal, el verificador de monedas valoriza la moneda introducida según la información de salida programada, entonces, este canal se 'bloquea' mediante el interruptor DIL correspondiente o la línea de entrada respectiva.

Tras la medición, una barrera de luz comprueba en el área de las chapaletas de aceptación, si la moneda introducida cae libremente al conducto de la caja. Este 'control de caja' se amplía en equipos con clasificación mediante un 'control de clasificación'. Sólo luego de pasar este sensor de prueba adicional es entregada la señal de moneda.

Con esto se supervisa el área de aceptación completa, incluyendo la dirección de caída, y se contrapone a eventuales manipulaciones.

2.3.4. MONITOR TÁCTIL

El monitor táctil es la interfaz que sirve de comunicación entre el usuario y el sistemas.

Con él, el usuario puede ordenar al sistema que traiga una u otra foto a la pantalla, mostrar varias a la vez o agrandar otra.

El monitor táctil tiene que posicionarse de forma ergonómica en el Terminal, para que la visualización de fotografías sea lo más cómoda para el usuario.

Entre la variedad de monitores tactiles del mercado, una buena opción es el 'Monitor LCD panorámico táctil de 23 pulgadas HP 2310ti', cuyas características mostarmos a continuación:

Requisitos de energía y operación	
Requisitos de alimentación	De 90 a 265 V CA, de 45 a 63 Hz
Consumo energético	56 vatios máximo, 47 vatios típica
Consumo de energía (en espera)	2 W
Margen de temperaturas operativas	de 5 °C a 35 °C
Intervalo de humedad en funcionamiento	De 20 a 80% HR

Características del sistema		
Valoración de HP para este producto	Ajustable	
	Cumplimiento	
	Rendimiento	
	enfoque de coste total de propiedad	
Tamaño de pantalla (diagonal)	58,4 cm (23")	
Resolución de la pantalla	1920 x 1080	
Brillo (típico)	275 cd/m²	
Relación de contraste	1000:1	
Ángulo de visión de la pantalla	160° en horizontal; 160° en vertical	
Señal de entrada	1 VGA; 1 DVI-D	
Multimedia	Altavoces activos integrados (2 x 2 W por canal) con HP Power Sound y control de volumen frontal	
Color del producto	Negro	
Ángulo de movimiento de la pantalla	Inclinación: de - 5 a + 55 grados	
Seguridad física	acepta cerradura Kensington	

2.3.5. IMPRESORA

La impresora es también una de las partes más importantes de nuestro sistema, ya que es la que imprime la fotografía que el usuario se llevará consigo. Dos de las características principales que tiene es la calidad de impresión (impresión fotográfica de alta definición), y un sistema de suministro de tinta que permite imprimir un número de fotografías relativamente alto, sin tener que cambiar el cartucho con mucha frecuencia.

Una de las propuestas del mercado que mejor se adapta a nuestras necesidades es la impresora Mitsubishi 9550DW.

Las características técnicas de esta impresora son:

- **Dual Line Technology: Aumento de la Calidad y Nitidez de Imagen**.

Los cabezales "dual line" han sido desarrollados para mejorar la resolución de la impresión. En comparación con el resto de impresoras de sublimación del mercado, las nuevas impresiones de la CP9550DW y la CP9550DW-S se realizan mediante 2 cabezales por cada punto.

El resultado es una alta calidad y una mayor textura de la imagen.
Las características de las nuevas impresoras permiten una dramática reducción de los problemas de enfoque.

- **Alta Capacidad Producción-Reducción De Los Costes Variables**

Los nuevos consumibles para las impresoras CP9550DW y CP9550DW-S permiten aumentar considerablemente la autonomía de las mismas, debido a que permiten realizar hasta 600 copias sin necesidad de cambiar el papel o la tinta.

2.4. MONITOR EXTERIOR

El monitor es el 'relaciones publicas' y la imagen de nuestro sistema. Es lo primero en lo que se fija un posible cliente (usuario), por tanto es de especial importancia que sea un monitor relativamente grande y de alta definición, ya que cuando nuestro sistema este en funcionamiento será el escaparate de nuestro 'producto'.

Uno de los monitores que mejor encaja para nuestra propuesta es el 'Asus VH192C', un monitor de alta definición. El cual posee las siguientes características:

- **Calidad de imagen y ahorro energético**.

El gran consumo energético al que sometemos al planeta en estos días se está haciendo notar de un modo exponencial. Por esa razón, la eficiencia energética de la tecnología cada vez tiene un mayor peso a la hora de crear un producto. El monitor LCD VH192C adopta una tecnología que sólo emplea dos lámparas para iluminar la pantalla, ahorrando más de un 28% del consumo y una película óptica que asegura una representación de imágenes con todo el brillo que esperas.

- **La pantalla panorámica Color Shine Widescreen te ofrece la visión más panorámica**.

Adopta la Tecnología Color Shine que emplea una pantalla antireflejos que optimiza los contornos de las imágenes, la densidad del color y la saturación, mejorando la definición de la imagen y haciéndola perfecta para la reproducción de películas o juegos. El VH192Cofrece una imágen vibrante y ahorro energético.

- **ASCR(ASUS Smart Contrast Ratio) 10000:1**.

La tecnología ASCR (ASUS Smart Contrast Ratio), ajusta dinámicamente el contraste entre del panel retroiluminado en consonancia con el contenido de la imagen, y mejora el ratio de contraste a 10000:1, lo que resulta en una representación más realista de escenas nocturnas para juegos o películas.

- **Excelentes funciones y calidad de imagen**.

・Pantalla panorámica de 18.5” 16:9 con resolución HD, ofrece contenidos a pantalla completa con los colores más vibrantes.

・La entrada DVI soporta HDCP(High-Bandwidth Digital Content Protection).

・Los 16.7 millones de colores representan imágenes ricas y llenas de color.

- **Tecnología Splendid™ Video Intelligence**.

Integra un motor de color que detecta la activación y el uso de aplicaciones vídeo para optimizar la calidad de la imagen mediante mejoras de color, brillo, contraste y nitidez para ofrecerte los mejores resultados. Splendid™ también integra 5 modos de vídeo para escenas, teatro, gaming, imágenes nocturnas e imágenes estándar; 3 tonos de piel: amarillento, natural y rojizo, para añadir un toque humano a la reproducción de visuales. Además, podrás acceder a los modos vía hotkey.

2.5. SOFTWARE

Como ya hemos comentado, todo el software que hace funcionar nuestro sistema está compuesto por diferentes aplicaciones.

El software previo, tiene diferentes funciones:

- Por un lado la aplicación llamada 'álbum', es la encargada de la presentación en la pantalla táctil de las fotografías en forma de álbum.
La forma en que se ordenan las fotografías en el álbum es en función del orden en el que fueron hechas y del nombre de las fotografías.

Para la presentación en el álbum se utilizan las 'mini fotografías', ya que son más rápidas de cargar en pantalla, y tienen una calidad aceptable para la presentación.

- La segunda parte de este software es el de impresión, que gestiona la impresión de la fotografía. Esta impresión no se realiza de la copia en miniatura si no de la fotografía original.

3. APLICACIÓN SENSOR

Como hemos visto en nuestro sistema hardware, existe un sensor (catadióptrico), del cual podemos decir que es el origen de todo.
El sensor es el subsistema cuya misión es la de avisar a los demás subsistemas que tienen que actuar.

Como para la prueba y depuración de mi software no pude contar con el sistema hardware, tuve que hacer unas aplicaciones que 'simularan' estas partes del sistema, y esta es una de ellas.

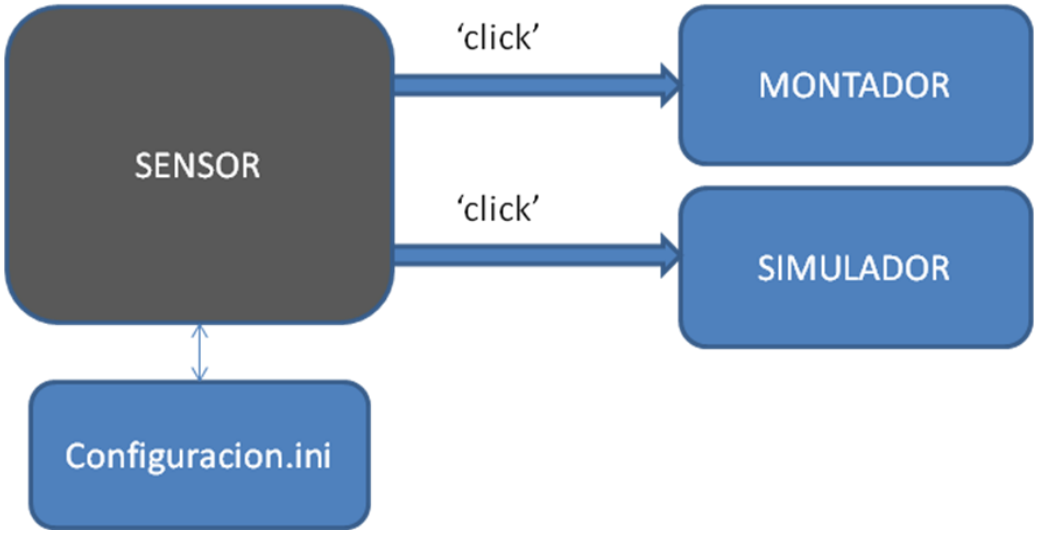
La 'aplicación sensor' es la que se encarga de simular el sensor del sistema.

Para su programación, los requisitos son:

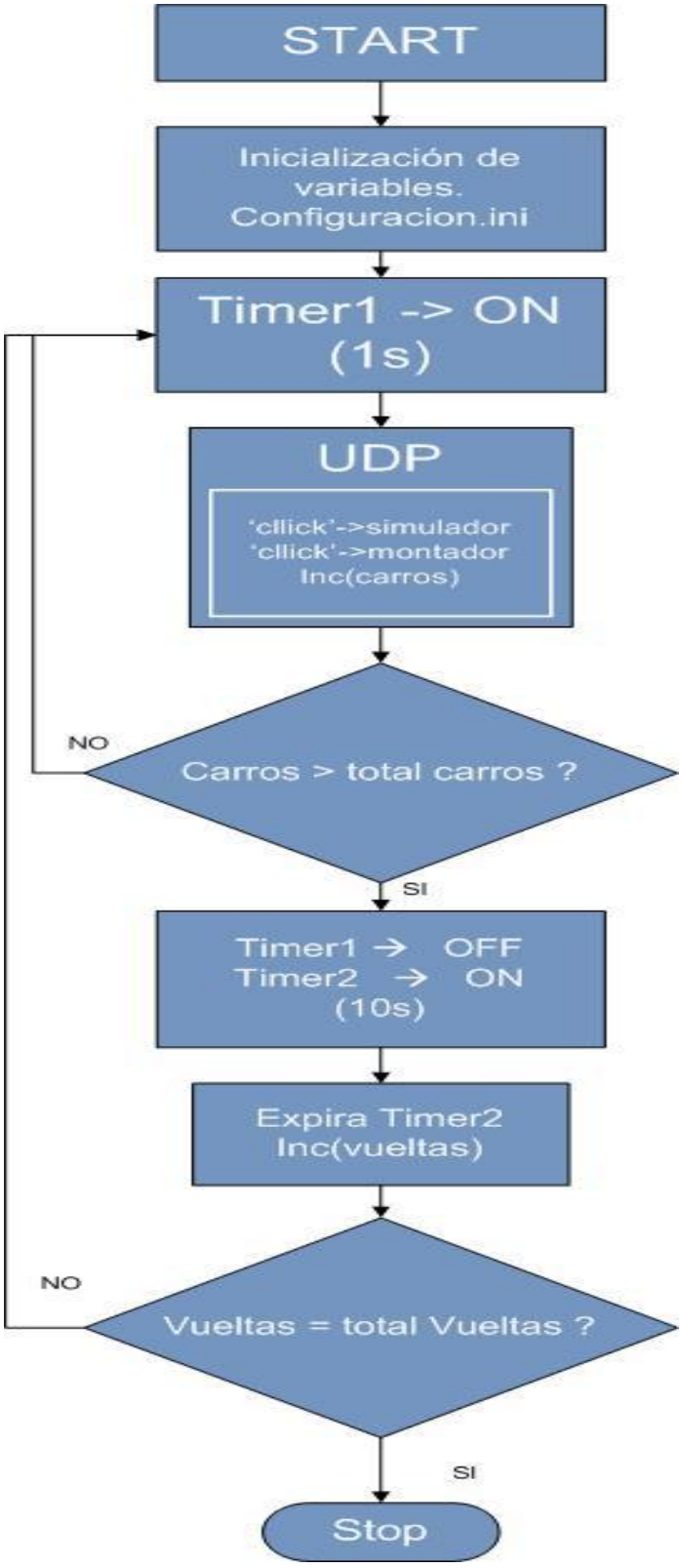
- La aplicación 'sensor' tomará los parámetros para su ejecución de un archivo exterior llamado 'configuracion.ini'. En este archivo, se indicará a la aplicación la duración en milisegundo entre una 'interrupción' y otra.

- La 'interrupción' será simulada por la aplicación enviando el string 'click' a cada uno de las aplicaciones relacionadas, que en este caso son la aplicación 'simulador' y la aplicación 'montador'.

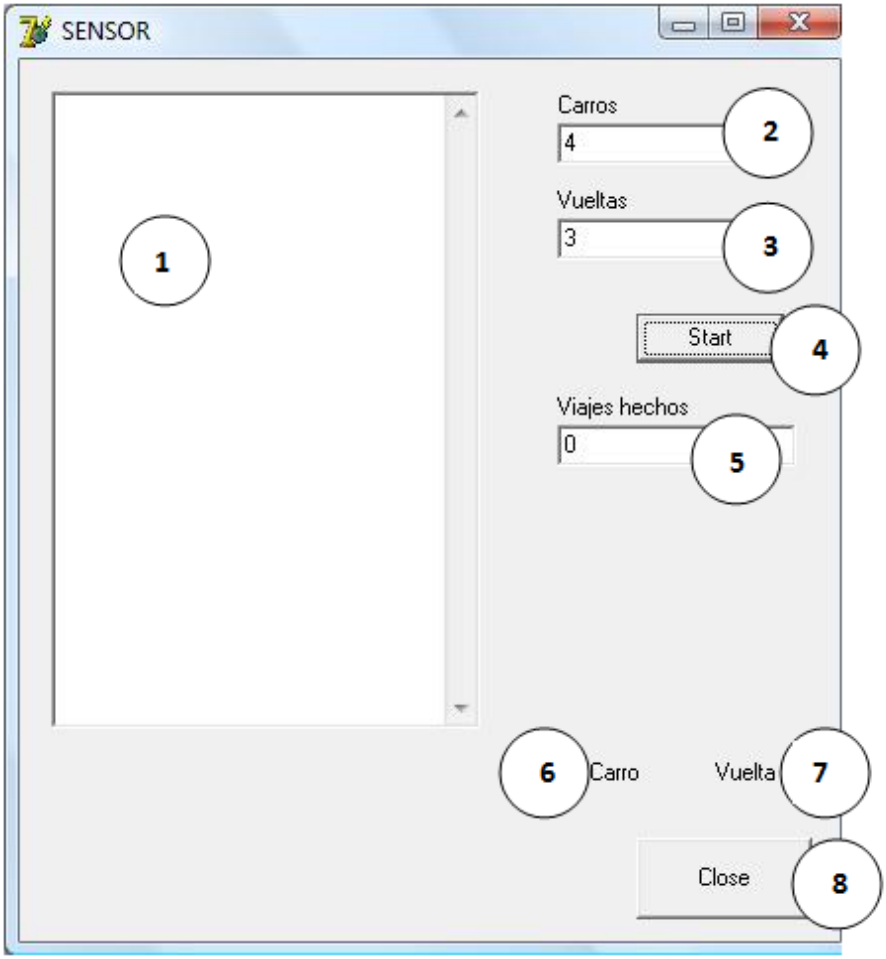
- La comunicación entre las aplicaciones se realizara a través del protocolo 'UDP'.



- Esquema de la caja negra sensor:



- La siguiente figura corresponde a la pantalla de la aplicación sensor:



A continuación a vamos a describir sus componentes:

1. Memo: Es la pantalla por la cual salen todos los mensajes de información y de control de programa Sensor.
2. Edit: En esta celda se introduce el número de carros por vuelta.
3. Edit: En esta celda se introduce el número de vueltas por viaje.
4. Button: Botón 'Start'. Al pulsar este botón se inicia el programa y se inicializan una serie de variables.
5. Edit: Celda que cuenta el número de viajes realizados.
6. Timer: Contador. Cuenta el tiempo entre un carro y otro.
7. Timer: Contador. Cuenta el tiempo entre una vuelta y otra.
8. Button: Botón Close. Al pulsar este botón se deshabilitan todas la comunicaciones y contadores y se cierra la aplicación.

Funcionamiento de la aplicación Sensor

A continuación vamos a explicar cómo funciona el simulador sensor.

Al presionar el botón 'Start' se inicializan una serie de variables.

Las variables que toman el tiempo que duran los contadores, se inicializan con los valores del archivo de configuración 'configuración.ini'. En el caso la variable del TimerCarros toma el valor de 1000 ms y la de TimerVueltas el de 10000 ms. Y el contador TimerCarros se pone en modo 'habilitado'.

Con el contador TimerCarros en modo 'habilitado', cada vez que este expira (cada segundo), la aplicación sensor envía el string 'click' a través de los puertos UDP a las aplicaciones 'simulador' y 'montador' y se incrementa el contador carros en una unidad.

Una vez que se han enviado todos los strings correspondientes a esa vuelta, el contador TimerCarros se deshabilita, y se habilita el contador TimerVueltas, que expira a los 10 segundos. En el momento en el que expira, incrementa una unidad a la variable que cuenta las vueltas. Si se han cumplido todas las vueltas del viaje la aplicación se para, sino se desactiva el contador TimerVueltas y se activa de nuevo el contador TimerCarros, y así hasta que se completen todas la vueltas de un viaje.

4. APLICACIÓN SIMULADOR

Otra parte del sistema hardware es la cámara fotográfica. En nuestro caso la aplicación simulador es la encargada de simular este hardware.

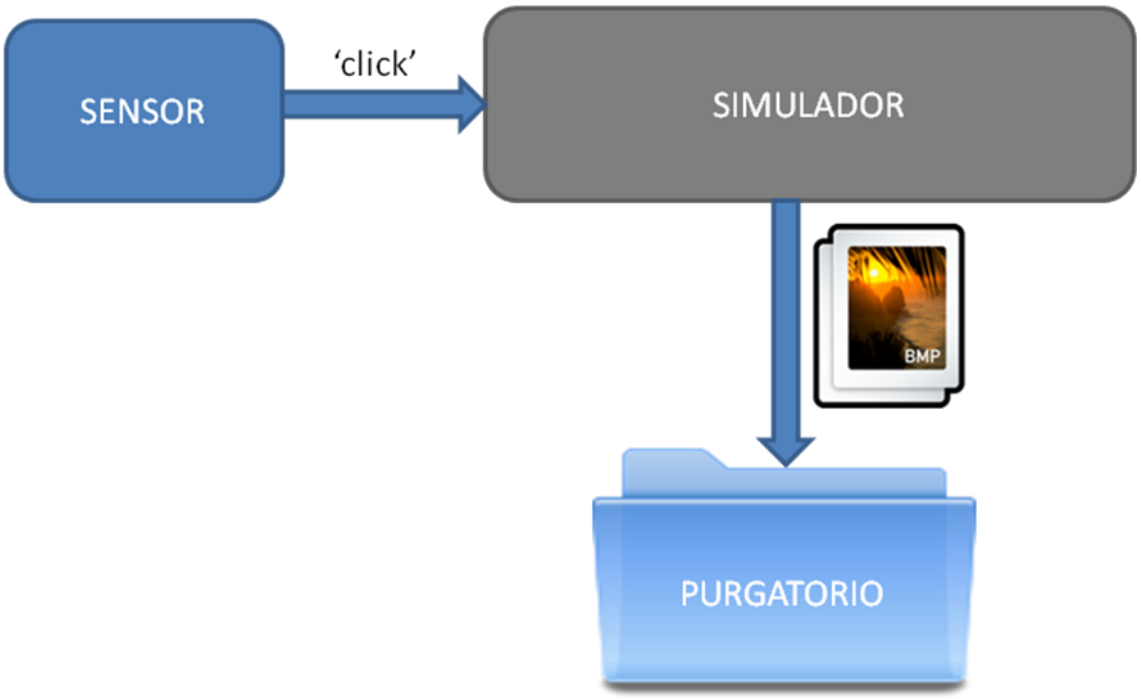
Para su programación los requisitos son:

- La aplicación simulador no estará activa hasta que la aplicación montador conozca todos los parámetros del sistema para su funcionamiento.

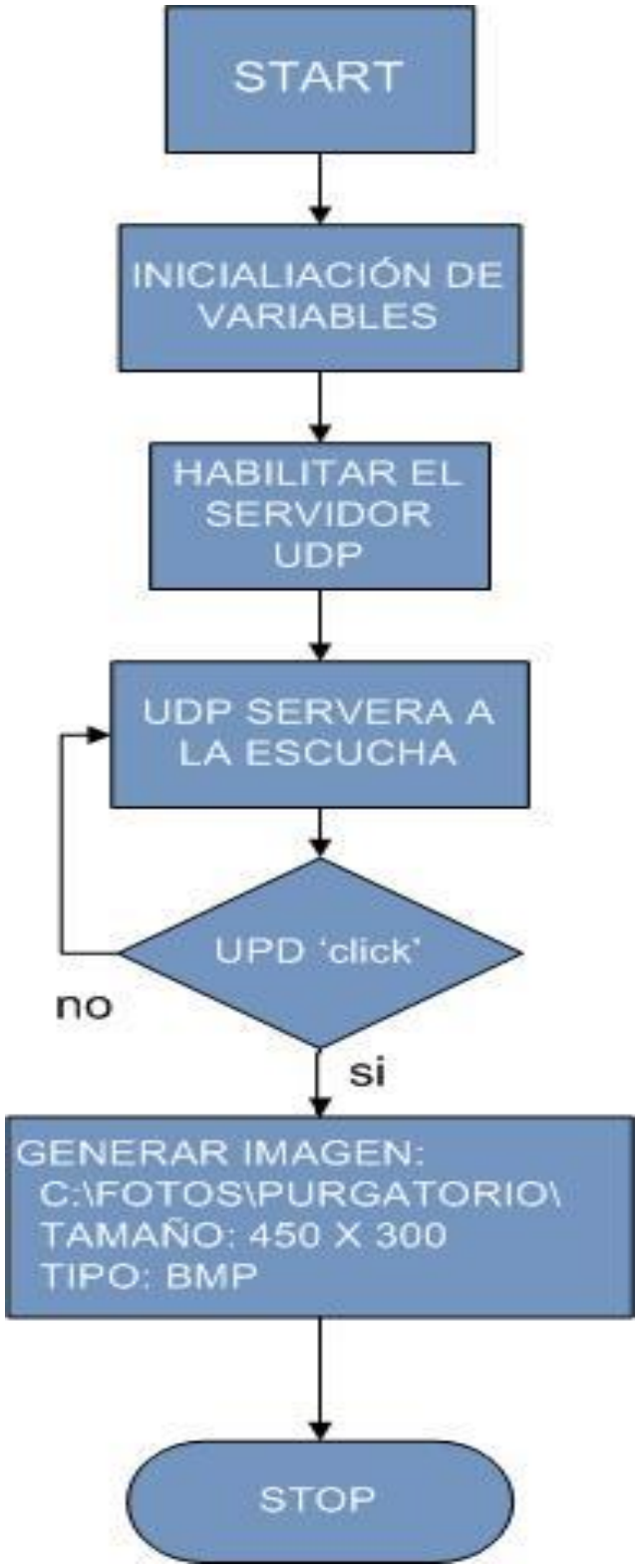
- La aplicación montador será la encargada de activar la aplicación simulador.

- Será una de las aplicaciones que reciba el 'click' por parte de la aplicación sensor, y lo hará a través de un servidor UDP.

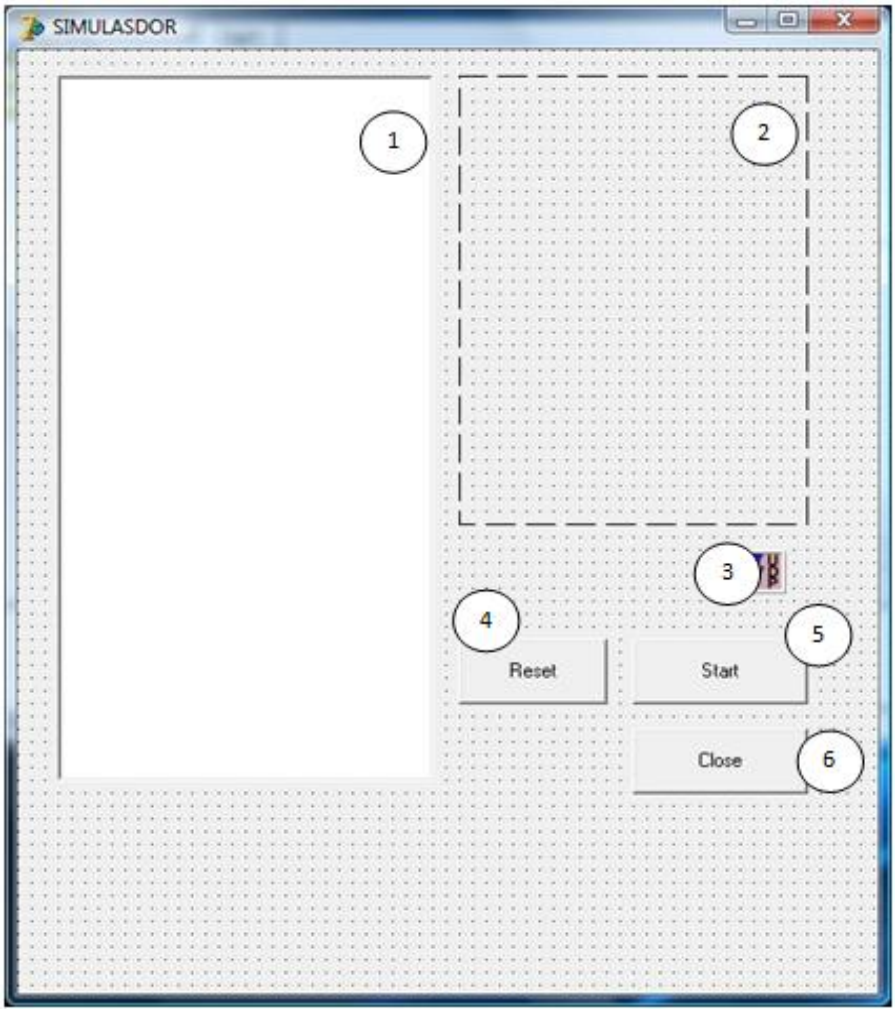
- Cada vez que la aplicación reciba un 'click' esta genera en la ruta 'C:\FOTOS\PURGATORIO\' una imagen en formato bmp. Esta imagen es la que simula una fotografía.



- Esquema de la caja negra sensor:



La siguiente figura corresponde a la pantalla de la aplicación simulador:



A continuación a vamos a describir sus componentes:

1. Memo: Es la pantalla por la cual salen todos los mensajes de información y de control de programa Sensor.
2. Image: Es el lugar de la pantalla donde se mostrarán las fotos en el instante siguiente a su creación.
3. IdUDPServer: Es el servidor UDP que está a la escucha de las 'señales' que reciba por parte del sensor.
4. Button: Botón de 'Reset'. Su función es la de limpiar la pantalla y reinicializar el contador que nombra las fotografías.
5. Button: Botón de 'Start'. Su función es la de limpiar la pantalla y habilitar el servidor UDP para ponerlo a la escucha.

6. Button: Botón de 'Close'. Su función es la de desactivar el servidor UDP y cerrar la aplicación.

A continuación vamos a explicar cómo funciona el simulador de la aplicación.

Esta aplicación se inicializa a 'petición' de la aplicación montador y cuando esta tiene todos los parámetros. Aunque tiene los botones Start/Stop para la inicialización manual de la aplicación.

Una vez que la aplicación está inicializada, también lo está el servidor y a la escucha. En ese momento, su misión es la de crear una fotografía en la ruta **'C:\FOTOS\PURGATORIO\'** cada vez que recibe un 'click'.

Cuando genera una fotografía/imagen, la aplicación la nombra con un número empezando desde el 1 y de manera ordenada. Si en cualquier momento se pulsara el botón 'Reset' se limpiaría la pantalla, y las fotografías empezarían a nombrarse desde el número 1 y de manera ordenada.

En el caso de que ya existiera en el directorio una imagen con el mismo nombre esta imagen se sobrescribiría.

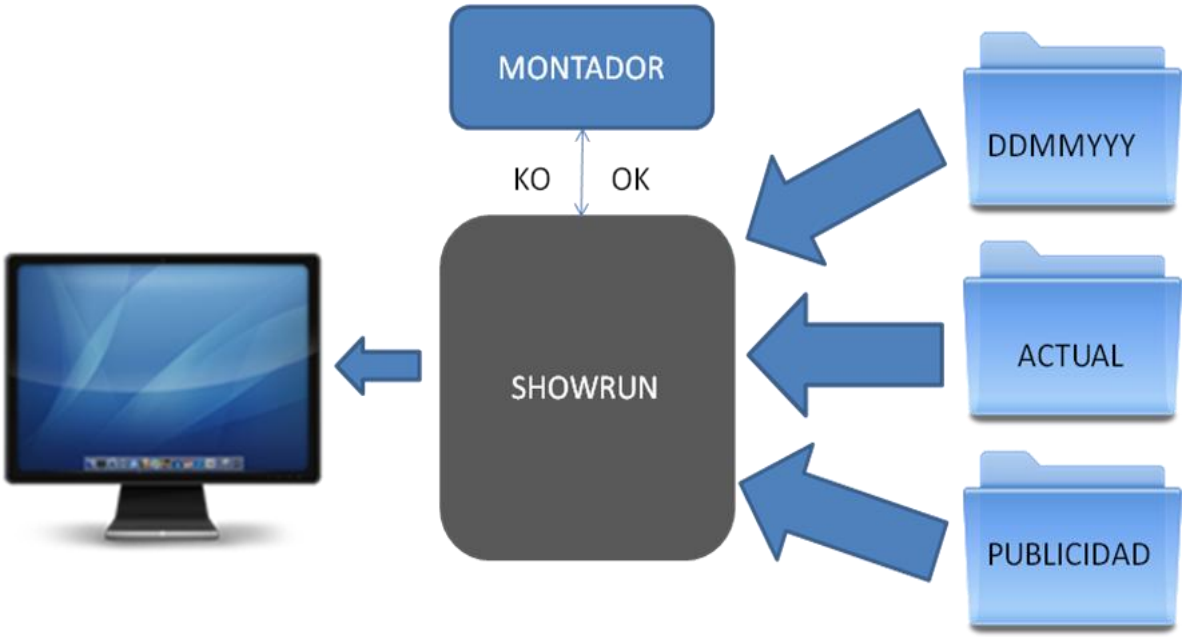
5. APLICACIÓN SHOWRUN

La aplicación showrun es parte del software del sistema, y su función es la de mostrar una galería de imágenes.

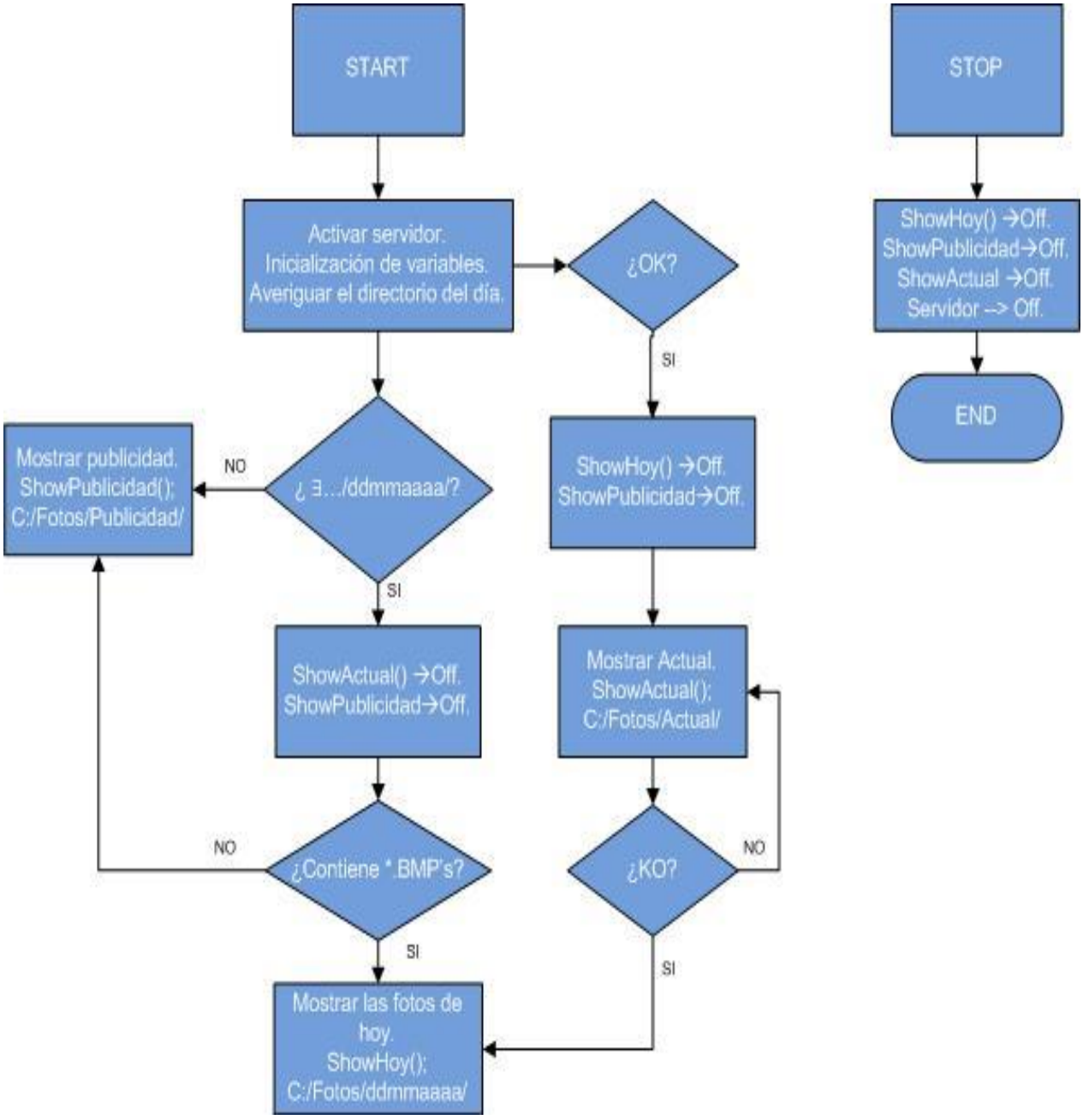
Para su programación, los requisitos son:

- Detectar si existe el directorio del día, y en el caso de que esté vacío, presentar publicidad.

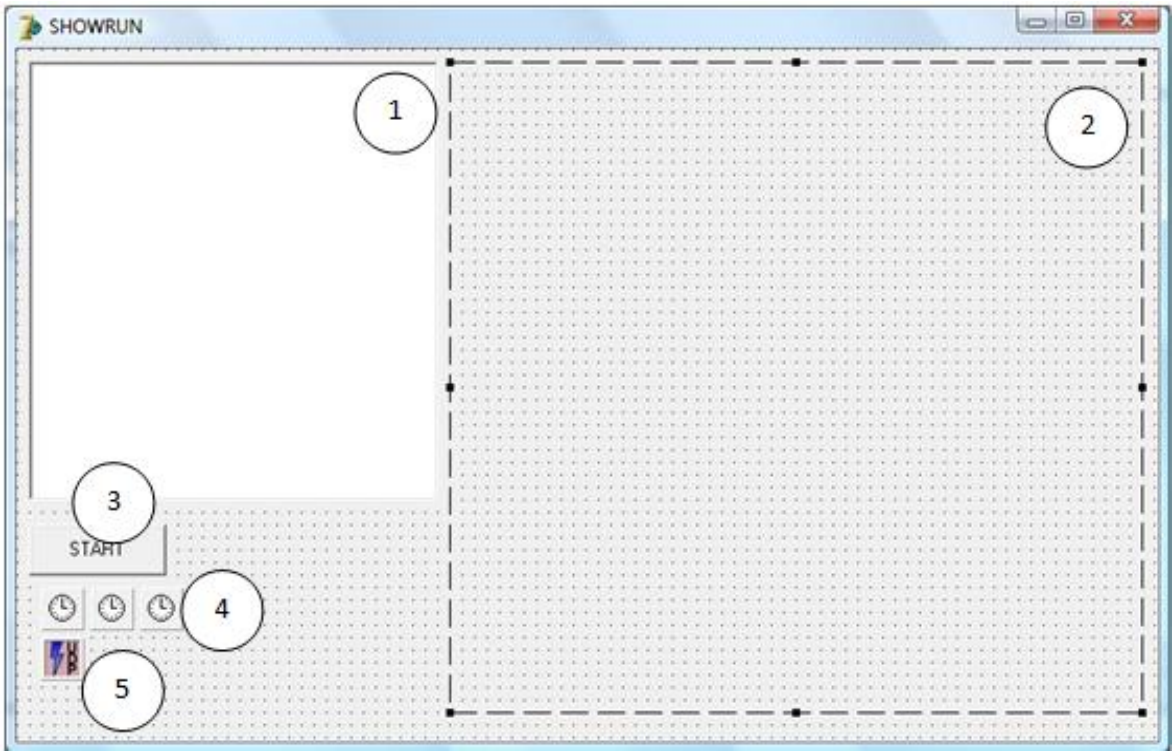
- Presentar las fotografías/imágenes de un directorio en función de los mensajes recibidos por la aplicación montador.



- Esquema de la caja negra showrun:



- La siguiente figura corresponde a la pantalla de la aplicación Showrun:



1. ListBox: Es la pantalla por donde salen todos los mensajes de información y de control.
2. Image: Región del formulario donde se muestra la galería de imágenes correspondiente.
3. Button: Botón 'Start/stop'. Su función es la de poner la aplicación en funcionamiento y parar su funcionamiento.
4. Timers: Son 'relojes' cuya función es la de controlar el tiempo de transición entre una fotografía u otra.
5. IdUDPServer: Servidor udp cuya función es la de recibir a través de puerto 8010, los mensajes provenientes de la aplicación montador.

A continuación vamos a explicar cómo funciona la aplicación showrun.

Esta aplicación comienza su funcionamiento presionando el botón 'Start'. En ese momento calcula el día de hoy en formato **'C:\FOTOS\ddmmaaaa\'**. Si existe el directorio, y se hallan fotografías en él, las muestra y sino, muestra las fotografías de publicidad que hay en **'C:\FOTOS\PUBLICIDAD\'**.

Mientras muestra las fotografías de publicidad, el servidor está escuchando y en el momento que reciba el mensaje 'ok' deja de mostrar publicidad, para mostrar las fotografías que se están guardando en ese momento en el directorio **'C:\FOTOS\ACTUAL\'**, correspondiente al viaje en curso. Cuando el viaje finalice el servidor recibirá el mensaje 'ko'. En ese instante mostrará las fotografías **'C:\FOTOS\ddmmaaaa\'**, que es el directorio donde se encuentran todas las fotografías del día, y así hasta que vuelva a recibir el mensaje 'ok' o se pare la aplicación (STOP).

6. APLICACIÓN MONTADOR

La aplicación montador es parte del software del sistema, y podemos decir que es el corazón del sistema y que se encarga de automatizar la gestión de las fotografías.

Para su programación los requisitos son:

- Detectar el principio de un viaje.

- Detectar viajes de vueltas, número de vueltas por viaje, y carros por vuelta (parámetros del sistema).

- No habilitar la cámara hasta que se obtengan los parámetros del sistema.

- Verificar, y crear si no existiese, el directorio raíz **'C:\FOTOS**.

- Verificar, y crear si no existiese, el directorio del día actual **'C:\FOTOS\ddmmaaaa\'**, y el directorio **'C:\FOTOS\ddmmaaaa\MINI\'**.

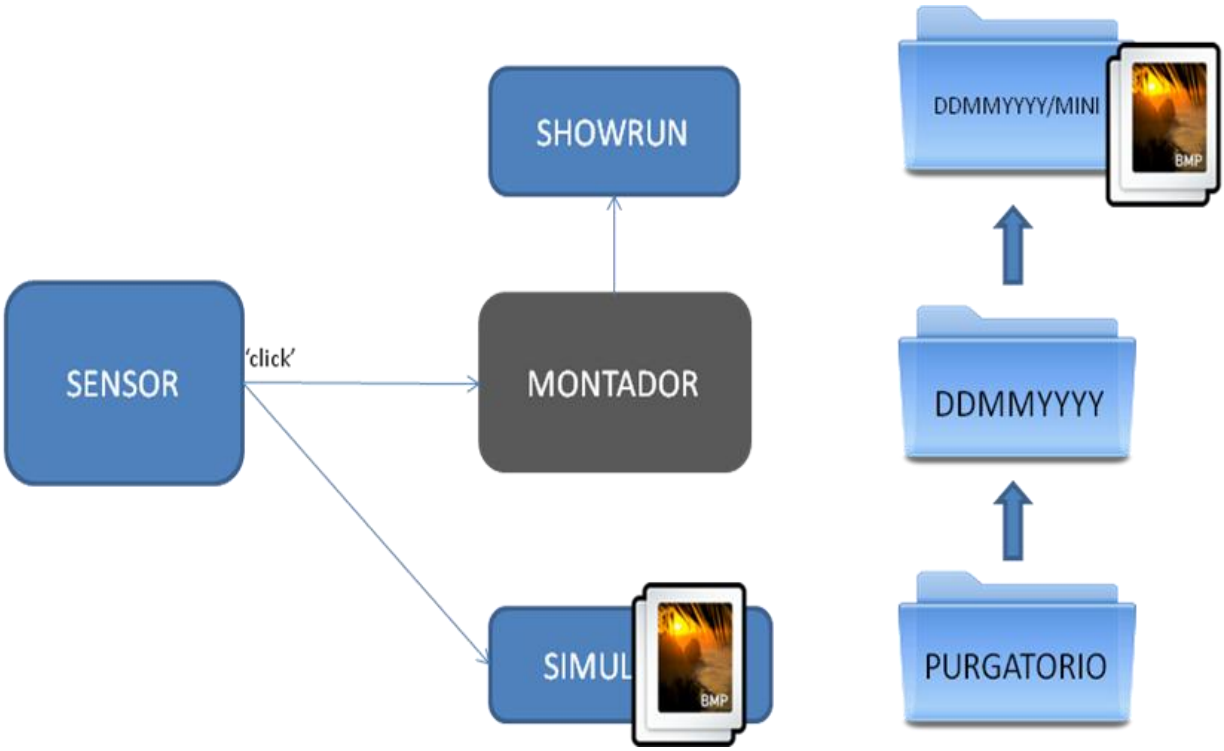
- Cuando se inicie un viaje, y durante su duración, tomará las fotos del directorio **'C:\FOTOS\ PURGATORIO\'** correspondiente a ese viaje y las moverá al directorio **'C:\FOTOS\ACTUAL\'**, asignándoles su nombre identificativo.

- El nombre identificativo tiene el formato **Vxvycz**, donde 'x' es el número del viaje, 'y' es el número de vuelta, y 'z' es el número de carro.

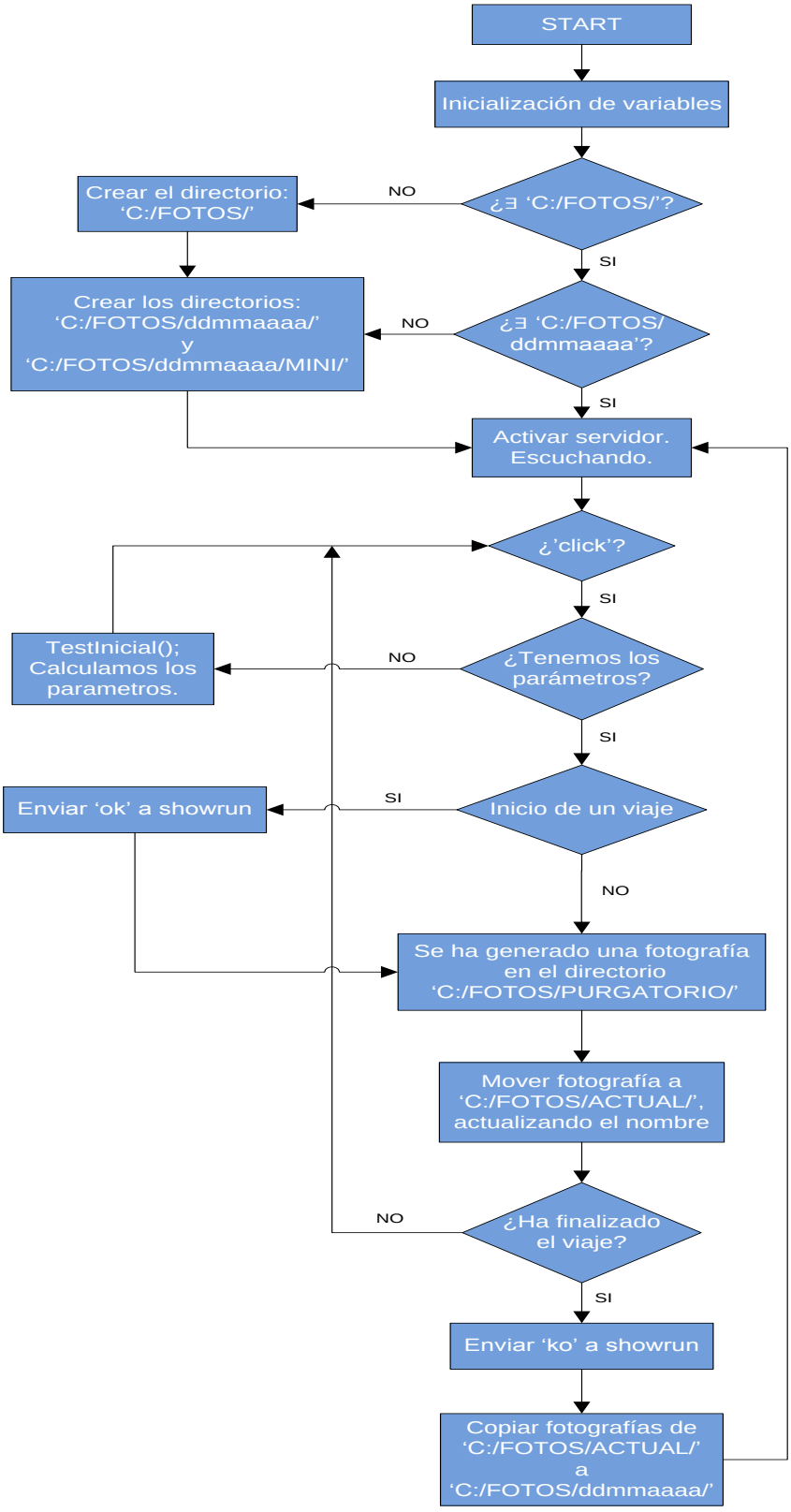
- La aplicación generara una foto en miniatura de cada una de las fotos que se generen y las guardará en el directorio **'C:\FOTOS\ddmmaaaa\MINI\'**.

- A la finalización de cada viaje las fotos se copiarán del directorio **'C:\FOTOS\ACTUAL\'** al directorio del día **'C:\FOTOS\ddmmaaaa\'**.

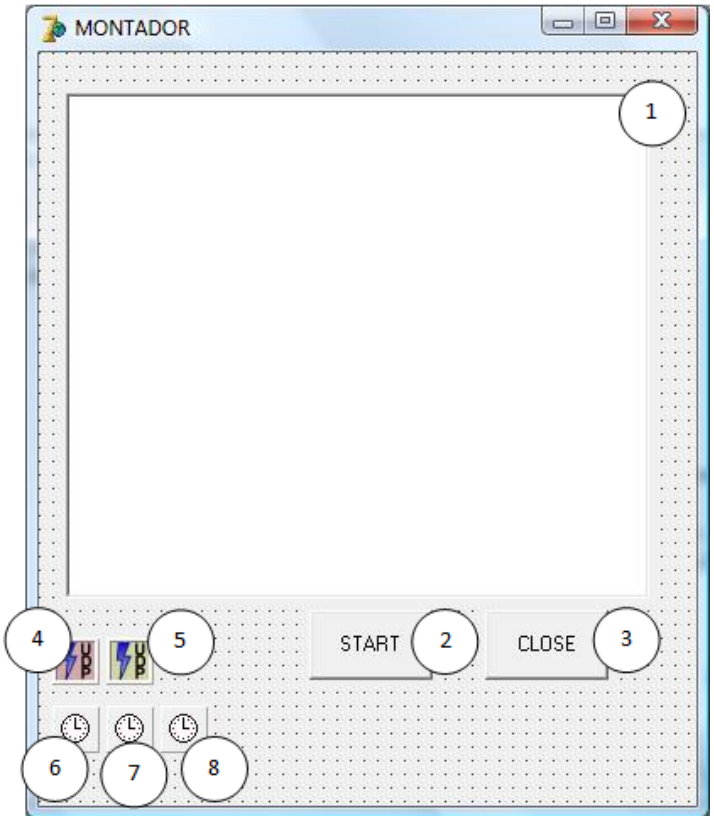
- Si al inicio de un nuevo viaje existen fotos en el directorio **'C:\FOTOS\ACTUAL\'**, de un viaje anterior, estas serán eliminada.


SHOWRUN
DDMMYYYY/MINI
BMP
SENSOR
'click'
MONTADOR
DDMMYYYY
SIMUL
BMP
PURGATORIO

- Esquema de la caja negra montador:



- La siguiente figura corresponde a la pantalla de la aplicación Montador:



1. ListBox: Es la pantalla por la cual salen todos los mensajes de información y de control.
2. Button: Botón 'Start'. Su función es la de poner la aplicación en funcionamiento e inicializar las variables del sistema. Se evalúa si se tienen o no los parámetros del sistema, y queda a la espera de obtener los parámetros o que llegue un 'inicio de viaje'.
3. Button: Botón 'Stop'. Su función es la de deshabilitar los Timer's, el servidor y parar la aplicación.
4. IdUDPServer: Servidor UDP, cuya función es la recibir las señales ('click') de la aplicación sensor y actuar en función de los parámetros del sistema.
5. IdUDPClient: Cliente UPD, cuya función es la de enviar la señal 'ok' y 'ko' a la aplicación Showrun, en función si comienza o termina un viaje.
6. Timer: Timer1. Se utiliza en la función 'TestInicial', cuando vence el Timer1 significa que el siguiente 'click' recibido es el inicio de un viaje.
7. Timer: Timer2. Se utiliza en la función 'TestInicial'. Cuando vence el Timer2 significa que ha finalizado un vuelta.

8. Timer: Timer3. Se utiliza en la función 'TestInicial'. Cuando vence el timer3 significa que ha finalizado le viaje.

A continuación vamos a explicar cómo funciona la aplicación montador.

La aplicación comienza su funcionamiento al presionar el botón 'start'. En ese momento se inicializan las variables de la aplicación y además se evalúa si existe el directorio raíz **('C:\FOTOS\')**, y el directorio y subdirectorio de trabajo del día **('C:\FOTOS\ddmmaaaa\' y 'C:\FOTOS\ddmmaaaa\MINI\')**, y en el caso de que no existan los crea.

Ya estamos listos para 'escuchar' las señales de la aplicación sensor, por tanto habilitamos el servidor y este se pone a la escucha. En este instante se pueden dar dos situaciones, que empecemos a recibir señales del sensor o que no. En cualquier caso, la aplicación aún tiene que 'aprender' los parámetros del sistema (el número de carros, número de vueltas por viaje…etc), entonces sí:

- Empiezan a llegar señales al servidor ('click's). La aplicación hace caso omiso de ellas hasta que dejen de llegar. Entonces espera un tiempo (15s), y si en ese tiempo no llega ninguna señal, la aplicación considera que en la siguiente señal recibida estará al inicio de un nuevo viaje.

- No recibe ninguna señal, espera un tiempo (15s), y si en ese tiempo no llega ninguna señal, la aplicación considera que en la siguiente señal recibida estará al inicio de un nuevo viaje.

En este momento, con el siguiente 'click' recibido, y asumiendo que estamos al inicio de un nuevo viaje, la aplicación inicializará un test para 'aprender' los parámetros del sistema, que durará todo lo que perdure el viaje en el que se está realizando el test.

A la finalización del test la aplicación montador ya conoce cuales son los parámetros del sistema y puede empezar a gestionar las fotografías.

Si en este marco, (donde ya conocemos los parámetros del sistema) la aplicación montador recibe otra señal, sabrá que se ha tomado una fotografía y se ha guardado en el directorio **'C:\FOTOS\PURGATORIO\'**. En ese instante toma la foto y la mueve al directorio **'C:\FOTOS\ACTUAL\'** con su nombre correspondiente.

Por ejemplo si esa fotografía corresponde al carro uno de la vuelta primera del primer viaje, la fotografía se guardaría como **'V1v1c1.bmp'**. Si fuese la fotografía correspondiente al carro quinto de la tercera vuelta del viaje segundo, se guardaría como **'V2v3c5.bmp'**.

Durante la duración del viaje, las fotografía correspondientes a ese viaje se guardan en el directorio **'C:\FOTOS\ACTUAL\'** y tras la última fotografía de cada viaje se hará una copia de todas las fotografías al directorio del día **'C:\FOTOS\ddmmaaaa\'**.

Hasta el inicio del siguiente viaje habrá dos copias de las fotografías del último viaje, una en **'C:\FOTOS\ACTUAL\'**, y otra en **'C:\FOTOS\ddmmaaaa\'**.

En el inicio del siguiente viaje se borraran todas las fotografías del directorio **'C:\FOTOS\ACTUAL\'**, y todo comienza de nuevo.

Cada vez que una fotografía se mueva del directorio **'C:\FOTOS\PURGATORIO\'** al **'C:\FOTOS\ACTUAL\'**, la aplicación realizará una copia de menor tamaño en el directorio **'C:\FOTOS\ddmmaaaa\MINI\'**, la cual tendrá el mismo nombre que la fotografía original pero precedida de la palabra 'MINI-'. Por ejemplo si la foto original se llama **'V1v1c1.bmp'**, su correspondiente seria **'MINI-V1v1c1.bmp'**.

Además nuestra aplicación, a través del cliente UDP, enviará a la aplicación showrun la señal 'ok' al inicio de cada viaje y 'ko' a la finalización del mismo.

7. COMUNICACIÓN ENTRE APLICACIONES

Como hemos podido ver a lo largo de los puntos anteriores, el proyecto está compuesto de una serie de aplicaciones que se comunican entre sí.

Para 'producir' esta comunicación vamos a utilizar el paradigma Cliente/Servidor y dentro de éste, el protocolo UDP.

¿Por qué el protocolo UDP?

Si analizamos el sistema completo, tanto la parte software como hardware, vemos que todo se centraliza dentro de una misma computadora. Por lo tanto podemos afirmar que nuestras aplicaciones trabajarán en un 'entorno' estable y robusto donde la probabilidad de un fallo en la comunicación o pérdida de un paquete es muy poco probable.

En estas condiciones lo más eficiente es utilizar UDP, que aunque no sea un protocolo orientado a la conexión, en un entorno seguro nos proporciona rapidez y efectividad, cualidades deseadas en sistemas que funcionan en tiempo real como el nuestro.

La aplicación del protocolo UDP en delphi se ha realizado con componentes 'Indy'. Sus autores lo definen como Internet direct (Que es lo que significa Indy), un grupo de componentes de código abierto escritos en Delphi y basados en "Blocking sockets".

Para nuestras aplicaciones hemos usado tanto servidores como cliente UDP.

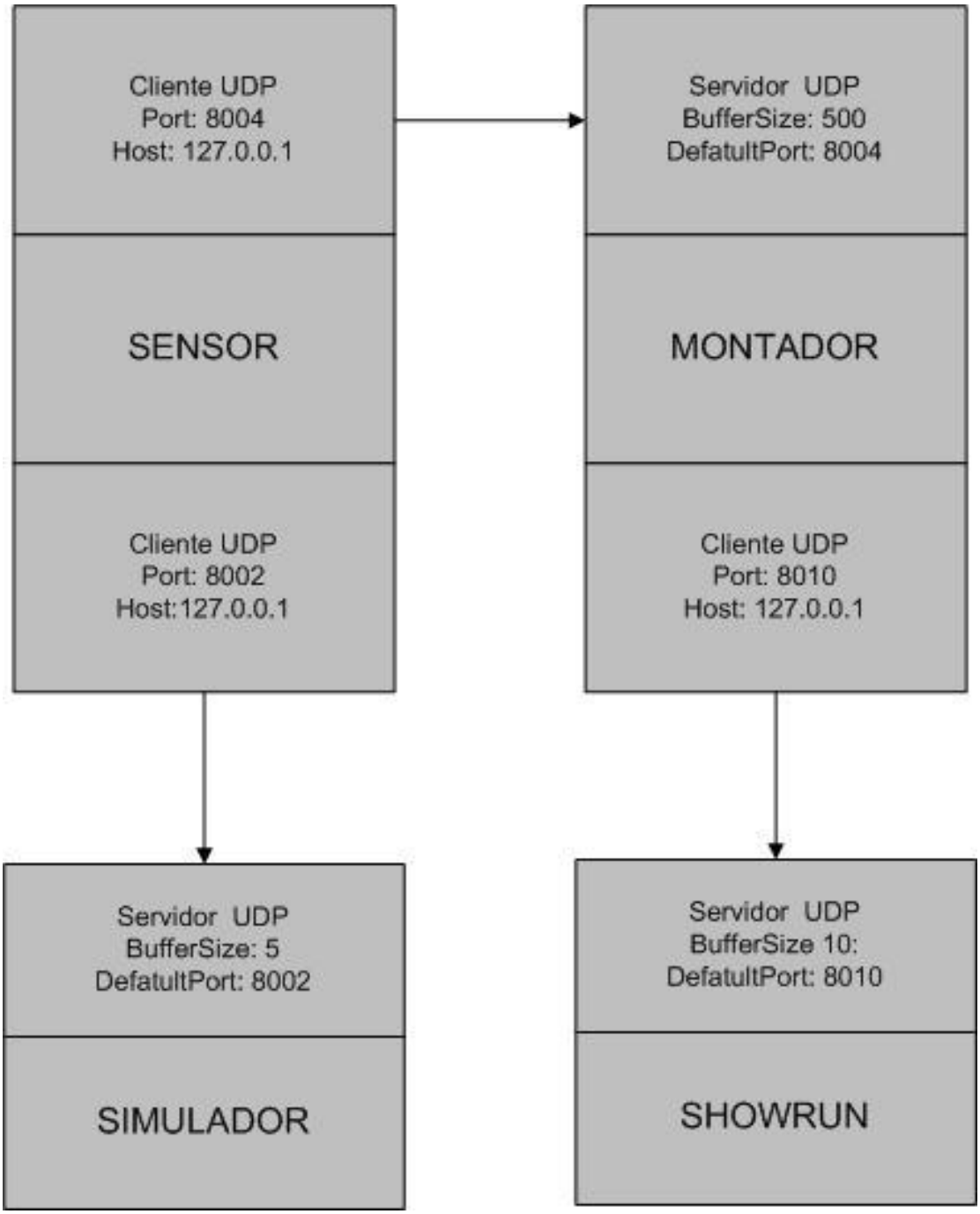
Para el servidor UDP tendremos que configurar las siguientes propiedades:

- Buffersize: Tamaño del buffer.
- DefaultPort: Puerto por el que 'escucha' o recibe.

Para el cliente UDP tendremos que configurar las siguientes propidades:

- Port: Puerto por el que 'habla' o emite.
- Host: dirección ip donde se encuentra el servidor.

- La siguiente figura muestra la interconexión entre las aplicaciones:



8. DIFICULTADES DURANTE EL DESARROLLO Y COMO SE SALVARON

Cuando Francisco Miguel Monzó Sanchez, tutor en mi proyecto fin de carrera, me habló del mismo, me dijo que el lenguaje en el que yo efectuara el proyecto podía ser de mi elección. Si por mi parte había alguna preferencia, por él no habría problema. Pero él me aconsejó que podría realizarlo en Delphi ya que el podría orientarme en este lenguaje.

Y aunque poder contar con mi tutor para orientarme en la programación de las aplicaciones era un punto a mí favor, fue precisamente mi desconocimiento sobre Delphi lo que en última instancia me hizo decidirme por él como lenguaje para el proyecto.

Realizar el P.F.C en Delphi me daría valor añadido como profesional, y además conocería otro paradigma de programación.

Una de las primeras dificultades que me encontré fue que no conocía nada de Delphi, por tanto tenía que ir aprendiendo a la vez que lo desarrollaba, lo que en algunas ocasiones me llevó a tener que 'reprogramar' lo ya hecho, ya que conforme adquiría más experiencia podía reducir algunos segmentos de código a una línea, e implementar otros de forma más eficiente.

De todos los problemas que me fueron surgiendo puedo destacar dos, que son lo que más trabajo me llevó encontrar una solución.

El primero me surgió en la aplicación montador.
Cuando se genera una fotografía en el subdirectorio…\PURGATORIO\, la aplicación montador recibe una señal avisando, que en ese momento tiene que 'mirar' en el subdirectorio, tomar la fotografía, ponerle el nombre correspondiente y mover la foto a otro subdirectorio. El problema surge cuando la aplicación mira y no encuentra la fotografía, esto ocurre porque el sistema operativo no la tiene disponible.

Después de investigar e intentar varias soluciones, pude solucionarlo durmiendo la aplicación durante 95 ms antes de buscar la fotografía. Con esta solución todo funciona de forma correcta.

El segundo problema surgió en la aplicación showrun.
Para hacer las presentaciones, la aplicación miraba en el directorio correspondiente y mediante un bucle while mostraba las fotografías. El problema era que si utilizaba un bucle para mostrar las fotografías, durante la duración del bucle, la aplicación se quedaba 'colgada' y no se podía presionar ningún botón, ni cerrar la aplicación. Además si movía el formulario de la aplicación por el escritorio durante la transición de una fotografía a otra, se producía un error y la aplicación se bloqueaba.

Para solucionar este problema, tuve que rehacer la aplicación por completo, y añadir un nuevo elemento para poder cargar las fotografía una a una antes de que se muestren. Así pude evitar el bucle, la aplicación no se 'cuelga' y puedo mover el formulario libremente por el escritorio sin que falle.

9. SOFTWARE ¿MÁS O MENOS INTELIGENTE?

Esta fue una de las cuestiones que Francisco Miguel Monzó y yo estuvimos debatiendo cuando me detallaba los requisitos del software.

Dotar al software de mayor o menor 'inteligencia', en cierto modo, y permitiéndome el uso de la palabra 'inteligencia', es dejar al control del software ciertas variables del sistema.

En nuestro caso, de las aplicaciones programadas, es la aplicación 'montador' la que se ha dotado de cierta inteligencia, ya que es el motor de nuestro software.

Al 'montador' se le ha dotado la capacidad de aprender cuando empieza y termina un viaje, el numero de carros que tiene que fotografiar y el número de vueltas de cada viaje. Además de evaluar si existen o no el directorio raíz y el directorio de trabajo del día.

10. APLICACIÓN PRÁCTICA

La aplicación práctica para la cual se ha desarrollado este software es, por ejemplo, para las atracciones de feria como el 'tren de la bruja', y las atracciones clásicas de coches y motos que recorren un trazado preestablecido. Aunque es seguro que será la imaginación la única barrera que pondrá freno a nuevas utilidades.

A continuación vamos a explicar una aplicación práctica de uso de nuestro software.

La primera aplicación en iniciarse y empezar a funcionar es la aplicación montador:

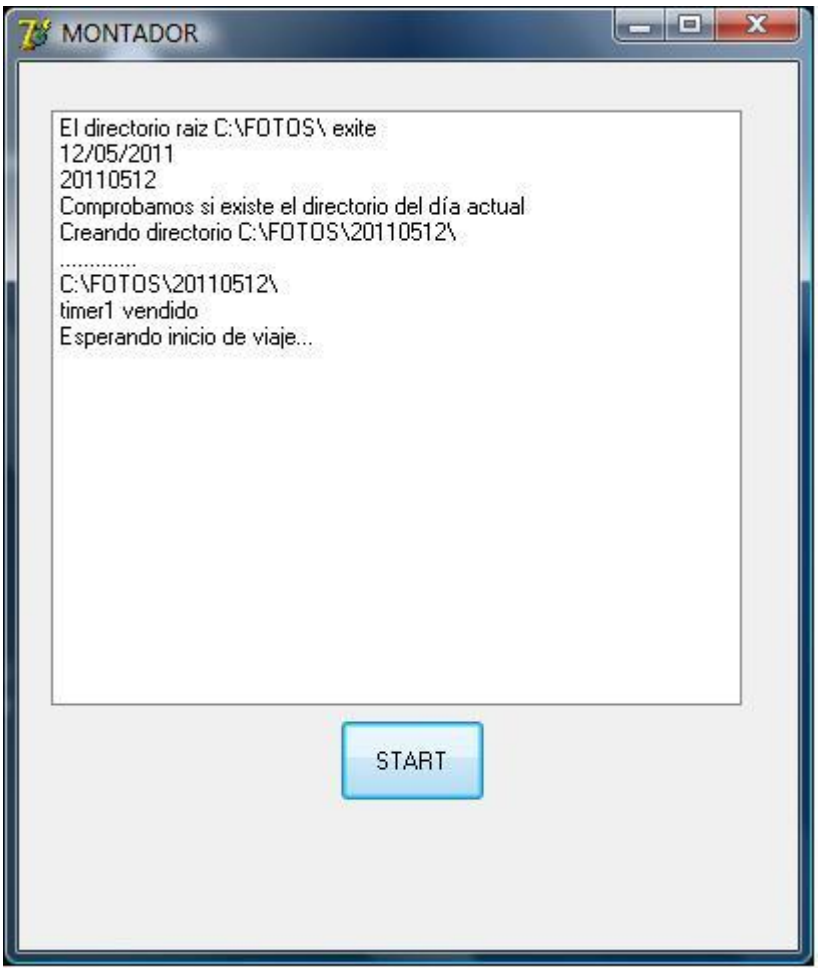


Figura ejemplo de la aplicación 'Montador'

Como vemos en la figura anterior, cuando presionamos el botón ‘start’ la aplicación comienza su funcionamiento.

Analiza si existe el directorio raíz, como en nuestro caso es así, a continuación analiza si existe el directorio del día de hoy (12/05/2011). Este directorio no existe, por tanto lo crea, para ello pasa la fecha 12/05/2011 al formato 20110512, para ponerle este nombre al subdirectorio del día de hoy.

A continuación se queda a la espera de recibir señales por parte de la aplicación sensor, y como no recibe nada en un tiempo predeterminado, asume que la siguiente señal que reciba será el inicio de un nuevo viaje.

Ahora ponemos en funcionamiento la aplicación showrun:

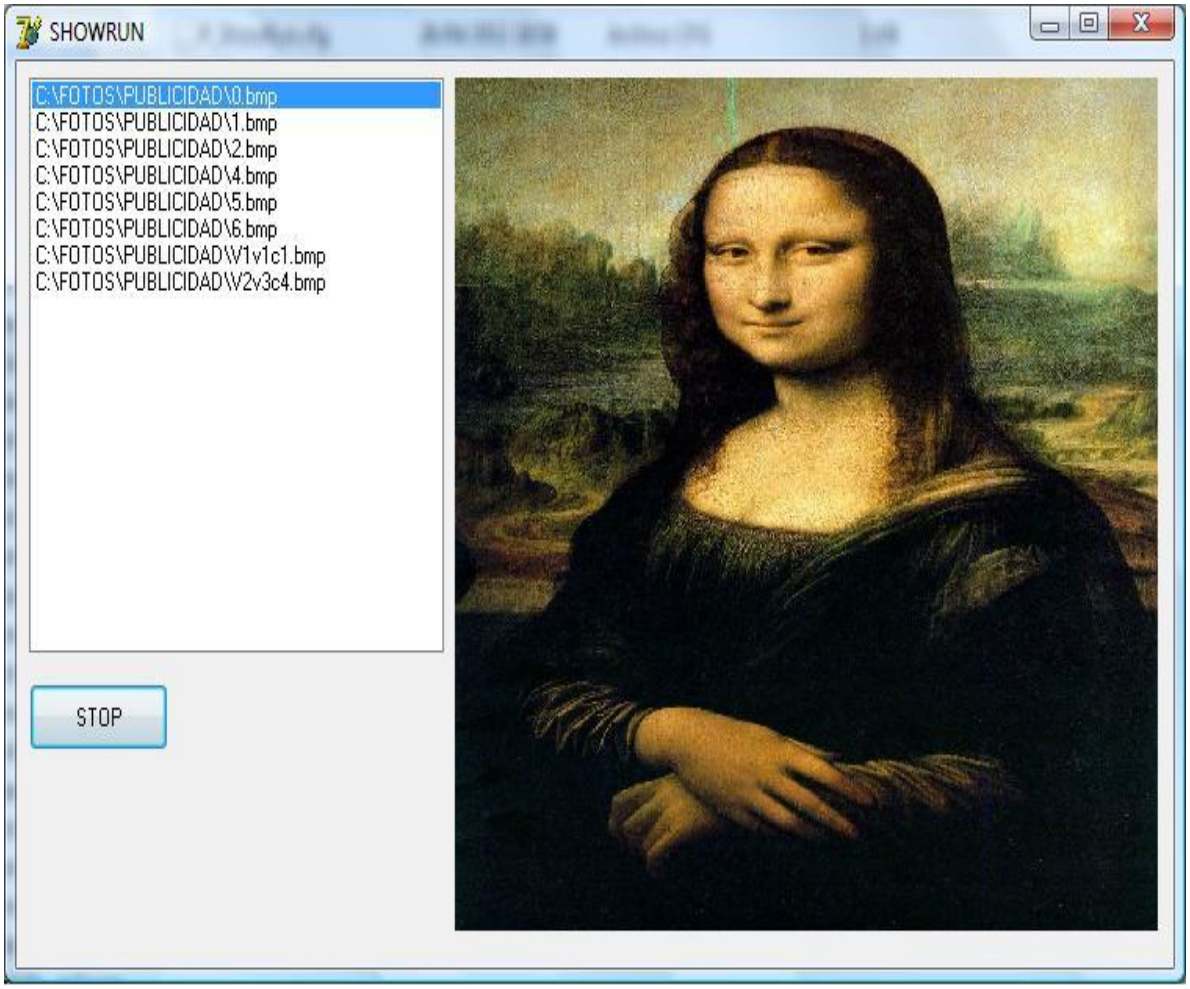


Figura ejemplo de la aplicación ‘showrun’

La aplicación Showrun funciona como se programó, aunque existe el subdirectorio del día actual **'C:\FOTOS\20110512\'**, está vacio en este momento, por tanto empieza a mostrar las fotografías que se encuentran dentro del directorio de publicidad **(C:\FOTOS\PUBLICIDAD\)**.

A continuación inicializamos la aplicación 'sensor':

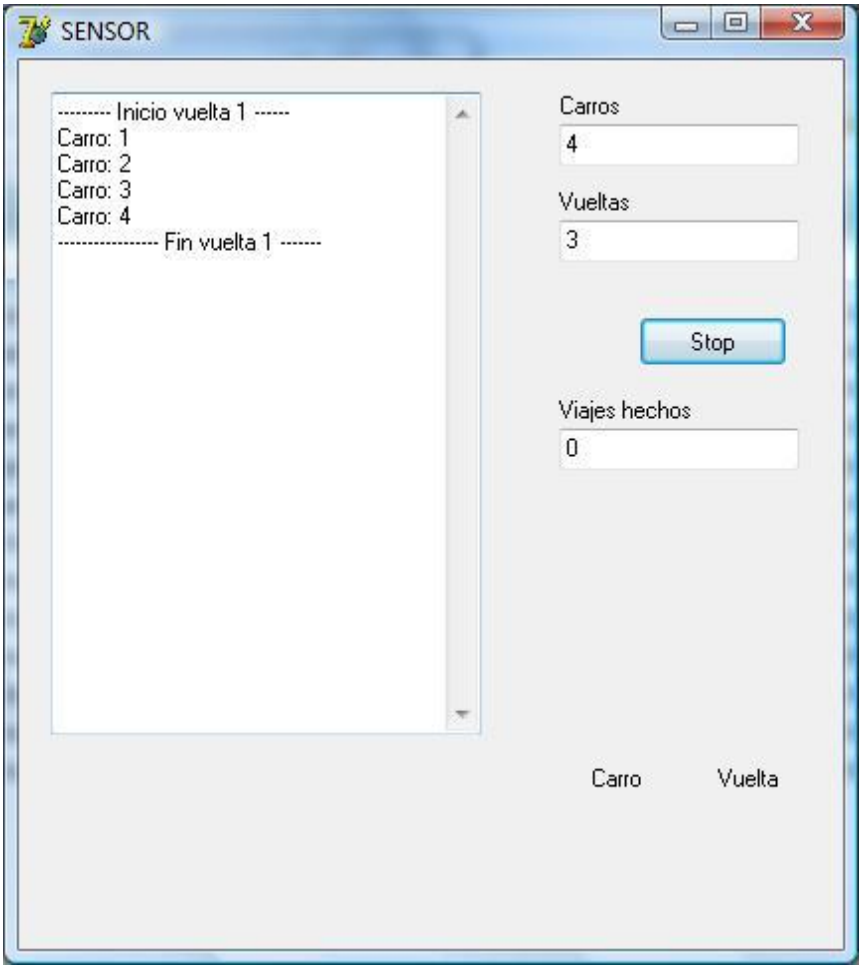
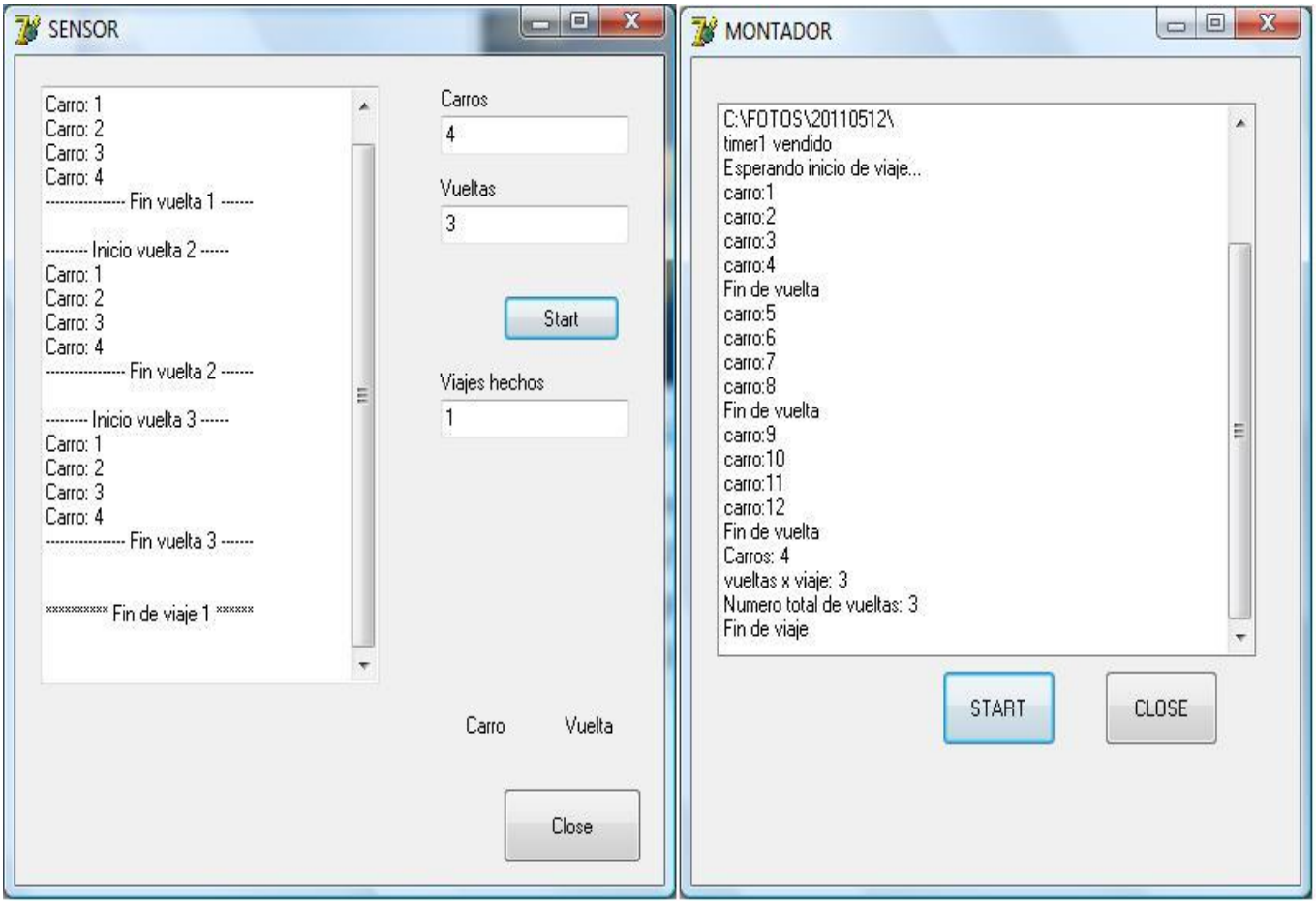


Figura ejemplo de la aplicación 'sensor'.

Como vemos, se han introducido los valores para el número de vueltas por viaje, tres, y el número de carros por vuelta, cuatro. Esto significa que por cada vuelta la aplicación enviará cuatro 'click´s' a la aplicación montador, y otros cuatro a la aplicación simulador para que genere la fotografía correspondiente.

Para la aplicación montador, cuando el sensor termine el viaje, ya tendrá los parámetros del sistema.



Figuras ejemplo de las aplicaciones 'sensor y montador'.

Una vez la aplicación Montador conozca todos los parámetros, lanzará la aplicación Simulador.

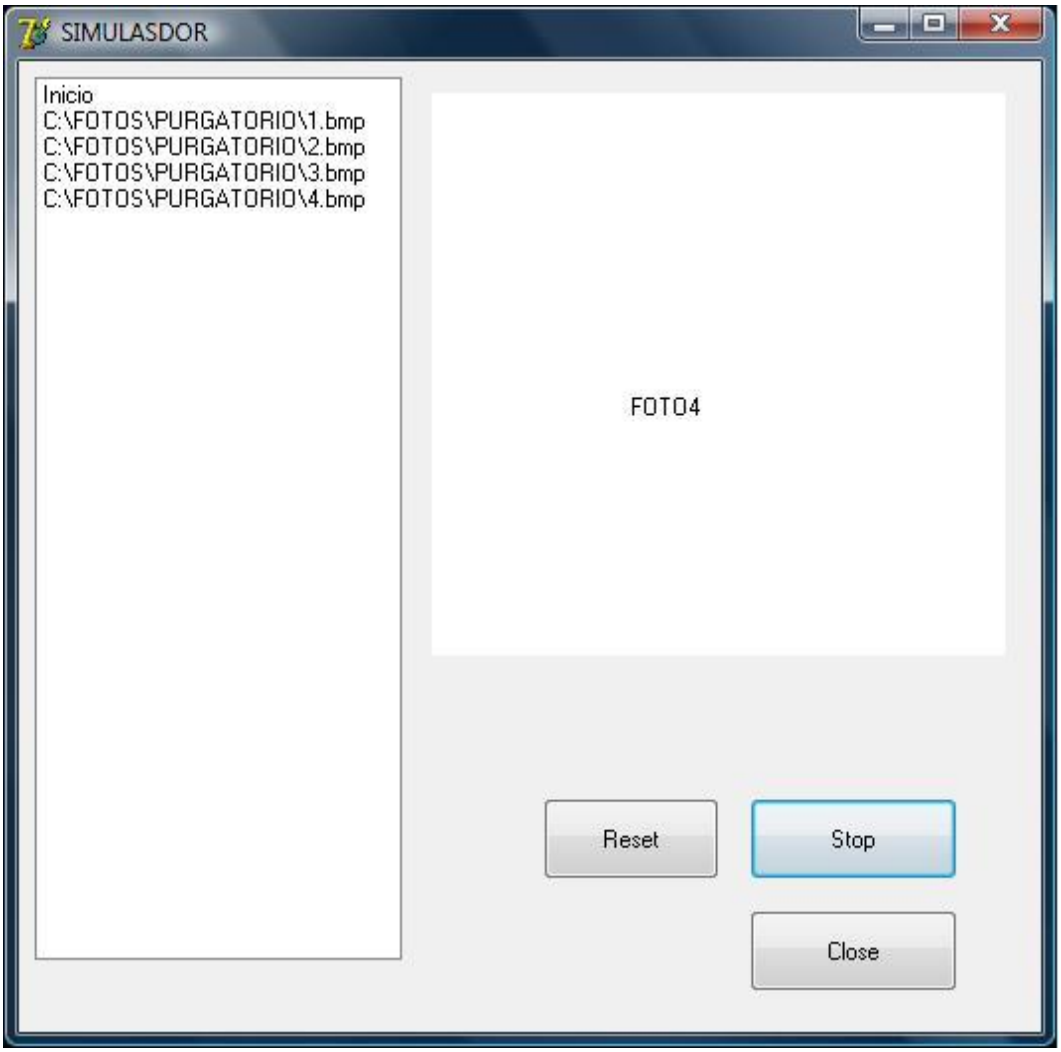
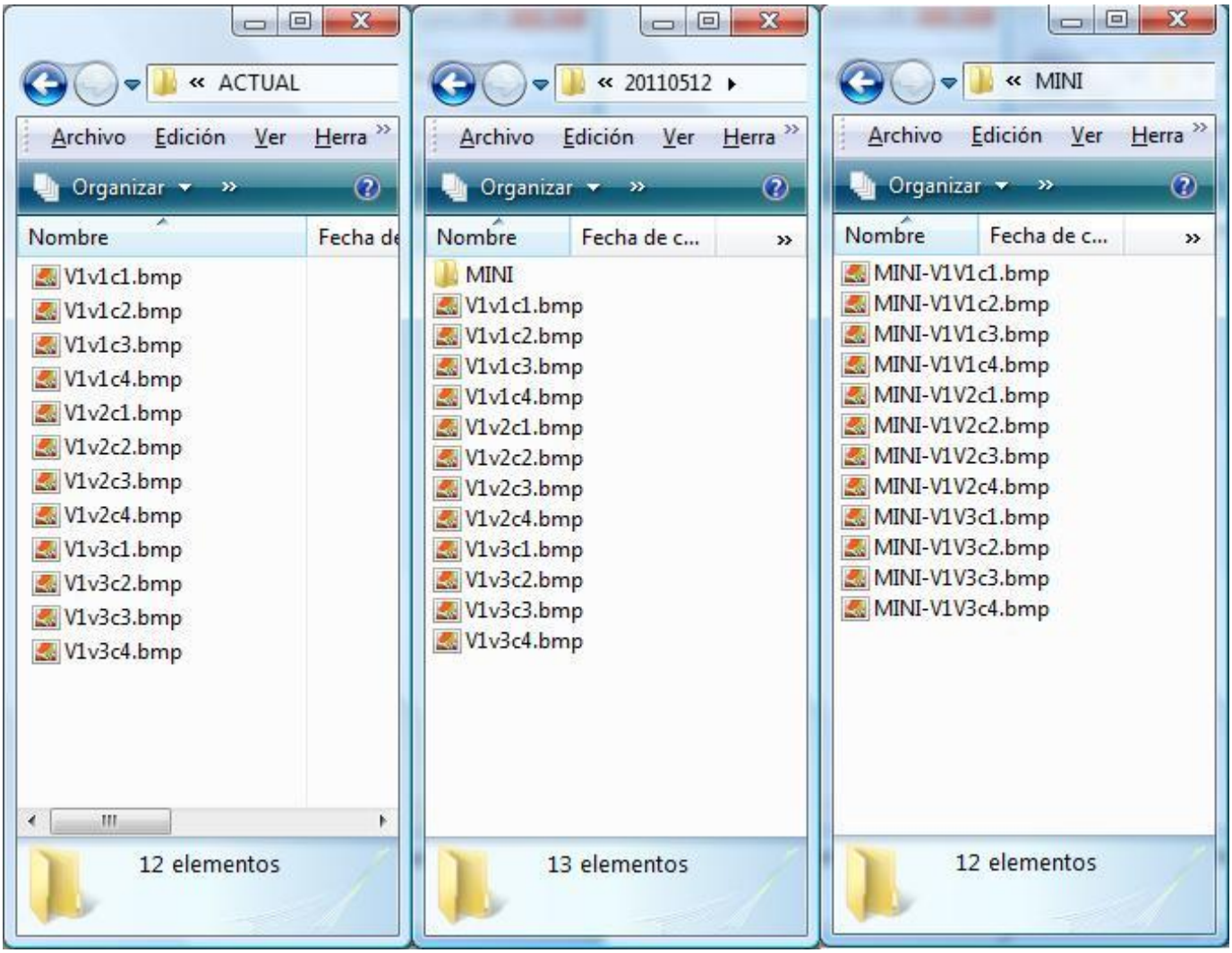


Figura ejemplo de la aplicación 'simulador'.

Cuando se lanza, de forma inmediata comienza a funcionar, sin ser necesario presionar el botón 'Start'. Ésta por cada 'click' que llega por parte del Sensor, genera una fotografía con extensión BMP que guarda en el subdirectorio C:\FOTOS\PURGATORIO\. Como también se puede observar, cada fotografía generada es también mostrada por la aplicación.

En este momento ya todas la aplicaciones están funcionando y sincronizadas.

Y como podemos ver se empiezan a gestionar las fotografías, y la aplicación ShowRun comienza a mostrar las fotografías generadas en vez de la publicidad.

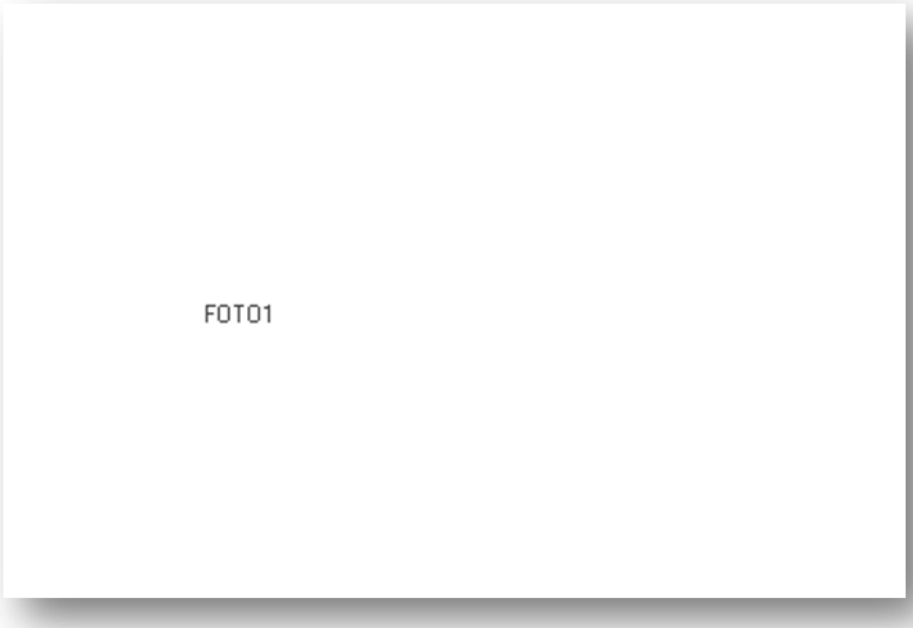


Figuras de los directorios con los que el programa interactua

En la figura anterior, vemos que en la subcarpeta 'C:\FOTOS\ACTUAL\' se encuentran las fotografías del último viaje, que serán eliminadas justo cuando comience el siguiente viaje para guardar las fotografías de ese viaje.

En el subdirectorio del día 'C:\FOTOS\20110512\' se encuentran todas las fotografías del día hechas hasta el momento. Y en el subdirectorio 'C:\FOTOS\20110512\MINI\' se encuentra una copia de todas las fotografías del día de menor tamaño.

A continuación mostraremos una Fotografía original, que la crea la aplicación Simulador y su copia correspondiente en miniatura:



11. CONCLUSIÓN

Una de las conclusiones a las que he llegado es, que desarrollar este software en Delphi, me ha permitido descubrir la potencia y flexibilidad de esta herramienta y el lenguaje de programación.

Que aun teniendo un atractivo e intuitivo entorno de programación y desarrollo, no dejan por ello de ser, las aplicaciones creadas, eficientes y competitivas en el mercado, comparándolas con otras aplicaciones de otras herramientas y lenguajes de programación u otros paradigmas.

Además, las dudas surgidas durante el desarrollo, me han llevado a acudir a foros, manuales, ejemplos y a conocer el trabajo que otros usuarios realizan con Delphi, y en consecuencia conocer también, vía mail, a los propios usuarios, los cuales me han aportado su punto de vista de esta herramienta y lenguaje de programación, siempre y como es lógico, con sus pros y contras, y una visión más amplia que la que el propio proyecto por si sólo me habría dado.

Dejando a un lado la herramienta en sí, he intentado desarrollar este software como un proyecto profesional, es decir, dividiéndolo en fases y marcándome objetivos y fechas de entrega, y he de decir que la experiencia ha sido buena y además he aprendido de ella.

Lo primero que pude experimentar es que planificar y estimar en el mundo de la informática no es fácil, y sobre todo la experiencia previa es fundamental.

Mi experiencia fue que al igual que en el entorno profesional, durante el desarrollo, algunas ‘fechas de entrega’ de algunos objetivos se demoraron, e hicieron que el ‘plazo de entrega final’ también lo hiciese.

También varios objetivos cambiaron de prioridad, ya que durante el desarrollo de la aplicación surgen necesidades, o se observas detalles que no se tuvieron en cuenta en el proceso de análisis. Estos hacen que, en tiempo de desarrollo tengas que volver al análisis, y por tanto esto haga que la prioridad del desarrollo varíe en función del nuevo estudio.

Por eso es tan importante la fase de análisis, y aunque a priori pueda parecer la fase ‘menos productiva’ del proyecto, es todo lo contrario, un buen análisis aplicado a la toma de requisitos, hace que se cumplan en un 99% los plazo y fases determinadas a priori, y esto en la informática no es algo trivial.