UNIVERSIDAD CARLOS III DE MADRID

TRABAJO FIN DE GRADO

MINIATURIZACIÓN Y AUTOMATIZACIÓN DE SISTEMA DE CAPTURA DE IMÁGENES VASCULARES

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: David Manso Yuste

Tutor: Raúl Sánchez Reíllo

Leganés, 5 de septiembre de 2013

Dedicatoria

En primer lugar, quiero dedicarle este trabajo a la gente sin la cual sería imposible estar escribiéndolo hoy: a mis padres, Clemente y María del Carmen, por su apoyo y ánimo constante, y a mis hermanas Sonsoles y Nuria.

También a Sandra y a todos mis amigos, sin nombres, para evitar dejarme alguno, por haber estado ahí siempre que lo he necesitado.

Agradecimientos

Quiero agradecer la ayuda y atención recibida (para llevar a cabo este proyecto) por mis tutores Jaime de Uriarte y Raúl Sánchez Reíllo, ya que sin su dirección y guía, hubiera sido imposible el desarrollo del proyecto.

También quiero agradecerles a mis compañeros e investigadores del GUTI (Grupo Universitario de Tecnologías de Identificación) su ayuda, ya que siempre que surgía una duda o problema, han estado dispuestos a echar una mano.

Resumen

La identificación biométrica es la identificación basada en las características biológicas, físicas o de comportamiento de una persona. Desde hace ya varios años es uno de los métodos más usados para confirmar la identidad de una persona. Su excelente tasa de reconocimiento, sumado a su riesgo prácticamente nulo de falsificación (si el sensor dispone de mecanismos de detección de sujeto vivo), ha hecho notable su uso para infinidad de aplicaciones.

Por otra parte hay modalidades o mecanismos de identificación biométrica más desarrollados que otros, debido a su mayor antigüedad.

En el presente Trabajo de Fin de Grado se va a desarrollar una parte de un sistema de identificación vascular. Partiendo del sensor de captura de imágenes vasculares ya construido, nos vamos a encargar de que la máquina tome las imágenes más apropiadas (la que mejor nos vengan para el sistema de identificación biométrica) y que lo haga de manera automática.

Si a este trabajo se le añadiera una base de datos (que almacenara las imágenes de los sujetos) y un algoritmo capaz de comparar el patrón de las venas de la mano, se podría obtener un sistema de identificación biométrico completo y valido para gran variedad de usos, como el paso a zonas privadas, el reconocimiento para algún tipo de transacción, etcétera.

Abstract

Biometric identification is the identification based on biological, physical or behavioral characteristics of a person. For several years is one of the most used methods to confirm the identity of a person. Its excellent recognition rate, coupled with low risk of forgery, has made remarkable its use for many applications.

On the other hand there are methods or biometric identification mechanisms more developed than others, due to their greater antiquity.

This Final Project Grade is going to develop a portion of a vein identification system. Starting from the machine and vascular imaging built, we`re going to take care of the machine to pick the most appropriate images (the best one that applies to the biometric identification system).

If this work is augmented by a database (which will store the images of subjects) and an algorithm able to compare the pattern of the veins of the hand, you will get a full biometric identification system and valid for a huge variety of uses such as granting access to private areas, recognition for some sort of transaction, etc.

Índice

Índice de Figuras

Índice de Tablas

Listado de Acrónimos

ADN	Ácido Desoxirribonucleico
API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
BLC	Base Class Library (Clase Base de Librería)
CCD	Charge Coupled Device (Dispositivo de Carga Acoplada)
CER	Cross-over Error Rate (Tasa de Punto de Cruce)
FAR	False Acceptance Rate (Tasa de Falso Positivo)
FMR	False Match Rate (Tasa de Fallo de Alistamiento)
FRR	False Rejection Rate (Tasa de Falso Negativo)
GUI	Graphical User Interface (Interfaz Gráfica de Usuario)
GUTI	Grupo Universitario de Tecnologías de Identificación
IR	Infrared (Infrarrojo)
LED	Light-Emitting Diode (Diodo Emisor de Luz)
NIR	Near Infrared (Infrarrojo Cercano)
PHP	Hypertext Pre-Processor
TFG	Trabajo Fin de Grado
UC3M	Universidad Carlos III de Madrid
XML	Extensible Markup Language (Lenguaje de Marcas Extensivas)

1 Introducción

A lo largo de esta memoria se va a presentar un TFG que tiene como finalidad la automatización de un sensor capaz de capturar imágenes vasculares de manos, dedos y muñecas (el TFG tomará como referencia las venas de la palma de la mano para su realización). Y a su vez, de la creación de una interfaz gráfica para que un usuario cualquiera sea capaz de utilizar el sensor.

1.1 Motivación

Basándose en los métodos biométricos existentes hasta el momento (iris, huella dactilar, reconocimiento de firma…), en su momento se creó un nuevo sistema de identificación biológico que tuviera las ventajas de los ya existentes (robustez, precisión, no invasivo…), y que evitara las desventajas (mantenimiento, desgaste…): el reconocimiento vascular.

El sistema de reconocimiento vascular posee la precisión y robustez requerida por un sistema de identificación, así como su mantenimiento y desgaste es nulo, ya que la superficie de la mano no entra en contacto con ningún componente del sensor encargado de recoger y procesar la imagen válida para la identificación.

El sensor en el que se basa este TFG es parte de un sistema de identificación biométrica desarrollado por J. Enrique Suarez Pascual, Jaime Uriarte-Antonio, Raúl Sanchez-Reillo, Michael G. Lorenz. El sensor del que partimos hoy es el desarrollo de varios prototipos anteriores:

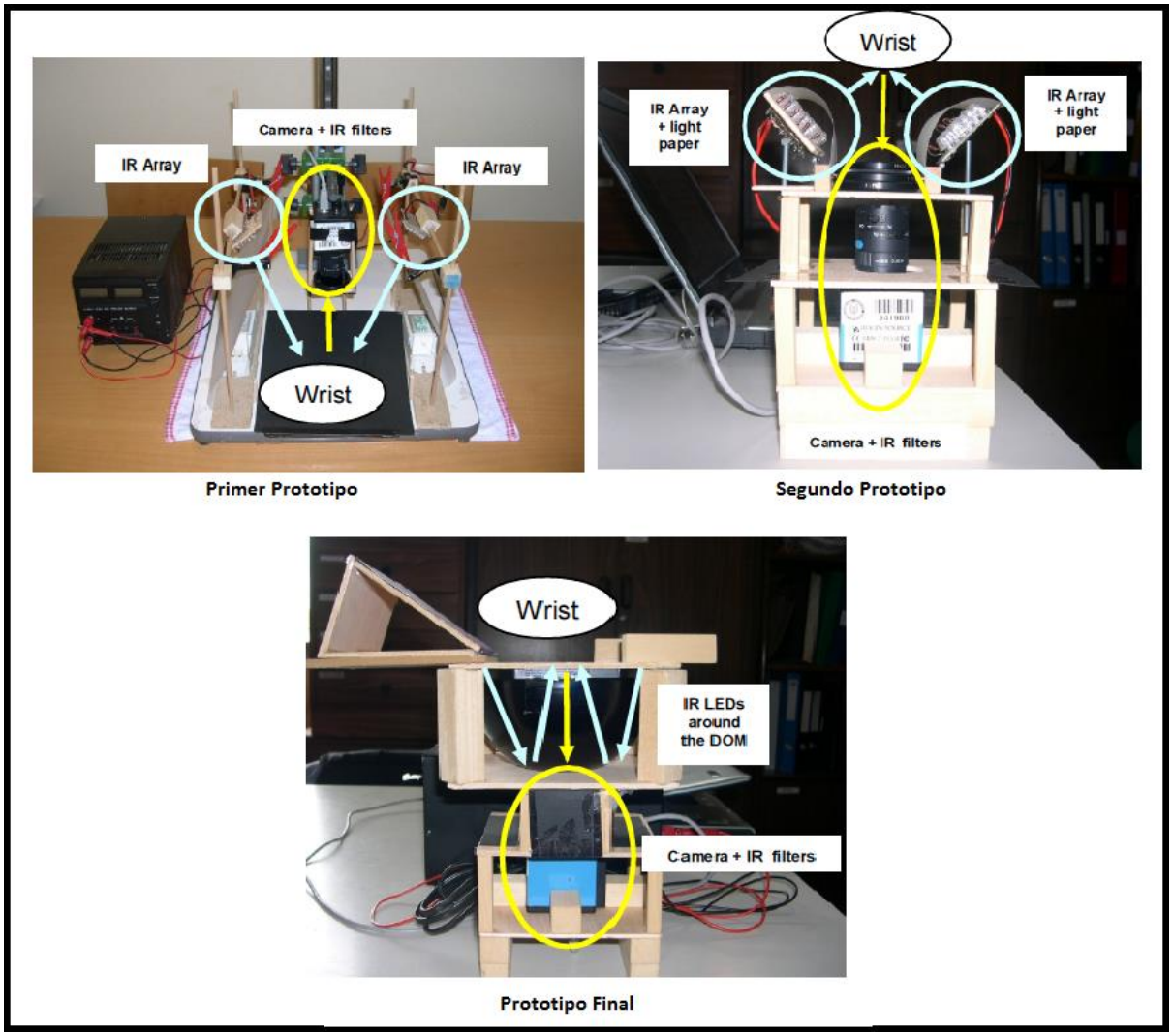

Fig. 1 – Evolución prototipos de sensores de captura de imágenes vasculares [1]

La motivación de este TFG es la dificultad de manejo para el usuario común. La manipulación dificultosa de cualquier aplicación, hace que las personas sean reacias a su uso. Lo que se pretende es la automatización y miniaturización de dicha máquina para que el usuario que vaya a utilizarla, únicamente tenga que seguir las instrucciones del programa que se desarrollará y colocar la mano frente a la cámara, para que cuando dicho programa lo crea adecuado (condiciones correctas de brillo, nitidez y enfoque), capture la imagen para su posterior procesado.

1.2 Objetivos

En el presente TFG se parte del sensor de captura de imágenes vasculares ya construido, con todos los componentes necesarios (cámara para la captura de imágenes, LED emisores de luz infrarroja, y una cúpula de homogenización para distribuir correctamente la luz emitida por los LED; todos ellos a su vez, montados en una estructura de madera), así como de una fuente de alimentación (encargada de suministrar energía a ciertos componentes de la máquina).

Hasta el momento, con el sensor encendido (LED encendidos) y conectado a un ordenador, es capaz de capturar una imagen de las venas de la parte del cuerpo que se coloque en el visor de la cámara. Para que sea efectiva esta captura, el usuario debe ser capaz de colocar la mano sobre el visor de la cámara y mantenerla inmóvil, mientras que con la otra mano (y previo conocimiento del programa) debe coger el ratón y clicar sobre el botón de "capturar" en el programa que controla la cámara.

Con el propósito de evitar este costoso proceso al usuario y hacerlo más rápido y fiable, se pretende automatizar dicha máquina, para que lo único que tenga que hacer el usuario sea colocar la palma de la mano sobre el objetivo de la cámara, y si no está bien colocada, que el mismo programa que se desarrollará, guíe de manera eficiente al usuario para que coloque la mano de manera correcta. Asimismo, se quiere desarrollar una interfaz gráfica "amistosa" para el usuario, donde se muestren las indicaciones anteriormente descritas, así como la imagen final capturada.

Para hacer el sensor más manejable para su traslado y adaptación a distintos lugares, se pretende llevar a cabo una reducción de su tamaño en la medida de lo posible.

1.3 Estructura

En el presente documento se realizará una breve introducción a la estructura del trabajo de fin de grado. En el capítulo 2, se desarrollara el estado del arte y las tecnologías asociadas. Este apartado incluye la definición de biometría y de métodos biométricos. A su vez, se expondrán las diferentes modalidades de identificación biométricas y su funcionamiento. También se abordará el lenguaje de programación empleado (C Sharp), una introducción a la automatización y el desarrollo de .NET.

En el capítulo 3 se explicará cómo se ha llegado al diseño de la solución. Partiendo del planteamiento del problema: máquina de captura de imágenes vasculares, que debe ser manejada por el usuario (no entrenado), con los consiguientes problemas que eso acarrea. Posteriormente se planteará una solución para solucionar este problema; esta solución para por la automatización de la máquina a la hora de capturar las imágenes. Y para finalizar este apartado, se expondrá el diseño final de la solución para automatizar la máquina, es decir, los diferentes métodos que se podrán llevar a cabo para la automatización.

Tras esto, en el capítulo 4 se expondrá de forma precisa el desarrollo de dicha solución, que costa de una introducción al funcionamiento de la cámara, el desarrollo del código y la interfaz gráfica para la automatización y la disminución del tamaño de la máquina. En este apartado también se recogerán los problemas encontrados y la forma de resolverlos,

En el siguiente capítulo (el 5) aparecerán las pruebas realizadas para asegurar su correcto funcionamiento.

Por último, en el sexto capítulo, se encontrará una breve conclusión que incluye el peso que puede tener este proyecto en líneas de trabajo futuras. Así como unas referencias a los artículos, libros y páginas webs usadas para llevar a cabo este trabajo de fin de grado. Para concluir el TFG se incluirá un presupuesto detallado del coste de la máquina y su automatización.

2 Estado del Arte

En este capítulo se introducirá al lector en la tecnología. Se realizará un breve resumen de la historia de la biometría y se hablará de los diversos sistemas biométricos existentes. A continuación, se realizará una introducción a la automatización, desde sus inicios hasta la actualidad.

2.1 Biometría e identificación biométrica

Según el Diccionario de la Real Academia Española, se define biometría como «Estudio mensurativo o estadístico de los fenómenos o procesos biológicos». Esta definición se hace más específica cuando se utiliza el término de Biometría dentro del campo de la Identificación de Personas.

La identificación biométrica es la ciencia que clasifica, registra e identifica a las personas mediante sus características biológicas, físicas y/o de comportamiento. Se cree que los antiguos egipcios ya usaban métodos de identificación biométricos, concretamente la firma, para confirmar la identidad de las personas, y se sabe con seguridad que los comerciantes chinos distinguían a los niños mediante impresiones en papel de la huella de la palma de la mano ya desde el siglo XIV.

En Europa, la identificación biométrica comenzó a utilizarse en el siglo XIX con fines policiales. Originalmente la comparación se hacía de forma tosca, confiando en la memoria visual; básicamente consistía en reconocer o no al sospechoso, basándose en las características físicas que el policía encargado consiguiera recordar, hasta que en 1883 un miembro de la policía parisina, Alphonse Bertillon, desarrolló el sistema antropométrico de clasificación.

Este método funcionaba midiendo de forma precisa ciertas longitudes y anchuras del cuerpo y la cabeza del individuo, y registrando marcas personales indelebles como tatuajes, manchas o cicatrices. Fue usado ampliamente hasta que se descubrieron dos personas diferentes con el mismo conjunto de medidas. Este descubrimiento relegó la antropometría a la categoría de pseudociencia, y obligó a buscar una alternativa para la identificación policial de sospechosos.

Fue en el año 1892 cuando Francis Galton, un antropólogo inglés, realizó las primeras pruebas satisfactorias de identificación de criminales mediante huella dactilar basándose en los estudios de Henry Faulds sobre la unicidad y estabilidad de las mismas durante la vida de un individuo. Este nuevo método se impuso rápidamente, convirtiéndose en poco tiempo en la forma oficial de identificación en diversos países. Es tal su eficacia que en la actualidad, aunque los métodos de archivo y comparación han mejorado notablemente, aún se usan los cuatro rasgos seleccionados por Galton para diferenciar dos huellas entre sí: arcos, presillas internas, presillas externas y verticilos.

Aunque la huella dactilar sigue siendo el más empleado, a lo largo del siglo XX se han propuesto y estudiado una gran cantidad de métodos biométricos nuevos. En 1936 el patrón de iris fue sugerido como método de identificación por Frank Burch, idea que fue desarrollada finalmente entre 1985 y 1994 por Leonard Flom y Aran Safir, usando los algoritmos de reconocimiento de John Daugman. En el 2006 se evaluaron los mejores algoritmos de reconocimiento facial, resultando algunos tan exactos que distinguían entre gemelos idénticos. También se han propuesto y desarrollado métodos de identificación que emplean la vascularización, la firma, la voz, el modo de andar [2]…

Prácticamente cualquier característica biofísica tiene su propio estudio biométrico, con mayor o menor eficacia. La eficacia de un método biométrico depende de su tasa de falso positivo o FAR (False Acceptance Rate), de su tasa de falso negativo o FRR (False Rejection Rate) y de la de fallo de alistamiento o FMR (False Match Rate). En los dispositivos reales se cruzan ambas tasas y el sistema se ajusta según el punto de cruce, conocido como CER (Cross-over Error Rate), como puede verse en la figura 2. Cuanto más bajo es el CER, más exactitud proporciona el sistema [3].

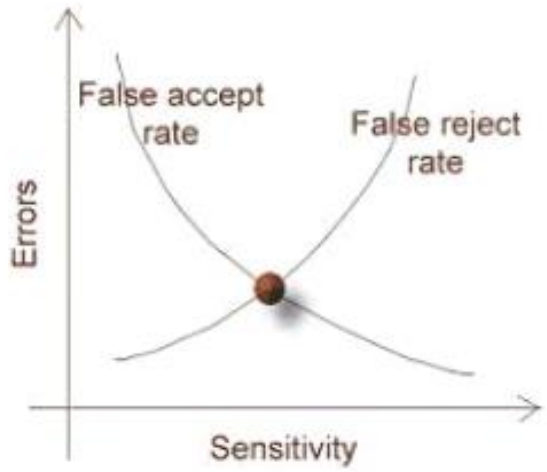

Fig. 2 – Cruce de FAR y FRR. Localización del CER [4]

2.2 Etapas en un sistema de identificación biométrica

Las técnicas de identificación biométrica son muy diversas, ya que cualquier elemento significativo de una persona es potencialmente utilizable como elemento de identificación biométrica. Las distintas técnicas que existen serán tratadas en el próximo apartado. Sin embargo, incluso con la diversidad de técnicas existentes, a la hora de desarrollar un sistema de identificación biométrica, se mantiene un esquema totalmente independiente de la técnica empleada. Los sistemas, tal y como se puede ver en la Figura 3, se basan en dos fases totalmente diferenciadas [5]:

1) **Reclutamiento:** en esta fase, se toma una serie de muestras del usuario, y se procesan, para posteriormente extraer un patrón, el cual se almacenará y será el conjunto de datos que caracterizará a ese usuario. Si se captura más de una muestra, el patrón suele ser el resultado de una media de las características obtenidas. Este proceso se hace de forma supervisada, es decir, existe una persona encargada de controlar cómo se produce la captura de los datos, así como de asegurar la identidad de la persona que se está reclutando en el sistema. Además, se aprovecha esta fase para enseñar al usuario cómo funciona el sistema y aclararle todas las dudas que pudiera tener.
2) **Utilización:** una vez que se tiene almacenado el patrón del usuario, éste puede utilizar el sistema con normalidad, y sus características son comparadas con el patrón almacenado, determinando el éxito o fracaso de esa comparación.

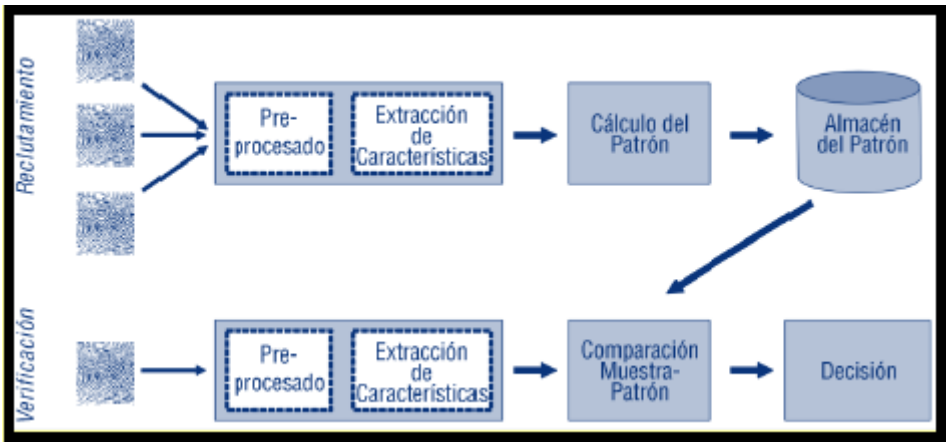

Fig. 3 – Etapas en un sistema de identificación biométrica [5]

Pero como se observa en la Figura 3, cada una de las fases mencionadas, está basada en una serie de bloques que hacen que las características biológicas o de comportamiento del individuo acaben siendo un elemento que lo identifique. Estas fases son [5]:

- **Captura:** se toman los datos biofísicos o de comportamiento del sujeto. La toma de los datos depende, evidentemente, de la técnica biométrica empleada, pero también se pueden encontrar muchas variaciones para la misma técnica biométrica. Por ejemplo, la huella dactilar puede ser obtenida por cámara de vídeo, ultrasonidos, efecto capacitivo sobre un semiconductor o exploración por láser.
- **Pre-procesado:** en este bloque se adecuan los datos capturados para facilitar el tratamiento que tiene que realizar el siguiente bloque. Este bloque se encarga, dependiendo de la técnica, de tareas como: reconocer el inicio de una frase y medir el ruido de fondo, hacer una extracción de bordes de la imagen capturada, localizar la muestra, rotarla y ampliarla (o reducirla), para que se encuentre entre los márgenes que reconoce el algoritmo siguiente, etc.
- **Extracción de Características:** se puede considerar el bloque más significativo de la técnica a utilizar. Es el bloque en el que se fundamenta la capacidad del sistema de distinguir entre sujetos. Sin embargo, debido a distintas aproximaciones al problema, este bloque puede seguir orientaciones muy diversas, e incluso contradictorias, para la misma técnica, creándose distintos métodos dentro de una misma técnica. Por otro lado, en algunas ocasiones, el desconocimiento sobre las características que se deben extraer, lleva a utilizar técnicas basadas en Redes Neuronales, que mediante entrenamiento de las mismas, se intentan adecuar a los resultados esperados.
- **Comparación:** una vez extraídas las características de la muestra capturada, se han de comparar éstas con las previamente almacenadas, es decir, el patrón . Lo más importante que hay que dejar claro cuando se habla de este bloque, es que no se trata de una comparación binaria (o de igualdad), sino que la variación de las muestras, por variaciones en la captura o leve variación de las características de sujeto, hacen que la comparación dé como resultado una probabilidad de semejanza. Por tanto, para determinar el éxito o fracaso de la comparación, habrá que determinar un umbral en esa probabilidad.

  Sobre los conceptos expuestos cabe hacer un par de puntualizaciones. La primera de ellas tiene que ver con la elección del umbral , ya que si éste se incrementa, hará que el sistema se relaje y permita una mayor probabilidad de accesos por parte de personas no autorizadas (FAR), mientras que si se disminuye, el sistema se volverá muy restrictivo, aumentando la probabilidad de rechazo de personas autorizadas (FRR). Por lo tanto, la elección del umbral dependerá del grado de seguridad, y amigabilidad hacia el usuario, que se le quiera dar al sistema.

  Por otro lado, el modo en el que se hace el reclutamiento no es tampoco trivial. En algunas técnicas basta una única toma de los datos, mientras que en otras puede ser necesario tomar varias muestras y en distintas sesiones (días o semanas), tal y como ocurre, por ejemplo, en los sistemas basados en voz. A todo esto habrá que añadir que si el reclutamiento resulta muy pesado, los usuarios del sistema tenderán a rechazar el sistema de identificación, por lo que habrá que buscar una solución de compromiso entre la comodidad del usuario, y la obtención de un patrón óptimo.
- **Reconocimiento:** se basa en identificar a un usuario dentro de todos los usuarios que ya se encuentran en el sistema. Por lo tanto, se comparan las características extraídas con los patrones de todos los usuarios reclutados por el sistema. Este esquema de

funcionamiento, necesario para muchas aplicaciones, tiene como inconvenientes la necesidad de una base de datos de patrones (con los requisitos oportunos de capacidad de almacenamiento y seguridad de los datos) y la existencia de una red de comunicaciones, siempre on-line, que comunique los puestos de identificación con la base de datos. El resultado de la comparación puede ser: siempre positivo (es decir, se identifica siempre con el usuario que ha dado una probabilidad más alta), o puede indicar rechazos (si el usuario con la mayor probabilidad no supera un determinado umbral).

- **Autenticación:** también llamado verificación. Trata de responder a la pregunta: ¿es este sujeto la persona que dice ser? El sistema se encarga, entonces, de comparar las características extraídas, con el patrón del usuario indicado. Si la comparación supera un determinado umbral de parecido, se considera que el usuario es el indicado, rechazando la comparación en caso contrario. El patrón del usuario puede estar almacenado en una base de datos, tal y como se hace en los sistemas de reconocimiento, o, si el patrón es suficientemente pequeño, en un sistema portátil de información como puede ser una tarjeta. En este último caso no son necesarias ni la base de datos ni la red de comunicaciones de los sistemas de reconocimiento.

2.3 Modalidades biométricas

Prácticamente cualquier característica biológica o de comportamiento de una persona puede ser usada para su identificación. Normalmente las biológicas se consideran más exactas, ya que cualquier característica de comportamiento es potencialmente imitable, y por tanto, podría no garantizar la identificación con un nivel de error lo suficientemente bajo. Algunos de los parámetros a tener en cuenta al analizar la eficacia de una técnica biométrica son la universalidad, es decir, si las características se pueden extraer de cualquier usuario, la unicidad, que es la probabilidad de que no existan dos personas con las mismas características, la facilidad de captura, el rendimiento, la aceptación por los usuarios, la estabilidad de la muestra, es decir, si permanece inalterable durante la vida del usuario, la robustez frente al fraude o el coste [6].

A continuación se presentan algunos de los métodos que se estudian actualmente para la identificación biométrica, incluyendo algunos cuya implantación es ya una realidad, haciendo un breve repaso de sus pros y sus contras. Es conveniente empezar esta presentación haciendo hincapié en el hecho de que no existe la técnica biométrica perfecta: una que proporcione una fiabilidad casi absoluta pero que exija un tiempo de proceso grande puede no ser la más adecuada si el tiempo es un factor limitante.

Cada técnica tiene sus ambientes de aplicación destacados, y es necesario un estudio previo del entorno antes de elegir una de ellas [5]:

1) **Huella dactilar:** Es la que tiene una aceptación mayor. Se usa desde el siglo XIX y existen numerosos estudios que prueban tanto la estabilidad de la huella a lo largo de

la vida como su unicidad. La facilidad de extracción y el escaso coste de los dispositivos que emplean esta técnica ha permitido que siga compitiendo con otras biométricamente más exactas pero más caras, como el iris, o de extracción más complicada, como el ADN.

Como punto en contra, su uso por parte de la policía para identificar criminales ha hecho que la población asocie este método con actos delictivos, por lo que algunas empresas prefieren invertir un poco más en una técnica más cara que no tenga implícita esa imagen policial.

2) **ADN:** Es la única técnica conocida que ha conseguido garantizar un cien por cien de éxito. Mediante comparación de ADN la identidad de una persona queda confirmada sin posibilidad de fallo, ya que no hay dos personas que compartan el mismo. Como contrapartida, existe una gran dificultad a la hora de procesar los resultados en tiempo real, necesitándose mucho más tiempo para realizar la comparación que con otros sistemas. Además, las técnicas de extracción (saliva, sangre, sudor) son muy invasivas y el usuario tiende a rechazarlas.

3) **Iris:** La mayoría de los sistemas que emplean esta técnica parten de los algoritmos creados por John Daugman en 1993. Tiene unos resultados excelentes y de gran fortaleza, y hay estudios que confirman la inalterabilidad del iris a lo largo de la vida, así como su unicidad, siendo esta última muy superior a la de la huella dactilar. Además, la técnica de extracción no es invasiva ni tiene connotaciones policiales negativas. La única razón por la que esta tecnología no es la más usada a nivel mundial es el elevado coste de los equipos.

4) **Voz:** Es una de las técnicas más estudiadas, ya que la creación de un sistema fiable de reconocimiento biométrico basado en la voz haría posible el reconocimiento de una persona de forma remota mediante algo tan sencillo, accesible y barato como un teléfono. Sin embargo, y aunque existen algunos sistemas que responden de forma bastante correcta, la voz está sujeta a múltiples variaciones, tales como la edad, las enfermedades o simples cambios en el estado de ánimo. El umbral del sistema debe rebajarse de forma notable para poder asimilar estos cambios, lo que lo convierte en menos fiable de lo preciso. Además, es muy propenso al fraude, ya que al no necesitar presencia física basta con una grabación para burlarlo. Para solucionar este problema, algunos sistemas obligan al usuario a repetir una locución aleatoria determinada para identificarse.

5) **Retina:** La geometría vascular de la retina es uno de los métodos más seguros de identificación. La unicidad presentada supera incluso la del iris, y no necesita métodos añadidos para detectar si el usuario está vivo. Sin embargo, para poder tomar la muestra es necesario usar láser y la mayoría de gente siente rechazo hacia ello.

6) **Reconocimiento facial:** Es una de las tecnologías que más rápidamente está progresando. A mediados de 2006 se celebró la Face Recognition Grand Challenge,

un congreso donde se presentaron nuevos algoritmos de reconocimiento facial. Se descubrió que proporcionaban una fiabilidad 10 veces superior a los mejores de 2002, y 100 veces superior a los de 1995. Como ventaja, el método de captura es muy sencillo y barato, y métodos como la hiperresolución del rostro hacen que las fotos en baja calidad no sean ya un problema. La captura es tan sencilla que en ocasiones el usuario no es consciente de que está siendo verificado, lo cual puede llegar a ser potencialmente peligroso si se usa con fines poco éticos. Como contra, es un método muy sensible a variaciones muy habituales como gafas, peinado o vello facial.

7) **Geometría de la mano:** Es el método biométrico más rápido. En un segundo es capaz de confirmar la identidad de una persona. Además, los modelos más modernos van actualizando su base de datos con cada introducción, variando algunos detalles del usuario que pueden cambiar de una autenticación a otra (por ejemplo, adelgazamiento o cicatrización de heridas). Es un método que está especialmente indicado para ambientes con una gran afluencia de individuos, donde la rapidez es un factor crítico y la seguridad puede ser más laxa, ya que la tasa de error es muy superior a otros sistemas.

8) **Firma:** Se utiliza desde antes que la huella dactilar y siempre se ha cuestionado su fiabilidad, ya que la falsificación es relativamente fácil. Desde hace unos años los métodos no solo comprueban la firma sino también la dinámica de la misma: rapidez, paradas, presión sobre el papel… De este modo se alcanza una fiabilidad similar a la de un método biométrico, pero incluso de este modo sus críticos afirman que es potencialmente falsificable con el entrenamiento adecuado.

9) **Forma de andar:** Su estudio se encuentra en desarrollo. De momento su fiabilidad es mínima, y presenta el mismo problema que el reconocimiento facial, el usuario podría no saber que está siendo analizado. De cualquier modo, como método basado en el comportamiento, no hay expectativas de que supere a la firma a corto o medio plazo.

10) **Vascular:** Consiste en usar el patrón de las venas de la mano o del dedo como método de identificación. Su uso está en estudio para ambientes donde la identificación digital externa (huella) sea difícil de extraer, ya sea por erosión de la misma o por suciedad.

    Otra ventaja de la identificación vascular, es que los otros métodos, como el de reconocimiento de iris, necesitan de la luz visible (longitud de onda entre 400 y 750 nanómetros), mientras que este sistema de reconocimiento utiliza luz infrarroja [4] (de longitud de onda desde 800 a 950 nanómetros).

Técnica	Ventajas	Inconvenientes
Voz 	- Muy bajo coste - En algunas aplicaciones puede resultar inapreciable al usuario (p.e., servicios telefónicos)	- Rendimiento bajo - Se está estudiando el aumentar la unicidad y estabilidad
Huella 	- Muy estudiado/desarrollado - Unicidad, estabilidad y rendimiento altos - Reconocimiento legal - Medio coste	- Connotaciones «policiales» para el usuario - Detección de dedo vivo depende de pruebas colaterales a la captura
Iris 	- Unicidad mayor que huella - Gran estabilidad por protección de la córnea - FAR prácticamente nula - Fácil detección de ojo vivo	- Alto coste - Inicialmente incómodo para el usuario
Mano 	- Fácil uso y gran aceptación por el usuario - Medio coste - Bajo coste computacional - Sin connotación «policial»	- Unicidad y estabilidad no probadas en grandes poblaciones - Detección de mano viva depende de pruebas colaterales
Rostro 	- Cómodo, e incluso inapreciable, para el usuario - Medio coste	- Sensible a cambios del sujeto (barba, gafas, pelo, ...) - Todavía en investigación y desarrollo

Tabla 1 – Comparativa Ventajas/Desventajas de las técnicas más usadas[5]

2.4 Automatización

Automatización (del griego antiguo *auto*: guiado por uno mismo) es el uso de sistemas o elementos computarizados y electromecánicos para controlar maquinarias y/o procesos.

La automatización como una disciplina de la ingeniería que es más amplia que un mero sistema de control, abarca la instrumentación industrial, que incluye los sensores, los transmisores de campo, los sistemas de control y supervisión, los sistemas de transmisión y recolección de datos y las aplicaciones de software en tiempo real para supervisar y controlar las operaciones de máquinas, plantas o procesos.

Las primeras máquinas simples sustituían una forma de esfuerzo en otra forma que fueran manejadas por el ser humano, tal como levantar un peso pesado con sistema de poleas

o con una palanca. Posteriormente las máquinas fueron capaces de sustituir formas naturales de energía renovable, tales como el viento, mareas, o un flujo de agua por energía humana.

Todavía después, algunas formas de automatización fueron controladas por mecanismos de relojería o dispositivos similares utilizando algunas formas de fuentes de energía artificiales como algún tipo de resorte, un flujo canalizado de agua o vapor para producir acciones simples y repetitivas, tales como figuras en movimiento, creación de música, o juegos. Dichos dispositivos caracterizaban a figuras humanas, fueron conocidos como autómatas y datan posiblemente desde 300 AC.

En 1801, la patente de un telar automático utilizando tarjetas perforadas fue dada a Joseph Marie Jacquard, quien revolucionó la industria del textil.

La parte más visible de la automatización actual puede ser la robótica industrial. Algunas ventajas son repetitividad, control de calidad más estrecho, mayor eficiencia, integración con sistemas empresariales, incremento de productividad y reducción de trabajo.

Para mediados del siglo XX, la automatización había existido por muchos años en una escala pequeña, utilizando mecanismos simples para automatizar tareas sencillas de manufactura. Sin embargo el concepto solamente llegó a ser realmente práctico con la adición (y evolución) de las computadoras digitales, cuya flexibilidad permitió manejar cualquier clase de tarea. Las computadoras digitales con la combinación requerida de velocidad, poder de cómputo, precio y tamaño empezaron a aparecer en la década de 1960s. Antes de ese tiempo, las computadoras eran exclusivamente computadoras analógicas y computadoras híbridas. Desde entonces las computadoras digitales tomaron el control de la mayoría de las tareas simples, repetitivas, tareas parcialmente especializadas y especializadas, con algunas excepciones notables en la producción e inspección de alimentos.

En la actualidad, convivimos de manera activa con la automatización. La mayoría de las tareas a nuestro alrededor se encuentran parcial o totalmente automatizadas.

La automatización cobra especial interés para el sector industrial (debido a la repetitividad de sus tareas). La automatización para la industrias supone ahorro, mayor velocidad de proceso, aumento de la seguridad… Por estos motivos la automatización industrial es la más desarrollada.

Existe un concepto fundamental y muy actual en torno a la automatización industrial (que también puede aplicarse a la automatización de las máquinas): el DCS (Sistemas de Control Distribuido). Un Sistema de Control Distribuido está formado por varios niveles de automatización que van desde un mínimo de 3 hasta 5. Los mismos de denominan: nivel de campo (donde se encuentran los sensores y actuadores), nivel de control (donde se encuentran los PLCs o las Estaciones de Automatización), nivel de supervisión (donde se encuentran las Estaciones de Operación y los Servidores de Proceso), nivel MES (donde se encuentran PCs con software especializado para la distribución de toda la información de planta así como la generación de reportes) y el nivel ERP (donde se encuentran igualmente PCs con software especializados para la planificación y administración de la producción de toda la industria o empresa).

2.5 Tecnología utilizada C#

El lenguaje de programación elegido para el desarrollo del proyecto es C Sharp. C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

El nombre C Sharp fue inspirado por la notación musical, donde '#' (sostenido, en inglés *sharp*) indica que la nota (*C* es la nota do en inglés) es un semitono más alta, sugiriendo que C# es superior a C/C++. Además, el signo '#' se compone de cuatro signos '+' pegados.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco Mono - DotGNU, el cual genera programas para distintas plataformas como Windows, Unix, Android, iOS, Windows Phone, Mac OS y GNU/Linux.

2.6 .NET

C Sharp forma parte de la plataforma .NET por lo que es importante dedicar una sección de este capítulo para su mejor comprensión.

.NET es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Oracle Corporation y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones (o como la misma plataforma las denomina, soluciones) permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

La plataforma .NET de Microsoft es un componente de software que puede ser añadido al sistema operativo Windows. Provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma. Esta solución es el

producto principal en la oferta de Microsoft, y pretende ser utilizada por la mayoría de las aplicaciones creadas para la plataforma Windows.

.NET Framework se incluye en Windows Server 2008, Windows Vista y Windows 7. De igual manera, la versión actual de dicho componente puede ser instalada en Windows XP, y en la familia de sistemas operativos Windows Server 2003. Una versión "reducida" de .NET Framework está disponible para la plataforma Windows Mobile, incluyendo teléfonos inteligentes.

La norma (incluido en ECMA-335, ISO/IEC 23271) que define el conjunto de funciones que debe implementar la biblioteca de clases base (BCL por sus siglas en inglés, tal vez el más importante de los componentes de la plataforma), define un conjunto funcional mínimo que debe implementarse para que el marco de trabajo sea soportado por un sistema operativo. Aunque Microsoft implementó esta norma para su sistema operativo Windows, la publicación de la norma abre la posibilidad de que sea implementada para cualquier otro sistema operativo existente o futuro, permitiendo que las aplicaciones corran sobre la plataforma independientemente del sistema operativo para el cual haya sido implementada [8].

2.6.1 Historia

Durante el desarrollo de la plataforma .NET, las bibliotecas de clases fueron escritas originalmente usando un sistema de código gestionado llamado Simple Managed C (SMC).

En enero de 1999, Anders Hejlsberg formó un equipo con la misión de desarrollar un nuevo lenguaje de programación llamado Cool (Lenguaje C orientado a objetos). Este nombre tuvo que ser cambiado debido a problemas de marca, pasando a llamarse C#. La biblioteca de clases de la plataforma .NET fue migrada entonces al nuevo lenguaje.

Hejlsberg lideró el proyecto de desarrollo de C#. Anteriormente, ya había participado en el desarrollo de otros lenguajes como Turbo Pascal, J++ y Embarcadero Delphi.

2.6.2 Objetivos de .NET

El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes [9]:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.

- Ofrecer un entorno de ejecución de código que fomente la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.
- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

2.6.3 Componentes

.NET Framework contiene dos componentes principales: Common Language Runtime y la biblioteca de clases de .NET Framework. Common Language Runtime es el fundamento de .NET Framework. El motor en tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la interacción remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que fomentan su seguridad y solidez. De hecho, el concepto de administración de código es un principio básico del motor en tiempo de ejecución. El código destinado al motor en tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado [9].

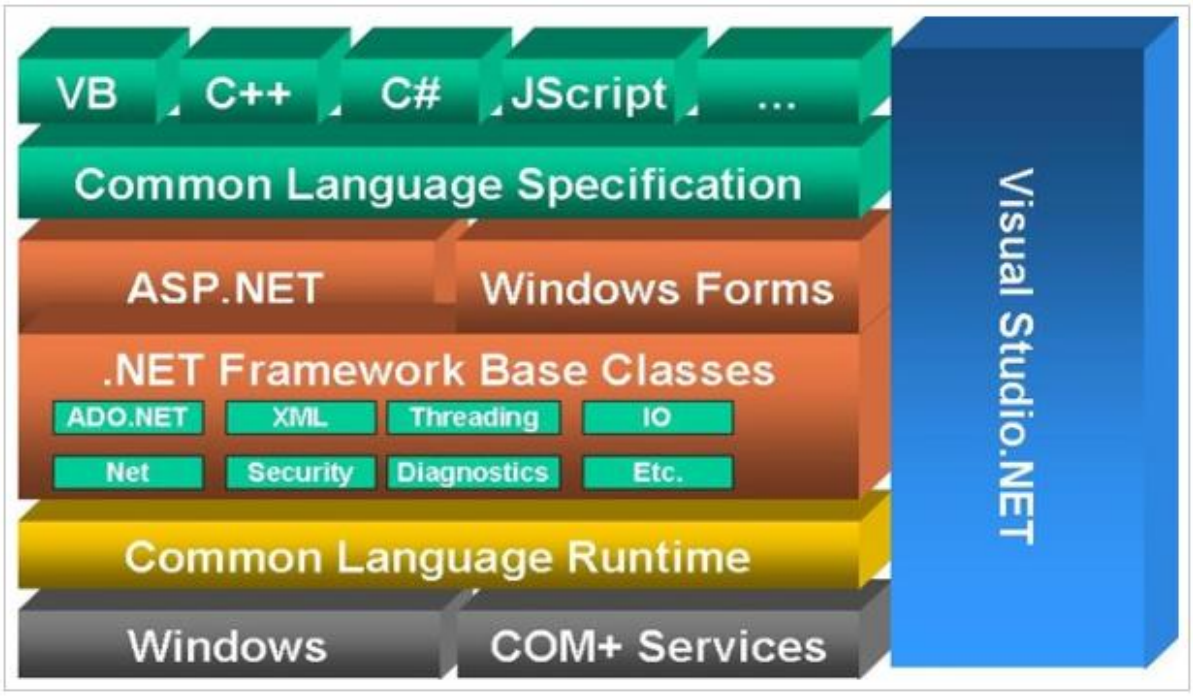

Fig. 4 – Estructura interna de Common Language Runtime [10]

La biblioteca de clases, el otro componente principal de .NET Framework, es una completa colección orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta las aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como los formularios Web Forms y los servicios Web XML.

.NET Framework puede alojarse en componentes no administrados que cargan Common Language Runtime en sus procesos e inician la ejecución de código administrado, con lo que se crea un entorno de software en el que se pueden utilizar características administradas y no administradas. En .NET Framework no sólo se ofrecen varios hosts de motor en tiempo de ejecución, sino que también se admite el desarrollo de estos hosts por parte de terceros.

Por ejemplo, ASP.NET aloja el motor en tiempo de ejecución para proporcionar un entorno de servidor escalable para el código administrado. ASP.NET trabaja directamente con el motor en tiempo de ejecución para habilitar aplicaciones de ASP.NET y servicios Web XML, que se tratan más adelante en este tema.

Internet Explorer es un ejemplo de aplicación no administrada que aloja el motor en tiempo de ejecución. Al usar Internet Explorer para alojar el motor en tiempo de ejecución, puede incrustar componentes administrados o controles de Windows Forms en documentos HTML. Al alojar el motor en tiempo de ejecución de esta manera se hace posible el uso de

código móvil administrado (similar a los controles de Microsoft® ActiveX®), pero con mejoras significativas que sólo el código administrado puede ofrecer, como la ejecución con confianza parcial y el almacenamiento aislado de archivos.

En la siguiente figura se muestra la relación de Common Language Runtime y la biblioteca de clases con las aplicaciones y el sistema en su conjunto. En la ilustración se representa igualmente cómo funciona el código administrado dentro de una arquitectura mayor.

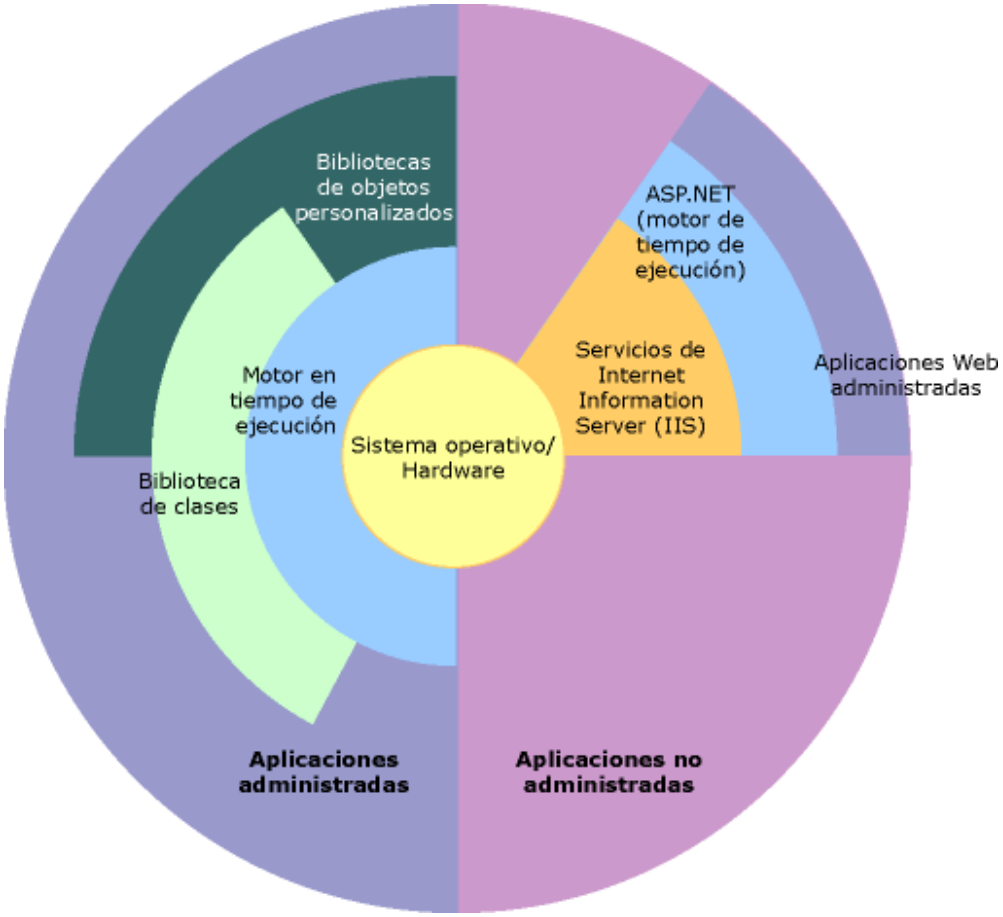

Fig. 5 – Componentes principales de .NET Framework [9]

2.6.4 Características

Es el encargado de proveer al código administrado, es decir, un entorno que provee servicios automáticos al código que se ejecuta. Los servicios son variados [9]:

- **Cargador de clases:** permite cargar en memoria las clases.

- **Compilador MSIL a nativo:** transforma código intermedio de alto nivel independiente del hardware que lo ejecuta a código de máquina propio del dispositivo que lo ejecuta.
- **Administrador de código:** coordina toda la operación de los distintos subsistemas del Common Language Runtime.
- **Recolector de basura:** elimina de memoria objetos no utilizados automáticamente.
- **Motor de seguridad:** administra la seguridad del código que se ejecuta.
- **Motor de depuración:** permite hacer un seguimiento de la ejecución del código aun cuando se utilicen lenguajes distintos.
- **Verificador de tipos:** controla que las variables de la aplicación usen el área de memoria que tienen asignado.
- **Administrador de excepciones:** maneja los errores que se producen durante la ejecución del código.
- **Soporte de multiproceso (hilos):** permite desarrollar aplicaciones que ejecuten código en forma paralela.
- **Empaquetador de COM:** coordina la comunicación con los componentes COM para que puedan ser usados por el .NET Framework.
- **Biblioteca de Clases Base:** que incluye soporte para muchas funcionalidades comunes en las aplicaciones.

2.6.5 Versiones

Las versiones que se han llevado a cabo desde su lanzamiento son muchas, como se puede apreciar en la siguiente tabla. Y como se puede ver, .NET ha sufrido una gran evolución desde su salida al mercado.

Nombre de la Versión	Número de Versión	Lanzamiento
Pre-beta	1.0.00000.00000	Julio 2000
1.0 Beta 1	1.0.0000.0	Noviembre 2000
1.0 Beta 2	1.0.2914.0	20-06-2001
1.0 RTM	1.0.3705.0	05-01-2002
1.0 SP1	1.0.3705.209	19-03-2002
1.0 SP2	1.0.3705.288	07-08-2002
1.0 SP3	1.0.3705.6018	31-08-2004
1.1 RTM	1.1.4322.573	01-04-2003
1.1 SP1	1.1.4322.2032	30-08-2004

1.1 SP1 (W2k3)	1.1.4322.2300	30-03-2005
2.0 RTM	2.0.50727.42	07-11-2005
2.0 RTM (Vista)	2.0.50727.312	30-01-2007
2.0 (KB928365)	2.0.50727.832	10-07-2007
2.0 SP1	2.0.50727.1433	19-11-2007
2.0 SP2	2.2.30729	16-01-2009
3.0 RTM	3.0.4506.30	06-11-2006
3.0 RTM (Vista)	3.0.4506.26	30-01-2007
3.0 SP1	3.0.4506.648	19-11-2007
3.0 SP2	3.2.30729	19-11-2007
3.5 RTM	3.5.21022.8	19-11-2007
3.5 SP1	3.5.30729.01	11-08-2008
4.0 RTM	4.0.30319.1	12-04-2010
4.5 RTM	4.5.50709.17929	15-08-2012

Tabla 2 – Versiones .NET [8]

2.6.6 El futuro de .NET

A largo plazo Microsoft pretende reemplazar el API Win32 o Windows API con la plataforma .NET. Esto debido a que el API Win32 o Windows API fue desarrollada sobre la marcha, careciendo de documentación detallada, uniformidad y cohesión entre sus distintos componentes, provocando múltiples problemas en el desarrollo de aplicaciones para el sistema operativo Windows. La plataforma .NET pretende solventar la mayoría de estos problemas proveyendo un conjunto único y expandible con facilidad, de bloques interconectados, diseñados de forma uniforme y bien documentados, que permitan a los desarrolladores tener a mano todo lo que necesitan para producir aplicaciones sólidas.

3 Diseño de la solución

En este apartado se va a analizar detalladamente el problema planteado, considerando las diversas soluciones que existen al mismo. Se indicará la solución escogida, haciendo un razonamiento de sus ventajas e inconvenientes frente a otras opciones de resolución

3.1 Planteamiento del problema

La finalidad de este trabajo es la automatización de un sensor de captura de imágenes vasculares para la identificación biométrica del usuario.

La máquina en cuestión está formada por una estructura de madera en alberga una cámara (en blanco y negro). La estructura es de madera debido a su bajo coste y peso reducido. También es fácil de trabajar, lo que es especialmente importante para este tipo de estructuras que necesitan albergar distintos objetos.

Como puede observarse en la figura 6, la estructura está dividida en 3 "pisos". El inferior consta de la cámara y una sujeción para fijar la cámara, y que esta no se mueva a la hora de ser usada. En el segundo piso se encuentra el objetivo de la cámara con sus parámetros analógicos (apertura de la lente, distancia focal y zoom). En este piso se encuentran tres de las cuatro paredes tapadas de modo que entre la luz visible en la menor medida posible. Y en el piso superior se encuentra el DOM o bóveda de homogenización.

Fig. 6 – Estructura de madera

El sistema de captura d imágenes está basado en una cámara CCD de bajo coste (Imaging Source DM 21BU054) y con un objetivo (EYSEO TV8570 1/3”). La cámara CCD toma imágenes de 640 x 480 pixel con 256 niveles en la escala de grises. La cámara puede observarse en la figura 7.

Fig. 7 – Cámara adaptada en la estructura

El sensor también consta de unos diodos LED, como se muestra en la figura 8 (que emiten luz infrarroja). Los diodos LED forman parte del sistema de iluminación del sensor. Estos van adheridos al DOM. Se encuentran situados en la parte de la circunferencia del DOM y emiten su luz hacia la bóveda.

Fig. 8 – Tira de LED de infrarrojos

En la figura 9, aparece el filtro para el espectro de luz visible (que hace que a la cámara solo llegue la luz infrarroja de los LED), compuesto por dos filtros de infrarrojo, modelos B+W 52 092 y B+W 52 093.

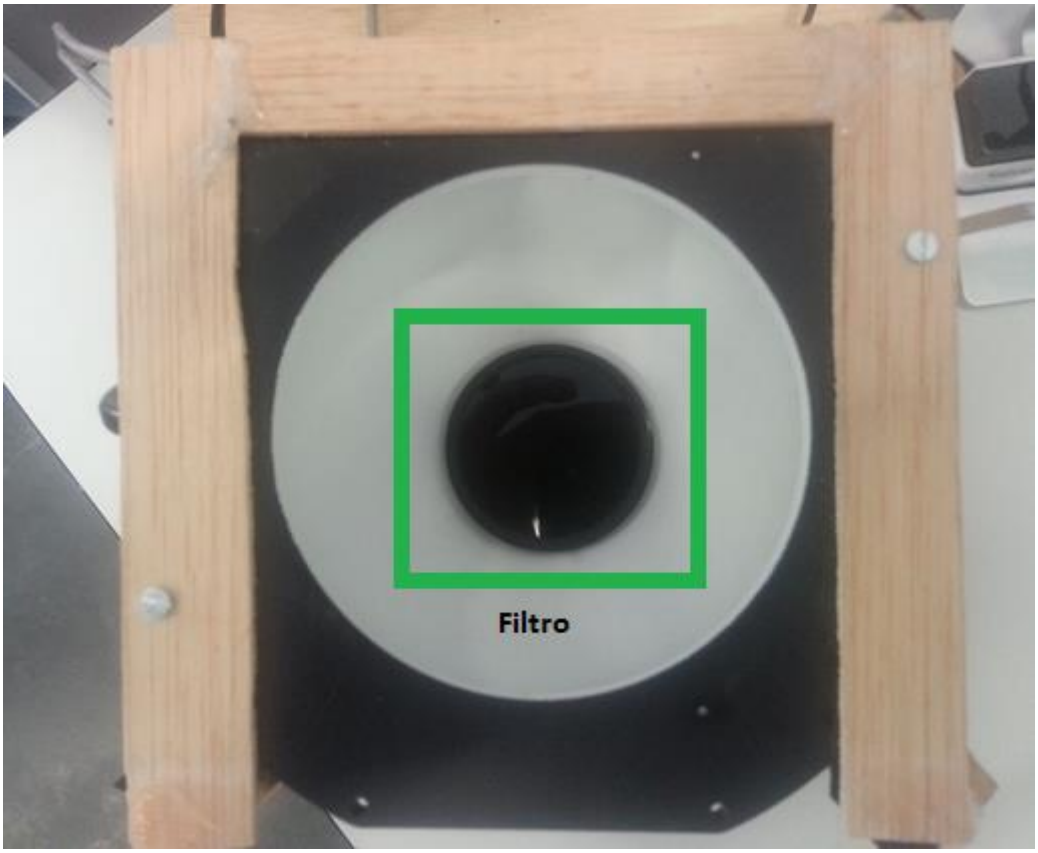

Fig. 9 – Filtro de infrarrojos (vista de planta de la máquina)

La parte de iluminación de la máquina de captura de imágenes vasculares también incluye una bóveda cerámica (que homogeniza la luz emitida por los LED). Esta cúpula, también denominada DOM (modelo DOM 1410, DCM System, 880 nm, 24V DC y 1585 mA) aunque aumenta el tamaño del sensor en gran medida, es necesaria para reflejar la luz de los LED hacia la palma de la mano de la siguiente manera:

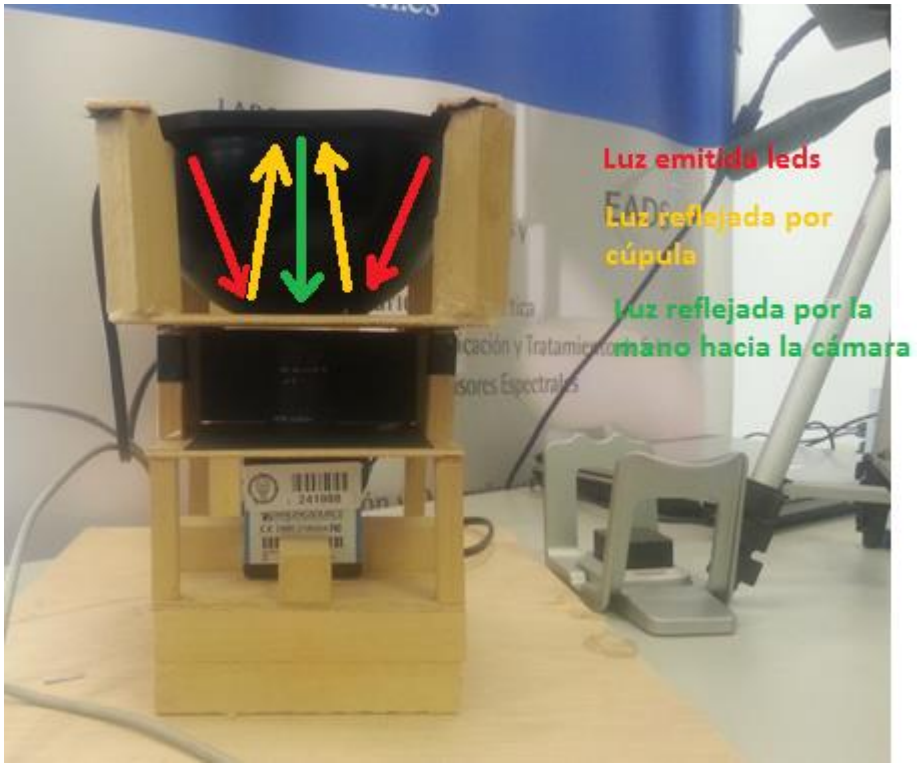

Fig. 10 – Funcionamiento de la bóveda de homogenización

Como se puede observar en la imagen 11, el DOM es una semiesfera rodeada (en la parte de la circunferencia) por 112 LED de infrarrojos de 880 nanómetros de longitud de onda y tensión máxima de 24 voltios.

Fig. 11 – Cúpula de homogenización (perfil)

Viendo el DOM desde arriba (figura 12), se observa que el interior es de color blanco. Esto hace que la luz emitida por los LED tenga una mayor reflexión y una mejor homogenización de la luz.

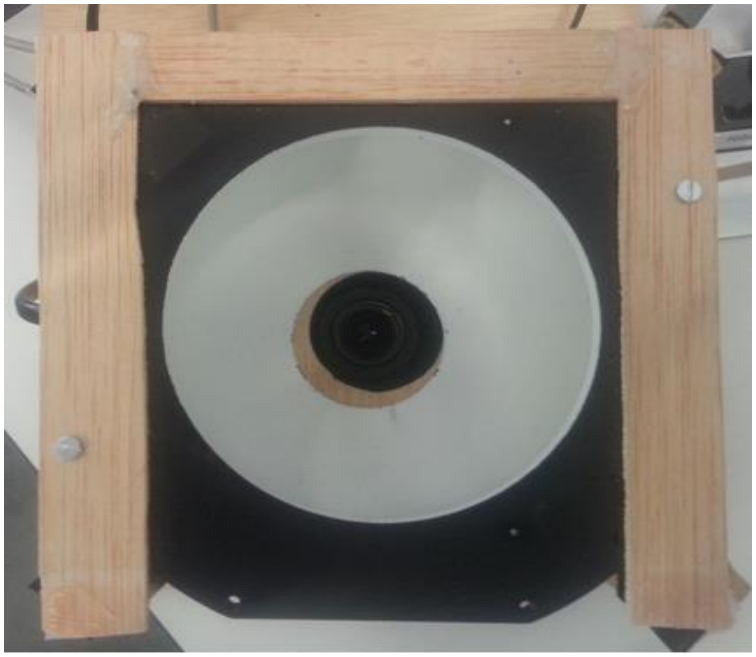

Fig. 12 – Cúpula de homogenización (planta)

Por último, también forma parte de la máquina una fuente de alimentación que da a los diodos la corriente necesaria para su iluminación. Para el correcto funcionamiento del sensor, se necesita una fuente de alimentación como la de la figura 13, ya que es capaz de convertir la corriente alterna de la red en continua, y podemos especificar que voltaje y amperaje queremos que entregue a los LED.

Fig. 13 – Fuente de alimentación [11]

El sensor totalmente ensamblado y compuesto por todos los componentes anteriormente descritos queda de la siguiente manera:

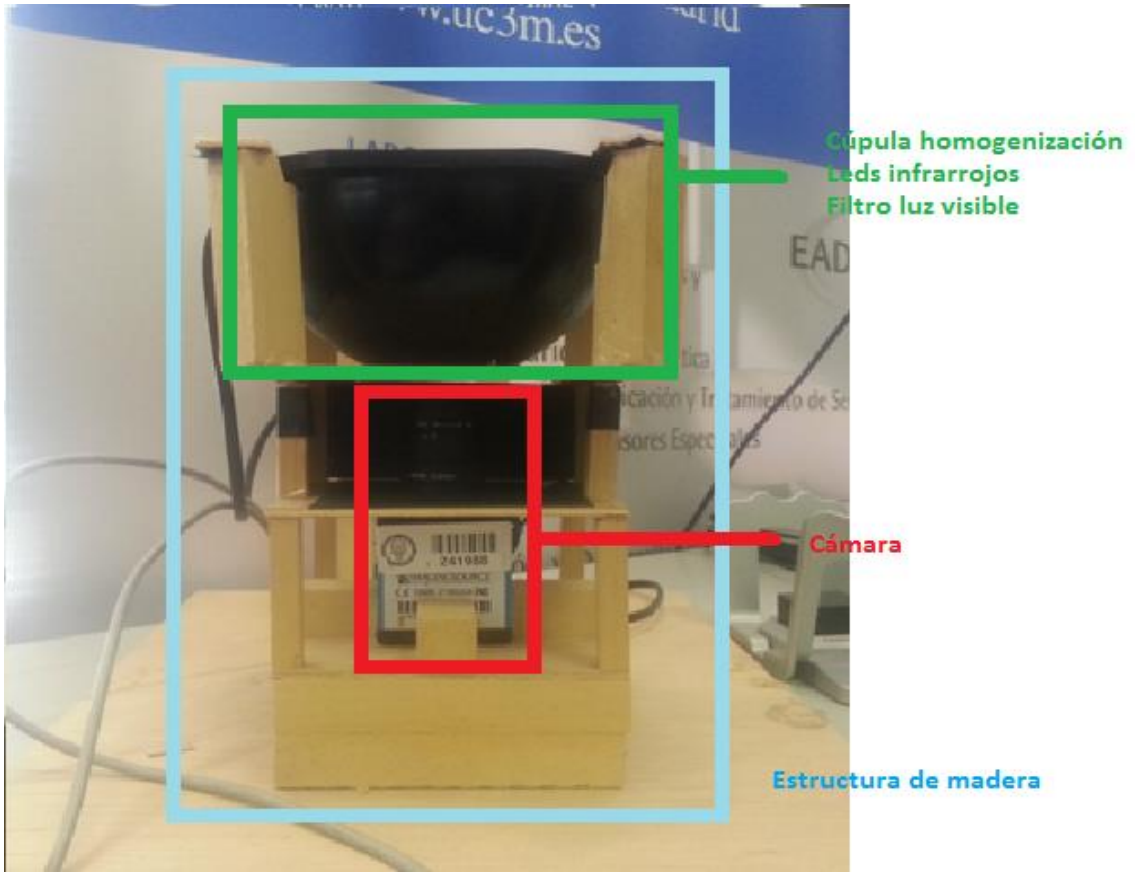

Fig. 14 – Sensor completo

El principal problema que plantea el sensor es la necesidad del usuario de identificar el momento en el que la imagen fuera nítida, estuviera centrada en el visor de la cámara, y lo más importante, que tuviera que apretar un botón para que la imagen quedase registrada. Dando aspecto de imprecisión y predisponiendo negativamente al usuario.

Lo que buscamos con este proyecto es eliminar todos estos inconvenientes, de modo que el usuario únicamente colocando la mano sobre el visor de la cámara, sea detectada por el programa y éste ofrezca unas sencillas instrucciones para que la imagen esté centrada y sea la más adecuada; y en el momento que se identifique una imagen con las características preconcebidas, capturarla. También se persigue miniaturizar la máquina en la medida de lo posible, sin perder calidad en la imagen que se va a capturar.

3.1.1 Principio de funcionamiento del sensor

Antes de comenzar con el planteamiento y el diseño de la solución, lo primero es conocer el funcionamiento del sensor y en que se basa su funcionamiento.

Diferentes partes del cuerpo humano pueden ser usadas para desarrollar sistemas biométricos, pero este en concreto está basado en el patrón de las venas de la palma de la mano. Hasta ahora, este tipo de sistemas no trabaja con todo tipo de manos, porque se ve afectado por las condiciones del ambiente, como temperatura, humedad, luz…

Rayos X, escáner de ultrasonidos y sistemas termales se usan en la actualidad para obtener imágenes de las venas. Si bien es cierto que estos sistemas son bastante fiables y su funcionamiento muy correcto, son, por otra parte, invasivos y caros. Eso hace que no sean totalmente aceptados para aplicaciones biométricas.

Por este motivo se desarrolló la maquina anteriormente descrita (en el apartado 3.1). Para, utilizando la tecnología infrarroja, obtener imágenes de las venas de la mano, usando materiales de bajo coste para abaratar el precio del sistema biométrico.

De este modo, con esta máquina es posible capturar imágenes de las venas superficiales de la palma de la mano. Además, este método no resulta invasivo para el ser humano, ya que no requiere la inyección de ningún tipo de sustancia de contraste en la sangre. Esta máquina también es capaz de tomar imágenes de las venas d la muñeca (con mucho más detalle que las de la mano, ya que las venas se encuentran más superficiales), aunque centraremos nuestro estudio en las venas de la palma de la mano.

El principio de funcionamiento de este sensor consiste en la luz infrarroja, y en su absorción por la hemoglobina de la sangre. El ojo humano solo puede detectar la luz visible (entre 400 y 750 nm de longitud de onda), una pequeña parte del espectro electromagnético. El patrón de las venas humanas es muy poco perceptible con la luz visible.

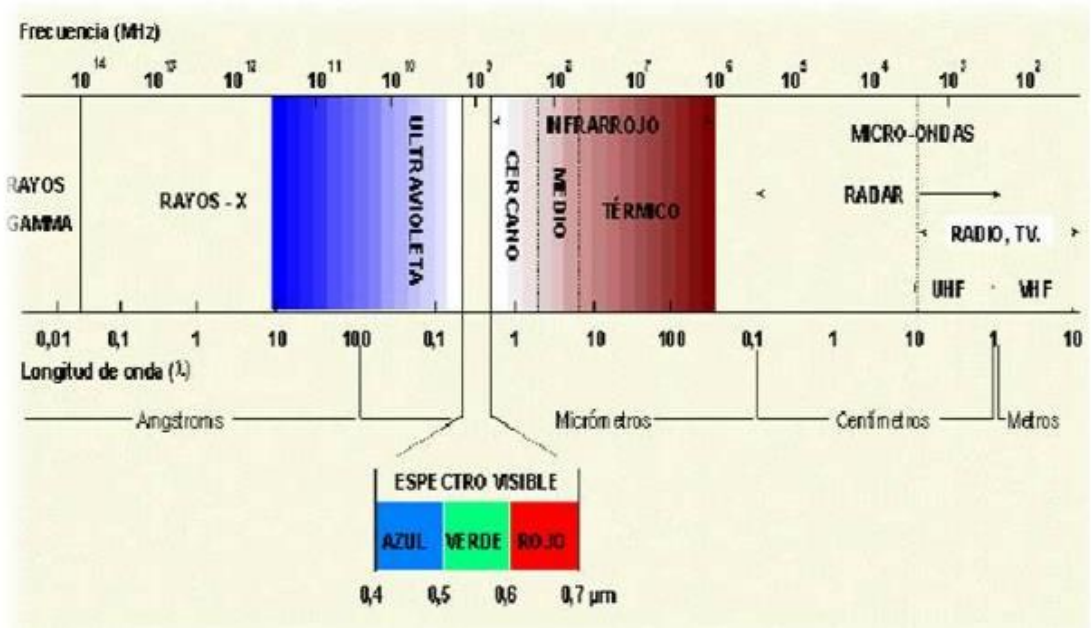

Fig. 15 – Espectro electromagnético [12]

Esto se puede solucionar aplicando luz infrarroja cercana a la piel (NIR, o Near-Infrared) de entre 800 y 950 nm de longitud de onda. La luz infrarroja puede penetrar aproximadamente 3 milímetros en la piel. Por otra parte, la hemoglobina que se encuentra en la sangre de las venas superficiales de la palma de la mano es capaz de absorber dicha luz.

De modo que si se captura, en el momento de incidencia de la luz infrarroja, una imagen con una cámara CCD (con un filtro IR), se obtendrá una imagen con el patrón de las venas más oscuro que el resto de la mano, fácilmente diferenciable.

Se ha demostrado que cuando se usa una longitud de onda de 850 nanómetros aproximadamente, los resultados son mejores que con cualquier otra longitud de onda [13].

Espectro electromagnético (entorno al infrarrojo)	
Denominación	**Longitud de Onda**
Luz Ultravioleta	100 a 3.900 Å
Luz Visible	3.900 a 7.500 Å
Luz Infrarroja (fotográfica)	7.500 a 15.000 Å
Infrarrojo Cercano	15.000 a 200.000 Å
Infrarrojo Lejano	0,002 a 0,1 cm.
Microondas (ondas de radar)	0,1 a 250 cm.

Tabla 3 – Longitudes de onda del espectro electromagnético entorno al infrarrojo [14]

Como el diseño original de la máquina estaba concebido para obtener un patrón de las venas de la muñeca, en la siguiente imagen se muestran unos ejemplos de las imágenes capturadas:

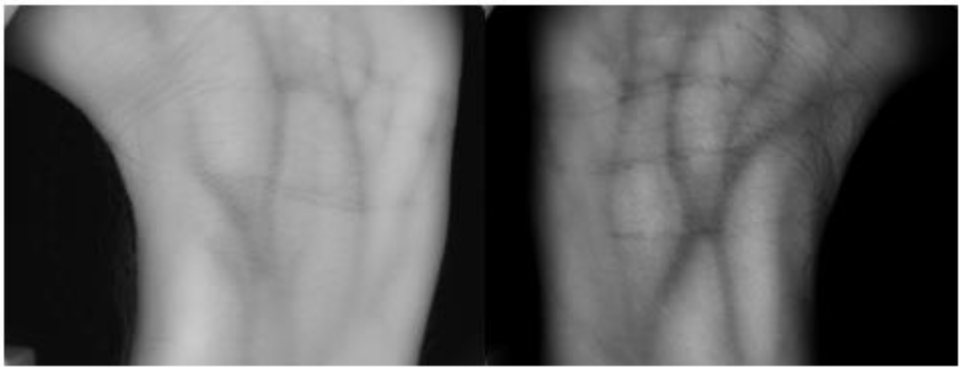

Fig. 16 – Ejemplos de imágenes vasculares de la muñeca [1]

Puesto que se pueden variar algunos parámetros de la cámara, se probó con distintas configuraciones de ganancia y exposición, para obtener la imagen más nítida y con más contraste posible.

3.2 Planteamiento de la solución

Para llevar a cabo la automatización de la máquina de captura de imágenes vasculares, lo primero es elegir un lenguaje de programación adecuado. Para desarrollar este proyecto se ha elegido C#, debido a que es un lenguaje orientado a objetos y a la vez puede crear una interfaz gráfica que será la encargada de interactuar con el usuario una vez finalizado el programa.

Una vez hecho esto, se estudiara las opciones de la cámara encargada de capturar las imágenes, teniendo en cuenta que el enfoque, el zoom y la apertura de la lente son manuales, cobrará especial importancia otras característica como el brillo y el tiempo de apertura de la lente, que si pueden ser manejadas mediante el lenguaje de programación.

También tenemos que tener en cuenta la iluminación. Para que se puedan apreciar correctamente los vasos sanguíneos de la mano, la iluminación debe ser la adecuada, ya que si los diodos emiten demasiada luz infrarroja la imagen puede difuminarse, y si no emiten la suficiente, las venas no resaltaran con respecto al resto de la piel.

Con respecto a la miniaturización del sensor se estudiará en que partes o componentes la disminución de tamaño puede tener mayor impacto, siempre teniendo como principal objetivo la nitidez y claridad de las imágenes.

Una vez tenidos en cuenta todos estos aspectos de la máquina de captura de imágenes vasculares se comenzará con la programación para crear una aplicación capaz de tomar las imágenes de la mano automáticamente. Asimismo la aplicación deberá constar de una interfaz gráfica (creada paralelamente a la programación), y que una vez terminada, será modificada y adaptada para que al usuario le resulte lo más cómoda y atractiva posible, para asegurarnos una mejor predisposición a su manejo.

Para asegurarnos de evitar el rechazo por parte del usuario, la interfaz gráfica también ofrecerá una serie de pasos y guías a seguir, para evitar al usuario una sensación de inseguridad que aparece al usar ciertos sistemas por primera vez.

3.3 Diseño de la solución

Antes de comenzar con el desarrollo, es necesario diseñar la solución del problema. Mediante un análisis del diseño de la solución se busca llegar a aquella que es la óptima entre todas las posibles, buscando la que más eficientemente cumpla todos los requisitos e intentando que los alcance de una forma sencilla. Para ello, hay que valorar las ventajas y desventajas de cada una de las soluciones que se podrían llevar a cabo, escogiendo después la que mejor se adapte a los objetivos que se buscan.

El diseño de la solución se dividirá en cuatro partes: parámetros de la imagen, comparación de la imagen, la interfaz gráfica y miniaturización del sensor. En este apartado se mostrarán las ideas iniciales y cual fue finalmente elegida, para cada una de las partes del diseño de la solución.

3.3.1 Parámetros de la imagen

Para la realización de este proyecto, cuya parte más importante es la programación de la cámara en el lenguaje de programación de C#, se buscarán los parámetros de la cámara que más nos interesen para su implementación. Para identificar estos parámetros se probara con el programa IC Capture (de IC Imaging Control) que viene asociado con la cámara de captura de imágenes.

Se ejecutará el programa, y con el sistema de iluminación de la cámara encendido (para que se ajuste a como se tomarán las imágenes finalmente) se harán distintas pruebas para comprobar que parámetros son los que más varían a la hora de que haya o no una mano sobre el visor de la cámara.

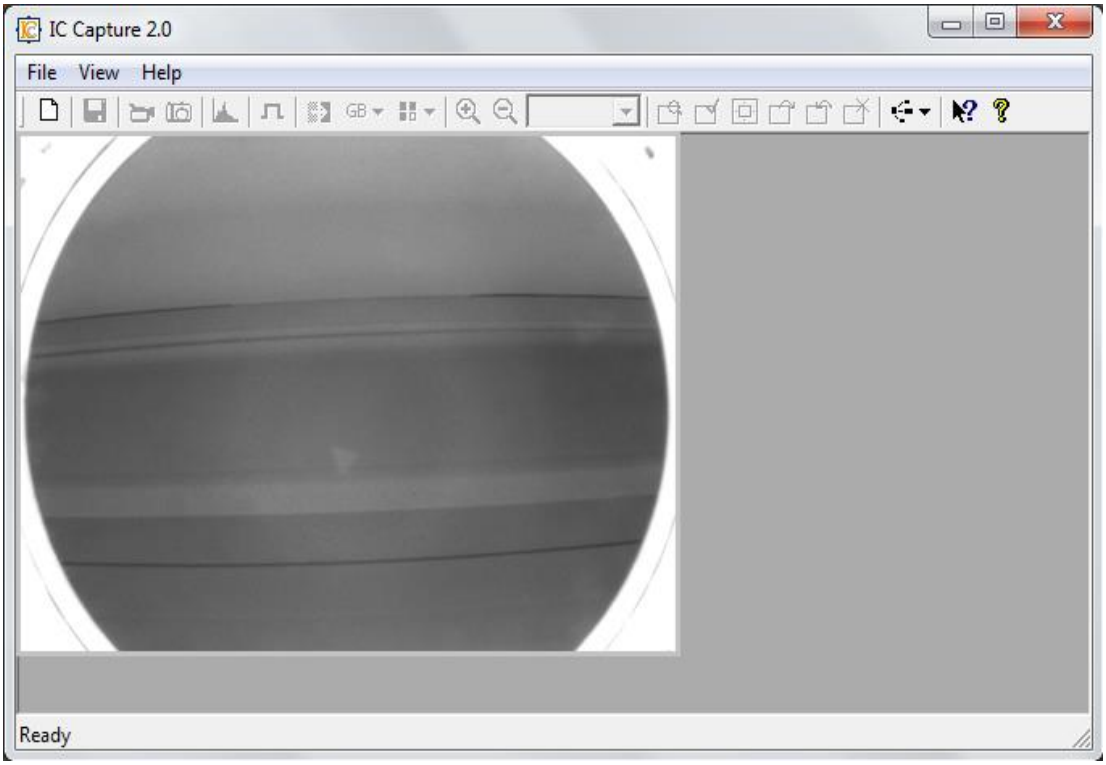

Fig. 17 – IC Capture 2.0

Después de llevar a cabo las pruebas, y tomar diferentes imágenes con el sensor, se determinó el parámetro más óptimo para llevar a cabo el proyecto: el brillo.

3.3.2 Comparación de la imagen

Una vez identificado el parámetro de la imagen que nos va a servir para identificar si existe una mano sobre el objetivo de la cámara, lo próximo que vamos a estudiar es cómo podemos utilizar esos parámetros para: primero, asegurar que hay una mano en el objetivo, y segundo, capturar la imagen cuándo la mano este colocada de la manera más propicia para una mejor identificación vascular, es decir, cuando la mano se encuentre centrada en la imagen.

Para nuestro primer objetivo, comprobar si existe mano sobre el visor, tendremos que comparar el brillo de las dos imágenes. La cámara tiene una resolución de 640 x 480 pixeles y 256 niveles en escala de grises. Cada pixel tiene un valor RGB. Este valor está asociado al brillo del pixel, y en una imagen a color, determina el color de dicho pixel, ya que nos da un valor de rojo (Red), un valor de verde (Green) y un valor de azul (Blue); como en este caso es una cámara de blanco y negro, los valores de rojo, verde y azul son iguales. Esto nos simplifica bastante el tratamiento de la imagen, ya que solo nos es necesario comparar un valor (el de rojo, el de verde o el de azul) para ver la diferencia entre dos imágenes.

Para llevar a cabo esta comparación hay diversas fórmulas, todas ellas bastante efectivas, pero con distinto grado de complejidad:

- **Comparación pixel a pixel:** esta comparación, aunque la más exhaustiva, es inviable ya que el valor del mismo pixel de la imagen para dos imágenes bastante parecidas, puede variar en gran medida. Por eso este método de comparación de imágenes queda descarado.
- **Comparación por sectores:** esta técnica consiste en dividir la imagen por sectores:

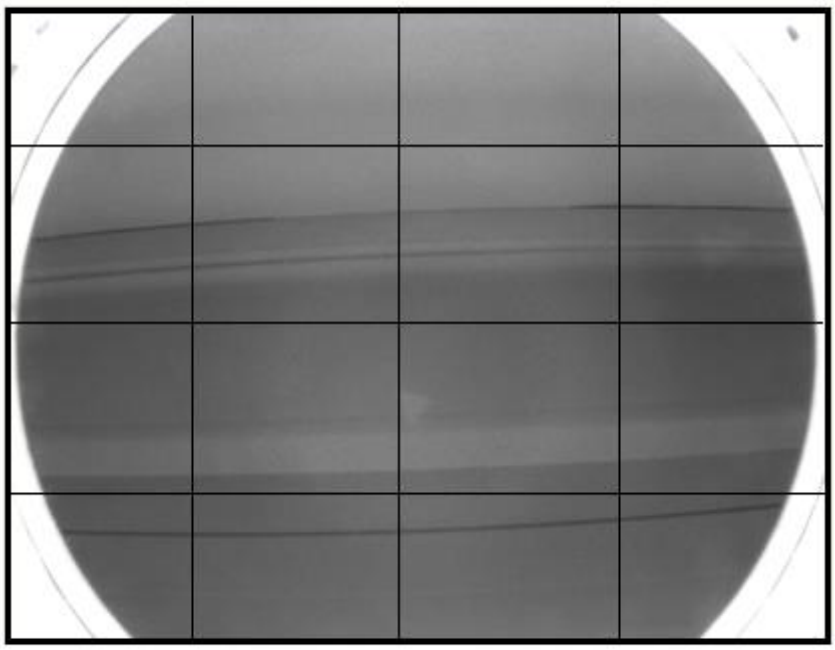

Fig. 18 – Imagen dividida por sectores

De esta manera, hacer la media del valor de los pixeles de cada sector, y compararlo con la media del mismo sector de la siguiente imagen. Este método es bastante apropiado, pero a la hora de generar el código de comparación, resultaría algo complejo (extenso) ya que hay que hace y comparar la media de 16 sectores como se muestra en la imagen 18.

- **Comparación de la imagen completa:** este método consiste en sumar el valor de todos los pixeles de la imagen, y comparar esta suma con la suma de los pixeles de otra imagen. Este sistema será el adecuado si la diferencia es lo suficientemente grande de cuando no hay mano en la imagen a cuando si la hay.

Para el segundo objetivo (tener la mano centrada en la imagen) también existen varias opciones a la hora de comparar y elegir una imagen adecuada:

- **Comparación de los perímetros de las imágenes:** se suma el valor de los pixeles de las áreas seleccionadas (según se muestra en la imagen 19).

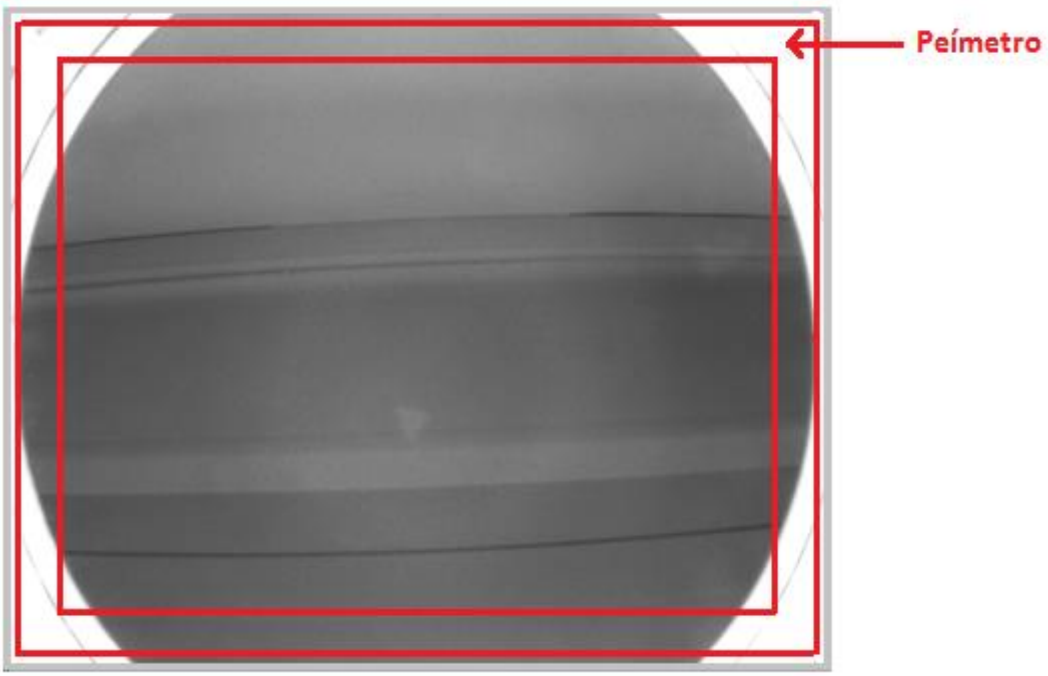

Fig. 19 – Perímetro seleccionado

Este método de comparación puede verse afectado por las esquinas de la imagen. Como puede apreciarse en la imagen 13, las esquinas de la imagen se encuentran en un color blanco. Eso se debe a que la cámara recoge la luz de los leds que se encuentran en la bóveda de homogenización directamente, por lo que esa parte de la imagen siempre va a ser blanca (tener el mismo valor en cuanto a pixeles se refiere), por lo que va a ser información redundante, por lo que no tiene sentido incluirla en la comparación. Por ese motivo este sistema queda descartado.

- **Comparación de rectángulos perimetrales:** varía con respecto al método anterior en que ya no se va a comparar todo el perímetro, sino que solo se van a comparar las partes que nos den información útil de las imágenes, como se muestra en la figura 20

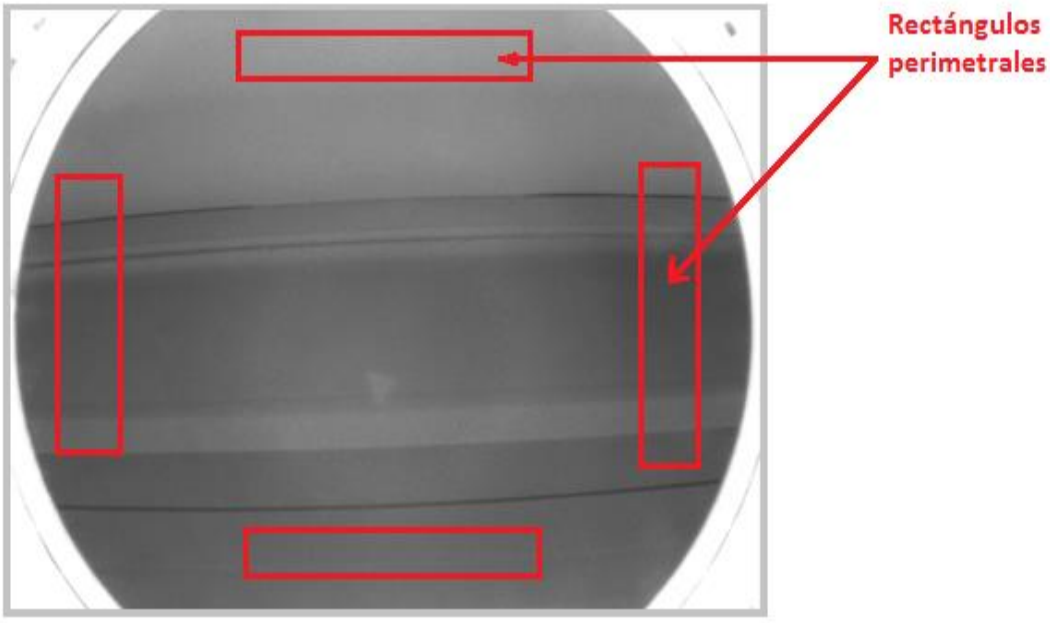

Fig. 20 – Rectángulos perimetrales seleccionados

Este sistema es algo más complicado a la hora de hacer el cálculo, ya que tenemos que hacer la suma y comparación de cuatro rectángulos (se va a comparar la suma de cada rectángulo de una imagen con la de su homólogo de la otra imagen para observar diferencias)

No es necesario comparar el centro de una imagen con el centro de otra, ya que si hay o no mano, se comprobará con el primer objetivo descrito en este apartado.

3.3.3 La interfaz gráfica

La interfaz gráfica que se desarrollará a lo largo del presente TFG, estará pensada para que el usuario (aunque tenga un total desconocimiento de la aplicación) sea capaz de utilizar la máquina de captura de imágenes vasculares. Para ello, la interfaz deberá contar con una serie de pasos para guiar al sujeto que está utilizando el sensor, hasta obtener una imagen correcta de su mano.

A priori, los elementos que no le pueden faltar a nuestra interfaz son: unos botones de apagado y encendido de la aplicación; la imagen que la cámara está captando en tiempo real; las instrucciones que debe seguir el usuario para que la cámara pueda capturar una imagen correcta (también en tiempo real), así como unas indicaciones gráficas de hacia dónde dirigir la mano; y por último, debe contar de algún tipo de comando que nos diga cuándo se ha capturado la imagen, y si fuera posible, que nos mostrara también la imagen que se ha capturado.

3.3.4 Miniaturización

La miniaturización que se llevará a cabo es la disminución del tamaño de los elementos que integran la máquina de captura de imágenes vasculares. La disminución del tamaño se centrará en los siguientes puntos:

- Cámara: Se podría utilizar una cámara de menor tamaño.
- Bóveda de homogenización: se puede sustituir por una más pequeña o directamente eliminarla, siempre y cuando la calidad de la imagen no se vea disminuida.
- Estructura de madera: la estructura de madera se reducirá en función de los elementos que componen la máquina, ya que es la encargada de albergarlos.

La miniaturización siempre va a estar sujeta a la automatización. Es decir, no se comprometerá ningún aspecto de la calidad de las imágenes capturadas por hacer un sensor de menor tamaño.

4 Desarrollo

En este apartado se van a explicar los detalles de la realización del trabajo. Una vez fijados los objetivos y el diseño final del mismo, ya puede llevarse a cabo su creación. A lo largo de las próximas páginas se resumirá el proceso de desarrollo, describiendo los métodos empleados, los problemas encontrados y las soluciones aplicadas.

4.1 Introducción

Antes de empezar con la programación en C#, lo primero fue conocer el manejo de la cámara encargada de capturar la imagen de la palma de la mano. Para ello. El software encargado del manejo de la cámara es el IC Imaging Control. IC Imaging Control es un software propietario, capaz de manejar todo tipo de cámaras, grabadoras y convertidores de "The Imaging Source" (la cámara que se va utilizar durante todo el desarrollo del proyecto pertenece a Imaging Source)

Una vez instalado el software para su manejo se comenzó con una serie de pruebas para ver si su funcionamiento era el adecuado. Una vez que se comprobó esto, se hicieron otra serie de pruebas, para ver que parámetros influían de manera más notoria en las imágenes que se capturaban. Estos parámetros son: el brillo, contraste, nitidez, tiempo de apertura del foco… Con estas pruebas, se llegó a la conclusión de que el parámetro más determinante era el brillo, y se eligió como atributo principal para el desarrollo del proyecto.

En esta primera parte del proyecto también se calibraron los parámetros analógicos de la cámara (zoom, distancia focal y apertura de la lente), ya que como la distancia de la cámara a la mano siempre va a ser la misma, haciendo la calibración una sola vez y comprobando que la imagen es nítida es suficiente para el resto del proyecto. La calibración de la apertura de la lente se hace teniendo colocado el filtro de luz visible y con los LED de infrarrojos encendidos, ya que será así como tenga que tomar las imágenes para el funcionamiento de la aplicación.

Otro punto importante a tener en cuenta es la iluminación de los LED. Las venas de la mano se acentúan más con una determinada iluminación. Para ello se fue probando con

distintos valores de tensión y corriente en la fuente de alimentación, hasta que se llegó a la conclusión de que las venas se veían de forma más nítida a 20 voltios de tensión y 190 miliamperios de corriente (la corriente viene asociada al voltaje en la fuente de alimentación)

Una vez hecho este estudio preliminar es el momento de empezar con la programación de la cámara para automatizar la captura de imágenes.

4.2 Desarrollo del código en C#

Lo primero para poder empezar a trabajar en la programación de la cámara, es comenzar un nuevo proyecto de C# en Visual Studio. Abierto el nuevo proyecto ya disponemos de nuestro formulario (interfaz gráfica que se ejecutará cuando el usuario utilice nuestra aplicación). El siguiente paso es añadir las referencias de la propia cámara a las ya existentes en C#. Las referencias añadidas son:

- AxInterop.ICImagingControl3
- ImagingControl3
- Interop.ICImagingControl3

Una vez hecho esto, el siguiente paso para comenzar con la programación fue familiarizarse con las librerías y métodos propios de la cámara, ya que son los métodos que se van a utilizar para la implementación del proyecto. Se hizo especial hincapié en esto para entender correctamente su funcionamiento y sus características.

La implementación de la solución para llevar a cabo nuestra aplicación se basa en dos partes. La primera es el funcionamiento básico de la cámara (encendido, apagado, captura de imagen, almacenamiento de la imagen…), y la segunda parte consta del análisis de la imagen, es decir, si la imagen que está contemplando la cámara es la adecuada para ser almacenada.

4.2.1 Funcionamiento de la cámara

Una vez que se han añadido las referencias propias de la cámara, ya se puede disponer en el cuadro de herramientas de un nuevo componente “ICImagingControl”. Esta herramienta nos servirá para ver en la pantalla del ordenador lo que la cámara está visualizando.

El primer programa de prueba que se llevó a cabo fue un sencillo programa, en cuyo su formulario únicamente aparecían la ventana del “ICImagingControl” y dos botones, uno “Encender” (que hace que la cámara comience a tomar imágenes) y otro “Apagar” (encargado de que la cámara deje de tomar imágenes). El método necesario para que el formulario se ejecute es el siguiente (figura 21):

```
private void Form1_Load_1(object sender, EventArgs e)
{
    if (!icImagingControl1.DeviceValid)
    {
        icImagingControl1.ShowDeviceSettingsDialog();

        if (!icImagingControl1.DeviceValid)
        {
            MessageBox.Show("No device was selected.", "Programa_Piloto",
MessageBoxButtons.OK, MessageBoxIcon.Information);

            this.Close();
            return;
        }
    }
}
```

Fig. 21 – Código de inicio de programa

Es necesario un código para seleccionar la cámara y comprobar que está conectada al ordenador (icImagingControl1.ShowDeviceSettingsDialog()). Este método muestra la ventana de selección de la cámara, necesaria para que se muestre la imagen por pantalla.

El código utilizado por los botones de "Encendido" y "Apagado" se muestra en la figura 22:

```
private void Encender_Click(object sender, EventArgs e)
{
        icImagingControl1.LiveStart();
}

private void Apagar_Click(object sender, EventArgs e)
{
        icImagingControl1.LiveStop();
}
```

Fig. 22 – Código botones Encender y Apagar

Tanto el método "icImagingControl1.LiveStart()" como el método "icImagingControl1.LiveStop()" son métodos propios de las librerías de la cámara. El primero es el encargado de que la cámara comience a captar y emitir imágenes por pantalla (el método se activa al pulsar el botón Encender); y el segundo revierte el efecto del primer, es decir, hace que la cámara deje de captar imágenes (el método se activa al pulsar el botón Apagar).

4.2.2 Captura automática de la imagen

Una vez comprendido el funcionamiento básico de la cámara, lo que se busca a continuación es la captura y almacenado de la imagen de manera automática.

Para llevar esto a cabo se necesitaba, en primer lugar, un procedimiento que pudiéramos seguir para detectar que hay colocada una mano encima del visor. Y si fuera posible, utilizando un parámetro común para todas las imágenes que capturase la cámara.

Con el propósito de encontrar el parámetro que identificase la presencia o ausencia de mano encima de la cámara se hicieron una serie de pruebas para ver que parámetros de la imagen cambiaban en función de si había mano o no encima del visor de la cámara.

La conclusión que se obtuvo de dichas pruebas, fue que el parámetro que más significativamente cambiaba cuando se colocaba la mano en la máquina era el brillo. Por lo tanto, teniendo en cuenta únicamente este parámetro, se podía saber (con una fiabilidad muy eficiente) si había una mano o no encima del objetivo de la cámara.

Para obtener unos números con los que poder trabajar e identificar la diferencia del brillo de cada imagen, seguimos el siguiente procedimiento: sumar el valor de todos los pixeles de la imagen. Se eligió este procedimiento porque todas las imágenes están formadas por pixeles, que a su vez tienen un valor numérico asociado al brillo. Cada pixel tiene tres parámetros (R, G y B) distintos para una imagen a color, pero como la cámara utilizada es de blanco y negro, el valor R, G y B de cada pixel es el mismo. En este caso se utilizó el valor R. El código que efectúa la suma es el siguiente:

```
int suma1 = 0;
for (int i = 0; i < 640; i++)
{
        for (int j = 0; j < 480; j++)
        {
                suma1 = suma1 + (byte)image1.GetPixel(i, j).R;
        }
}
```

Fig. 23 – Código que suma el valor de los pixeles de la imagen

Este cogido se encarga de recorrer la imagen por filas y columnas, y suma el valor R de todos los pixeles. La imagen de la cámara es de 640 x 480 pixeles.

Una vez conocido el código que ofrece un valor numérico de cada imagen, se hizo otra serie de pruebas, para establecer que valores indicaban la presencia de una mano encima del visor de la cámara y cuáles no. Después de hacer diferentes pruebas con distintas manos y diferentes tipos de luz ambiente, se llegó a la conclusión de que había una mano situada en la máquina, cuando el programa ofrecía un valor por encima de 39.000.000 de la variable "int suma1". Si el valor de dicha variable no superaba esa cifra, se asumía que no había mano sobre la cámara **(Primera condición para la captura de la imagen)**.

Llegados a este punto, lo próximo que se necesitaba era que el programa tomase una serie de imágenes, una vez hubiéramos pulsado el botón "Encender" y las comparara con los valores de brillo que se manejaban cuando había mano sobre el visor y si este valor es superior al descrito anteriormente, capturase automáticamente la imagen.

Para poder programar esto, en primer lugar se necesitaba poder almacenar la imagen y convertirla en un "bitmap" para poder hacer la suma de sus pixeles con una mayor facilidad. Este es el código que se encarga de ello:

```
icImagingControl1.MemorySnapImage();
icImagingControl1.MemorySaveImage(@"C:\Documents and
Settings\Guti\Escritorio\Programa TFG\temporal.jpeg");

Bitmap image1 = new Bitmap(@"C:\Documents and
Settings\Guti\Escritorio\Programa TFG\temporal.jpeg");
```

Fig. 24 – Código encargado de guardar y convertir la imagen

Los dos primeros métodos que aparecen son los encargados de almacenar la imagen. Más concretamente, el primer método (icImagingControl1.MemorySnapImage()) la salva, y el segundo método (icImagingControl1.MemorySaveImage()) la guarda en la dirección de memoria que se especifica (la dirección especificada se encuentra en color rojo).

El comando "Bitmap image1 = new Bitmap(@"C:\Documents and Settings\Guti\Escritorio\Programa TFG\temporal.jpeg")", trasforma la imagen que se ha guardado en la dirección de memoria especificada y la convierte en un bitmap.

Para evitar la saturación del programa (debido a los cálculos y la conversión de imagen a bitmap que tiene que llevar a cabo) se recurrió a un temporizador. El temporizador iba a ser el encargado de que el cálculo de la suma de los pixeles de la imagen, así como su almacenado, no se hiciera de manera continua; el temporizador sería el encargado de que solo se llevara a cabo el procesado de una imagen teniendo en cuenta un tiempo lógico para que una persona coloque su mano sobre la cámara, de manera que no fuera demasiado pequeño, ya que no habría prácticamente variación de una imagen a otra, pero que tampoco fuera demasiado grande para evitar perder información valiosa.

Con este propósito se hicieron pruebas con diferentes tiempos entre capturas de imágenes, y se tomó un tiempo de 700 milisegundos como el más óptimo. El temporizador se activaría con el botón "Encender" y capturaría una imagen con el intervalo de tiempo anteriormente descrito. Por lo tanto el botón "Encender" quedaba configurado de la siguiente manera, tal y como se muestra en la imagen 25:

```
private void Encender_Click(object sender, EventArgs e)
{
      icImagingControl1.LiveStart();
      icImagingControl1.Visible = true;

      timer1.Enabled = true;
      timer1.Interval = 700;
}
```

Fig. 25 – Código asociado al botón encender

Por lo tanto, las funciones del temporizador (llamado timer1) son las siguientes hasta este punto del programa:

```
private void timer1_Tick(object sender, EventArgs e)
{
    icImagingControl1.MemorySnapImage();
    icImagingControl1.MemorySaveImage(@"C:\Documents and
Settings\Guti\Escritorio\Programa TFG\temporal.jpeg");

    Bitmap image1 = new Bitmap(@"C:\Documents and
Settings\Guti\Escritorio\Programa TFG\temporal.jpeg");

    int suma1 = 0;
    for (int i = 0; i < 640; i++)
    {
        for (int j = 0; j < 480; j++)
        {
            suma1 = suma1 + (byte)image1.GetPixel(i, j).R;
        }
    }

    if (suma1 > 39000000 && suma1 < 40800000)
    {
        timer1.Stop();
        label13.Text = "Imagen Capturada";
    }
    else
    {
        aux = suma1;
        label2.Text = Convert.ToString(aux);
        label13.Text = null;
     }
     image1.Dispose();
}
```

Fig. 26 – Código asociado al timer1

Por este orden, y cada 700milisegundos, el programa captura una imagen, la guarda, la trasforma en un bitmap, suma el valor de sus pixeles, y si la suma de ellos supera los 39.000.000 se detiene el temporizador y la imagen capturada es válida. En caso de no superar ese valor el proceso se repetirá dentro de 700 milisegundos. Por último se eliminará el bitmap image1 mediante el método "Dispose()" (image1.Dispose())

Como se puede apreciar en el código se han añadido un par de etiquetas (label12 y label13). La etiqueta label13 es la encargada de mostrar en el formulario cuando una imagen ha sido dada por válida y por lo tanto capturada (en ese caso mostrara: Imagen Capturada); en caso de que la imagen no haya sido valida, esta etiqueta no mostrará nada.

La etiqueta label12 ha sido colocada para mostrar por pantalla el valor numérico de la suma de los pixeles de cada imagen, y así poder comprobar que no captura ninguna imagen con un valor inferior al deseado. Esta etiqueta también será muy útil para ver el valor de cada imagen en función de cómo coloquemos la mano, ya que siempre va a facilitar información en tiempo real al actualizarse con el temporizador.

Con la intención de evitar errores, también se ha puesto un límite superior para la variable suma1 (suma del valor de los pixeles de la imagen); es decir, si la variable suma1 supera el valor 40.800.000 la imagen tampoco será capturada. Este límite superior se estableció porque las imágenes con un valor superior a este tienen una calidad que no es aceptable para el desarrollo del proyecto.

Hasta este punto, una vez iniciado el programa (mediante el botón "Encender"), es capaz de reconocer cuando hay una mano situada en el objetivo de la cámara, capturar la imagen (cuando sea correcta) y guardarla. Este es el objetivo del proyecto, aunque todavía se necesitaba un poco más de precisión, ya que en algunas ocasiones aunque capturaba la imagen de la palma de la mano, ésta no estaba centrada.

Con la intención de hacer frente a estas imprecisiones y corregirlas, se llevó a cabo un sistema de detección por zonas basándose en la geometría de la imagen. Es decir, se seleccionaron 4 perimetrales de 20 x 40 pixeles cada uno (los laterales en posición vertical, y el superior e inferior colocados horizontalmente), y dispuestos de la siguiente manera:

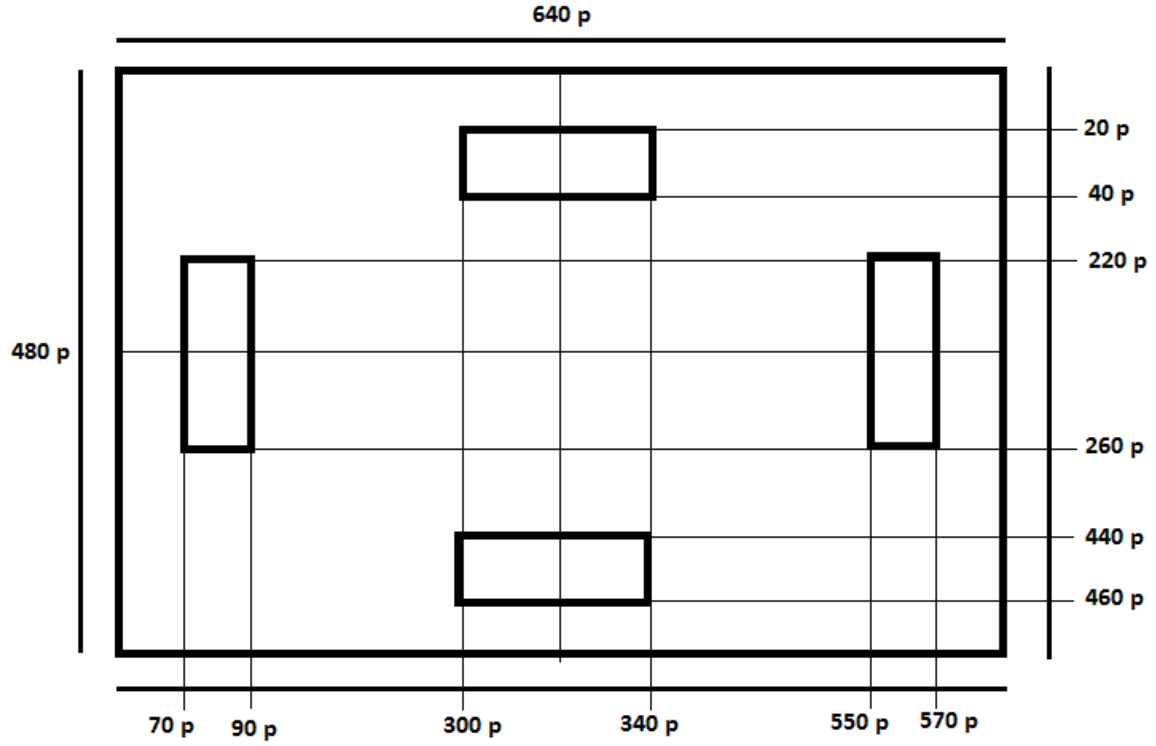

Fig. 27 – Rectángulos perimetrales

Se eligió esta distribución de las zonas de análisis debido a su mejor precisión después de efectuar una serie de pruebas con distintos valores.

Los rectángulos más gruesos representan los grupos de pixeles que se han cogido para hacer la comprobación de que la palma de la mano está centrada sobre el visor. Para cada rectángulo de pixeles seleccionado se actúa igual que para la imagen completa, sumando el valor de cada pixel:

```
int cuadrado_superior = 0;
for (int i = 300; i < 340; i++)
{
  for (int j = 20; j < 40; j++)
  {
   cuadrado_superior = cuadrado_superior + (byte)image1.GetPixel(i, j).R;
  }
}
label3.Text = Convert.ToString(cuadrado_superior);

int cuadrado_inferior = 0;
for (int i = 300; i < 340; i++)
{
  for (int j = 440; j < 460; j++)
```

```
  {
   cuadrado_inferior = cuadrado_inferior + (byte)image1.GetPixel(i, j).R;
  }
}
label4.Text = Convert.ToString(cuadrado_inferior);

int cuadrado_izquierda = 0;
for (int i = 70; i < 90; i++)
{
  for (int j = 220; j < 260; j++)
  {
   cuadrado_izquierda = cuadrado_izqda + (byte)image1.GetPixel(i, j).R;
  }
}
label5.Text = Convert.ToString(cuadrado_izquierda);

int cuadrado_derecha = 0;
for (int i = 550; i < 570; i++)
{
  for (int j = 220; j < 260; j++)
  {
   cuadrado_derecha = cuadrado_derecha + (byte)image1.GetPixel(i, j).R;
  }
}
label6.Text = Convert.ToString(cuadrado_derecha);
```

Fig. 28 – Código que suma el valor de los pixeles de los rectángulos perimetrales

Como se muestra en la figura 28, este código hace que se vaya recorriendo cada rectángulo y sumando el valor de sus pixeles. A cada rectángulo se le ha añadido una etiqueta que muestra por pantalla y en tiempo real la suma de los pixeles.

Con estas etiquetas que nos daban el valor de cada rectángulo en tiempo real, y haciendo una serie de pruebas colocando la mano en diferentes zonas de la imagen, se llegó a la conclusión de que en los rectángulos que "había" mano, el valor de la suma de sus pixeles era superior al de los rectángulos donde no había mano. Esto hace posible, que comparando cada rectángulo con su opuesto (el inferior con el superior y el derecho con el izquierdo), se consiga que solo se tome la imagen cuando la mano este centrada en el objetivo.

Añadimos estas dos nuevas condiciones a la que ya teníamos antes para la captura y procesado de la imagen:

```
if (suma1 > 39000000 && suma1 < 40800000
    && Math.Abs(cuadrado_izquierda - cuadrado_derecha) < 50000
    && Math.Abs(cuadrado_inferior - cuadrado_superior) < 50000
    && Math.Abs(suma1 - aux) < 120000)
{
            timer1.Stop();
            label13.Text = "Imagen Capturada";
}
```

Fig. 29 – Condiciones para la captura de la imagen

Por lo tanto y finalmente, la imagen solo será capturada y almacenada si cumple la primera condición (explicada anteriormente) y si la diferencia entre la suma del valor de los pixeles de sus rectángulos opuestos es inferior a 50.000. Esto nos asegura que hay mano sobre el visor, y que además está centrada.

Como se puede apreciar se ha añadido una cuarta condición: "Math.Abs(suma1 - aux) < 120000". Esta condición se ha desarrollado con el propósito de estabilizar la imagen; es decir, si el usuario está colocando la mano sobre el visor, la diferencia de brillo entre una imagen y la anterior es notable. Si únicamente tomamos la imagen cuando la diferencia (entre las dos imágenes) es lo suficientemente pequeña, nos aseguramos de que la imagen se tome cuando la mano de la persona este quieta, ya que en ese caso evitamos que la captura sea borrosa o poco nítida, y haya que desecharla. El umbral que la diferencia de imágenes no puede superar se estableció en 120.000 (después de probar con diferentes cantidades).

4.2.3 Desarrollo de la interfaz gráfica

Para mejorar la interfaz gráfica que utilizará y guiará al usuario en el futuro, el programa se va a servir de los rectángulos de pixeles creados anteriormente para dar una serie de instrucciones por pantalla, que ayuden al usuario a colocar la mano de la manera más adecuada sobre el objetivo.

Hasta el momento, en la interfaz gráfica únicamente aparece la ventana de visualización de la cámara, los botones "Encender" y "Apagar", la etiqueta que nos da el valor de la imagen y las que nos dan el valor de los rectángulos perimetrales.

Utilizando el valor y las diferencias de estos rectángulos perimetrales, se van a mostrar por pantalla (mediante el uso de etiquetas) una serie de sugerencias para que el usuario coloque su mano correctamente. Además, estas sugerencias irán acompañadas de una señal gráfica (en forma de flecha) de hacia dónde desplazar la mano. El código que se encarga de este proceso podemos verlo en la imagen 30:

```
if ((cuadrado_inferior - cuadrado_superior) > 50000)
{
    label11.Text = "Mueva la mano hacia delante";
    pictureBox2.Image = Image.FromFile(@"C:\Documents and
Settings\Guti\Escritorio\Programa TFG\flecha_adelante.jpg");
}

if ((cuadrado_inferior - cuadrado_superior) < -50000)
{
    label11.Text = "Mueva la mano hacia atras";
    pictureBox2.Image = Image.FromFile(@"C:\Documents and
Settings\Guti\Escritorio\Programa TFG\flecha_atras.jpg");
}

if (Math.Abs(cuadrado_inferior - cuadrado_superior) < 50000)
{
    label11.Text = null;
                pictureBox2.Image = null;
}

if ((cuadrado_izquierda - cuadrado_derecha) > 50000)
{
    label12.Text = "Mueva la mano hacia la izquierda";
    pictureBox3.Image = Image.FromFile(@"C:\Documents and
Settings\Guti\Escritorio\Programa TFG\flecha_izquierda.jpg");
}

if ((cuadrado_izquierda - cuadrado_derecha) < -50000)
{
    label12.Text = "Mueva la mano hacia la derecha";
    pictureBox3.Image = Image.FromFile(@"C:\Documents and
Settings\Guti\Escritorio\Programa TFG\flecha_derecha.jpg");
}

if (Math.Abs(cuadrado_izquierda - cuadrado_derecha) < 50000)
{
    label12.Text = null;
    pictureBox3.Image = null;
}
```

Fig. 30 – Código encargado de utilizar las diferencias entre los rectángulos perimetrales

La imagen de la flecha que apunta hacia delante y la que apunta hacia detrás aparecerá en el picturebox2 y las flechas que apuntan a derecha e izquierda en el picturebox3. Se mostraran únicamente cuando la diferencia entre los rectángulos opuestos sea mayor que 50.000.

Como último paso para mejorar la interfaz gráfica se ha añadido el picturebox1, encargado de mostrar por pantalla la imagen que se ha capturado mediante el siguiente código:

```
if (suma1 > 39000000 && suma1 < 40800000
    && Math.Abs(cuadrado_izquierda - cuadrado_derecha) < 50000
    && Math.Abs(cuadrado_inferior - cuadrado_superior) < 50000
    && Math.Abs(suma1 - aux) < 120000)
{
        timer1.Stop();
        label13.Text = "Imagen Capturada";

        if (pictureBox1.Image != null)
        {
            pictureBox1.Image.Dispose();
        }

        pictureBox1.Image = Image.FromFile(@"C:\Documents and
Settings\Guti\Escritorio\Programa TFG\imagen.jpeg");

        System.Threading.Thread.Sleep(3000);
        pictureBox1.Image = null;
        label13.Text = null;
        timer1.Start();
}
```

Fig. 31 – Código encargado del picturebox

Para evitar los errores en el picturebox1, antes de cargar la imagen capturada es necesario asegurarse que no hay ninguna otra imagen, y si hay, borrarla ante de cargar una nueva. Tanto la imagen que aparece en el picturebox1, como el texto de la etiqueta 13 (imagen capturada), desaparecerán al cabo de 3 segundos (System.Threading.Thread.Sleep(3000)). Por último, trascurrido este tiempo, el temporizador se reiniciara y el programa quedará listo para capturar la siguiente imagen.

4.3 Resultado final del software

Tras llevar a cabo el desarrollo de la aplicación anteriormente descrito, la interfaz para el usuario queda de la siguiente manera una vez se ejecuta el programa (figura 32):

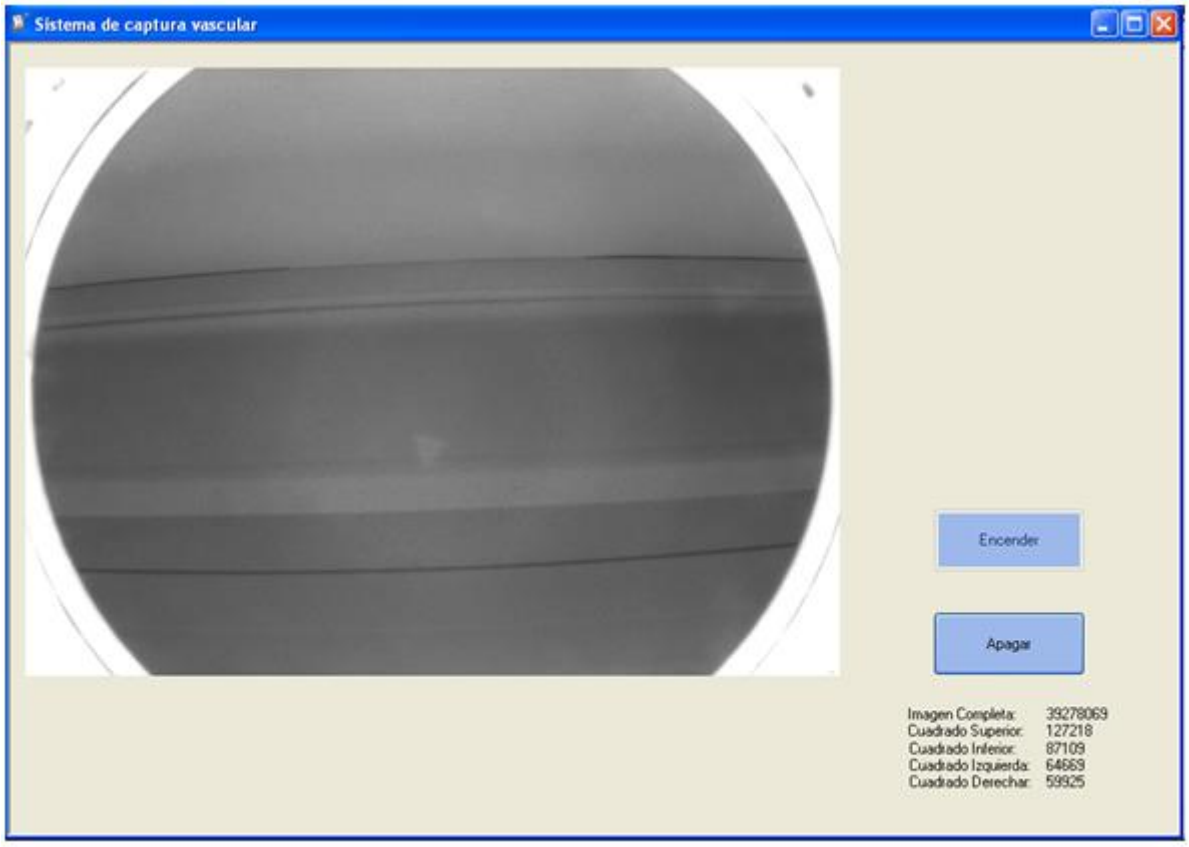

Fig. 32 – Interfaz gráfica final

Cuando el programa se ejecuta y el usuario pulsa el botón encender, la cámara comienza a emitir imágenes por pantalla, tal y como se puede ver en el recuadro. Aparecen también los botes "Encender" y "Apagar", para el accionamiento y desactivación de la aplicación.

Debajo de estos dos botones, aparecen una serie de etiquetas, y a su derecha unos números, que son la suma de los pixeles de la imagen completa, y de los rectángulos perimetrales, que se actualizan cada 700 milisegundos y son muy útiles para probar y desarrollar el programa. En la interfaz final para el usuario se podrían suprimir estas etiquetas, ya que no serían de ninguna utilidad.

El último detalle que falta por añadir es el filtro de luz visible y encender los LED de luz infrarroja. Una vez hecho esto, el resultado se muestra en la imagen 33:

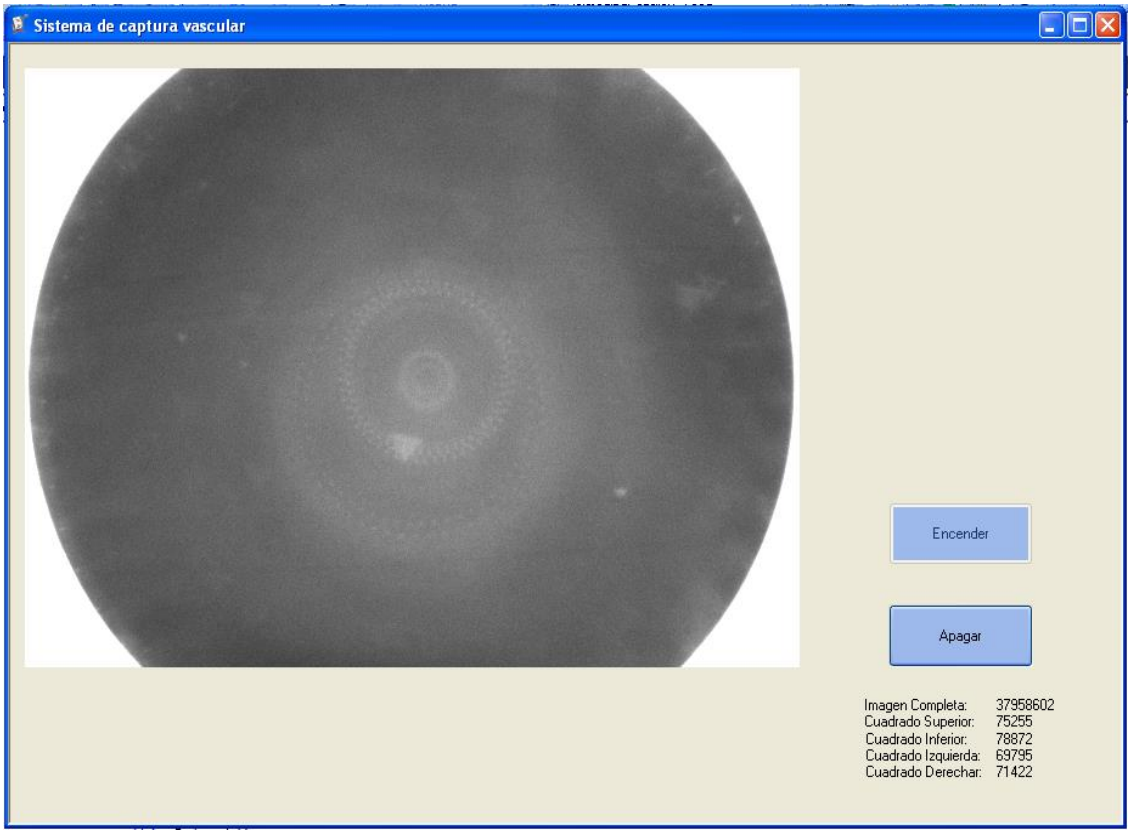

Fig. 33 – Interfaz gráfica final con LED encendidos

Los círculos que aparecen en la imagen que transmite la cámara se deben al reflejo de los LED en el filtro, pero no tendrá mayores consecuencias para el resultado final de captura de la imagen.

4.4 Miniaturización

Una vez desarrollado totalmente el código que automatiza el sensor, y creada la interfaz gráfica para interactuar con el usuario, se aborda el tema de la miniaturización.

La máquina consta de tres elementos susceptibles de miniaturización: el sistema de iluminación, la cámara y la estructura de madera. Se van a analizar estos tres campos, para determinar en cuál de ellos sería más efectiva la reducción del tamaño.

- **La cámara:** Es el elemento de la máquina en que más fácil resultaría la reducción de tamaño, debido a que únicamente bastaría con la sustitución de la cámara actual por una de menor tamaño; y siempre y cuando esta nueva cámara fuera de "Imaging

Source", no habría que efectuar ningún cambio en la programación, ya que el programa desarrollado hasta ahora nos sería totalmente compatible.

Debido a su elevado precio, y su facilidad de cambio cuando sea necesario, hemos decidido no incluirla en el presente proyecto. Aunque habría supuesto una reducción en la altura total de la máquina de entre 2 a 5 centímetros (para lo cual habría sido necesario la modificación de la estructura).

- **El sistema de iluminación:** consta de dos partes, los LED emisores de luz infrarroja, y la bóveda de homogenización.

  Los LED son de un tamaño bastante reducido, por lo que no es posible su miniaturización.

  Por otra parte la bóveda de homogenización es la parte más voluminosa del sensor de captura de imágenes vasculares. Como se muestra en la siguiente imagen, la bóveda es la encargada de reflejar la luz emitida por los diodos (colocados en el borde superior de la cúpula) hacia la mano.

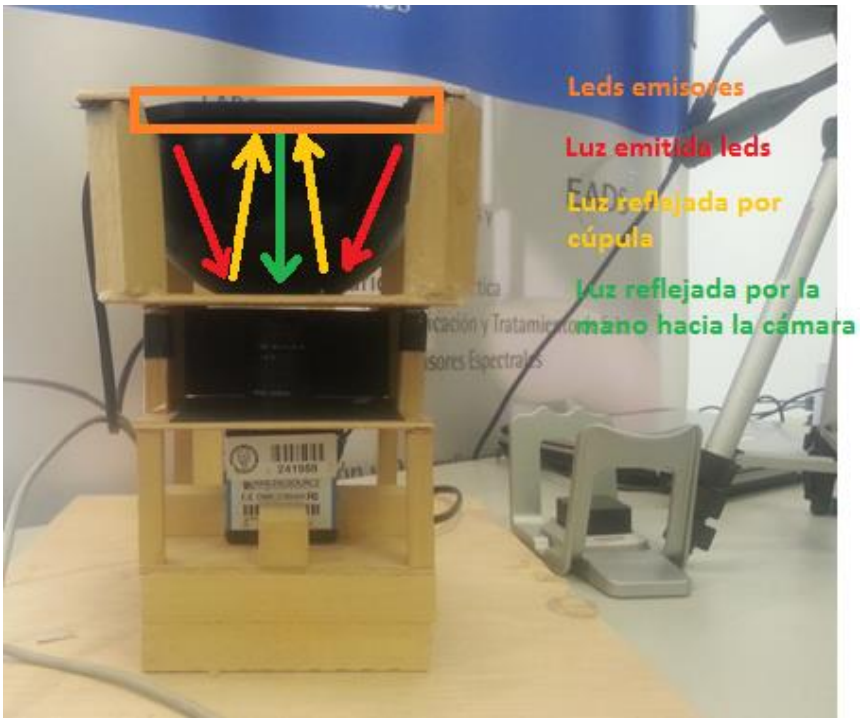

Fig. 34 – Funcionamiento de la bóveda

Para poder reducir esta sección de la maquina se pensó en dos procedimientos. El primero era colocar una bóveda más pequeña, lo que se descartó rápidamente debido a que el diámetro de la semiesfera debe ser mayor que cualquier mano para que se pueda tomar una imagen completa de cualquier persona. Y el segundo era eliminar la bóveda, colocando los LED justo enfrente de la mano para que la luz

llegue directamente a la mano sin necesidad de reflejarse en ningún tipo de objeto. La idea era la siguiente:

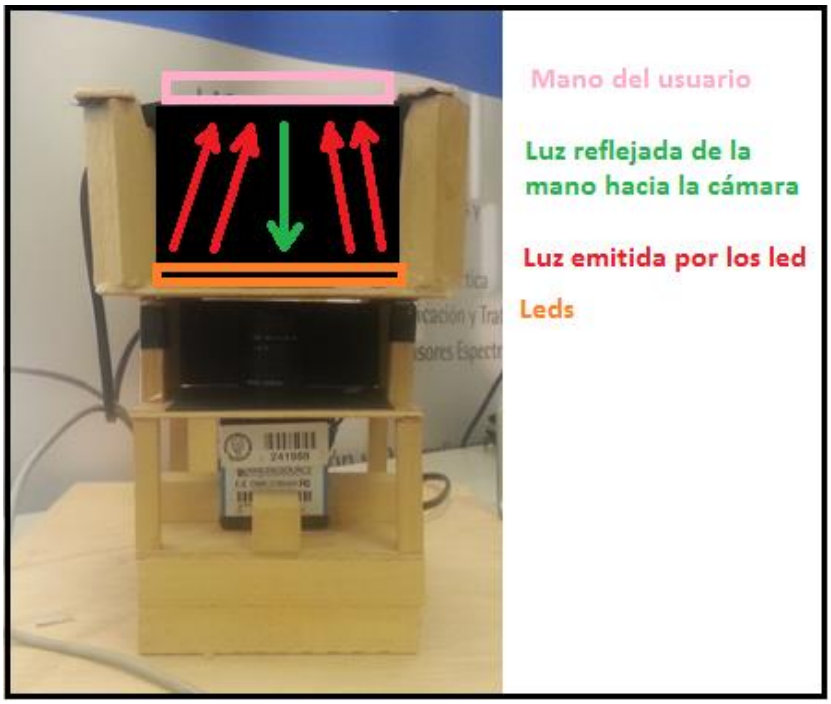

Fig. 35 – Funcionamiento alternativo sin bóveda

Como se demuestra en el artículo "Capturing Hand or Wrist Vein Images for Biometric Authentication Using Low-Cost Devices" publicado por J. Enrique Suarez Pascual, Jaime Uriarte-Antonio, Raúl Sanchez-Reillo, Michael G. Lorenz [1], el desarrollo del sensor, parte de un prototipo sin DOM, o bóveda de homogenización. Cuando se efectuaron las pruebas con este prototipo de la máquina se concluyó que las imágenes obtenidas no eran las adecuadas para su posterior tratamiento y procesado.

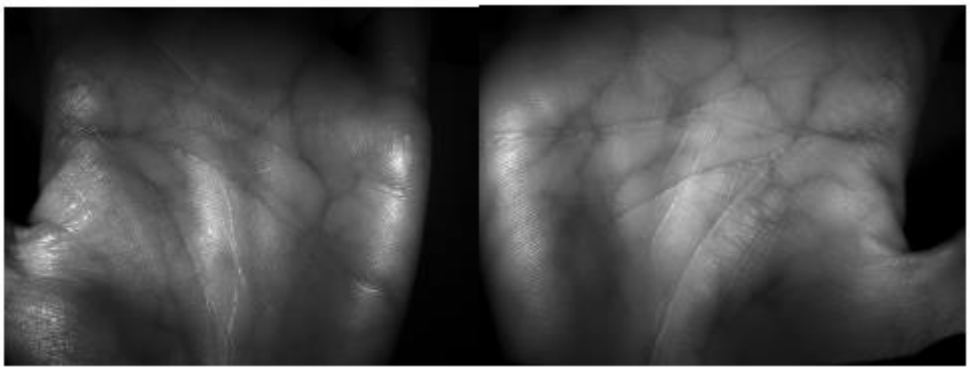

Fig. 36 – Imágenes tomadas sin DOM

Como se puede observar en la figura 36, la luz no está distribuida correctamente por toda la superficie de la mano. Para que la luz emitida por los diodos se distribuya uniformemente por la superficie, es necesario que esté presente la bóveda de homogenización. Por lo que tampoco podemos prescindir de este elemento.

- **La estructura:** el diseño de la estructura está sujeto a los componentes que necesita la máquina de captura de imágenes vasculares, para su correcto funcionamiento. Cómo no se ha efectuado ningún cambio en ninguno de los componentes principales de la máquina (sistema de iluminación y cámara), el diseño de la estructura tampoco ha variado.

Una vez terminado el desarrollo del proyecto (tanto su parte de software, como de hardware), está listo para llevar a cabo las pruebas necesarias que aseguren su correcto funcionamiento para todos los casos que se puedan dar.

5 Pruebas

En el presente apartado se presentarán las pruebas realizadas para comprobar el correcto funcionamiento de la aplicación creada; a su vez sirve para comprobar el funcionamiento del código para la captura automática de imágenes.

5.1 Prueba de robustez

Antes de dar por valido el programa de automatización de la máquina de captura de imágenes vasculares, será sometido a una serie de pruebas para valorar su robustez y así poder asegurar que no se recogerán imágenes defectuosas o no válidas para su tratamiento.

Para ello se simulara un usuario que nunca ha utilizado dicha máquina y no sabe muy bien donde debe colocar la mano, suponiendo unas condiciones iniciales de sensor y LED ya encendidos. En primer lugar, pulsará el botón "Encender". En este momento, la cámara comenzará a emitir por pantalla las imágenes que toma por el objetivo. Con el botón "Encender" ya pulsado, la interfaz gráfica que aparece es la que se muestra en la figura 37.

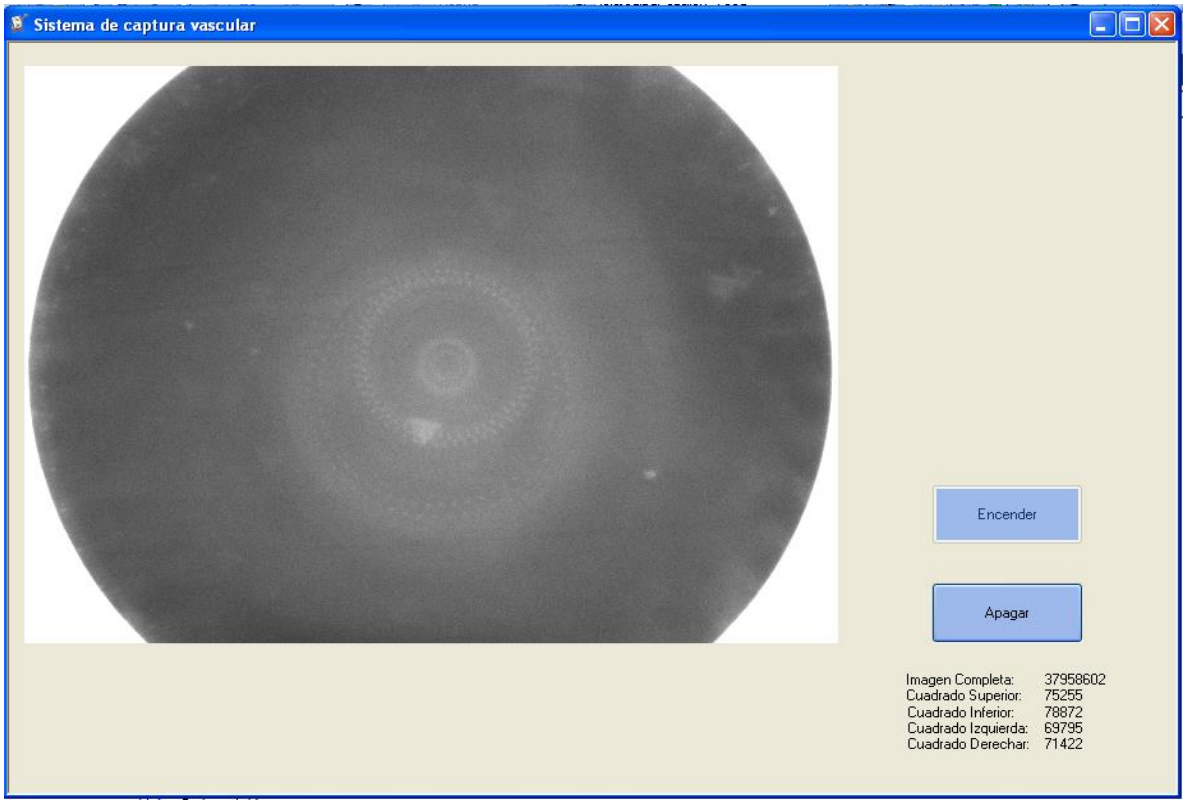

Fig. 37 – Interfaz gráfica final con LED encendidos

En la imagen 38, y en las posteriores imágenes, se va a mostrar lo que el usuario vería por pantalla si la posición de la mano no es la adecuada:

Fig. 38 – Posición errónea. Mano demasiado a la derecha

En la figura 38, la posición de la mano no es la adecuada. El programa lo detecta y emite las sugerencias correspondientes.

En esta imagen se puede observar cómo, pese a estar la mano colocada a la izquierda de la imagen, tanto la orden escrita como la visual (flecha verde), nos indican que movamos la mano hacia la izquierda. Esto se debe a que la máquina de captura de imágenes tiene la cámara girada 180º y su fabricación no se podía modificar; pero se ha tenido en cuenta, y las órdenes que da el programa son las adecuadas.

Lo mismo ocurrirá cuando coloquemos la mano al lado opuesto, tal y como se puede apreciar en la imagen 39.

Fig. 39 – Posición errónea. Mano demasiado a la izquierda

A continuación se probara introduciendo en el visor de la cámara sólo la punta de los dedos, y la parte final de la mano (no válidas para nuestro proyecto). Este supuesto ocurrirá cuando el usuario comience a introducir la mano para la captura de la imagen, y en ningún caso queremos que se captura le imagen antes de tener la palma de la mano colocada sobre el visor de la cámara. Esta prueba es de obligado cumplimiento para ver si es capaz de desechar dichas imágenes y no capturarlas.

Esta prueba se llevara a cabo en las imágenes 40 y 41.

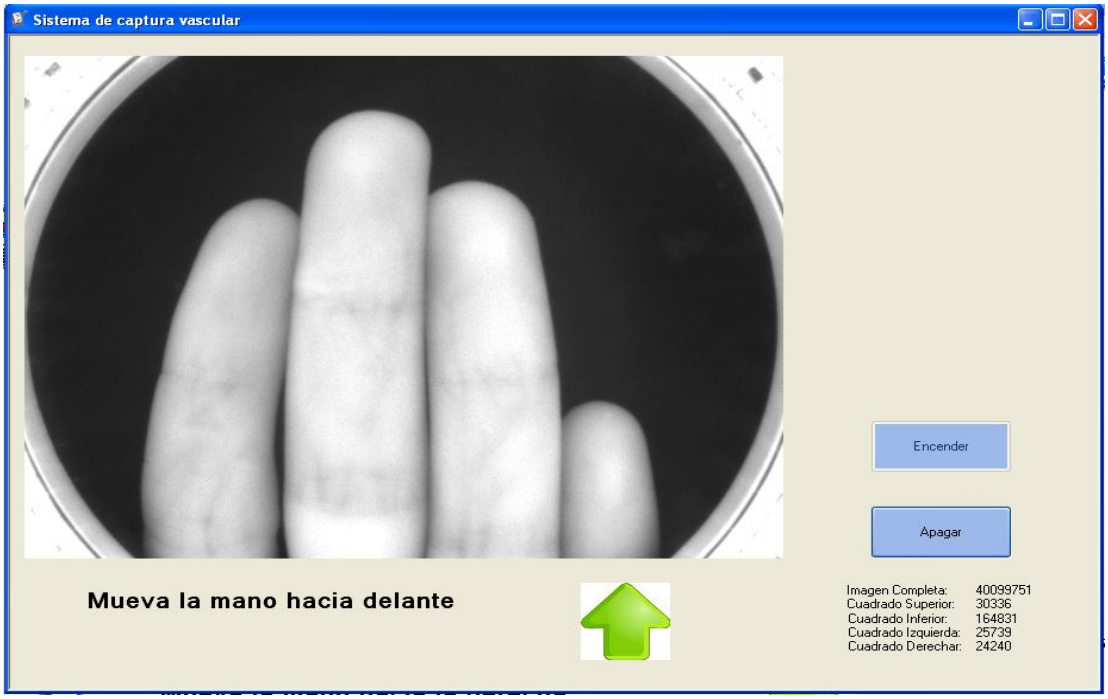

Fig. 40 – Posición errónea. Mano demasiado atrás

La prueba de la mano demasiado hacia delante no es muy representativa de la realidad, ya que el usuario no pasará la mano de largo sobre el sensor, pero para evitar posibles capturas erróneas provocadas por un usuario "no amistoso", también se ha tenido en cuenta, y la imagen no se capturará en este caso.

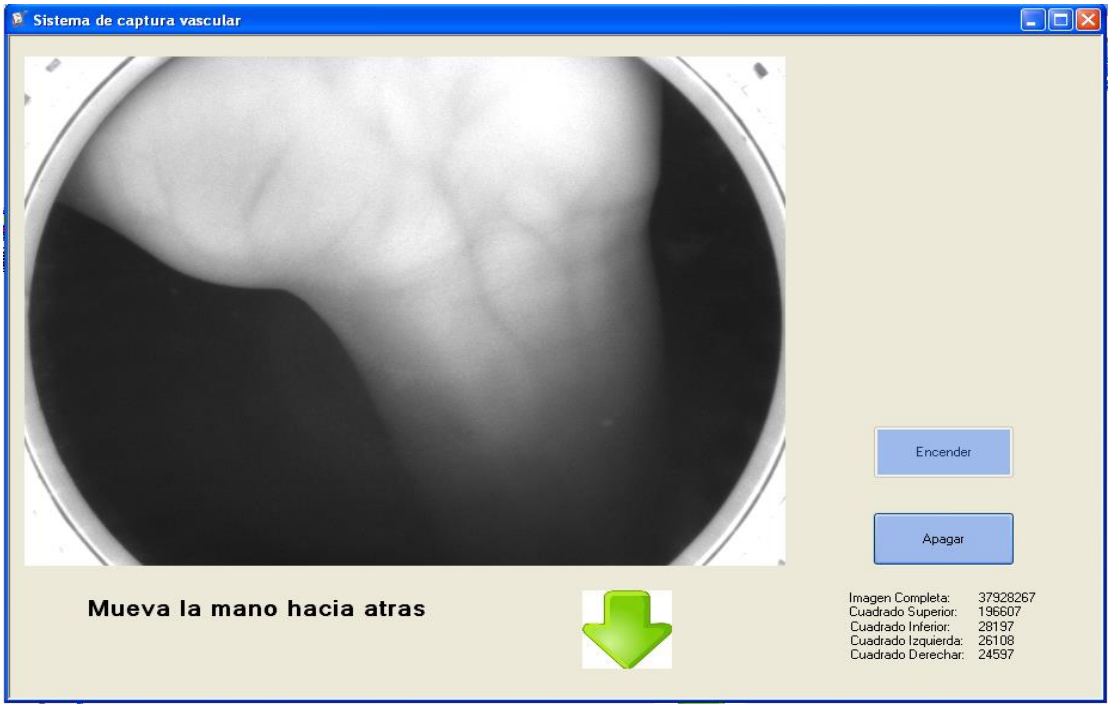

Fig. 41 – Posición errónea. Mano demasiado adelante

En estos casos (imágenes 40 y 41) el programa también actúa como se esperaba no capturando ninguna imagen, e indicando al usuario hacia donde tiene que desplazar la mano.

El programa también es capaz de emitir dos órdenes a la vez si la mano se encuentra mal colocada en las posiciones horizontal y vertical como puede observarse en la imagen 42.

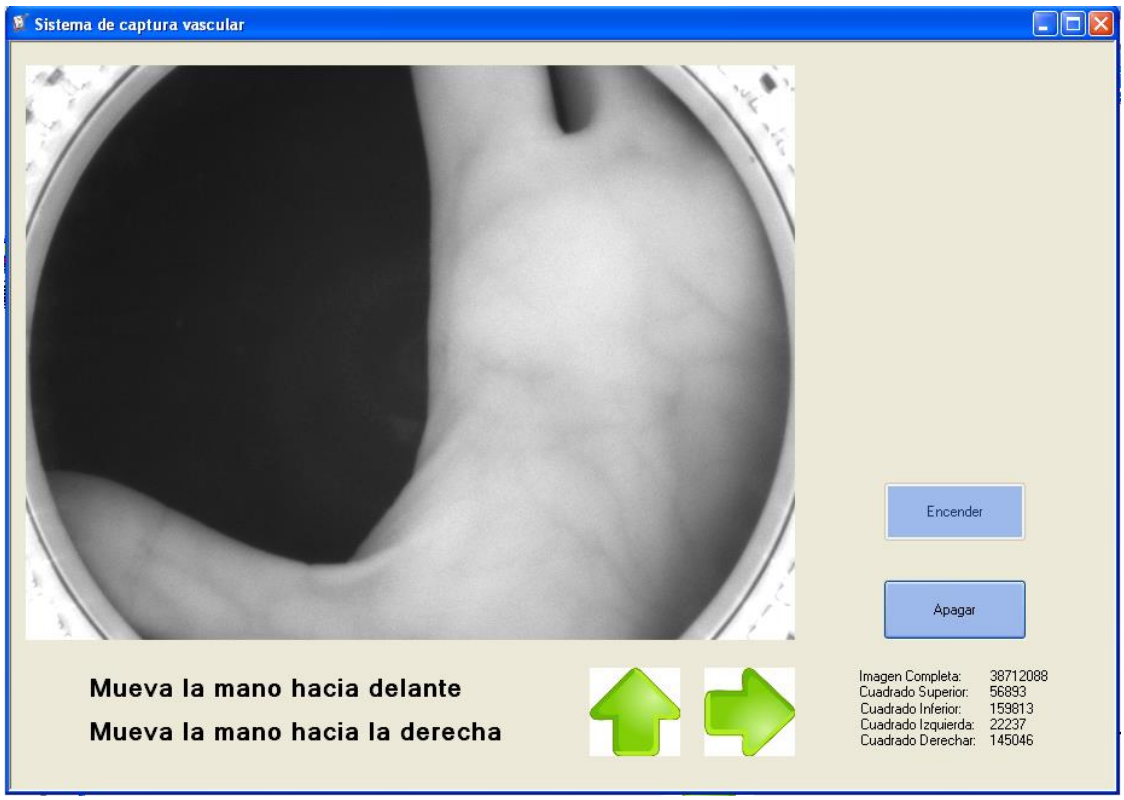

Fig. 42 – Posición errónea. Mano demasiado a la izquierda y atrás

Por último, el programa tampoco capturará la imagen si la mano no se encuentra a la distancia adecuada de la cámara, es decir, la mano debe estar colocada sobre el soporte para que pueda ser capturada, como se muestra en la figura 43.

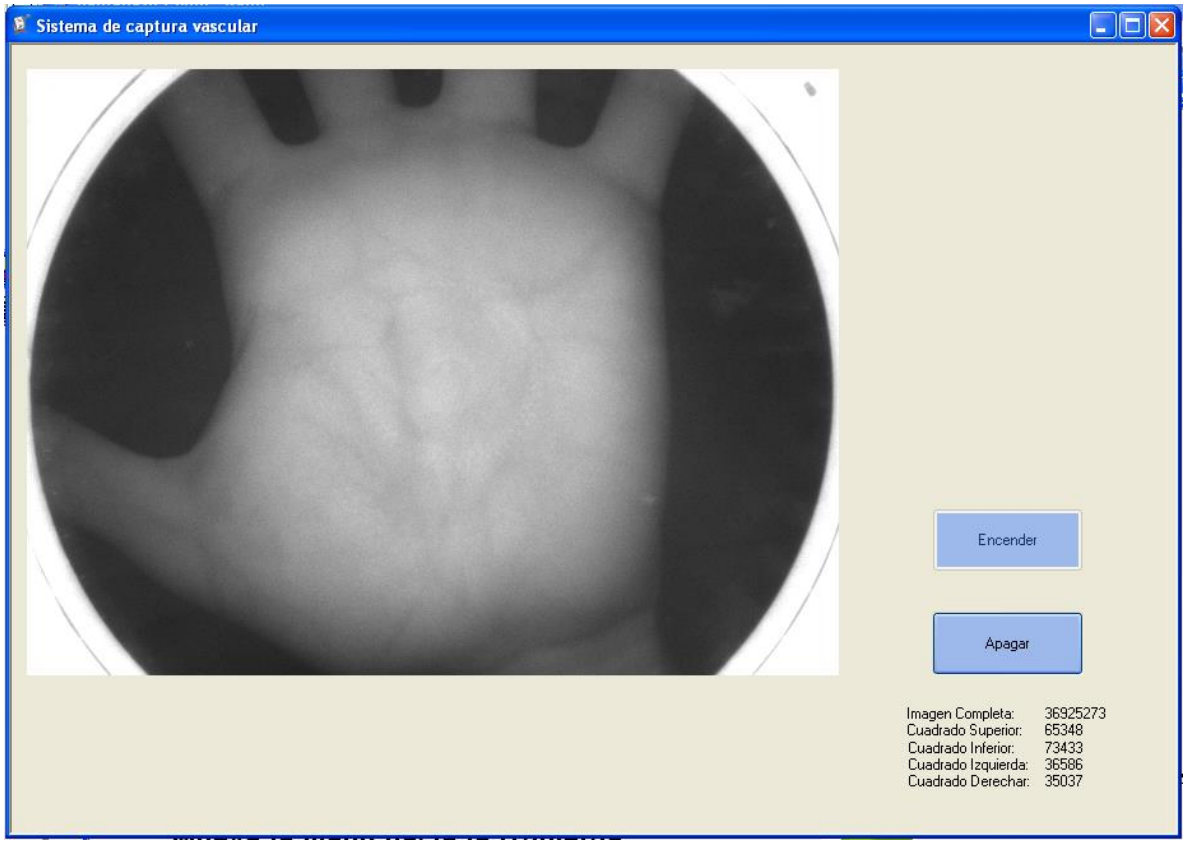

Fig. 43 – Posición errónea. Mano demasiado alejada

Una vez tenidos en cuenta todos los casos en los que la cámara no debe capturar la imagen, se seguirán las indicaciones del programa, hasta que la imagen quede capturada. Una vez hecha la captura de la imagen, el programa debería mostrar por pantalla (por medio de un picturebox), la imagen almacenada. Esta imagen es la que se muestra por pantalla cuando se han seguido todas las indicaciones que ofrece el programa:

Fig. 44 – Mano en posición correcta. Imagen Capturada

Como se observa en la figura 44, una vez que se han cumplido todas las especificaciones y condiciones de la imagen exigidas por el programa, la imagen queda registrada y guardada.

Para mayor tranquilidad del usuario, a parte de la confirmación de que la imagen ha sido capturada, como se puede ver en la figura, también se muestra la imagen que ha sido almacenada.

Una vez terminado el proceso de captura de la imagen, la imagen capturada se mostrará por pantalla durante 3 segundos. Trascurrido este tiempo, la cámara seguirá mostrando imágenes a tiempo real de lo que se ve en el objetivo, de modo que pueda ser utilizada por otro usuario.

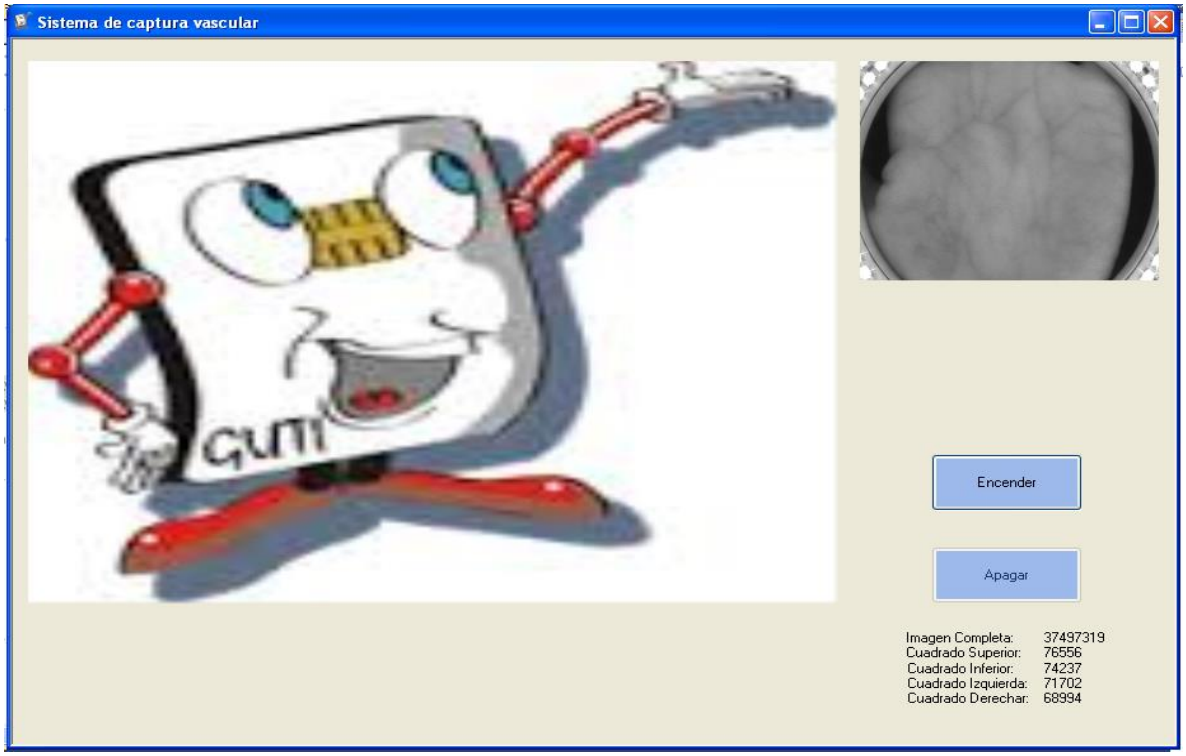

Fig. 45 – Programa preparado para volver a ser usado

Una vez finalizado el proceso de captura y pulsado el botón "Apagar", la cámara deja de emitir imágenes, y se muestra por pantalla el icono del GUTI (laboratorio en el cual se ha desarrollado dicho proyecto), quedando el programa listo para ser usado por el próximo usuario.

6 Conclusiones y líneas futuras de investigación

En el presente apartado se expondrán las conclusiones obtenidas tras la realización del TFG, así como las posibles líneas de investigación futuras y el uso que se puede obtener de esta aplicación.

6.1 Conclusiones

La principal conclusión que se obtiene tras la realización del TFG, es que se han llevado a cabo los objetivos que se perseguían al comienzo de éste: la automatización de un sensor de captura de imágenes vasculares, y la creación de una interfaz gráfica para que interactúe de manera simple pero efectiva con el usuario.

Los principales problemas surgidos durante el proceso de creación han estado referidos a la identificación y correcto tratamiento de los parámetros que hacen que una imagen sea válida para su captura. Una vez identificados estos parámetros de la imagen, se aplicó la formación obtenida en las asignaturas de programación estudiadas a lo largo del grado universitario.

Otro punto de inflexión en el presente TFG (una vez identificado el brillo como parámetro principal en el que se iba a basar todo el desarrollo del proyecto) fue como tratar este parámetro; es decir, como hacer que el sensor capturase las imágenes automáticamente basándose en el brillo de la imagen. Para ello se propusieron una serie de condiciones en base al brillo de la imagen general, y el de ciertas zonas de la imagen, comparando el brillo de esas zonas entre sí, para poder evaluar si la mano está colocada sobre el sensor de forma correcta y así poder capturarla.

Cuando el programa fue capaz de capturar las imágenes correctas de la palma de la mano, el paso siguiente fue la construcción de la interfaz gráfica, de modo que el usuario

pudiera utilizar el sensor de manera fácil y cómoda. Se llevó a cabo con indicaciones gráficas y de texto, siempre pensando en facilitar su uso.

Por último, y como conclusión personal, el alumno partía desde un desconocimiento casi absoluto del tema a tratar, y concluye el trabajo con los conocimientos y habilidades suficientes para poder seguir investigando en el terreno de las aplicaciones biométricas, por lo que su consideración es que se han cumplido todos los objetivos, tanto técnico como personales, especificados al inicio del presente documento.

6.2 Líneas futuras de investigación

El presente proyecto permitirá a los desarrolladores con conocimientos de C Sharp diseñar sus propios sistemas biométricos respetando los estándares que exige la industria. Más en concreto, el trabajo llevado a cabo en este TFG, servirá para desarrollar una aplicación completa de una nueva forma de identificación biométrica (identificación vascular).

Las posibilidades existentes en cuanto a aplicaciones son enormes, pero para tener una mayor funcionalidad, se debería desarrollar un software que comparase las imágenes obtenidas por la máquina, con las de una base de datos previamente creada.

Existe un algoritmo, desarrollado por Jaime Uriarte-Antonio, Daniel Hartung, J. Enrique Suarez Pascual y Raul Sanchez-Reillo desarrollado en 2011 [15], capaz de comparar el patrón de las venas de la mano de dos imágenes.

Este algoritmo de comparación utiliza la ubicación relativa de los puntos característicos y la orientación relativa de dichos puntos. El uso de mediciones relativas permite comparar dos patrones que no están normalizados en el mismo punto. El algoritmo necesita un tratamiento previo que sirve para extraer los puntos característicos. Y posteriormente, un código de comparación incluyendo el método de comparación de los característicos, como se observa en las imágenes 46 y 47.

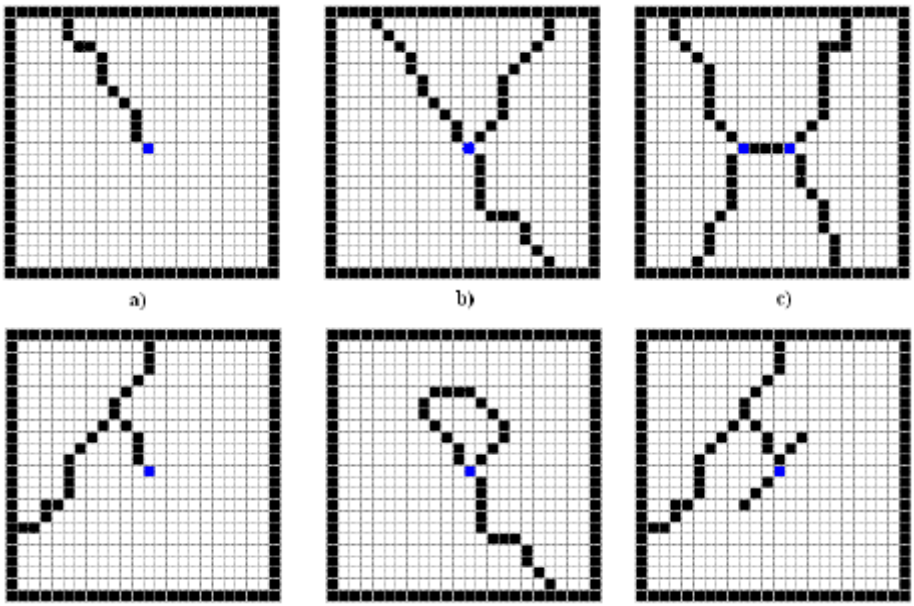

Fig. 46 – Extracción de puntos característicos [15]

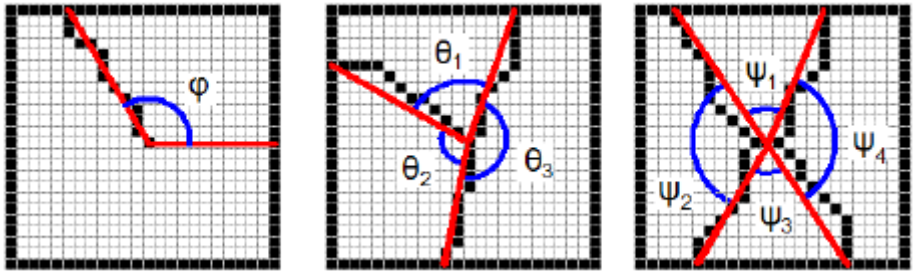

Fig. 47 – Comparación de los puntos característicos [15]

Si se combinasen, una base de datos con imágenes vasculares, este programa que compara las imágenes de la base de datos con la imagen capturada y el trabajo que se ha llevado a cabo en el presente TFG, se obtendría un sistema de identificación biométrico completo, con una robustez e inviolabilidad similar al sistema de identificación de iris, ya que la distribución de las venas de la mano no es variable ni susceptible de copia. Uno de sus principales usos sería como sistema de seguridad a la hora de permitir el acceso a ciertas zonas, solo aptas para las personas autorizadas.

Bibliografía

[1] SUAREZ PASCUAL, J.E. ; URIARTE-ANTONIO, J. ; SANCHEZ-REILLO, R. ; LORENZ, M.G. "Capturing Hand or Wrist Vein Images for Biometric Authentication Using Low-Cost Devices". Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing. Páginas 318-322. 2010.

[2] Historia de la biometría. En *Biometría*. Disponible en: http://www.biometria.gov.ar/acerca-de-la-biometria/historia-de-labiometria.aspx (30 de agosto de 2013).

[3] PEREZ, J.M. Biometrics error, 2005. Recuperado de http://commons.wikimedia.org/wiki/File:Biometrics_error.jpg?uselang=es (5 de septiembre de 2013).

[4] URIARTE-ANTONIO, J. ; SUAREZ-PASCUAL, J.E. ; GARCIA-LORENZ, M. ; SANCHEZ-REILLO, R. "Parametrical Study of a Vascular Biometric System". 2011 International Conference, Hand-Based Biometrics (ICHB). 2011.

[5] SÁNCHEZ REÍLLO, RAÚL. "Identificación biométrica y su unión con las tarjetas inteligentes". Revista SIC. Vol. 19. Agora (Abril 2000).

[6] Biometría. (2005, 10 de octubre). En Wikipedia, la enciclopedia libre. Disponible en: http://es.wikipedia.org/wiki/Biometr%C3%ADa (5 de septiembre de 2013).

[7] Historia de la Automatizacion. En Wikipedia, la enciclopedia libre. Disponible en: http://es.wikipedia.org/wiki/Automatizaci%C3%B3n_industrial (5 de septiembre de 2013).

[8] .NET (2007). En Wikipedia, la enciclopedia libre. Disponible en: http://es.wikipedia.org/wiki/Microsoft_.NET (8 de septiembre de 2013).

[9] Información general y conceptual sobre .NET Framework. Disponible en: http://msdn.microsoft.com/es-es/library/zw4w595w(v=vs.90).aspx. (8 de septiembre de 2013).

[10] Common Language Runtime. Imagen disponible en: http://www.techiegeex.com/dotNet/dn001.php (8 de septiembre de 2013).

[11] Fuente de alimentación continúa. Disponible en: http://es.aliexpress.com/wholesale/wholesale-high-power-110v-dc-power-supply.html (8 de septiembre de 2013).

[12] Representación del espectro electromagnético. Disponible en: http://www.inegi.org.mx/geo/contenidos/imgpercepcion/imgsatelite/elementos.aspx (8 de septiembre de 2013).

[13] S. Fantini and M. A. Franceschini, "Handbook of Optical Biomedical Diagnostics", chapter 7, SPIE Press, Bellingham, WA, 2002.

[14] Longitudes de onda del espectro electromagnético entorno al infrarrojo. Disponible en: http://enciclopedia.us.es/index.php/Espectro_electromagn%C3%A9tico (8 de septiembre de 2013).

[15] URIARTE-ANTONIO, J. ; HARTUNG, D. ; PASCUAL, J.E.S. ; SANCHEZ-REILLO, R. "Vascular Biometrics based on a Minutiae Extraction Approach". Security Technology (ICCST), 2011 IEEE International Carnahan Conference. (2011)

Anexo A: Planificación y Presupuesto

En el presente anexo se calculará de forma precisa y detallando cada coste por separado el valor del proyecto presentado a lo largo del documento. Se dividirá en costes de material y costes personales.

A.1 Costes materiales

Los costes materiales incluyen los costes de hardware y los recursos de oficina o fungibles. Dentro del hardware se incluye un ordenador portátil (necesario para ejecutar el entorno de desarrollo y para la búsqueda de información), los componentes necesarios para máquina de captura de imágenes (cámara, LED de infrarrojos, cúpula de homogenización y madera) y una fuente de alimentación. En cuanto a recursos fungibles, únicamente hay que tener en cuenta el acceso a internet.

			Costes (€)
Hardware	Ordenador		200
	Madera		20
	Cámara		300
	Iluminación	LED	900
		Cúpula de homogenización	
	Fuente de alimentación		120
Fungibles	Internet		30
Total			1.570

Tabla 4 – Costes Materiales

A.2 Costes de personal

Los costes de personal se han separado siguiendo dos parámetros diferentes: el primero, el tipo de personal, y el segundo, el apartado del proyecto. La primera división obedece a la diferencia de salarios existente entre el ingeniero y el director de proyecto, las dos personas que han intervenido en la creación. La segunda división es para detallar las horas empleadas de una forma más precisa, dada la amplitud del trabajo. La división de costes del proyecto se ha estructurado de la siguiente manera:

- **Planteamiento**: Incluye la familiarización con los entornos de desarrollo y los lenguajes de programación, la búsqueda de información sobre el proyecto y los tutoriales necesarios para llevarlo a cabo y el diseño de la solución.
- **Desarrollo**: Incluye los apartados de desarrollo del código y de la aplicación.
- **Pruebas**: Incluye las pruebas de funcionamiento del código y de la interfaz gráfica.
- **Redacción de la memoria**: Incluye tanto la redacción como la corrección de la misma.

		Ingeniero	Director Proyecto
Planteamiento	Familiarización (h)	20	0
	Búsqueda Información (h)	20	10
	Diseño (h)	50	0
Desarrollo	Desarrollo del código (h)	60	0
	Desarrollo de la aplicación (h)	40	10
Pruebas	Pruebas del código (h)	15	2
	Pruebas interfaz gráfica (h)	5	2
Memoria	Redacción (h)	65	0
	Corrección (h)	5	20
Total	Total (h)	280	44
	Salario / hora	30	50
	Salario total (€)	8.400	2.200

Tabla 5 – Costes de Personal

A.3 Costes totales

Sumando los dos apartados anteriores, añadiéndole los costes indirectos, de un veinte por ciento, y el IVA, de un veintiuno por ciento, se obtiene el precio total del proyecto:

	Coste (€)
Personal	10.600
Materiales	1.570
Costes indirectos (20%)	2.434
Subtotal	14.604
IVA (21%)	3.066,84
Total	17.670,84

Tabla 6 – Costes de Personal

El coste total del proyecto es de DIECISIETE MIL SETECIENTOS SETENTA EUROS CON OCHENTA Y CUATRO CÉNTIMOS.

Leganés, 5 de Septiembre de 2013

El ingeniero