Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

CONTROL DE ACCESO CON IDENTIFICACIÓN MEDIANTE IMÁGENES

Alumno: **Neftalí Ríos del Moral**

Tutor: Juan Carlos Cuevas Martínez
Depto.: Ingeniería de Telecomunicación

Septiembre, 2018

AGRADECIMIENTOS

A mi tutor, el Dr. Juan Carlos Cuevas Martínez, por permitirme trabajar con él, siempre dispuesto a recibirme y resolviendo cada duda. Así, todo es muy fácil.

DEDICATORIA

A mi madre, tantas y tantas veces pidiéndome que estudiara que ha conseguido que, aún hoy, con más edad que ella cuando me lo decía, siga estudiando y con ganas de más.

A mis preciosos hijos, y, sobre todo, a mi amada esposa, Ana, que tanto han sufrido mis “ausencias” este tiempo. Una vez acabado, ya puedo volver a centrarme, de nuevo, en el proyecto más importante para mí: vosotros.

RESUMEN

En este trabajo fin de grado se presenta el diseño e implementación de un sistema de control de acceso, centralizado y seguro, basado en el reconocimiento de parámetros biométricos, concretamente el reconocimiento facial, de cuyas técnicas más relevantes se expone un estudio comparativo. El sistema se compone de: un servidor central, responsable del reconocimiento y la gestión del sistema; cámaras situadas en cada espacio de un edificio cuyo acceso se desea controlar, responsables de captar el rostro de los usuarios y enviarlos al servidor; y de dispositivos actuadores, encargados de iniciar los procedimientos que correspondan en función del usuario reconocido y la configuración del sistema.

Todo el sistema ha sido diseñado de manera flexible, para permitir su uso en edificios destinados a diferentes fines y con diferentes niveles de seguridad. Asimismo, todas las comunicaciones entre los diferentes dispositivos que forman el sistema han sido cifradas para mantener la integridad de las mismas.

Palabras clave: Control de acceso, reconocimiento facial, sistema biométrico, comunicaciones cifradas.

ABSTRACT

This final degree project presents the design and implementation of a centralized and secure access control system based on the recognition of biometric parameters, specifically facial recognition, of which a comparative study presents the most relevant techniques. The system consists of a central server, the cameras and the actuator devices. The central server is the responsible of the recognition and management of the system. The cameras, which are located in each space of a building whose access is controlled, capture the users' faces and send them to the central server. The actuator devices are the responsible for initiating the corresponding procedures according to the recognized user and the configuration of the system.

The system has been designed with flexibility, to allow its use in buildings intended for different purposes and with different levels of security. Moreover, all communications between the different devices of the system are encrypted to maintain the integrity of the same.

Keywords: Access control, facial recognition, biometric system, encrypted communications.

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

ÍNDICE DE ABREVIATURAS

AES:	Estándar de encriptación avanzado
DES:	Estándar de encriptación de datos
FMR:	Tasa de falso positivo
FNMR:	Tasa de falso negativo
HOG:	Histograma de gradientes orientados
ISO:	Organización internacional de normalización
LBP:	Patrones locales binarios
LDA:	Análisis discriminante lineal
LGBPHS*:*	Secuencia de histogramas de patrones binarios locales de Gabor
LPP:	Proyecciones de preservación local
MSS:	Tamaño máximo de segmento
MTU:	Tamaño máximo de trama
PCA:	Análisis de componentes principales
PGM:	Imagen portable en grises
ROI:	Región de interés
SDK:	Kit de desarrollo de software
SIFT:	Transformación de características en escala invariable
SSL:	Capa de puertos seguros
SVM:	Máquina de vectores de soporte
TEA:	Algoritmo de encriptación ligero
TLS:	Seguridad de la capa de transporte
XML:	Lenguaje de marcas extensible
XTEA:	TEA extendido
XXTEA:	XTEA extendido

1 INTRODUCCIÓN

La biometría se ha convertido en una herramienta muy útil para la seguridad, sobre todo cuando se trata de controlar el acceso a edificios o dependencias. Conseguir su implementación de manera robusta, segura y con garantías de éxito, es un reto actual.

En este trabajo se ha desarrollado un sistema que permite controlar y gestionar el acceso de personas a diferentes espacios o servicios de un edificio. Para ello, y después de valorar las diferentes alternativas, se ha optado por utilizar algoritmos de reconocimiento facial. El estudio realizado justificando esta decisión se incluye en el presente trabajo fin de grado.

El sistema diseñado está compuesto por un servidor central y dos tipos de clientes, capturadores y actuadores. Así, un dispositivo capturador es aquel que está equipado con una cámara, captura la imagen de un usuario que intenta acceder a un espacio o usar un servicio y envía dicha imagen al servidor, después de ser procesada mediante técnicas de visión por computador. Una vez el servidor recibe la imagen, utilizará algoritmos de reconocimiento facial para identificar al usuario. Si la identificación es positiva, y el usuario se encuentra autorizado, se inician todos los procesos ligados a este acceso, que pueden ser tan dispares como la apertura de una puerta, el encendido de luces o permitir usar una máquina de refrescos. Estos procesos son iniciados por el servidor mediante el envío de los mensajes apropiados a uno o varios actuadores, según esté configurado.

Un factor determinante es la seguridad del sistema. Por un lado, se debe garantizar, en la medida de lo posible, que la captura es fiable y que la identificación del usuario es correcta. La tecnología y los algoritmos actuales presentan altos porcentajes de acierto, pero lejos del 100%, por lo que se han incluido métodos suplementarios de identificación, como la lectura de huella dactilar, para aquellos casos en que sea necesario aumentar la seguridad. Por otro lado, es imprescindible que las comunicaciones sean seguras, por lo que se han utilizado algoritmos de cifrado que dotan de seguridad al sistema. Se han realizado y documentado las pruebas necesarias que demuestran el riesgo que implica no usar seguridad en las comunicaciones.

El sistema, del cual puede verse un esquema en la Figura 1, ha sido implementado a nivel de prototipo, incluyendo el servidor central, diversos dispositivos capturadores y actuadores; los cuales serán descritos con más detalle en los capítulos de DISEÑO DEL PROYECTO e IMPLEMENTACIÓN. Además, ha sido diseñado para permitir flexibilidad en su configuración con objetivo de poder adaptarlo a diferentes entornos, en función de las necesidades.

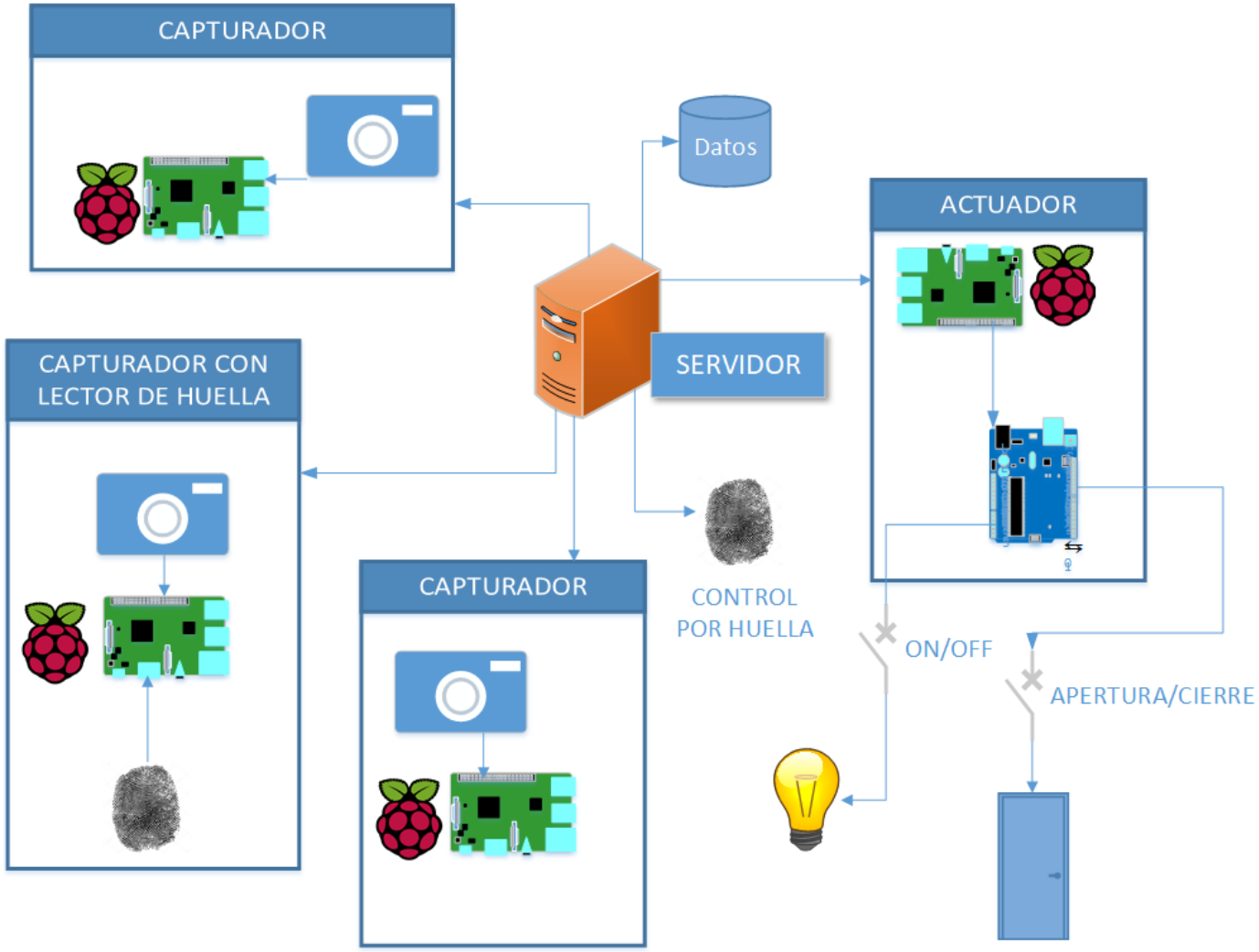

Figura 1 Esquema general del sistema

1.1 Estructura del trabajo

El presente trabajo ha sido estructurado en los siguientes capítulos:

1. **Introducción**. Breve introducción del sistema diseñado.
2. **Estudio del reconocimiento por imágenes**. Se realiza un repaso de las diferentes alternativas para el reconocimiento biométrico mediante imágenes. Especialmente en aquellas que se han considerado para el presente trabajo.
3. **Estado del arte**. Se mencionan diferentes alternativas disponibles actualmente para las tecnologías evaluadas.
4. **Diseño**. Las tecnologías elegidas y los motivos de su elección serán presentadas en este capítulo, además, se describe con detalle cómo se ha estructurado el sistema y qué posibilidades ofrece, los diferentes dispositivos utilizados y las funciones que cada uno realiza.
5. **Implementación**. En el capítulo previo se ha analizado qué hace el sistema y ahora se detalla cómo lo hace. Se detallan los archivos en los que se divide el trabajo y como logran los objetivos particulares de cada uno de ellos.
6. **Pruebas y resultados.** Para evaluar y testear el sistema diseñado se han realizado una serie de pruebas cuyos resultados permiten conocer las limitaciones del mismo y ofrecer recomendaciones que permitan implementarlo y explotarlo de forma satisfactoria.
7. **Pliego de condiciones**. Estándares y legislación que deben ser tenidos en consideración a la hora de implantar el sistema serán revisados en este capítulo.
8. **Estudio económico**. Se ofrecen una valoración aproximada del coste de cada dispositivo y el presupuesto del proyecto desarrollado.
9. **Conclusiones y Líneas futuras**. En este capítulo se evalúa el resultado final del proyecto considerando los objetivos inicialmente marcados. Además se describen una serie de posibles mejoras que no han podido ser abordadas en este proyecto.
10. **Anexos**. Donde se incluyen manuales de usuario y administrador, resultados obtenidos en las pruebas y otros datos relevantes que no se hayan incluido en la memoria.

1.2 Objetivos

El objetivo principal de este proyecto es el diseño de un sistema centralizado para el control de acceso a estancias basado en el reconocimiento por imágenes mediante algún parámetro biométrico. Además:

- El sistema deberá tener su información centralizada y poder ser gestionada a través de un portal de administración en un servidor.
- Se deberán diseñar e implementar los dispositivos necesarios para poder realizar la captación de imágenes, su procesado y envío de información al servidor central.
- El servicio de control de acceso deberá contar con las medidas de seguridad apropiadas para proteger la información y las comunicaciones.

Como objetivo secundario se tiene que la identificación correcta de un sujeto pueda desencadenar acciones personalizadas en el entorno en función de una programación previa.

2 ESTUDIO DEL RECONOCIMIENTO POR IMÁGENES

En este capítulo se presenta un breve análisis de las diferentes opciones disponibles para realizar un reconocimiento por imágenes haciendo uso de algún parámetro biométrico. El objeto fundamental es presentar las técnicas más comunes y empleadas en los sistemas de acceso, información que servirá como base para fundamentar las elecciones al respecto en el presente trabajo.

2.1 Biometría

Se podría definir la biometría como el conjunto de técnicas y métodos estadísticos que permiten analizar determinadas características físicas o de comportamiento para identificar y autenticar personas.

En función de la característica física que se emplee para la identificación se pueden distinguir:

- Huella dactilar
- Geometría de la mano
- Iris del ojo
- Retina del ojo
- Venas de la mano, del brazo, de los dedos, del dorso de la mano, etc.
- Geometría de la cara

Existen otras alternativas, menos utilizadas, como los poros de la piel o, incluso, el olor corporal. Aunque no todas tienen la misma aceptación, ni ofrecen las mismas garantías de éxito.

Por otro lado, existen las de comportamiento o biometría dinámica, entre las que destacan la voz, el texto manuscrito, la forma de andar o de teclear, entre otras.

Desde el punto de vista del funcionamiento se pueden distinguir dos usos:

- Autenticación. Se comprueba si el usuario se corresponde con un usuario concreto registrado en el sistema. Es un sistema 1:1.
- Identificación. Se comprueba si la característica estudiada en el usuario se corresponde con la de algún usuario registrado y cual. Se trata de un sistema 1:N.

Para poder ser utilizada como técnica biométrica es conveniente que la característica estudiada reúna las siguientes propiedades:

- **Universalidad**. Todas las personas deben tenerla.
- **Unicidad**. No se puede repetir o, al menos, la posibilidad de que ocurra debe ser ínfima, en definitiva, debe ser única en cada persona.

- **Permanencia**. La característica es invariante en el tiempo, o, al menos, en periodos largos de tiempo. El reconocimiento facial sí que sufre en este sentido, pero los cambios son lo suficientemente lentos en la gran mayoría de los casos. Para solventarlo es suficiente con actualizar los registros cada cierto tiempo.
- **Mensurabilidad**. Se puede medir la característica.

Además de las características mencionadas, existen otras deseables como la aceptabilidad de la técnica, el rendimiento y la dificultad de usurpación.

2.1.1 Conceptos clave en Biometría

Antes de continuar es conveniente mencionar algunos conceptos claves en biometría. (INTECO 2012)

- **Sensor**. Dispositivo que recoge las características biométricas. Puede ser una cámara, un lector de huellas o un teclado.
- **Inscripción**. Cuando se recogen datos biométricos de un individuo para almacenarlos para su posterior uso para autenticar o identificar al usuario.
- **Identificación**. Procedimiento mediante el cual se comparan las características biométricas extraídas del usuario y se comparan con todas las almacenadas.
- **Autenticación**. Es un proceso similar a la identificación, salvo que en este caso la comparación se realiza únicamente con un único registro almacenado entre los previamente inscritos. Es necesario un proceso previo para conocer la identidad del registro con el que se debe comparar.
- **Umbral** (en inglés *threshold*). Valor que establece el límite para la distancia o diferencia, en función de la técnica biométrica, a partir del cual un reconocimiento se entiende como positivo.
- **Tasa de error en la adquisición**. Se trata de la proporción de veces en las que la adquisición no es válida para su posterior procesado.
- **Tasa de error de registro**. Proporción de la población a la que no es posible extraerlo muestras de calidad.
- **Tasa de falso negativo** (*False Non Match Rate*, FNMR o *False Rejection Rate*, FRR). Probabilidad de que un usuario no sea identificado con su propia plantilla ya existente en el registro.
- **Tasa de falso positivo** (*False Match Rate*, FMR o *False Acceptance Rate,* FAR). Probabilidad de que un individuo sea asociado, por error, con otro ya registrado.

- **Tasa de igual error** (*Equal Error Rate*, EER). Coincide con el valor de umbral en el que las tasas de falso positivo y falso negativo coinciden.
- **Curvas ROC** (Receiver Operator Characteristic Curve). Representan la tasa de verdaderos positivos frente a la tasa de falsos positivos según se varía el valor de umbral.

2.2 Reconocimiento Facial

2.2.1 Detección facial

La detección facial, que permite determinar si en una imagen concreta existe un rostro humano, no es una técnica biométrica, pero es necesaria para la posterior identificación. Para que se produzca la detección es necesario evaluar la imagen en busca de patrones que sean indicativos de un rostro. Para ello se hacen uso de descriptores, que se utilizan para caracterizar imágenes digitales. Para conseguir estos descriptores es necesario utilizar técnicas que reduzcan la dimensionalidad[1] de la imagen, y en función de la técnica utilizada se disponen de diferentes métodos de detección facial. Se van a describir dos de las técnicas más utilizadas en detección facial.

2.2.1.1 Descriptores Haar y Algoritmo de Viola-Jones

Alfred Haar presentó en 1910 lo que ahora se conocen como primer Wavelet o Wavelet de Haar (Haar 1910). En 1998 Papageorgiou y otros (Papageorgiou, Oren y Poggio 1998) presentan un trabajo sobre detección de objetos en imágenes en el que se presentan las funciones base o características de Haar. Schapire y Freund desarrollan *Adaboost* (Freund y Schapire 1995), mejorado en 1999 (Schapire y Singer 1999). *Adaboost* o *boosting* adaptado se basa en utilizar clasificadores simples para obtener un clasificador robusto.

En 2001 Paul Viola y Michael Jones presentaron el conocido **algoritmo de Viola-Jones** (Viola y Jones 2001) para una rápida detección de objetos, en el que se realizaron contribuciones clave para el desarrollo de la detección facial, como la definición de lo que ellos llamaron la imagen integral, la presentación de un algoritmo de aprendizaje, basado en *Adaboost*, que selecciona un pequeño número de características críticas permitiendo obtener clasificadores más eficientes, y la combinación de estos clasificadores en una cascada en la que a cada paso se van descartando partes de la imagen para centrarse en las partes más prometedoras.

Aunque tiene más aplicaciones, ha sido en el campo de la detección facial donde ha cambiado por completo el panorama. Aumenta enormemente la velocidad de detección,

[1] En Análisis estadístico se entiende por reducir la dimensionalidad al proceso por el que el número de variables aleatorias existente se reduce para simplificar los cálculos.

permite realizar detecciones casi en tiempo real y posibilita la incorporación de esta funcionalidad en dispositivos tan sencillos como una cámara de fotos o un teléfono móvil.

Viola-Jones utiliza los conocidos como *Haar-like feature* o **características tipo Haar**, que son los elementos básicos para la detección, basados en el Wavelet de Haar, y que sirven como clasificadores débiles. Estas características se calculan sobre la imagen, en escala de grises. Se van utilizando rectángulos de pixeles de tamaño determinado y asignando un determinado peso a cada uno para obtener el valor de la característica.

En 2002 Lienhart y Maydt (Lienhart y Maydt 2002) proponen rotar las características para una mejor detección.

En 2003 Viola y Jones presentaron una extensión de su anterior trabajo (Jones y Viola 2003). Un ejemplo de características tipo Haar se puede apreciar en la Figura 2.

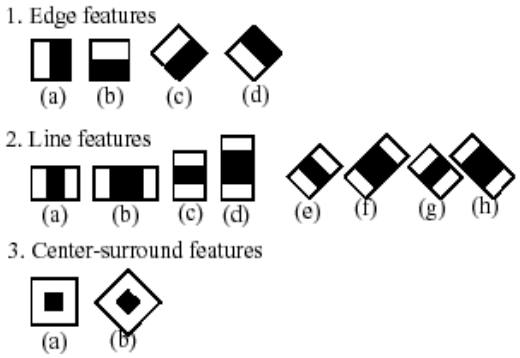

Figura 2 Características tipo Haar.

Fuente: («OpenCV library» 2018a)

2.2.1.2 Descriptores HOG

Una alternativa al algoritmo de Viola-Jones es el uso de los conocidos como HOG (*Histogram of Oriented Gradients*) (Dalal y Triggs 2005). Se utiliza el gradiente de la imagen, que es dividida en celdas, para obtener contornos, posteriormente se hacen histogramas y se combinan. («Histogram of Oriented Gradients | Learn OpenCV» 2018).

Para la clasificación de las imágenes se puede utilizar, por ejemplo, un clasificador SVM (*Support Vector Machine*) (Rosebrock 2014). Se trata de un algoritmo que permite encontrar clasificadores lineales en espacios transformados, desarrollado por Vapnik (Cortes y Vapnik 1995).

2.2.2 Introducción al reconocimiento facial

En 1971 se presentó un trabajo en el que se proponía la utilización de 22 marcadores faciales para definir los rostros humanos (Goldstein, Harmon y Lesk 1971).

El primer sistema de reconocimiento facial automático fue desarrollado por Tadeo Kanade (Kanade 1973). La limitación computacional de la época no permitió su expansión hasta que se propone utilizar los análisis de componentes principales en 1987.

Cox (Cox, Ghosn y Yianilos 1996) propuso un sistema de 35 puntos faciales extraídos de forma manual con buenos resultados. Estos puntos pueden apreciarse en la Figura 3.

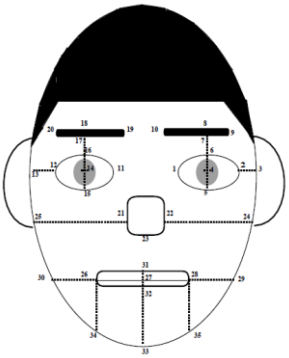

Figura 3 Puntos faciales propuestos por Cox.

Fuente: (Cox, Ghosn y Yianilos 1996)

La identificación facial automática permite identificar un usuario dentro de un conjunto de usuarios registrados mediante la comparación de una imagen del rostro con las diferentes imágenes utilizadas en el entrenamiento del sistema. Existen dos alternativas a la hora de realizar las comparaciones, se puede utilizar el conjunto de todas las características del rostro, enfoque holístico, o utilizar diferentes partes del rostro de manera individual. En función del enfoque y de los algoritmos utilizados para reducir la dimensionalidad y simplificar los cálculos, existen diferentes métodos de reconocimiento facial. En cualquier caso, en la mayoría de ellos es necesaria una fase de entrenamiento previa con las imágenes de los usuarios ya registrados.

En los últimos años han empezado a surgir alternativas con nuevos enfoques, como el uso de imágenes 3D en lugar de 2D, imágenes con profundidad, etc.

En los siguientes apartados se exponen algunos de los algoritmos más utilizados actualmente en el reconocimiento facial.

2.2.3 Eigenfaces

Este método se basa en el **análisis de componentes principales** o *PCA (Principal Component Analysis*), cuyo objetivo es reducir el número de dimensiones con la menor perdida de información. Técnica ampliamente utilizada en estadística para análisis de datos multivariantes y que permite reducir el número de variables simplificando el sistema.

El método PCA fue propuesto por Karl Pearson en 1901 (Pearson 1901), y desarrollado por Hotelling en 1933 (Hotelling 1933). En 1987 Kirby y Sirovich (Sirovich y Kirby 1987) emplean con éxito el análisis de componentes principales para el reconocimiento facial.

Posteriormente Turk y Pentland (Turk y Pentland 1991) hacen uso de una nueva técnica, que ya denominan **Eigenfaces**, para el reconocimiento facial.

Gottumukkal diseño un sistema en tiempo real utilizando la técnica de Eigenfaces en 2003 (Gottumukkal y Asari 2004).

El matemático Hilbert fue el primero en utilizar, en 1903, el término eigen (traducido posteriormente al español como propio o auto) que se utiliza para definir los vectores propios, autovectores o eigenvectores que se utilizan en el cálculo del PCA junto con los autovalores. En el caso de la identificación facial estos vectores propios tienen la misma dimensión que la imagen original y son denominados Eigenfaces, de ahí el nombre de esta técnica. Aproximadamente a partir de 300 eigenvectores se podría empezar a tener una cara reconocible («Face Recognition with OpenCV — OpenCV 2.4.13.6 documentation» 2018) haciendo uso de la base de datos de caras de AT&T, aunque este valor depende de los datos de entrada; en la Figura 4 se muestra un ejemplo gráfico. Una reducción considerable, teniendo en cuenta que una imagen de, por ejemplo, 96x96 pixeles tendría 9216 dimensiones.

Se trata de una técnica con enfoque holístico.

Figura 4 Evolución de Eigenfaces.

Fuente: («OpenCV library» 2018b)

2.2.4 Fisherfaces

En este caso es el **Discriminante lineal de Fisher** o *LDA (Linear Discriminant Analysis*), propuesto por Ronald Aylmer Fisher (Fisher 1936), la técnica estadística que permite un nuevo algoritmo de reconocimiento facial. Esta técnica persigue maximizar la varianza entre clases y minimizar la de cada clase, en definitiva, la mejor discriminación entre clases. Para, de forma similar al PCA, reducir el número de dimensiones a considerar.

Fisherfaces (Belhumeur, Hespanha y Kriegman 1997) (Zhao et al. 1998) utiliza una matriz específica de transformación y no captura tan bien la iluminación como Eigenfaces. Por lo que la iluminación le afecta en menor medida.

La reconstrucción en Fisherfaces difiere mucho de la de Eigenfaces, como se puede apreciar en la Figura 5.

Figura 5 Evolución de Fisherfaces.

Fuente: («OpenCV library» 2018a)

Es importante destacar que esta técnica requiere la existencia de, al menos, dos usuarios para el entrenamiento. A diferencia de las otras técnicas que solo requieren uno.

De nuevo se trata una técnica con enfoque holístico.

2.2.5 Local Binary Pattern

En 1990 (He y Wang 1990) fue propuesto el **patrón local binario** o *LBP (Local Binary Pattern*) como clasificador para visión por computador y en 2002 fue descrito en profundidad (Ojala, Pietikainen y Maenpaa 2002).

LBP es un descriptor de texturas. Cada pixel recibe una etiqueta en función de su valor y el de los pixeles adyacentes. Primero, los vecinos son etiquetados como uno o cero, según sea su valor mayor o menor que el pixel central, la secuencia de pixeles forma un valor binario que pasa a ser el valor LBP del pixel central al pasarlo a decimal. Es un proceso bastante rápido (Alvarado Moreno 2012)

En 2006 se utiliza para reconocimiento facial (Ahonen, Hadid y Pietikainen 2006) y en 2009 (Wang, Han y Yan 2009) se introduce el uso de histograma de gradiente con múltiples orientaciones para mejorar los resultados. Esta técnica utiliza los patrones locales binarios vistos, el rostro se divide en partes a las que se les aplica un histograma (de ahí el nombre de la técnica) y se obtiene el operador que describe la información de la región y que permite comparaciones para reconocer rostros. Por lo que esta técnica, a diferencia de las anteriores, no utiliza un enfoque holístico.

2.2.6 Otras técnicas

Existen otras técnicas de identificación facial, menos empleadas y documentadas:

- *Locality Preserving Projections* (LPP) (He y Niyogi 2004) (Cai, He y Han 2005) otra técnica de reducción de la dimensionalidad.
- *Laplacianfaces* (He et al. 2005)
- *Orthogonal Laplacianfaces* (Cai et al. 2006)
- *Convolutional Neural Network* (Li et al. 2015)
- *Scale Invariant Feature Transform* (*SIFT*)
- *Local Binary Pattern Histrogram Sequences (LGBPHS). Utiliza LBPH y wavelets de Gabor* (Zhang et al. 2005)

2.2.7 Comparación de técnicas de reconocimiento facial

No forma parte del objetivo del presente trabajo fin de grado realizar una comparación exhaustiva de las diferentes técnicas de reconocimiento facial disponibles. No obstante, ya existen multitud de trabajos que realizan estas comparaciones y cuyos resultados pueden ser útiles y ser tomados en consideración, por lo que han sido evaluados algunos de ellos:

Delbiaggio (Delbiaggio 2017) realiza en su tesis un profundo trabajo de comparación de algunas de las principales técnicas de reconocimiento facial. Presenta un estudio comparativo de las técnicas LBPH, Eigenfaces y Fisherfaces utilizando la librería OpenCV, además del software OpenFace («OpenFace» 2017), que utiliza una red convolucional neural. Los resultados de OpenFace son ligeramente superiores; obteniendo muy buenos resultados con todas las técnicas, en un entorno similar para todas las imágenes, las de entrenamiento y las de testeo. Sin embargo, los resultados obtenidos con OpenFace son superiores si las condiciones del entorno son variantes.

De especial relevancia es el trabajo de Belhumeur y otros (Belhumeur, Hespanha y Kriegman 1997) que compara las técnicas de Eigenfaces y Fisherfaces. Obteniendo mejores resultados con Fisherfaces que con Eigenfaces, cuando hay variaciones de luz y de expresión facial.

En el trabajo de SudhaNarang (SudhaNarang y MeghaSaxena 2018) se comparan los algoritmos implementados en la librería OpenCV para hacer reconocimiento en tiempo real, considerando la probabilidad de error, obteniendo mejores resultados con LBPH.

En el trabajo de Tesillo (Gómez y Mayumi 2016) se realiza una comparación de las técnicas Fisherfaces y LBPH, obteniendo mejores resultados con Fisherfaces cuando se producen cambios en la iluminación o en la pose del usuario, similar resultado se obtiene en el trabajo de Serrano (Serrano et al. 2018) al comparar Eigenfaces y Fisherfaces. Sin

embargo, Özdil (Özdil y Özbilen 2014) obtiene mejores resultados con LBPH, comparándolo con Eigenfaces y Fisherfaces.

Henry Arguello (Fuentes 2011) realiza un trabajo de recopilación en el que compara diferentes combinaciones de técnicas de reconocimiento facial, con las que se obtienen mejores resultados que con una sola técnica. Corregir los cambios en iluminación, escalado y alineación en una etapa de preprocesado aumenta considerablemente el desempeño de los sistemas. La combinación de Eigenfaces y red neuronal artificial es la que, según el estudio, parece tener mejor desempeño.

En general, Fisherfaces, según la revisión realizada de la bibliografía, parece obtener mejores resultados que Eigenfaces, sobre todo cuando hay ligeros cambios en la iluminación. Desde el punto de vista estadístico estos resultados serían los esperados (Anaya-Isaza et al. 2017), considerando el carácter discriminatorio de LDA, utilizado en Fisherfaces. Sin embargo, en determinadas circunstancias, como disponer de un reducido número de muestras por grupo (imágenes por usuario) los resultados podrían cambiar, y ser favorables al PCA, utilizado en Eigenfaces (Martínez y Kak 2001).

Es complicado evaluar, en cualquier caso, qué técnica es la más adecuada, pues son muchos los factores que afectan a los resultados. Aunque hay consideraciones más allá de la eficacia de una u otra técnica, como por ejemplo el hecho de que las técnicas con enfoque holístico requieren un nuevo entrenamiento cada vez que el sistema es modificado, es decir, en un sistema ya entrenado, la inclusión de una nueva imagen o un nuevo usuario, requeriría realizar nuevamente el entrenamiento completo, a diferencia de las técnicas no holísticas, que no necesitan reiniciar el entreno para seguir funcionando.

2.2.8 Proceso del reconocimiento facial por imágenes

Se pueden distinguir diferentes etapas en un proceso de reconocimiento facial. Cada una de estas etapas deberá ser implementada en el sistema diseñado.

1. **Captura**. En primer lugar, es necesario que el sistema sea capaz de capturar una imagen. Tanto el servidor como los clientes deberán incorporar cámaras compatibles que sean capaces de realizar capturas que reúnan las condiciones mínimas exigidas para conseguir una imagen adecuada.
2. **Detección**. En esta fase se intenta encontrar áreas de la imagen capturada en las que el sistema considera probable que haya un rostro. Para lo que se utilizará alguno de los algoritmos de detección facial.
3. **Normalización**. Una vez se ha detectado un rostro, el siguiente paso es normalizar la imagen. El objetivo es que todas las imágenes con las que trabaja el sistema tengan similares características. Dentro de los procesos habituales incluidos en esta etapa se encuentran el escalado, para que todas las imágenes tengan el mismo tamaño, el recorte, para que se trabaje

exclusivamente con la región de interés o *ROI (Region Of Interest)*, la rotación, para que los rasgos faciales se encuentren alineados, utilizándose habitualmente las coordenadas de los ojos en la imagen como puntos de referencia. También se utilizan otras técnicas como la normalización de histogramas para reducir los efectos producidos por diferentes iluminaciones y contrastes.

4. **Extracción de características**. La imagen es procesada para extraer las características en función del algoritmo de reconocimiento utilizado.
5. **Comparación**. Por último, se realiza una comparación con las características extraídas de las imágenes almacenadas en la base de datos para determinar qué usuario es el que se corresponde. Las técnicas de comparación utilizan las distancias entre las características extraídas, como la Euclídea o la de Mahalanobis, o técnicas de clasificación, como una máquina de vectores soporte o *SVM (Support Vector Machine*).

2.2.9 Calidad de imagen en el reconocimiento facial por imágenes

En la **ISO/IEC 19794-5** (Normalización 2006) («ISO/IEC 19794-5:2005 - Information technology -- Biometric data interchange formats -- Part 5: Face image data» 2006) de 2005, revisado en 2011 («ISO/IEC 19794-5:2011 - Information technology -- Biometric data interchange formats -- Part 5: Face image data» 2012), se definen algunos parámetros de estandarización sobre la calidad de las imágenes de rostros para aplicaciones de reconocimiento facial.

Existen trabajos, como el de Heydi Méndez (Vazquez et al. 2012) que evalúa nueve de los requisitos que establece la norma ISO, el de Marciniak (Marciniak et al. 2015) que evalúa cómo afecta una baja resolución, otro de Boom (Boom et al. 2006), o un documento de la Interpol («File» 2018) que evalúan los requisitos en cuanto a calidad de una imagen.

En la Figura 6 y en la Figura 7 se muestran ejemplos de imágenes y un algoritmo de evaluación de la calidad de la imagen.

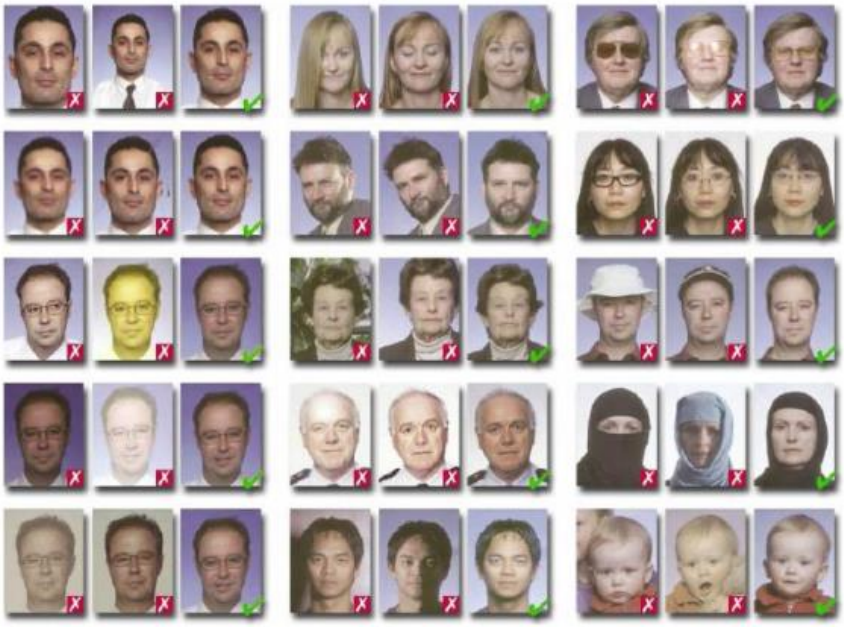

Figura 6 Cumplimiento de requisitos ISO 19794-5.

Fuente: (Méndez-Vázquez et al. 2012)

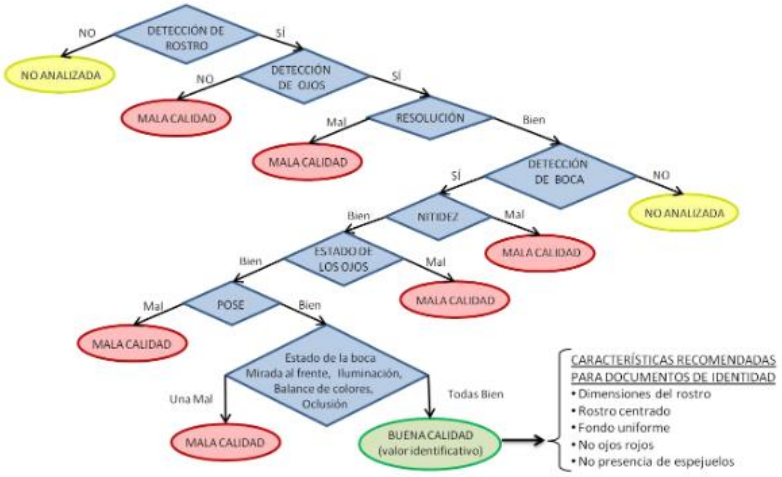

Figura 7 Evaluación de la calidad de una imagen de rostro.

Fuente: (Méndez-Vázquez et al. 2012)

Algunas de las recomendaciones extraídas de la norma ISO son las siguientes:

- 60 pixeles entre ojos como mínimo.
- Ausencia de sombras.
- Ojos abiertos, con iris visible.
- Imagen frontal, nítida y con colores naturales.

Además, recomienda:

- 70-80% de la imagen para el alto de la cabeza.

- los ojos deben estar entre el 50% y el 70% del alto de la imagen.

La web de Axis («Identificación y reconocimiento | Axis Communications» 2018), empresa especializada en la video-vigilancia, recomienda 80 pixeles para el rostro, y establece algunas limitaciones en cuanto a distancia focal y distancia del sujeto para una correcta identificación

La web de la Universidad de Bolonia («Biometric System Laboratory» 2018a) dispone de una web muy completa sobre biometría donde se incluyen una lista de requerimientos en una imagen para una correcta identificación facial.

En el trabajo de Ferrara, Maltoni y Franco (Ferrara, Franco y Maltoni 2008) sobre la norma ISO, los especialistas en reconocimiento facial y con huella dactilar recomiendan:

- 240 pixeles de ancho de imagen y 320 de alto. O, de forma más general, W * 0,75, siendo W el ancho de la imagen.
- Para las coordenadas de los ojos recomiendan 0,6 * W en altura y (0,375 * W) - 1 para un ojo y (0,625 * W) - 1 para el otro.
- 0,25 * W para distancia entre ojos (60 para W=240)

La web Neurotechnology («Neurotechnology» 2018), empresa especializada en biometría y visión por computador, habla de imágenes de 640x480 mínimo y, al menos, 50 pixeles entre ojos, aunque recomienda subir a 75 o más.

Aunque las diferentes recomendaciones que se han mencionado puedan sufrir ligeras variaciones entre ellas, las diferencias son relativamente pequeñas y permiten disponer de unas condiciones mínimas que las imágenes a tratar deben cumplir para conseguir una identificación facial aceptable.

Un ejemplo de imagen aceptable según la norma ISO se incluye en la Figura 8.

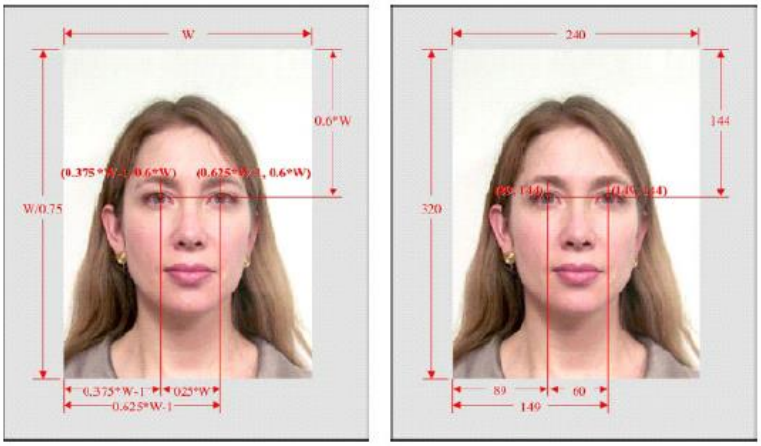

Figura 8 Estandar ISO19794-5.

Fuente: (Sang, Lei y Li 2009)

Hay varios trabajos que ofrecen mejoras en las imágenes para cumplir con algunos de los estándares, como el de Sang (Sang, Lei y Li 2009), que propone aplicar algoritmos para mejorar la calidad de las imágenes, como filtros de Gabor para la simetría de la imagen.

2.2.10 Problemática asociada al reconocimiento facial

El reconocimiento facial mediante imágenes se enfrenta a múltiples problemas relacionados con las variaciones que puede sufrir la apariencia del rostro. Se pueden diferenciar dos tipos de factores (Jafri y Arabnia 2009) que afectan a estas nuevas imágenes: los **factores intrínsecos**, relacionados con la naturaleza física del rostro, y los **factores extrínsecos**, relacionados con el observador, la iluminación, etc.

Los factores intrínsecos pueden ser el envejecimiento, las expresiones faciales (Hsieh, Lai y Chen 2010), los diferentes cortes de pelo, el maquillaje, etc. En el estudio de Givens (Givens et al. 2004) se comprueban estos efectos en tres algoritmos de reconocimiento diferentes.

Factores extrínsecos como la Iluminación, la pose, la resolución, el escalado, el ruido o la oclusión (Gonzalez-Sosa et al. 2016).

Existen trabajos con propuestas para mejorar la respuesta debida a los cambios de iluminación (Tan y Triggs 2010), otros proponen el uso de infrarrojos para solucionarlos (Li et al. 2007).

2.3 Reconocimiento por huellas dactilares

Aunque existen aportaciones anteriores, se podría considerar que fue Marcelo Malpighi, biólogo italiano, el primero que realizó un estudio sobre las huellas dactilares en 1686.

Ya en 1856 William Herschel (Herschel 1916) empezó a utilizar la huella para personas analfabetas firmaran contratos, al comprobar que las huellas no variaban con el tiempo. El Dr. Henry Faulds (Faulds 1880) publicó un artículo en la revista Nature sugiriendo el uso de la huella en la detección de criminales. Seguido de varios manuales sobre su uso (Faulds 1912)

Francis Galton también trabajó con huellas dactilares, llegando a publicar en 1892 un libro sobre el tema. («Finger Prints by Francis Galton» 1912)

La huella de cada individuo presenta un dibujo único, formado por las crestas que sobresalen de la piel, dejando más baja la parte conocida como valle. El grosor de estas líneas es de apenas unas décimas de milímetro.

La clasificación más habitual de las huellas dactilares se realiza siguiendo las indicaciones de Edward Henry (Henry 1900) aunque no siempre se dividen en los mismos grupos.

Figura 9 Huella arco.

Fuente: (Henry 1900)

Así, para las huellas dactilares se distinguen habitualmente los siguientes grupos:

- Arco. Un ejemplo se incluye en la Figura 9.
- Arco Pronunciado
- Bucle hacia la derecha
- Bucle hacia la izquierda
- Doble bucle
- Remolino

Para esta clasificación se utilizan dos singularidades que aparecen en las huellas, los **núcleos** y los **deltas**.

En ciertos puntos las líneas que forman las crestas se cortan de forma abrupta, se bifurcan, desvían, etc. Estas peculiaridades se conocen como **minucias**. En la Figura 10 se pueden observar algunas de ellas.

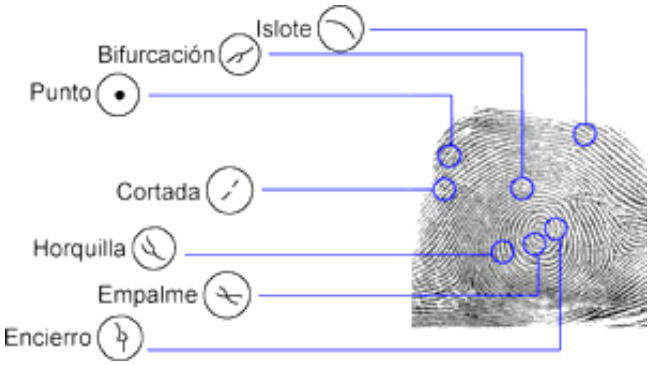

Figura 10 Minucias.

Fuente: («Huella Digital - Aplicaciones Tecnologicas» 2018)

El primer paso en el proceso de reconocimiento es realizar la captura de la huella, para lo que es necesario un sensor preparado. Los hay ópticos, térmicos y capacitivos, aunque existen nuevas alternativas como el uso de la cámara de un móvil (Derawi, Yang y Busch 2011)

Para el procesado de una huella digital se realiza, en primer lugar, una eliminación de ruido, para mejorar la imagen en todo lo posible.

A continuación, la imagen se convierte a un formato binario, lo que consiste en convertir la imagen original a una en blanco y negro. Se continúa haciendo un adelgazamiento de las crestas para poder encontrar más fácil los puntos característicos de la huella.

El último paso sería la extracción de estos puntos característicos o minucias. Una vez se dispone del dibujo con la posición exacta de estos puntos, se procede a comparar la imagen con las almacenadas en la base de datos.

Para reconocer patrones en huellas dactilares se emplean diferentes técnicas:

- Basadas en la comparación de patrones de minucias. Como la técnica de relajación (Ranade y Rosenfeld 1978).
- Basadas en las características de la estructura de crestas y valles.
- Basadas en la textura de la imagen

Se pueden encontrar diferentes algoritmos de comparación como el de Jiang (Jiang y Yau 2000), los de Medina-Pérez (Medina-Perez et al. 2011) (Medina-Pérez, Gutiérrez-Rodríguez y García-Borroto 2009), de Parziale (Parziale y Niel 2004) o el de Ferrara y otros (Cappelli, Ferrara y Maltoni 2010).

En los últimos años el uso de la huella dactilar como método de identificación se ha extendido, hasta tal punto que se suele incluir en la mayoría de teléfonos móviles que se venden hoy día. En la Figura 11 se puede ver uno de estos lectores.

Figura 11 Lector de huellas incluido en un Iphone 7.

Fuente: cultofmac.com

2.4 Reconocimiento del iris del ojo

En 1987 Flom y Safir (Flom y Safir 1987) proponen un sistema de reconocimiento utilizando el iris del ojo, enumerando las etapas necesarias y describiendo el proceso. El mencionado sistema requiere actuación humana en el proceso de detección. La

confirmación de que el iris podría utilizarse como característica para el reconocimiento biométrico llegó en 1991 en un trabajo de Johnson (Johnson 1991).

Posteriormente, en 1994, fue John Daugman quien mejoró el trabajo anterior presentando una patente de un sistema completamente funcional y automático. Otros trabajos relevantes de Daugman fueron presentados en 2007 (Daugman 2007) y 2009 (Daugman 2009).

Aunque existen otras alternativas, casi todos los métodos utilizados en reconocimiento mediante el iris se basan en los trabajos de Daugman y en los presentados en 1997 por Richard Wildes (Wildes 1997).

Las etapas en que se divide el proceso de reconocimiento mediante el iris del ojo suelen ser cuatro: localización, normalización, extracción de características y reconocimiento. Para cada una de ellas existen diferentes alternativas disponibles. Daugman proponía el uso de wavelets de Gabor para la extracción de características y utilizar la distancia de Hamming para el reconocimiento. Wildes propone la codificación mediante Laplaciano de Gaussiana que utiliza la imagen en varias escalas y diferentes filtros, para el reconocimiento propone la correlación normalizada.

Para la normalización, las técnicas más habituales para detectar la curvatura del iris utilizadas en reconocimiento son el uso de un operador integro-diferencial propuesto por Daugman, Figura 12, y, la más utilizada, la transformada circular de Hough, implementada habitualmente junto a un detector de bordes de Canny (Canny 1987).

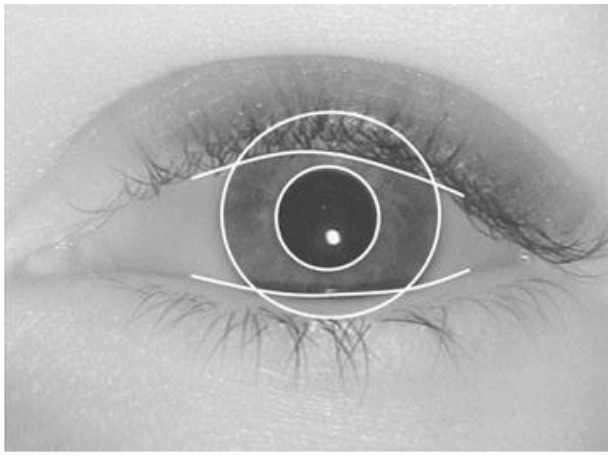

Figura 12 Aplicación operador integro-diferencial.

Fuente: (Bowyer y Burge 2016)

2.5 Reconocimiento por las venas de la mano

Existen diferentes alternativas en cuanto a localización para el uso de las venas del cuerpo. Todas se basan en el mismo principio, la luz ultrarroja es absorbida por la

hemoglobina de la sangre que, desoxigenada, regresa al corazón. Esta es, de hecho, una técnica utilizada en oximetría[2].

Existen estudios que demuestran que la disposición de las venas de la mano es única en cada individuo (Badawi 2006) (Choi 2001). Además de la unicidad, reúne el resto de condiciones necesarias para utilizarse como técnica biométrica

El primer paso en un sistema de reconocimiento por venas de la mano es adquirir una imagen donde estas sean apreciables. Se realiza utilizando una fuente de luz infrarroja sobre la mano y una cámara apta para captar esta luz. Se puede hacer por reflexión, con la fuente y la cámara, ambas, frente a la palma de la mano o por transmisión, cambiando la fuente a la parte dorsal. Ambas opciones se pueden ver en la Figura 13.

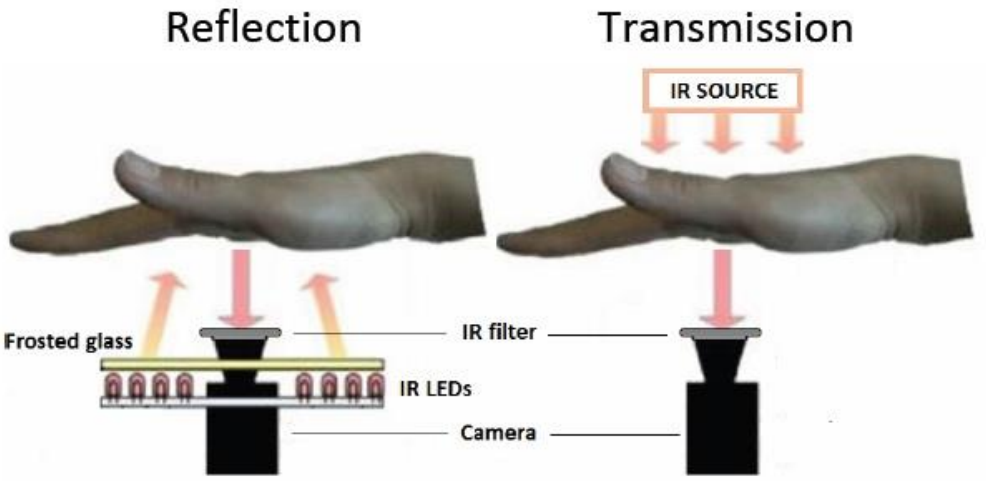

Figura 13 Reflexión y Transmisión.

Fuente: («Biometric identification on a picture of veins of a palm» 2018)

Existen cuatro procedimientos distintos para realizar la extracción de las características biométricas (Rahul, Cherian y Mohan 2015):

- Basados en línea
- Basados en apariencia
- Basados en codificación
- Basados en textura

En la Figura 14 se aprecian los procedimientos y algunos de los algoritmos disponibles en cada uno de ellos.

[2] Técnica médica que permite medir la saturación de oxígeno en la sangre de una persona.

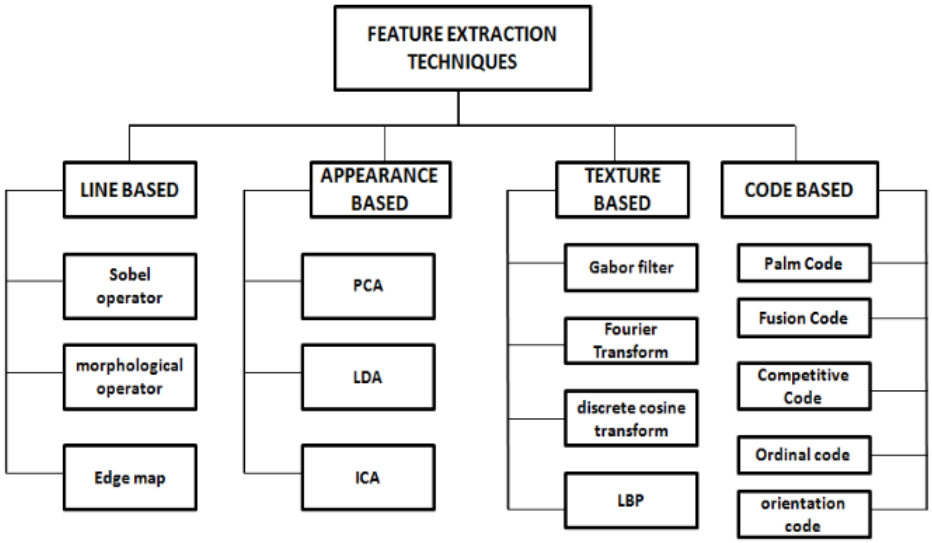

Figura 14 Algoritmos de comparación para venas de la mano.

Fuente: (Rahul, Cherian y Mohan 2015)

Una vez finalizado el estudio, se mostrarán, en el siguiente capítulo, varios sistemas y soluciones actuales que emplean algunos de los métodos presentados. En el capítulo 4, dedicado al diseño del proyecto, se presentarán los métodos empleados en el presente trabajo fin de grado, así como una justificación de su uso.

3 ESTADO DEL ARTE

En este capítulo se van a abordar los estados actuales de las tecnologías en diferentes técnicas biométricas. Solo se consideran las técnicas que han sido valoradas para la realización de este trabajo.

También se va a estudiar el presente de otra tecnología utilizada, la visión por computador.

3.1 Reconocimiento facial

Existen multitud de empresas como Ayonix («Hotel Security» 2015) o Axis («Reconocimiento facial | Axis Communications» 2018) que ofrecen sistemas de reconocimiento facial que proporcionan seguridad en edificios y, en algunos casos, incluyen servicios añadidos asociados a la identificación de usuarios.

Otras empresas han presentado sus propios algoritmos de reconocimiento facial como Facebook y sus Deepface (Taigman et al. 2014) y Piper («Facebook researchers boost facial recognition with PIPER poselets» 2015), o FaceNet (Zhang et al. 2015) de Google.

3.1.1 Neurotechnology

Empresa especializada en reconocimiento mediante parámetros biométricos ofrece Verilook SDK (Software Development Kit) («VeriLook face identification» 2017), un paquete que permite desarrollar software con reconocimiento facial. Los precios (de junio de 2017) son de 339 € para la versión estándar y 859 € para la versión completa.

3.1.2 Amazon AWS

Entre los servicios que la empresa Amazon («Amazon.es» 2018) ofrece, se incluye la posibilidad de utilizar su API de reconocimiento facial («What Is Amazon Rekognition? - Amazon Rekognition» 2018). El precio varía según el país. En Europa, el minuto de video en directo analizado, tiene un coste de 0,12 $, más 0,01 $ por cada rostro almacenado al mes.

3.1.3 Microsoft

Entre los múltiples servicios que ofrece Microsoft en la nube, lo que denominan Azure («Azure» 2018), se incluye la Face API («Face API» 2018), que permite, entre otras funciones, detectar, reconocer y verificar rostros.

3.1.4 IBM

La empresa IBM ofrece su propio servicio en la nube para reconocimiento facial, denominado Watson («IBM Watson Developer Cloud» 2018). Con precio de 0,10 $ por imagen en entrenamiento y 0,004 $ por imagen en reconocimiento («New lower price for Watson Visual Recognition» 2016).

3.1.5 BaseApp

La empresa BaseApp («Deepsight Image Recognition SDK» 2018) ofrece un SDK denominado DeepSight. Existe una versión gratuita de este SDK, con ciertas limitaciones, y una versión de pago por 29 $.

3.1.6 FacePhi

La empresa española FacePhi (FacePhi 2018), especializada en reconocimiento facial, ofrece un SDK completo para reconocimiento facial.

3.1.7 Código abierto

Existen librerías de código abierto, como OpenCV o Emgu, que incorporan algoritmos de reconocimiento facial totalmente funcionales ya implementados.

3.1.8 Ejemplos de implementación.

Recientemente, el gobierno chino ha implantado un sistema de vigilancia que incorpora reconocimiento facial para más de mil millones de personas. La empresa Yitu Technology ha sido la responsable del algoritmo de inteligencia artificial Dragonfly Eye («China's Minority Report-style» 2017), encargado de esta colosal tarea.

Por otro lado, el gobierno francés ha incluido sistemas de reconocimiento facial en sus principales aeropuertos («Aeropuertos parisinos instalan sistemas de reconocimiento facial» 2018).

3.2 Huellas dactilares

Es habitual que los fabricantes de lectores de huellas dactilares dispongan de algoritmos propios para el cotejado (*matching*) y faciliten su uso con un software proporcionado con el lector, generalmente en forma de SDK que permiten generar aplicaciones fácilmente. También existen empresas que desarrollan algoritmos propios y que integran en un SDK que permite su uso con diferentes lectores.

3.2.1 Griaule Biometrics

La empresa Griaule Biometrics («Griaule Biometrics |» 2018) ha ofrecido a lo largo de los últimos años diferentes alternativas que permiten su uso con los lectores de huellas más populares, como DigitalPersona. Ejemplos de estos SDK son Fingerprint SDK 2009 («Fingerprint SDK 2009:Setup and Usage - Griaule Wiki» 2018) y su versión anterior GrFinger.

3.2.2 Crossmatch

La empresa Crossmatch («Fingerprint Verification | Biometric Identity SDK» 2018), que adquirió DigitalPersona en 2014, proporciona un SDK para los lectores de huella

dactilar fabricados por la propia DigitalPersona. Existen versiones para varios lenguajes de programación. Las versiones distribuidas son OneToch o, las más recientes, U.Are.U SDK.

3.2.3 Neurotechnology

Verifinger SDK («VeriFinger SDK» 2010) es uno de los diferentes SDK que proporciona la empresa Neurotechnology. Capaz de trabajar con la mayoría de los lectores de huella. Los precios (de junio de 2017) son de 339 € para la versión estándar y 859€ para la versión completa.

3.2.4 Otras alternativas

La universidad de Bolonia desarrolló un SDK conocido como MCC («Biometric System Laboratory» 2018b). De la mano de especialistas como son Cappelli, Ferrara, Maltoni y Maio.

Daniel Peralta, de la universidad de Granada, ofrece el código utilizado en sus trabajos («Fast Fingerprint Identification for Large Databases | Soft Computing and Intelligent Information Systems» 2018) donde se implementan varios algoritmos de *matching*.

La librería libfprint («libfprint» 2016) permite utilizar algunos de los lectores de huellas más populares, incluidas las versiones 4000 y 4500 de DigitalPersona, en sistemas Linux.

En el libro Blueprints OpenCV (Howse et al. 2015) se ofrece el código para implementar un sencillo algoritmo en C++.

3.2.5 Lectores de huellas

Existen multitud de empresas que ofrecen lectores de huellas digitales. Una de las más populares es Crossmatch y sus lectores DigitalPersona, ya mencionados. Otras opciones son Fujitsu, Futronics, Lumidigm, Secugen, Suprema, ZKSoftware, e, incluso, Microsoft. En la Figura 15 se pueden apreciar (de izquierda a derecha) los lectores, Microsoft finger Reader, DigitalPersona U.Are.U 4500, la versión 4000 de U.Are.U y R308 para Arduino.

Figura 15 Lectores de huellas evaluados

Desde la aparición de Arduino, y gracias a su popularidad, han ido apareciendo diferentes lectores de huellas enfocados a su uso junto a la famosa placa. Estos lectores, limitados en cuanto a prestaciones en comparación con otros más profesionales, como los de DigitalPersona, son sencillos de utilizar y, sobre todo, muy económicos. Para su uso es suficiente con conectarlos con la placa Arduino y conocer el protocolo que sigue. Un ejemplo de estos lectores es el R308 («Wiring for use with Arduino Fingerprint Sensor» 2018) compatible con las librerías de Adafruit («Adafruit» 2018)

3.3 Iris

De forma similar a lo que ocurre con los lectores de huella, los escáneres de iris suelen venir acompañados de un software que permite el reconocimiento, aunque, también, se dispone de paquetes SDK que permiten tratar la imagen del iris capturado, procesarla y realizar un reconocimiento.

3.3.1 VeriEye SDK

De la empresa Neurotechnology, VeriEye («Neurotechnology» 2018) permite realizar el reconocimiento a partir de la imagen del iris obtenida mediante un escáner de retina, como el I Scan 2 de CrossMatch. Los precios (de junio de 2017) son de 339 € para la versión estándar y 859 € para la versión completa, iguales al resto de SDK para biometría que ofrece esta empresa.

3.3.2 iData SDK

Similar al anterior, de la empresa IRIS ID (Id 2018).

3.3.3 Trident

De la empresa CredenceID («Trident® – Credence ID» 2018), es un escáner que permite la captura del iris cumpliendo con los requisitos exigidos para una correcta identificación. Su precio es de unos 3.200 € («Credence Trident» 2018)

Imagen 3.1 Escáner Trident.

Fuente: («Trident® – Credence ID» 2018)

3.3.4 I-Scan2 de CrossMatch

Similar al anterior, de la empresa CrossMatch («Escáner de Ojo Cross Match» 2018).

Imagen 3.2 I-Scan 2.

Fuente: («Escáner de Ojo Cross Match» 2018)

3.4 Venas de la mano

Existen diferentes opciones comerciales que permiten la identificación por venas de la mano.

3.4.1 Fujitsu

La empresa Fujitsu ofrece *PalmSecure* («palmsecure.pdf» 2018), un sistema de reconocimiento que utiliza las venas de la palma de la mano. Su precio es de unos 400 €

Imagen 3.3 PalmSecure de Fujitsu.

Fuente: («Fujitsu PalmSecure» 2018)

3.4.2 Hitachi

Hitachi ofrece el sistema *Finger Vein*, capaz de utilizar las venas de un dedo para realizar la autenticación del usuario.

3.4.3 M2-Sys

Otra opción es M2-PalmVein de la empresa M2sys. El precio es de 481 $.

Imagen 3.4 M2-PalmVein Reider de la empresa M2-SYS.

Fuente: *(«Palm Vein Scanner to Secure Biometric Recognition | M2SYS» 2018)*

3.4.4 Otras alternativas

También existen implementaciones realizadas por particulares («Biometric identification on a picture of veins of a palm» 2018) con cierto éxito. Suárez Pascual (Suárez Pascual 2011) en su tesis demuestra que es factible diseñar un sistema de reconocimiento mediante venas de la mano, obteniendo buenos resultados sin recurrir a soluciones comerciales.

3.5 Visión por computador

Según los profesores de la Universidad Politécnica de Valencia, Sáchez-Salmeron y Ricolfe-Viala, en el libro Conceptos y Métodos en Visión por Computador (Gutierrez Alegre, Pajares y Escalera Hueso 2016), la visión por computador podría definirse como sigue:

> "La visión artificial consiste básicamente en la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, a partir de una o varias imágenes bidimensionales de ese mundo. En esta área de conocimiento se aúnan conceptos de la física del color, óptica, electrónica, geometría, algorítmica, sistemas de computación, etc."

Así, para la identificación biométrica estática, es necesario el uso de sistemas que permitan capturar imágenes, tratarlas y procesarlas para, posteriormente, utilizar los algoritmos necesarios que permitan determinar, en función de patrones preestablecidos, si la imagen se corresponde con el individuo a identificar.

Dependiendo de la parte del cuerpo que se desee capturar será necesario adaptar la cámara. Por ejemplo, para capturar las venas de la mano o del brazo es necesario usar una cámara de infrarrojos, capaz de capturar una imagen donde las venas puedan distinguirse. Mientras que para capturar el iris del ojo con calidad es necesario una cámara con suficiente resolución.

Se requiere, por tanto, disponer de un sistema de visión por computador que permita capturar y tratar las imágenes capturadas para su evaluación biométrica. Se puede optar por utilizar un único paquete que englobe ambas funcionalidades o por utilizar dos paquetes diferentes.

Existen multitud de herramientas que incorporan el procesado de imágenes, algunas comerciales como *The Matrox Image Library* (MIL) («Matrox Imaging Library (MIL)» 2018), *Khoros*, *eVision* («eVision - Visual Search Technology» 2018), etc. También existen paquetes no comerciales como ImageLib («ImageLib» 2018), TargetJr («The Oxford TargetJr Page» 2018) u OpenCV que proporcionan entornos completos para el

procesado de imágenes. Dentro de este segundo grupo destacan OpenCV y Gandalf («Gandalf Home Page» 2012), ambas herramientas incorporan el tratamiento de imágenes y la visión por computador

3.5.1 OpenCV

La librería open source OpenCV («OpenCV library» 2018a) está enfocada a la visión por computador e incorpora algoritmos para el tratamiento de imágenes. Permite su uso en diferentes lenguajes de programación como C/C++, Python, Java, etc., e incluso, existe una versión para dispositivos Android. Su uso está muy extendido, hay disponibles foros, web y abundantes libros donde encontrar información (Calvo y Joshi 2018) (García et al. 2015) (Howse 2015) (Kaehler y Bradski 2016). En cuanto a la información disponible en línea, destacan las webs de Mallick («Learn OpenCV (C++ / Python)» 2018), Rosebrock («PyImageSearch - Be awesome at OpenCV, Python, deep learning, and computer vision» 2018) o acodigo («acodigo.blogspot.com» 2017) entre otras. Sobresaliente es el trabajo de Steven Puttemans (Puttemans 2018), que participa en el desarrollo de OpenCV y es muy activo en foros resolviendo dudas.

OpenCV, desarrollado por Intel, anunciado en la *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* bajo la licencia BSD, está compuesto por más de 300 funciones, escritas en C. Dentro de sus muchas funciones se encuentran el análisis y seguimiento de objetos, el procesado de imágenes o el reconocimiento de patrones.

3.5.2 Emgu

Emgu («Emgu CV: OpenCV in .NET (C#, VB, C++ and more)» 2018) Realmente es un envoltorio para poder trabajar con la biblioteca OpenCV en la plataforma .Net de Microsoft. Es prácticamente igual que OpenCV, con casi todas sus funcionalidades, sin embargo, su uso no está tan extendido, no existe tanta documentación ni existe un banco de usuarios tan grande. No obstante, en caso de ser necesario en uso de lenguajes .Net es una alternativa a OpenCV.

3.5.3 OpenFace

OpenFace («OpenFace» 2017) (Amos, Ludwiczuk y Satyanarayanan 2016) se trata de otra librería open source, similar a OpenCV, aunque menos extendida y, por tanto, con menos documentación.

3.5.4 Aforge.NET

Aforge («AForge.NET» 2017) es un framework de visión por computador para trabajar con C#.

3.5.5 Biblioteca Dlib

Desarrollado en C++, dlib («dlib C++ Library» 2018) es una librería que permite trabajar con imágenes entre otras muchas funciones, también incorpora capacidad para detectar los puntos de referencia de una cara, o landmarks, como se puede apreciar en la Figura 16. Su uso junto con OpenCV es habitual.

Figura 16 Ejemplo de Landmarks obtenidos con Dlib

3.5.6 Matlab

La herramienta de software matemático Matlab ofrece, entre sus múltiples opciones, la posibilidad de trabajar con imágenes, y, gracias a sus capacidades y potencial para el cálculo matemático es posible realizar reconocimiento facial.

4 DISEÑO DEL PROYECTO

En este capítulo serán enumeradas las tecnologías seleccionadas para la elaboración del sistema, se presentará una especificación general del mismo, en la que se incluirán descripciones de los diferentes dispositivos utilizados junto a los servicios que requieren y proporcionan. Por último, se especifican los diferentes protocolos de comunicaciones establecidos para interconectar los diferentes dispositivos.

4.1 Tecnologías utilizadas

A continuación, se presentan las tecnologías elegidas para la realización de este proyecto y las causas que justifican esta elección.

4.1.1 *Visión por computador*

Dado que varias de las técnicas biométricas valoradas exigen el uso de visión por computador, el primer paso ha sido seleccionar la técnica más apropiada en este campo. Ha sido determinante para su elección, el hecho de que **OpenCV** («OpenCV library» 2018a) implemente algunos de los algoritmos de reconocimiento facial más extendidos, además de ser una de las alternativas más documentada y con una bibliografía más amplia.

Esta elección condiciona un poco las siguientes, como el lenguaje de programación y el entorno, pues OpenCV dispone de versiones para múltiples lenguajes y se puede integrar con casi cualquier entorno, aunque habría que descartar alguno como C#.

4.1.2 Técnica biométrica

Encontrar la técnica biométrica apropiada para controlar el acceso a los espacios de un edificio y cumplir con los objetivos propuestos para el presente proyecto requería el estudio y la comparación, no solamente de las diferentes opciones en cuanto a la técnica biométrica, sino que también era necesario evaluar, dentro de cada técnica, las posibles alternativas disponibles.

Para el reconocimiento por huella dactilar ha sido posible realizar pruebas con diferentes lectores de huella: las versiones 4000 y 4500 de U.Are.U de DigitalPersona, el lector de huellas de Microsoft Fingerprint Reader y un modelo R308 junto con una placa Arduino. Para poder evaluar convenientemente los lectores antes mencionados, se han utilizado versiones de evaluación de Verifinger SDK («VeriFinger SDK» 2010) y Fingerprint SDK («Fingerprint SDK 2009:Setup and Usage - Griaule Wiki» 2018). Existen manuales detallados con las funciones de cada uno y para cada lenguaje de programación con el que pueden trabajar.

Para el lector de huellas R308 ha sido necesario utilizar el IDE de Arduino y las librerías requeridas por el lector, así como el SFG demo, que se puede descargar de la web de Adafruit («Adafruit Optical Fingerprint Sensor» 2018) y que sirve para realizar unas

primeras pruebas con el lector. En la Figura 17 se muestra un resultado obtenido con este software.

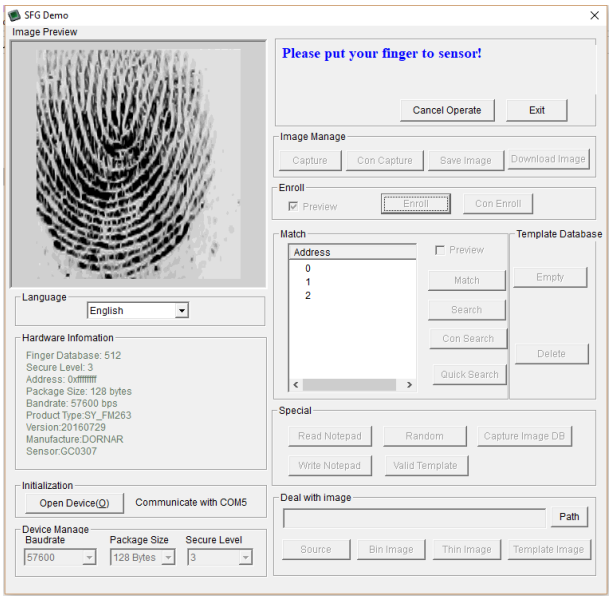

Figura 17 Huella dactilar del autor extraída con R308 usando SFG demo

Las pruebas realizadas con los lectores de huella han evidenciado que integrar un SDK con una aplicación no es una tarea sencilla. Se ha descartado el uso de un SDK por el elevado precio de los lectores de huella, en comparación con las cámaras probadas para el reconocimiento facial, además de la necesidad de pagar licencias para el uso del software. Como única alternativa ha quedado el uso de un lector de huellas R308 junto con una placa Arduino. Este lector tiene integrado todo el software necesario y solo necesita la placa Arduino y la librería correspondiente para funcionar. Los resultados iniciales han sido suficientemente satisfactorios, la implementación es viable y el coste es mucho más reducido que un lector más profesional.

Los dispositivos hardware diseñados para escanear el iris del ojo tienen un coste considerable en comparación con el hardware necesario para otras técnicas biométricas vistas, por lo que se han descartado y se ha pasado a evaluar la posibilidad de utilizar una cámara similar a las utilizadas para reconocimiento facial.

OpenCV incorpora algunas de las funciones que se requieren en el tratamiento de imágenes necesario para el reconocimiento biométrico mediante el iris ocular, como *Canny* o la transformada de Hough para los círculos (*HoughCircles*); pero los resultados no han sido satisfactorios, por lo que se ha descartado esta técnica biométrica. Se han realizado pruebas de captura con una Raspberry Pi 3B y la versión 2.1 de su cámara. La resolución de la imagen no parece adecuada, además es necesario, de forma considerable, la colaboración activa del usuario, al ser imprescindible encontrarse cerca de la cámara y en una posición muy determinada para capturar bien el iris del ojo.

Al igual que ocurre con los escáneres de iris, los lectores de venas de la mano tienen un precio elevado, por lo que igualmente se han descartado. No obstante, se ha intentado diseñar un lector económico que pudiera obtener la imagen con calidad suficiente para su posterior tratamiento. Pero los resultados obtenidos no han sido buenos. Por un lado, es necesario disponer de luz infrarroja, para ello se han recurrido a diodos leds, y, en concreto, se han probado los dispositivos que se pueden ver en la Figura 18 y en la Figura 20.

Figura 18 Leds de infrarrojos

Figura 19 Led driver

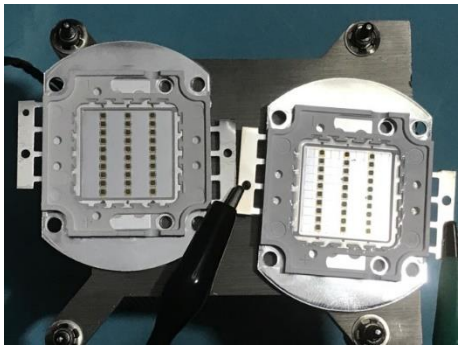

Figura 20 Leds de 730mm y 850nm

En la Figura 18 se muestra una placa con 48 leds, incorpora un sensor de luz que hace la función de interruptor encendiendo los leds cuando no hay iluminación. Esta opción no es lo suficientemente potente como para permitir observar las venas de la mano. También se ha probado con los leds de la Figura 20, ambos de 30W, uno de 730nm y otro de 850nm. Azueto (Azueto-Ríos et al. 2017) obtiene mejores resultados con una longitud de onda de 830nm, al compararla con 720 y 960nm. Similares conclusiones obtiene Van Tien (Van Tien et al. 2015) obteniendo mejores resultados con 850nm frente a las alternativas evaluadas de 750nm y 940nm.

Los resultados, aunque mejores, han sido insuficientes como para continuar. En ningún caso se han obtenido imágenes con la calidad suficiente como para realizar una identificación con garantías. Además, estas placas presentan múltiples problemas: se calientan rápidamente y en exceso, por lo que es necesario utilizar disipadores de calor e, incluso, un ventilador. También requieren una tensión muy estable por lo que es necesario adquirir un led driver, como el de la Figura 19, que aún encarece más el sistema. En ambos casos la cámara utilizada ha sido la versión 2.1 Noir, sin filtro de infrarrojos, diseñada para trabajar con Raspberry pi.

Por lo tanto, dados los malos resultados obtenidos en las primeras pruebas, junto al elevado coste del conjunto de elementos necesarios para el reconocimiento mediante venas de la mano, hacen que esta opción sea descartada.

Después de evaluar todas las alternativas disponibles, se ha optado por utilizar el **reconocimiento facial**. Tras realizar unas primeras pruebas preliminares, con una implementación haciendo uso de OpenCV para reconocimiento facial, los resultados obtenidos han sido bastante prometedores. Dentro de los algoritmos de identificación facial se han podido evaluar los algoritmos Eigenfaces, Fisherfaces y LBPH. Todos ellos implementados en OpenCV. Evitando la necesidad de recurrir a nuevas librerías.

Además, el reconocimiento facial no requiere el contacto directo con el usuario, algo que, en determinados casos, puede ser una gran ventaja, y es mejor considerado por el usuario que el resto de técnicas evaluadas (INTECO 2012). Por otro lado, aunque sí se requiere la colaboración activa, esta no es tan invasiva como en otras técnicas, como el reconocimiento mediante venas o el iris ocular.

El reconocimiento mediante huella, usando un lector R308, supone una alternativa interesante al reconocimiento facial que proporciona OpenCV, y será utilizado en este proyecto en combinación con él, en determinados casos, para proporcionar una doble capa de seguridad biométrica. Esta elección tiene un inconveniente, el lector R308 es poco flexible y no es posible realizar una gestión centralizada del mismo, ni utilizar una misma huella para varios dispositivos, por lo que sería necesario registrar a los usuarios en cada lector que puedan utilizar. Esta limitación descarta su uso, en favor del reconocimiento facial, como técnica principal para el reconocimiento, dado que uno de los objetivos principales que se persigue es conseguir un sistema centralizado, sin embargo se considera un inconveniente asumible, como sistema alternativo, dado que su rol será complementario.

4.1.3 Comunicaciones seguras con el Servidor principal

Para mantener la integridad del sistema es necesario que las comunicaciones permanezcan seguras. Para ello se ha optado por utilizar **OpenSSL** («OpenSSL» 2018),

herramienta que permite implementar ciertos protocolos de seguridad de las comunicaciones como SSL (Secure Sockets Layer) o TLS (Transport Layer Security).

4.1.4 Comunicaciones seguras con Arduino

Para poder dotar de seguridad a las comunicaciones en las que intervienen placas Arduino o compatibles es necesario poder cifrarlas, las alternativas de las que se disponen, en cuanto a algoritmos de cifrado son múltiples, aunque la limitación computacional de la placa limita el número de estos algoritmos, algunos de los cuales serían:

- DES. El algoritmo de cifrado DES (Data Encryption Standard) fue publicado en 1975, basado en el algoritmo Lucifer desarrollado por IBM. Utiliza una clave de 56 bits y bloques de cifrado de 64. Hoy día es considerado inseguro y no se recomienda su uso.
- 3DES. Desarrollado en 1998 por la empresa IBM para solucionar las debilidades de DES. En este caso se hace un triple cifrado DES, donde la clave es de 128 bits, compuesto por 2 claves diferentes de 56 y 16 bits de paridad, o con una clave de 192 bits, tres claves diferentes y 24 de paridad. Los bloques se mantienen en 64 bits.
- AES (Advanced Encrycptation Standard) nace como alternativa a DES, presentado en 1998 (Quisquater y Cardis 1998 (Louvain-La-Neuve 2000), como Rijndael, con más detalle en el trabajo de los autores de 1999 (Daemen y Rijmen 1999).
- TEA. Dentro de los algoritmos de cifrado, en los últimos años han recibido cierta popularidad los algoritmos ligeros, principalmente por la explosión del IoT (Internet of Things). Entre estos algoritmos de cifrado ligeros destaca el algoritmo de encriptación ligero o TEA (*Tiny Encryption Algorithm*) (Wheeler y Needham 1994). TEA es un algoritmo de cifrado fácilmente implementado e ideal para proyectos donde hay problemas con la capacidad de cómputo.
- XTEA y XXTEA. El Algoritmo desarrollado en 1994 demostró tener muchas debilidades, de ahí que fueran necesarias diferentes mejoras. En 1996 (Needham y Wheeler 1996) se desarrolló una versión mejorada, que corregía algunas deficiencias, conocida como XTEA. Y una nueva versión se publicó en 1998 (Wheeler y Needham 1998), conocida como XXTEA. Aunque un estudio (Yarrkov 2010) ya demostró que también presenta debilidades.

Como ya se ha comentado, las limitaciones en cuanto a cómputo dificultan el cifrado en Arduino por lo que es necesario tener en consideración la velocidad de respuesta de la placa. También se debe tener en cuenta el considerable tamaño que podrían alcanzar las

implementaciones de estos algoritmos en relación con la capacidad de almacenamiento de la que disponen estas placas.

Existen varias implementaciones de estos algoritmos válidas para Arduino: de DES y del 3DES, desarrollada, específicamente para Arduino, por Tim Riemann en 2013 («DES and 3DES Encryption Library for Arduino and Raspberry Pi: DES library for Arduino and Raspberry pi.» 2018), partiendo del trabajo de Daniel Otte («AVR-Crypto-Lib/en – LaborWiki» 2018); existe, al menos, una del protocolo AES desarrollada por Davy Landman (Landman 2018); Oscar Delgado Mohatar (Delgado Mohatar 2011) en su Tesis Doctoral implementa el algoritmo XXTEA, con el código disponible para su uso libre en la web (Serra 2015); desarrollada por Rhys Weatherley («Arduino Cryptography Library: Arduino Cryptography Library» 2018) existe una librería que implementa varios de estos algoritmos de cifrado.

En la web Heli («Mini algoritmo de cifrado XXTEA para arduino | Heli» 2018) existe un interesante trabajo donde se estudia la implementación de varios algoritmos y se crea una versión diferente de **XXTEA**. Los resultados obtenidos por el autor son buenos, en cuanto a eficiencia, en comparación con otros algoritmos, por lo que se ha optado por utilizar este protocolo y esta librería, ligeramente modificada, en Arduino, con autorización del autor.

Es necesario utilizar una implementación del protocolo en los clientes que requieran comunicarse con Arduino, en este caso se ha utilizado una específica para Python («xxtea» 2018).

En el servidor se ha utilizado la misma librería que en Arduino desarrollada en lenguaje C++, con ligeras modificaciones.

4.2 Diseño general del sistema

Se trata de un sistema que requiere de una serie de dispositivos, con funcionalidades bien diferenciadas y con distintos módulos o servicios implementados en cada uno de ellos. El elemento principal del sistema será el **servidor principal**, responsable de la identificación facial. Se dispone, también, de dos tipos de clientes. Clientes **capturadores**, provistos de una cámara y con la capacidad de detectar caras en las imágenes con ella capturadas. Y clientes **actuadores**, que procederán tras recibir la oportuna instrucción del servidor. Además, de estos elementos principales el sistema dispone de otros elementos auxiliares, necesarios para el correcto funcionamiento del sistema en su conjunto.

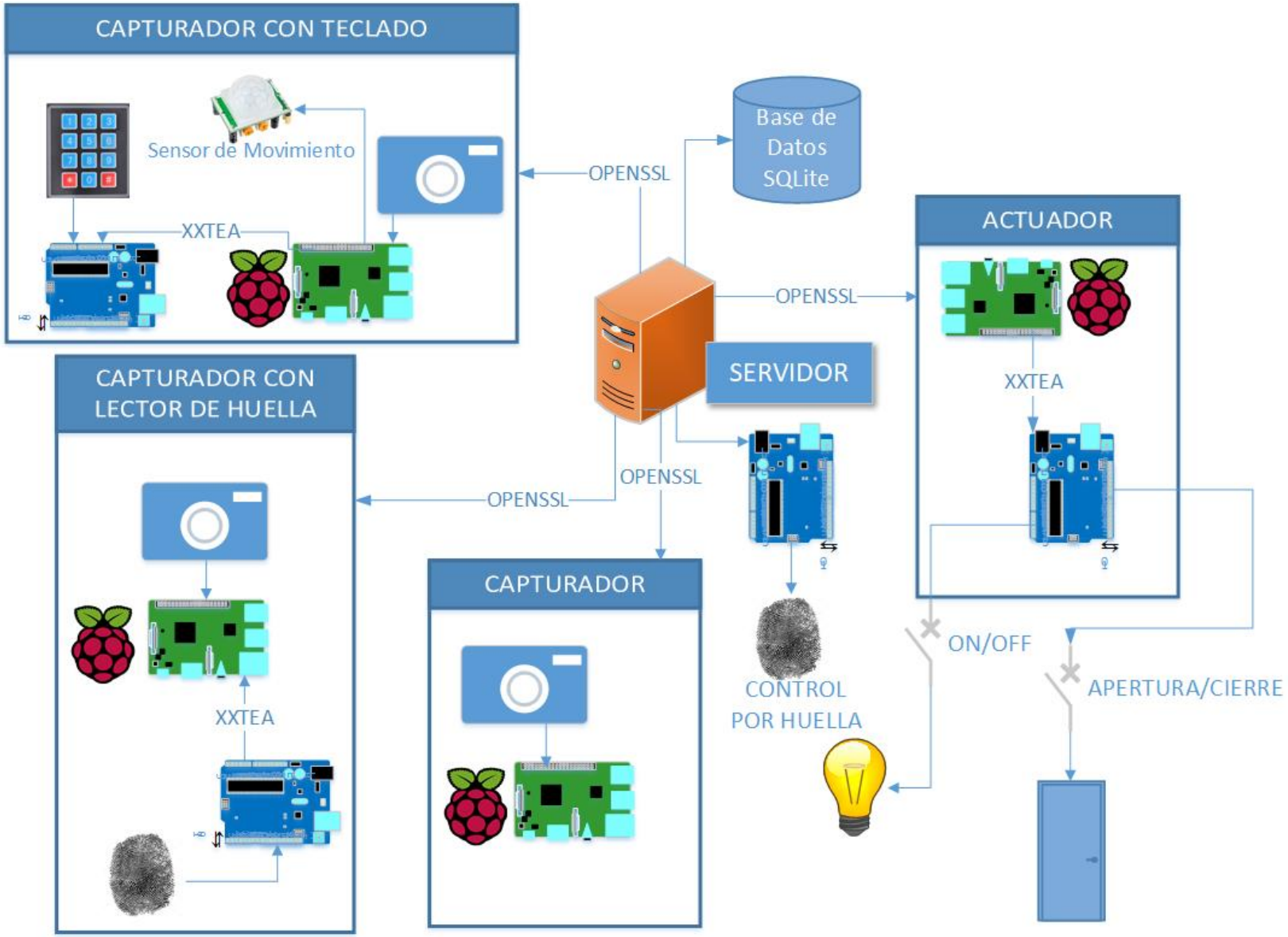

Figura 21 Diagrama general del sistema

4.3 Servidor

Se trata del elemento principal del sistema. Responsable de la identificación de los usuarios y de desencadenar los procesos que correspondan tras una identificación positiva.

Cuando la aplicación del servidor es iniciada el módulo de reconocimiento facial es cargado, este procedimiento requiere realizar un entrenamiento con las imágenes almacenadas e identificadas previamente, y deja un socket TCP (puerto **4433**) a la espera de conexiones por parte de los clientes capturadores.

Dentro de las funciones del servidor se incluyen:

- La aplicación permite gestionar toda la información almacenada en la base de datos a través de formularios (se puede ver con más detalle en el manual de usuario, incluido en el Anexo I).
- Dispone de la posibilidad de realizar reconocimiento facial de una imagen almacenada en un archivo, con formatos pgm o jpg, también en videos almacenados, formatos avi o mov.
- También es posible utilizar una cámara local o la de un cliente capturador para, de forma manual, después de detectar un rostro, capturarlo y realizar un reconocimiento.
- Recibir imágenes de rostros capturados en los dispositivos capturadores y, en función de la configuración, ejecutar las respuestas correspondientes.
- La posibilidad de testear los diferentes canales de los que dispone cada actuador.
- Capacidad para añadir nuevos usuarios con las imágenes capturadas con una cámara local o remota. También existe la posibilidad de añadir nuevas imágenes para un usuario ya existente.
- Una vez se detecta un rostro existe la posibilidad de realizar un seguimiento del mismo.

Si el rostro recibido de un capturador es reconocido positivamente, el primer paso es comprobar si el usuario reconocido tiene autorización para acceder al espacio o utilizar el servicio al que da acceso este capturador. Cada capturador está relacionado única y exclusivamente con un espacio[3]. Si el usuario tiene permiso para acceder, el siguiente paso es comprobar qué acciones se desencadenan con este acceso. La acción básica es permitir el acceso, si es una puerta, o permitir usar el servicio. Pero se permite añadir otras acciones que se desencadenen con cada acceso. Las posibilidades son infinitas, desde encender las luces de un despacho hasta iniciar la calefacción del edificio.

[3] Se hace referencia, a partir de ahora, a “espacio” para indicar indistintamente un espacio físico o un servicio proporcionado que requiera permiso para su uso.

Para llevar a cabo todas estas funciones, los principales servicios que integra el servidor son los siguientes:

- Comunicaciones.
- Seguridad de la información.
- Seguridad en el acceso como administrador.
- Detección facial.
- Reconocimiento facial.
- Gestión de la información.
- Interfaz de usuario.

4.3.1 Comunicaciones

El servidor está capacitado para establecer comunicaciones por dos medios diferenciados. Mediante una **red IP**, a la que deben conectarse los clientes, que deben conocer la dirección del servidor. Cuando se pone en funcionamiento el servidor, se lanza un socket que queda a la espera de recibir conexiones en el puerto **4433**. En el momento en que uno de los clientes se conecta con el servidor, es necesario lanzar un nuevo hilo que permita al servidor atender al cliente y, simultáneamente, permanecer en escucha para atender nuevas peticiones por el mismo medio. El nuevo hilo será el responsable de mantener la conexión con el cliente, asignar un nuevo puerto para las comunicaciones e iniciar los procedimientos correspondientes según los datos recibidos.

Por el mismo medio se dispone de la posibilidad de conectar con el segundo tipo de clientes, los actuadores, para iniciar los procesos programados tras una identificación positiva. Los accesos a los actuadores se realizan a través del puerto **4455** de estos.

Los capturadores ofrecen la posibilidad, al servidor, de conectarse con ellos para recibir en directo las imágenes que la cámara remota está capturando. Permitiendo identificar a los usuarios que se encuentren dentro del rango de visión de la cámara y realizar capturas de nuevos usuarios. El puerto utilizado es el **4444** del capturador.

Por otro lado, el servidor está capacitado para establecer comunicaciones a través del **puerto serie,** para dotar al sistema de una capa extra de seguridad; una placa Arduino es la responsable de las comunicaciones con un dispositivo lector de huellas o un teclado numérico que permitiría insertar un código de autenticación.

4.3.2 Seguridad de la información

El sistema se ha diseñado de modo que las comunicaciones sean lo suficientemente seguras para proteger la integridad del sistema y los datos. Para ello se ha dispuesto que servidor y clientes dispongan de capacidad para realizar un cifrado de las comunicaciones. De modo que cuando se inician las comunicaciones, mediante sockets,

se negocia el cifrado que se va a utilizar y, una vez se determina el método, las comunicaciones son cifradas con él.

Se ha habilitado la opción de que el gestor del sistema pueda determinar los protocolos de seguridad que el servidor aceptará, lo que incluye la posibilidad de no cifrar las comunicaciones. La principal motivación es dejar abierta la posibilidad de realizar testeos en el futuro sin modificar el sistema actual, dado que mantener una política de comunicaciones no cifradas no es recomendable, algo sobre lo que se profundizará más adelante, en Pruebas de seguridad.

El software elegido para el cifrado de las comunicaciones ha sido OpenSSL. Esta herramienta de software libre permite implementar protocolos de seguridad como SSL o TLS. Es recomendable mantener actualizada la versión de OpenSSL. Las pruebas de diseño han sido realizadas con la versión 1.0.1f, vulnerable a algunas amenazas recientes, como Heartbleed («Heartbleed, otro fallo extremadamente grave en una librería SSL» 2016), por lo que se recomienda mantener actualizado el servidor con la última versión estable. La reciente versión 1.1.1 soporta TLS 1.3, aunque no se ha utilizado en este trabajo.

4.3.3 Seguridad en el acceso como administrador

El sistema incorpora un mecanismo de seguridad para el acceso como administrador. Se requiere una autenticación positiva del administrador mediante una o varias de las siguientes opciones (según se configure el sistema):

- **Captura y reconocimiento facial**. Es necesario incorporar en el servidor la posibilidad de capturar imágenes. Para ello el dispositivo debe incorporar una cámara. Una vez el sistema dispone de una imagen capturada, esta se procesa para realizar una detección facial en la misma y el rostro capturado será tratado para su reconocimiento.
- **Lector de huellas**. Para aumentar la seguridad del sistema central se añade la posibilidad de requerir autenticación del administrador gracias a una de las tecnologías biométricas más utilizadas y conocidas, las huellas dactilares.
- **Teclado numérico**. Otra forma de aumentar la seguridad es utilizar un teclado numérico que permita introducir un código. Se ha utilizado un teclado externo que permita insertar el código desde el exterior, para acceder a la sala de administración.

Para que el acceso como administrador no solo habilite todas las funcionalidades del sistema, sino que además, pueda facilitar el acceso físico al espacio donde se realiza la administración del sistema, se han reservado un espacio, que se corresponderá con el

espacio donde se encuentra el servidor del sistema, y un canal de un actuador que se reserva para ser el que facilite la apertura del espacio una vez el administrador es reconocido. Se considerarán, por tanto, administradores aquellos usuarios que tengan permiso para acceder al espacio determinado.

4.3.4 Detección facial

Para poder incorporar nuevos usuarios al sistema es necesario disponer en el servidor de la capacidad de detección facial. Los nuevos rostros se podrán incorporar desde una cámara local o desde un capturador gracias a la posibilidad de realizar conexiones con estos para recibir imágenes en directo. En ambos casos es el servidor el que realiza la detección facial y el que procesa las imágenes para procesarlas y almacenarlas, en caso de estar ingresando un nuevo usuario.

Es posible, además, añadir nuevas imágenes de rostros capturadas para los usuarios ya registrados.

Se ofrece también la posibilidad de realizar un seguimiento (*tracking)* de un rostro detectado. Durante un determinado número de capturas se puede sustituir la detección por el seguimiento de un rostro, lo que permite, por un lado, mejorar el rendimiento, dado el menor coste computacional del seguimiento en comparación con la detección, y, por otro, se abre la posibilidad de incluir en el sistema otros servicios en el futuro.

4.3.5 Reconocimiento facial

Se trata del módulo que permite el reconocimiento de usuarios. Cuando una imagen es recibida, bien desde un cliente o bien desde una cámara conectada al servidor, el sistema será capaz de realizar los tratamientos necesarios en la misma para, finalmente, utilizar uno de los algoritmos de identificación facial vistos y conseguir la identidad del individuo, en caso de encontrarse registrado en el sistema. Cuando se utiliza un algoritmo de reconocimiento facial en OpenCV se obtienen dos valores, el identificador del usuario y la confianza (o confidence en OpenCV), que, en cierto modo, equivale a la distancia, entre las variables consideradas para el reconocimiento, de la imagen del rostro capturado y el rostro identificado, por tanto, a menor valor mayor garantía de éxito en el reconocimiento. Su valor depende del procedimiento utilizado. Cuando se utiliza LBPH los valores de confianza obtenidos suelen ser siempre inferiores a 100 e inferiores a 50 (Minichino y Howse 2015) para reconocimientos positivos. Dependiendo de las condiciones este valor puede ser muy inferior. Para establecer un límite, en cuanto a un valor mínimo para reconocimientos positivos, OpenCV permite introducir un valor de umbral, de modo que cuando la confianza es superior se considera reconocimiento negativo. Este valor es modificable en la configuración del servidor.

Considerando la variabilidad del valor de confianza, en función de las condiciones, es recomendable ajustarlo una vez el sistema ha sido implementado y probado, considerando estas condiciones, la seguridad requerida, etc.

Los rangos de valores para el valor de confianza son muy superiores para Eigenfaces y Fisherfaces, en torno a los 10.000 y los 2.500, respectivamente, serían buenos valores para empezar a calibrar.

4.3.6 Gestión de la información

Para el correcto funcionamiento del sistema es necesario disponer de una base de datos que contenga las imágenes de los usuarios registrados, además de toda la información adicional necesaria, relativa a los dispositivos, las autorizaciones, etc.

En el caso de las imágenes se ha optado por almacenarlas en un directorio con una estructura predefinida. El resto de información se ha almacenado en una base de datos SQLite, que es un sistema de gestión de bases datos relacionales que permite integrarse por completo en la aplicación al tener un reducido tamaño. Su uso está muy extendido en dispositivos Android, por su reducido tamaño en memoria. No ofrece todas las funcionalidades de un sistema más complejo, ni la seguridad, pero es suficiente para los propósitos de este proyecto. Sin embargo, dependiendo del uso y de los requerimientos en cuanto a seguridad, eficiencia y robustez podría ser recomendable optar por un sistema más complejo, como en el caso de bases de datos centrales de grandes corporaciones o empresas hoteleras, cuyo volumen de datos requeriría de otro tipo de gestión.

La principal información que es necesario almacenar en la base de datos se podría resumir con las siguientes tablas:

- Usuarios. Tabla que almacena la información de cada usuario registrado en el sistema
- Cámaras o capturadores. Con información relativa a la ubicación de las cámaras y su dirección de red.
- Actuadores. Similar a la anterior, pero para los actuadores.
- Permisos. Con la información relativa a los permisos de cada usuario
- Registro. Con los intentos de acceso al sistema.

4.3.7 Interfaz de usuario

El sistema se ha diseñado con una interfaz simple e intuitiva, que permite gestionarlo desde una ventana principal y una serie de ventanas auxiliares.

En la ventana principal se dispone de hasta cuatro espacios destinados a cargar las imágenes capturadas por los dispositivos remotos y que corresponden con intentos de identificación. En cada uno de ellos se mostrará información del usuario identificado, el espacio al que desea acceder y si se ha autorizado. Se dispone, además, de un espacio

destinado a mostrar video en directo capturado por una cámara remota o una local. En la Figura 22 se muestra la interfaz principal.

Desde este formulario de inicio es posible acceder al resto de formularios, destinados a la configuración de los dispositivos y usuarios. En el manual de usuario, incluido en el Anexo, se describen con más detalle las diferentes ventanas de las que dispone el servidor y sus respectivas funciones.

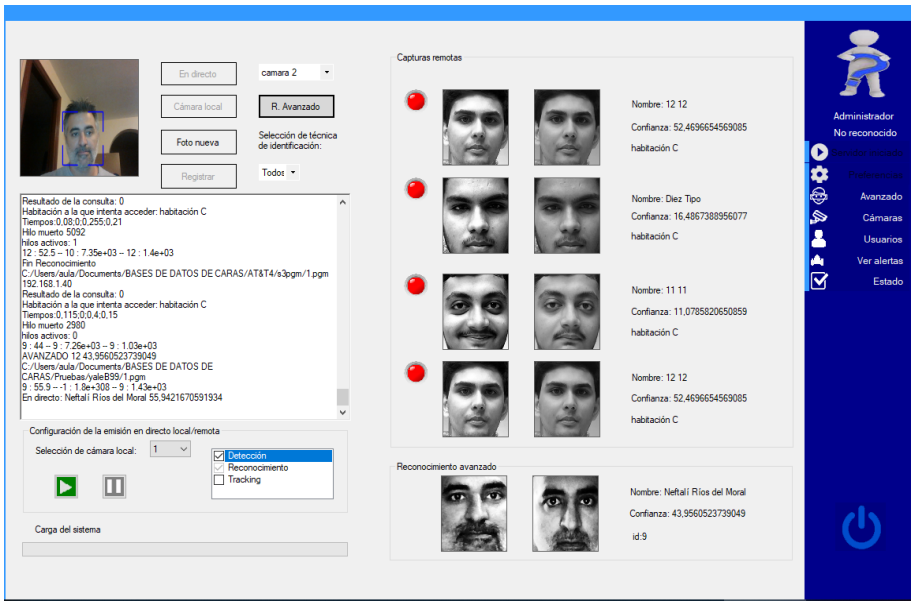

Figura 22 Interfaz de usuario

4.4 Capturador

En cada una de las estancias o para controlar el uso de diferentes servicios es necesario utilizar un elemento con capacidad para capturar una imagen del rostro del usuario que intenta acceder al espacio o usar un servicio. Además, es necesario que el cliente sea capaz de comunicarse con el servidor para enviar la imagen con el rostro del usuario. Para completar todas sus funciones requiere de las siguientes funcionalidades divididas en módulos:

- Comunicaciones
- Seguridad de la información
- Seguridad frente a suplantación (*Spoofing*)
- Captura
- Detección facial

Para poder enviar alertas al servidor se ofrece la posibilidad de pulsar un botón físico en los capturadores que lo incorporen y que permitiría, por ejemplo, alertar al administrador del sistema que un usuario está preparado para registrar sus huellas o para realizar un alta como usuario.

4.4.1 Comunicaciones

Para poder enviar al servidor el rostro del usuario es necesario poder comunicar con el servidor. Para ello se incorpora en cada cliente este módulo de comunicaciones.

Existen dos diferentes módulos de comunicaciones con el servidor, en función del uso que se vaya a dar al cliente en el momento. Además de la posibilidad de conectarse con el servidor para enviar un rostro capturado, se incluye en el cliente la funcionalidad para poder emitir video en directo. En este caso el cliente haría las funciones de servidor quedando a la espera de que el servidor se conecte con él para solicitarle la emisión. Para evitar conflictos con el uso de la cámara será necesario deshabilitar un módulo cuando el otro se encuentre en uso, o incorporar una segunda cámara en el cliente. El puerto utilizado por el cliente para esta emisión en directo es el 4444.

Las conexiones necesarias para las comunicaciones con el lector de huellas y con el teclado numérico se realizarán a través de una placa Arduino, conectado a través de un puerto serie.

Es necesario, además, disponer de capacidad para comunicarse con el resto de dispositivos que se conecten al capturador.

4.4.2 Seguridad de la información

Todas las comunicaciones que se realicen con el servidor serán cifradas. Asimismo, será necesario cifrar todas las comunicaciones que se realicen con dispositivos adicionales que se conecten al capturador a través de una placa Arduino.

4.4.3 Seguridad frente a suplantación (Spoofing)

Es recomendable incluir seguridad cuando se captura la imagen de un usuario para evitar que se realice algún tipo de suplantación, por ejemplo mediante una fotografía, que es el método más sencillo para quebrantar la seguridad de un sistema de identificación facial, aunque existen otras como el uso de una reproducción de video o una máscara (Galbally, Marcel y Fierrez 2014) (Reeba y Shanmugalakshmi 2015).

Si se utiliza una fotografía del usuario, al que se pretende suplantar, para evitarlo se disponen de algunas opciones como la detección de parpadeo, o, más general, lo que se conoce como detección de vida. Existen sistemas que requieren la colaboración del usuario, obligando a este a realizar un determinado movimiento o un parpadeo voluntario en determinado momento. Pan y otros proponen controlar el parpadeo del ojo (Pan, Wu y Sun 2008) (Pan et al. 2007). Más avanzado sería el uso de cámaras 3D o de profundidad, como Kinect (Erdogmus y Marcel 2013).

En este trabajo se han añadido diferentes opciones para dotar al sistema de cierta seguridad frente a ataques de suplantación: un segundo sistema de control biométrico, como la huella dactilar, el uso de un código personal, y, por último, se incluye un sistema

básico de reconocimiento de vida basado en la detección del parpadeo («Detección de sueño y parpadeo» 2017) (Se ha utilizado como base el código que se encuentra en la referencia, con autorización expresa del autor), basado en el trabajo de Soukupová (Soukupová y Cech 2016).

En la Figura 23 se aprecian los diferentes puntos guía (*landmarks*) utilizados y su numeración; para comprobar si se ha parpadeado será necesario evaluar el estado de los etiquetados entre el 37 y el 42 y entre el 43 y el 48.

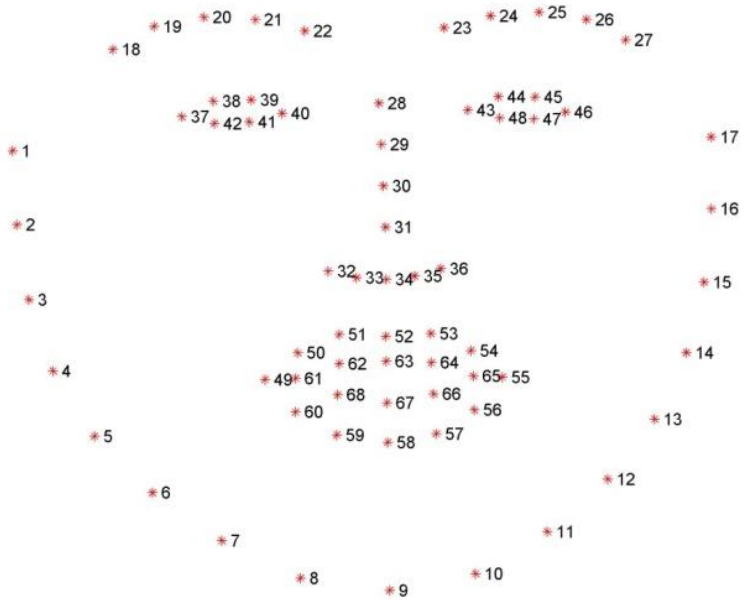

Figura 23 Puntos landmarks.

Fuente: («300 Faces In-the-Wild Challenge» 2013)

En la Figura 24 se aprecia la diferente posición entre los *landmarks* asociados al ojo cuando este se encuentra abierto y cerrado.

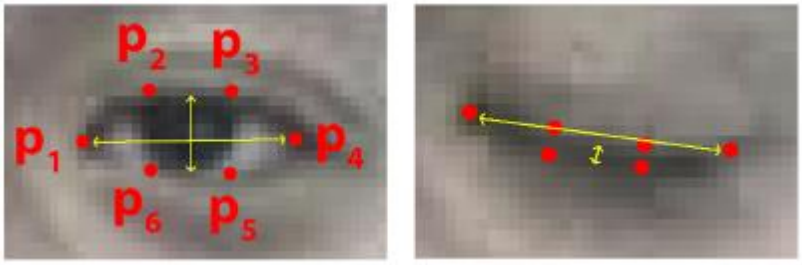

Figura 24 Landmarks con ojo abierto y cerrado.

Fuente: (Soukupová y Cech 2016)

El proyecto europeo Tabula Rasa («Trusted Biometrics under Spoofing Attacks» 2015), ya finalizado, tenía como objetivo principal combatir los ataques de suplantación en sistemas de reconocimiento biométricos.

4.4.4 Captura

Es necesario incorporar en el cliente la posibilidad de capturar imágenes. Para ello el dispositivo debe incorporar una cámara, además del software necesario para gestionarla, realizar capturas y procesarlas.

4.4.5 Detección facial

Una vez el sistema dispone de una imagen capturada se procesa esta imagen para realizar una detección facial en la misma. El rostro capturado será enviado al servidor principal.

4.5 Actuador

En este caso, el cliente, en lugar de iniciar una comunicación, se encuentra a la espera de recibirla, y, cuando suceda, serán instrucciones acerca de qué actuación se debe llevar a cabo. El servidor dispone de una base de datos que le indica que actuaciones deben iniciarse según qué usuario ha sido identificado y dónde. Para su correcto funcionamiento requiere los siguientes servicios:

- Comunicaciones.
- Seguridad de la información.

Las actuaciones serán responsables de enviar una señal a un dispositivo electrónico para cambiar su estado: una cerradura que se abre, una luz que se enciende, etc. Pero el sistema diseñado no controlará la vuelta al estado anterior de estos dispositivos. Ese control se debe gestionar desde el exterior. Aunque el sistema sí contempla la posibilidad de enviar señales contrarias, de encendido y de apagado, no se proporciona ningún mecanismo automático que las envíe, ni se controla el estado actual de estos dispositivos, dado que esto queda fuera de los objetivos del presente trabajo.

4.5.1 Comunicaciones

Este tipo de cliente solo tiene un modo de funcionamiento en relación al servidor, queda a la espera de que el servidor se comunique con él para darle instrucciones sobre las actuaciones a llevar a cabo. Estas comunicaciones se realizan a través de una red IP, lo hace mediante un socket que utiliza el puerto **4455**. Además, para poder ejecutar las actuaciones encomendadas es necesario que el dispositivo se comunique con ciertos componentes electrónicos, y para proteger la Raspberry se hará uso de placas Arduino, como ya se ha comentado. Las comunicaciones con estas placas se realizarán a través de un puerto serie.

4.5.2 Seguridad de la información

De nuevo, las comunicaciones con el servidor y con el resto de dispositivos deberán ser cifradas, por lo que existe un módulo responsable de esta tarea.

4.6 Protocolos de comunicaciones

Para realizar las comunicaciones entre los diferentes dispositivos utilizados se hace uso de protocolos estándares de comunicaciones, en los que no se va a profundizar, como TCP/IP, pero, además, es necesario diseñar mecanismos que permitan las comunicaciones entre los dispositivos de forma correcta. Todas las posibles comunicaciones que se producen en el sistema quedan reflejadas en la Figura 25.

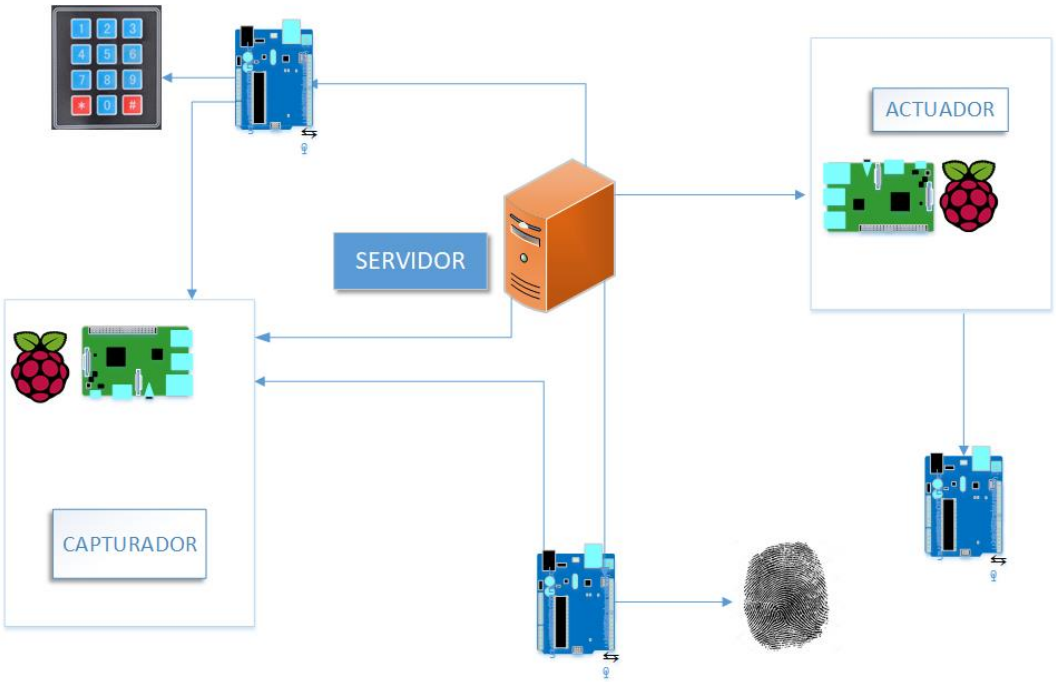

Figura 25 Comunicaciones en el sistema

4.6.1 Comunicaciones Servidor – Capturador (Rpi)

Las comunicaciones entre el servidor y un cliente capturador se pueden establecer siguiendo dos procedimientos diferentes en función del servicio del que se trate. Por un lado, las comunicaciones iniciadas por un cliente que desea enviar un rostro al servidor para su identificación. Esta conexión se realiza a través del puerto **4433** del servidor. El cliente envía un código al servidor para indicar el procedimiento a seguir, las opciones disponibles son:

a) '**720'** si se envía una imagen completa capturada por el cliente. Será el servidor quien se encargue de detectar el rostro y procesar la imagen. Reduce el procesado en el cliente, pero aumenta la carga del servidor. Podría ser necesario en capturadores que utilicen Raspberry Pi Zero. El tamaño de la imagen debe ser 1080x720.

b) '**192'** si se envía una imagen de un rostro con el formato establecido de 168x192.

c) ‘**código’** si se envía el código introducido en el teclado junto a una imagen con el formato 168x192.
d) ‘**huella’** si se va a enviar el código reconocido con el lector de huellas junto a la imagen.
e) ‘**alarma’** si se está recibiendo una alarma generada por un capturador.

En los dos primeros casos se procede a realizar un reconocimiento directamente. En los dos siguientes, además del reconocimiento facial, se realiza una comprobación de que la identidad reconocida con ambos métodos coincide.

Por otro lado, existe la opción de que sea el servidor el que inicie la comunicación accediendo al puerto **4444** del servidor, disponible para emitir imágenes capturadas en tiempo real. En este caso el servidor conecta y el cliente comienza a enviar imágenes capturadas directamente.

4.6.2 Comunicaciones con lector de huellas

Para que los equipos Raspberry, o el servidor, se puedan comunicar con el lector de huellas es necesario utilizar una placa Arduino intermedia, por lo que es necesario gestionar dos comunicaciones diferentes. La conexión entre la Raspberry Pi y la placa Arduino se realiza utilizando el puerto serie a 9600bps, mientras que la conexión entre Arduino y el lector se realiza conectando los cables del dispositivo con los puertos de la placa, además, en este caso la comunicación se realiza, forzosamente, a 57.600 bps (aunque esta limitación depende del modelo).

Las comunicaciones son iniciadas por la Raspberry que envía un carácter ASCII a Arduino, y este actúa en función del carácter recibido:

- ‘8’: Para iniciar las comunicaciones.
- ‘9’: Para solicitar identificación del dispositivo. Necesario para distinguir entre lector de huellas y teclado. El lector de huellas devuelve ‘2’ para identificarse.
- ‘1’. Se va a proceder a dar de alta un usuario.
- ‘2’. Se va a realizar la comprobación de la identidad de un usuario.
- ‘3’. Se va a proceder a eliminar una huella registrada en el lector.

Además de este sencillo código se envían múltiples mensajes, en función del estado, de Arduino a la Raspberry o el Servidor:

- ‘2’. Lector encontrado.
- ‘3’. Lector no encontrado
- ‘4’. Esperando id de usuario para crear uno en el registro del lector
- ‘5’. Comenzando alta de usuario nuevo

- ‘6’. Hay que poner el dedo sobre el sensor.
- ‘7’. Tiempo agotado.
- ‘8’. Fin de Reconocimiento de huella.
- ‘9’. Huella detectada correctamente.
- ‘10’. Sin huella.
- ‘11’. Error en las comunicaciones.
- ‘12’. Error con la imagen.
- ‘13’. Error desconocido.
- ‘14’. Imagen sucia.
- ‘15’. Fallo en las características de la huella.
- ‘16’. Imagen invalida.
- ‘17’. Mala ubicación del dedo.
- ‘18’. Huella movida.
- ‘19’. Comparada con éxito.
- ‘20’. Error.
- ‘21’. Encontrada coincidencia.
- ‘22’. Sin coincidencia.
- ‘23’. Se va a proceder al envío del id con el que coincide.
- ‘24’. Se va a proceder al envío de la distancia entre huella capturada y la huella registrada por el usuario identificado.

En las comunicaciones con un capturador se añade la posibilidad de que el usuario pulse dos botones. Estos envían los caracteres ASCII ‘4’ y ‘5’ respectivamente, desde Arduino a la Raspberry pi, el primero de ellos avisa al capturador de que un usuario quiere iniciar el proceso de escaneo de huella y la correspondiente captura facial para ser enviadas al servidor. El segundo indica al capturador que envíe una señal de alerta al servidor. Esta funcionalidad no existe en el lector de huellas que se conecte al servidor.

4.6.3 Comunicaciones con el teclado numérico

Es un caso similar al del lector de huellas, existen dos comunicaciones diferentes, por un lado, el servidor o la Raspberry que se comunican con la placa Arduino intermedia, de nuevo esta comunicación se realizará por el puerto serie a 9.600 bps.

Por otro lado, las comunicaciones con el teclado numérico se realizarán mediante el cableado específico del teclado que se conectará a los puertos de Arduino. De modo similar al anterior se ha diseñado un código para las comunicaciones, aunque más sencillo en este caso, dado que solo se necesita un mensaje de inicio, un ‘9’ para comenzar y

también para solicitar un identificador de dispositivo. Arduino devolverá un ‘1’ para identificarse como gestor de teclado.

El resto del tiempo Arduino se limita a enviar al dispositivo que inició las comunicaciones los valores marcados en el teclado.

4.6.4 Comunicaciones Actuador (Rpi) - Arduino

Las comunicaciones entre el dispositivo actuador y la placa Arduino responsable de controlar, en última instancia, las actuaciones, se realizan a través del puerto serie. Se ha definido una velocidad de conexión de 9.600 bps.

La comunicación será iniciada por el Actuador, que enviará un mensaje a la placa compuesto por un mínimo de tres caracteres. Los dos primeros serán los caracteres que indique si hay que encender/apagar o abrir/cerrar (ON/OF) el dispositivo electrónico que se está controlando. El resto se corresponde con el canal que hay que utilizar en la placa Arduino.

4.6.5 Comunicaciones Servidor - Actuador (Rpi)

El encargado de iniciar estas comunicaciones será el servidor. Se establecen a través del puerto 4455 del actuador, que se encuentra a la espera de recibir conexiones en este puerto y las correspondientes instrucciones.

El formato de las comunicaciones será similar al utilizado para las comunicaciones Actuador-Arduino, ya que el Actuador se limitará a reenviar las actuaciones.

4.6.6 Comunicaciones Capturadores (Rpi) – Arduino y sensor

De nuevo a través del puerto serie y con una velocidad de conexión de 9600pbs. En este caso las placas Arduino serán responsables de recibir la respuesta de los sensores utilizados, de distancia o de movimiento y comunicar al capturador que un usuario que se encuentra dentro de la distancia establecida o que se ha detectado un movimiento. En ambos casos Arduino enviará ‘1’ cuando se cumpla la condición y esperará un ‘9’ como confirmación. Cuando se quiera enviar una alarma desde Arduino se utilizará un ‘5’ en ambos casos.

5 IMPLEMENTACIÓN

5.1 Hardware

5.1.1 Servidor

El servidor no requiere un hardware muy específico. No obstante, es recomendable disponer de suficiente potencia para realizar el reconocimiento facial y el resto de funciones en un tiempo razonable. Las distintas opciones configurables en la aplicación permiten ajustar el tiempo de cómputo, pero a costa de perder seguridad o funcionalidades.

En cualquier caso, las necesidades finales varían en función del tamaño del edificio que se desee controlar y del número de usuarios registrados. Un aumento considerable en cuanto al número de usuarios podría ralentizar rápidamente el servidor e incluso llegar a hacerlo inviable si el equipo no es lo suficientemente potente.

5.1.2 Raspberry Pi 3

Para los clientes, tanto actuadores como capturadores se ha optado por utilizar una Raspberry Pi 3B, como la que se puede ver en la Figura 26. También se ha evaluado, como capturador, con una Raspberry Pi Zero W. La razón principal por la que se ha optado finalmente por este tipo de mini ordenador es el precio. La relación calidad precio que presenta en comparación con el resto de competidores es inmejorable. Existen equipos más potentes, con tamaño similar, pero con una diferencia de potencia que no compensa la diferencia en el coste. En cualquier caso, la potencia de la Raspberry Pi es suficiente para el propósito de este proyecto. Por otro lado, es, de entre todos los equipos valorados, el más popular y, por tanto, del que más documentación existe y el que más probado. Además de ser un equipo relativamente fácil de adquirir, en comparación con la competencia.

Recientemente lanzada al mercado, no se ha podido evaluar la versión 3B+, aunque las diferencias entre ambas versiones no son considerables. En algunas pruebas se ha probado con la versión Zero W, con resultados discretos en cuanto a tiempo de respuesta en comparación con la versión elegida. La recomendación, en cualquier caso, sería utilizar la versión más potente que sea posible y, solo si el espacio impide usar una versión estándar, utilizar la versión Zero W asumiendo que el rendimiento será inferior.

Figura 26 Raspberry Pi 3

Después de realizar diferentes pruebas con estos equipos se han obtenido resultados buenos en cuanto al tiempo de computación para la detección facial, incluidos en el apartado 6.6. También se han presentado algunas dificultades que son analizadas a continuación.

5.1.2.1 Problemas de calentamiento

Hay problemas de calentamiento con la Raspberry Pi cuando se está en pleno proceso de detección de imágenes. Se ha evaluado la temperatura que es capaz de alcanzar en distintos escenarios obteniendo los siguientes resultados:

- Sin ventilador ni montar disipadores sobre el microprocesador la temperatura media que se alcanza situando el equipo en un espacio bien ventilado es de entre 48 ºC y 52 ºC en estado de reposo. Sin embargo, si se inicia un proceso de detección es fácil alcanzar los 70 ºC y entrar en niveles críticos para la seguridad física del equipo. Algo menos de temperatura alcanza si solo se realizan constantes capturas (para emisión en directo, por ejemplo), pero aún pueden alcanzarse temperaturas superiores, incluso, si se intenta realizar reconocimiento, integrando todo el proceso, en la Raspberry.
- Con un ventilador estándar de una Raspberry y añadiendo disipadores, se consigue reducir en varios grados la temperatura alcanzada, sin embargo, la temperatura media de trabajo en detección sigue alcanzando niveles peligrosos para el equipo.

Existen ventiladores más potentes, se podría utilizar uno de 12 V, que requeriría una fuente externa, que podrían mantener la temperatura dentro de rangos apropiados para trabajar en detección e, incluso, en identificación. Sin embargo, el coste de estos elementos es alto, pudiendo incluso duplicar el coste de la propia Raspberry. Además, no siempre estará el cliente ubicado en espacios donde la ventilación sea suficiente ni el espacio apropiado.

Para reducir el riesgo de calentamiento, se ha optado por montar en los clientes mecanismos que inicien el proceso de detección permitiendo mantener la Raspberry en reposo el resto del tiempo.

Los mecanismos que se han incorporado para iniciar la captura de imágenes y la detección de rostros en las mismas son:

- **Detectores de movimiento**. El sensor HC-SR501 permite detectar cuando hay un movimiento. En la Figura 27 se puede ver uno de estos sensores.

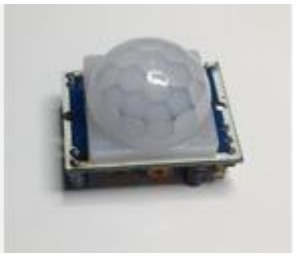

Figura 27 Sensor de movimiento

- **Sensor de distancia** por ultrasonidos. HC-SR04 es un sensor que calcula la distancia a un objeto gracias a ultrasonidos.
- **Pulsadores** que inician la captura, o algún procedimiento previo como el reconocimiento mediante huella o el uso de un teclado matricial.

5.1.2.2 Riesgos de los puertos GPIO

Los puertos de entrada/salida de propósito general o GPIO (*General Purpose Input/Output*) presentes en la Raspberry permiten conectar el equipo directamente con elementos electrónicos como relés que facilitarían la posibilidad de controlar dispositivos electrónicos que trabajen a voltajes superiores a los que utiliza la Raspberry. El inconveniente que tiene este uso es la posibilidad de que se produzcan variaciones puntuales en los voltajes o en la intensidad que podrían dañar la placa, dado que los puertos de la Raspberry no presentan ningún tipo de protección interna para estos casos. Además, la Raspberry trabaja con 3,3 V, lo que dificulta la conexión directa con algunos dispositivos, pudiendo requerir conversiones de voltaje con placas como la *BOB-12009 de SparkFun*, que convierte 5 V en 3,3 V.

Es importante tener especial cuidado, por tanto, con los voltajes y con el límite de 50mA que la Raspberry puede suministrar en total.

Este problema se ha resuelto utilizando placas intermedias, más adecuadas para ser conectadas directamente con componentes electrónicos. Las elegidas han sido las placas Arduino.

5.1.3 Cámaras

Se han evaluado las siguientes cámaras:

- Cámara para Raspberry Pi v2.1. En la Figura 28 se puede ver una de estas cámaras. Tiene una resolución de 8 Mega píxeles.
- Cámara para Raspberry Pi v2.1. NOIR. Similar a la anterior, pero sin filtro de infrarrojos incorporado.
- Cámara para Raspberry Pi v1. Se trata del primer modelo, con 5 Mega píxeles de resolución.
- Cámara Logitech C920HD Pro. Con un sensor de 15 Mega píxeles
- Webcam integrada en el servidor.

El modelo Logitech tiene un precio, 90 € aproximadamente, muy superior a los modelos diseñados para trabajar con la Raspberry Pi, entre 10 € y 25 € dependiendo del modelo, y los resultados obtenidos en las pruebas realizadas no presentan diferencias sustanciales. Por lo que se han elegido para este proyecto las cámaras diseñadas para la Raspberry Pi.

En general, un factor a tener en cuenta a la hora de elegir una cámara apropiada para usar con la Raspberry Pi es el voltaje requerido por las cámaras, algunos modelos requieren alimentación externa, y la posibilidad de configurarlas convenientemente para su uso en Raspbian.

Figura 28 Cámara para Raspberry Pi, versión 2.1

5.1.4 Arduino

Se han utilizado varios modelos de placas Arduino, en función de las necesidades y del espacio físico disponible. Por defecto se utiliza el modelo UNO. En los casos en los que el espacio físico es un hándicap es posible utilizar el modelo NANO, menos potente, pero con un tamaño muy inferior. Por último, si el número de pines de E/S de la placa era un factor de determinante se recomienda el uso de la placa MEGA.

5.1.5 *Lector de huellas R308*

Una vez evaluadas las alternativas disponibles el conjunto formado por el lector de huellas R308, una placa Arduino y la librería Adafruit («Adafruit Optical Fingerprint Sensor» 2018), estos demostraron ser la alternativa más económica y sencilla de implementar. Manteniendo, además, un bajo porcentaje de error, según el fabricante es inferior al 0,001% de falsos positivos e inferior al 0,1% de tasa de falso negativo. Existen diferentes fabricantes y, por ejemplo, la capacidad de almacenamiento puede variar entre las 162 y las 512 huellas almacenadas.

Para su implementación es suficiente con un sencillo cableado entre el lector y la placa Arduino. Además de utilizar unas librerías apropiadas, como las suministradas por Adafruit.

5.1.6 *Teclado numérico*

Con el objetivo de añadir un extra de seguridad se ha incorporado la opción de introducir un código de acceso personal (PIN) que, junto al reconocimiento facial, se pueda utilizar para reconocer usuarios.

Para su utilización se utiliza la librería Keypad («Keypad Library» 2018), así como la librería XXTEA («xxtea» 2018) ligeramente adaptada. Como hardware se utilizará, además del teclado numérico matricial, una placa Arduino o compatible modelo NANO, no es necesario recurrir a la versión UNO en este caso, y permite reducir el tamaño del dispositivo.

5.1.7 *Electrónica*

Además de los elementos mencionados ha sido necesaria la utilización de algunos componentes electrónicos para poder desarrollar el proyecto completo. Estos elementos son:

- Pantallas LCD/OLED
- Sensores de movimiento y distancia
- Leds, resistencias y otros componentes electrónicos.

5.2 Herramientas utilizadas

Para llevar a cabo este proyecto se han utilizado las siguientes herramientas: **Visual Studio 2017** versión Community, se trata de la versión gratuita de Visual Studio (Webster 2018), un entorno de desarrollo (IDE) que permite programar en multitud de lenguajes y para diferentes plataformas. Se ha utilizado para desarrollar, en lenguaje **C++**, la aplicación Servidor de nuestro sistema. Aunque existen otras alternativas, en cuanto al lenguaje de programación, se ha optado por utilizar C++ al ser el mismo lenguaje que se utiliza para programar las placas Arduino. Es necesario configurar Visual Studio para que puedan

integrarse en el programa las librerías de **OpenCV, OpenSSL** y las de la base de datos **Sqlite**. En el manual de mantenimiento, incluido en el Anexo II, se explica con más detalle.

Por otro lado, también se ha desarrollado en Visual Studio el código que permite recuperar las imágenes y el video transmitidos sin cifrar y que han sido capturados con Wireshark en las pruebas de seguridad. **Wireshark** («Wireshark · Go Deep.» 2018) es un software gratuito que permite analizar el tráfico de una red. Entre sus funciones están la captura y seguimiento de paquetes. Se ha utilizado para capturar el tráfico de la red, analizarlo y realizar pruebas de seguridad.

Para poder trabajar en OpenCV con reconocimiento de patrones es necesario entrenar el sistema, proporcionándole imágenes del objeto que se quiere detectar junto con imágenes donde no se encuentra el objeto, lo que se conoce como imágenes positivas y negativas. Para el caso concreto de la detección facial es posible realizar el entrenamiento haciendo uso de una base de datos de rostros, pero la librería ya incorpora una serie de archivos de lenguaje de marcado extensible o XML (*Extensible Markup Language*) entrenados previamente y que evitan tener que realizar el entrenamiento. Estos archivos se encuentran en el directorio *haarcascades* incluido en la mayoría de versiones de OpenCV. Hay diferentes versiones, y no solo para detectar caras, también permiten la detección de boca y ojos entre otros. En este trabajo se han utilizado *haarcascade_eye_tree_eyeglasses.xml* para la detección de ojos y *haarcascade_frontalface_alt2.xml* para la detección facial.

Thonny, un sencillo IDE, incluido en las distribuciones del sistema operativo Raspbian, software libre por tanto, que permite programar de forma sencilla en lenguaje **Python**, cuya versión 2.7 ha sido la utilizada para la programación en Raspberry Pi.

Las placas Arduino han sido programadas utilizando el entorno de desarrollo propio de la plataforma Arduino, conocido como **Arduino IDE**, versión 1.6.9, y utilizando C++ como lenguaje de programación.

Además de las herramientas y lenguajes mencionados se hace uso de las siguientes librerías:

- **Dlib** («dlib C++ Library» 2018). Se trata de una potente librería desarrollada en C++ con múltiples funciones, entre las que se encuentran el aprendizaje máquina. Además, se utiliza el modelo entrenado para localizar los puntos de referencia del rostro, *shape_predictor_68_face_landmarks.dat*, y que servirá para contabilizar el parpadeo, como se verá más adelante.
- **Numpy** («NumPy» 2018). Añade a Python mayor potencia para el tratamiento de vectores y matrices. Necesaria para hacer uso de OpenCV en Python de forma eficiente.
- **Imutils** (Rosebrock 2018). Librería de Python con múltiples utilidades.

- **Keypad** («Keypad Library» 2018). Librería para poder usar un teclado numérico con Arduino.
- **Adafruit fingerprint sensor** («Adafruit Optical Fingerprint Sensor» 2018). Librería para usar el lector de huellas R308.
- **XXTEA** para Python («xxtea» 2018) y para C++ («Mini algoritmo de cifrado XXTEA para arduino | Heli» 2018).
- Otras librerías, como las utilizadas para el control de las pantallas OLED o LCD utilizadas en los dispositivos.

5.3 Software en el servidor

5.3.1 Diagrama de clases

En la Figura 29 se presenta un diagrama de clases simplificado. En él se han representado las principales clases utilizadas en la programación del servidor. En los siguientes apartados se explicarán con detalle estas clases.

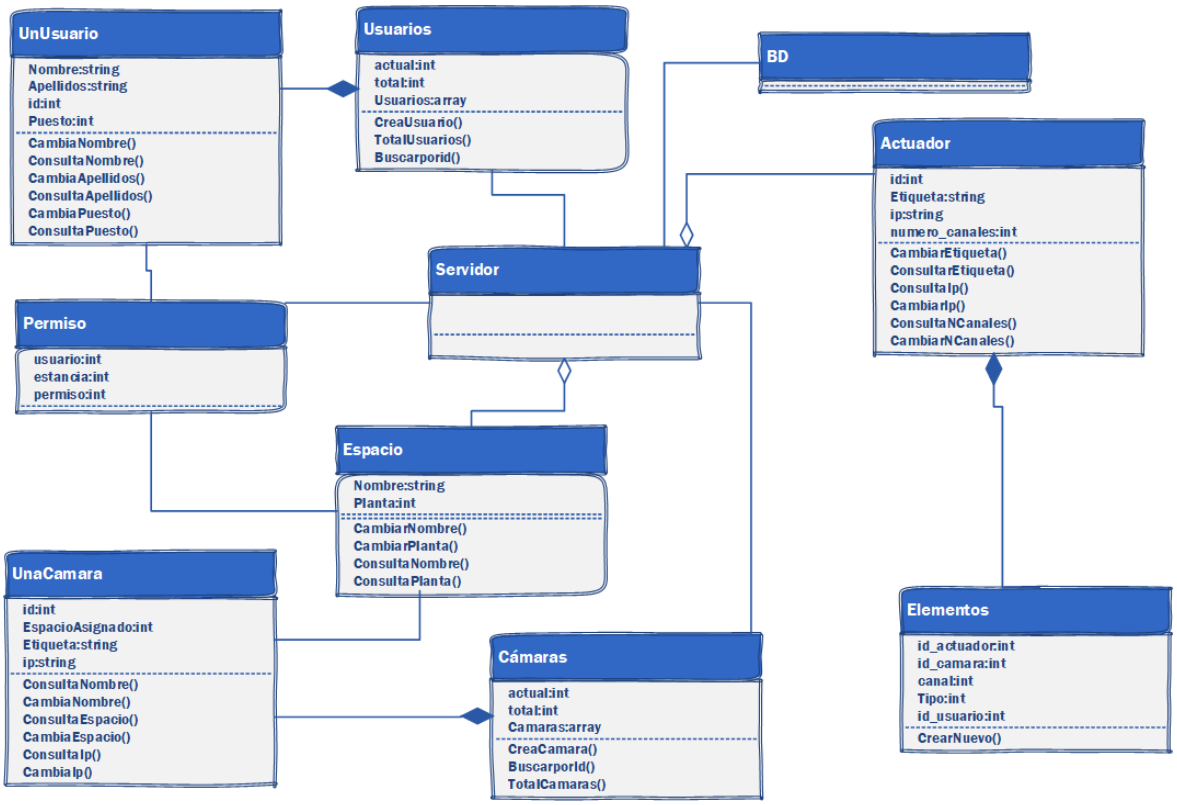

Figura 29 Diagrama de clases simplificado

5.3.2 Clases Principales

Se listan a continuación las principales clases utilizadas para desarrollar la aplicación del servidor principal.

1. **BD**.

Es la responsable de iniciar y gestionar las comunicaciones y consultas con la base de datos Sqlite incluida en la aplicación. Dispone de todos los procedimientos necesarios para consultar y modificar las tablas necesarias. Al arrancar el sistema se comprueba si existen en la base de datos todas las tablas necesarias, en caso de no existir se crean.

2. **Servidor**.

Representa el objeto responsable de iniciar todos los servicios del sistema, lo que incluye iniciar los sockets que quedan en espera de la conexión de un cliente y realizar el entrenamiento necesario en los sistemas de reconocimiento. También será responsable de instanciar el resto de objetos necesarios para el correcto funcionamiento del sistema.

Para el servidor se ha optado por usar un hilo para cada conexión. Permitiendo conexiones simultaneas. Un sistema con pocas conexiones por parte de los clientes deja más tiempo ocioso para el servidor, y, por tanto, podría no ser necesario el uso de hilos. Sin embargo, si empiezan a coincidir en tiempo las conexiones se podría alcanzar un punto crítico que haría peligrar la fiabilidad del sistema.

Hay que destacar que en la programación con hilos y formularios, en C++ con Visual Studio, se requiere el uso de funciones delegadas, para poder modificar el contenido de los formularios desde procesos diferentes del que creo el formulario a modificar.(dotnet-bot 2018).

Las principales funciones implementadas en esta clase son:

a) ***ServidorSeguro***. Responsable de crear el socket principal del servidor, creando hilos para cada una de las conexiones establecidas por los capturadores. Se utiliza el puerto 4433 para establecer la conexión, cuando esta conexión pasa a un nuevo hilo utilizará un puerto diferente y el 4433 vuelve a permanecer en escucha a la espera de una nueva conexión. Es el encargado de configurar los protocolos de cifrado para las comunicaciones utilizando OpenSSL.

b) ***Carga_modelo***. Responsable de cargar los modelos de reconocimiento facial en función de las opciones seleccionadas en el sistema, como el directorio de las imágenes o el valor umbral para el límite del reconocimiento para cada algoritmo.

c) ***Nuevo_hilo*** / ***nuevo_hilo_sin_cifrar***. Funciones responsables de iniciar hilos que gestionan las comunicaciones con y sin cifrado. El primer dato recibido una vez se establece la comunicación establece si el cliente está enviando una imagen que contiene un rostro, en cuyo caso se recibe el alto de la imagen, o se trata de una imagen acompañada del identificador obtenido por el usuario al utilizar un lector de huella en el

capturador, o si está acompañada por el código introducido en el teclado numérico introducido en el capturador. En función de los datos recibidos se realizan las comprobaciones oportunas para verificar si la imagen del usuario es reconocida positivamente y, si fuera necesario, comprobar si el identificador obtenido por el lector de huella o el código introducido coinciden, también, con el usuario reconocido. Si, finalmente, todos los resultados son positivos se lanzan las oportunas instrucciones a los actuadores. Esta función es, también, responsable de deserializar la imagen recibida.

d) ***Directo_seguro*** / ***Directo***. Responsables de conectar con un actuador, con y sin cifrado, para mostrar en directo, a través del interfaz, lo que la cámara está capturando. Permite realizar capturas para crear un nuevo usuario.

e) ***ComprobarActuaciones***. Encargada de recolectar las actuaciones que deben iniciarse después de un reconocimiento positivo y en función del usuario reconocido y el actuador que ha enviado la imagen del rostro.

f) ***Envia_actuador***. Función que permite al servidor comunicarse con el actuador para dictarle las oportunas instrucciones obtenidas por la función anterior.

3. **Usuarios**.

Es el objeto que contiene el vector con cada uno de los usuarios registrados, es decir, es un vector de objetos de tipo Usuario. Sus datos son cargados por el objeto servidor después de ser leídos de la base de datos a través del objeto BD. Permite crear nuevos usuarios y gestionar el vector que los contiene.

4. **Usuario**.

Contiene información de un usuario y proporciona procedimientos para consultarla y para poder modificarla.

5. **Cámaras**.

Es el objeto que gestiona y crea las cámaras o capturadores. Al igual que usuarios es cargado por el servidor con los datos extraídos de la base de datos.

6. **Cámara**.

Contiene información relativa a un capturador.

7. **Espacio**.

Contiene información sobre un espacio o servicio cuyo acceso será controlado. Son gestionados por el servidor donde se crea un vector de espacios.

8. **Permiso**.

Almacena información de un permiso y es gestionado, de forma similar a espacio, por el servidor directamente mediante un vector.

9. **Actuador**.

Representa un cliente actuador. Permite consultar y modificar sus datos. Son gestionados a través de un vector

10. **Elementos**.

Se corresponden con cada una de las posibles acciones de un actuador a través de los canales de que dispone.

5.3.3 Implementación del reconocimiento facial

El algoritmo que se utilizará depende de la configuración del servidor. Se ofrece la posibilidad de utilizar LBPH, Eigenfaces y Fisherfaces e, incluso, los tres a la vez, todos ellos incluidos en OpenCV.

Cuando se utilizan los tres a la vez se entiende que existe reconocimiento positivo si, al menos, dos de ellos coinciden en el reconocimiento. El resultado mostrado incluirá, además del identificador del usuario reconocido, el valor de confianza obtenido en el procedimiento LBPH (aunque hay que tener en cuenta que podría darse el caso de que, precisamente, LBPH hubiera identificado otro usuario).

Un factor a tener en cuenta a la hora de cargar las imágenes con diferentes algoritmos son el tamaño de las mismas, aunque es muy recomendable que todas tengan el mismo tamaño, si se utiliza LBPH no da error, pero en Eigenfaces y Fisherfaces sí que darían error, por lo que es obligado que tengan el mismo tamaño. Además, Fisherfaces requiere que haya, al menos, dos usuarios registrados en el sistema para comenzar, una de las restricciones a la hora de usar LDA, utilizado en Fisherfaces, es la necesidad de disponer de, al menos, dos grupos y, para cada grupo, deben existir dos o más casos. De no cumplirse esta condición se produce un error.

Las imágenes recibidas por los capturadores serán recibidas en escala de grises, pero si se lee una imagen desde un archivo o se utiliza una cámara local será necesario modificarla, pues los algoritmos utilizados para el reconocimiento facial trabajan con imágenes en escala de grises.

En cuanto al tamaño de imagen, después de analizar los estudios al respecto, se ha optado por utilizar imágenes con formato PGM (*Portable GrayMap*) de tamaño 168x192 pixeles para almacenar los rostros.

A la hora de hacer el reconocimiento es recomendable que el usuario se encuentre frente a la cámara y con la cabeza lo más recta posible, si el ángulo de inclinación de la cabeza supera los 15º el rostro podría no ser detectado. Además, a la hora del reconocimiento podría influir negativamente. Para mejorar los resultados se ha incorporado un pequeño algoritmo (Baggio 2012) que, después de localizar la posición de ambos ojos, calcula la inclinación del rostro y la corrige, además de recortar el rostro a un tamaño más ajustado que el que se obtiene con una simple detección, se obtienen mejores resultados aunque el tiempo que requiere para ser procesado puede ser un problema en determinados casos.

Además, para reducir los efectos producidos por cambios de iluminación se realiza una normalización del histograma[4] de las imágenes, tanto al capturar una imagen para realizar un reconocimiento como al registrar un nuevo usuario. . Este procedimiento consiste en modificar el histograma de la imagen para que se utilice el rango completo de valores en la imagen en escala de grises (de 0 a 255). Este efecto puede verse en la Figura 30, donde aparece una imagen en escala de grises y su transformación. Los histogramas de ambas imágenes aparecen en la Figura 31. El nuevo histograma reparte sus valores en el rango 0-255.

Figura 30 Normalización de una imagen

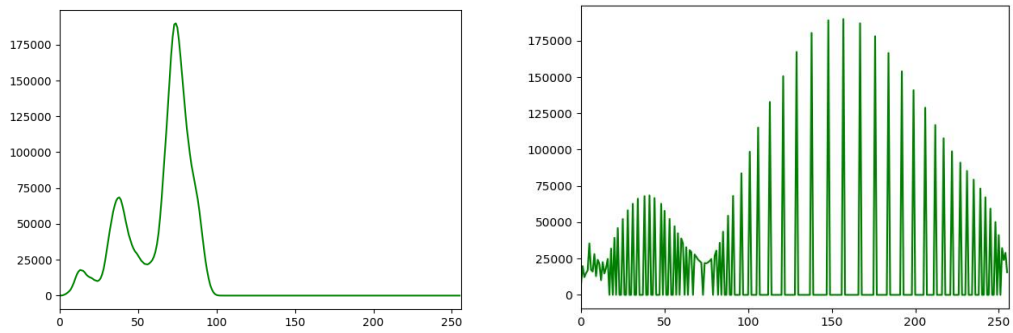

Figura 31 Histogramas

Cuando se registra un nuevo usuario en el sistema se incluyen las imágenes del rostro obtenidas con la detección, después de ser normalizadas, y las procesadas con el algoritmo de rotación y recorte, además de las imágenes volteadas de ambas para ayudar

[4] Gráfica que muestra el número de píxeles de cada nivel de gris que hay en una imagen.

a corregir problemas puntuales con la fuente de iluminación. Almacenar las imágenes previas a la rotación permite realizar reconocimientos más rápidos cuando se hacen directos locales y remotos ya que en estos casos las imágenes de los rostros detectados no son procesadas con este algoritmo para no sufrir demasiado retraso en el procesado de las imágenes. Aunque estos resultados, en general, no son buenos ni fiables.

No obstante, se ha incluido un botón que permite realizar un reconocimiento completo, incluyendo la rotación de la cara, de las imágenes que se están emitiendo en directo, y que es similar al que se realizará a las imágenes provenientes de capturadores.

OpenCV permite guardar el sistema entrenado en un archivo YML, que permite iniciar el sistema sin necesidad de realizar un entrenamiento cada vez, ahorrando tiempo en el proceso.

5.3.4 Gestión de los dispositivos

El interfaz de usuario proporciona acceso a diferentes formularios que permiten gestionar cada uno de los dispositivos. Su funcionamiento, con detalle, es explicado en el Anexo I.

Para el espacio administrador se reserva el espacio con identificador uno. Además, para facilitar el acceso a este espacio y su gestión se reserva el primer canal del actuador con identificador uno para controlar el acceso a este espacio.

Los identificadores de los lectores, para simplificar su gestión, utilizarán identificadores correlativos en función del capturador al que van asociados, de modo que el que tenga identificador igual a uno tendrá asociados los lectores uno y dos, que se corresponderán con el teclado y el lector de huellas; el capturador dos tendrá reservados los lectores tres y cuatro; y así sucesivamente.

Para poder usar el lector de huellas desde el exterior se han configurados los botones ‘*’ y ‘#’ para empezar el código e intentar el acceso, respectivamente. La tecla ‘#’ equivale a iniciar un inicio de sesión en el menú de la aplicación y realizará las mismas comprobaciones. Por tanto, para realizar un inicio de sesión desde el exterior de la sala requeriría la utilización del teclado para, al menos, utilizarlo para iniciar del procedimiento.

5.3.5 Gestión de las actuaciones

La gestión de las acciones se controla gracias a la base de datos, en la tabla elementos. En ella los campos *id_usuario* e *id_camara* indican cuando se debe cambiar el estado del dispositivo conectado a un canal de un actuador. Los valores permitidos para estas variables son un identificador de usuario o el valor -1. En ambos casos el valor -1 indica que se ejecutará sea cual sea el usuario o la ubicación, como se aprecia en la Tabla 1. Por tanto, las acciones correspondientes se realizan si el usuario reconocido está

autorizado, y las acciones a realizar serán aquellas contenidas en la tabla elementos y que cumplan las condiciones de la tabla.

Id_usuario	Id_camara	Actuación
-1	**-1**	Se realiza siempre que haya un reconocimiento autorizado
Y	**-1**	Se realiza cuando el usuario Y es reconocido, sea cual sea su ubicación
-1	**Y**	Se realiza cuando cualquier usuario permitido es reconocido en la cámara Y
X	**Y**	Se realiza cuando el usuario X es reconocido en el espacio Y

Tabla 1 Opciones

Para el acceso al espacio como administrador solo será necesario tener permiso para acceder al espacio con identificador igual a uno, no será necesario añadir una actuación específica. Este procedimiento solo se produce en este caso, en cualquier otro si es necesario incluir una acción dentro del listado de elementos.

5.3.6 Configuración del servidor

Para facilitar la configuración del servidor se ha dejado abierta la posibilidad de modificar algunos de los parámetros del sistema y adecuarlos al entorno en que se haya implementado. Para poder modificarlos es necesario autenticarse como administrador.

El método para la identificación de un administrador se establece en las opciones del sistema.

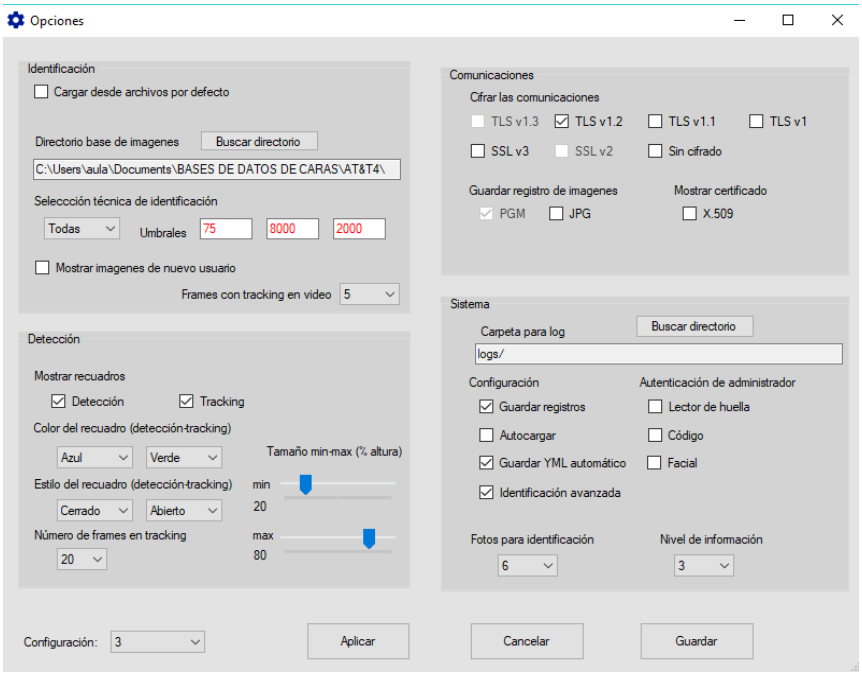

Figura 32 Opciones configurables en el sistema.

En la Figura 32 se muestra el formulario del sistema que permite establecer la configuración del mismo. Las múltiples opciones que se pueden configurar se pueden ver con más detalle en el manual del usuario, incluido en el Anexo I.

Cuando se aplican cambias en la configuración actual no todos se pueden aplicar de forma inmediata, como los cambios en el algoritmo a utilizar, y sería necesario guardarlos y reiniciar.

5.3.7 Comunicaciones cifradas

Las comunicaciones con los clientes serán cifradas haciendo uso de OpenSSL. Se inician haciendo uso de un socket. Posteriormente se envuelve el socket haciendo uso de una capa SSL con un contexto específico. Una vez se envuelve la comunicación se comprueban los certificados y se negocia el protocolo de cifrado; en el servidor se pueden establecer cuáles serán soportados.

Para el cifrado de las comunicaciones se utilizará una autenticación de dos vías: la aplicación cliente verifica la identidad del servidor y luego será el servidor quien verificará la identidad del cliente. Aunque es posible utilizar autoridades que certifiquen los certificados, se usarán auto-firmados; es necesario que el servidor tenga el certificado del cliente y viceversa para que las comunicaciones cifradas puedan realizarse. En el ANEXO II. Manual de mantenimiento se especifica todo el proceso con más detalle.

En caso de hacer uso de dispositivos auxiliares, lector de huellas o teclado numérico, las comunicaciones con estos son cifradas con XXTEA utilizando una clave de 16 bytes fijada previamente. Todo lo relacionado con este cifrado se encuentra en el archivo *xxtea.h* del servidor.

5.3.8 Otras funcionalidades

El servidor ofrece otras funciones menores que, sin ser imprescindibles, proporcionan valor añadido:

- **Seguimiento** (*tracking*). Se ha utilizado el trabajo de Zhang (Zhang, Zhang y Yang 2012). Inicialmente se implementó el algoritmo propuesto por OpenCV («OpenCV: tracking» 2012) pero los resultados no fueron satisfactorios. El algoritmo de seguimiento implementado sirve para seguir un rostro detectado. En el sistema diseñado no tiene ninguna función específica, pero permitiría incorporar en el futuro funciones de seguimiento con cámaras móviles o realizar el seguimiento de objetos que no necesariamente tendrían que ser un rostro. Se ha implementado tanto en directos como en la reproducción de videos, en ambos casos utiliza una imagen de un rostro detectado como entrada para el algoritmo de seguimiento, el cual realiza esta función durante un número de fotogramas

o imágenes determinado (modificable en opciones, en ambos casos). Una vez termina se vuelve a realizar una detección facial. Mientras se está realizando seguimiento no se realizan detección ni identificación.

- **Reconocimiento en videos e imágenes.** Es posible abrir imágenes, en jpg o pgm, y videos, avi o mov, para realizar detección y reconocimiento en ellos. El proceso es el mismo que si se tratara de una imagen recibida desde un cliente, en el caso de las imágenes.

  Para el caso del video, el servidor podría no ser capaz de reproducirlo a la velocidad adecuada. Para evitarlo se ha utilizado el algoritmo de tracking, introduciéndolo intercalado con el reconocimiento; además, se reduce el tamaño del video, si es superior a 576 píxeles de altura, para que no se ralentice en exceso.
- **Pruebas en Actuadores.** Con el fin de poder testear los diferentes canales disponibles en los actuadores, se ha abierto la posibilidad de realizar pruebas, pudiendo enviar tanto instrucciones de encendido, ON, como de apagado, OF. Se pueden realizar desde el listado de actuaciones.

5.4 Base de datos SQLite

La gestión del sistema requiere el uso de una base de datos. Existen alternativas gratuitas, como MySQL, pero, en este caso, se ha optado por utilizar una sencilla base de datos SQLite, que se incluye dentro del mismo servidor.

5.4.1 Modelo E/R

La base de datos utilizada sigue el modelo entidad relación que se muestra en la Figura 33.

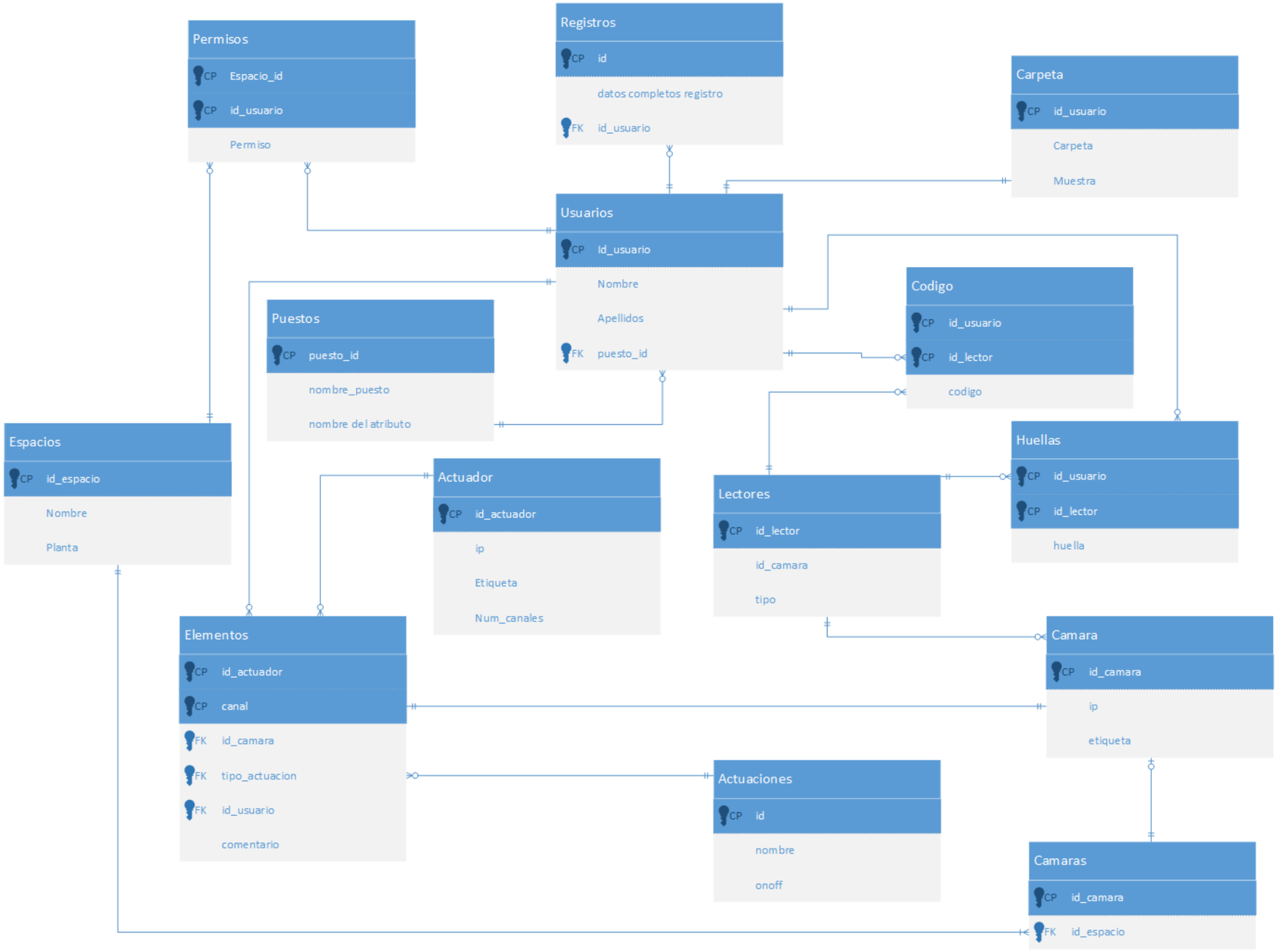

Figura 33 Modelo E/R de la base de datos (I)

Las principales tablas utilizadas por el sistema son:

- **Usuarios**. Contiene la información relativa a cada uno de los usuarios registrados.
- **Puestos**. Contiene una relación de los diferentes puestos de la empresa. Aunque no se ha utilizado en el presente trabajo podría añadirse la funcionalidad de gestionar permisos en función del puesto.
- **Carpeta**. Permite establecer el directorio en el que se encuentran las imágenes del usuario, así como su imagen de muestra, utilizada por la interfaz de usuario, como imagen por defecto para el usuario en cuestión.
- **Cámara**. Contiene los datos relativos a un cliente capturador, que dispone de una cámara. Incluido una etiqueta identificativa y su dirección de red.
- **Espacio**. Corresponde con el listado de espacios del edificio, también se deben incluir en ella los servicios que requieran autorización del sistema para su uso. No se hace distinción entre espacios o servicios proporcionados.
- **Cámaras**. Relaciona cada cámara con el espacio o servicio al que da acceso.
- **Huellas** y **Código**. Almacenan la información relativa a los dispositivos que proporcionan una capa extra de seguridad, lectores de huella y teclados numéricos respectivamente. Relaciona código interno del lector de huellas o el pin para el teclado numérico con el id del usuario al que corresponde.
- **Lectores**. Indica en que capturador o cámara se encuentra el dispositivo correspondiente.
- **Permisos**. Almacena el tipo de permiso que tiene un usuario en un espacio concreto. Además del tipo de permiso. Aunque en este trabajo solo se ha contemplado la opción de autorizado o no autorizado se ha diseñado como tipo entero el posible permiso para facilitar expansiones que contemplen diferentes niveles de permisos.
- **Actuador**. Contiene información sobre un cliente actuador, incluida su dirección de red.
- **Elementos**. Cada actuador dispone de una serie de canales utilizables, en esta tabla se almacena información sobre cada canal del actuador y que condiciones deben darse para cambiar de estado. Que usuarios reconocidos o que espacios en los que se ha producido un acceso autorizado implican un cambio. Aunque la tabla permisos establece quien

está autorizado es la tabla elementos la que realmente contiene los procedimientos que se desencadenan.

- **Actuaciones**. Contiene un listado con las posibles actuaciones y que valor lógico corresponde con el encendido, apertura o activación de este tipo de dispositivos.

Además de las tablas mencionadas, es necesario disponer de una serie de tablas de configuración, donde se almacenan las opciones elegibles en la Figura 32:

- **Opciones_comunicaciones**.
- **Opciones_sistema**.
- **Opciones_detección**.
- **Opciones_reconocimiento**.
- **Opción**.

Existe la posibilidad de guardar cuatro configuraciones diferentes, seleccionando la activa en el formulario de opciones. La elegida queda almacenada en la tabla opción.

5.5 Software en capturadores

Para el funcionamiento de los clientes capturadores se han desarrollado los siguientes programas, todos desarrollados en Python:

- Capturador básico.
- Capturador con detector de movimiento.
- Capturador con sensor de distancia.
- Capturador sin cifrado en las comunicaciones.
- Capturador con teclado numérico.
- Capturador con lector de huella.
- Capturador con control de parpadeo.
- Servidor.

Para completar un capturador, además de una Raspberry Pi y una cámara, que formarían el modelo básico, pueden ser necesarios otros elementos; principalmente se hace uso de placas Arduino. En estos casos el software necesario para las placas Arduino ha sido programado en C++, utilizando el entorno Arduino IDE («Arduino - Software» 2018).

Todas las comunicaciones con Arduino son cifradas con XXTEA utilizando una clave de 16 bytes fijada previamente. Así será en todos los capturadores en los que se requiera el uso de una placa Arduino.

Gracias a todas estas alternativas es posible disponer de capturadores muy flexibles, que se pueden configurar en función de las necesidades específicas.

5.5.1 Capturador básico

Este programa representa el sistema de captura más básico. Más útil en fase de desarrollo que de explotación. No hay que olvidar los problemas de calentamiento detectados. Por lo que no se recomienda utilizar este script, salvo para realizar algún tipo de prueba.

Para su funcionamiento es necesario que el sistema disponga de la librería OpenCV instalada, además, por supuesto, de contar con una cámara. El sistema está realizando constantemente capturas de imágenes con la cámara y sobre estas realizar una detección facial. Cuando un rostro es detectado se inicia un contador de cinco segundos, tiempo para que el usuario se posicione y se acerque a la cámara. Además de evitar enviar al servidor imágenes constantemente y de baja calidad, sobre todo por el movimiento del usuario desde que es detectado hasta que se encuentra a la distancia y en la posición recomendada.

Una vez el tiempo dispuesto termina se realiza una nueva captura del rostro. Si no hay detección facial, normalmente porque el usuario se ha desplazado en el momento de la captura, el sistema continuará realizando capturas.

Si, por el contrario, el sistema detecta un rostro, esta imagen es capturada, convertida en escala de grises, es serializada para ser enviada al servidor. Para ello se hace uso de la función *reshape* de Python, que convierte la matriz que utiliza el tipo de dato Mat de OpenCV en un vector.

El envío de la imagen se produce después de que el cliente y el servidor negocien los protocolos de seguridad. Estos dependerán de la configuración del servidor en cuanto a versión y protocolo utilizar. Una vez se establece la comunicación el cliente envía la altura de la imagen enviada en píxeles y, posteriormente, la imagen serializada. Siguiendo el protocolo establecido.

Una vez se ha concluido con el envío se cierran las comunicaciones y la aplicación continua con una nueva detección.

5.5.2 Capturador con detector de movimiento

Se trata de un sencillo script que se mantiene el equipo en reposo hasta que un sensor de movimiento conectado a un Arduino detecta actividad y envía una señal de aviso al capturador para que abandone el estado actual e inicie las capturas con la cámara. A partir de ahí el funcionamiento es similar al del script anterior. Con la salvedad de que cuando se detecta un rostro el programa envía un mensaje a la misma placa Arduino utilizada para la detección de movimiento para que muestre en una pantalla LCD, conectada para tal efecto, un contador descendente que avisa de una nueva captura.

Dado que se dispone de una pantalla se ha utilizado esta para mostrar mensajes sobre el estado del capturador:

- EN ESPERA: Indicando que no se ha detectado movimiento
- DETECTADO: Indica que se ha detectado movimiento y, por tanto, se está en periodo de detección.
- CONTADOR: En este caso se trataría de un contador de tiempo descendente desde cinco a cero. Una vez termina el contador se realiza la captura que será enviada.

Los mensajes mostrados en el LCD se encuentran en la programación de la placa Arduino. Cuando un rostro es detectado se envía un mensaje de activación a la placa Arduino, que comienza la cuenta atrás en la pantalla. Mientras el script permanece en un bucle infinito a la espera de recibir el mensaje que indica que el contador ha llegado a 0 por parte de la placa Arduino.

El Hardware que debe acompañar a este software es, además de la Raspberry Pi, una placa Arduino, con su correspondiente configuración software, y un sensor de movimiento.

5.5.3 Capturador con sensor de distancia

Similar al anterior, salvo que en este caso se utiliza un sensor ultrasónico de distancias, HC-SR04 que activa la detección cuando detecta movimiento continuo a una distancia inferior a 25 cm. durante, al menos, dos segundos y medio. Una pequeña pantalla OLED muestra indicaciones.

El Hardware necesario es similar al anterior con sensor de movimiento, modificando el software de Arduino y el sensor utilizado.

5.5.4 Capturador sin cifrado en las comunicaciones

De nuevo se trata de un código diseñado para pruebas. Es similar al código del capturador básico, pero en este caso se ha eliminado el cifrado de las comunicaciones. Se trata de un script diseñado para pruebas. Su uso no está recomendado, primero por el calentamiento, y segundo por la ausencia de medidas de seguridad en las comunicaciones. En el capítulo correspondiente a pruebas y resultados se puede comprobar la fragilidad de un sistema sin cifrado, como sería este.

5.5.5 Capturador con teclado numérico

Similar al descrito con el sensor de movimiento, pero en este caso las capturas se inician cuando se introduce un código de cuatro dígitos mediante un lector numérico conectado a una placa Arduino con la que se comunicará con la Raspberry pi. En este caso, además, justo antes enviar la imagen al servidor, se envía el código introducido para su evaluación, junto con la imagen del rostro.

Las comunicaciones se inician cuando se envía un ‘9’ a la placa Arduino a la que se ha conectado el teclado numérico y esta responde con un ‘1’, también cifrado, que sirve para diferenciar estas placas de las que están conectadas con un lector de huellas. Para facilitar su uso este protocolo se puede usar siempre que exista comunicación. El resto del tiempo la placa Arduino se limita a enviar el número tecleado, siempre cifrado.

5.5.6 Capturador con lector de huella

El primer paso es iniciar el lector de huellas siguiendo las indicaciones establecidas cuando se definieron los protocolos de comunicaciones para este proyecto. Esto es, iniciando las comunicaciones con la placa Arduino que controla el lector de huellas enviando un carácter ‘9’, y, a continuación, tras recibir un ‘2’ como respuesta, enviar un carácter ‘8’, siguiendo el mismo protocolo.

El sistema permanecerá en espera hasta recibir comunicaciones y actuar en función de las recibidas. Las posibilidades son:

- Recibir comunicación desde la placa Arduino. Normalmente porque se ha pulsado un botón que será el responsable de iniciar el reconocimiento de huella, o alguna de las comunicaciones consecuencia de ello
- Comunicaciones provenientes del servidor con el objetivo de gestionar el lector de huellas.

En la placa Arduino se encuentra el software responsable de gestionar las comunicaciones con el lector de huellas R308, haciendo uso de la librería de Adafruit («Adafruit Optical Fingerprint Sensor» 2018), y siguiendo los protocolos establecidos. En este caso, la función principal es hacer de puente entre Raspberry y el lector de huellas principalmente. El procedimiento de captura de huella por parte de lector se inicia después de que uno de los pulsadores sea utilizado.

Se dispone de un segundo pulsador que, al ser utilizado, lanza un mensaje de alerta al servidor.

Además de utilizar la librería para la gestión del sensor de huellas, se utilizan las librerías, ya mencionadas, que sirven para gestionar el cifrado de las comunicaciones mediante XXTEA y para hacer uso del display OLED utilizado.

Complementando el programa principal, para poder gestionar las huellas almacenadas desde el servidor, se ha desarrollado un pequeño programa independiente, al que el servidor se puede conectar en el puerto 4466 y que permite eliminar huellas almacenadas en el lector y dar de alta nuevas huellas.

5.5.7 Capturador con control de parpadeo

Similar al capturador con detección de movimiento. Salvo que, en este caso, se le añade la posibilidad de detectar el parpadeo y realizar un conteo del mismo. Se ha

establecido en cinco el número de parpadeos necesarios para que el sistema realice la detección facial y envíe al servidor el rostro. Para facilitar una correcta posición se realiza una señal de aviso cuando se detecta el quinto parpadeo y se espera un segundo antes de realizar la captura.

Se hace uso de la librería *dlib* («dlib C++ Library» 2018) para obtener el landmarks del rostro y con ellos localizar los parpados, resultados que se aprecian en la Figura 34. Se realiza un control de estos puntos detectados, en cada fotograma, para comprobar que la distancia entre ellos es suficiente, en caso contrario se considera que se ha realizado un parpadeo.

Figura 34 Landmarks y contornos obtenidos con dlib

Cuando se alcanza el quinto parpadeo se continúa con la captura y el envío del rostro.

5.5.8 Servidor

La función de este script es permitir al servidor central conectarse con el capturador, que en este caso actúa como servidor, para recibir de él las imágenes que se van capturando y que hacen las funciones de fotogramas de una imagen de video.

El capturador permanecerá a la espera, cuando recibe una conexión, en el puerto 4444, empieza a capturar y enviar las imágenes sin procesar, salvo su conversión a escala de grises y la serialización de las mismas.

Las comunicaciones serán cifradas, aunque existe una versión que no las cifrará, diseñada para las pruebas.

En los dispositivos en los que se habilite esta opción será necesario realizar un control del acceso concurrente a la cámara por diferentes programas, ya que el intento de acceso a la cámara mientras otro proceso la mantiene en uso generaría un error. El control concurrente al dispositivo se realiza con un programa diseñado para tal efecto, **bloqueos.py**, que utiliza un archivo de texto vacío, que simula ser la cámara, y que mantiene en espera el nuevo proceso que solicita la cámara hasta que el anterior la deja libre.

5.6 Software en actuadores

5.6.1 Actuador

El script actuador es responsable de trasladar las directrices marcadas por el servidor tras una identificación a los elementos finales del sistema, haciendo uso de una placa Arduino intermedia. El mensaje recibido indica el canal del actuador que se debe modificar y el nuevo estado. De esta manera el actuador se comunica con la placa Arduino, responsable de las comunicaciones directas con la electrónica, para que cambie el estado del canal correspondiente. El resultado dependerá de qué se disponga más allá del sistema diseñado, pues el actuador se limita a cambiar el valor de las salidas determinadas de Arduino siguiendo los protocolos establecidos.

5.6.2 Actuador cifrado

Similar al anterior, pero incorpora un cifrado de las comunicaciones, haciendo uso de OpenSSL para las comunicaciones con el servidor y de XXTEA en las comunicaciones con las placas Arduino.

5.6.3 Arduino

Diseñado para trabajar con la placa Mega de Arduino, es el responsable de gestionar el valor lógico de los pines de salida de la placa en función de los mensajes enviados por el servidor a través del cliente actuador. El protocolo establecido dictamina que se enviarán los caracteres ‘ON’ u ‘OF’ para indicar si hay que cambiar la salida a nivel alto o bajo, respectivamente, seguidos de los caracteres que representan el número de canal sobre el que hay que actuar.

6 PRUEBAS Y RESULTADOS

Para las pruebas realizadas se ha utilizado un equipo ACER Aspire E1-572, que cuenta con un procesador Intel i5-4200U a 1.6GHz equipado con 16GB de RAM.

6.1 Prototipos

Para implantar en un entorno real el sistema serán necesarios multitud de dispositivos capturadores y actuadores, además, en función de las necesidades específicas pueden ser necesarias diferentes versiones de uno u otro.

Para que el sistema pueda ser más flexible se han desarrollado diferentes versiones de capturadores, que permitirán a los futuros clientes poder adaptar el sistema a las necesidades reales.

Para realizar las pruebas del sistema se han implementado algunos prototipos de los capturadores diseñados:

- **Prototipo A. Capturador simple**. Compuesto por una Raspberry Pi 3B con cámara. Este dispositivo está diseñado para ejecutar las versiones del software que no requieran dispositivos adicionales. Sirve como base para el resto de dispositivos diseñados.
- **Prototipo B. Capturador con lector de huellas**. El sistema está compuesto por una Raspberry Pi, con cámara, y un dispositivo conectado a ella mediante un cable USB, que incluye una placa Arduino, una pantalla OLED de 0,96 pulgadas, un lector de huellas y dos pulsadores que permiten iniciar la lectura de huellas o lanzar una alerta al servidor respectivamente. En la Figura 35 se puede ver un esquema simplificado de las conexiones necesarias para este capturador. En la Figura 36 y en la Figura 37 se muestra el dispositivo que se conectaría a la Raspberry Pi y que ha sido utilizado en las pruebas.

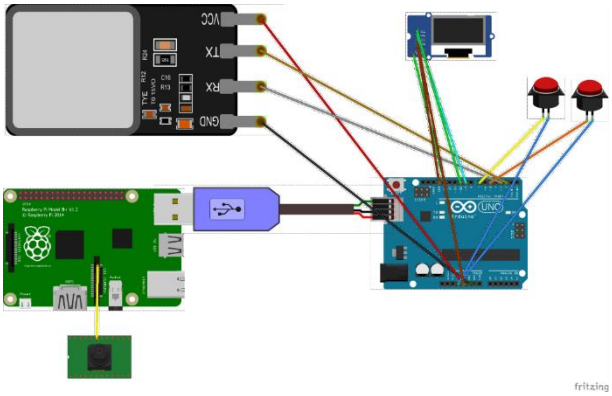

Figura 35 Conexiones del capturador con lector de huellas

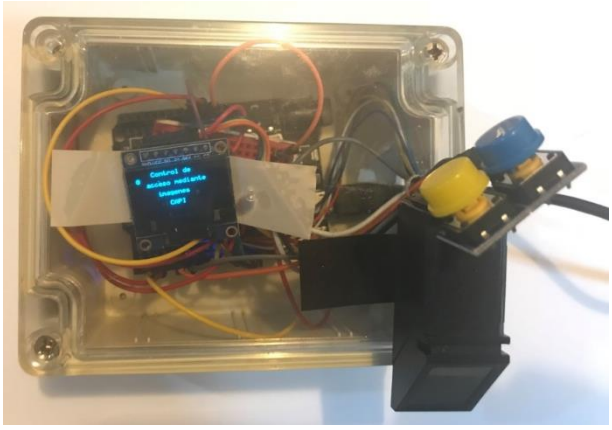

Figura 36 Prototipo creado para un capturador con lector de huellas

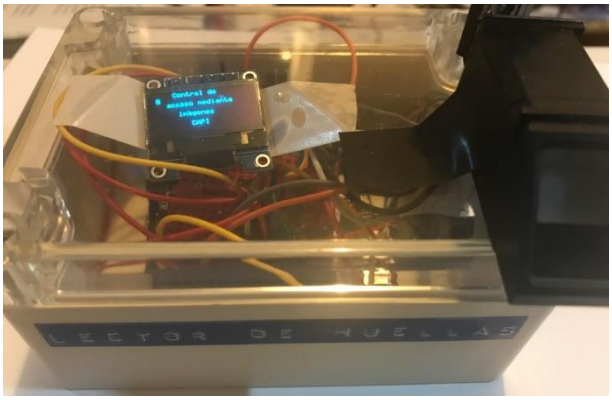

Figura 37 Prototipo creado para un capturador con lector de huellas (II)

- **Prototipo C. Capturador con teclado numérico**. Se hace uso de una placa Arduino Nano, en este caso la potencia necesaria es mínima y es suficiente con una de estas placas. Se utilizará un teclado matricial y un display de siete segmentos con cuatro dígitos para mostrar los números tecleados. Este dispositivo se conectaría, de nuevo, a una Raspberry Pi con cámara.
- **Prototipo D. Capturador con sensor de movimiento**. Una Raspberry Pi 3 gestiona el capturador, a ella se conecta el dispositivo que se muestra en la Figura 38 y en la Figura 39, compuesto por un Arduino UNO, una pantalla LCD y un sensor de movimiento.

Figura 38 Prototipo creado para un capturador con sensor de movimiento

Figura 39 Prototipo creado para un capturador con sensor de movimiento (II)

- **Prototipo E. Capturador con sensor de distancia.** Similar al anterior, salvo que en este caso el sensor utilizado es un sensor de distancia por ultrasonidos y la información se muestra mediante una pantalla OLED de 0.96 pulgadas. Se puede apreciar un prototipo del mismo en la Figura 40 y en la Figura 41.

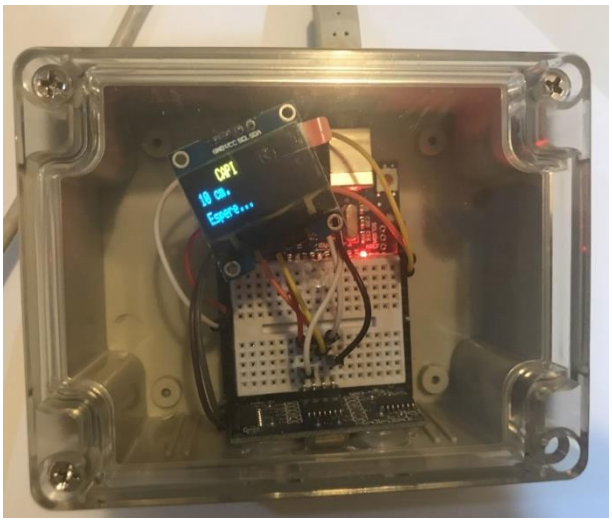

Figura 40 Prototipo creado para un capturador con sensor de distancia

Figura 41 Prototipo creado para un capturador con sensor de distancia(II)

- **Prototipo F. Capturador estándar instalado en una Raspberry Pi Zero W**. Similar al prototipo A pero haciendo uso del modelo más reducido de Raspberry Pi. Es conveniente que, para no calentar en exceso la Raspberry Pi, el tiempo en el que sistema se encuentre realizando capturas sea mínimo, dada la menor potencia de este dispositivo. Todos los prototipos diseñados podrían funcionar sustituyendo como elemento base el prototipo A por este de menor tamaño, con la consecuente pérdida de prestaciones.
- **Prototipo G. Capturador completo con contador de parpadeo**. En este caso se han montado tanto la Raspberry Pi como la placa Arduino en un mismo dispositivo. Para reducir el tamaño se ha utilizado una placa Arduino Nano, en este caso es responsable de gestionar una pantalla OLED. La versión de Raspberry Pi es la 3B, no es recomendable usar la Zero para este propósito al requerir, el contador de parpadeo, un procesamiento considerable.

Además, se ha diseñado e implementado un prototipo de **Actuador, prototipo H**. El sistema ha sido diseñado para poder ser implantado directamente, por lo que los actuadores son capaces de gestionar directamente, mediante diferentes componentes electrónicos, como relés, todas las actuaciones. El prototipo diseñado requiere de dos entradas, una de 5 V y otra de 12 V, responsables de alimentar un relé con cuatro canales y los dispositivos controlados por este, respectivamente. Para actuadores en fase de implementación estos voltajes dependerán exclusivamente de los dispositivos que se desee controlar y la alimentación que requiera el relé que los controlara.

Se han incluido en el prototipo Actuador una serie de leds que representan el estado de otros tantos canales del mismo. Cada uno de estos canales podría controlar un elemento electrónico directamente, si su voltaje es de 5 V, o con un voltaje superior, mediante la utilización de relés.

Todos estos prototipos son totalmente funcionales y representarían ejemplos de capturadores y actuadores que podrían formar parte del sistema una vez implementado, con ligeras modificaciones. No obstante, estos dispositivos son enormemente flexibles y sus funciones se podrían modificar fácilmente.

6.2 Pruebas de seguridad

Cifrar las comunicaciones es vital para la seguridad de los sistemas. Cualquier sistema conectado a una red está expuesto a ataques y, en cierta medida, siempre será vulnerable. En este trabajo se ha incluido como uno de los objetivos principales en su desarrollo dotar al sistema de seguridad en las comunicaciones cifrando las mismas. El sistema podría seguir siendo vulnerable, pero, al menos, se habrá dificultado el trabajo del atacante. En esta sección se va a demostrar que si las comunicaciones se realizan sin cifrar el sistema estará en peligro y podría ser atacado y quebrantado sin demasiada dificultad.

Para esta prueba se ha dejado el sistema desprovisto de software para cifrado, es decir, se ha deshabilitado el uso de OpenSSL, librería responsable del cifrado en las comunicaciones, de modo que las comunicaciones entre clientes y servidor se realizan sin cifrar.

6.2.1 Captura de paquetes no cifrados

El primer paso para este ataque es utilizar un software capaz de capturar paquetes de una red a la que se está conectado. Lo que se conoce como *sniffer*. Uno de los más populares es *Wireshark*, nacido como *Ethereal*. Ha sido seleccionado por su popularidad, aunque existen otras alternativas también válidas.

Cuando las comunicaciones entre cliente y servidor son capturadas por Wireshark se pueden utilizar algunas funciones de este software para filtrar el tráfico capturado y poder centrar la atención en una comunicación determinada.

Un filtro posible sería:

((ip.addr==10.0.64.83) and (tcp.port==4444))

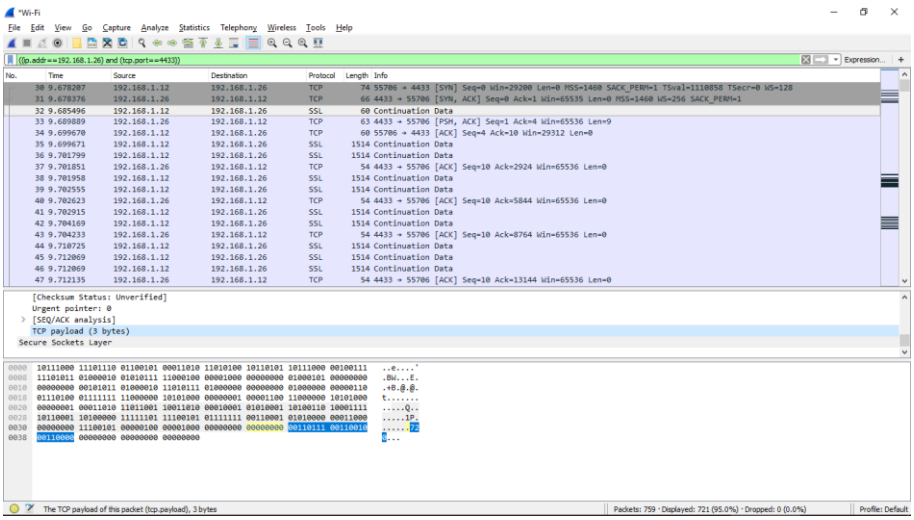

Figura 42 Tamaño de la imagen (720) no cifrado

Para la IP del cliente 10.0.64.83 y el puerto 4444 para el directo (se puede cambiar si es otra comunicación la que se pretende capturar). Cuando se hayan capturado paquetes que se correspondan con el filtro se puede utilizar la opción *follow-tcp stream*, al pulsar con el botón derecho sobre uno de estos paquetes, para disponer de la comunicación completa.

En la Figura 42 se muestra un ejemplo de captura en el que se ha enviado el código 720 desde un capturador al servidor sin cifrar y ha sido capturado.

Cuando se tenga todo, se dispone de la opción de exportar como *Hex Dump*. El archivo resultante, con este formato, será el utilizado en el siguiente paso del procedimiento.

Es posible que las capturas no siempre permitan obtener la información que se persigue. En este caso concreto el atacante debería conocer la dirección IP del cliente, la del servidor, el puerto y, además, saber que se está buscando. Esta información podría no estar disponible, pero es solo cuestión de tiempo que un atacante sea capaz de conocer las direcciones y el puerto solo siguiendo la información que circula por la red, sin cifrar. Si, además, se conoce la existencia de un sistema de reconocimiento facial la posibilidad de que imágenes faciales circulen por la red es alta. El sistema desarrollado utiliza el formato de imagen básico de uno de los sistemas de visión por computador más utilizados, por lo que acabar realizando pruebas en buscan de una imagen con formato Mat, de OpenCV, es cuestión de tiempo.

Se ha desarrollado un pequeño software con C++ capaz de convertir el archivo, con formato HexDump, capturado en el apartado anterior, en una imagen jpg, e, incluso, ha sido posible capturar video emitido sin cifrar, aunque en este caso el resultado no ha sido tan satisfactorio.

Aunque el desarrollo de este software no forma parte de los objetivos del proyecto se ha considerado conveniente realizar un estudio minucioso sobre las posibles consecuencias de no dotar al sistema de seguridad.

El funcionamiento de la aplicación es sencillo: Se supone un tamaño de 1500 bytes por paquete, unidad máxima de transferencia (Maximum Transmission Unit, MTU), con tamaño máximo de segmento (*Maximum Segment Size, MSS*) de 1460. Para una imagen de tamaño, por ejemplo, 640x480 pixeles se tendrán 307200 bytes de información, 300KB, si la imagen se encuentra en escala de grises. Dado que el formato de archivo es en hexadecimal se tiene un byte por cada dos dígitos hexadecimal. Siguiendo el formato del archivo es sencillo diseñar un bucle que lo recorra, teniendo en cuenta que cada línea del archivo tiene 16 bytes de información y que cada 92 líneas se tendrá una finalización de paquete (serían 1472 bytes, a los que no alcanza un único paquete). Ajustando estos parámetros cuando se recorre el archivo y volcando los datos leídos en un archivo con formato Mat se obtiene como resultado la imagen enviada.

Realizando unas pequeñas modificaciones sobre el archivo se puede capturar video (realmente son una secuencia de imágenes). En este caso, y con el limitado software de conversión desarrollado, es fácil que tras unos pocos fotogramas la imagen no sea del todo correcta, pero es suficiente para comprobar que sencillo puede llegar a ser atacar con éxito el sistema.

6.2.2 Pruebas de seguridad biométrica

Uno de los problemas a los que se enfrenta un sistema de reconocimiento biométrico que use el rostro, es a la posibilidad de utilizar imágenes impresas que permitan suplantar a un usuario registrado en el sistema. Existen diferentes alternativas para evitar esta posibilidad, como el uso de cámaras de profundidad o 3d, que puedan comprobar que el rostro es real. En este trabajo se han incluido tres opciones que podrían evitar la suplantación mediante una imagen impresa, la utilización de un PIN o código personal, el uso de un lector de huellas o el conteo de parpadeos.

Para los actuadores donde no se haya implementado ninguna de estas técnicas alternativas de seguridad se han realizado unas pruebas con imágenes impresas de usuarios registrados. También se han probado imágenes de usuario no registrados para comparar los resultados. En la Figura 46 se muestran algunas de las imágenes utilizadas en estas pruebas de suplantación.

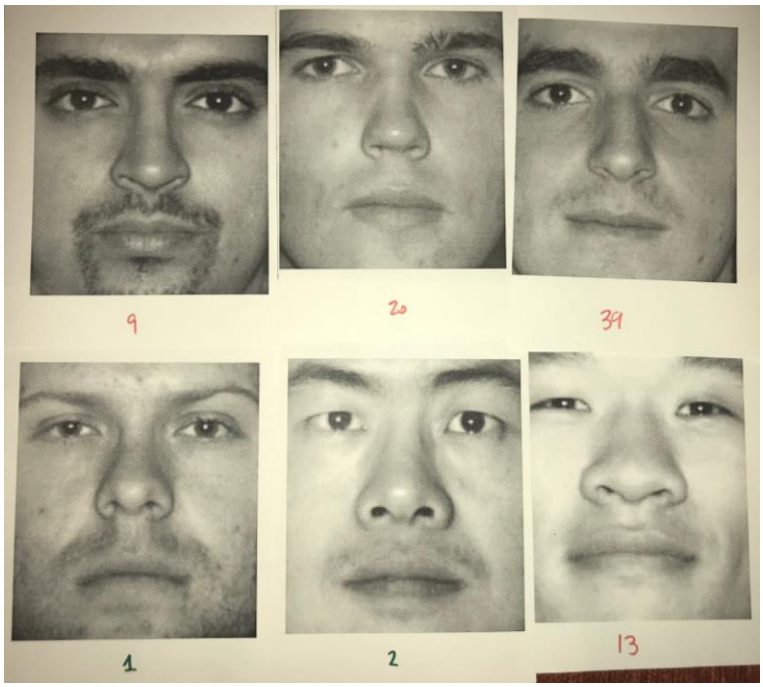

Figura 43 Imágenes impresas para pruebas de seguridad

Se han realizado pruebas con los diferentes algoritmos implementados y los resultados obtenidos han sido:

- LBPH. De las nueve imágenes de usuarios registrados han sido reconocidas ocho, mientras que de las imágenes de usuarios no registrados dos de ellas han dado falsos positivos. El valor de umbral utilizado ha sido 50. El porcentaje de error es elevado, por lo que se recomendaría utilizar un umbral más bajo.
- Eigenfaces. De las nueve del primer grupo, ocho han sido reconocidas positivamente. Del segundo grupo cuatro han sido reconocidas siendo falsos positivos. El porcentaje de error es alto debido al valor de umbral utilizado, 10.000. La imagen registrada no reconocida es la del tercer usuario, al igual que ocurrió con LBPH, es una imagen algo más clara que el resto y que impresa parece sobreexpuesta, lo que parece ser un problema serio para el reconocimiento.
- Fisherfaces. Con un valor de umbral de 2.500, solo cuatro usuarios han sido reconocidos del primer grupo y solo se ha producido un falso positivo. Este algoritmo parece más sensible a las pérdidas de información que se producen al imprimir la imagen.

Capturas de los resultados obtenidos se incluyen en el Anexo.

Las imágenes han sido impresas con una calidad media en papel estándar, y estas pruebas se han realizado directamente con la cámara local del servidor. Estos resultados

dejan claro que es relativamente fácil utilizar una imagen impresa para engañar al sistema por lo que es recomendable utilizar métodos que proporcionen seguridad extra, como el conteo de parpadeos incluido en el sistema.

6.3 Pruebas de detección facial

Durante las pruebas realizadas de detección facial, con el servidor, se han realizado un total de 14.907 pruebas de detección con tamaños de búsqueda de 100x100 hasta 300x300. El tiempo medio empleado ha sido de **0,01841 s**.

Para un tamaño de entre 50x50 y 400x400 se han realizado 43.337 pruebas y el tiempo medio para la detección facial ha sido de **0,0507 s**.

La diferencia del tiempo en detección es notable al aumentar el rango de valores para los tamaños de las cajas, aunque en las pruebas realizadas, en ningún caso, ha supuesto un problema en el rendimiento. En todas las pruebas se han utilizado las imágenes que proporciona la cámara directamente, con tamaño 1280x720.

Aunque no se han contabilizado, apenas se han producido falsos positivos y la tasa de error es realmente ínfima.

6.4 Pruebas de reconocimiento

Para verificar el correcto funcionamiento del sistema se han realizado diferentes pruebas de reconocimiento facial.

6.4.1 Pruebas de tiempo

Se han realizado 8.217 pruebas y se ha tenido un tiempo medio de **0,0399 segundos** utilizando el algoritmo Eigenfaces.

Con el algoritmo LBPH se han realizado 13.927 pruebas y el tiempo medio es de **0,0411 segundos**.

Por último, utilizando el algoritmo Fisherfaces se han realizado 6.504 pruebas y el tiempo medio para el reconocimiento facial ha sido de **0,00165 segundos**.

El resumen de los datos obtenidos, con el número total de pruebas (p) y el tiempo medio invertido en segundos (t), se puede ver en la Tabla 2 y su representación en la Figura 44.

	LBPH			Eigenfaces			Fisherfaces		
	Total	505 img	161 img	Total	505 img	161 img	Total	505 img	161 img
p	13.927	4.969	8.958	8.217	4.969	3.248	6.504	4.969	1.535
t	0,0411	0,0612	0,030	0,0399	0,0556	0,0159	0,0016	0,018	0,012

Tabla 2 Tiempos de reconocimiento

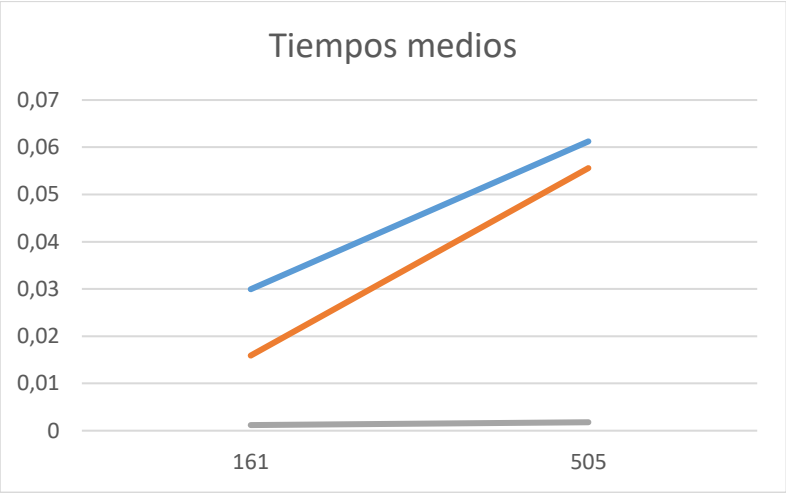

Figura 44 Tiempos medios en reconocimiento

Los tiempos de reconocimiento son mejores con el algoritmo Fisherfaces. Aunque ninguno de los tres algoritmos de reconocimiento ha presentado problemas en cuanto al tiempo necesario para su procesamiento en las pruebas realizadas, el número de imágenes a procesar puede ser un factor a considerar en caso de sistemas con un elevado número de imágenes considerando los resultados obtenidos.

6.4.2 Pruebas con imágenes en entorno controlado

Una vez el sistema ha sido configurado, se han realizado pruebas de reconocimiento haciendo uso de imágenes obtenidas en un entorno controlado como es el de la base de datos Yale («Cropped the Extended Yale Face» 2018). Esta base de datos contiene imágenes faciales de múltiples individuos, y, para cada uno de ellos, contiene imágenes con variaciones en la potencia lumínica y en la posición de la fuente de iluminación. Los rostros son neutros y las imágenes ya se encuentran convenientemente recortadas para el reconocimiento facial.

Para este estudio se han utilizado imágenes aleatorias de los ocho primeros individuos de la base de datos para entrenar el sistema, con el único condicionante de que estas imágenes no tuvieran una variación en cuanto a la potencia de luz ideal superior al 35%, ni una variación del ángulo del foco de iluminación superior a 25º. Por tanto, la selección ha sido aleatoria, pero acotando el grupo a las imágenes obtenidas en mejores condiciones. El resto de imágenes se ha dividido entre las que pertenecían a este grupo, pero no han sido seleccionadas para el entrenamiento, y las que no pertenecían a este grupo por no cumplir las condiciones, disponiendo de 57 y 345 respectivamente. (Para distinguir los grupos en adelante se denominarán “buenas” y “malas”)

Para el entrenamiento se ha incluido un usuario nuevo con imágenes obtenidas con el sistema, dejando una imagen para el testeo.

Para estas pruebas se ha utilizado ***Enviapruebas.py*** para el envío y se ha configurado el servidor para realizar el reconocimiento con los tres protocolos.

Las tablas con los resultados se incluyen en el ANEXO III. Resultados de las pruebas.

- **LBPH**. De un total de 57 imágenes de calidad se ha obtenido identificación positiva correcta en 56. Una no ha sido identificada. El valor promedio de la confianza ha sido: 27,69 y el umbral preestablecido ha sido 50.

  Para el grupo “malas”, de las 345, 78 han sido reconocidas con una confianza media de 37,89.
- **Eigenfaces**. Para el grupo de “buenas” se ha obtenido una confianza media de 4897,19, aunque tres imágenes fueron reconocidas de forma errónea teniendo, por tanto, un **5,17%** de error.

  Para el caso de las imágenes “malas” se han obtenido 297 reconocimientos de 345, aunque con un porcentaje de error inasumible, **30,72%** debido, posiblemente, al alto valor utilizado como umbral (10.000)
- **Fisherfaces**. Por último, se han realizado las mismas pruebas para Fisherfaces con un umbral de 1300. Con el primer grupo se obtiene un reconocimiento correcto para 54 imágenes, cuatro quedan sin reconocer.

  Para el segundo grupo solo 49 son reconocidas, aunque no hay faltos positivos.

  El bajo valor obtenido en cuanto a porcentaje de reconocimiento invita a pensar en un valor umbral excesivamente bajo. Se realizan nuevas pruebas alterando este valor. En primer lugar, se utiliza un valor de 5000. Con este valor, todas las imágenes del primer grupo se reconocen bien. Mientras que para el segundo grupo se reconocen 281 de forma correcta y 64 incorrectamente, un **18,55%** de error, un valor excesivo. No parece ser 5000 un valor adecuado.

  Por último, se establece en 2500 y se obtiene un reconocimiento perfecto de todas las imágenes del primer grupo, y para el segundo grupo se reconocen positivamente 182, se reconocen erróneamente y 149 se quedan sin reconocer. Un error del **4,06%**.

6.5 Pruebas de entrenamiento y carga

Se han realizado pruebas con las diferentes técnicas y variando el número de usuarios y de imágenes totales en el entrenamiento. Los resultados obtenidos son:

- Con **161 imágenes,** para nueve usuarios, se han realizado 105 pruebas de entrenamiento y guardado del mismo en un archivo YML. Después se han realizado 280 cargas utilizando los archivos YML.
- En segundo lugar, se ha utilizado una base con **400 imágenes** y 40 usuarios. 26 veces se ha entrenado y guardado y en 56 ocasiones se ha cargado desde archivo.
- Las últimas pruebas se han realizado con **849 imágenes** y 47 usuarios. Se han realizado 27 pruebas de entrenamiento y 59 cargas desde archivo.

Un resumen de los resultados obtenidos, en segundos, tras concluir estas pruebas se incluyen en la Tabla 3.

	LBPH			Eigenfaces			Fisherfaces		
	Entreno	Guardar	Carga	Entreno	Guardar	Carga	Entreno	Guardar	Carga
1	0,85	1,03	0,76	3,44	5,21	3,74	2,06	0,28	0,26
2	2	2,94	2,4	29,43	15,87	12,09	17,58	1,58	1,23
3	3,89	5,33	5,17	133,48	29,5	26,01	95,93	1,64	1,46

Tabla 3 Pruebas de entrenamiento y carga

6.6 Pruebas en Raspberry

6.6.1 Raspberry Pi 3B

Se han realizado pruebas de temperatura con una Raspberry Pi 3B mientras se realizada detección facial. Con una cámara v1.0.

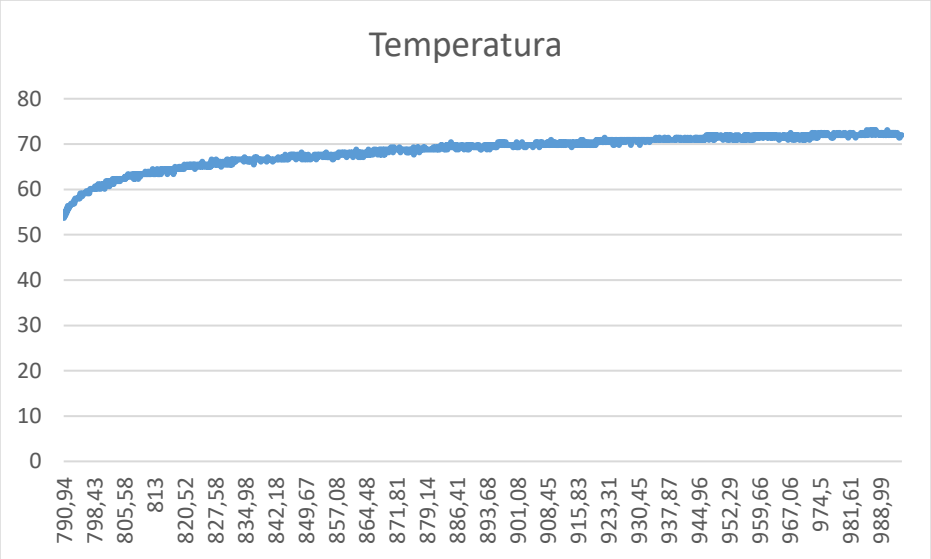

Tabla 4 Temperatura de Raspberry en detección

En la Tabla 4 se muestran los datos obtenidos. El tiempo está en segundos. Se puede apreciar cómo tan solo después de haber transcurrido unos 200 segundos la Raspberry ha pasado de una temperatura cerca de 52ºC a pasar de 73ºC.

Por otro lado, se han realizado pruebas para el tiempo de detección, variando los tamaños mínimo y máximo de los cuadros en los que buscar, obteniendo los resultados que se muestran en la Tabla 5.

	Sin límite		100x100 – 300x300		50x50 – 400x400	
	Con rostro	Sin rostro	Con rostro	Sin rostro	Con rostro	Sin rostro
Imágenes	452	456	1182	952	490	1386
Tiempo (s)	0,084	0,065	0,018	0,009	0,051	0,031

Tabla 5 Tiempos de detección en Raspberry Pi 640x480

El tamaño de las imágenes en estas pruebas es de 640x480, inferior al utilizado en las pruebas del servidor.

Probamos ahora con el tamaño máximo de imagen que soporta la cámara v1 de la Raspberry, 2592x1944, y obtenemos los resultados de la

	Sin límite		500x500 – 1000x1000	
	Con rostro	Sin rostro	Con rostro	Sin rostro
Imágenes	25	88	132	140
Tiempo (s)	1,62	0,829	0,03	0,022

Tabla 6 Tiempos de detección en Raspberry Pi 2592x1944

6.6.2 Raspberry Pi Zero W

Para un tamaño de imagen de 2592x1944 se han obtenido los resultados que aparecen en la Tabla 7.

	Sin límite		500x500 – 1000x1000	
	Con rostro	Sin rostro	Con rostro	Sin rostro
Imágenes	5	16	132	140
Tiempo (s)	16	6	0,052	0,023

Tabla 7 Tiempos de detección en Raspberry Pi Zero 2592x1944

Con un tamaño de imagen de 3280x2464 se obtienen unas medias de 0,034 en imágenes sin rostros a detectar y 0,063 en imágenes con rostros.

Por último, se ha reducido el tamaño de la imagen a 640x480 y se consiguen los resultados que aparecen en la Tabla 8.

	Sin límite		100x100 – 300x300		50x50 – 400x400	
	Con rostro	Sin rostro	Con rostro	Sin rostro	Con rostro	Sin rostro
Imágenes	80	1200	644	798	80	621
Tiempo (s)	1,084	0,394	0,089	0,058	0,441	0,166

Tabla 8 Tiempos de detección en Raspberry Pi Zero 640x480

6.7 Pruebas de vídeo

Para probar la detección y reconocimiento en video se han utilizado las imágenes y los videos incluidos en la base de datos Seas («SEAS Face Recognition Database» 2016). Se han incluido los tres usuarios con imágenes en la base de datos del sistema, además de los nueve utilizados en las pruebas de reconocimiento en entorno controlado. Se han procesado las imágenes originales para mantener el mismo formato y tamaño que las imágenes de la base de datos de Yale.

Una vez los usuarios han sido incluidos en el sistema se procede reproduciendo los videos incluidos en la base de datos para detectar y reconocer los rostros que en ellos aparecen. El sistema tiene una limitación para el tamaño de las cajas en detección configurable, para esta prueba se ha establecido entre el 20 y el 80% del alto de la imagen. Se han establecido tres frames de seguimiento, de modo que realiza una detección cada cuatro fotogramas de video.

En el video para test de la base de datos, *all_slide*, en el que los usuarios se van desplazando lateralmente, apareciendo los cinco de la base de datos, solo tres registrados se obtienen, para 163 imágenes procesadas, con el algoritmo LBPH:

- 0 falsos positivos.
- 42 veces reconocido el usuario dos (primero en aparecer)
- 24 veces el usuario uno (tercero en aparecer)
- 0 veces el usuario tres (quinto en aparecer).

Figura 45 Reconocimiento en video

El número de veces que el rostro de un usuario es procesado para su reconocimiento depende del número de imágenes en que aparece y de si aparece con otro usuario, el sistema solo hace un reconocimiento por imagen. Los resultados parecen

buenos, excepto para el usuario tres, que no es capaz de reconocerlo en ninguna de las imágenes. Para encontrar el problema vamos a realizar pruebas con los videos individuales de cada usuario que hay en la base de datos. Hay cuatro videos por usuario:

1. El usuario parece mantener una conversación frente a la cámara, no permanece inmóvil.
2. El usuario realiza gestos exagerados con la cara y realiza movimientos continuos del rostro.
3. El usuario comienza el video con un rostro neutro que ven cambiando hasta un rostro histriónico.
4. El usuario permanece prácticamente inmóvil, con rostro neutro y mirando a la cámara.

Se han obtenido los resultados de la Tabla 9 para valores de umbral 50 en LBPH, 7500 en Eigenfaces y 2500 en Fisherfaces.

Usuarios	**1**				**2**				**3**			
Videos	1	2	3	4	1	2	3	4	1	2	3	4
Imágenes	30	30	30	31	31	31	31	31	30	30	31	31
LBPH	**13**	**0**	**16**	**25**	**12**	**5**	**19**	**31**	**20**	**0**	**0**	**0**
Eigenfaces	**9**	**0**	**2**	**25**	**15**	**0**	**0**	**31**	**17**	**21**	**14**	**29**
FAR	2	0	0	0	0	0	0	0	3	0	0	0
Fisherfaces	**10**	**3**	**15**	**25**	**31**	**15**	**19**	**31**	**28**	**30**	**31**	**31**
FAR	19	11	0	0	0	16	12	0	1	0	0	0

Tabla 9 Reconocimientos en video

Hay que resaltar que el usuario tres aparece con gafas en todos los videos, excepto en el primero, único donde es reconocido utilizando el algoritmo LBPH. El usuario dos también aparece con gafas en los videos, aunque en este caso si se han obtenido buenos resultados.

Si el rostro del video es muy pequeño, tamaño inferior al 20% del alto del fotograma, no será detectado. Pero reducir este tamaño mínimo para detección provoca que el sistema se ralentice en exceso. Ya, este rango, 20-80%, genera problemas en cuando a la reproducción de videos. Estos valores se deberían adaptar en función de las necesidades y de que se va a reproducir.

6.8 Conclusiones de las pruebas y recomendaciones

Considerando los resultados obtenidos en las pruebas presentadas se pueden alcanzar las siguientes conclusiones:

OpenCV permite, mediante la instrucción ***detectmultiscale*** realizar la detección facial, entro otras funciones. Este procedimiento permite ser configurado utilizando varios parámetros, y en estas pruebas se ha modificado y adaptado sus valores:

- *ScaleFactor*. Permite determinar el factor de incremento de escala utilizado en el algoritmo de búsqueda en cascada, es decir, cuanto se aumenta el cuadro de búsqueda en cada paso. Un valor excesivamente alto podría dejar caras sin detectar, mientras que uno muy bajo puede afectar al rendimiento del sistema negativamente. El valor, siguiendo las recomendaciones de la mayoría de documentos sobre OpenCV consultados, debe situarse entre 1,05 y 1,4. Para este proyecto se ha fijado en 1,1 para el servidor y 1,3 en los capturadores, y se obtienen buenos resultados; aunque es un valor que se podría modificar en caso de ser necesario.
- *MinNeigbours*. Especifica cuantos vecinos debe tener cada cuadro para ser considerado positivo en la detección. El algoritmo de detección realiza buscadas recorriendo la imagen con cuadros de diferente tamaño y que generan varios positivos sobre un mismo rostro, este valor especifica cuantos cuadros positivos deben coincidir para que realmente se considere un positivo en detección. Cuando se termina la detección el algoritmo agrupa los cuadros resultando uno solo. El valor recomendado en la bibliografía utilizada varía entre tres y seis. Se han utilizado tres en el servidor y cinco en los capturadores. En las pruebas se ha demostrado que son valores adecuados.
- *minSize* y *maxSize*. Determinan los tamaños mínimo y máximo para los cuadros de búsqueda. Rostros con tamaño inferior al mínimo o mayor al máximo no serán detectados, por otro lado, un rango de valores excesivamente alto puede provocar excesivos incrementos en el tiempo de detección de rostros. En cualquier caso, el valor adecuado para estos valores depende de factores como la distancia a la cámara a la que se puede encontrar el sujeto cuyo rostro se desea capturar, en definitiva, el tamaño del rostro. Por eso se ha dejado abierta la posibilidad de que sea el administrador el que determine el valor de estos parámetros.

Los resultados obtenidos en cuanto al tiempo para el reconocimiento dejan claro que el número de imágenes en la base de datos puede ser un factor determinante y debe ser tenido en cuenta. Los algoritmos de reconocimiento utilizados parecen no tener problemas de rendimiento en entornos reducidos, con cientos de usuarios.

El reconocimiento facial utilizando imágenes obtenidas en un entorno controlado evidencia la necesidad de controlar el valor umbral a la hora de implantar el sistema. Tras varias pruebas se han conseguido unos primeros valores, 50 para LBPH, 7500 en Eigenfaces y 2500 para Fisherfaces. Aunque su idoneidad dependerá de las necesidades puntuales y del entorno donde el sistema sea implantado. Por lo que la recomendación es que sean adaptados a la hora de implantar el sistema.

Queda fuera de los objetivos de este proyecto evaluar en profundidad los algoritmos de reconocimiento disponibles, no obstante, con los resultados obtenidos se pueden obtener unas primeras conclusiones: LBPH presenta las mejores prestaciones en cuanto a tiempos de entrenamiento y guardado del mismo. Fisherfaces presenta los mejores tiempos a la hora de cargar desde un archivo.

Las pruebas realizadas parecen indicar que el algoritmo LBPH sufre bastante cuando el usuario utiliza gafas. Sin entrar evaluar cómo afecta el uso de gafas al reconocimiento facial, y, en principio, la recomendación sería no usarlas cuando se realizan pruebas de reconocimiento facial.

Considerando todos los datos obtenidos, relativos a las pruebas de reconocimiento facial, se recomienda dar prioridad en el uso a los algoritmos LBPH y Fisherfaces sobre Eigenfaces. En un sistema muy dinámico con frecuentes actualizaciones LBPH ofrece la posibilidad de actualizar el sistema sin volver a cargar. Mientras que, por el contrario, para un sistema estable, con pocas actualizaciones y que permita la carga desde un archivo, Fisherfaces parece la mejor opción.

Los resultados obtenidos confirman que es conveniente que el usuario permanezca inmóvil, con un rostro neutro y sin moverse frente a la cámara para mejorar la eficacia en el reconocimiento. Esta premisa se cumple con los tres algoritmos utilizados; se puede apreciar, con claridad, en la mejora en los resultados en la cuarta versión de los videos por usuario en la base de datos Seas. En cualquier caso, son determinante para los resultados las imágenes que se utilicen en el entrenamiento.

En cuanto al uso de las Raspberry Pi en los capturadores, los resultados presentados demuestran que su uso es posible para el caso de la versión 3B. No obstante, los resultados con la versión Zero W no son buenos en cuanto al tiempo de respuesta y, sin llegar a descartar su uso, si se recomienda utilizar solo en casos especiales y asumiendo los posibles retrasos en cuanto al tiempo en detección.

Para obtener resultados satisfactorios es necesario cumplir con unos requisitos en cuanto a la calidad de las imágenes que contienen los rostros. En el apartado correspondiente a la calidad de la imagen, se avanzó sobre este tema y se pueden obtener una serie de condiciones necesarias para un correcto funcionamiento del sistema; la precisión del mismo dependerá, en gran medida, de que estas condiciones se cumplan:

- La fuente de iluminación debe ser frontal para que el rostro permanezca iluminado de forma homogénea.
- Se deben evitar las sombras y cualquier elemento que pueda ocultar parte del rostro.
- Aunque existen posibles mejoras que facilitarían el reconocimiento del rostro con gafas, no han sido incluidas y pueden provocar reflejos que dificulten el reconocimiento, por lo que es conveniente realizarlo sin gafas.
- El usuario debe permanecer inmóvil, salvo cuando se esté realizando con control de vida mediante parpadeo, frente a la cámara. Un ángulo excesivo en el rostro puede provocar que este no se reconocido, por lo que es conveniente que el rostro se encuentre con la menor inclinación posible.
- El usuario debe mantener una pose neutra. Tanto al registrarse como al realizar el reconocimiento
- Aunque las opciones de configuración permiten modificar valores que afectan a la distancia a la que un usuario es detectado y reconocible es recomendable que el reconocimiento se produzca a una distancia que permita obtener la imagen con suficiente resolución. Tanto para el reconocimiento como para dar de alta al usuario. La distancia recomendable varía en función de las condiciones del entorno y las características de la cámara finalmente utilizada. Distancia focal, ancho del espacio o resolución de la cámara son solo algunos parámetros a considerar. En cualquier caso, como ya se avanzó en 2.2.9, la condición general es disponer de unos 80 pixeles para el ancho del rostro como mínimo.
- Es aconsejable realizar actualizaciones de las imágenes con el paso del tiempo, sobre todo si los usuarios permanecen en el sistema durante largos periodos de tiempo, para evitar problemas de reconocimiento relacionados con los cambios físicos.

7 PLIEGO DE CONDICIONES

En este apartado se van a definir una serie de condiciones necesarias que se deben cumplir para poder utilizar el sistema diseñado.

7.1 Condiciones generales

Una vez el sistema se encuentre en explotación es necesario que cumpla las leyes y normas que le sean de aplicación. La más destacada será la nueva legislación acerca de la protección de datos, que será explicada a continuación. Además, sería recomendable que el sistema cumpliera con los estándares que marca la industria, algunos de los cuales también se relacionan en los apartados siguientes.

7.1.1 Legislación

El nuevo Reglamento General de Protección de datos, RGPD o GDPR (*General Data Protection Regulation*), ha entrado en vigor el 24 de mayo de 2018, aunque fue publicado en 2016 («Reglamento 2016/679 del Parlamento Europeo y del Consejo» 2016), es una norma europea que afecta a todos los países de la Unión europea. En ella se definen los datos biométricos como:

> *"datos personales obtenidos a partir de un tratamiento técnico específico, relativos a las características físicas, fisiológicas o conductuales de una persona física que permitan o confirmen la identificación única de dicha persona, como por ejemplo imágenes faciales o datos dactiloscópicos."*

El punto 1 del artículo 9 establece estos como una categoría especial de datos personales y prohíbe su tratamiento, pero en su punto 2 especifica una serie de excepciones entre las que se incluye el consentimiento por parte del usuario.

En la agencia de protección de datos se dan una serie de consideraciones que se deben tener en consideración («Guía videovigilancia AGPD» [sin fecha]):

- Debe existir una relación de proporcionalidad entre la finalidad perseguida y el modo en el que se traten los datos.
- Debe informarse sobre la captación y/o grabación de las imágenes.
- Si el sistema de videovigilancia genera un fichero el responsable deberá notificarlo previamente a la Agencia Española de Protección de Datos, para su inscripción en el Registro General de la misma. Esto ocurrirá siempre que exista algún tipo de grabación. En caso de tratarse de ficheros de titularidad pública deberá procederse primero a su creación mediante disposición de carácter general publicada en el correspondiente diario oficial conforme a lo

establecido en el artículo 20 LOPD para posteriormente proceder a su inscripción.

- El uso de instalaciones de cámaras o videocámaras sólo es admisible cuando no exista un medio menos invasivo.
- Las cámaras y videocámaras instaladas en espacios privados no podrán obtener imágenes de espacios públicos.
- Podrían tomarse imágenes parciales y limitadas de vías públicas cuando resulte imprescindible para la finalidad de vigilancia que se pretende, o resulte imposible evitarlo por razón de la ubicación de aquéllas.

Para la implementación del sistema en un edificio sería necesario, por tanto:

- Dar de alta el fichero en la Agencia de protección de datos.
- Colocar en las zonas sensibles de ser vigiladas distintivos que informen de que podrían ser grabados.
- Disponer de toda la documentación exigida.

7.1.2 Estándares

Se definen cuatro tipos de estándares: interfaces técnicas, formato de intercambio de datos biométricos, perfiles de aplicación y pruebas de rendimiento.

En 1998 nació **BioAPI**, con el apoyo de grandes compañías como COMPAQ o HP. Se trata de un consorcio que persigue definir dos interfaces, uno de programación de aplicaciones y otro de proveedores de servicio.

En mayo de 2000 Microsoft apoyó y utilizó el estándar BAPI (Biometric Programming Application Interface) desarrollado por I/O Software, más tarde Intel también lo apoyaría, convirtiéndose en una alternativa a BioAPI.

Algunos ejemplos de estándares que se deben considerar:

- ANSI X.9.84. Nace en 2001 y define las condiciones de los sistemas biométricos para el sector bancario.
- ANSI / INCITS 358. Estándar creado en 2002, garantiza que los productos y sistemas son interoperables entre sí.
- ISO/IEC JTC1/SC37, de 2003, que se encarga de estandarizar todos los aspectos de la industria del reconocimiento biométrico tales como terminología, métodos de evaluación, interfaces de programación y los tipos de datos permitidos en esta.
- ISO/IEC 30106-3, BioAPI en los lenguajes programación con orientación a objetos.
- ANSI 378: creado en 2004, establece criterios para representar e intercambiar la información de las huellas dactilares a través del uso de minucias.
- ANSI 379: similar al anterior, pero para el iris ocular.

- ISO 19794-2: creado en 2005 por la ISO/IEC con propósitos similares a la norma ANSI 378, respecto a la que guarda mucha similitud.
- ISO 19794-4: también de 2005 para huellas dactilares.
- ISO 19794-6: para imágenes de datos del iris.
- PIV-071006: creado en 2006 por el NIST y el FBI, establece los criterios de calidad de imagen que deben cumplir los lectores de huellas dactilares para poder ser usados en procesos de verificación de identidad en agencias federales.
- ISO/IEC 19785 CBEFF (Common Biometric Exchange File Format) Se trata de un estándar para el formato de los archivos que contienen información biométrica. Establece la cabecera y los campos que debe tener un archivo para el intercambio de esta información entre dispositivos biométricos.
- ANSI INCITS 378-2004, define el formato de imagen para representar huellas usando minucias.
- ISO/IEC 10918, estándar para imágenes en formato jpg.
- ISO/IEC 19794-5, determina si las imágenes tienen valor identificativo para reconocimiento facial. (Vazquez et al. 2012) (Thakare y Thakare 2012) (Ferrara, Franco y Maltoni 2008)

Existen herramientas como Fingerprint BSS o Iris BSS de Neurotechnology que permiten cambiar entre los formatos estándares y los suyos propios.

BEAT (Biometrics Evaluation And Testing) («Biometrics Evaluation and Testing (BEAT)» 2018) es un proyecto financiado con fondos europeos que nace con el objeto de proporcionar criterios y métricas estandarizados para evaluar sistemas biométricos para ámbitos académicos y comerciales.(Marcel 2013). Proporciona criterios estandarizados y métricas que permiten evaluar los sistemas biométricos.

7.2 Condiciones técnicas

Todas las herramientas utilizadas en la implementación final son software libre, pero es necesario evaluar las licencias que utilizan y posibles limitaciones para el entorno en que el sistema sea implantado:

- OpenSSL. Utiliza licencia Apache. No requiere que el software desarrollado utilice la misma licencia.
- OpenCV. Licencia BSD. Similar a la anterior.
- Visual Studio. La licencia Community permite su uso libre, aunque el software desarrollado sea con ánimo de lucro, siempre que el desarrollador sea un particular.

Las librerías desarrolladas por particulares y que se han incluido, con ciertas modificaciones, cuentan con la autorización para su uso libremente.

En cuanto a los requerimientos técnicos el servidor ha sido diseñado para ser instalado en un sistema de 64 bits con el Sistema Operativo Windows 10. Se recomienda utilizar un mínimo de 16 GB de memoria RAM.

Es necesario que las cajas en las que se monten los clientes dispongan de suficiente ventilación para evitar el sobrecalentamiento de sus componentes.

8 ESTUDIO ECONÓMICO

Se divide en dos partes: presupuesto para el diseño y la implementación del sistema principal, incluyendo el servidor únicamente, y los presupuestos para implementar cada uno de los prototipos diseñados. Estos últimos son precios aproximados, su coste final puede variar bastante en función del número de elementos, las diferentes configuraciones de cada uno y el precio del material en el momento del encargo.

8.1 Sistema de control de acceso

En este apartado se incluyen los costes necesarios para diseñar e implementar el sistema. No se incluyen los dispositivos capturadores ni actuadores.

Recursos humanos (30 €/hora):

- 80 horas de estudio y comparativa de tecnologías 2.400,00 €
- 160 horas para el diseño y el desarrollo del sistema 4.800,00 €
- 40 horas para el testeo del sistema y las pruebas 1.200,00 €
- 20 horas para preparar la documentación 600,00 €

Total:

- Subtotal 9.000,00 €
- IVA 1.890,00 €
- **Total** **10.890,00 €**

8.2 Prototipo A. Capturador Simple

Equipamiento necesario:

- Raspberry pi 3B 40,00 €
- Cámara Raspberry Pi v2.1 20,00 €
- Caja y ventilador 5,00 €

Total:

- Subtotal 65,00 €
- IVA 13,65 €
- **Total** **78,65 €**

Se ha presupuestado considerando el uso de la versión 3B de Raspberry pi, y la v2.1 de la cámara. El uso de las versiones Zero W y v1 supondría una reducción de 15 y 10 € respectivamente. Con un importe total aproximado de 54 €.

8.3 Prototipo B. Capturador con lector de Huellas

A los costes del prototipo anterior habría que añadir:

- Lector de huellas R308 20,00 €
- Arduino UNO 15,00 €
- Caja para lector de huellas 5,00 €
- Cableado, pantalla LCD, potenciómetro, etc. 10,00 €
- Una hora extra en mano de obra para el montaje 40,00 €

Obteniendo un sobrecoste de 90 €. Y un total:

- Subtotal 155,00 €
- IVA 32,55 €
- **Total** **187,55 €**

Se ha presupuestado considerando una tarjeta Arduino UNO, el coste de una tarjeta compatible podría reducir el presupuesto en 10 € aproximadamente.

8.4 Prototipo C. Capturador con teclado matricial

A los costes del prototipo A habría que añadir:

- Teclado matricial 2,00 €
- Caja para teclado 3,00 €
- Cableado y display 3,00 €
- Arduino NANO 7,00 €

Con un sobrecoste de 15 €. Total:

- Subtotal 80,00 €
- IVA 16,80 €
- **Total** **96,80 €**

8.5 Prototipo D. Capturador con sensor de movimiento

A los costes del prototipo A hay que añadir:

- Arduino UNO 15,00 €
- Caja para montar el dispositivo 5,00 €
- Cableado, Pantalla OLED, soporte del sensor, etc. 10,00 €

Lo que supone un sobrecoste de 30 € con respecto al prototipo A. Total:

- Subtotal 95,00 €
- IVA 19,95 €
- **Total** **104,95 €**

8.6 Prototipo H. Actuador

En este caso no es necesario el uso de cámara, por lo que se tiene un ahorro con respecto al prototipo A de 20 €.

Por otro lado, se tienen unos sobrecostes de:

- Arduino MEGA 20,00 €
- Caja para el dispositivo 10,00 €
- Cableado, relé, leds, etc 20,00 €

Con un sobrecoste total de 30 € con respecto al prototipo A. Total:

- Subtotal 95,00 €
- IVA 29,95 €
- **Total** **104,95 €**

9 CONCLUSIONES Y LÍNEAS DE FUTURO

El objetivo principal de este proyecto era conseguir un sistema centralizado que permitiera controlar el acceso a espacios de un edificio mediante el reconocimiento de usuarios, a través de un parámetro biométrico, en este caso utilizando imágenes. Considerando los resultados obtenidos podemos alcanzar las siguientes conclusiones:

- Tras valorar diferentes alternativas en cuanto a la técnica biométrica a utilizar, se ha optado por el reconocimiento facial. Las tres técnicas de reconocimiento facial implementadas presentan pocos errores y la tasa de acierto es muy alta siempre que las condiciones ambientales sean las adecuadas. Además, los tiempos de respuesta son bajos, lo que las hace viables para implementar un sistema de control de acceso. En definitiva, el reconocimiento facial es válido y los resultados obtenidos así lo demuestran.
- El sistema diseñado sería muy vulnerable en ausencia de sistemas que proporcionen seguridad en las comunicaciones, y así se ha demostrado en las pruebas realizadas. Por lo que se han incorporado mecanismos que permiten cifrar estas comunicaciones.
- Utilizar software libre y dispositivos de bajo coste, como Raspberry Pi, para realizar un sistema de control de acceso haciendo uso de reconocimiento facial es viable.
- El sistema diseñado es flexible y adaptable a las necesidades particulares que se puedan requerir a la hora de implantarlo y a las condiciones del entorno donde suceda.

Como reflexión final, se puede concluir que los objetivos principales marcados se han cumplido en su totalidad. No obstante, el reconocimiento facial aún presenta un porcentaje de error que, aun utilizando técnicas de reconocimiento complementarias como un lector de huellas, hace que su uso no pueda garantizar un reconocimiento 100% fiable y no se pueda recomendar su uso en determinados entornos donde la fiabilidad sea determinante.

9.1 Líneas futuras

El uso de la tecnología para realizar reconocimiento de personas mediante técnicas biométricas está en auge. Con frecuencia se producen avances en diferentes áreas relacionadas y estar al día de estas tecnologías requiere una renovación constante. Mientras este sistema ha sido desarrollado han surgido nuevas técnicas que no han sido evaluadas, del software utilizado han aparecido nuevas versiones, etc.

Para mantener el sistema actualizado sería conveniente realizar un mantenimiento de las diferentes librerías que se utilizan en él, como las nuevas versiones de OpenCV (actualmente 3.4) o de OpenSSL, cuya última versión soporta TLS1.3.

La interfaz podría mejorar ciertos aspectos:

- No se ha incorporado una búsqueda de usuarios, por nombre, puesto, etc. Lo que podría ser necesario en el caso de utilizar el sistema en un edificio grande con mucho tránsito de personas diferentes, como un hotel.
- Tampoco se da la posibilidad de eliminar un usuario. Se podría haber incorporado algún mecanismo, dado que, por ejemplo, una empresa, si podría tener la necesidad de eliminarlos.
- No se permite descartar imágenes o cambiar la imagen de portada. Salvo haciéndolo de forma manual.
- Permitir gestionar permisos por tipo de puesto/cliente ofrecería valor añadido.
- No se pueden eliminar registros de acceso. Añadir la posibilidad de mantenerlos solo durante un tiempo o un número máximo podría ser necesario para controlar el tamaño de la base de datos y el número de imágenes almacenadas.
- Incorporar mejoras visuales a la interfaz, como mapas del edificio que permitan una mejor gestión.

En cuanto al sistema de reconocimiento facial, además de incluir los nuevos algoritmos que puedan surgir, existen opciones como:

- Actualizaciones automáticas de las imágenes almacenadas, permitiendo añadir imágenes actualizadas en la base de datos cuando se producen nuevos reconocimientos positivos.
- Uso de cámaras de infrarrojos para permitir el reconocimiento en ausencia de luz o con poca iluminación. Varios trabajos de Stan Z. Li (Li, Zhang, et al. 2006) (Li, Chu, et al. 2006) (Li et al. 2007) y otros de Yi (Yi et al. 2007) o Xie (Xie y Liu 2011) estudian esta posibilidad.
- Se podría mejorar la seguridad haciendo uso de cámaras de profundidad, como la que incorpora el sistema Kinect de Microsoft.
- Ampliar su funcionalidad y permitir múltiples detecciones y reconocimientos simultáneos. El trabajo se ha centrado en la detección de un solo rostro, imágenes con más de uno podrían generar errores.

Otras opciones que podrían mejorar el sistema diseñado son:

- Incorporar movilidad en las cámaras mediante servo motores. Permitiría su control manual desde el servidor o, incluso, su movimiento automático para el seguimiento de un objeto o una persona haciendo uso de la función de seguimiento ya incorporada.
- Habilitar el sistema para el intercambio de clave XXTEA, que se podrían generar de forma aleatoria e intercambiarse entre los dispositivos para mejorar la seguridad. En este momento la clave es estática y su modificación requeriría intervenir en cada dispositivo.
- Incorporar sonido en los actuadores, con simples zumbadores, por ejemplo, o, incluso, utilizar voz digitalizada con herramientas como *espeak* («eSpeak: Speech Synthesizer» 2018) para dar instrucciones a los usuarios, en lugar de hacerlo con pantallas o leds. Facilitando el uso a personas invidentes o con problemas de visión
- Existen en el mercado muchas cámaras de video-vigilancia que hacen uso del protocolo onvif («Onvif» 2018). Para poder hacer uso de estas cámaras y ampliar funcionalidades se podría permitir utilizar este protocolo para la recepción del video que capturan estas cámaras.
- Se podrían añadir funcionalidades en cuanto a la grabación de video.
- No se ha valorado, en ningún caso, el valor de confianza que ofrece el lector de huellas, ni se han realizado pruebas que validen su idoneidad para incluirlo en el sistema. Otra opción sería incluir en los accesos el uso de otros mecanismos como las tarjetas con identificación por radiofrecuencia (RFID).

A lo largo de la presente memoria se han presentado estándares y recomendaciones cuyo cumplimiento añadiría valor al sistema, al permitirlo interactuar con otros sistemas o dispositivos que cumplan con estos.

Partiendo del trabajo realizado, existe la posibilidad de dotar al sistema de funciones domóticas completas. Por ejemplo, en el caso de una cerradura o una puerta automática, incorporar un contador de tiempo, podría ser suficiente con unos pocos segundos, tras los que se volvería a cerrar. Para la iluminación de un espacio se podría controlar que no queden usuarios dentro del mismo, lo que incluiría una contabilidad de personas por espacio, para proceder con el apagado de las luces. En definitiva, añadir control total sobre el estado de los dispositivos electrónicos finales, o, incluso, integrar el sistema dentro de un sistema domótico más complejo.

10 ANEXO I. MANUAL DE USUARIO

10.1 Servidor.

10.1.1 Interfaz principal

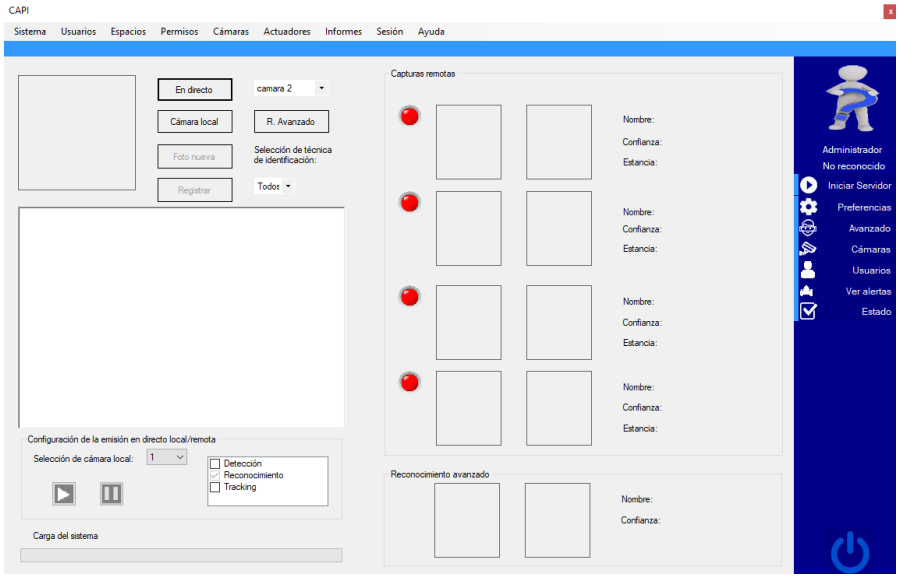

Figura 46 Interfaz principal

En la Figura 46 se muestra la ventana principal del servidor. En ella se aprecian las siguientes secciones diferenciadas:

- **Menú**. Desde el que se pueden acceder a todas las funcionalidades que ofrece el sistema. Más adelante se detallan cada una de las opciones que ofrece.
- **Control del directo**. Un espacio destinado a mostrar las emisiones en directo, tanto en remoto como en local. Incluye también una serie de botones que permiten:
    - **En directo**. Inicia la emisión en directo haciendo uso de la cámara seleccionada en el combo que se encuentra justo a su derecha. Este combo muestra la etiqueta asociada a cada una de las cámaras remotas (en la interfaz de usuario se hace referencia a los capturadores como cámaras para simplificar). Después pulsar podría no iniciarse la emisión, bien por no estar disponible el dispositivo o no disponer de emisión directo (es necesario que disponga del script correspondiente y se encuentre en ejecución), o bien porque la cámara se encuentre ocupada por otro proceso.
    - **Cámara local**. Se inicia la emisión en directo utilizando una cámara local. En la caja inferior se puede elegir el número de cámara en caso de que el servidor disponga de más de una.

    - **Foto nueva**. Cuando se está emitiendo en directo permite realizar una captura de un rostro para generar un nuevo usuario.
    - **Registrar**. Si se ha alcanzado el número mínimo de imágenes de un usuario se habilita la opción de registrar este usuario. Una vez se pulsa se crea el usuario y se muestra un formulario que permite cambiar los datos del usuario. Si a través del menú Usuarios se habilita la opción de Añadir imágenes, con un id valido, este botón cambia a **Añadir a usuario**, y las fotos que se realicen se incluirán en el usuario correspondiente.
    - **Reconocimiento avanzado**. Cuando se hace una emisión en directo, para no ralentizar en exceso, el reconocimiento no incluye el recorte y la rotación de las imágenes, lo que permite acelerar el proceso, pero hacer perder valor a los resultados. Para poder realizar un reconocimiento mejorado se puede hacer uso de este botón, que realiza un reconocimiento completo de la imagen que se está mostrando en el directo. El resultado aparece en el frame de reconocimiento avanzado.
    - **Combo para técnica**. En caso de tener el sistema entrenado para reconocer mediante más de una técnica este combo permite seleccionar la que se debe utilizar.
    - **Configuración de la emisión en directo**. Donde se permite habilitar las funciones para el directo, como iniciar o para el reconocimiento. Existen botones que permiten iniciar y pausar la emisión.
- **Ventana de información**. En la que se va mostrando diferente información de los procesos que realiza el servidor. Entre las preferencias existe la opción de configurar tres niveles que establecen la información que se mostrará por esta ventana.
- **Barra de carga**. Muestra cual es el estado actual del sistema en cuanto al número de hilos activos. El valor máximo de esta barra es de 100.
- **Capturas remotas.** Donde se muestra la información de las últimas cuatro conexiones remotas realizadas. El icono que representa un led rojo se cambia a verde si el reconocimiento corresponde con un acceso autorizado para el espacio correspondiente. En el cuadro de información se mostrará información sobre las actuaciones iniciadas.

- **Reconocimiento avanzado.** Muestra los resultados que se obtienen cuando se realiza un reconocimiento completo después de pulsar el botón correspondiente.

El cuadro que muestra las emisiones en directo de las cámaras, locales o remotas, permite, haciendo clic en él, doblar su tamaño. Siendo suficiente con volver a clicar sobre él para volver al estado normal.

Dentro de la cabecera se dispone de un menú con diferentes opciones:

- Sistema
- Usuarios
- Espacios
- Permisos
- Cámaras
- Actuadores
- Informes
- Sesión
- Ayuda

Para simplificar la interfaz, la presentación por defecto oculta el menú principal, para tener acceso a él es necesario hacer clic sobre la barra horizontal que hay en la parte superior. En la parte derecha de la ventana hay un menú reducido donde se tiene acceso rápido a las principales funciones del sistema. Que esté oculto o se muestre este menú adicional se controla haciendo clic sobre la barra vertical que lo incluye.

10.1.2 Submenú sistema. Preferencias

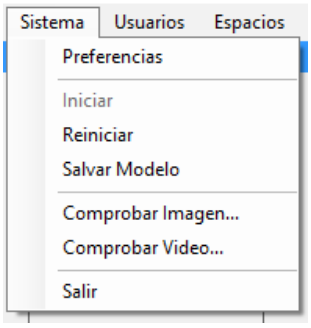

Figura 47 Submenú sistema

El submenú de **sistema**, que se muestra en la Figura 47, permite acceder a las preferencias, donde se puede configurar el sistema (si el sistema requiere que el usuario sea administrador debe haber un reconocimiento positivo o estará inactiva). Las otras posibilidades son **Iniciar**, que comienza con el entrenamiento del sistema, **Reiniciar**, que permite reiniciar el sistema, útil si se han realizado modificaciones en la configuración que

necesitan un reinicio. Es posible realizar el **reconocimiento en una imagen**, con formatos pgm o jpg, almacenada, o de un **video**, formatos avi o mov. En este caso, el video se reproducirá en una ventana nueva. Se puede detener pulsando la tecla q. En el video se mostrará información de reconocimiento y del tracking que se realiza en él.

También se dispone de la posibilidad de salvar el modelo que se ha entrenado. Hay que tener en cuenta que si entre las opciones se ha seleccionado el autosalvado no es necesario hacerlo de forma manual, y que si en se están usando los algoritmos Eigenfaces o Fisherfaces y se han creado nuevos usuarios o se han introducido nuevas imágenes en el sistema el modelo que se salve en ningún caso estará actualizado, habría que reiniciar y volver a entrenar antes para guardar.

Por último, está la opción de Salir del sistema.

Existe la posibilidad de configurar el sistema para que el entrenamiento para el reconocimiento se realice de forma automática al iniciar el sistema, se verá cómo hacerlo más adelante. En cualquiera de los dos casos aparecerá una ventana como la que aparece en la Figura 48 avisando del estado del proceso. El tiempo necesario para la carga dependerá del número de imágenes, de usuarios e, incluso, de la potencia del equipo.

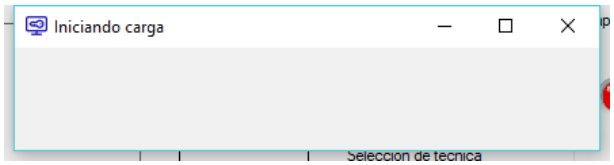

Figura 48 Aviso de inicio de entrenamiento

Una vez se accede al menú de preferencias es posible configurar el sistema.

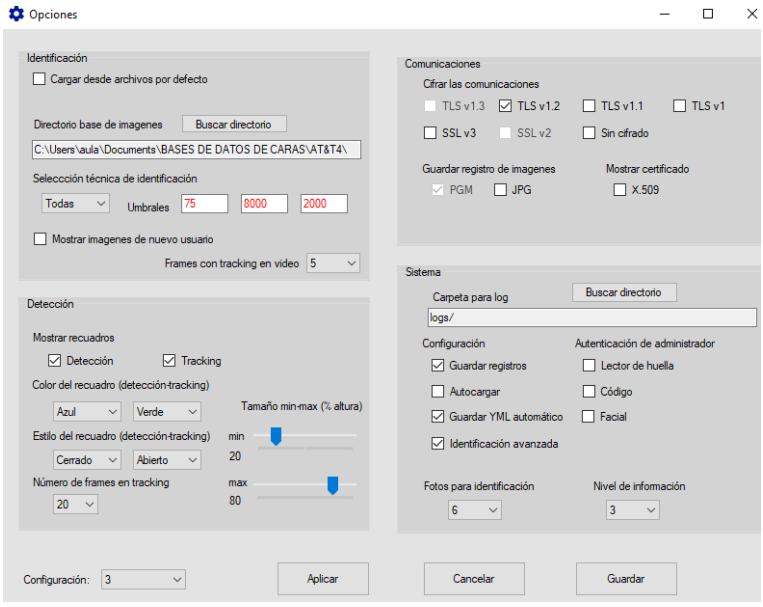

Figura 49 Preferencias del sistema

Dentro de las opciones de configuración se distinguen:

1. Identificación
2. Detección
3. Comunicaciones
4. Sistema

Existen hasta cuatro formas diferentes de configurar el sistema que pueden ser guardadas. En la parte inferior izquierda del formulario se elige la deseada. Cuando una de estas configuraciones se modifica y se pulsa en guardar la configuración almacenada se actualiza, pero no se aplica, para que se apliquen cambios en la configuración actual es necesario pulsar el botón Aplicar. Si se desea guardar y aplicar es necesario hacerlo en este orden, primero guardar las modificaciones y luego aplicar la configuración.

10.1.2.1 Identificación

Dentro de las opciones configurables en el bloque de identificación existen:

- La opción de que, por defecto, se realice la carga desde un archivo determinado, evitando el tiempo de entrenamiento. Esta opción es útil si no se han realizado modificaciones desde que se guardó el archivo.
- También se puede modificar el directorio donde se guardan las imágenes. Modificar esta opción en un sistema en funcionamiento puede provocar errores en el sistema, solo se recomienda modificar este valor al configurar el sistema por primera vez.
- Seleccionar la técnica de reconocimiento a utilizar y el valor de umbral de cada técnica. Cuando se modifica la técnica de reconocimiento es necesario guardar y reiniciar el sistema para que surta efecto este cambio.
- Es posible determinar si se mostrarán las imágenes capturadas cuando se da de alta un nuevo usuario en ventanas independientes.
- Por último, se selecciona el número de frames en los que se utilizará seguimiento en lugar de reconocimiento.

10.1.2.2 Detección

Cuando se realiza una detección los valores configurables son:

- Mostrar los cuadros de detección y seguimiento.
- Es posible seleccionar el color y la forma (abierto o cerrado) de estos cuadros.
- Se puede elegir el número de frames durante los que se realiza el seguimiento, en lugar de la detección, en los directos.
- Por último, es posible modificar los tamaños mínimo y máximo de los cuadros en los que se realizará una detección facial. Estos valores afectan

al rendimiento de sistema y no es aconsejable dar un rango excesivamente amplio. En las pruebas realizadas 20% y 80%, respectivamente, han dado buenos resultados.

10.1.2.3 Comunicaciones

En este bloque se pueden configurar parámetros que determinarán la forma en que se realizarán las comunicaciones:

- Es posible elegir el protocolo a seguir para el cifrado de las comunicaciones. Es recomendable elegir la opción más segura, TLSv1.2 en la versión actual (la v1.3 una vez se actualice). El resto de opciones son menos seguras. Existe, además, la opción de no cifrar las comunicaciones, esta versión ya se ha demostrado que es muy vulnerable, pero podría ser interesante en entornos donde la seguridad prime menos que la velocidad y, por ejemplo, sea necesario utilizar dispositivos menos potentes, como la Raspberry pi Zero. Si está marcada esta opción todas las comunicaciones se realizarán sin cifrar.
- Existe la opción de elegir el formato en que se guardarán las imágenes en los registros de acceso. Por defecto las imágenes se guardan siempre en formato PGM, pero opcionalmente se pueden guardar también en formato JPG.
- Si se muestran datos de los certificados cuando se realicen conexiones.

10.1.2.4 Sistema

Por último, es posible modificar parámetros del funcionamiento del sistema:

- Es posible elegir si se guardan registros de las conexiones.
- Autocargar indica al sistema que todos los servicios arrancan al iniciar el servidor o si es necesario hacerlo manualmente. Estos servicios incluyen el entrenamiento para reconocimiento o el lanzamiento de los sockets de escucha.
- Guardar YML automático ofrece la posibilidad de que cada vez que el sistema realiza un entrenamiento para reconocimiento el correspondiente archivo YML con el entrenamiento se guarda automáticamente. La carpeta donde se encuentre instalado el sistema es la seleccionada por defecto.
- Identificación avanzada indica al sistema que realice un recorte y una rotación de las imágenes recibidas de los capturadores para el reconocimiento. Similar al botón de reconocimiento avanzado, pero para las

imágenes recibidas. Es recomendable que esta opción esté activada siempre.

- La autenticación para administrador indica que técnicas se utilizarán para identificar un administrador en el servidor. Si no hay ninguna marcada las opciones de administrador permanecen siempre habilitadas (incluye modificar opciones del sistema y dar permisos). Es recomendable que, al menos una de estas opciones, esté activa.
- Se puede seleccionar el número mínimo de imágenes que se deben tener de un usuario para poder darlo de alta en el sistema.
- Finalmente, el nivel de información indica que mensajes se mostrarán del funcionamiento del sistema. Hay tres niveles: el primer nivel muestra solo la información básica, en el segundo nivel se incluye información relativo al estado del sistema en cuanto al número de hilos en funcionamiento, datos de los certificados en las conexiones, etc. En el tercer nivel se incluye toda la información de los procesos que se están ejecutando.

10.1.3 Submenú de Usuarios.

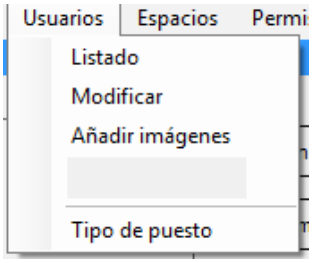

Figura 50 Submenú Usuarios

Este submenú permite gestionar los datos de los usuarios. En la Figura 51 se muestra la ventana que contiene una lista de los usuarios que existen actualmente en la base de datos. Haciendo doble clic en uno de los usuarios listados o seleccionando uno y pulsando el botón modificar se accede al formulario de la Figura 52, donde se pueden modificar los dados de un usuario. Cuando, a través de la interfaz principal, se crea un nuevo usuario un formulario como este permite cambiar los datos del nuevo usuario. Por defecto se utiliza “nuevo” para el nombre y los apellidos.

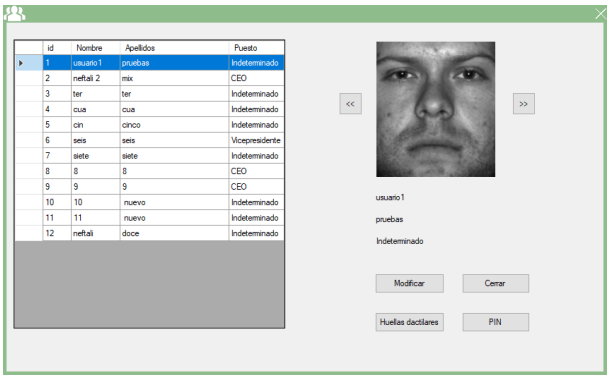

Figura 51 Lista de usuarios

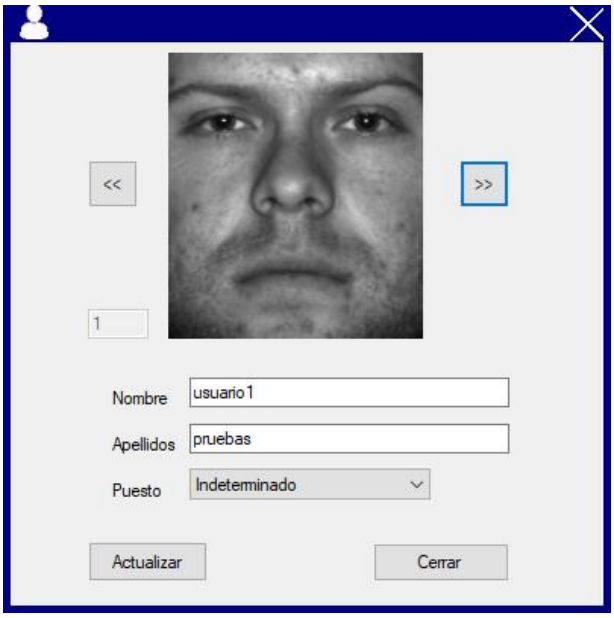

Figura 52 Datos de un usuario

Además del botón de modificar, la ventana proporciona acceso, mediante los botones **huellas dactilares** y **PIN**, a los formularios encargados de gestionar las huellas dactilares almacenadas por el usuario en un determinado lector de huellas, y a los códigos de acceso relacionados con los capturadores para el usuario.

El lector de huellas y el lector de códigos con identificadores dos y uno, respectivamente, se reservan para los lectores que se conecten al servidor.

Las ventanas que permiten gestionar las huellas y los códigos de acceso se pueden ver en la Figura 53 y en la Figura 54. En ambos casos se distingue cuando se trata de la gestión de un lector conectado a un capturador o la gestión de un lector conectado al propio servidor.

Para eliminar un código es suficiente con seleccionarlo de la lista y pulsar borrar. En la parte superior se puede añadir uno nuevo, seleccionando el lector correspondiente, introduciendo el código y pulsado en Añadir. Salvo si se quiere añadir en el lector asignado

al servidor, en este caso solo es necesario pulsar en Añadir servidor después de introducir el código correspondiente al usuario.

La gestión de las huellas se realiza de forma similar, salvo que cuando se está asignando una nueva será necesario utilizar el lector y seguir los pasos que se van indicando en la pantalla para registrar la nueva huella. En este caso el código asignado para la gestión interna del lector será el primero libre.

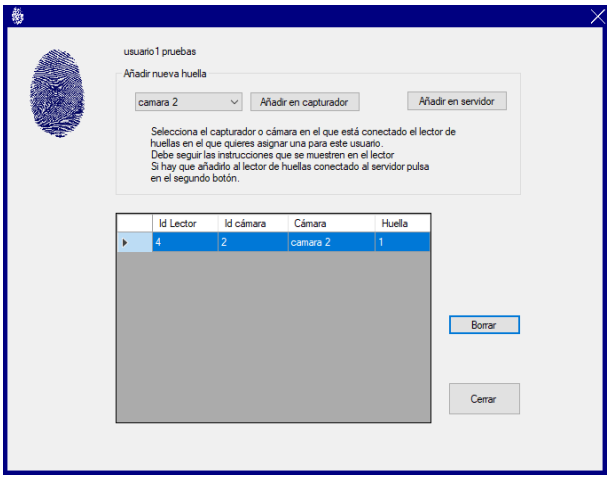

Figura 53 Gestión de huellas

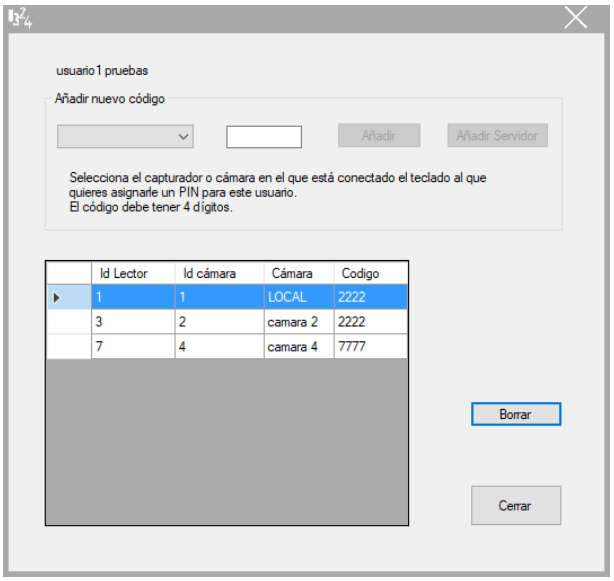

Figura 54 Gestión de códigos

Otra opción de la que se dispone es la de añadir nuevas imágenes a un usuario ya existente. En este caso se debe introducir el código del usuario y pulsar en Añadir imágenes dentro del submenú usuarios, tras lo cual el botón de registrar cambia de función, y su texto, y añadirá las nuevas capturas a la base de datos. Una vez se comienza para detener el proceso hay que pulsar el botón sin haber realizado ninguna captura, si se ha realizado la única forma de pararlo es cerrando la aplicación.

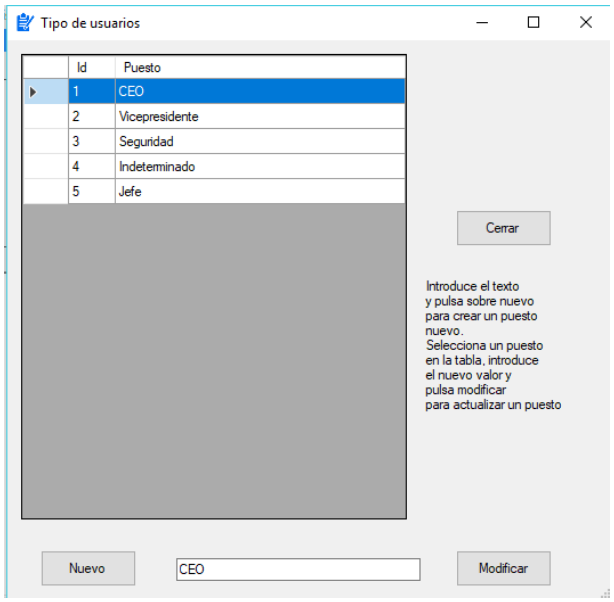

Figura 55 Tipos de usuario

Por último, existe la opción de gestionar el tipo de usuario. Aunque la aplicación realmente no hace distinción en función del tipo de usuario se ofrece esta posibilidad para facilitar la gestión de usuarios.

10.1.4 Submenú de Espacios

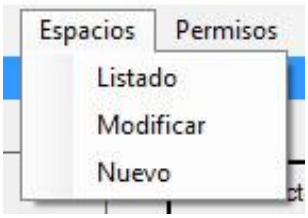

Figura 56 Submenú Espacios

Este submenú permite configurar no solo las estancias de un edificio también se podrían incluir en estos listados servicios cuyo uso pueda ser controlado. Como, por ejemplo, el uso de una máquina de café.

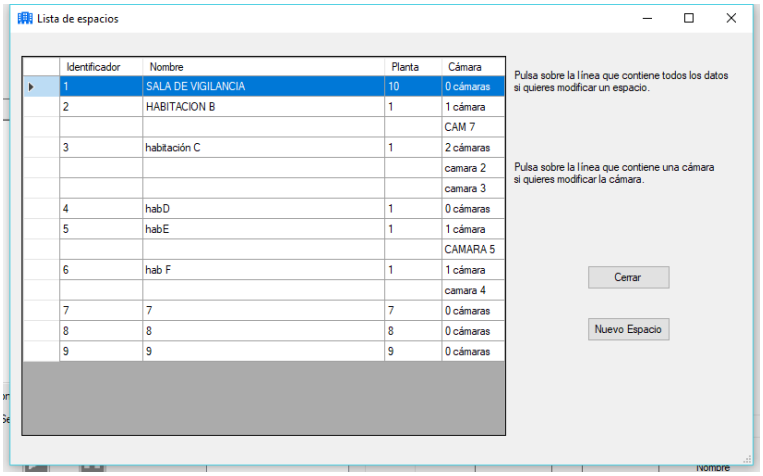

Figura 57 Lista de espacios

El listado de espacios de la Figura 57 incluye, además de los espacios, las cámaras asociadas a los mismos. Pulsando sobre los espacios se accede al formulario de la Figura 58 donde se pueden gestionar los mismos. En él es posible asignarle o quitarle una cámara, además de gestionar los permisos relacionados con el espacio en cuestión y para cada usuario.

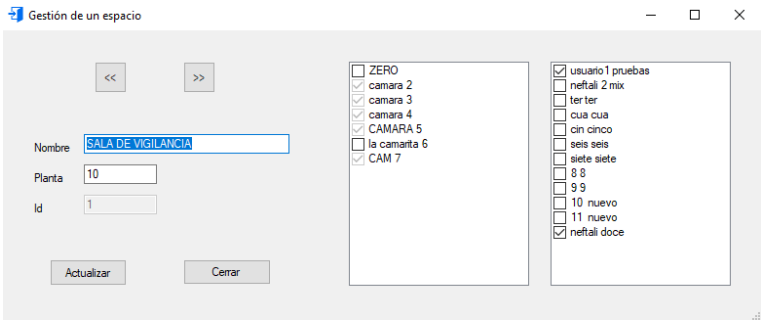

Figura 58 Gestión de un espacio

Dentro del listado de Espacios también se puede acceder a la configuración de las cámaras directamente, haciendo clic en el listado de espacios. En este caso se mostraría un formulario como el de la Figura 59. Donde se pueden configurar parámetros de la cámara.

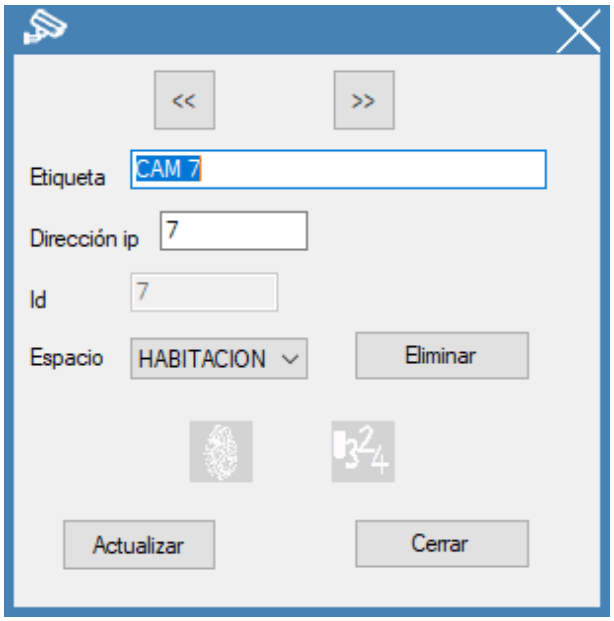

Figura 59 Configuración de cámara

10.1.5 Submenú Permisos

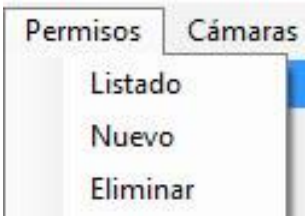

Figura 60 Submenú permisos

Dentro del submenú permisos, que se pude ver en la Figura 60, es posible acceder al listado de permisos, como el de la Figura 61. En este formulario se permite realizar un filtro de búsqueda, por espacio o por usuario, seleccionándolos en el combo correspondiente. Para volver al listado general habría que pulsar el botón de reset.

Es posible acceder, también a la gestión de permisos, Figura 62. Se puede acceder pulsando en Nuevo o en Eliminar, en ambos casos el formulario es similar y se puede cambiar la funcionalidad dentro del mismo pulsando el botón de la doble flecha. Una vez se selecciona un espacio y un usuario solo hay que pulsar en la opción correspondiente y la base de datos será actualizada.

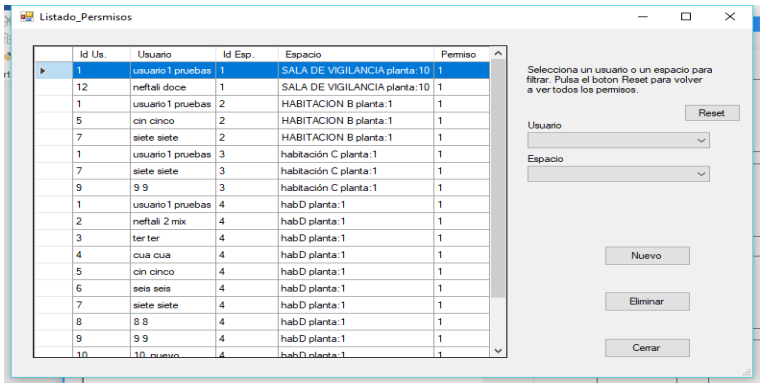

Figura 61 Listado de permisos

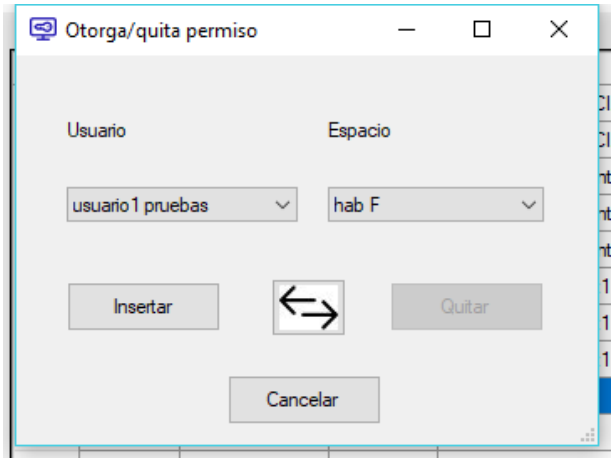

Figura 62 Gestión de permiso

10.1.6 Submenú Cámaras

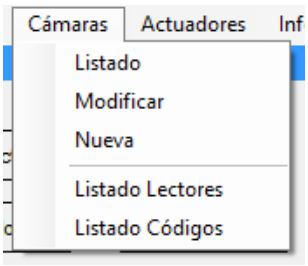

Figura 63 Submenú cámaras

Aunque dentro de la aplicación se denomina cámaras a este tipo de elementos realmente se corresponde con un dispositivo capturador. Dado que no tienen sentido el uno sin el otro se han considerado el mismo dispositivo y, por simplificar, se gestionan como si fueran una cámara IP.

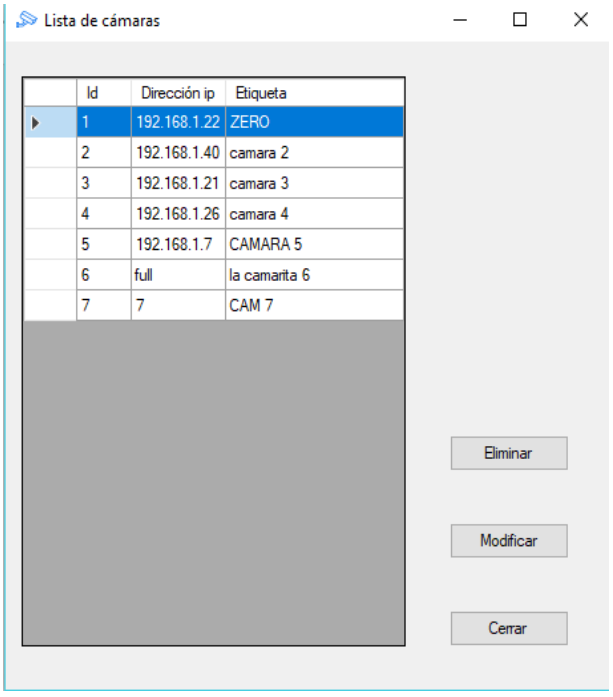

Figura 64 Lista de cámaras

Desde el listado de cámaras, Figura 64, se puede acceder a la configuración de una, Figura 59, seleccionándola y pulsando en modificar. Desde el submenú se puede crear una nueva cámara también.

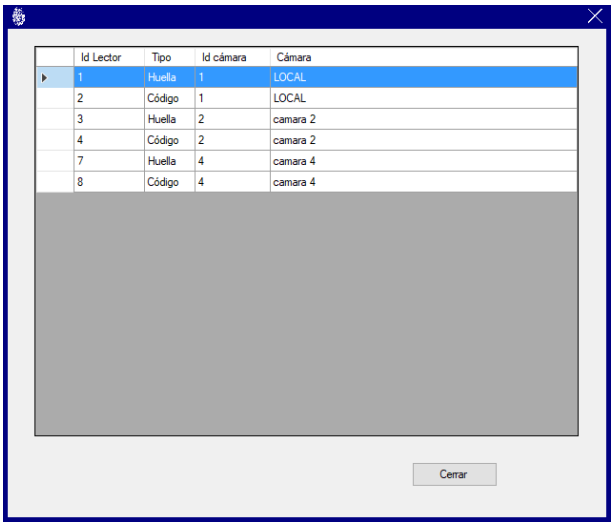

Figura 65 Lista de lectores

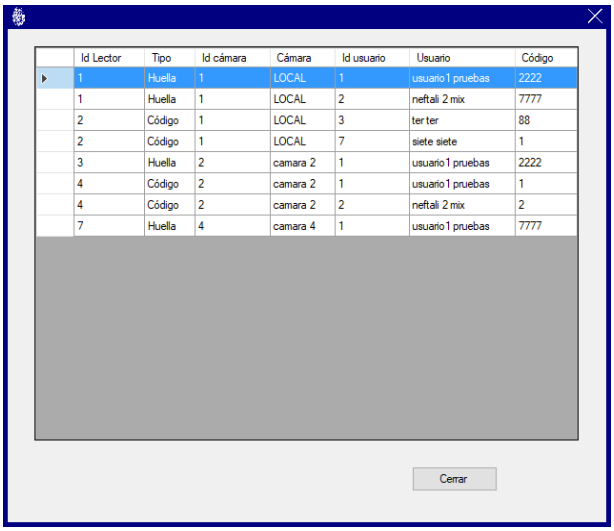

Figura 66 Listado de códigos

En la Figura 65 y en la Figura 66 se muestran el listado de lectores y el listado de códigos asociado a estos mismos lectores. No se incluye gestión de lectores dentro de este menú, ya que la creación y borrado de los mismos se gestiona desde el menú de cámara, donde la existencia o no de lectores asociados determina si existen o no en la base de datos. En la Figura 59 se aprecia una cámara sin lectores asociados, mientras que en la Figura 67 se muestra la configuración de una cámara en la que se han asociado lectores de huellas y teclado numérico. Para activar o desactivarlos basta con pulsar el botón. El botón verde indica que el lector correspondiente existe.

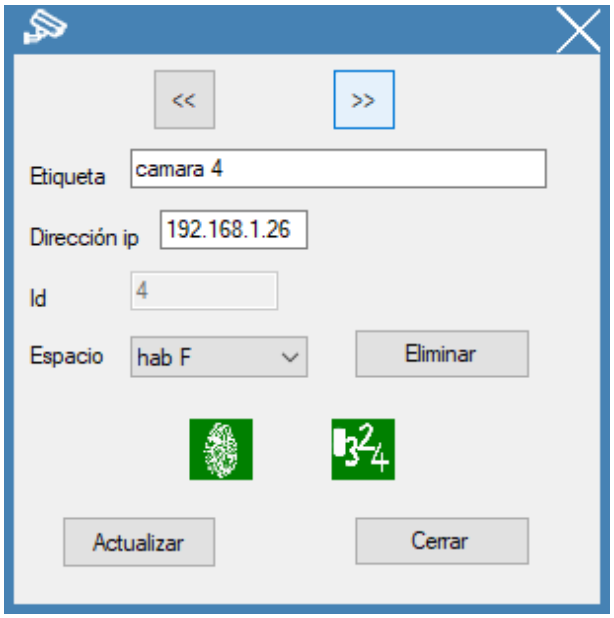

Figura 67 Cámara con lector de huellas y teclado

10.1.7 Submenú Actuadores

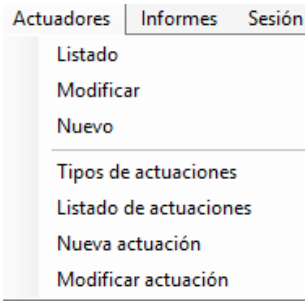

Figura 68 Submenú actuadores

El submenú de los actuadores, Figura 68, permite acceder al listado de actuadores, Figura 69, desde donde se pueden crear nuevos actuadores o modificar datos de uno ya existente en un formulario como el de la Figura 70.

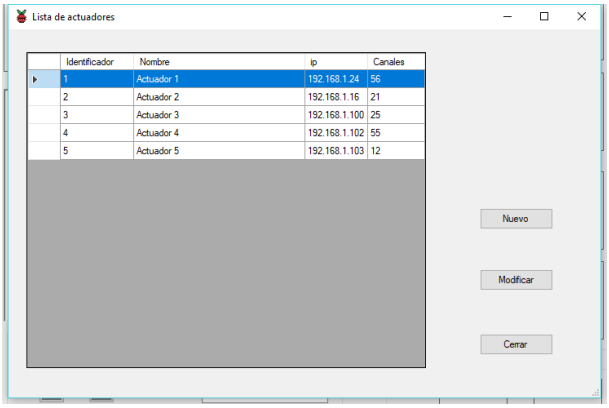

Figura 69 Lista de actuadores

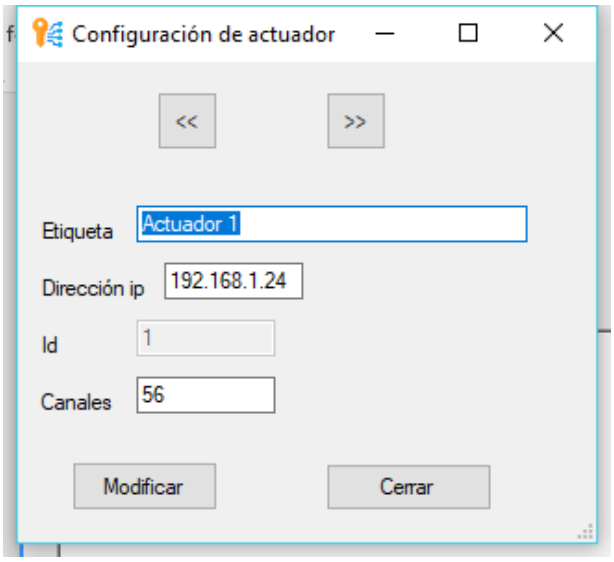

Figura 70 Configurar un actuador

Es posible mostrar un listado de actuaciones, Figura 71, donde se muestran los canales activos de los que dispone cada actuador. Cada canal se corresponde con una actuación, y se puede configurar. Cuando un usuario que tiene permiso para acceder a un determinado espacio es reconocido por una cámara que da acceso al mismo se ejecutarán todas las actuaciones incluidas en este formulario que cumplan las siguientes premisas:

- Como se ha comentado, el usuario ha de estar autorizado (Figura 61)
- Se activarán todas actuaciones cuya cámara coincida con la que ha realizado el reconocimiento positivo y que tengan como usuario el reconocido o que se activen con cualquiera (valor de -1 en la tabla Figura 71).
- Del mismo modo se activarán todas las actuaciones ligadas a la identificación del usuario en esta cámara o las que se activen con cualquier cámara (-1 en la columna id cam de la Figura 71)

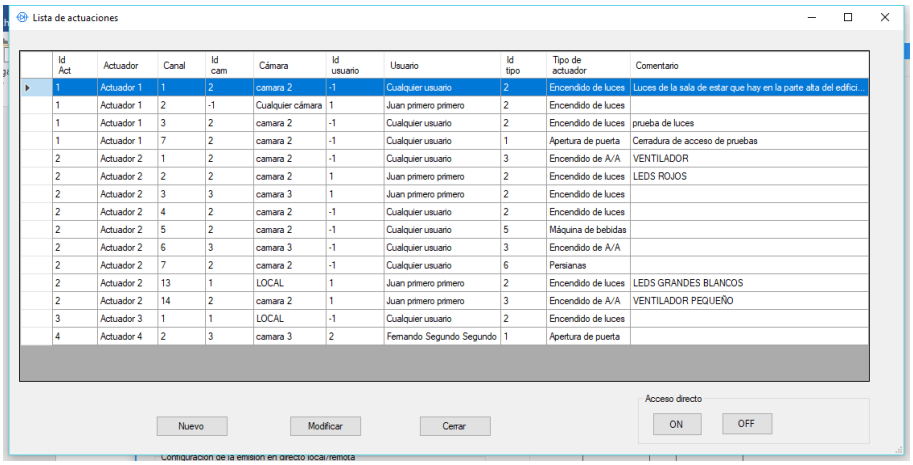

Figura 71 Lista de actuaciones

Para el ejemplo. Si se reconociera al usuario con id=2 en la cámara con id=3 se ejecutarían las actuaciones:

- Canal seis del actuador dos. Se activa con cualquier usuario en la cámara tres.
- Canal dos del actuador cuatro. Se actica con el reconocimiento del usuario dos en la cámara tres exclusivamente.

Otra de las opciones incluidas es la de crear y modificar elementos. Es simplemente el tipo de elemento que existe, para poder distinguirlos y gestionarlos más fácilmente.

Se ha incluido la posibilidad de controlar desde la base de datos si su activación se realiza con un valor lógico alto o bajo, aunque realmente esta posibilidad se podría gestionar de forma sencilla con la electrónica. Si el valor de activación es alto cuando una actuación es lanzada es un valor lógico uno el que se enviará por el canal correspondiente del actuador.

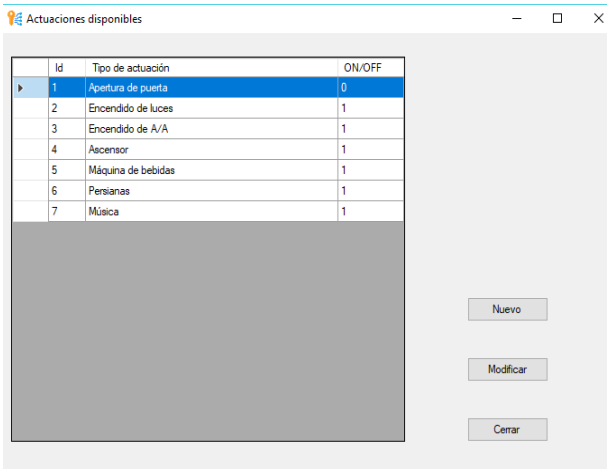

Figura 72 Actuaciones posibles

10.1.8 Submenú de Informes

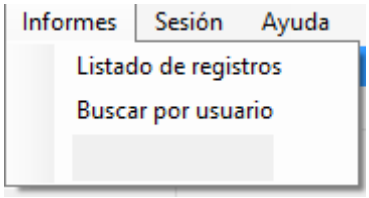

Figura 73 Submenú informes

El submenú informes, Figura 74, permite mostrar un listado de registros, como el de la Figura 75. Existe la opción de filtrarlo por id de usuario desde el propio submenú.

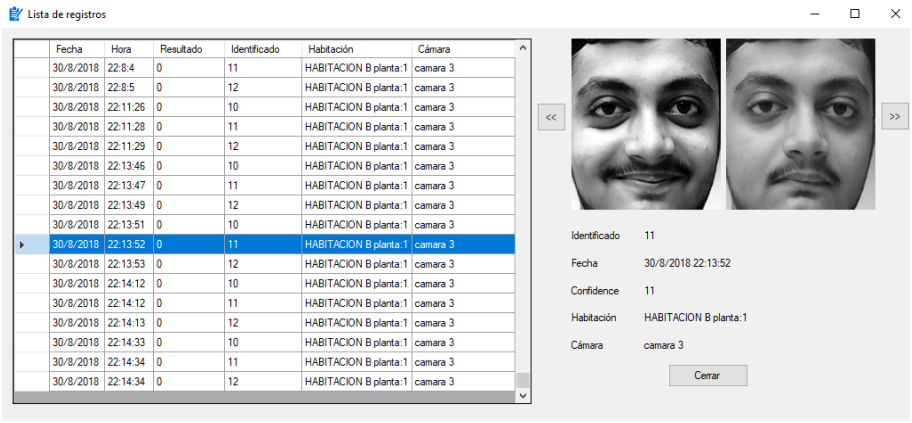

Figura 74 Listado de registros

10.1.9 Submenú Sesión

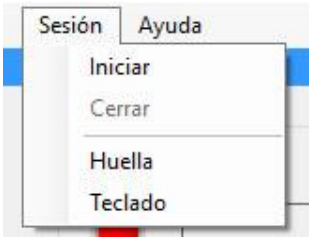

Figura 75 Submenú sesión

Desde el submenú Iniciar sesión, Figura 75, se puede proceder al inicio de sesión como administrador. Dependiendo de la configuración del sistema será necesario iniciar sesión de una u otra manera.

En este mismo submenú se mostrará el puerto utilizado por los lectores, en caso de estar conectados. Pulsando sobre el lector correspondiente se envía un mensaje de inicio de lector, necesario si el lector no se ha reconocido correctamente (aunque podría ser necesario reiniciar en algunos casos).

10.1.10 Submenú Ayuda

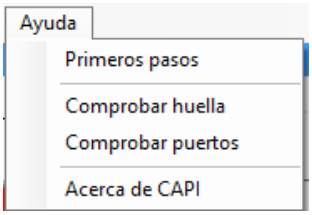

Figura 76 Submenú Ayuda

El submenú de Ayuda, Figura 76, proporciona acceso a un manual de usuario electrónico. Permite poder realizar la comprobación de una huella dactilar y comprobar el estado de los puertos serie, utilizador por el lector de huellas y el teclado matricial.

10.1.11 Barra lateral

La barra lateral permite acceder a algunos de los principales elementos del menú. Todas las funciones en ella incluida se encuentran en alguno de los submenús ya vistos, a excepción de la señal de alerta. Si en alguno de los actuadores se pulsa el botón de alarma el icono de “ver alertas” incluido en esta barra cambia de color y muestra un listado de las cámaras que han emitido una alerta al ser pulsado.

10.2 Prototipos.

El funcionamiento de los prototipos es similar en todos los casos. Existe un desencadenante que inicia la captura del rostro del usuario, una vez el rostro es capturado se envía al servidor y se vuelve al estado inicial.

En función del Prototipo este desencadenante es:

- La pulsación de un botón, como en el capturador completo (Prototipo G) o cuando se usa un lector de huellas (Prototipo B). En ambos casos se requiere colaboración, realizando parpadeos o colocando la huella dactilar.
- La detección de un movimiento (Prototipo D).
- Permanecer un tiempo a una distancia inferior a la marcada (Prototipo E).

En cualquier caso, el usuario solo debe actuar en función del tipo de capturador y, además, la mayoría de ellos incorpora pantallas OLED o LED que van dando indicaciones.

En algunos capturadores se pueden incluir botones de alarma, que permiten enviar una alerta al servidor en caso de ser necesario un aviso, para dar de alta una huella dactilar o iniciar el alta de un usuario en la base de datos, por ejemplo.

11 ANEXO II. MANUAL DE MANTENIMIENTO

11.1 Instalación y configuración de Raspbian

Independientemente del uso que se vaya a dar a la Raspberry que se está configurando, capturador o actuador, los parámetros de configuración son, básicamente, los mismos. El primer paso es descargar el sistema operativo de la web de Raspberry («Raspbian» 2018).

Entre las opciones que existen para instalar el sistema se puede optar por usar la herramienta Etcher («Etcher by resin.io» 2018). Es necesario descargar la imagen del sistema, formatear una tarjeta SD siguiendo las indicaciones de la web y utilizar el software Etcher para instalar el sistema en la tarjeta. Es recomendable un mínimo de 8 GB, aunque con 4 GB podría ser suficiente en función del uso que se le vaya a dar. En los dispositivos diseñados para este trabajo se ha optado por tarjetas de 8 y 16 GB.

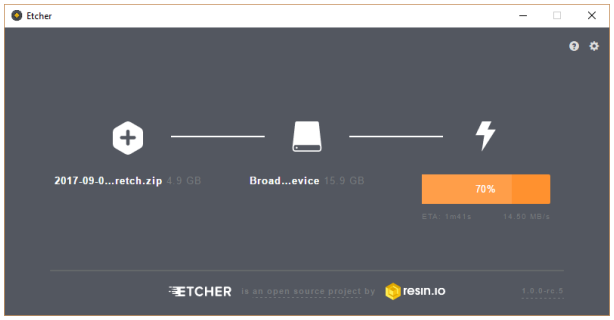

Figura 77 Etcher

Una vez se tenga el sistema instalado, se comienzan con las actualizaciones del mismo. Para ello se hace uso del terminal. Como paso previo se pueden cambiar el idioma, teclado, hora, wifi, y se activa cámara en configuración. Se actualiza el sistema:

sudo apt-get update

sudo apt-get upgrade

sudo rpi-update

Alguna versión de Raspbian incluye una versión antigua de OpenCV que puede dar problemas y es conveniente desinstalar:

*sudo apt-get remove libopencv**

sudo apt-get autoremove

Es necesario reiniciar.

sudo reboot

Se instalan algunas herramientas necesarias:

sudo apt-get install build-essential cmake cmake-curses-gui pkg-config

Y ciertas librerías:

```
sudo apt-get install \
  libjpeg-dev \
  libtiff5-dev \
  libjasper-dev \
  libpng12-dev \
  libavcodec-dev \
  libavformat-dev \
  libswscale-dev \
  libeigen3-dev \
  libxvidcore-dev \
  libx264-dev \
  libgtk2.0-dev
```

Es necesario instalar las cabeceras de Python, en este proyecto se ha utilizado Python 2.7. Esta versión de Pyhton viene ya instalada en el sistema Raspbian.

```
sudo apt-get install python2.7-dev
```

Tras tener todas las dependencias y librerías instaladas, el siguiente paso es descargar el código fuente de la web de OpenCV, o usar el repositorio de github. Se ha utilizados la versión 3.2, ya está disponible la 3.4, pero la 3.2 es suficiente para nuestros objetivos.

```
wget -O opencv.zip https://github.com/opencv/opencv/archive/3.2.0.zip
unzip opencv.zip
```

Para poder hacer uso de todas las funcionalidades de OpenCV, incluido el reconocimiento facial, es necesario incluir la librería *contrib* junto con la principal de OpenCV.

```
wget -O opencv_contrib.zip
https://github.com/Itseez/opencv_contrib/archive/3.2.0.zip
unzip opencv_contrib.zip
```

También será necesario el gestor de paquetes pip.

```
wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
```

Llegados a este punto se puede considerar la posibilidad de usar entornos virtuales, aunque no será necesario para el presente proyecto al estar dedicadas en exclusiva al mismo las diferentes Raspberry. El siguiente paso es instalar *numpy*, biblioteca de Python que permite operar con vectores y matrices, necesaria para trabajar con las imágenes de OpenCV.

pip install numpy

Ahora se compila OpenCV:

```
$ cd ~/opencv-3.2.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
        -D CMAKE_INSTALL_PREFIX=/usr/local \
        -D BUILD_WITH_DEBUG_INFO=OFF \
        -D BUILD_DOCS=OFF \
        -D BUILD_EXAMPLES=OFF \
        -D BUILD_TESTS=OFF \
        -D BUILD_opencv_ts=OFF \
        -D BUILD_PERF_TESTS=OFF \
        -D INSTALL_C_EXAMPLES=OFF \
        -D INSTALL_PYTHON_EXAMPLES=ON \
        -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-
3.2.0/modules \
        -D ENABLE_NEON=ON \
        -D WITH_LIBV4L=ON \
```

Tras tener todo listo:

```
 $ make -j4
```

En este comando es importante el j4. Que indica que se utilizan los cuatro núcleos (si se usa la versión Zero solo sería uno).

```
sudo make install
sudo ldconfig
sudo apt-get install python-opencv
sudo apt-get install libgtk2.0-dev pkg-config
```

Para poder usar la cámara de la Raspberry Pi con el comando *videocapture* se necesita:

```
sudo modprobe bcm2835-v4l2
sudo apt-get -y install libv4l-dev v4l-utils
```

Se instala imutils (Rosebrock 2018)

```
pip install imutils
```

Más librerías necesarias

sudo apt-get install libatlas-base-dev gfortran

Para controlar la temperatura de la Raspberry Pi se puede crear un alias añadiendo al final del archivo *bashrc* (útil considerando los problemas de calentamiento que sufren cuando se cargan de trabajo):

sudo nano .bashrc (en home/pi/)

alias temperatura='/opt/vc/bin/vcgencmd measure_temp'

Para poder controlar el parpadeo es necesario añadir:

sudo apt-get install python-scipy

Y para dlib:

sudo apt-get install libgtk-3-dev

sudo apt-get install libboost-all-dev

pip install scipy

pip install scikit-image

pip install dlib

Con estos pasos se tiene configurado el sistema básico en una Raspberry Pi. Para poder trabajar en el sistema final sería necesario añadir otras funcionalidades, como OpenSSL o XXTEA, que se explican a continuación.

11.2 Configurando OpenSSL

En el servidor, con sistema operativos Windows, y en los capturadores y actuadores se instala OpenSSL («OpenSSL» 2018) siguiendo las instrucciones de su web. Se accede a la carpeta bin de openssl (en Raspbian no es necesario) y **se genera un clave RSA** escribiendo el siguiente comando:

openssl genrsa -aes256 -out local.key 2048

Se está utilizando OpenSSL para generar una clave AES con un bloque de 256 bits que quedará almacenada en el archivo local.key y que utiliza una clave de 2048 bits. Pedirá una contraseña (se ha utilizado “linares”).

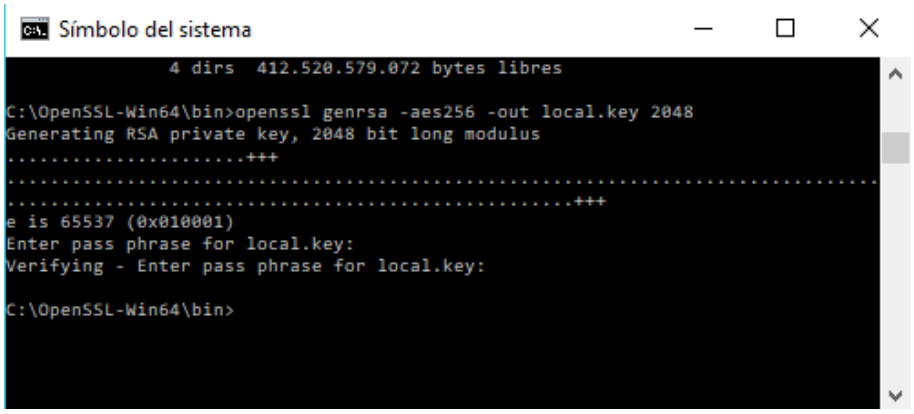

Figura 78 Generando clave AES

El siguiente paso es **generar el CSR** (*Certificate Signing Request*). Para esto, se utiliza el siguiente comando:

openssl req -new -key local.key -out local.csr

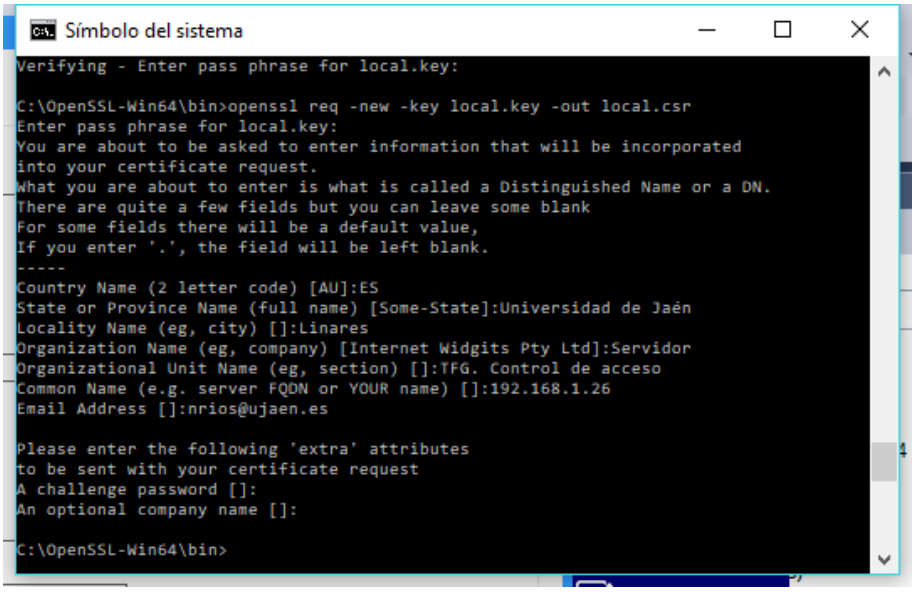

Figura 79 Generando CSR

Donde se está creando un certificado PKCS#10, que utilizará la clave privada generada anteriormente. Se crea como local.csr. Pide la contraseña de la clave privada, y algunos datos:

- Nombre del país (utilizando para ello 2 letras especificadas en el ISO 3166).
- Nombre completo del estado o provincia.
- Nombre de la localidad (por ejemplo, la ciudad).
- Nombre de la organización (por ejemplo, de la compañía).
- “Common Name”, aquí se debe introducir el dominio (preferiblemente sin *www*) de la web donde se va a utilizar este certificado. En este caso, será **localhost**.
- Dirección de email.
- Datos complementarios: una contraseña y un nombre opcional para la compañía.

Si no se quiere rellenar alguno de los datos, simplemente se pone un punto y se pulsa Enter.

Para evitar que pida la contraseña cada vez, lo único que se debe hacer es generar una copia de la clave privada sin contraseña. Se genera una copia, igual nombre añadiendo una extensión org al final, teniendo así el archivo key original como clave privada sin contraseña:

copy local.key local.key.org (cp en Raspberry)

Y ahora se le indica a OpenSSL que le quite la contraseña, reemplazando el archivo original.

openssl rsa -in local.key.org -out local.key

Y, por último, se debe **crear y firmar el certificado**. Esto se hace de la siguiente manera:

openssl x509 -req -days 365 -in local.csr -signkey local.key –out local.crt

Esto es:

x509 -> Utilidad de OpenSSL para mostrar y firmar certificados.

-req -> Con esta opción, en lugar de un certificado se espera una *petición de certificado* (CSR).

-days [valor numérico] -> Número de días que se espera que el certificado sea válido.

-in [nombre del archivo] -> Especifica el nombre del archivo desde donde se van a tomar los datos.

-signkey [nombre del archivo] -> Esta opción hace que el archivo sea auto-firmado.

-out [nombre del archivo] -> Establece el nombre de salida del archivo firmado.

Finalizado este proceso, se deben tener en la carpeta de certificados, los siguientes archivos:

local.crt será el certificado.

local.csr que es una solicitud de firma de certificado.

local.key con la clave privada sin contraseña.

local.key.org con la clave privada con contraseña.

Para ver datos del certificado se utiliza:

openssl x509 -in local.crt -noout -text

Para evitar ciertos errores es conveniente ejecutar en la línea de comandos:

set OPENSSL_CONF=C:\OpenSSL-Win64\bin\openssl.cfg

Para facilitar la gestión a todos los archivos relacionados con el servidor se les ha renombrado como “server” seguidos de la extensión correspondiente. Por su parte, en los clientes se han generado todos los archivos manteniendo un nombre identificativo (PiZero.crt por ejemplo para la Raspberry zero) dependiendo del modelo de capturador, aunque para simplificar se han creado copias de los archivos key y crt con el nombre local (que es como se han denominado en el código). Es necesario incluir los certificados de los clientes en el servidor y el del servidor en los clientes.

Para usar certificados auto-firmados, en el servidor, que, además, pueden repetirse (se podrían clonar sistemas en dispositivos iguales) se debe usar *SSL_CTX_load_verify_locations* con *CApath*. Para ello se crea la carpeta CA (vale

cualquier nombre) y dentro de ella se copian los certificados cambiando el nombre del archivo por el hash que se obtiene:

openssl x509 -in "nombre.crt" -noout -subject_hash

Ahora se pueden hacer enlaces al CA original o cambiar el nombre del archivo que debe ser el hash.X siendo x un número, se usa por si hay 2 hash iguales (sin incluir .crt). («SSL_CTX_load_verify_locations» 2018).

Por ejemplo, en las pruebas realizadas el certificado PiZero.crt correspondiente al certificado de este dispositivo utilizado en el servidor se ha convertido en 24647dc9.0.

28a88d58.0 se corresponde con el certificado PiLego.crt.

9b682844.0 con el PiActuador.crt.

b6b777b0.0 para PiCompleta.crt.

11.3 Configuración de Visual Studio

Para configurar Visual Studio y poder gestionar y trabajar con el sistema diseñado es necesario realizar una serie de configuraciones previas.

Primero es necesario instalar la versión 3.2 de OpenCV. La versión estándar no incluye las librerías *contrib* («opencv_contrib» 2018) que deben ser descargadas e incluidas de forma independiente. Es suficiente con seguir los pasos que se indican en la web de OpenCV para compilar la solución incluyendo estas librerías, la vía más sencilla es usar Cmake («CMake» 2018).

La instalación de OpenSSL es sencilla. Basta con descargar de su web («OpenSSL» 2018) la versión correspondiente e instalarla.

Hay que descargar la versión de SQLite amalgamation, que incluye todo el sistema en dos archivos, uno .cpp y otro .h y que basta con incluirlos dentro del proyecto. Es necesario excluir estos archivos del encabezado precompilado (stdafx.h)

En Propiedades de configuración -> C/C++ -> Encabezado precompilado, se establece la opción "Encabezado precompilado" a "No utilizar encabezados precompilados.".

El siguiente paso es configurar el proyecto, para ello será necesario realizar las siguientes configuraciones, en propiedades del proyecto (clic con el botón derecho sobre el nombre del proyecto):

- En directorios de VC++ -> Archivos ejecutables:
  - C:\openCV3-64\build\install\x64\vc15
  - C:\OpenSSL-Win64\include
- En directorios de VC++ -> En Archivos de inclusión:
  - C:\OpenSSL-Win64\include\openssl
- En directorios de VC++ -> Archivos de bibliotecas:

    - C:\openCV3-64\build\install\x64\vc15\lib
- En C/C++ -> Directorios adicionales:
    - C:\openCV3-64\build\install\include
- Vinculador -> Entrada -> Dependencias adicionales (para la versión final)
    - opencv_core320.lib
    - opencv_face320.lib
    - opencv_highgui320.lib
    - opencv_imgproc320.lib
    - opencv_videoio320.lib
    - opencv_videostab320.lib
    - opencv_aruco320.lib
    - opencv_bgsegm320.lib
    - opencv_bioinspired320.lib
    - opencv_calib3d320.lib
    - opencv_ccalib320.lib
    - opencv_datasets320.lib
    - opencv_dnn320.lib
    - opencv_objdetect320.lib
    - opencv_imgcodecs320.lib
    - opencv_video320.lib
- Vinculador -> Entrada -> Dependencias adicionales (para la versión depuración):
    - opencv_core320d.lib
    - opencv_face320d.lib
    - opencv_highgui320d.lib
    - opencv_imgproc320d.lib
    - opencv_videoio320d.lib
    - opencv_videostab320d.lib
    - opencv_aruco320d.lib
    - opencv_bgsegm320d.lib
    - opencv_bioinspired320d.lib
    - opencv_calib3d320d.lib
    - opencv_ccalib320d.lib
    - opencv_datasets320d.lib
    - opencv_dnn320d.lib
    - opencv_objdetect320d.lib
    - opencv_imgcodecs320d.lib

- opencv_video320d.lib

Habría que modificar las rutas para utilizar aquellas en las que se hayan instalado OpenCV y OpenSSL.

11.4 XXTEA

Para poder utilizar XXTEA en los capturadores y en el actuador es necesario instalar en el sistema Raspbian la librería correspondiente. Se puede hacer mediante:

pip install xxtea –U

Para el servidor y las placas Arduino la versión utilizada («Mini algoritmo de cifrado XXTEA para arduino | Heli» 2018) ha debido ser modificada para ajustarla a las necesidades del sistema. Para simplificar su uso se han diseñado las funciones Cifrador y Descifrador. También ha sido necesario crear la función textFromHexString que se se encarga de transformar el código hexadecimal recibido a un string estándar. Similares cambios se han realizado para el servidor, donde se utilizan las funciones EnviaCifrado y RecibeCifrado, además de la función de apoyo hex_a_string.

11.5 Notas

- Se deber poner especial atención con las diferentes versiones de placas de Arduino, cambia el nombre del puerto utilizado en Linux en función del modelo, por lo que es necesario adaptar el código en función del modelo.
- Para generar el instalador del programa del servidor se ha utilizado Microsoft Visual Studio 2017 Installer Project («Microsoft Visual Studio 2017 Installer Projects» 2017). Es necesario incluir las librerías utilizadas por OpenCV y OpenSSL.
- La estructura para las imágenes almacenadas será de un directorio por usuario y dentro las imágenes en formato pgm. La imagen 1.pgm se corresponderá con la imagen a mostrar por defecto. El nombre de las carpetas será una ‘s’ seguido del número de usuario. Se pueden añadir imágenes a las carpetas o eliminarlas sin problema, no existe registro de las que hay.

12 ANEXO III. RESULTADOS DE LAS PRUEBAS

En este apartado se incluyen los resultados obtenidos en las pruebas realizadas y que no se han incluido en la memoria.

12.1 Pruebas de seguridad realizadas con imágenes

En las pruebas con **LBPH** las capturas son las que aparecen en la Figura 80.

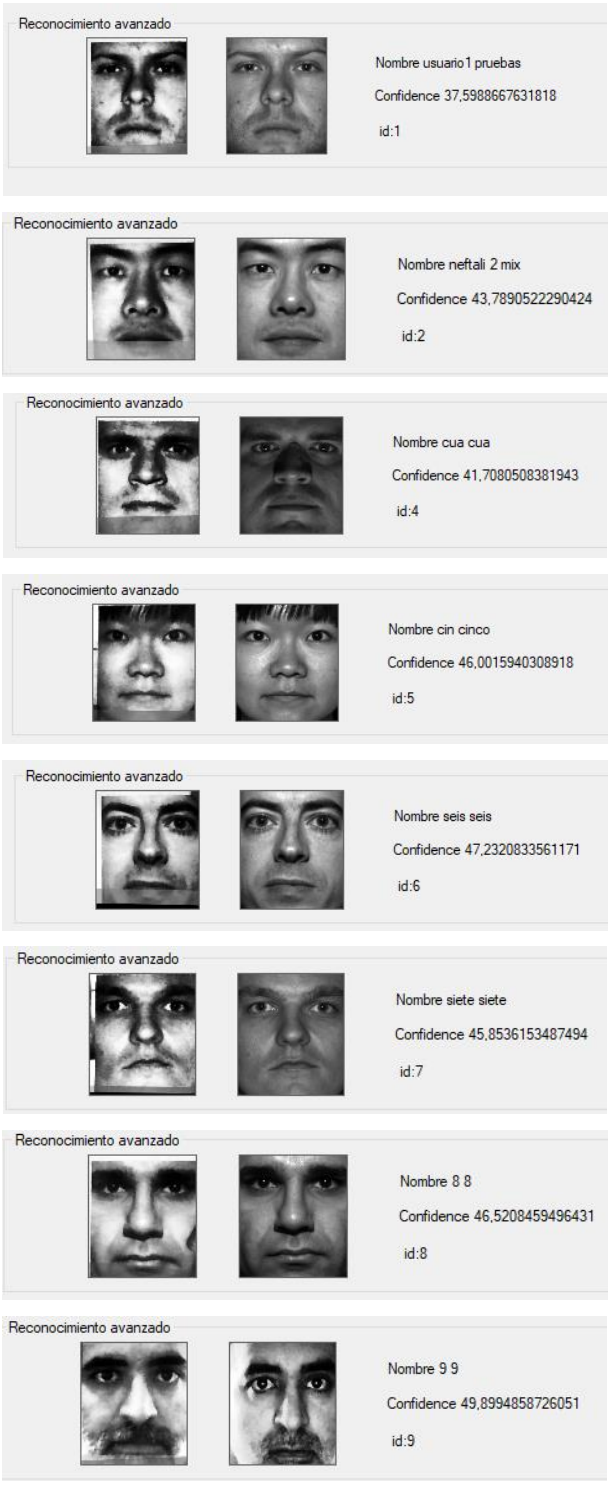

Figura 80 Verdaderos positivos en LBPH

Y los falsos positivos en la Figura 81.

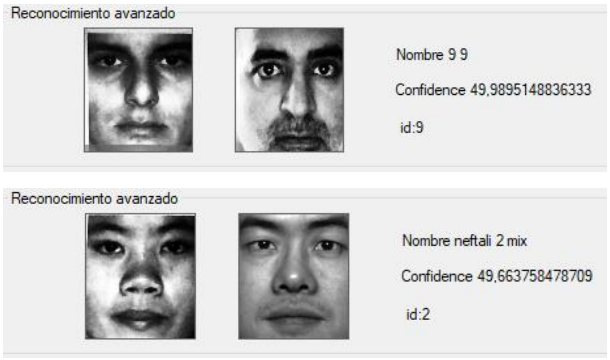

Figura 81 Falsos positivos en LBPH

En las pruebas **Eigenfaces** tenemos los resultados en la Figura 82.

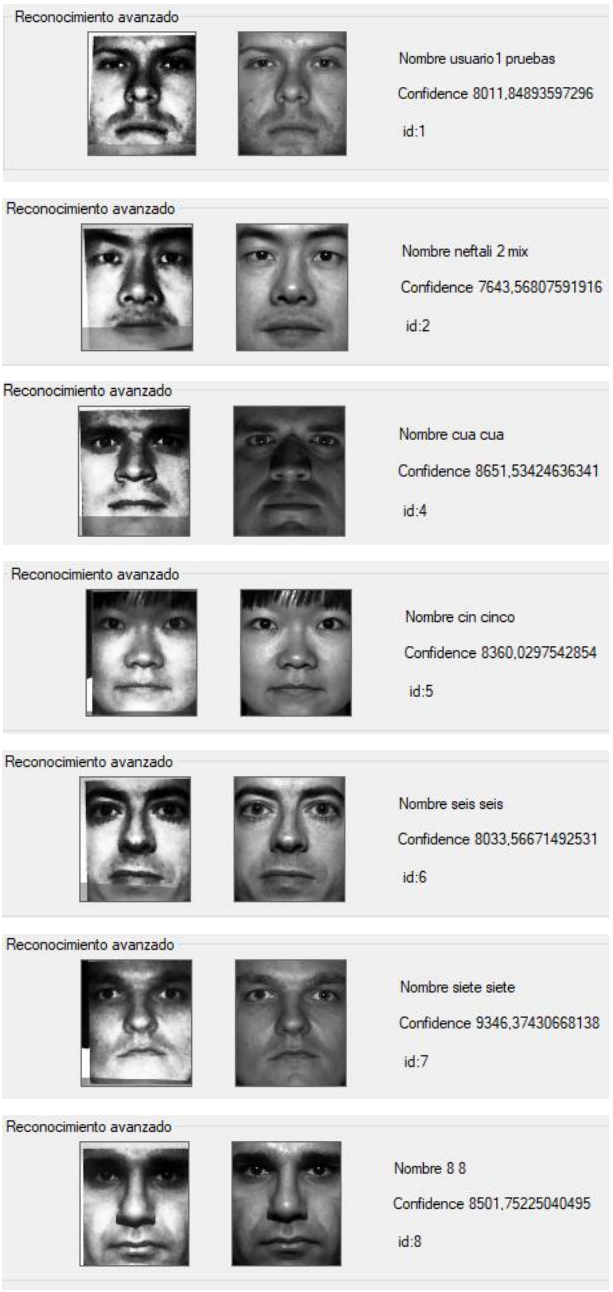

Figura 82 Verdaderos positivos en Eigenfaces

Y los falsos positivos en la Figura 83.

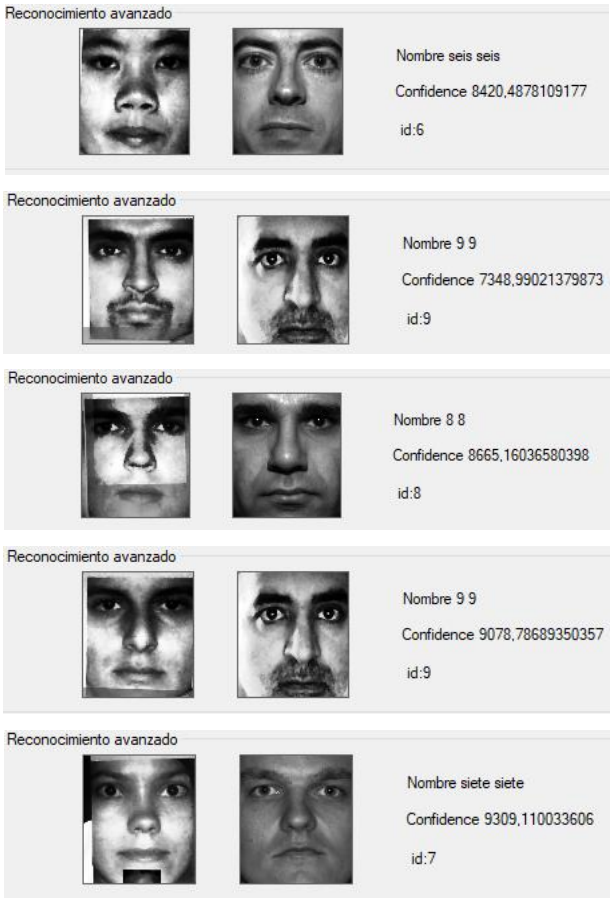

Figura 83 Falsos positivos en Eigenfaces

Y, por último, para **Fisherfaces** en la Figura 84.

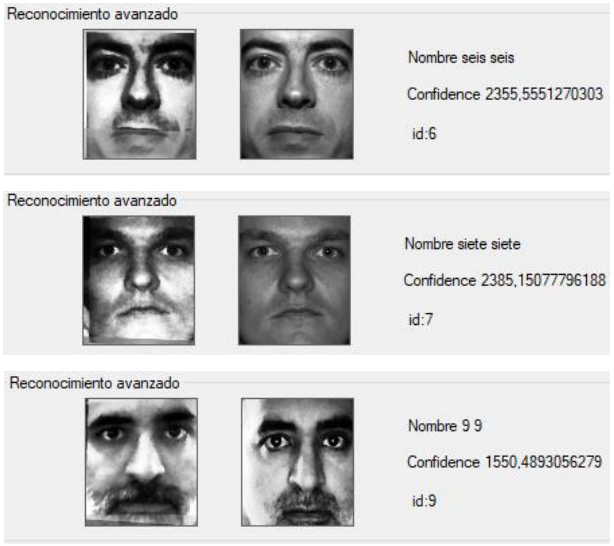

Figura 84 Verdaderos positivos en Fisherfaces

Y el falso positivo en la Figura 85.

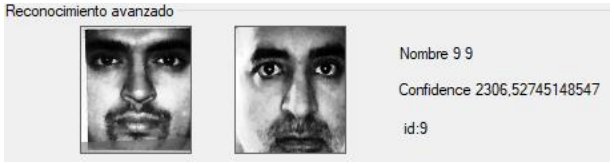

Figura 85 Falso positivo en Fisherfaces

12.2 Pruebas realizadas con imágenes en un entorno controlado

Dentro del capítulo de pruebas se ha incluido un apartado, 6.4.2, en el que se han realizado diferentes pruebas de reconocimiento con las imágenes de la base de datos de Yale. Se incluyen aquí los resultados completos obtenidos en estas pruebas.

Para las pruebas de reconocimiento se han utilizado aquellas imágenes de los usuarios registrados con suficiente calidad y que no habían sido utilizadas en el entrenamiento.

Se han utilizado ocho usuarios de la base de datos, como usuario 9 se han utilizado fotos del autor, dejando una imagen, como test inicial, para las pruebas.

En las tablas siguientes se muestran los resultados obtenidos en las pruebas teniendo que:

- Usuario: se corresponde con el id en la base de datos del usuario registrado.
- Total: número de imágenes del usuario en cuestión a las que se les ha realizado el test para el reconocimiento.
- Correctas: número de imágenes reconocidas correctamente.
- Error: Número de imágenes reconocidas erróneamente

- S/R: Imágenes cuyo usuario no ha superado el umbral y no ha sido reconocido.

Usuario	Total	Correctas	Error	S/R
9	1	1	0	0
1	8	8	0	0
2	8	7	1	0
3	9	7	2	0
4	8	8	0	0
5	5	5	0	0
6	7	6	1	0
7	4	4	0	0
8	7	6	1	0
Total	58	53	5	0

Tabla 10 LBPH. "Buenas" con valor de umbral 50

Usuario	Total	Correctas	Error	S/R
9	1	1	0	0
1	8	8	0	0
2	8	8	0	0
3	9	8	1	0
4	8	8	0	0
5	5	5	0	0
6	7	6	1	0
7	4	4	0	0
8	7	6	1	0
Total	58	55	3	0

Tabla 11 Eigenfaces. "Buenas" y valor de umbral de 10.000

Usuario	Total	Correctas	Error	S/R
1	46	13	26	7
2	45	26	11	8
3	41	23	12	6
4	43	32	4	7
5	43	35	0	8

6	43	14	27	2
7	42	28	7	7
8	42	20	19	3
Total	345	191	106	48

Tabla 12 Eigenfaces con umbral 10.000 Grupo de imágenes "malas"

Usuario	Total	Correctas	Error	S/R
9	1	1	0	0
1	8	8	0	0
2	8	7	1	0
3	9	7	2	0
4	8	8	0	0
5	5	5	0	0
6	7	7	0	0
7	4	4	0	0
8	7	6	1	0
Total	58	54	4	0

Tabla 13 Fisherfaces con umbral 1.300 Grupo de imágenes "buenas"

Usuario	Total	Correctas	Error	S/R
1	46	5	0	41
2	45	6	0	39
3	41	2	0	39
4	43	9	0	34
5	43	11	0	42
6	43	5	0	38
7	42	9	0	43
8	42	2	0	40
Total	345	49	0	296

Tabla 14 Fisherfaces con umbral 1.300 Grupo de imágenes "malas"

Usuario	Total	Correctas	Error	S/R
1	46	38	8	0
2	45	38	7	0
3	41	32	9	0

4	43	36	7	0
5	43	43	0	0
6	43	26	17	0
7	42	35	7	0
8	42	33	9	0
Total	345	281	64	0

Tabla 15 Fisherfaces con umbral 5000 Grupo de imágenes "malas"

Usuario	Total	Correctas	Error	S/R
1	46	22	2	22
2	45	26	2	17
3	41	21	1	19
4	43	30	0	13
5	43	33	0	10
6	43	10	8	25
7	42	25	0	17
8	42	15	1	26
Total	345	182	14	149

Tabla 16 Fisherfaces con umbral 2.500 Grupo de imágenes "malas"

13 ANEXO IV. BASES DE DATOS UTILIZADAS

Existen múltiples («Face Recognition Homepage - Databases» 2018) bases de datos de rostros que pueden ser usadas para testear software de reconocimiento facial. Dos de las más populares, ORL y Yale, han sido las utilizadas para testear el software desarrollado.

13.1.1 ORL Face Database

La base de datos de caras de AT&T («ORL. The Database of Faces» 2015), formalmente Base de datos de caras ORL, contiene imágenes del rostro de 40 sujetos, en escala de grises y de un tamaño de 92x112. El formato de las imágenes es PMG.

Esta ha sido la base de datos utilizadas en las primeras pruebas del sistema. El único inconveniente encontrado ha sido el tamaño de las mismas, algo menor del que se ha considerado necesario para este proyecto, por lo que se ha requerido la modificación del tamaño de las mismas.

13.1.2 Extended Yale Cropped B

La base de datos conocida como Yale Face DB B (Georghiades, Belhumeur y Kriegman 2001) es ampliamente utilizada en estudios sobre reconocimiento facial. Existe una versión con las caras recortadas (Lee, Ho y Kriegman 2005) («Extended Yale Face Database B (B+) | vision.ucsd.edu» 2005) que contiene 2547 imágenes de 38 personas diferentes en las que se van modificando las condiciones de iluminación. El tamaño de las imágenes de la versión recortada es de 168x192, coincide con el tamaño utilizado en el presente trabajo por lo que han sido muy útiles para realizar pruebas.

13.1.3 Ubiris

La base de datos Ubiris («UBIRIS» 2015) contienen imágenes de iris y están disponibles para descargarlas y utilizarlas con la autorización de Hugo Proença. Se ha conseguido la autorización y la base de datos, pero al descartar en una primera fase el uso del iris apenas han sido utilizadas.

13.1.4 Seas

La base de datos Seas («SEAS Face Recognition Database» 2016) es una pequeña base de datos con imágenes de cinco usuarios. Pero viene acompañado de pequeños videos que se han utilizado para ver los resultados para detección y reconocimiento facial en video.

14 ANEXO V. SOFTWARE ENTREGADO

El software que acompaña al presente trabajo está estructurado en directorios en función del dispositivo en el que se ejecutará:

- Servidor. Todo lo necesario para poner en marcha el servidor. Se incluye el proyecto completo diseñado en Visual Studio y el instalador obtenido.
- Capturador. Incluye todos los archivos necesarios para poner en marcha las diferentes versiones de capturadores. Se incluyen, además, otros archivos utilizados en las pruebas realizadas al sistema.
- Actuador. Con lo necesario para preparar un dispositivo como el diseñado.
- Arduino. Se incluyen las diferentes versiones a instalar en las placas Arduino. en función del dispositivo o los sensores que vaya a controlar.
- Memoria. Se incluye los documentos correspondientes a la memoria de este proyecto.
- Pruebas. Dentro de esta carpeta se incluyen algunos de los resultados obtenidos en las pruebas realizadas. Incluido el software utilizado para la recomposición de las imágenes y los videos capturados por Wireshark.

14.1 Servidor

Contiene el proyecto completo de Visual Studio utilizado para el servidor, en la carpeta TFG. Además, en la carpeta Instalador, se encuentran los archivos que permiten instalar el servidor en cualquier equipo. La versión generada es exclusivamente para 64 bits.

14.2 Capturador

En este directorio se pueden encontrar los archivos que deben incluirse en las Raspberry Pi de los capturadores, y algunos archivos que se han utilizado para las pruebas realizadas:

- **1capturador.py**. Para utilizar en el prototipo A o en el prototipo F. Cifra las comunicaciones con TLSv2.1.
- **1capturadorSin.py**. Igual al anterior, pero sin cifrar las comunicaciones.
- **2Huella.py.** Para el prototipo B, que incluye lector de huellas.
- **2HuellaGestionv2.py**. Necesario en el prototipo B para dar acceso al servidor al lector para crear y borrar huellas.
- **3Teclado.py**. Para el prototipo C. Incluye cifrado de comunicaciones total.
- **4SensorMovimiento.py**. Se usa en el prototipo D, que dispone de sensor de movimiento.

- **5SensorDistanciav2.py**. Para trabajar con el prototipo E, que incluye sensor de distancia.
- **7PiCompleta.py.** Proporciona todas las funcionalidades al prototipo G.
- **Bloqueos.py**. Script auxiliar para controlar el acceso no concurrente a la cámara. Se utiliza cuando el capturador incorpora la posibilidad de emitir en directo.
- **ControlTemperatura.py**. Utilizado para las pruebas, permite almacenar en un archivo *csv* datos relativos a la temperatura de la Raspberry Pi.
- **ControlTiempo.py**. Utilizado para las pruebas, permite almacenar en un archivo *csv* datos relativos al tiempo empleado por la Raspberry en procesos de detección facial.
- **Enviadbseas.py**. Envía las imágenes contenidas en una de las carpetas de la base de datos *dbseas* al servidor. Estas imágenes se encuentran en formato jpg y su tamaño debe modificarse antes de ser enviadas.
- **EnviaPruebas.py**. Utilizado en la fase de desarrollo del software. Recorre un directorio y envía las imágenes incluidas en él al servidor. Las imágenes deben estar en formato pgm y tener un tamaño de 192x168.
- **ServidorSeguro.py**. Se utiliza en los capturadores para permitir que el servidor acceda a sus cámaras y realizar una emisión en directo con las capturas. Es importante utilizar bloqueos.py para evitar el acceso concurrente a la cámara y el consecuente error.
- **Servidorv2.py**. Similar al anterior, pero sin cifrar las comunicaciones. Útil para pruebas de seguridad.
- **shape_predictor_68_face_landmarks.dat**. Utilizado para localizar landmarks y poder realizar el control de parpadeo o mostrar los landmarks por pantalla.
- **haarcascade_frontalface_default.xml**. Contiene los datos de un sistema entrenado para detección facial.
- **Bloqueo.py**. Se encarga de proporcionar control de concurrencia para la cámara del dispositivo.
- **Cámara.txt**. Se utiliza para simular la cámara y dar control de concurrencia en el acceso a la misma.
- **Landmark.py**. Muestra por pantalla los *landmarks* de un rostro detectado.
- **Histograma.py**. Permite obtener el histograma de una imagen y de la imagen normalizada gracias al histograma.

Además de las subcarpetas:

- PgmMalas. Contiene las imágenes de la base de datos de Yale utilizadas para las pruebas, y categorizadas como “malas”.
- Pgmtests. En este caso contiene las imágenes utilizadas como “buenas” en las mismas pruebas.
- Dbseas. Contiene las imágenes de esta base de datos. No se incluyen los videos.

14.3 Actuador

- **Actuador.py**. Para el prototipo H, no incluye cifrado de comunicaciones con el servidor ni con la placa Arduino.
- **ActuadorCifrado.py**. Incorpora al anterior el cifrado en las comunicaciones con Arduino, utilizando XXTEA.
- **ActuadorCifradov2.py**. Añade al anterior el cifrado en las comunicaciones con el servidor

14.4 Arduino

En este directorio se pueden encontrar las siguientes subcarpetas, que contienen los correspondientes archivos con extensión “ino” para Arduino:

- **Fingerprintv4**. Para trabajar con el lector de huellas.
- **KeypadLed**. Necesario para controlar el teclado matricial.
- **SensorMovimientoLCD**. Responsable de controlar el sensor de movimientos.
- **SensorDistanciaOLED**. Similar al anterior, pero para el sensor de distancia.
- **PiCompleta**. Para utilizar en la placa Arduino nano incluida en el prototipo G. En este modelo no se han cifrado las comunicaciones para no cargar en exceso la versión nano, y porque ambas placas se encuentran en el mismo dispositivo
- **Actuadorv2**. Para usar con Actuador. Esta versión no utiliza cifrado.
- **Actuadorv2XXTEA**. Similar a la anterior incorporando cifrado con XXTEA.

14.5 Relación de software entre dispositivos

Prototipo	Raspberry Pi (*.py)	Arduino (*.ino)	TLS	XXTEA
A Raspberry Pi B3	1capturadorSin	-	-	-
F Raspberry Pi Zero	1capturador	-	X	-
B con lector de huellas	2Huella	Fingerprintv4	X	X

C con teclado numérico	3Teclado	KeypadLed	X	X
D con sensor de movimiento	4SensorMovimiento	SensorMovimientoLCD	X	X
E con sensor de distancia	5SensorDistanciav2	SensorDistanciaOLED	X	X
G completa	7PiCompleta	PiCompleta	X	-
H Actuador	Actuador	Actuadorv2	-	-
	ActuadorCifrado	Actuadorv2XXTEA	-	X
	ActuadorCifradov2		X	X

15 REFERENCIAS BIBLIOGRÁFICAS

300 Faces In-the-Wild Challenge. [en línea], 2013. [Consulta: 25 mayo 2018]. Disponible en: https://ibug.doc.ic.ac.uk/resources/300-W/.

acodigo.blogspot.com. *OpenCV* [en línea], 2017. [Consulta: 9 mayo 2018]. Disponible en: http://acodigo.blogspot.com.es/p/tutorial-opencv.html.

Adafruit. [en línea], 2018. [Consulta: 21 mayo 2018]. Disponible en: https://www.adafruit.com/.

Adafruit Optical Fingerprint Sensor. [en línea], 2018. [Consulta: 22 mayo 2018]. Disponible en: https://learn.adafruit.com/adafruit-optical-fingerprint-sensor/downloads.

Aeropuertos parisinos instalan sistemas de reconocimiento facial. [en línea], 2018. [Consulta: 8 julio 2018]. Disponible en: https://www.elpais.com.uy/vida-actual/aeropuertos-parisinos-instalan-sistemas-reconocimiento-facial.html.

AForge.NET. [en línea], 2017. [Consulta: 7 mayo 2018]. Disponible en: http://www.aforgenet.com/.

AHONEN, T., HADID, A. y PIETIKAINEN, M., 2006. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 12, pp. 2037–2041.

ALVARADO MORENO, J., 2012. Análisis de textura en imágenes a escala de grises, utilizando patrones locales binarios (LBP). *ENGI REVISTA ELECTRÓNICA DE LA FACULTAD DE INGENIERÍA*, vol. 1, pp. 2012.

Amazon.es. [en línea], 2018. [Consulta: 8 julio 2018]. Disponible en: https://www.amazon.es/.

AMOS, B., LUDWICZUK, B. y SATYANARAYANAN, M., 2016. Openface: A general-purpose face recognition library with mobile applications. *CMU School of Computer Science*,

ANAYA-ISAZA, A.J., PELUFFO-ORDOÑEZ, D.H., ALVARADO-PÉREZ, J.C., IVAN-RIOS, J., CASTRO-SILVA, J.A., ROSERO-MONTALVO, P.D., PEÑA-UNIGARRO, D.F. y UMAQUINGA-CRIOLLO, A.C., 2017. Estudio comparativo de métodos espectrales para reducción de la dimensionalidad: LDA versus PCA. *INCISCOS 2016*. S.l.: s.n.,

Arduino - Software. [en línea], 2018. [Consulta: 26 mayo 2018]. Disponible en: https://www.arduino.cc/en/Main/Software.

Arduino Cryptography Library: Arduino Cryptography Library. [en línea], 2018. [Consulta: 15 mayo 2018]. Disponible en: https://rweather.github.io/arduinolibs/crypto.html.

AVR-Crypto-Lib/en – LaborWiki. [en línea], 2018. [Consulta: 14 mayo 2018]. Disponible en: https://wiki.das-labor.org/w/AVR-Crypto-Lib/en.

AZUETO-RÍOS, Á., SANTIAGO-GODOY, R., HERNÁNDEZ-GÓMEZ, L. y HERNÁNDEZ-SANTIAGO, K., 2017. Implementación de un sistema de imagenología infrarroja para la detección vascular del antebrazo y mano. *Revista Mexicana de Ingeniería Biomédica*, vol. 38, no. 2.

Azure. [en línea], 2018. [Consulta: 8 julio 2018]. Disponible en: https://azure.microsoft.com/es-es/free/search/.

BADAWI, A.M., 2006. Hand Vein Biometric Verification Prototype: A Testing Performance and Patterns Similarity. *IPCV*, vol. 14, pp. 3–9.

BAGGIO, D.L., 2012. *Mastering OpenCV with Practical Computer Vision Projects*. S.l.: Packt Publishing Ltd. ISBN 978-1-84951-783-6.

BELHUMEUR, P.N., HESPANHA, J.P. y KRIEGMAN, D.J., 1997. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711-720. ISSN 0162-8828. DOI 10.1109/34.598228.

Biometric identification on a picture of veins of a palm. [en línea], 2018. [Consulta: 22 mayo 2018]. Disponible en: http://developers-club.com/posts/149424/.

Biometric System Laboratory. [en línea], 2018a. [Consulta: 7 mayo 2018]. Disponible en: http://biolab.csr.unibo.it/research.asp?organize=Activities&select=&selObj=20&pathSubj=333%7C%7C20&Req=&.

Biometric System Laboratory. [en línea], 2018b. [Consulta: 21 mayo 2018]. Disponible en: http://biolab.csr.unibo.it/research.asp?organize=Activities&select=&selObj=82&pathSubj=111%7C%7C8%7C%7C82&Req=&.

Biometrics Evaluation and Testing (BEAT). *BEAT* [en línea], 2018. [Consulta: 24 mayo 2018]. Disponible en: https://www.beat-eu.org.

BOOM, B., BEUMER, G., SPREEUWERS, L.J. y VELDHUIS, R.N., 2006. The effect of image resolution on the performance of a face recognition system. *Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on*. S.l.: IEEE, pp. 1–6.

BOWYER, K.W. y BURGE, M.J., 2016. *Handbook of Iris Recognition*. S.l.: Springer. ISBN 978-1-4471-6784-6.

CAI, D., HE, X. y HAN, J., 2005. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 12, pp. 1624-1637. ISSN 1041-4347. DOI 10.1109/TKDE.2005.198.

CAI, D., HE, X., HAN, J. y ZHANG, H.-J., 2006. Orthogonal laplacianfaces for face recognition. *IEEE transactions on image processing*, vol. 15, no. 11, pp. 3608–3614.

CALVO, G.G. y JOSHI, P., 2018. *OpenCV 3.x with Python By Example: Make the most of OpenCV and Python to build applications for object recognition and augmented reality, 2nd Edition*. S.l.: Packt Publishing Ltd. ISBN 978-1-78839-676-9.

CANNY, J., 1987. A computational approach to edge detection. *Readings in Computer Vision*. S.l.: Elsevier, pp. 184–203.

CAPPELLI, R., FERRARA, M. y MALTONI, D., 2010. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2128–2141.

China's Minority Report-style. *Mail Online* [en línea], 2017. [Consulta: 13 agosto 2018]. Disponible en: http://www.dailymail.co.uk/sciencetech/article-5170167/China-unveils-Minority-Report-style-AI-security-system.html.

CHOI, H.-S., 2001. *Apparatus and method for identifying individuals through their subcutaneous vein patterns and integrated system using said apparatus and method*. S.l.: Google Patents.

CMake. [en línea], 2018. [Consulta: 18 agosto 2018]. Disponible en: https://cmake.org/.

CORTES, C. y VAPNIK, V., 1995. Support-vector networks. *Machine learning*, vol. 20, no. 3, pp. 273–297.

COX, I.J., GHOSN, J. y YIANILOS, P.N., 1996. Feature-based face recognition using mixture-distance. *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*. S.l.: IEEE, pp. 209–216.

Credence Trident. *Innovatrics Online Store* [en línea], 2018. [Consulta: 9 julio 2018]. Disponible en: https://shop.innovatrics.com/products/credence-trident.

Cropped the Extended Yale Face. *ResearchGate* [en línea], 2018. [Consulta: 22 mayo 2018]. Disponible en: https://www.researchgate.net/figure/Cropped-example-face-images-from-the-Extended-Yale-Face-Database-B-database-under_fig11_266022687.

DAEMEN, J. y RIJMEN, V., 1999. AES proposal: Rijndael. ,

DALAL, N. y TRIGGS, B., 2005. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. S.l.: IEEE, pp. 886–893.

DAUGMAN, J., 2007. New methods in iris recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 5, pp. 1167–1175.

DAUGMAN, J., 2009. How iris recognition works. *The essential guide to image processing*. S.l.: Elsevier, pp. 715–739.

Deepsight Image Recognition SDK. *BaseApp Systems* [en línea], 2018. [Consulta: 8 julio 2018]. Disponible en: https://www.baseapp.com/deepsight-image-recognition-sdk/.

DELBIAGGIO, N., 2017. A comparison of facial recognition's algorithms. [en línea], [Consulta: 9 mayo 2018]. Disponible en: http://www.theseus.fi/handle/10024/132808.

DELGADO MOHATAR, Ó., 2011. Nuevos protocolos y esquemas de seguridad para redes ad-hoc móviles inalámbricas. [en línea], [Consulta: 15 mayo 2018]. Disponible en: https://e-archivo.uc3m.es/handle/10016/11576.

DERAWI, M.O., YANG, B. y BUSCH, C., 2011. Fingerprint recognition with embedded cameras on mobile phones. *International Conference on Security and Privacy in Mobile Information and Communication Systems*. S.l.: Springer, pp. 136–147.

DES and 3DES Encryption Library for Arduino and Raspberry Pi: DES library for Arduino and Raspberry pi. [en línea], 2018. [Consulta: 14 mayo 2018]. Disponible en: http://spaniakos.github.io/ArduinoDES/.

Detección de sueño y parpadeo. *Cursos Python desde 0 a Experto* [en línea], 2017. [Consulta: 25 mayo 2018]. Disponible en: https://www.aprenderpython.net/deteccion-sueno-parpadeo/.

dlib C++ Library. [en línea], 2018. [Consulta: 23 mayo 2018]. Disponible en: http://dlib.net/.

DOTNET-BOT, 2018. Cómo: Realizar llamadas seguras para subprocesos en controles de formularios Windows Forms. [en línea]. [Consulta: 13 mayo 2018]. Disponible en: https://docs.microsoft.com/es-es/dotnet/framework/winforms/controls/how-to-make-thread-safe-calls-to-windows-forms-controls.

Emgu CV: OpenCV in .NET (C#, VB, C++ and more). [en línea], 2018. [Consulta: 7 mayo 2018]. Disponible en: http://www.emgu.com/wiki/index.php/Main_Page.

ERDOGMUS, N. y MARCEL, S., 2013. Spoofing in 2D face recognition with 3D masks and anti-spoofing with Kinect. *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. S.l.: IEEE, pp. 1–6.

Escáner de Ojo Cross Match. *M2SYS Technology* [en línea], 2018. [Consulta: 30 mayo 2018]. Disponible en: http://m2sys.com.mx/cross-match-i-scan-2-iris-eye-scanner/.

eSpeak: Speech Synthesizer. [en línea], 2018. [Consulta: 1 agosto 2018]. Disponible en: http://espeak.sourceforge.net/.

Etcher by resin.io. [en línea], 2018. [Consulta: 9 julio 2018]. Disponible en: https://etcher.io.

eVision - Visual Search Technology. [en línea], 2018. [Consulta: 9 julio 2018]. Disponible en: http://www.evisionglobal.com/index.html.

Extended Yale Face Database B (B+) | vision.ucsd.edu. [en línea], 2005. [Consulta: 14 julio 2018]. Disponible en: http://vision.ucsd.edu/content/extended-yale-face-database-b-b.

Face API: software de reconocimiento facial | Microsoft Azure. [en línea], 2018. [Consulta: 21 mayo 2018]. Disponible en: https://azure.microsoft.com/es-es/services/cognitive-services/face/.

Face Recognition Homepage - Databases. [en línea], 2018. [Consulta: 14 julio 2018]. Disponible en: http://www.face-rec.org/databases/.

Face Recognition with OpenCV — OpenCV 2.4.13.6 documentation. [en línea], 2018. [Consulta: 10 mayo 2018]. Disponible en: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html.

Facebook researchers boost facial recognition with PIPER poselets. *Biometric Technology Today*, 2015. vol. 2015, no. 7, pp. 1-2. ISSN 0969-4765. DOI 10.1016/S0969-4765(15)30106-5.

FACEPHI, 2018. FacePhi. *Facephi* [en línea]. [Consulta: 8 julio 2018]. Disponible en: https://www.facephi.com/es.

Fast Fingerprint Identification for Large Databases | Soft Computing and Intelligent Information Systems. [en línea], 2018. [Consulta: 21 mayo 2018]. Disponible en: http://sci2s.ugr.es/ParallelMatching.

FAULDS, H., 1880. *Guide to Finger-print Identification*. S.l.: William S. Hein.

FAULDS, H., 1912. *Dactylography: or, The study of finger-prints*. S.l.: Milner & company.

FERRARA, M., FRANCO, A. y MALTONI, D., 2008. Evaluating Systems Assessing Face-Image Compliance with ICAO/ISO Standards. . S.l.: s.n., pp. 191-199. DOI 10.1007/978-3-540-89991-4_20.

Finger Prints by Francis Galton. [en línea], 1912. [Consulta: 7 mayo 2018]. Disponible en: http://galton.org/books/finger-prints/.

Fingerprint SDK 2009:Setup and Usage - Griaule Wiki. [en línea], 2018. [Consulta: 21 mayo 2018]. Disponible en: http://www.griaulebiometrics.com/wiki/index.php/Fingerprint_SDK_2009:Setup_and_Usage.

Fingerprint Verification | Biometric Identity SDK. *Crossmatch* [en línea], 2018. [Consulta: 21 mayo 2018]. Disponible en: https://www.crossmatch.com/biometric-identity-solutions/products/software/sdk/.

FISHER, R.A., 1936. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, vol. 7, no. 2, pp. 179-188. ISSN 2050-1439. DOI 10.1111/j.1469-1809.1936.tb02137.x.

FLOM, L. y SAFIR, A., 1987. *Iris recognition system*. S.l.: Google Patents.

FREUND, Y. y SCHAPIRE, R.E., 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. *European conference on computational learning theory*. S.l.: Springer, pp. 23–37.

FUENTES, H.A., 2011. Sistemas de reconocimiento basados en la imagen facial. *Avances en Sistemas e Informática*, vol. 8, no. 3, pp. 7-16. ISSN 1909-0056.

Fujitsu PalmSecure. [en línea], 2018. [Consulta: 23 mayo 2018]. Disponible en: http://www.fujitsu.com/global/solutions/business-technology/security/palmsecure/.

GALBALLY, J., MARCEL, S. y FIERREZ, J., 2014. Biometric antispoofing methods: A survey in face recognition. *IEEE Access*, vol. 2, pp. 1530–1552.

Gandalf Home Page. [en línea], 2012. [Consulta: 9 julio 2018]. Disponible en: http://gandalf-library.sourceforge.net/.

GARCÍA, G.B., SUAREZ, O.D., ARANDA, J.L.E., TERCERO, J.S., GRACIA, I.S. y ENANO, N.V., 2015. *Learning Image Processing with OpenCV*. S.l.: Packt Publishing Ltd. ISBN 978-1-78328-766-6.

GEORGHIADES, A.S., BELHUMEUR, P.N. y KRIEGMAN, D.J., 2001. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 6, pp. 643–660.

GIVENS, G., BEVERIDGE, J.R., DRAPER, B.A., GROTHER, P. y PHILLIPS, P.J., 2004. How features of the human face affect recognition: a statistical comparison of three face recognition algorithms. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. S.l.: IEEE, pp. II–II.

GOLDSTEIN, A.J., HARMON, L.D. y LESK, A.B., 1971. Identification of human faces. *Proceedings of the IEEE*, vol. 59, no. 5, pp. 748–760.

GÓMEZ, T. y MAYUMI, C., 2016. Análisis comparativo de los algoritmos Fisherfaces y LBPH para el reconocimiento facial en diferentes condiciones de iluminación y pose, Tacna–2015. ,

GONZALEZ-SOSA, E., VERA-RODRIGUEZ, R., FIERREZ, J. y ORTEGA-GARCIA, J., 2016. Validación Experimental de la Influencia de Oclusiones en Reconocimiento Facial. *In Proc. of The XXXI Simposium Nacional de la Unión Científica Internacional de Radio (URSI)*. S.l.: s.n.,

GOTTUMUKKAL, R. y ASARI, V.K., 2004. An improved face recognition technique based on modular PCA approach. *Pattern Recognition Letters*, vol. 25, no. 4, pp. 429-436. ISSN 01678655. DOI 10.1016/j.patrec.2003.11.005.

Griaule Biometrics |. [en línea], 2018. [Consulta: 21 mayo 2018]. Disponible en: http://www.griaulebiometrics.com/new/.

Guía videovigilancia AGPD [en línea], [sin fecha]. S.l.: s.n. [Consulta: 19 mayo 2018]. Disponible en: https://www.agpd.es/portalwebAGPD/canaldocumentacion/publicaciones/common/pdfs/guia_videovigilancia.pdf.

GUTIERREZ ALEGRE, E., PAJARES, M. y ESCALERA HUESO, A. de la, 2016. *Conceptos y métodos en visión por computador*. España: s.l. ISBN 978-84-608-8933-5.

HAAR, A., 1910. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, vol. 69, no. 3, pp. 331–371.

HE, D.-C. y WANG, L., 1990. Texture unit, texture spectrum, and texture analysis. *IEEE transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 509–512.

HE, X. y NIYOGI, P., 2004. Locality Preserving Projections. , pp. 8.

HE, X., YAN, S., HU, Y., NIYOGI, P. y ZHANG, H.-J., 2005. Face recognition using laplacianfaces. *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 3, pp. 328–340.

Heartbleed, otro fallo extremadamente grave en una librería SSL. [en línea], 2016. [Consulta: 14 julio 2018]. Disponible en: https://www.genbeta.com/seguridad/heartbleed-otro-fallo-extremadamente-grave-en-una-libreria-ssl.

HENRY, S.E.R., 1900. *Classification and Uses of Finger Prints*. S.l.: George Routledge and Sons.

HERSCHEL, W.J., 1916. *The Origin of Fingerprinting (1916)*. S.l.: Kessinger Publishing. ISBN 978-1-104-66225-7.

Histogram of Oriented Gradients | Learn OpenCV. [en línea], 2018. [Consulta: 25 mayo 2018]. Disponible en: https://www.learnopencv.com/histogram-of-oriented-gradients/.

Hotel Security. *Ayonix Corporation - Face Recognition Company* [en línea], 2015. [Consulta: 21 mayo 2018]. Disponible en: http://ayonix.com/solutions/security/retail-solutions/hotel-security/.

HOTELLING, H., 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* [en línea], [Consulta: 10 mayo 2018]. ISSN 0022-0663. DOI 10.1037/h0071325. Disponible en: https://www.scienceopen.com/document?vid=d7b816f1-b9cd-41b9-b046-a9c271855c85.

HOWSE, J., 2015. *OpenCV for Secret Agents*. S.l.: Packt Publishing Ltd. ISBN 978-1-78328-738-3.

HOWSE, J., PUTTEMANS, S., HUA, Q. y SINHA, U., 2015. *OpenCV 3 Blueprints*. S.l.: Packt Publishing Ltd. ISBN 978-1-78439-142-3.

HSIEH, C.-K., LAI, S.-H. y CHEN, Y.-C., 2010. An optical flow-based approach to robust face recognition under expression variations. *IEEE transactions on image processing*, vol. 19, no. 1, pp. 233–240.

Huella Digital - Aplicaciones Tecnologicas. [en línea], 2018. [Consulta: 23 mayo 2018]. Disponible en: http://www.aplicacionestecnologicas.com/Biometria_Aplicada/Huella_Digital/index.html.

IBM Watson Developer Cloud. [en línea], 2018. [Consulta: 8 julio 2018]. Disponible en: https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/curl.html?curl.

ID, I., 2018. iData SDK. *Iris ID* [en línea]. [Consulta: 30 mayo 2018]. Disponible en: http://www.irisid.com/productssolutions/softwareproducts/idatasdk/.

Identificación y reconocimiento | Axis Communications. [en línea], 2018. [Consulta: 7 mayo 2018]. Disponible en: https://www.axis.com/es-es/learning/web-articles/identification-and-recognition.

ImageLib. [en línea], 2018. [Consulta: 9 julio 2018]. Disponible en: http://wohlberg.net/public/software/misc/imagelib/.

INTECO, 2012. Estudio sobre tecnologías biométricas aplicadas a la seguridad. . S.l.:

INTERPOL–Imágenes faciales. [en línea], 2018. [Consulta: 7 mayo 2018]. Disponible en: http://studylib.es/doc/4558186/file---interpol%E2%80%93im%C3%A1genes-faciales.

ISO/IEC 19794-5:2005 - Information technology -- Biometric data interchange formats -- Part 5: Face image data. [en línea], 2006. [Consulta: 9 mayo 2018]. Disponible en: https://www.iso.org/standard/38749.html.

ISO/IEC 19794-5:2011 - Information technology -- Biometric data interchange formats -- Part 5: Face image data. [en línea], 2012. [Consulta: 9 mayo 2018]. Disponible en: https://www.iso.org/standard/50867.html.

JAFRI, R. y ARABNIA, H.R., 2009. A survey of face recognition techniques. *Jips*, vol. 5, no. 2, pp. 41–68.

JIANG, X. y YAU, W.-Y., 2000. Fingerprint minutiae matching based on the local and global structures. *Pattern recognition, 2000. Proceedings. 15th international conference on*. S.l.: IEEE, pp. 1038–1041.

JOHNSON, R., 1991. Can iris patterns be used to identify people. *Los Alamos National Laboratory, CA, Chemical and Laser Sciences Division, Rep. LA-12331-PR*,

JONES, M. y VIOLA, P., 2003. Fast multi-view face detection. ,

KAEHLER, A. y BRADSKI, G., 2016. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. S.l.: O'Reilly Media, Inc. ISBN 978-1-4919-3800-3.

KANADE, T., 1973. *Picture Processing System by Computer Complex and Recognition of Human Faces*. S.l.: Department of Information Science, Kyoto Univ.

Keypad Library. [en línea], 2018. [Consulta: 1 junio 2018]. Disponible en: https://playground.arduino.cc/Code/Keypad.

LANDMAN, D., 2018. *AESLib: Arduino Library for AES Encryption (source based on avr-crypto-lib)* [en línea]. C. S.l.: s.n. [Consulta: 14 mayo 2018]. Disponible en: https://github.com/DavyLandman/AESLib.

Learn OpenCV (C++ / Python). [en línea], 2018. [Consulta: 9 mayo 2018]. Disponible en: http://www.learnopencv.com/.

LEE, K.-C., HO, J. y KRIEGMAN, D.J., 2005. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 5, pp. 684–698.

LI, H., LIN, Z., SHEN, X., BRANDT, J. y HUA, G., 2015. A convolutional neural network cascade for face detection. [en línea]. S.l.: IEEE, pp. 5325-5334. [Consulta: 10 mayo 2018]. ISBN 978-1-4673-6964-0. DOI 10.1109/CVPR.2015.7299170. Disponible en: http://ieeexplore.ieee.org/document/7299170/.

LI, S.Z., CHU, R., AO, M., ZHANG, L. y HE, R., 2006. Highly accurate and fast face recognition using near infrared images. *International Conference on Biometrics*. S.l.: Springer, pp. 151–158.

LI, S.Z., CHU, R., LIAO, S. y ZHANG, L., 2007. Illumination invariant face recognition using near-infrared images. *IEEE Transactions on pattern analysis and machine intelligence*, vol. 29, no. 4, pp. 627–639.

LI, S.Z., ZHANG, L., LIAO, S., ZHU, X., CHU, R., AO, M. y HE, R., 2006. A near-infrared image based face recognition system. *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*. S.l.: IEEE, pp. 455–460.

libfprint. [en línea], 2016. [Consulta: 6 julio 2018]. Disponible en: https://www.freedesktop.org/wiki/Software/fprint/libfprint/.

LIENHART, R. y MAYDT, J., 2002. An extended set of haar-like features for rapid object detection. *Image Processing. 2002. Proceedings. 2002 International Conference on*. S.l.: IEEE, pp. I–I.

MARCEL, S., 2013. BEAT–biometrics evaluation and testing. *Biometric technology today*, vol. 2013, no. 1, pp. 5–7.

MARCINIAK, T., CHMIELEWSKA, A., WEYCHAN, R., PARZYCH, M. y DABROWSKI, A., 2015. Influence of low resolution of images on reliability of face detection and recognition. *Multimedia Tools and Applications*, vol. 74, no. 12, pp. 4329-4349. ISSN 1380-7501, 1573-7721. DOI 10.1007/s11042-013-1568-8.

MARTÍNEZ, A.M. y KAK, A.C., 2001. Pca versus lda. *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 2, pp. 228–233.

Matrox Imaging Library (MIL). [en línea], 2018. [Consulta: 9 julio 2018]. Disponible en: https://www.matrox.com/imaging/en/products/software/mil/.

MEDINA-PEREZ, M.A., GARCIA-BORROTO, M., GUTIERREZ-RODRIGUEZ, A.E. y ALTAMIRANO-ROBLES, L., 2011. Robust fingerprint verification using m-triplets. *Hand-Based Biometrics (ICHB), 2011 International Conference on*. S.l.: IEEE, pp. 1–5.

MEDINA-PÉREZ, M.A., GUTIÉRREZ-RODRÍGUEZ, A. y GARCÍA-BORROTO, M., 2009. Improving fingerprint matching using an orientation-based minutia descriptor. *Iberoamerican Congress on Pattern Recognition*. S.l.: Springer, pp. 121–128.

MÉNDEZ-VÁZQUEZ, H., CHANG, L., RIZO-RODRÍGUEZ, D. y MORALES-GONZÁLEZ, A., 2012. Evaluación de la calidad de las imágenes de rostros utilizadas para la identificación de las personas. *Computación y Sistemas*, vol. 16, no. 2, pp. 147–165.

Microsoft Visual Studio 2017 Installer Projects. [en línea], 2017. [Consulta: 1 septiembre 2018]. Disponible en: https://marketplace.visualstudio.com/items?itemName=VisualStudioClient.MicrosoftVisualStudio2017InstallerProjects.

Mini algoritmo de cifrado XXTEA para arduino | Heli. [en línea], 2018. [Consulta: 15 mayo 2018]. Disponible en: http://heli.xbot.es/?p=427.

MINICHINO, J. y HOWSE, J., 2015. *Learning OpenCV 3 Computer Vision with Python*. S.l.: Packt Publishing Ltd. ISBN 978-1-78528-977-4.

NEEDHAM, R.M. y WHEELER, D.J., 1996. Tea extensions. *Notes*,

Neurotechnology. [en línea], 2018. [Consulta: 7 mayo 2018]. Disponible en: http://www.neurotechnology.com/.

New lower price for Watson Visual Recognition. *IBM Cloud Blog* [en línea], 2016. [Consulta: 8 julio 2018]. Disponible en: https://www.ibm.com/blogs/bluemix/2016/12/lower-price-watson-visual-recognition/.

NORMALIZACIÓN, O.I. de, 2006. *ISO/IEC 19794-5:2005, Information Technology - Biometric Data Interchange Formats: Technologies D L'information - Formats D'échange de Données Biométriques. Face image data. Données d'image de la face*. S.l.: ISO.

NumPy. [en línea], 2018. [Consulta: 13 junio 2018]. Disponible en: http://www.numpy.org/.

OJALA, T., PIETIKAINEN, M. y MAENPAA, T., 2002. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987.

Onvif. *ONVIF* [en línea], 2018. [Consulta: 1 agosto 2018]. Disponible en: https://www.onvif.org/.

OpenCV library. [en línea], 2018a. [Consulta: 7 mayo 2018]. Disponible en: https://opencv.org/.

OpenCV library. [en línea], 2018b. [Consulta: 21 mayo 2018]. Disponible en: https://opencv.org/.

OpenCV: tracking. [en línea], 2012. [Consulta: 26 agosto 2018]. Disponible en: https://docs.opencv.org/3.3.1/d2/dc1/camshiftdemo_8cpp-example.html.

opencv_contrib: Repository for OpenCV's extra modules [en línea], 2018. C++. S.l.: OpenCV. [Consulta: 18 agosto 2018]. Disponible en: https://github.com/opencv/opencv_contrib.

OpenFace. [en línea], 2017. [Consulta: 7 mayo 2018]. Disponible en: https://cmusatyalab.github.io/openface/.

OpenSSL. [en línea], 2018. [Consulta: 10 mayo 2018]. Disponible en: https://www.openssl.org/.

ORL. The Database of Faces. [en línea], 2015. [Consulta: 22 mayo 2018]. Disponible en: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

ÖZDIL, A. y ÖZBILEN, M.M., 2014. A survey on comparison of face recognition algorithms. *Application of Information and Communication Technologies (AICT), 2014 IEEE 8th International Conference on*. S.l.: IEEE, pp. 1–3.

Palm Vein Scanner to Secure Biometric Recognition | M2SYS. [en línea], 2018. [Consulta: 23 mayo 2018]. Disponible en: http://www.m2sys.com/palm-vein-reader/.

palmsecure.pdf. [en línea], 2018. [Consulta: 22 mayo 2018]. Disponible en: https://www.fujitsu.com/ca/en/Images/palmsecure.pdf.

PAN, G., SUN, L., WU, Z. y LAO, S., 2007. Eyeblink-based anti-spoofing in face recognition from a generic webcamera. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. S.l.: IEEE, pp. 1–8.

PAN, G., WU, Z. y SUN, L., 2008. Liveness detection for face recognition. *Recent advances in face recognition*. S.l.: InTech,

PAPAGEORGIOU, C.P., OREN, M. y POGGIO, T., 1998. A general framework for object detection. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. S.l.: s.n., pp. 555-562. DOI 10.1109/ICCV.1998.710772.

PARZIALE, G. y NIEL, A., 2004. A fingerprint matching using minutiae triangulation. *Biometric Authentication*. S.l.: Springer, pp. 241–248.

PEARSON, K., 1901. *On Lines and Planes of Closest Fit to Systems of Points in Space*. S.l.: University College.

PUTTEMANS, D.I.S., 2018. Personal Website of Steven Puttemans. *Personal Website of Steven Puttemans* [en línea]. [Consulta: 9 mayo 2018]. Disponible en: http://stevenputtemans.github.io/.

PyImageSearch - Be awesome at OpenCV, Python, deep learning, and computer vision. *PyImageSearch* [en línea], 2018. [Consulta: 9 mayo 2018]. Disponible en: https://www.pyimagesearch.com/.

QUISQUATER, J.-J. y CARDIS 1998 (LOUVAIN-LA-NEUVE, B., 2000. *Smart Card. Research and Applications: Third International Conference, CARDIS'98 Louvain-la-Neuve, Belgium, September 14-16, 1998 Proceedings*. S.l.: Springer Science & Business Media. ISBN 978-3-540-67923-3.

RAHUL, R.C., CHERIAN, M. y MOHAN, M., 2015. Literature Survey On Contactless Palm Vein Recognition. ,

RANADE, S. y ROSENFELD, A., 1978. Point pattern matching by relaxation. . S.l.: MARYLAND UNIV COLLEGE PARK COMPUTER SCIENCE CENTER.

Raspbian. *Raspberry Pi* [en línea], 2018. [Consulta: 25 mayo 2018]. Disponible en: https://www.raspberrypi.org/products/.

Reconocimiento facial | Axis Communications. [en línea], 2018. [Consulta: 21 mayo 2018]. Disponible en: https://www.axis.com/es/solutions-by-application/facial-recognition.

REEBA, Y.B. y SHANMUGALAKSHMI, R., 2015. Spoofing face recognition. *Advanced Computing and Communication Systems, 2015 International Conference on*. S.l.: IEEE, pp. 1–5.

Reglamento 2016/679 del Parlamento Europeo y del Consejo. , 2016. pp. 88.

ROSEBROCK, A., 2014. Histogram of Oriented Gradients and Object Detection. *PyImageSearch* [en línea]. [Consulta: 25 mayo 2018]. Disponible en: https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/.

ROSEBROCK, A., 2018. *imutils* [en línea]. Python. S.l.: s.n. [Consulta: 7 mayo 2018]. Disponible en: https://github.com/jrosebr1/imutils.

SANG, J., LEI, Z. y LI, S.Z., 2009. Face image quality evaluation for ISO/IEC standards 19794-5 and 29794-5. *International Conference on Biometrics*. S.l.: Springer, pp. 229–238.

SCHAPIRE, R.E. y SINGER, Y., 1999. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, vol. 37, no. 3, pp. 297–336.

SEAS Face Recognition Database. [en línea], 2016. [Consulta: 24 agosto 2018]. Disponible en: http://hellosaumil.github.io/SEAS-FR/.

SERRA, L., 2015. *crypto-arduino-library* [en línea]. C++. S.l.: s.n. [Consulta: 15 mayo 2018]. Disponible en: https://github.com/leoserra/crypto-arduino-library.

SERRANO, F.E.A., ORTEGA, J.C.P., ARAIZA, E.A.R. y ARAIZA, J.E.R., 2018. DESARROLLO DE UN PROCESO DE AUTENTICACIÓN FACIAL EN UN SISTEMA ANDROID UTILIZANDO EL ALGORITMO LDA (ANÁLISIS DE DISCRIMINACIÓN LINEAL). *Pistas Educativas*, vol. 39, no. 128.

SIROVICH, L. y KIRBY, M., 1987. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, vol. 4, no. 3, pp. 519–524. DOI 10.1364/JOSAA.4.000519.

SOUKUPOVÁ, T. y CECH, J., 2016. Real-time eye blink detection using facial landmarks. *21st Computer Vision Winter Workshop*. S.l.: s.n.,

SSL_CTX_load_verify_locations. [en línea], 2018. [Consulta: 18 agosto 2018]. Disponible en: https://www.openssl.org/docs/man1.0.2/ssl/SSL_CTX_load_verify_locations.html.

SUÁREZ PASCUAL, J.E., 2011. Mecanismos de captura y procesado de imágenes de venas para identificación personal. ,

SUDHANARANG, K.J. y MEGHASAXENA, A., 2018. Comparison of Face Recognition Algorithms Using Opencv for Attendance System. , vol. 8, no. 2, pp. 6.

TAIGMAN, Y., YANG, M., RANZATO, M. y WOLF, L., 2014. Deepface: Closing the gap to human-level performance in face verification. *Proceedings of the IEEE conference on computer vision and pattern recognition*. S.l.: s.n., pp. 1701–1708.

TAN, X. y TRIGGS, B., 2010. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE transactions on image processing*, vol. 19, no. 6, pp. 1635–1650.

THAKARE, N.M. y THAKARE, V., 2012. Biometrics standards and face image format for data interchange-a review. *International Journal of Advances in Engineering & Technology*, vol. 2, no. 1, pp. 385.

The Oxford TargetJr Page. [en línea], 2018. [Consulta: 9 julio 2018]. Disponible en: http://www.robots.ox.ac.uk/~tgtjr/.

Trident® – Credence ID. [en línea], 2018. [Consulta: 30 mayo 2018]. Disponible en: https://credenceid.com/trident/.

Trusted Biometrics under Spoofing Attacks. [en línea], 2015. [Consulta: 6 junio 2018]. Disponible en: http://www.tabularasa-euproject.org/.

TURK, M.A. y PENTLAND, A.P., 1991. Face recognition using eigenfaces. *1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Proceedings*. S.l.: s.n., pp. 586-591. DOI 10.1109/CVPR.1991.139758.

UBIRIS. [en línea], 2015. [Consulta: 1 junio 2018]. Disponible en: http://iris.di.ubi.pt/.

VAN TIEN, T., MIEN, P.T., DUNG, P.T. y LINH, H.Q., 2015. Using Near-Infrared Technique for Vein Imaging. *5th International Conference on Biomedical Engineering in Vietnam*. S.l.: Springer, pp. 190–193.

VAZQUEZ, H., CHANG, L., RIZO, D. y MORALES-GONZÁLEZ, A., 2012. Evaluación de la calidad de las imágenes de rostros utilizadas para la identificación de las personas. *Computación y Sistemas*, vol. 16, pp. 147-165.

VeriFinger SDK. [en línea], 2010. [Consulta: 21 mayo 2018]. Disponible en: http://www.neurotechnology.com/verifinger.html.

VeriLook face identification. [en línea], 2017. [Consulta: 8 julio 2018]. Disponible en: http://www.neurotechnology.com/verilook.html?gclid=CjwKCAjw7IbaBRBqEiwA6AyZgvbNONgjwipVdpS9639ROgzKpXLM2MzUIwf9JELa6ynyXhxGw_WQ9xoCuPEQAvD_BwE.

VIOLA, P. y JONES, M., 2001. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. S.l.: IEEE, pp. I–I.

WANG, X., HAN, T.X. y YAN, S., 2009. An HOG-LBP human detector with partial occlusion handling. *Computer Vision, 2009 IEEE 12th International Conference on*. S.l.: IEEE, pp. 32–39.

WEBSTER, L., 2018. Descargas | IDE, Code y Team Foundation Server. *Visual Studio* [en línea]. [Consulta: 30 mayo 2018]. Disponible en: https://www.visualstudio.com/es/downloads/.

What Is Amazon Rekognition? - Amazon Rekognition. [en línea], 2018. [Consulta: 8 julio 2018]. Disponible en: https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html.

WHEELER, D.J. y NEEDHAM, R.M., 1994. TEA, a tiny encryption algorithm. *Fast Software Encryption* [en línea]. S.l.: Springer, Berlin, Heidelberg, pp. 363-366. [Consulta: 15 mayo 2018]. ISBN 978-3-540-60590-4. DOI 10.1007/3-540-60590-8_29. Disponible en: https://link.springer.com/chapter/10.1007/3-540-60590-8_29.

WHEELER, D.J. y NEEDHAM, R.M., 1998. Correction to xtea. *Unpublished manuscript, Computer Laboratory, Cambridge University, England*,

WILDES, R.P., 1997. Iris recognition: an emerging biometric technology. *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1348–1363.

Wireshark · Go Deep. [en línea], 2018. [Consulta: 30 mayo 2018]. Disponible en: https://www.wireshark.org/.

Wiring for use with Arduino Fingerprint Sensor. [en línea], 2018. [Consulta: 21 mayo 2018]. Disponible en: https://learn.adafruit.com/adafruit-optical-fingerprint-sensor/wiring-for-use-with-arduino.

XIE, Z. y LIU, G., 2011. Infrared face recognition based on local binary pattern and pattern selection. *Journal of Computational Information Systems*, vol. 7, no. 12, pp. 4367–4374.

xxtea. *PyPI* [en línea], 2018. [Consulta: 1 junio 2018]. Disponible en: https://pypi.org/project/xxtea/.

YARRKOV, E., 2010. *Cryptanalysis of XXTEA*. S.l.: s.n.

YI, D., LIU, R., CHU, R., LEI, Z. y LI, S.Z., 2007. Face matching between near infrared and visible light images. *International Conference on Biometrics*. S.l.: Springer, pp. 523–530.

ZHANG, K., ZHANG, L. y YANG, M.-H., 2012. Real-time compressive tracking. *European conference on computer vision*. S.l.: Springer, pp. 864–877.

ZHANG, N., PALURI, M., TAIGMAN, Y., FERGUS, R. y BOURDEV, L., 2015. Beyond frontal faces: Improving person recognition using multiple cues. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. S.l.: s.n., pp. 4804–4813.

ZHANG, W., SHAN, S., GAO, W., CHEN, X. y ZHANG, H., 2005. Local Gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. S.l.: IEEE, pp. 786–791.

ZHAO, W., KRISHNASWAMY, A., CHELLAPPA, R., SWETS, D.L. y WENG, J., 1998. Discriminant analysis of principal components for face recognition. *Face Recognition*. S.l.: Springer, pp. 73–85.