UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D/Dª **Petar Mihaylov Petrov**, autor del Trabajo de Fin de Título **Librería de clases de control para cámara térmica industrial, captura/almacenado de termografías y post-procesamiento**,
correspondiente a la titulación **Grado en Ingeniería Informática**,
en colaboración con la empresa/proyecto (indicar en su caso) **Aerolaser System S.L.**

S O L I C I T A

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

[ ] Se ha iniciado o hay intención de iniciarlo (defensa no pública).
[x] No está previsto.

Y para que así conste firma la presente.

Las Palmas de Gran Canaria, a 24 de Noviembre de 2017.

El estudiante MIHAYLOV PETROV PETAR - X2922071J
Firmado digitalmente por MIHAYLOV PETROV PETAR - X2922071J
Fecha: 2017.11.29 09:18:09 Z

Fdo.: Petar Mihaylov Petrov

A rellenar y firmar **obligatoriamente** por el/los tutor/es

En relación a la presente solicitud, se informa:

[X] Positivamente

[ ] Negativamente
(la justificación en caso de informe negativo deberá incluirse en el TFT05)

TRUJILLO PINO AGUSTIN RAFAEL - 42843425Z
Firmado digitalmente por TRUJILLO PINO AGUSTIN RAFAEL - 42843425Z
Fecha: 2017.11.29 10:56:49 Z

Fdo.: _____

SUAREZ GONZALEZ, FRANCISCO (FIRMA)
Firmado digitalmente por SUAREZ GONZALEZ, FRANCISCO (FIRMA)
Fecha: 2017.11.29 16:50:50 Z

DIRECTOR DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería Informática

Trabajo Fin de Grado

Librería de clases de control para cámara térmica industrial,
captura/almacenado de termografías
y post-procesamiento.

Autor:
Petar Mihaylov Petrov

Tutores:
Agustín Rafael Trujillo Pino
Francisco Suárez González

11 de Julio de 2017

Agradecimientos

A mi familia, pues su apoyo y fe en mi me ha llevado hasta donde estoy hoy en día. A mis tutores, Agustín Trujillo y Francisco Suárez, pues sin su consejo este proyecto no hubiera sido posible.

Por último a mis amigos, y todos esos compañeros con los que hemos compartido momentos y afrontado retos durante el transcurso de esta carrera universitaria.

Índice general

Índice de figuras

Índice de cuadros

Resumen

Esta aplicación surge de la necesidad simplificar el uso de una cámara térmica y el desarrollo de programas de control.

Así, el objetivo fundamental de este proyecto es generar una librería de clases de control para una cámara térmica VarioCam HD Head 800 ofreciendo el control total de todas las características que proporcione el hardware con una interfaz simplificada. Como objetivo secundario, se desarrollará un contenedor para el almacenamiento de la información obtenida por la cámara, así como un programa de visionado y post-procesado de dicha información.

Este proyecto demostrará, por medio de la librería desarrollada, su utilidad en el campo técnico. Por otro lado, también resultará útil para ingenieros que necesiten evaluar dicho hardware para futuras aplicaciones.

Abstract

This application arises from the need to simplify the use of a thermal camera and the development of control programs.

Thus, the fundamental objective of this project is to create a library of control classes for a VarioCam HD Head 800 thermal camera, that offers full control of all hardware features with a simplified interface. As a secondary objective, a container will be developed for the storage of the information obtained by the camera, as well as a program for visualizing and post-processing of said information.

This project will demonstrate, through the developed library, its usefulness in the technical field. On the other hand, it will also be useful for engineers who need to evaluate the hardware for future applications.

1

Introducción y Objetivos

1.1. Línea de alta tensión

La red de transporte de energía eléctrica es la parte del sistema de suministro eléctrico constituida por los elementos necesarios para llevar hasta los puntos de consumo y a través de grandes distancias la energía eléctrica generada en las centrales eléctricas.

Parte de la red de transporte de energía eléctrica son las llamadas líneas de transporte.

Una línea de transporte de energía eléctrica o línea de alta tensión es básicamente el medio físico mediante el cual se realiza la transmisión de la energía eléctrica a grandes distancias. Está constituida tanto por el elemento conductor, usualmente cables de acero, cobre o aluminio, como por sus elementos de soporte, las torres de alta tensión.

Se hace por lo tanto imprescindible para mantener una revisión y un mantenimiento continúo de todas las líneas de alta tensión, con el fin de incrementar la seguridad y reducir pérdidas por una transmisión ineficiente.

1.2. Inspección de línea

Este tipo de inspección consiste en recorrer toda la línea eléctrica identificando deficiencias en los diferentes elementos de la misma. Para ello se utilizan cámaras ópticas y cámaras térmicas. Estas segundas se utilizan para identificar aquellos puntos en los que se produzcan variaciones significativas de la temperatura ya que éstos indicarán problemas y pérdidas en la red de distribución.

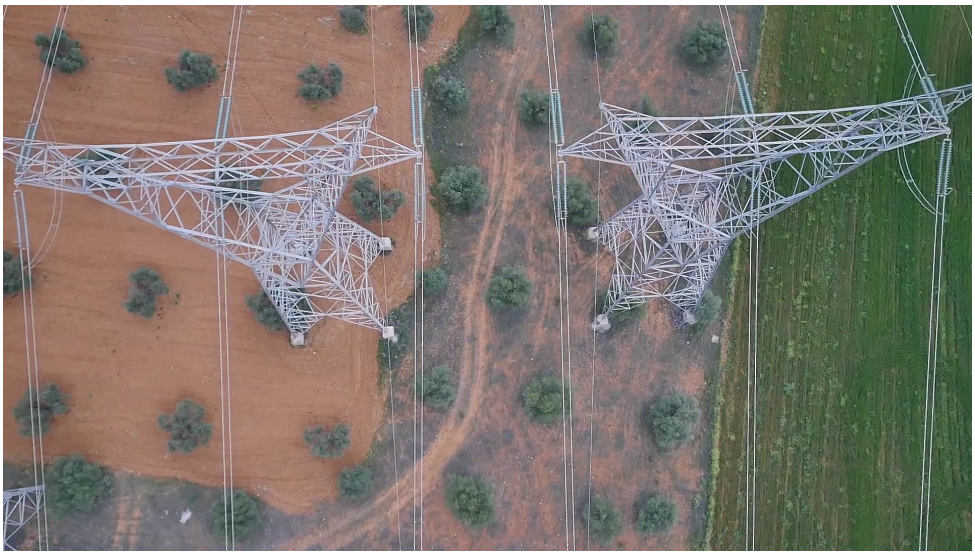


Figura 1.1: Ejemplo de inspección aérea.

1.2.1. Aplicación de la termografía

A nivel técnico, la aplicación de la termografía nos va a permitir visualizar los patrones de temperatura de los sistemas e instalaciones eléctricas. En este sentido, hay que tener en cuenta que una causa de fallo en los sistemas eléctricos es un exceso de temperatura provocado por diferentes motivos:

- Incremento de resistencia en puntos de conexión. De acuerdo a la Ley de Joule:P=I^2xR
- Corrientes de fuga en sistemas aisladores. La reducción de la resistencia de aislamiento debido a suciedad o contaminantes puede dar lugar a la aparición de corrientes de fuga y arcos que dan lugar al calentamiento de los cables y por lo tanto a su deterioro.

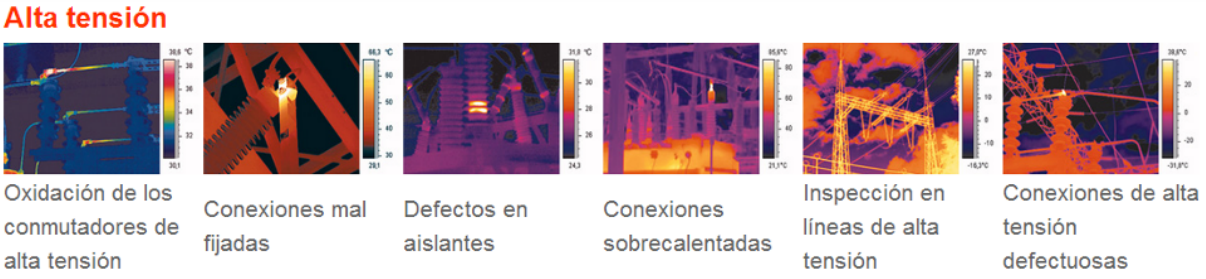


Figura 1.2: Ejemplo de anomalías en inspecciones de alta tensión

Con una cámara termográfica el técnico de inspección es capaz de examinar los componentes que componen el sistema de distribución eléctrica en busca de patrones de calentamiento, lo cual le va a permitir detectar y elaborar un informe con una posible incidencia antes de que de lugar a un fallo o interrupción en la línea.

1.3. Motivación

La empresa Aerolaser System S.L. entre otros servicios, ofrece inspecciones de líneas eléctricas que garantizan la verificación del 100 % de las instalaciones, a su vez, ha proporcionado una cámara térmica modelo VarioCam HD Head 800[13] y su respectivo SDK con su documentación con el fin de desarrollar una librería de clases de control de la cámara térmica para su fácil integración en diversos programas y proyectos de la empresa.

1.4. Objetivos

A raíz de la motivación, el objetivo principal de este proyecto será la familiarización con el respectivo hardware y su kit de desarrollo, y posterior elaboración de una librería de clases de control. Las características con las que contará dicha librería son:

- Parametrización de todas las funcionalidades que ofrece el hardware.
- Implementación de un programa interfaz de usuario de prueba para la librería de control.
- Implementación de una estructura de datos para el almacenado de datos proporcionados por la cámara en disco.

Finalmente como objetivo secundario se implementará un gestor/visor de la estructura de datos mencionada anteriormente para el post-procesado de los ficheros.

1.5. Recursos Utilizados

- **Visual Studio 2015 Professional:** Visual Studio ofrece un conjunto de herramientas necesarias para crear software. De forma predeterminada Visual Studio proporciona compatibilidad con C#, C y C++, JavaScript y Visual Basic. Entre las diferentes versiones que incorpora Visual Studio se empleó Visual Studio Professional 2015 para elaborar toda la lógica necesaria en C#.[12]
- **InstallShield 2015 Limited Edition:** InstallShield 2015 LE[11] es una herramienta para crear archivos de instalación con el fin de distribuirlos a usuarios para dar la posibilidad de instalar una aplicación o un componente de escritorio sin estar conectados a una red. InstallShield Limited Edition es gratuita para usuarios de Visual Studio Professional y Enterprise. Sustituye a la tecnología de Windows Installer, que ya no es compatible con Visual Studio[8].
- **WhiteStarUML:** Herramienta de modelado UML desarrollada como continuación al proyecto StarUML que está abandonado. Se usó esta herramienta frente a las muchas que existen ya que es gratuita y estamos familiarizados con StarUML ya que se usa en varias asignaturas que se imparten en la Escuela de Ingeniería Informática.[9]

- **LucidChart:** Herramienta online de creación de diagramas de flujo, se usó para la generación de los diagramas de flujo de esta memoria.[3]
- **IRBSDK v4.0:** IRBSDK es el kit de desarrollo proporcionado junto a la cámara Infratec VarioCam HD Head 800, incluye las licencias para el uso del hardware, wrappers para las funciones de control y proyectos de ejemplo para facilitar el desarrollo.
- **ShareLaTex:** Usado para la elaboración de este documento. Es un editor LaTex online con opciones de colaboración a tiempo real y compilación de proyectos a PDF[6].
- **Hardware:** Nuestro proyecto fue desarrollado en un sistema informático con las siguientes características. Un microprocesador Intel i7 3770k (3.50Ghz - 3.90Ghz, 4 núcleos, 8 hilos de ejecución[2]), 16 Gb RAM (666 MHz 9-9-9-24) y una tarjeta gráfica Nvidia 650GTX (384 núcleos [5]).
- **Cámara Térmica:** Infratec VarioCam HD head 800.[13]
- **Libro:** C# In a Nutshell: A Desktop Quick Reference.[10] Este libro se usó como fuente principal para adquirir los conocimientos necesarios para realizar este proyecto respecto al lenguaje C# y el .NET Framework, en especial, el capítulo 11 sobre la serialización de objetos.

1.6. Justificación de las competencias

La elaboración de este proyecto ha llevado a cubrir una serie de competencias profesionales. A continuación se presentan las competencias cubiertas en este Trabajo de Fin de Grado (TFG):

- **IC04. Capacidad de diseñar e implementar software de sistema y de comunicaciones.**
- **IC07. Capacidad para analizar, evaluar, seleccionar y configurar plataformas hardware para el desarrollo y ejecución de aplicaciones y servicios informáticos.**

  La propia elaboración de este proyecto constituye el cumplimiento de estas competencias.
- **TFG01. Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.**

  Esta competencia se cubrirá una vez se realice la presentación de este Trabajo de Fin de Grado ante los miembros del tribunal establecidos.

1.7. Aportaciones

El producto generado en este TFG aporta versatilidad a la cámara, permitiendo su fácil evaluación para proyectos ya que la librería de control incorpora también interfaz de usuario de prueba, lo que permite a los ingenieros evaluar la cámara y sus características para futuras aplicaciones directamente desde la librería de control, por otro lado, permite desarrollar nuevas aplicaciones de manera más ágil ya que abstrae el control del hardware exponiendo una interfaz simplificada a su vez también permite la fácil integración en proyectos existentes.

A parte, aprovecha las optimizaciones de los sistemas de 64 bits frente a los de 32 bits lo que permite tener un software adecuado para los equipos modernos.

Tanto la librería de control como el visor de termografías pueden servir como base para la adquisición de conceptos y metodologías aplicadas al control de cámaras industriales y el post-procesado de la información.

2

Planificación del proyecto

En este capítulo se expondrá la planificación del proyecto abordando la metodología escogida.

2.1. Planificación del proyecto

Para la elaboración de esta proyecto fue lícito realizar una planificación a la que acogernos de manera que tengamos pautas a seguir y no se implementaran cosas que no fueran estrictamente necesarias.

Para este proyecto se siguió un proceso de desarrollo guiado por el modelo iterativo incremental como se muestra en la figura a continuación

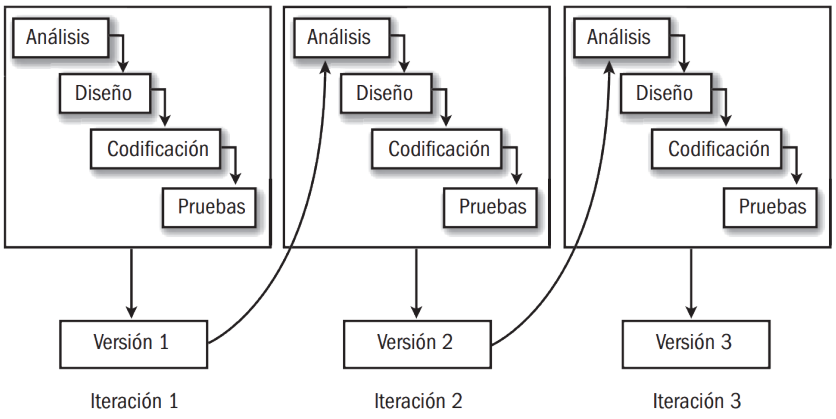


Figura 2.1: Modelo iterativo

La elaboración del proyecto supuso una etapa previa de análisis en la que se realizó un estudio del hardware, SDK y la documentación, a continuación se especificó con la empresa Aerolaser System S.L. un diseño y posteriormente se procedió a la codificación y prueba, en cada iteración, después de cada prueba se analizaban los reportes aportados por los ingenieros que evaluaron el proyecto en vuelo y se repitió el proceso.

Este modelo permitió que el proyecto se fuera generando de forma gradual, recolectando feedback del usuario, que permitió ajustar lo que elaborábamos en los prototipos a nuestro proyecto final.

2.2. Desarrollo evolutivo

En este modelo de desarrollo se entrelazan las actividades de especificación, desarrollo y validación. Inicialmente, se desarrolla rápidamente un sistema inicial a partir de una especificación abstracta. El sistema se va refinando con la información que van suministrando los usuarios hasta que se obtiene un sistema final que satisfaga todas las necesidades previstas.

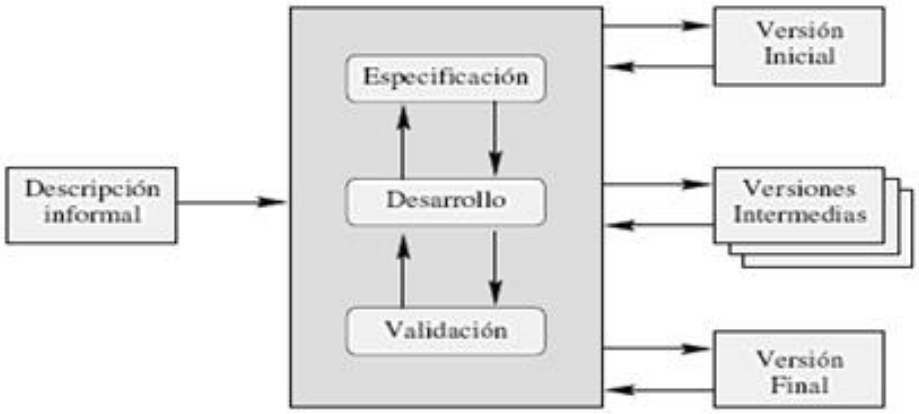


Figura 2.2: Desarrollo evolutivo

Existen dos tipos de procesos de desarrollo evolutivos, el desarrollo exploratorio y el desarrollo desechable, para la elaboración de este proyecto se llevo a cabo un desarrollo evolutivo exploratorio, su objetivo es trabajar con el cliente para identificar y construir el sistema final a partir de una especificación informal. El sistema evoluciona agregando nuevos atributos propuestos por el usuario y el resultado del proceso es el sistema final.

3

Análisis

3.1. Requisitos del Sistema

En esta sección se expondrán los diferentes requisitos del sistema desarrollado. Inicialmente se estudiará el demo proporcionado por el Software Development Kit (SDK) que servirá de base para el desarrollo del proyecto. Finalmente se comentarán las limitaciones del proyecto.

3.1.1. Análisis

El SDK proporciona varios proyectos de ejemplo como punto de inicio, dichos proyectos están disponibles para Lazarus/Pascal, Python, C# y C++. Como este proyecto será desarrollado en C# utilizaremos el demo de C# como base.

Las diferentes versiones (en diferentes lenguajes y entornos) no tienen diferencia apreciable entre si, ya que los demos simplemente son wrappers de ejemplo para las funciones exportadas proporcionadas por la Dynamic-link Library (DLL) del SDK.

En la siguiente figura se aprecia la interfaz de usuario del demo:

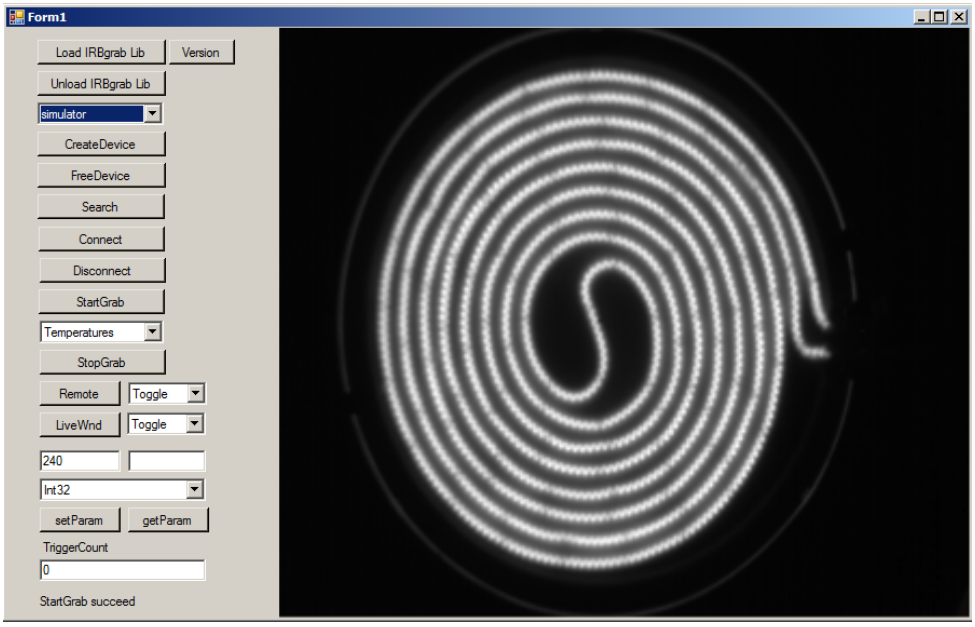


Figura 3.1: Demo

Estudiando esta Graphic User Interface (GUI) podemos apreciar que el paso inicial es cargar la DLL de funciones IRBgrab, esto activa un campo de selección en el que elegir que DLL cargar, esta DLL hace referencia al dispositivo que queremos conectar, en la figura anterior, se usó la DLL simulador ya que no requiere tener la cámara conectada.

Una vez seleccionado el dispositivo, vemos un botón “CreateDevice” esto implica crear el Callback hacia las funciones de IRBgrab.dll

Por último para realizar la conexión al hardware con el software hay que hacer uso del botón “Search” para que localice el hardware conectado y posteriormente “Connect”. Para empezar a obtener imagen del hardware hay que ejecutar “StartGrab”

Además, vemos que existen otros botones como “setParam” y “getParam” estos son para configurar u obtener información sobre configuración de diversas funciones como por ejemplo zoom, estos parámetros son un número de tres cifras y la lista de parámetros está en la documentación del SDK de la cámara.

Estudiando el funcionamiento del demo, podemos elaborar el siguiente diagrama de flujo simplificado:

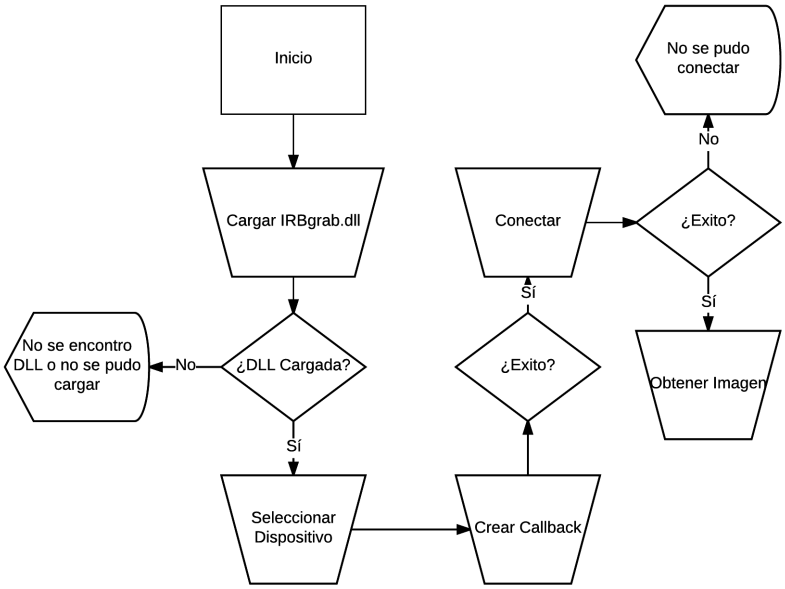


Figura 3.2: Diagrama de flujo demo

3.1.2. Requisitos

El objetivo principal de este proyecto es abstraer el control de la cámara por lo que implementó de acuerdo al siguiente diagrama de flujo:

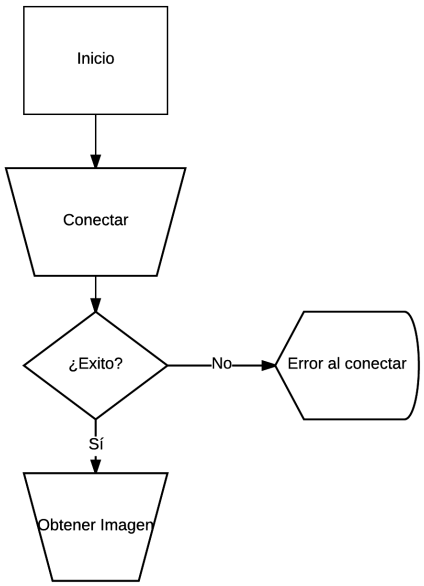


Figura 3.3: Diagrama de flujo propuesto

Además la empresa especificó las funciones que debe contener la librería de clases de control.

- Conectar / Desconectar.
- Numero de serie / Estado de la cámara.
- Enfoque.
- Empezar / Parar captura.
- Control de triggers.
- Guardar termografía.
- Control de frame rate.

3.2. Limitaciones del proyecto

Antes de comenzar con las labores de diseño hubo que tener en cuenta una serie de limitaciones.

- Limitación de hardware: La empresa dispone de una cámara, por lo que esta versión solo podrá controlar una cámara simultáneamente, para implementar el control de varias cámaras es necesario disponer del hardware.
- Limitación del software: El SDK no proporcionaba función de guardado de termografías ya que era un formato propietario, por lo que para cumplir con los requisitos previos hubo que desarrollar dos subproyectos, AEThermoSerializer y AEThermoViewer.
- Documentación pobre y en alemán, hemos hecho uso de traductor y usado conjuntamente la documentación para la anterior versión del SDK que estaba en estado incompleto, además hubo poca colaboración por parte del fabricante de la cámara.

4

Desarrollo del proyecto

4.1. Desarrollo

Disponemos de una cámara térmica industrial modelo VarioCam HD Head 800, con su correspondiente SDK y como anteriormente mencionamos, nuestro objetivo es desarrollar una librería de clases de control de dicha cámara, así como un programa de visionado y editado de termografías.
Para cumplir este objetivo fue apropiado desarrollar tres proyectos, **AEVarioCamControlLib**, **AEThermoSerializer**, **AEThermoViewer**.
A continuación se resumirá en que consiste cada proyecto.

- **AEVarioCamControLib:** Librería de clases de control que hace uso de las funciones que proporciona el SDK de la cámara en cuestión, así como del wrapper en C#, su funcion es proporcionar un control simplificado para la integración en otros proyectos, sin tener que recurrir al estudio de la documentación del fabricante.

- **AEThermoSerializer:** InfraTec ha desarrollado un formato (.irb) propio para el almacenado de la información captada por la cámara en disco, este formato es propietario y cerrado, no se ha adquirido con el SDK además de que la versión a usar no incluye funciones para el almacenado en dicho formato. Por otro lado, según las pruebas realizadas, este formato es bastante pesado en cuanto en tamaño en disco.

  Teniendo en cuenta lo anteriormente mencionado, se ha desarrollado AEThermoSerializer, una clase con función de serialización de un objeto "Thermograph"que contiene un vector de temperaturas escaladas (Para reducir el tamaño del fichero), entre otros campos como temperaturas máxima, mínima, escala, etc.

- **AEThermoViewer:** Para visionar el formato propietario anteriormente mencionado (.irb) se puede hacer uso de IRBISView que viene junto con el SDK y no requiere licencia adicional, este, solo permite ver las termografías, y exportarlas a imagen, mientras que, para editarlas y realizar otro tipo de acciones, como por ejemplo consultas de temperaturas, se requiere usar IRBIS 3 Infratec, este programa no viene con el SDK y requiere adquirir licencia adicional.

  Puesto que como anteriormente comentamos, el SDK no ofrece función de guardado, desarrollamos un serializador (AEThermoSerializer). Una termografía es un vector de temperaturas, esto es un formato máquina, pero necesitamos alguna forma de poder visualizar dichas temperaturas, por ello

se desarrollo AEThermoViewer, un programa para el visionado, editado del gradiente y guardado en disco en formato imagen para objetos creados por nuestro serializador.

Para visualizar un vector de temperaturas, creamos un gradiente de colores con límite inferior la temperatura mínima y límite superior la temperatura máxima y posteriormente las temperaturas capturadas por la cámara se escalan en dicho gradiente y se colorea un mapa de bits.

4.1.1. Entorno de desarrollo

Como hemos mencionado, nuestro entorno de desarrollo es Visual Studio 2015 a continuación se explicaran ciertos conceptos de este entorno.

4.1.1.1. Solución y Proyecto

Visual Studio dispone de dos contenedores con finalidad de administrar eficazmente los elementos necesarios para el desarrollo, como referencias, conexiones de datos, carpetas y archivos. Estos contenedores se denominan soluciones y proyectos.

Soluciones

Las soluciones contienen los elementos necesarios para crear la aplicación. Una solución puede incluir uno o más proyectos y un proyecto no puede existir por fuera de una solución, además de proyectos las soluciones pueden incluir archivos y metadatos que ayuden a definir la solución en conjunto.

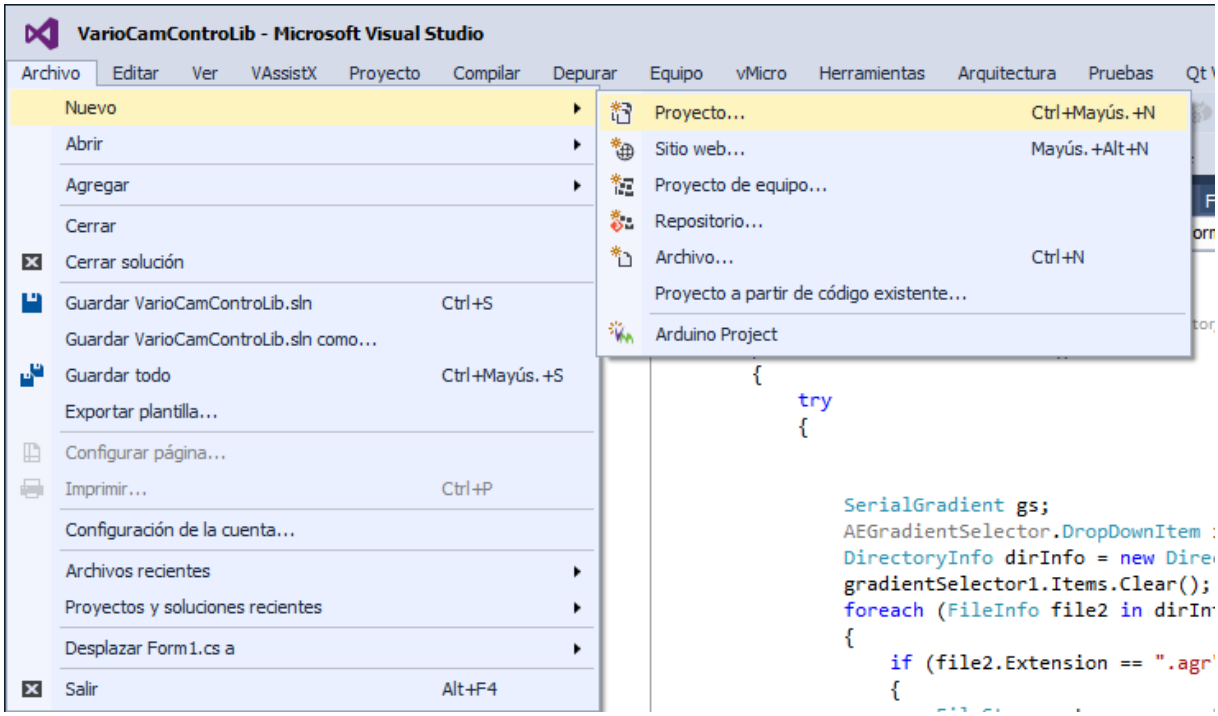


Figura 4.1: Creación de nuevo proyecto.

Ya que un proyecto no puede existir fuera de una solución, al crear un nuevo proyecto se nos abre un diálogo para elegir las opciones del proyecto, como la plataforma y lenguaje, aparte nos aparece si deseamos crear una nueva solución, o agregar el proyecto a la solución actual.

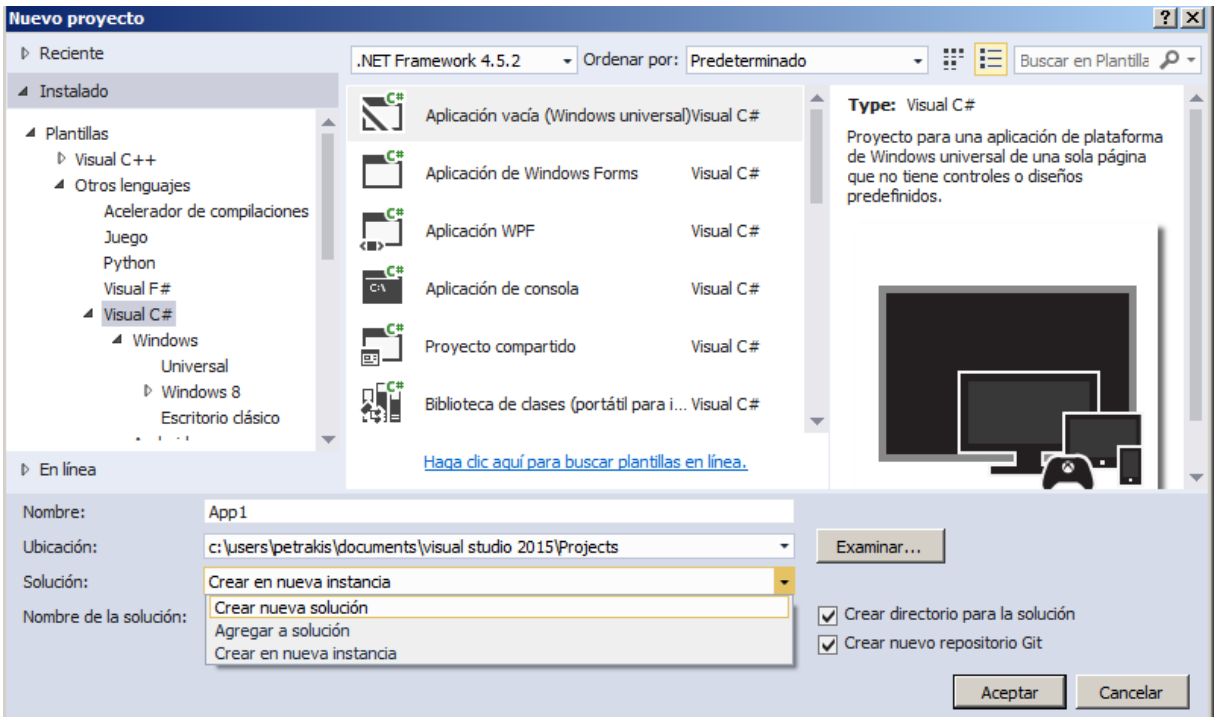


Figura 4.2: Diálogo de creación de nuevo proyecto.

En resumen, Visual Studio almacena la definición para una solución en dos archivos: .sln y .suo. El archivo de definición de solución (.sln) almacena los metadatos que definen la solución como:

- Los proyectos asociados a la solución.
- Los elementos que no están asociados a un proyecto determinado.
- Las configuraciones de compilación que determinan las configuraciones de proyecto a aplicar a cada tipo de compilación.

El archivo .suo se utiliza para almacenar los metadatos que se encargan de personalizar el IDE cuando la solución esté activa, como por ejemplo la posición de las ventanas, o estilo de la interfaz gráfica.

Proyectos

Los proyectos en una solución se utilizan para administrar de forma lógica, compilar, y depurar los elementos que forman parte de una aplicación. El resultado de un proyecto puede ser un programa ejecutable (.exe), un archivo de biblioteca (.dll o .lib) o un módulo, según como este definido.
Visual Studio proporciona varias plantillas predefinidas de proyecto como pudimos ver en 4.2.
Para concluir, el siguiente diagrama muestra la relación entre los proyectos y soluciones, y los elementos que estos contienen de forma lógica.

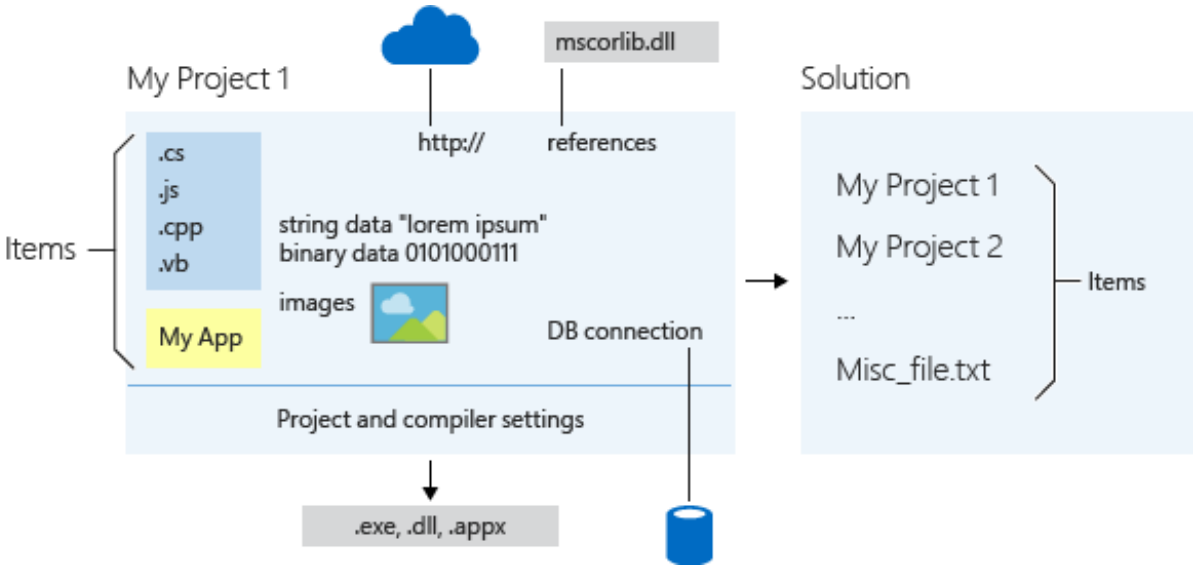


Figura 4.3: Relación entre soluciones y proyectos.

4.1.1.2. Explorador de soluciones

El Explorador de Soluciones es un área del entorno de desarrollo que contiene la solución sobre la que estamos trabajando y nos ayuda a administrar los archivos de proyecto, dichos archivos se muestran de manera jerárquica, al estilo del Explorador de Windows. Por defecto, el Explorador de Soluciones se encuentra en el lado derecho del entorno. Si no estuviera visible, podemos abrir el Explorador de Soluciones desde el menú Ver y a continuación, Explorador de Soluciones ó con la combinación de teclas Ctrl+Alt+L.

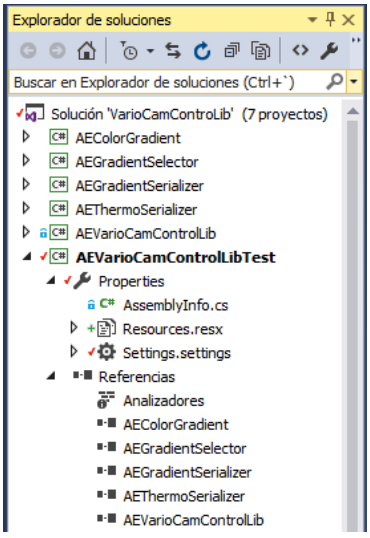


Figura 4.4: Explorador de Soluciones.

Desde este podemos administrar nuestros proyectos, como por ejemplo crear nuevos o eliminar existentes, renombrarlos, agregar referencias, entre otras opciones.

4.1.1.3. Referencias

Una referencia es un componente a agregar a nuestro proyecto para poder usar dicho componente en nuestra aplicación, Visual Studio proporciona cinco opciones en el cuadro de diálogo Agregar Referencia:

- **.NET** Enumera todos los componentes de .NET Framework disponibles para hacer referencias.
- **COM** Enumera todos los componentes de COM disponibles para hacer referencias.
- **Proyectos** Enumera todos los componentes utilizables creados en proyectos locales en la solución actual.
- **Examinar** Permite buscar un componente en el sistema de archivos.
- **Reciente** contiene una lista de componentes agregados recientemente a proyectos.

En resumen, tomando como ejemplo la librería de serialización creada, tenemos dos opciones para utilizarla en aplicaciones, una de ellas es compilar el proyecto y generar el archivo .DLL y agregar la referencia al proyecto, y la otra opción es incluir el proyecto (que puede estar en una solución aparte) desde el Explorador de Soluciones a nuestra solución actual y posteriormente agregar la referencia mediante la opción **Examinar** mencionada anteriormente.

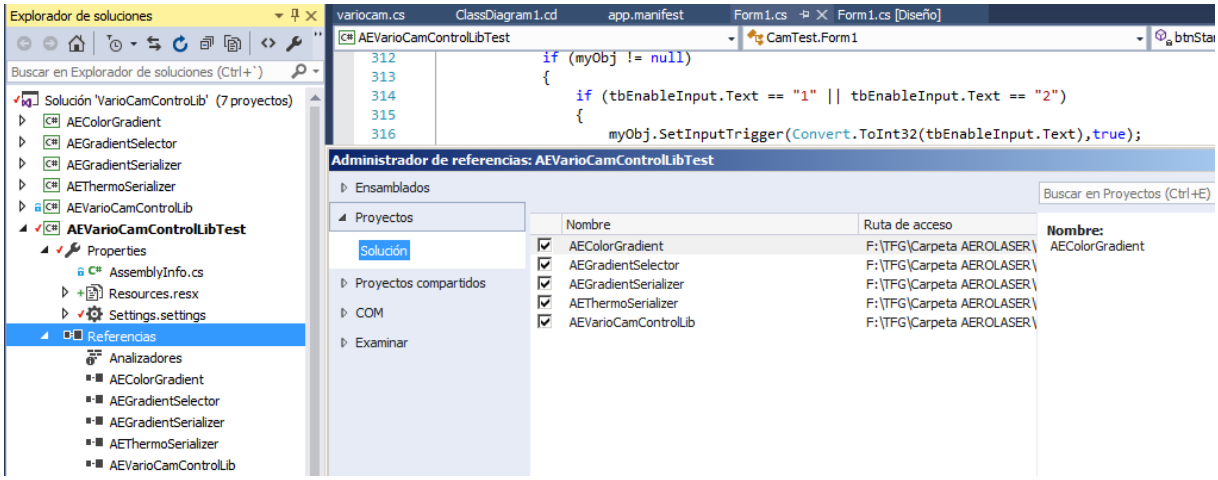


Figura 4.5: Referencias.

Durante la realización de este trabajo de fin de grado, se ha optado por la segunda opción ya que es más versátil, nos muestra de manera jerárquica todos los proyectos que involucran nuestra aplicación y nos permite realizar modificaciones a los componentes sin tener que abrir otra instancia de Visual Studio o solución.
Las referencias se resuelven en tiempo de compilación, es decir primero se compilan las referencias y luego se enlazan a la aplicación que las utilice.

4.1.1.4. Directorio de proyecto

A continuación se mostrará un ejemplo de un directorio de proyecto creado con Visual Studio para reforzar de manera práctica los conceptos explicados anteriormente.

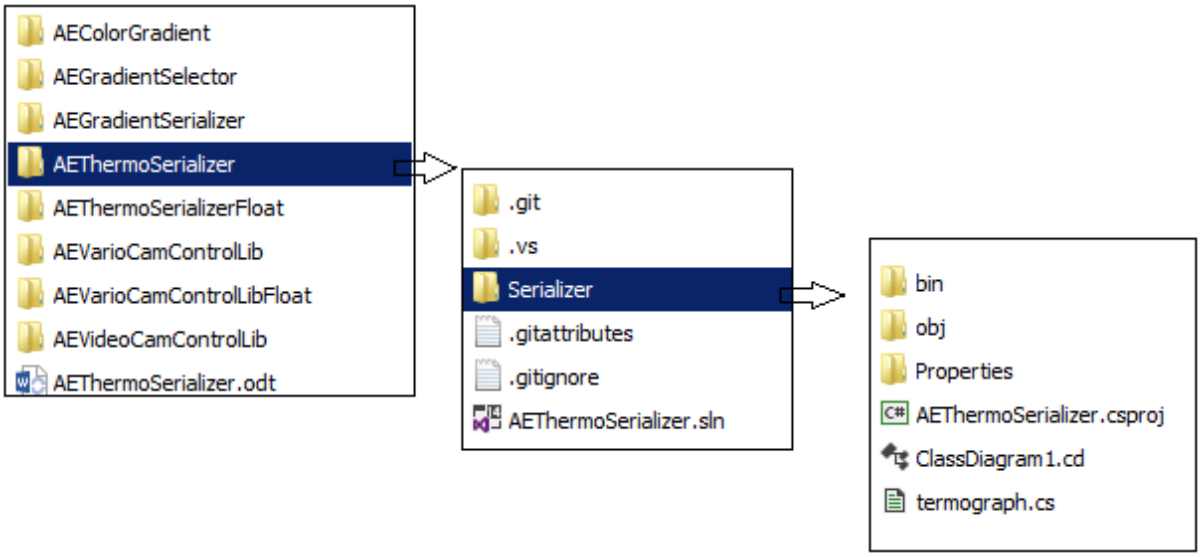


Figura 4.6: Jerarquía de carpetas.

Podemos apreciar que dentro de la carpeta raíz se encuentra el archivo de solución .sln, y en subcarpetas se encuentran los proyectos asociados a dicha solución, en este caso solo tenemos un proyecto alojado en la carpeta Serializer, dentro de dicha carpeta podemos ver el archivo de proyecto (.csproj) entre otros elementos como por ejemplo los archivos .cs que son los archivos fuente de C#.

4.1.2. AEThermoSerializer

4.1.2.1. Antecedentes

InfraTec ha desarrollado un formato (.irb) propio para el almacenado de la información captada por la cámara en disco este formato es propietario y cerrado, además de que la versión de SDK a usar no lo implementa. Por otro lado, según las pruebas realizadas, este formato es bastante pesado.

Teniendo en cuenta lo anteriormente mencionado, se ha desarrollado AEThermoSerializer, una clase con función de serialización de un objeto “Thermograph” que contiene un vector de temperaturas escaladas (Para reducir el tamaño de fichero) entre otros campos como temperatura máxima, escala, etc.

4.1.2.2. Características

- Tamaño de fichero óptimo (1.5 MB).
- Escalable, se puede ampliar su definición en un futuro.
- Permite escalado de datos para ganar precisión sin aumentar el tamaño de fichero.
- Permite guardar información de disparo trigger externo.

- Posee campo índice que puede ser usado para marcar diferentes objetos serializados.

4.1.2.3. Clase Thermograph Lista de funciones

Termograph()
Constructor por omisión necesario para la inicialización del objeto a la hora de de-serializar un archivo.

Thermograph(ushort index,short[] data, ushort width, ushort height, ushort minv, ushort maxv, ushort avg,ushort scale, bool triggered)
Constructor inicializador de un objeto a serializar.

Index
Campo libre útil por ejemplo para establecer un índice de termografías.

Data
Vector contenedor de la matriz de temperaturas de la cámara.

Width
Dimension horizontal en pixeles del objeto a serializar.

Height
Dimension vertical en pixeles del objeto a serializar.

Maxv
Valor de temperatura máxima registrada.

Minv
Valor de temperatura mínima registrada.

Avg
Temperatura media.

Scale
Valor de escalado de los datos.

Triggered
Información de si el objeto se ha obtenido por un trigger externo o no.

getIndex()
Devuelve valor de la variable libre index.

getHeight()
Devuelve la dimensión vertical en pixeles.

getWidth()
Devuelve la dimensión vertical en pixeles.

getMin()
Devuelve el valor de temperatura mínimo.

getMax()
Devuelve el valor de temperatura máximo.

getAvg()
Devuelve la temperatura media.

getDataArray()
Devuelve el array contenedor de la matriz de temperaturas.

isTriggered()
Devuelve si el termograma se ha obtenido por un trigger externo o no.

4.1.2.4. Uso

Para utilizar esta librería de serialización de objetos se debe agregar como referencia AEThermoSerializer.dll tanto a los proyectos que serialicen objetos como a los proyectos que de-serialicen objetos.

Ejemplos:

Serialización de objetos:

```
Thermograph tgObj = new Thermograph(0,inVec, (ushort)w,
    (ushort)h, (ushort)minvs, (ushort)maxv, (ushort)avg,
    (ushort)scale, trig);

FileStream stream = new FileStream("Images/" +
    DateTime.Now.ToString("yyyy-MM-dd HH-mm-ss.ffffff")
    + ".atr", FileMode.Create);

BinaryFormatter myFormatter = new BinaryFormatter();
myFormatter.Serialize(stream, tgObj);
stream.Close();
```

De-serialización de objetos:

```
FileStream stream = new FileStream(selectedFilePath,
    FileMode.Open);

BinaryFormatter myFormatter = new BinaryFormatter();

Thermograph obj = (Thermograph)(myFormatter.Deserialize(stream));
```

4.1.3. AEVarioCamControlLib

VarioCAM HD Head 800 es una cámara de alta calidad, y una herramienta de medición cuidadosamente calibrada, a continuación se expondrá la librería de control desarrollada.

Figura 4.7: Cámara térmica VarioCam HD Head 800

4.1.3.1. Antecedentes

VarioCAM HD Head es un sistema de termografía diseñado para la gama espectral infrarroja de ondas largas (LWIR) de (7.5 ... 14) µm. La lente reproduce la escena del objeto en una matriz de microbolómetro con (1.024 x 768) y / o (640 x 480) píxeles. La señal eléctrica de la matriz de detectores es procesada adicionalmente por la electrónica interna. En esto, la electrónica comprende todas las funciones requeridas para el funcionamiento de la cámara, como el accionamiento de la matriz de microbolómetros, la conversión A / D, la corrección de desplazamiento y ganancia, la corrección de píxeles y el accionamiento de las interfaces que difieren según el equipo.

El manejo se realiza utilizando la interfaz digital Ethernet (GigE Vision) , se requieren paquetes de software específicos para usar estas interfaces.

Características

- Rango de medida: -40 a 1.200 °C
- Resolución: 0,02 °C (a 30 °C)
- Precisión: ± 1 °C o ± 1
- Detector: 1.024 x 768 pixeles (UFPA microbolómeter)
- Tamaño pixel: 17 μm
- Rango espectral: 7,5 a 14 μm
- Campo visual: 16,5° (H) x 12,4° (V)

- Frecuencia refresco: 30 Hz
- Peso: 1,15 kg
- Rango dinámico: 16 bit

Documentación Oficial[7]

4.1.3.2. Librería dinámica

En este apartado comentaremos la estructura de la librería dinámica que ofrece el SDK y cubriremos el uso de las funciones que exporta. Para ver las funciones empaquetadas en una DLL haremos uso de DLL Export Viewer[4], un programa sencillo y gratuito.

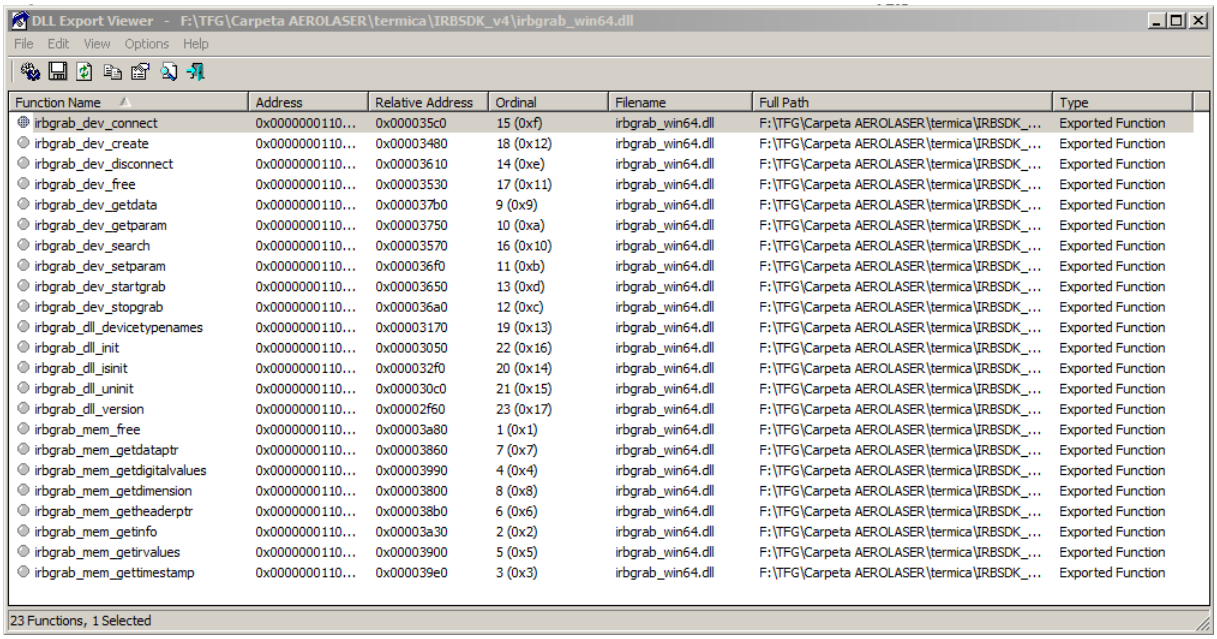


Figura 4.8: Funciones empaquetadas en la librería dinámica.

4.1.3.3. Convención de llamada

Las convenciones de llamada determinan el orden en el que se insertan en la pila los argumentos de una función, si la función llamadora o la llamada quita los argumentos de la pila al finalizar la llamada, y la convención para la creación de nombres representativos que utiliza el compilador para identificar funciones individuales.
A continuación explicaremos las convenciones existentes según la documentación de desarrolladores de microsoft.[1]

- **__cdecl:** Esta es la convención de llamada predeterminada en el entorno de visual studio. En los procesadores de x86, todos los argumentos de función se pasan en la pila de derecha a izquierda. En las arquitecturas ARM y x64, algunos argumentos se pasan por registro y el resto se pasa en la pila de derecha a izquierda. La rutina de llamada hace aparecer el argumento desde la pila.

- **__fastcall:** Algunos argumentos de una función con esta convención de llamada se pasan en registros y el resto se insertan en la pila de derecha a izquierda. La rutina invocada optiene estos argumentos desde la pila antes de su retorno. Esta convención de llamada solo está disponible en los compiladores destinados a x86 y la pasan por alto los compiladores destinados a otras arquitecturas. Normalmente, __fastcall disminuye el tiempo de ejecución.

- **__stdcall:** Los argumentos de una función con esta convención se insertan en la pila de derecha a izquierda y la función llamada obtiene estos argumentos de la pila antes de su retorno. Todas las funciones __stdcall deben tener prototipos. Esta convención de llamada solo está disponible en los compiladores destinados a x86 y la pasan por alto los compiladores destinados a otras arquitecturas, como ARM.

- **__vectorcall:** Los argumentos enteros de una función con esta convención se pasan por valor, utilizando hasta dos (en x86) o cuatro (en x64) registros Integer, el resto se pasa en la pila de derecha a izquierda. La función llamada limpia la pila antes de devolver un valor. Los valores devueltos del vector y del punto flotante se devuelven en el registro XMM0

Según la documentación del fabricante, la convención de llamada que debemos usar es __stdcall, a continuación mostraremos una figura de ejemplo de como se usa dicha convención para importar las funciones de la DLL.

```
4 referencias | Petar Mihaylov Petrov, Hace 82 días | 1 autor, 2 cambios
public class TIRBgrabLib
{
    //For x64:
    const string DLLExt = "_win64";
    //For x86:
    //const string DLLExt = "_win32";

    [DllImport(@"irbgrab" + DLLExt + ".dll", CallingConvention = CallingConvention.StdCall)]
    1 referencia | Petar Mihaylov Petrov, Hace 83 días | 1 autor, 1 cambio
    private static extern uint irbgrab_dll_version(byte[] aDLLversion, int aMaxLen);
    [DllImport(@"irbgrab" + DLLExt + ".dll", CallingConvention = CallingConvention.StdCall)]
    1 referencia | Petar Mihaylov Petrov, Hace 83 días | 1 autor, 1 cambio
    private static extern uint irbgrab_dll_init();
    [DllImport(@"irbgrab" + DLLExt + ".dll", CallingConvention = CallingConvention.StdCall)]
    1 referencia | Petar Mihaylov Petrov, Hace 83 días | 1 autor, 1 cambio
    private static extern uint irbgrab_dll_uninit();
    [DllImport(@"irbgrab" + DLLExt + ".dll", CallingConvention = CallingConvention.StdCall)]
    1 referencia | Petar Mihaylov Petrov, Hace 83 días | 1 autor, 1 cambio
    private static extern uint irbgrab_dll_isinit(ref int aInitCount);
    [DllImport(@"irbgrab" + DLLExt + ".dll", CallingConvention = CallingConvention.StdCall)]
```

Figura 4.9: Importación de funciones con la convención de llamada.

Para concluir, hay que mencionar que las funciones de nuestra librería de control están implementadas haciendo uso de una o más funciones de las mostradas anteriormente que proporciona la librería dinámica.

4.1.3.4. Lista de Funciones

Connect()
Llamada asíncrona implementada con background worker a los métodos de conexión de la cámara.

Genera un evento evConnected cuando la conexión se ha completado.

Disconnect()
Llamada asíncrona implementada con background worker a los métodos de desconexión de la cámara.

Lanza un evento evDisconnected cuando la desconexión se ha completado.

SerialNumber(int idx)
Devuelve string con información sobre el hardware.

Parámetro de entrada: índice de la cámara.

AutoFocus()
Envía comando de autoenfoque a la cámara.

Devuelve mensaje de respuesta de la cámara.

SetFocus(Single m)
Ajusta el enfoque según el parámetro de entrada en metros.

Devuelve mensaje de respuesta de la cámara.

GetFocus(ref float focus)
Obtiene el valor de enfoque en metros por referencia.

Devuelve mensaje de respuesta de la cámara.

SetFrameRate(Single rate)
Ajusta el framerate según el parámetro de entrada en Hz.
Nota: La cámara auto-ajusta ciertos valores de entrada.

Develve mensaje de respuesta de la camara.

GetFrameRate(ref Single rate)
Obtiene el framerate de la cámara por referencia.

Devuelve mensaje de respuesta de la cámara.

SetInputTrigger(int trigger, bool enabled)
Activa o desactiva el trigger indicado por parámetro.

Devuelve mensaje de respuesta de la cámara.

GetInputTrigger(int trigger, ref bool enabled)
Obtiene por referencia el estado del trigger de salida seleccionado.

Devuelve mensaje de respuesta de la cámara.

SetOutputTrigger((int trigger, bool enabled)
Activa o desactiva el trigger de salida seleccionado.

Devuelve mensaje de respuesta de la cámara.

GetOuptutTrigger(int trigger, ref bool enabled)
Devuelve el estado por referencia del trigger de salida seleccionado.

Devuelve mensaje de respuesta de la cámara.

CheckStatus(ref bool connected)
Función que comprueba el estado de la cámara, devuelve parámetro booleano por referencia que indica si la cámara está conectada o no, sirve para comprobar que la cámara sigue operativa.

Devuelve mensaje de respuesta de la cámara, se puede dar el caso de que obtengamos false, aunque la cámara esté operativa, este mensaje que devuelve la función nos aportará más información.

StartCapture()
Función para empezar la captura, esta función activa los eventos de frame.

StopCapture()
Función para parar la captura, al lanzar esta función se dejará de recibir eventos de frame de la cámara.

kill()
Esta función libera la DLL de control de la cámara de memoria, por lo tanto se liberan todos los controles y callbacks de la cámara.

MemFree(ref MemHandle aMem)
Función que libera la memoria asociada a una variable handle pasada por parámetro.

Nota: Se recomienda usar solo para depuración, la liberación de handles la realiza la propia librería de control.

SetFrameEvent(bool marked)
Función para especificar si se debe emitir un evento de frame por trigger externo, o emitir todos los eventos de frames.

GetFrameEvent(ref bool marked)
Devuelve por referencia la configuración de emisión de eventos.

GetTriggerCnt()
Devuelve el número de triggers externos registrados.

4.1.3.5. Uso

En la Solución de la librería de control, se incluye el proyecto AEVarioCamControlTest, este proyecto proporciona una interfaz gráfica de prueba que sirve para la verificación del funcionamiento de la librería de control, además ofrece a los ingenieros que evalúen la cámara, un punto de partida ya que pueden testear el hardware y la librería de clases planteada en este TFG sin la necesidad de desarrollar un programa con dicha librería.

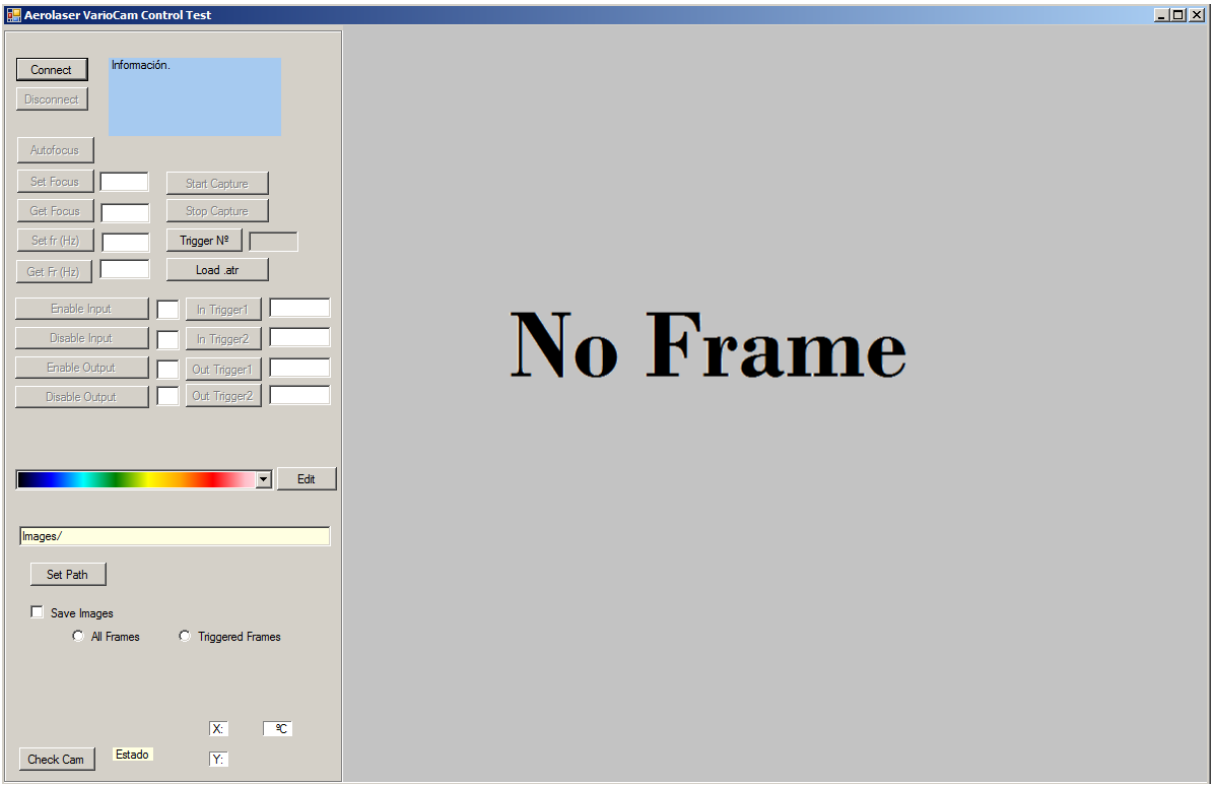


Figura 4.10: VarioCamControlTest

Este proyecto de test ofrece un control en la interfaz de usuario para cada función de la librería de control, además muestra un ejemplo de como guardar termografías, ya que como se puede ver, la librería de control, como su nombre indica, no se encarga de la visualización de termografías ni de su almacenado en disco.

A continuación se explicará brevemente el proceso de dibujar un frame enviado por la cámara en la interfaz gráfica. El almacenado en disco se cubrió en el apartado anterior correspondiente a la librería de serialización, AEThermoSerializer.

Primero se debe crear una función delegado y asignarla a OnNewFrame del SDK:

```
IRBgrabLib.SetOnNewFrameCallback(hDevHandle, this);
IRBgrabLib.OnNewFrame += new EventHandler(NewFrameArrived);
```

En dicha función delegado, nos ocuparemos de obtener los datos del frame, construir un evento y lanzarlo.

```
if (IRBgrabLib.GetDataPtr(aMem, ref pSrc, ref memsize) !=
    TIRBG_RET.IRBG_RET_OK) return;

FrameStatus frameEvent = new FrameStatus();

frameEvent.vecPtr = new float[ww * hh];
Marshal.Copy(pSrc, frameEvent.vecPtr, 0, ww * hh);

evFrame(this, frameEvent);
```

Una vez tenemos definido el evento, podemos por ejemplo, visualizar el vector de temperaturas que nos proporciona la cámara en formato matriz, guardada en un vector por filas, construyendo un mapa de bits. El proceso para generar dicho mapa de bits se explicará a continuación.

El primer paso que hay que realizar es cuantificar los valores de temperatura, de forma sencilla, se puede usar un rango valores de 0 a 255 para la cuantificación ya que estos valores representan un byte que podemos aprovechar para representar un componente de color, con este byte obtenido podemos construir un mapa de bits usando el mismo valor para cada componente R, G y B. Esto generará una termografía en escala de grises.

Por último, a la hora de desarrollar programas con esta librería de control, para garantizar el correcto funcionamiento de un programa realizado con esta, se debe tener en cuenta lo siguiente.

Rendimiento de disco

Windows por defecto genera un log para los eventos de red y nuestra cámara según su implementación, genera los frames en forma de evento, por lo que a una frecuencia de 30 Hz el rendimiento del disco se ve afectado con un consumo del 100 % porque los eventos se almacenan en un archivo especial en disco.

Para consultar la configuración de netevents, se debe ejecutar en consola elevada: netsh wfp show options netevents

Para cambiar la configuración: netsh wfp set options netevents=on/off

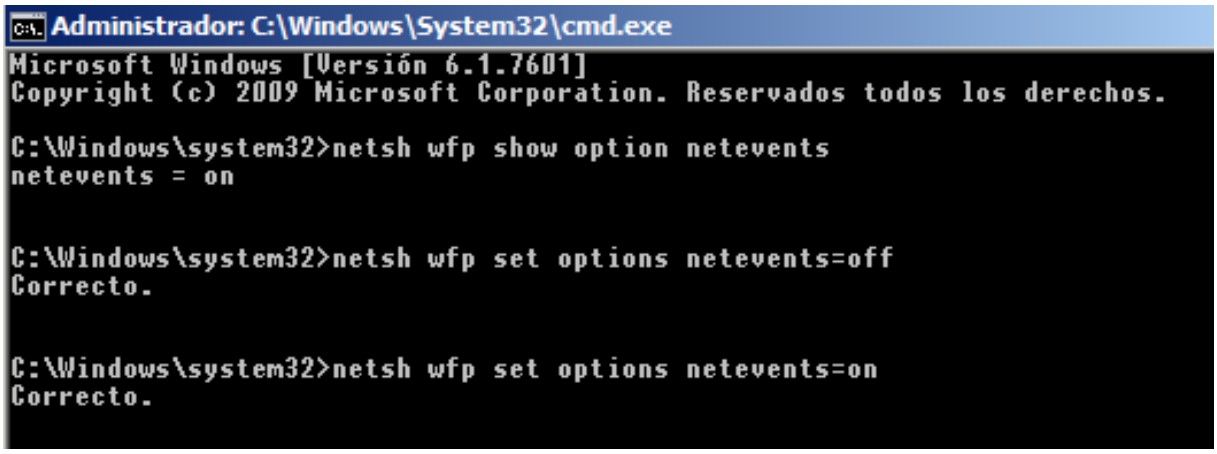


Figura 4.11: Consola de windows

Para realizar esto desde un programa desarrollado, hay que incluir el siguiente código:

```
System.Diagnostics.ProcessStartInfo myProcessInfo = new
    System.Diagnostics.ProcessStartInfo("cmd");

myProcessInfo.FileName =
    Environment.ExpandEnvironmentVariables("\%SystemRoot\%")
    + @"\System32\cmd.exe";
myProcessInfo.Arguments = "/c␣netsh␣wfp␣set␣options␣netevents=off";
myProcessInfo.WindowStyle =
    System.Diagnostics.ProcessWindowStyle.Hidden;
myProcessInfo.Verb = "runas";
System.Diagnostics.Process.Start(myProcessInfo);
```

Nota: Es buena práctica, comprobar primero en que estado está netevents, y al cerrar el programa, dejarlo como estaba por defecto.

Permisos de administrador

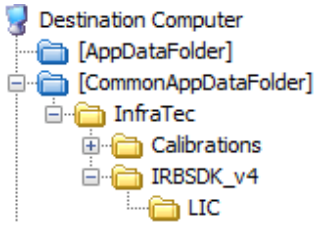
Para que el programa al ser ejecutado requiera permisos de administrador (Para desactivar netevents) hay que añadir al proyecto un fichero de manifest y modificar la siguiente linea requestedExecutionLevel:

```
<?xml version="1.0" encoding="utf-8"?>
<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1">
  <assemblyIdentity version="1.0.0.0" name="MyApplication.app"/>
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
        <!-- Opciones del manifiesto UAC
             Si quiere cambiar el nivel del Control de cuentas de usuario de Windows reemplace el
             nodo requestedExecutionLevel por uno de los siguientes.

        <requestedExecutionLevel  level="asInvoker" uiAccess="false" />
        <requestedExecutionLevel  level="requireAdministrator" uiAccess="false" />
        <requestedExecutionLevel  level="highestAvailable" uiAccess="false" />

            Especificar el elemento requestedExecutionLevel deshabilitará la virtualización de archivos y registros.
            Quite este elemento si la aplicación necesita esta virtualización para la compatibilidad
            con versiones anteriores.
        -->
        <requestedExecutionLevel level="requireAdministrator" uiAccess="false" />
      </requestedPrivileges>
    </security>
  </trustInfo>
```

Figura 4.12: Archivo de manifiesto del proyecto

Licencia

A la hora de crear un proyecto de setup se debe incluir la licencia en la instalación.



El SDK instala este archivo en %PROGRAMDATA %/InfraTec debemos mantener esta estructura en el proyecto setup.

Figura 4.13: Jerarquía de carpetas en PC destino

Mapa de código

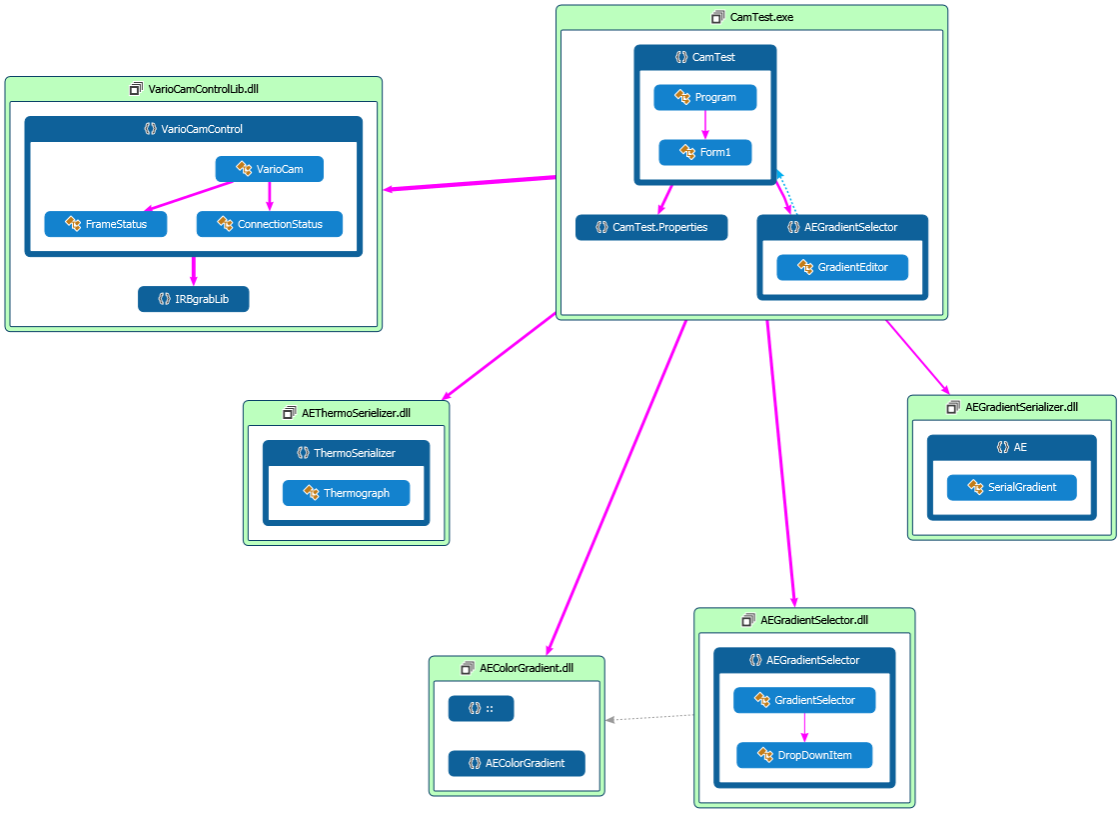


Figura 4.14: Diagrama AEVarioCamControLib

4.1.4. AEThermoViewer

4.1.4.1. Antecedentes

Ante la falta de opción para visualizar y post-procesar los datos obtenidos mediante los proyectos mencionados anteriormente, se optó por desarrollar esta aplicación mediante la cual un usuario podrá abrir un archivo serializado, visualizar el termograma, cambiar el gradiente de colores y exportar el resultado final a un formato imagen, entre otras opciones.

4.1.4.2. Estado del Arte

Para la captura y post-procesado de termografías existe el programa IRBIS 3, requiere adquirir licencia para ser usado. El principal inconveniente supone la limitación de tener que usar dicho software para adquirir las termografías, estas tienen una extensión .irb que es un formato propietario, mientras que el objetivo de este Trabajo de Fin de Grado es permitir la integración del hardware en otros programas.

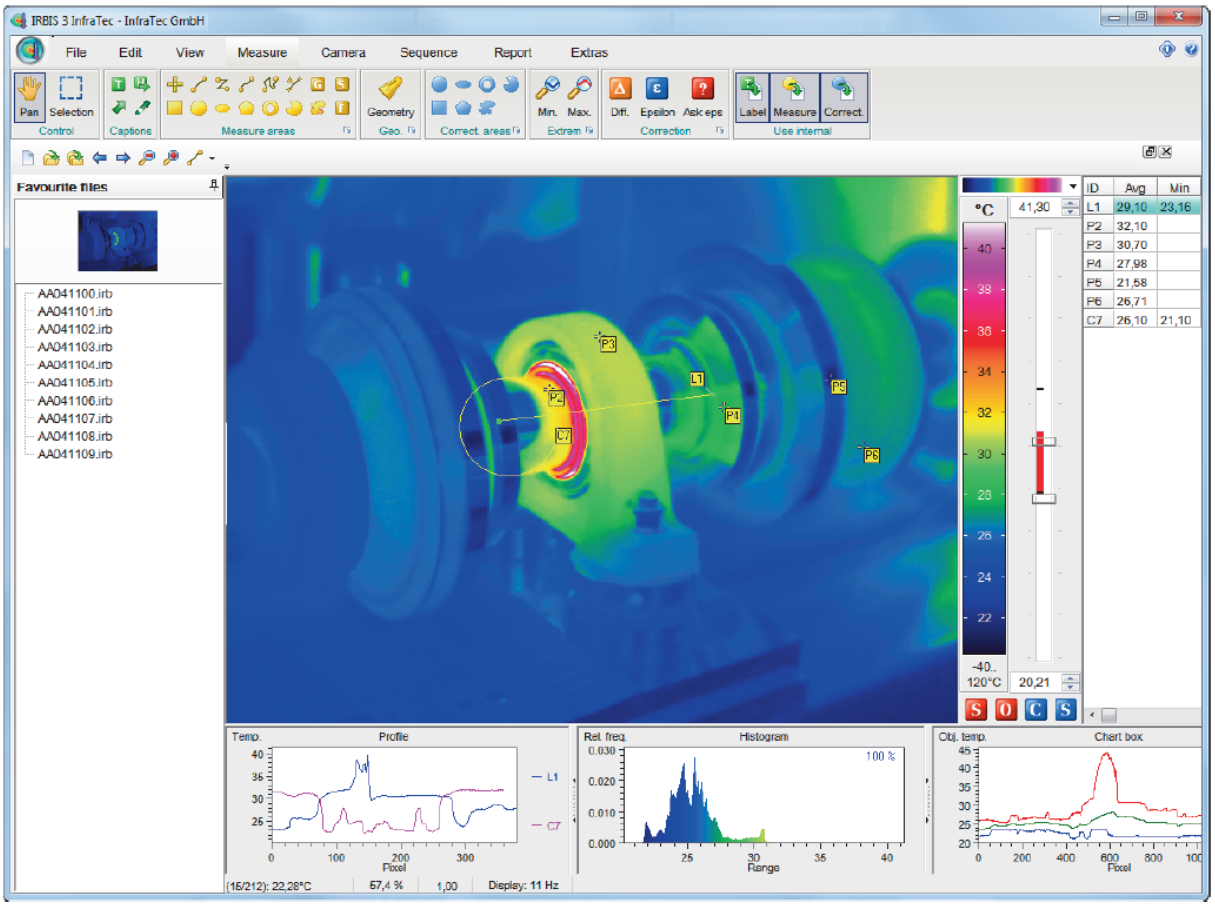


Figura 4.15: IRBIS 3

Por otro lado, junto al SDK se incluye el programa IRBISView, este a diferencia del anterior, no requiere licencia, aunque solo sirve para visualizar archivos .irb y exportarlos a formato imagen. Como ya hemos comentado anteriormente .irb es un formato propietario, el SDK no exporta funciones de guardado por lo que este programa no tiene utilidad alguna según nuestras necesidades, es por esto por lo que se desarrollo un contenedor propio (formato .atr) y este visor, AEThermoViewer.

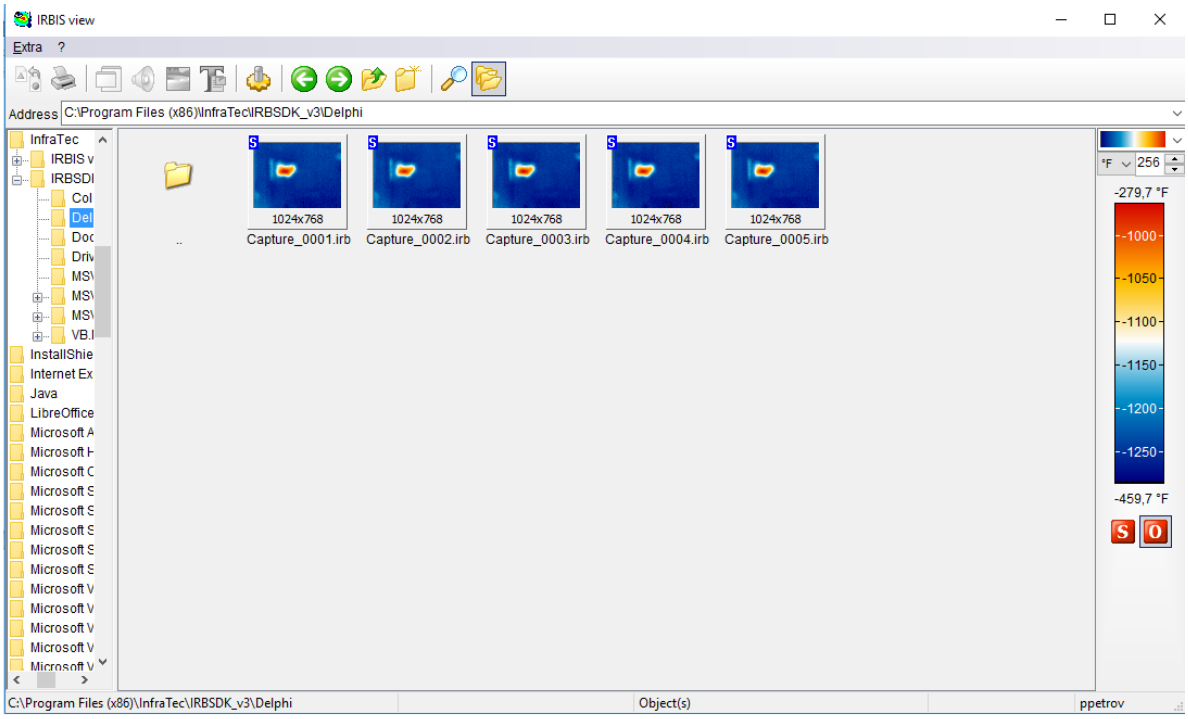


Figura 4.16: IRBISView

Conclusiones obtenidas

Tras el análisis de las aplicaciones mencionadas, obtuvimos una serie de características deseables para nuestro software. Pasamos a analizar dichas conclusiones:

- **Miniaturas:** El formato creado no es estándar, lo cual implica la necesidad de tener un software que muestre miniaturas de la imagen a mostrar para ayudar al usuario a la hora de seleccionar que archivo desea abrir.
- **Interfaz de uso:** Debe ser sencilla y directa, ya que hemos visto que por ejemplo IRBIS 3 es tan compleja que incluso incorpora manuales de uso. Deberíamos conseguir que nuestro software tenga un uso intuitivo y evitar la necesidad de manuales.
- **Exportación:** Nuestro formato es un formato apropiado para máquina, esto crea la necesidad de una vez post-procesada la termografía, poder exportarla a un archivo imagen.Por otro lado, las aplicaciones mencionadas, no ofrecían exportación por lotes, esta característica es útil en el campo en el que se usará nuestra aplicación.
- **Gradiente:** Los usuarios podrán cambiar de gradiente, modificarlo, importar y exportar.

- **Obtención de temperaturas:** El usuario puede consultar las temperaturas mínimas,máximas y medias del termograma, así como, consultar la temperaturas en coordenadas especificas, haciendo click en dicha coordenada.

4.1.4.3. Requisitos del software

La interacción que tiene el usuario con la interfaz se muestra a continuación a travéz de casos de uso.

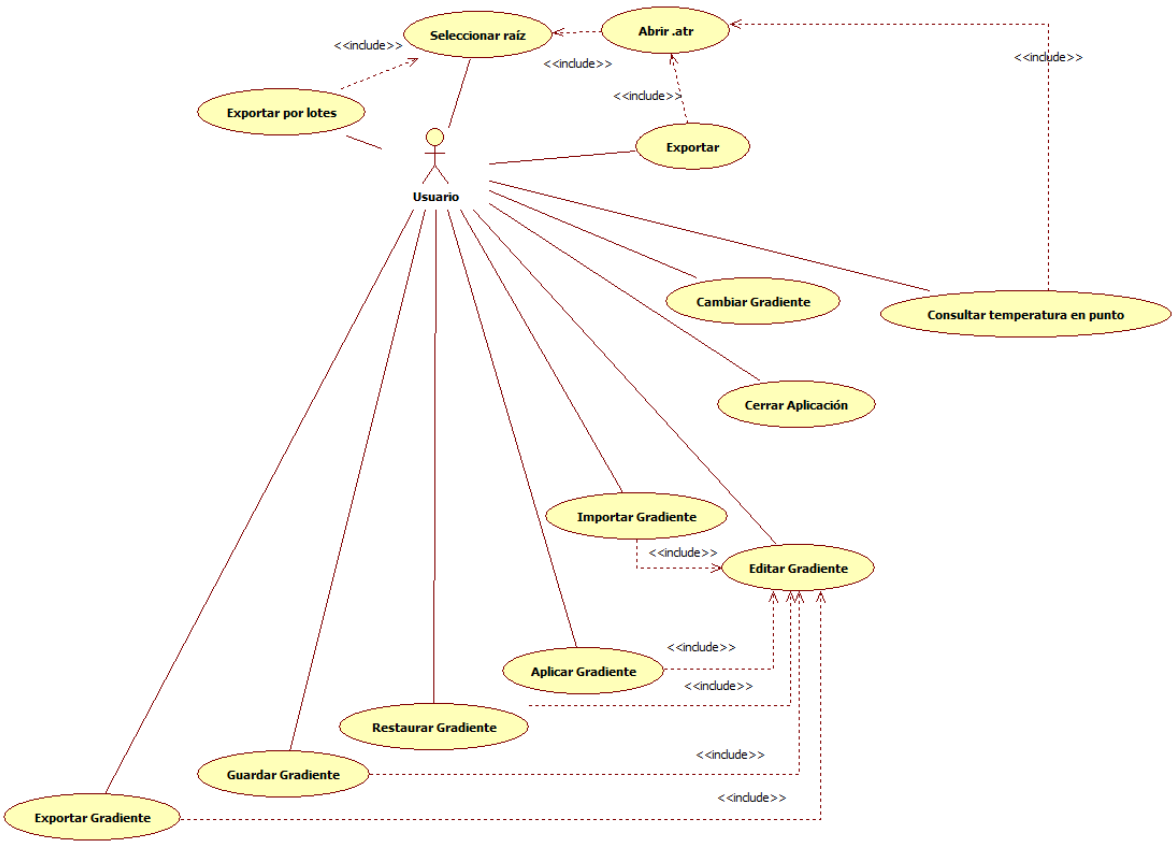


Figura 4.17: Casos de uso de la intefaz gráfica

Nombre del caso de uso	Seleccionar raíz
Descripción	Seleccionar el directorio raíz donde se encuentran los archivos deseados.
Actores	Usuario
Precondiciones	-
Postcondiciones	-
Flujo básico	El usuario navega por el árbol de directorios y selecciona el directorio raíz sobre el que trabajar.

Cuadro 4.1: Tabla Casos de uso: Seleccionar raíz.

Nombre del caso de uso	Abrir .atr
Descripción	El usuario abre una termografía
Actores	Usuario.
Precondiciones	Debe haberse seleccionado un directorio raíz (Haber realizado el caso de uso Seleccionar raíz)
Postcondiciones	-
Flujo básico	El usuario selecciona una termografía y la abre. Se muestra en el visor.

Cuadro 4.2: Tabla Casos de uso: Abrir .atr

Nombre del caso de uso	Exportar
Descripción	Se exportará la imagen con el formato seleccionado
Actores	Usuario.
Precondiciones	Debe haberse abierto un archivo .atr(Haber realizado el caso de uso Abrir .atr)
Postcondiciones	-
Flujo básico	El usuario selecciona exportar, y la ruta a guardar. La imagen que se muestra en la UI será exportada al formato seleccionado.

Cuadro 4.3: Tabla Casos de uso: Exportar.

Nombre del caso de uso	Exportar por lotes
Descripción	El usuario selecciona un directorio que contiene archivos .atr a exportar en el formato seleccionado.
Actores	Usuario.
Precondiciones	Haber realizado el caso de uso Seleccionar raíz
Postcondiciones	-
Flujo básico	El usuario selecciona el directorio raíz, ejecuta exportar por lotes y se exportan todos los ficheros de dicha raíz, con el gradiente actual.

Cuadro 4.4: Tabla Casos de uso: Exportar por lotes.

Nombre del caso de uso	Consultar temperatura en punto
Descripción	El usuario consulta la temperatura en el punto que desee.
Actores	Usuario.
Precondiciones	Haber realizado el caso de uso Abrir .atr
Postcondiciones	-
Flujo básico	El usuario hace click sobre algún punto de la termografía y la temperatura en ese punto aparece en la interfaz.

Cuadro 4.5: Tabla Casos de uso: Consultar temperatura en punto.

Nombre del caso de uso	Cambiar Gradiente
Descripción	El usuario cambia de gradiente con el que colorear las termografías.
Actores	Usuario.
Precondiciones	-
Postcondiciones	-
Flujo básico	El usuario abre una lista de gradientes y selecciona el gradiente deaseado, la termografía se colorea según dicho gradiente.

Cuadro 4.6: Tabla Casos de uso: Cambiar Gradiente.

Nombre del caso de uso	Cerrar Aplicación
Descripción	El usuario cierra la interfaz.
Actores	Usuario.
Precondiciones	-
Postcondiciones	-
Flujo básico	El usuario selecciona cerrar y la aplicación deja de ejecutarse.

Cuadro 4.7: Tabla Casos de uso: Cerrar Aplicación.

Nombre del caso de uso	Editar gradiente
Descripción	El usuario invoca una ventana de edición de gradiente.
Actores	Usuario.
Precondiciones	-
Postcondiciones	-
Flujo básico	Se abre la ventana de edición de gradiente.

Cuadro 4.8: Tabla Casos de uso: Editar gradiente.

Nombre del caso de uso	Aplicar Gradiente
Descripción	El usuario aplica las modificaciones al gradiente.
Actores	Usuario.
Precondiciones	Haber realizado el caso de uso Editar gradiente.
Postcondiciones	-
Flujo básico	El usuario aplica las modificaciones al gradiente y se refleja, coloreando la termografía con el nuevo gradiente.

Cuadro 4.9: Tabla Casos de uso: Aplicar Gradiente.

Nombre del caso de uso	Restaurar Gradiente
Descripción	El usuario revierte las modificaciones del gradiente.
Actores	Usuario.
Precondiciones	Haber realizado el caso de uso Editar gradiente.
Postcondiciones	-
Flujo básico	El usuario hace click en revertir y se restaura el gradiente por defecto y se colorea la termografía con dicho gradiente.

Cuadro 4.10: Tabla Casos de uso: Restaurar Gradiente.

Nombre del caso de uso	Guardar Gradiente
Descripción	El usuario guarda la configuración del gradiente.
Actores	Usuario.
Precondiciones	Haber realizado el caso de uso Editar gradiente.
Postcondiciones	-
Flujo básico	El usuario hace click en guardar gradiente, este se guarda en la carpeta de instalación de la aplicación.

Cuadro 4.11: Tabla Casos de uso: Guardar Gradiente

Nombre del caso de uso	Exportar Gradiente
Descripción	El usuario exporta la configuración del gradiente en la ruta deseada.
Actores	Usuario.
Precondiciones	Haber realizado el caso de uso Editar gradiente.
Postcondiciones	-
Flujo básico	El usuario hace click en Exportar, se abre dialogo del administrador de carpetas del sistema. Una vez seleccionada la ruta e introducido el nombre de fichero, se guarda el gradiente en formato .agr

Cuadro 4.12: Tabla Casos de uso: Exportar Gradiente

Nombre del caso de uso	Importar Gradiente
Descripción	El usuario importa la configuración de gradiente.
Actores	Usuario.
Precondiciones	Haber realizado el caso de uso Editar gradiente.
Postcondiciones	-
Flujo básico	El usuario hace click en el botón de importar, se abre un diálogo para seleccionar el archivo, una vez seleccionado, el gradiente se importa.

Cuadro 4.13: Tabla Casos de uso: Importar Gradiente

4.1.4.4. Arquitectura del software

A continuación se procederá a explicar los diferentes apartados en los que consintió la elaboración de este visor. Comenzaremos explicando el selector de gradientes y el gradiente en si, ya que el generador de gradientes es la función básica de nuestra aplicación. El serializador de termografías se cubrió anteriormente en esta memoria.

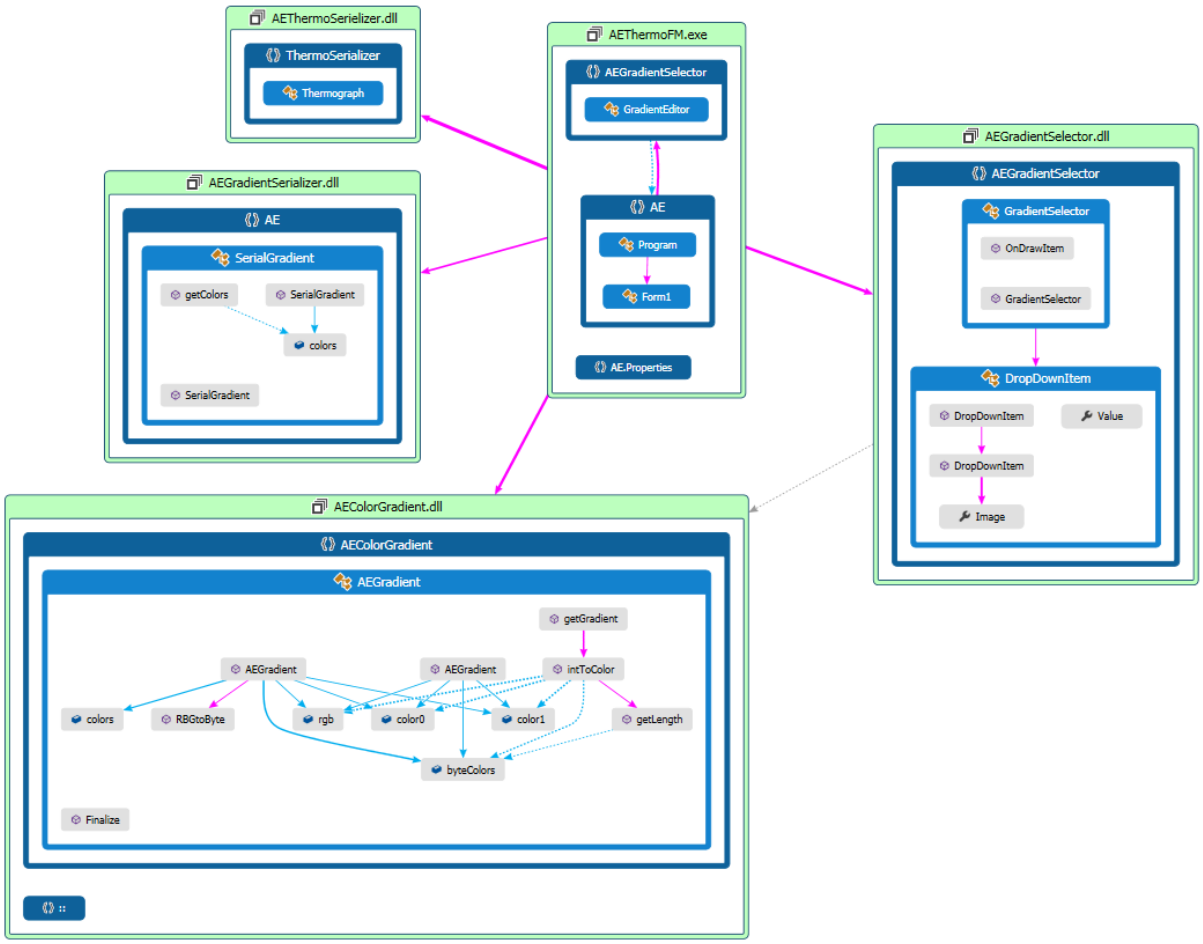


Figura 4.18: Arquitectura del software.

Clase AEGradient

Un gradiente es simplemente un vector de colores, el motivo principal que llevo a desarrollar esta clase fueron los resultados al usar la clase Color de .NET, la clase Color es un objeto con información RGB y el uso de este objeto suponía un retardo de alrededor de 2 segundos en el coloreado de nuestra termografía, esto llevó a la creación de esta clase, a la cual se le pasa un vector Colors[] y esta construye una matriz escalonada, con los bytes del gradiente correspondientes para cada componente R, G y B.

Clase AEGradientSelector

.NET no presenta un elemento gráfico del tipo desplegable en el que se puedan incluir imágenes (En este caso, gradiente) por lo que tuvimos que redefinir DropDownItem.
Se optó por crear estas clases por separado a modo de librería, para poder reutilizarlas fácilmente en el futuro, aparte, haciendo uso de dichas librerías se mejoró el programa demo de la librería de control añadiendo el gradiente y el selector, ya que previamente presentaba una imagen en escala de grises.

AEGradientSerializer

Para las funciones de exportado, guardado e importado, es necesario crear una clase de serialización. Esta es muy sencilla, en el constructor se le pasa un vector de Colors[] para serializar el objeto, y posee un método getColors() que devuelve dicho vector, a la hora de de-serializar.

4.1.4.5. Resultados

Tomando como referencia los objetivos que se marcaron en un principio para este visor, puede concluirse que los resultados son favorables pues se han implementado todas las funcionalidades propuestas y el resultado compite incluso con la aplicación del fabricante IRBISView.

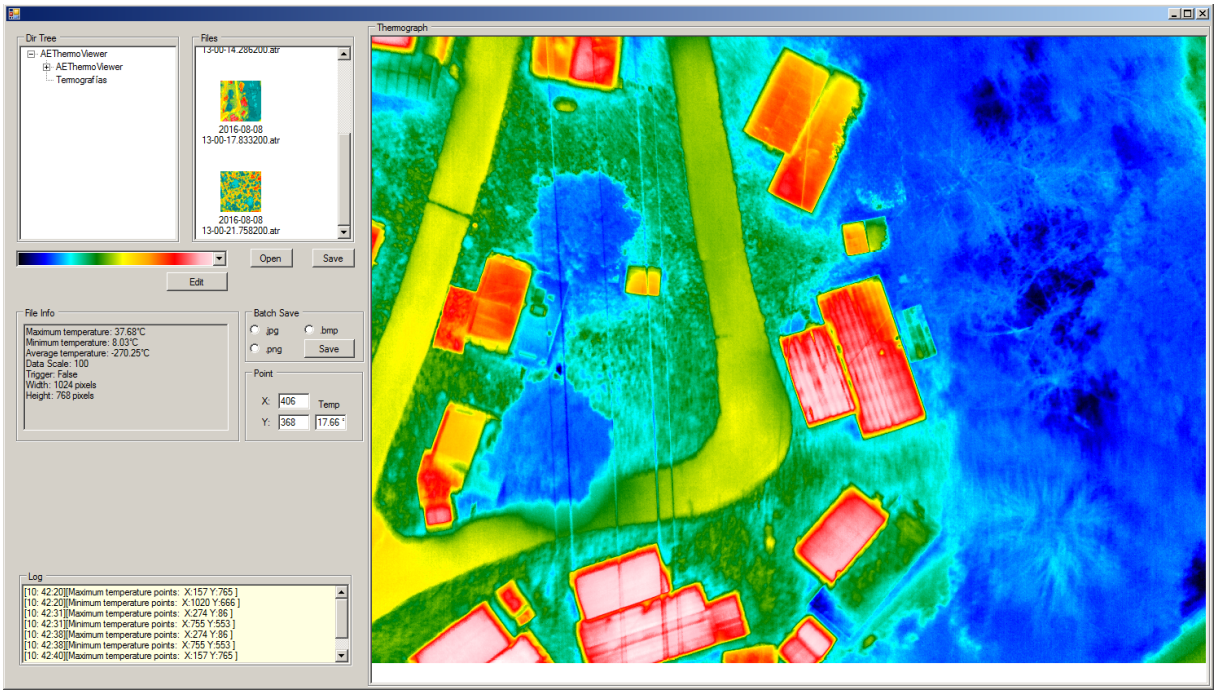


Figura 4.19: Resultado final: Termografía tomada por Aerolaser en vuelo de prueba de la librería.

5

Conclusiones y trabajos futuros

Para finalizar esta memoria se expondrán las conclusiones y el trabajo futuro.

5.1. Conclusiones

Pese a que estudiar documentación de un hardware junto a su SDK y crear una librería de control es igual de complejo como cualquier proyecto de software, este proyecto tuvo su complejidad añadida ya que la documentación era pobre y el SDK estaba incompleto, cosa que el fabricante confirmó por e-mail, afirmando que se completaría a finales de año.

Aunque nos encontrásemos con este tipo de problemas, hemos sido capaces de desarrollar varios incrementos de la aplicación funcional, en la cual están desarrolladas todas las funciones requeridas por la empresa. Se considera que este proyecto genera valor al aportar una librería y aplicación útiles que cubre una necesidad real.

Cabe mencionar que estamos orgullosos con el desarrollo del proyecto. Se ha aprendido mucho por el camino y hemos mejorado como profesionales, ya que nos hemos enfrentado a un desarrollo real de carácter laboral.

Por último, el hecho de que este proyecto, está siendo actualmente usado en la empresa Aerolaser System S.L., para la adquisición de imágenes y evaluación del hardware apoya nuestra conclusión.

Por tanto creemos férreamente haber alcanzado exitosamente los objetivos académicos planteados por el TFG.

5.2. Trabajos futuros

Como trabajos futuros se destacan los siguientes:

- Portar la librería a C++. Este proyecto se realizó en C# ya que la empresa pretendía usarla en proyectos existentes en C#, no obstante, la librería en C++ aportaría flexibilidad y mejoraría rendimiento al separarse del runtime de .NET.
- Añadir funciones al visor, como pueden ser, detección de patrones de temperaturas, búsqueda de puntos fríos / puntos calientes y selección de área para trazar diferencias de temperaturas.
- Crear un componente de sistema para que el explorador de windows genere thumbnails y poder previsualizar las termografías desde el explorador de windows sin la necesidad de abrir el visor de termografías.
- Añadir funcionalidad de guardar thumbnails del explorador de archivos de AEThermoViewer para no tener que generar los thumbnail en cada ejecución.

Acrónimos

DLL Dynamic-link Library. 20, 21, 34, 35, 37

GUI Graphic User Interface. 21

IDE Integrated Development Environment. 26

SDK Software Development Kit. 15, 20, 21, 23, 24, 29, 34, 39, 41, 44, 50

TFG Trabajo de Fin de Grado. 16, 17, 38, 50

Glosario

Callback Una función “A” que se usa como argumento de otra función “B”. Cuando se llama a “B”, ésta ejecuta “A”.. 21

funciones exportada Una función que ha sido compilada en una librería dinámica o estática.. 20

Solución Contenedor de Visual Studio con propósito de administrar eficazmente los elementos necesarios para el desarrollo, como referencias, conexiones de datos, carpetas y archivos. Una solución puede tener uno o varios proyectos.. 38

Estructura del repositorio

El repositorio que acompaña a este documento presenta la siguente estructura:

- **EII-GII-2017-07-MihaylovPetrov_P-TrujilloPino_A-TFG.pdf** Este documento, en formato PDF.
- **LaTex.zip** Carpeta comprimida con el proyecto LaTex usado para la generación de este documento.
- **Resumen.txt** Resumen en español e inglés de este documento en formato TXT.
- **Código/** Carpeta que incluye el código fuente en un comprimido .zip con la jerarquía de subcarpetas de Visual Studio.

A

Manual de usuario

A.1. AEThemoSerializer

Para poder usar esta librería en proyectos para la serialización o de-serialización de objetos, esta se debe incluir como referencia, existen dos opciones:

- Añadir referencia a un archivo binario (.dll)
- Añadir el proyecto de AEThermoSerializer a la solución actual y añadir la referencia a proyecto.

Para la primera opción, nos dirigimos a **Proyecto - Agregar Referencia - Examinar** Se nos abrirá un diálogo para seleccionar el archivo binario .dll que deseamos usar.

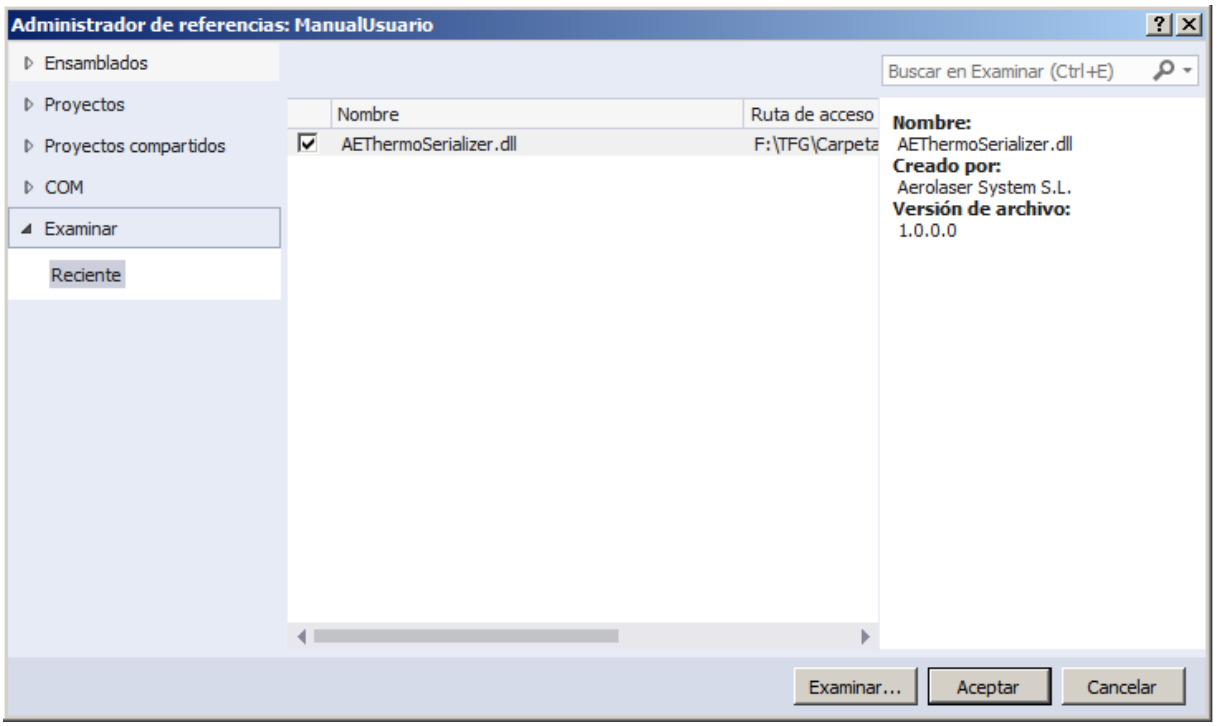


Figura A.1: Manual: Diálogo de añadir referencias.

Hecho esto, nos aparecerá en el explorador de soluciones en el apartado de referencias. Podemos darle doble click para invocar el explorador de objetos y ver los métodos que ofrece.

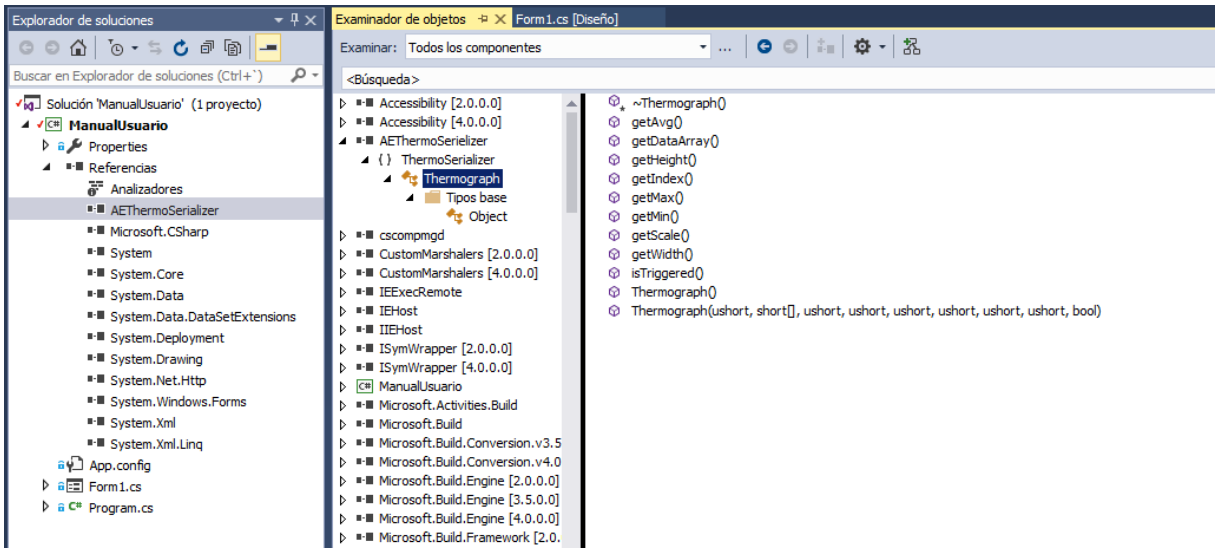


Figura A.2: Manual: Examinador de objetos.

Para la segunda opción, como mencionamos anteriormente debemos agregar el proyecto de la librería de serialización a la solución. Esto se realiza haciendo click derecho sobre la solución en el explorador de soluciones
sectionAgregar - Proyecto existente...

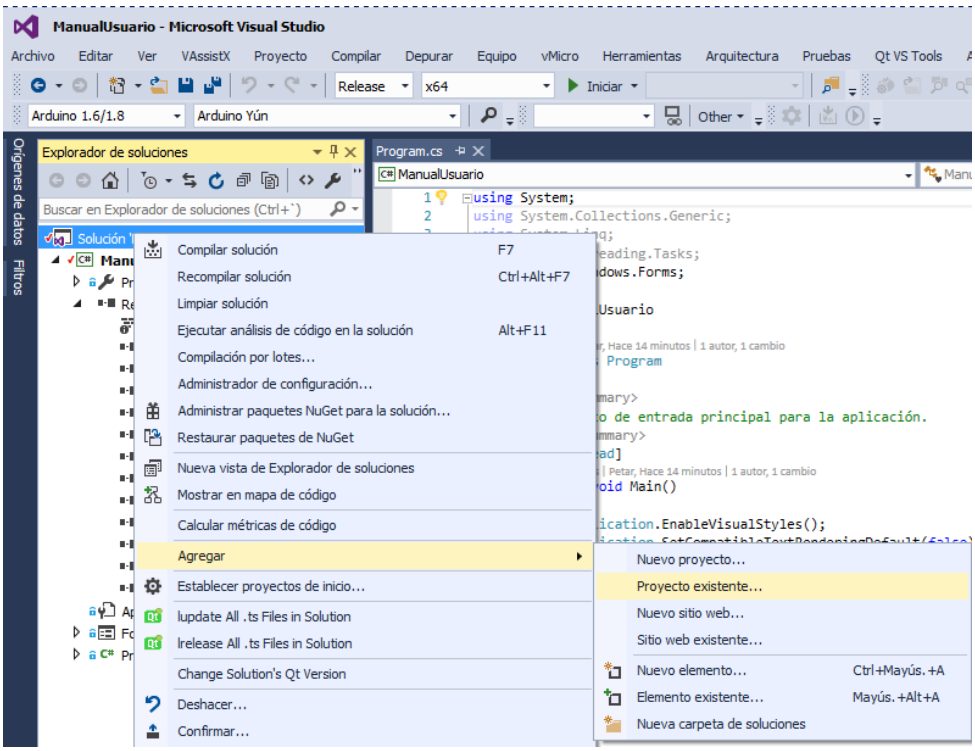


Figura A.3: Manual: Agregar proyecto existente.

Una vez tenemos nuestro proyecto incluido a la solución solo queda agregar la referencia de manera similar al paso anterior, pero en este caso: **Proyecto - Agregar Referencia - Proyecto**

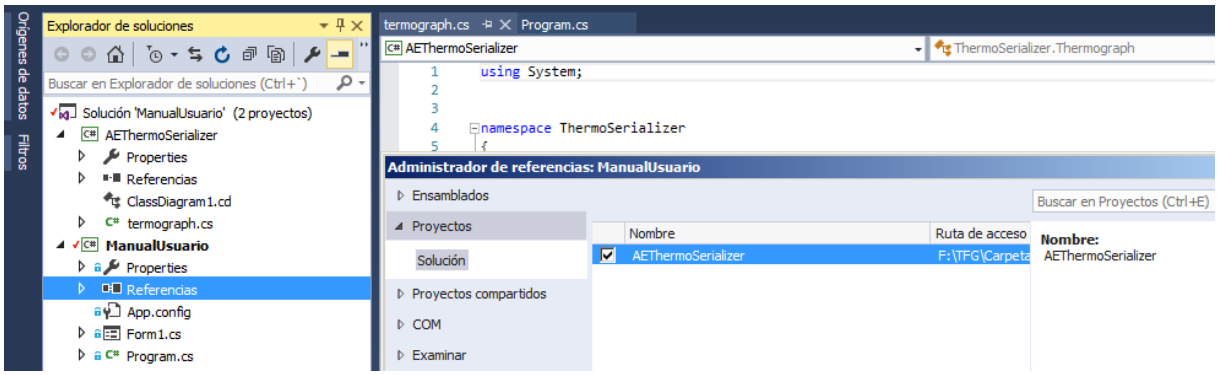


Figura A.4: Manual: Agregar referencia desde proyecto.

Por último, solo queda comentar el uso de esta clase, al tener la referencia agregada al proyecto ya inicializar objetos con dicha clase referenciada a continuación se mostrará código de ejemplo para la serialización y para la de-serialización.

Serialización de objetos:

```
Thermograph tgObj = new Thermograph(0,inVec, (ushort)w,
    (ushort)h, (ushort)minvs, (ushort)maxv, (ushort)avg,
    (ushort)scale, trig);

FileStream stream = new FileStream("Images/" +
    DateTime.Now.ToString("yyyy-MM-dd_HH-mm-ss.ffffff")
    + ".atr", FileMode.Create);

BinaryFormatter myFormatter = new BinaryFormatter();
myFormatter.Serialize(stream, tgObj);
stream.Close();
```

De-serialización de objetos:

```
FileStream stream = new FileStream(selectedFilePath,
    FileMode.Open);

BinaryFormatter myFormatter = new BinaryFormatter();

Thermograph obj = (Thermograph)(myFormatter.Deserialize(stream));


short scale = (short) obj.getScale();
float mintest = obj.getMin();
float minv = obj.getMin() / (float)scale;
float maxv = obj.getMax() / (float)scale;
float avg = obj.getAvg() / (float)scale;
bool trig = obj.isTriggered();
```

A.2. AEVarioCamControlLib

Este proyecto, como el anterior mencionado es una librería, para su uso se aplican los mismos conceptos detallados en el apartado anterior. Para poder usar la librería hay que agregarla como referencia.

Se recomienda agregarla según el segundo método explicado ya que posee comentarios xml y esto ayuda al entorno intelliSense de Visual Studio a mostrar las sugerencias, esto ayuda al programador al obtener información sobre un objeto o método desde el entorno de desarrollo sin la necesidad de tener que consultar los comentarios de código.

Aparte, se puede compilar el proyecto AEVarioCamControlLib con la opción /DOC, esto generará un archivo .XML en el directorio de salida con información respectiva a las clases y sus parámetros.

En la siguiente figura se muestra un ejemplo de uso de la librería, además se muestra lo mencionado anteriormente respecto a intelliSense de Visual Studio, podemos ver cómo el entorno nos ofrece información sobre un método al pasar el ratón por encima.

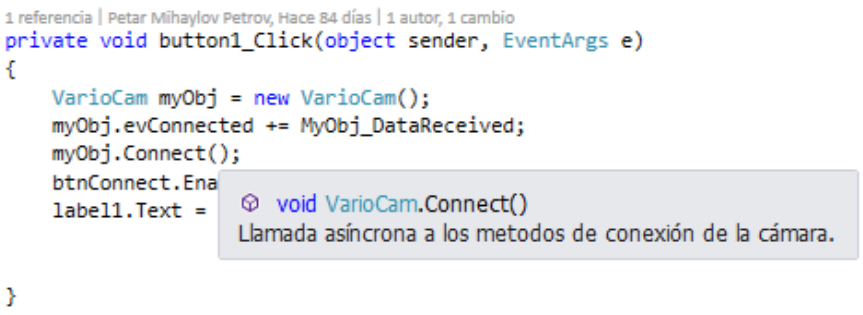


Figura A.5: Manual: Ejemplo de uso e intelliSense.

A.3. AEThermoViewer

Uno de los objetivos principales según los que se desarrollo esta aplicación, fue crear una interfaz de usuario sencilla e intuitiva que pueda ser usada por un usuario sin necesidad de un manual o tutorial.

A continuación se mostrará una figura de la interfaz, con los elementos principales enumerados.

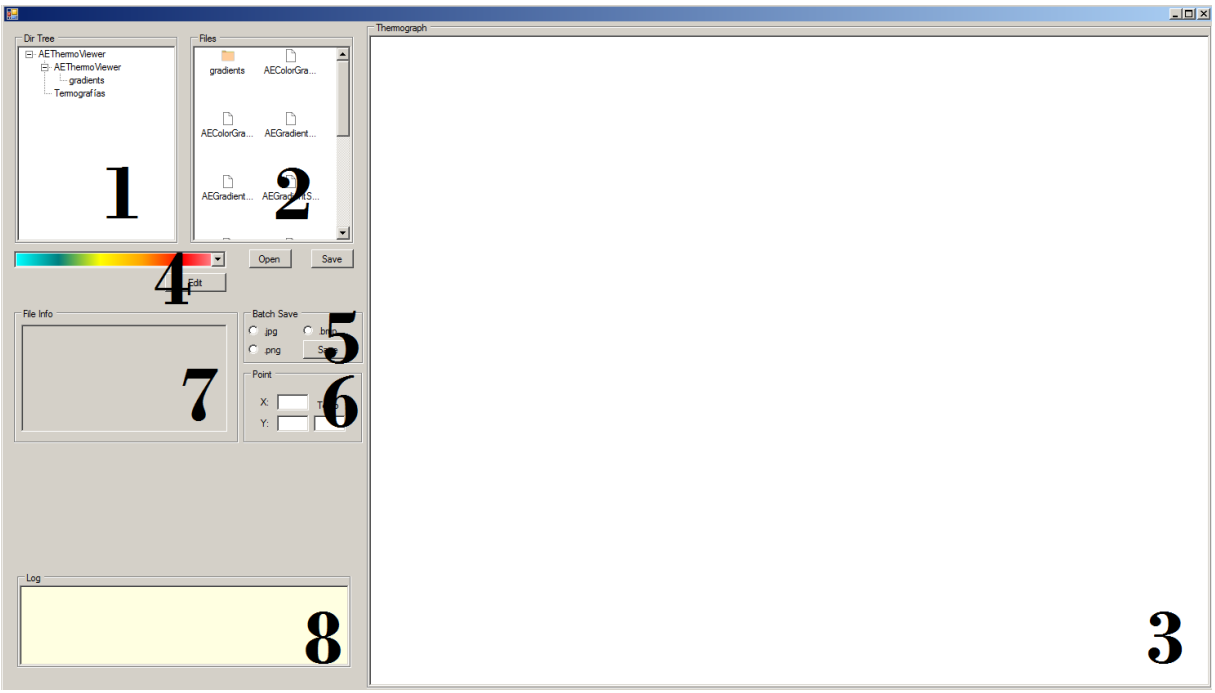


Figura A.6: Manual: Elementos de la interfaz de usuario AEThermoViewer.

- **1.** En este elemento se genera un árbol de directorios para facilitar la navegación
- **2.** Esto es un visor de directorios, muestra el contenido del directorio seleccionado en el elemento **1.**, si se encuentran archivos .atr en dicho directorio, se generará y mostrará un thumbnail. Un archivo se puede cargar haciendo doble click sobre el mismo, o desde el botón Open.
- **3.** Este área es donde se dibujará la termografía de un archivo .atr cargado,
- **4.** Mediante este elemento se puede cambiar la paleta de colores del gradiente con el que se coloreó la termografía. Haciendo click en el desplegable se pueden elegir paletas de gradiente predefinidas ó guardadas por usuario, mientras que haciendo click en editar, se abre un diálogo de edición de la paleta actual.
- **5.** Si deseamos exportar a imagen por lotes, seleccionaremos el formato deseado y la aplicación exportará todas las termografías encontradas en el elemento **2.** mientras que si deseamos exportar sólo la actual, haremos uso del botón Save.
- **6.** Este elemento muestra las coordenadas X e Y, además de la temperatura en grados del punto seleccionado sobre una termografía cargada. Para que este elemento muestre datos, debemos tener cargada una termografía y hacer click sobre un punto que deseemos consultar del elemento **3.**
- **7.** En este recuadro aparecen los campos de información embebidos en el archivo de termografía, como son: temperatura mínima, temperatura máxima, temperatura media, escala, trigger, etc.
- **8.** Este recuadro muestra el log de la aplicación, en el pueden aparecer mensajes variados o errores si los hubiera.

Bibliografía

[1] Convención de llamada. https://msdn.microsoft.com/es-es/library/46t77ak2.aspx.

[2] Intel core i7 3770k. https://ark.intel.com/es-es/products/65523/Intel-Core-i7-3770K-Processor-8M-Cache-up-to-3_90-GHz.

[3] Lucidchart. https://www.lucidchart.com/pages/es/paseo/.

[4] Nirsoft dll export. http://www.nirsoft.net/utils/dll_export_viewer.html.

[5] Nvidia gtx 650. http://www.nvidia.es/object/geforce-gtx-650-es.html#pdpContent=2.

[6] Sharelatex. https://www.sharelatex.com/about.

[7] Variocam hd head. http://www.infratec-infrared.com/thermography/infrared-camera/variocamr-hd-head-800.html.

[8] Visual studio 2015 installshield. https://msdn.microsoft.com/es-es/library/dn531020.aspx.

[9] Whitestaruml. https://sourceforge.net/projects/whitestaruml/.

[10] Peter Drayton, Ben Albahari, and Ted Neward. *C# In a Nutshell: A Destop Quick Reference.* O'Reilly, second edition, 2003.

[11] Flexera. Installshield 2015 limited edition. http://learn.flexerasoftware.com/content/IS-EVAL-InstallShield-Limited-Edition-Visual-Studio.

[12] Microsoft. Visual studio professional. https://www.visualstudio.com/es/vs/professional.

[13] Aerolaser System. Variocam hd head 800. http://aerolaser.es/es/zona_productos/productos/variocam-hd-head-800/17.html.