ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

TESIS DE GRADO

ESTUDIO COMPARATIVO DE ALGORITMOS DE COMPRESIÓN DE IMÁGENES DE HUELLAS DIGITALES IMPLEMENTADOS EN UN DSP DE LA FAMILIA C6000

Previa la obtención del Titulo de:

INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

Presentada por:

ERWIN JACKSON JURADO ALARCÓN

GUAYAQUIL – ECUADOR

AÑO

2008

AGRADECIMIENTO

Son muchas las personas a las cuales le estoy infinitamente agradecido por logra mi sueño de ser Ingeniero, disculpen si no puedo mencionar a todos; a través de estas letras manifiesto mis más sinceros agradecimientos:

A *Díos*, por iluminar mí camino hacia un futuro mejor.

A los Drs. *Juan Antonio Ortega* y *José Luís Romeral*, por su extraordinaria dirección y orientación en la realización de esta tesis en la UPC, España.

Al *Ing. Pedro Vargas*, por su colaboración en la pasantía y en el proyecto.

A mi esposa *Maria Fernanda Zurita* por su muestras de amor y aliento en los momentos de duda y a mi madre *Rita Inés*, por todo el cariño y muestras de apoyo que mostró durante la realización de esta tesis; a ellas que estuvieron en España conmigo, con sus palabras de apoyo en los momentos en los cuales pensaba que no podría culminar el trabajo tuve las fuerzas suficientes para culminar mi carrera.

A todos ellos un infinito GRACIAS!!!!!

DEDICATORIA

Durante mi vida escolar y secundaria, siempre fue muy introvertido, la universidad me cambio la vida. Por eso a través de este espacio quiero hacer énfasis que dedico este libro en primer lugar a la ESPOL, por haber cambiado mi vida y saber que luego de salir de sus aulas soy una persona de bien al servicio de mi ciudad, Guayaquil y mi país natal, Ecuador y mi segunda patria, España.

A mis padres, Jaime y Rita porque a pesar de su pobreza, supieron inculcar en mi las ansias de superación.

A mis tías Ruth y Ana, pues desde que soy pequeño confiaron en mí, colaborando para que yo pudiera estudiar.

A Don Elías, pues en el, a pesar de no serlo, encontré al abuelo que nunca tuve.

A mi esposa, Maria Fernanda, pues ha sido la guía de mi camino en momentos de oscuridad, bien dicen que detrás de todo gran hombre hay una gran mujer.

A mis hermanos: Nacho, Jaime, Shirley, Mishell y Jaime Andrés, por su cariño innegable.

Y por ultimo quiero hacer una especial dedicatoria a mi hija Ariadna Meritxell y mi sobrino Leonardo, pues son la sangre nueva y orgullo de nuestra familia.

TRIBUNAL DE GRADO

Ing. Holger Cevallos
SUBDECANO DE LA FIEC

Ing. Pedro Vargas
DIRECTOR DE TESIS

Ing. Carlos Valdivieso
VOCAL PRINCIPAL

Ing. Ludmila Gorenkova
VOCAL PRINCIPAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la **ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**".

(Reglamento de Graduación de la ESPOL).

ERWIN JACKSON JURADO ALARCÓN

RESUMEN

Este proyecto fue desarrollado en su totalidad en España, en el Departamento de Ingeniería en Electrónica de la Universidad Politécnica de Cataluña Campus Terrassa con la colaboración del Dr. Juan Antonio Ortega Redondo, profesor de UPC, quien fue el tutor del proyecto.

Esta tesis fue planteada con la finalidad de mejorar la eficiencia de un sistema de verificación de huellas digitales, en lo que respecta a tamaño de base de datos y velocidad de transmisión de datos.

Para lograrlo se hizo un estudio comparativo entre diferente algoritmos de compresión de huellas digitales (Compresión JPEG, Compresión Hadamard y Compresión Binaria), puesto que al comprimir las imágenes de las huellas se logra disminuir, tanto el tamaño de la base de datos como su tiempo de transmisión; el estudio mencionado fue implementado dentro de un sistema de verificación de huellas (sistema FADT, tarjeta de adquisición de datos TMS320C6713DSK y sensor de huellas FPC1010).

El análisis de resultados se realizó usando parámetros de seguridad establecidos (tasa de falso rechazo y falsa aceptación; entre más bajo son estos valores, el sistema de seguridad es mas eficiente) y el nivel de compresión de los algoritmos; de los indicadores usados, los parámetros de seguridad tenían mayor prioridad al momento de dar la conclusión; en base a esto se dedujo que la Compresión Binaria

optimizaba el sistema de verificación pues obtuvo el nivel más bajo de tasa de falso rechazo y falsa aceptación.

Adicionalmente, vale recalcar que este estudio puede servir como base para la implementación de un sistema real de verificación a distancia.

ÍNDICE GENERAL

Pág.

Capítulo 2
2. ANÁLISIS TEÓRICO DE LAS TÉCNICAS DE COMPRESIÓN DE IMÁGENES

Capítulo 3
3 IMPLEMENTACIÓN DEL SISTEMA

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE ANEXOS

INTRODUCCIÓN

La presente obra trata sobre un estudio comparativo entre diferentes métodos de compresión de imágenes de huellas digitales, que serán implementados en un DSP de la familia C6000.

La tesis tiene como finalidad definir cual método de compresión permitirá mejorar la eficiencia de un sistema de verificación de huellas digitales, en lo que respecta al tamaño de la base de datos y su velocidad de transmisión.

El libro esta divido en cuatro capítulos, de los cuales, el capítulo uno y dos trata sobre la teoría necesaria para entender el proyecto. Mientras el capítulo tres y cuatro trata sobre la parte práctica del proyecto: implementación del proyecto, análisis y resultados de los datos.

El capítulo uno menciona los tipos y el esquema de funcionamiento de los sistemas de reconocimiento biométricos más populares, explica los parámetros de seguridad de un sistema biométrico y los conceptos básicos de un sistema de huella digital.

El capítulo dos presenta conceptos básicos sobre los tipos de compresiones existentes y da una breve explicación sobre cada una de las compresiones a implementar.

El capítulo tres se describe la implementación del sistema: se explica el funcionamiento del equipo usado en el proyecto (Software y Hardware), el

procedimiento seguido para implementar los métodos de compresión; y el procesado de la imagen original. En el procesado de la imagen se realiza el registro de la huella, y la compresión y descompresión de las imágenes.

En el capítulo cuatro se realiza el análisis de los datos procesados y se presenta el resultado de la verificación de las huellas, para lo cual, se verifican las huellas descomprimidas y luego se comparan, tomando como referencia el tipo de compresión utilizada. Para obtener los resultados se realiza una comparación global entre todas las huellas, tomando como referencia las tasa de falsa aceptación y falso rechazo de cada tipo de compresión.

Capítulo 1

SISTEMAS DE RECONOCIMIENTO BIOMÉTRICO

1.1 INTRODUCCIÓN

En nuestro mundo, la necesidad de buscar seguridad a nuestras posesiones ha hecho que durante muchos años se haya usado sistemas de reconocimiento que manejan cierto tipo de información, la cual se basa en el conocimiento de un dato (contraseñas o pines) o la posesión de un objeto (tarjetas).

Este tipo de sistemas no tienen un alto índice de confiabilidad, pues en el caso de las contraseñas, estas se pueden olvidar; si se copian en algún lugar, se esta expuesto a que otra persona haga uso indebido de la misma; y en el caso de las tarjetas, se esta expuesto a la pérdida o el robo de las mismas.

Todo esto trae consigo, la ineficiencia en el sistema de seguridad a la hora de indicar si la persona que desea identificarse es quien dice ser.

Por esta razón, desde estas últimas décadas, la ciencia ha aumentado sus esfuerzo en la creación de sistemas de reconocimiento que identifiquen características únicas en los seres humanos, los cuales estén relacionados a sus rasgos físicos,

obviamente estos rasgos deben ser únicos en la persona, para que el sistema de identificación se lo más eficiente y seguro posible.

La **biometría** es la disciplina que permite identificar y/o obtener rasgos de la persona basándose en sus características físicas y/o en sus pautas de comportamiento. De esta forma estas tecnologías permiten establecer una relación entre una persona y un determinado patrón asociado a ella de forma segura e intransferible.

Los sistemas de reconocimiento que se han basado en esta disciplina reciben el nombre de **Sistemas de Reconocimiento Biométrico.**

1.2 TIPOS DE SISTEMAS DE RECONOCIMIENTO

En la actualidad existe una diversidad de sistemas en el mercado, estos se los denomina por el tipo de característica biométrica que usan, los más importantes son:

- Huellas dactilares.
- Caras.
- Geometría de manos y dedos.
- Iris.
- Retina.
- Voz.
- Firma.

1.2.1 HUELLAS

En este método se capturan y digitalizan las huellas dactilares, las cuales se comparan con las almacenadas en una base de datos.

Entre las aplicaciones de este sistema se pueden mencionar:

- Acceso a aplicaciones financieras.
- Autorización de transacciones de valor elevado en bancos.
- Acceso a redes.
- Acceso a estaciones de trabajo y recursos de red.
- Verificaciones de horario.
- Registro de votación.
- Registro en organismos civiles.

Más adelante se profundizará en los conceptos sobre este método.

1.2.2 CARAS

Se basa en la captura de la imagen de la cara a través de una cámara.

En la imagen capturada se analizan diversos parámetros tales como: la distancia entre los ojos y la nariz, curvatura de los huesos, asimetrías de puntos notables, etc.

Se elimina información extraña como barba, lentes o anteojos, y sombrero.

Figura 1.1: **Sistema de reconocimiento de caras comercial.**

Finalmente, los datos resultantes se convierten en un código digital, para su almacenamiento; en caso de que se este en la etapa de registro, en la base de datos.

Figura 1.2: **Verificación de un sistema por reconocimiento de caras.**

Su uso principal se orienta más a la verificación de una persona determinada.

Sus aplicaciones más usuales son:

- Ventas por menor.
- Pago de cheques.
- Operaciones financieras como: el e-commerce y de bolsa en línea.
- Atención de salud de sistemas de seguridad social.
- Control de fronteras.

1.2.3 GEOMETRÍA DE MANOS Y DEDOS

Un sistema de este tipo permite obtener un registro tridimensional de las principales características de la mano y/o dedos tales como longitud, ancho y altura, en algunas áreas particulares, así como posiciones relativas de dedos, nudillos, etc.

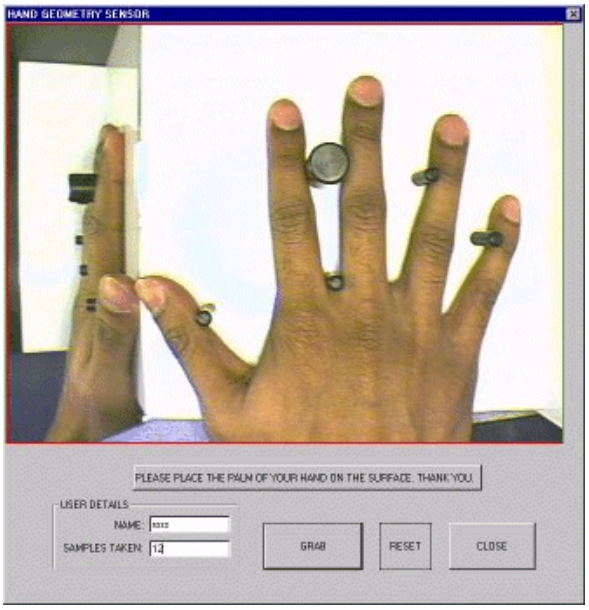


Figura 1.3: **Sistema de reconocimiento de geometría de manos.**

Con esta información, el sistema crea un mapa tridimensional del contorno de la mano que luego se convierte en un código de 9 bytes (72 bits).

Estos sistemas tienen tres variantes con su propia tecnología: geometría de manos, de un dedo y de dos dedos.

***Geometría de manos*:** Trabaja colocando la mano sobre una placa que tiene grúas para ubicar a cada uno de los dedos. La cámara toma una fotografía estableciendo hasta casi un centenar de características diferentes.

***Geometría de dedos*:** El dispositivo de captura tiene un pequeño pistón donde se ubica el dedo.

El sistema tiene en su parte interna un conjunto de pequeñas ruedas que se mueven alrededor del dedo cuando éste empuja el pistón. De esta manera se levantan mediciones de doce secciones diferentes a lo largo de unos cuatro centímetros.

Aplicaciones

- Chequeo de asistencia de personal (en combinación con un pin).
- Identificación de personas con cajeros automáticos y tarjetas de crédito.

1.2.4 IRIS Y RETINA

En ambas aplicaciones se utiliza una cámara de video para capturar los patrones de los tejidos del iris o las venas de la retina.

Reconocimiento de Iris: El iris es la franja de tejido que rodea la pupila del ojo.

Tiene una estructura compleja de estrías, anillos, surcos, coronas y flecos que ofrece prácticamente infinitas variaciones incluso entre ambos ojos de la misma persona; y, que además, permanece constante en el tiempo.

Los datos capturados se procesan por medio de complejos algoritmos matemáticos, para conformar un código de 256 bytes.

Figura 1.4: **Kit completo para reconocimiento de iris.**

Las implementaciones de este sistema responden a dos tipos: **Activos** y **Pasivos.**

Los productos **activos** requieren que el mismo usuario quede en foco con la cámara, moviéndose hacia atrás y adelante entre unos 15 a 30 cm.; mientras que los de carácter **pasivo**, consisten en un juego de cámaras que ubican automáticamente la cara y los ojos de los usuarios.

Reconocimiento de Retina: La retina es la capa más interna de la pared posterior del globo ocular, esta llena de venas.

Con una imagen adecuada se puede establecer un mapa muy preciso y único del patrón de conductos venosos.

Con este sistema se hace que un rayo de luz incandescente incida sobre la retina para retornar al escáner de captura.

1.2.5 VOZ

Este sistema se basa en el análisis de la voz, principalmente en el tono (intensidad y fuerza) y la altura (frecuencia) de los sonidos; generalmente por medio del análisis eléctrico de la densidad de la energía y formas de onda (por componentes armónicos), así como las inflexiones en el hablar y el propio comportamiento lingüístico.

Figura 1.5: **Uso del sistema de identificación por voz.**

Para el registro inicial generalmente se repite varias veces una misma frase y/o una serie de frases. De esta manera se establece el patrón individual contra el cual se compara cada vez que se recurre al sistema para verificación del usuario.

Aplicaciones

- Seguridad en el acceso telefónico
- Verificación de acceso a correo de voz
- Activación de tarjetas de crédito

- Confirmar la identidad de personas en libertad condicional con reclusión domiciliaria.

1.2.6 FIRMAS

Se basa en el uso de un sistema DSV (Verificación Dinámica de Firmas), el cual, pondera y mide características de la firma, como: la presión que se ejerce sobre la lapicera en diferentes partes, puntos en donde la lapicera se separa del papel, orden, velocidad y aceleración.

Figura 1.6: **Identificación de usuaria a través de la firma.**

La representación matemática de todas las características se guarda adecuadamente codificada para ser tomada como base cada vez que se solicite.

Aplicaciones

- Medio de verificación en sistemas bancarios.
- Autentificación de documentos electrónicos.

1.2.7 COMPARACIÓN ENTRE SISTEMAS DE RECONOCIMIENTO

Ventajas

Huellas dactilares:

Puede ser usado como sistema de verificación e identificación. No acepta ni siquiera una cinta donde se haya hecho la impresión de una huella.

Caras:

No es necesario que el usuario este muy cerca de la cámara, hasta se puede registrar directamente de una foto. Apto para aplicaciones de verificación.

Geometría de manos y dedos:

Es muy fácil de usar.

Iris y Retina:

Son los dispositivos más seguros entre los métodos biométricos, gracias a que los patrones individuales son únicos en cada persona y la calidad de los dispositivos de captura.

Apto para aplicaciones de verificación.

Voz:

Es el sistema más accesible a causa de su bajo costo y no ofrece muchas molestias al usuario en el momento del reconocimiento.

Firma:

Goza de gran aceptación, en especial en medios bancarios, por su parecido al método tradicional de verificación de firmas.

Desventajas

Huellas dactilares:

Pese a la exactitud del método, su eficacia puede ser mermada por suciedad, quemadura, grasa, cicatrices, etc. que deforman la imagen lo suficiente como para verse diferente a la original y provocar un falso rechazo.

Caras:

Requiere actualizaciones periódicas, pues las caras cambian con el tiempo, lo cual es una molestia para el usuario. Además, es costoso y puede ser sujeto de engaño.

Geometría de manos y dedos:

El sistema queda afectado cuando el usuario ha sufrido heridas, desgarros e hinchazones, lo cual, aumenta la tasa de falsos rechazos.

Por la cantidad de bytes que se deben almacenar, no es adecuada para ser usada con grandes bases de datos de identificación o verificación.

Iris y Retina:

En ambos métodos la principal desventaja es el alto costo de los componentes que capturan la imagen. Adicionalmente se presenta otro inconveniente, la molestia que causa al usuario la forma como se captura la imagen; pues en el caso de la retina, el usuario debe recibir un haz de luz sobre el ojo, lo cual requiere que este permanezca quieto, provocando el temor y rechazo al método de captura.

Voz:

La exactitud del sistema puede verse afectada principalmente por el ruido de fondo y las características del elemento que captura el sonido (teléfono mismo o un micrófono), así como las variaciones naturales como cambios de humor, fatiga, transcurso del tiempo.

A esto debemos añadir que también puede ser sujeto de engaño a través de grabaciones de voz o tonos de voz semejantes.

Incluso, el tamaño de espacio que ocupa seria un inconveniente si lo usamos para bases de datos de gran tamaño.

Firma:

Algunas limitaciones de este sistema se refieren a que los cambios naturales que se producen en las firmas con el tiempo, pueden aumentar el nivel de falsos rechazos, a lo que se suma el mayor costo con respecto a otros métodos.

1.3 ESQUEMAS DE FUNCIONAMIENTO

Los sistemas biométricos para su implementación necesitan realizar 3 tipos de funciones:

- Registro de usuario
- Verificación
- Identificación

1.3.1 REGISTRO DE USUARIO

Para establecer la identidad de una persona, se trabaja con las características biométricas que se capturan en el momento. Para poderlas comparar con las estadísticas de una base de datos, que guarda los **perfiles** de las personas.

Un perfil o patrón es una representación digital que incluye los atributos correspondientes a partir de un modelo matemático, esta representación es el resultado de la información inicial tomada por el sistema biométrico para su registro en la base de datos.

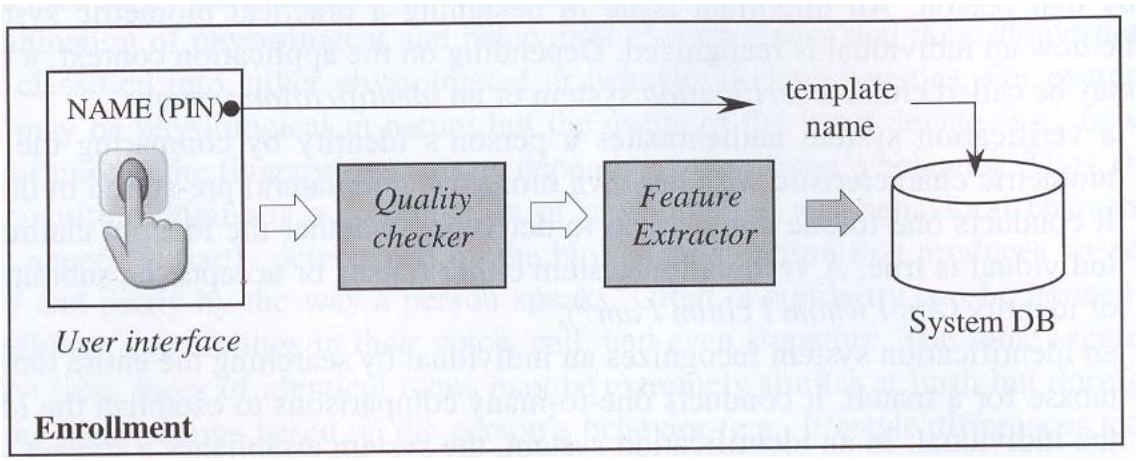


Figura 1.7: **Diagrama de bloques del proceso de registro en base de datos.**

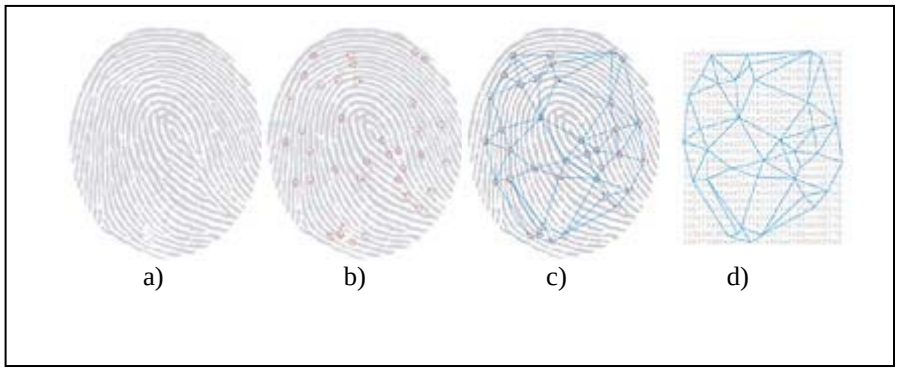


Figura 1.8: **Secuencia de pasos en el proceso de registro de una huella digital**
a) Huella capturada. b) Minutas identificadas c) Enlace entre minutas
d) Representación digital de los enlaces a ser almacenado en la base de datos

1.3.2 VERIFICACIÓN

La **verificación** se basa en la confirmación de la identidad de una persona, es decir la autentificación de esta persona. Se trata de responder a la pregunta: ¿Es quien dice que es?

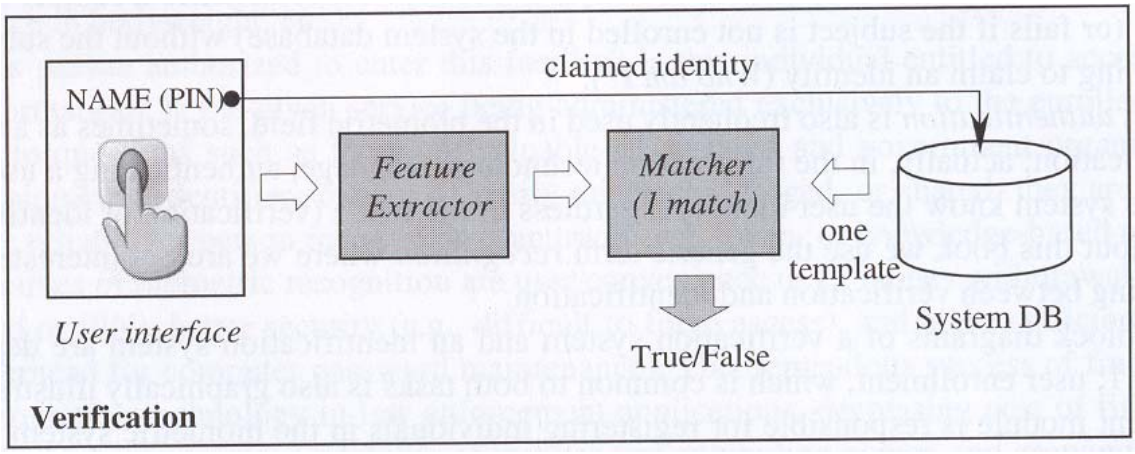


Figura 1.9: **Diagrama de bloques del proceso de verificación.**

1.3.3 IDENTIFICACIÓN

La **identificación** se basa en el reconocimiento de una persona entre muchas. Se trata de responder a la pregunta: ¿Quién es?

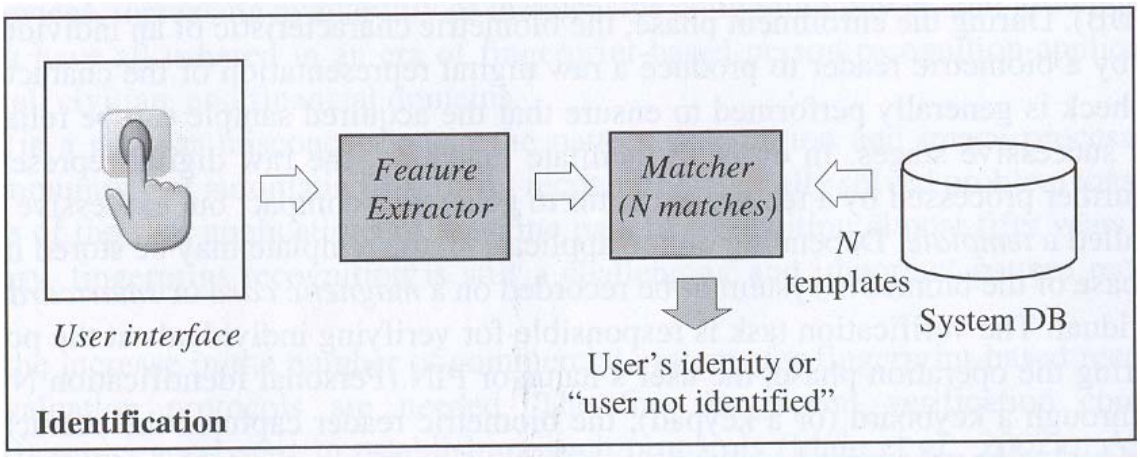


Figura 1.10: **Diagrama de bloques del proceso de identificación.**

1.4 PARÁMETROS DE SEGURIDAD EN UN SISTEMA BIOMÉTRICO

Existen diversas características que permiten indicar el grado de seguridad de autenticación del sistema, entre ellos destacan:

- Falsas Aceptaciones.
- Falsos Rechazos.

Estos dos parámetros se miden en porcentajes.

Las **Falsas Aceptaciones** se da cuando el sistema da como valido la autenticación de una persona que no es la que dice ser.

Mientras que, los **Falsos Rechazos**, es lo opuesto, es decir, el rechazo a persona que deberían ser reconocidas por el sistema.

Los porcentajes de estos 2 parámetros se denominan:
Tasa de Falsas Aceptaciones (FAR) y Tasa de Falsos Rechazos (FRR).

Mientras que un valor muy bajo de FAR es estrictamente necesario para mantener seguro al sistema que se protege, un valor no tan bajo podría ser aceptable en lo que se refiere al FRR.

Aunque si lo analizamos desde la perspectiva del usuario, en lo referente a lo dicho con respecto al FRR, este valor puede ser intolerable por las molestias que le puede causar.

1.4.1 TASA DE FALSOS RECHAZOS Y FALSAS ACEPTACIONES

Las tasas de falso rechazo y falsa aceptación son estadísticamente, mutuamente excluyentes, en la practica se relacionan de manera inversamente proporcional; puesto que, a medida que se desea que el método sea más seguro, es decir, más riguroso en sus comparaciones con los perfiles almacenados, la tasa de falsas aceptaciones baja pero acompañado con el aumento de la tasa de falsos rechazos.

Estas tasas varían en base a la configuración de fábrica que recibe el producto.

Adicionalmente, un aumento de la sensibilidad (grado de tolerancia con respecto a la colocación del dedo en relación a la ubicación inicial en el proceso de registro) del sistema acarrea mayor procesamiento y retardo en el reconocimiento.

Por esta razón, las compañías que fabrican sistemas biométricos de reconocimiento deben establecer una cierta cantidad de falsos rechazos con las consiguientes molestias para usuarios y administradores, a fin de asegurar un mayor grado de seguridad al reducir la tasa de falsas aceptaciones.

En concreto, prácticamente todos los fabricantes trabajan con configuraciones que trabajan con umbrales de falsos rechazos relativamente bajo y niveles de baja sensibilidad a las falsas aceptaciones.

Si se desea, el administrador puede a través del software de control del sistema aumentar el nivel de sensibilidad para lograr un aumento de seguridad, pero traerá como consecuencia un aumento en la tasa de falsos rechazos, acarreando molestias para los usuarios.

A continuación se muestra en la figura 1.5 una comparación de tasas de rechazos y aceptaciones falsas entre los diferentes métodos de identificación.

MÉTODO	RECHAZO (%)	ACEPTACIÓN (1)
Huellas Dactilares	0,5 a 3	Muy bajos
Caras	1 a 5	Bajos
Manos	0,1 a 1	Bajos
Iris	1 a 3	Muy bajos
Retina	0,5 a 1	Muy bajos
Firma	1 a 3	Bajos

(1) Los valores bajos están en el orden del 1 %, o algo menores; los muy bajos no llegan al milésimo y generalmente son mucho menores.

Tabla I.I: Porcentaje de falsos rechazos y aceptaciones en diferentes tipos de sistemas de reconocimiento biométrico.

Como se puede apreciar los métodos mas eficaces en su orden son: de **Retina** y luego el de **Huellas Dactilares**.

Mientras que la **Firma** es el método con mayores valores de falsa aceptación y rechazo.

1.5 SISTEMAS DE HUELLA DIGITAL

Durante años hemos utilizado a las huellas dactilares para podernos identificar, hemos empapado las yemas de nuestros dedos en tinta y luego impregnado en un papel, para usarlo como distintivo entre nuestros conciudadanos.

Esto, a medida que ha pasado el tiempo ha traído muchas complicaciones, tanto de tamaño de almacenamiento como de preservación del dato; por lo cual, la ciencia ha desarrollado nuevos mecanismos, para mejorar este método de reconocimiento; esto se ha traducido en la digitalización de la huella, la cual ha traído mejoras a las complicaciones que antes se producían.

Digitalizar una huella, para almacenarla en una base de datos, no significa que se tenga que guardar toda la imagen completa. Los sistemas de huella digital se aprovechan de características que poseen estas para su almacenamiento, puesto que solo guardan información sobre sus características.

1.5.1 CARACTERÍSTICAS DE UNA HUELLA DIGITAL

Las imágenes de huellas digitales se capturan al colocar las yemas de los dedos sobre una superficial lisa. Estas normalmente se capturan en tonalidades grises (en adelante la denominaremos como '*en escala de grises*'). La estructura característica de una huella es un modelo de **líneas de crestas** y **valles** entrelazados; en una imagen de huella digital las crestas son oscuras y los valles son claros, generalmente viajan en paralelo.

De manera global, las crestas toman diferentes formas (como curvaturas o terminaciones frecuentes); estas regiones denominadas **regiones singulares** o **singularidades** pueden ser clasificadas dentro de tres topologías:

- Lazo
- Arco
- Espiral

En el **lazo**, las líneas de cresta comienzan de un lado del dedo, llegan hasta un tope aproximadamente en el centro de la yema y regresan hacia el mismo lado.

En el **arco**, las líneas de cresta comienzan en un lado del dedo, llegan al centro de la yema y luego siguen hacia el otro lado del dedo, formando un arco que pasa por la zona central de la yema.

En el **espiral,** las líneas de cresta forman círculos aproximadamente concéntricos al centro de la yema.

De manera local, existen otras características importantes encontradas en los modelo de huella digital llamados **Minucias** (pequeños detalles). La American National Standards Institute (ANSI, 1986) propone una clasificación de minucias basada en cuatro tipos:

- Terminaciones
- Bifurcaciones
- Trifurcaciones
- Indeterminado

Cada una de las minucias se posiciona en un sistema de coordenadas X-Y, registrándose, entre otras características, la curvatura de las líneas de cresta y el espacio entre las mismas en la minucia.

1.5.2 DIGITALIZACIÓN DE UNA HUELLA.

1.5.2.1 REPRESENTACIÓN DE LA HUELLA EN 2-D

La imagen digitalizada de una huella en escala de grises se representa como una función de dos dimensiones de n niveles grises; en donde, cada valor de esta función representa su nivel de intensidad en la imagen y cada posición de la imagen, que se identifica con las coordenadas x i y, se denomina píxel. Las líneas de crestas se la relacionan con los píxeles oscuros (valores cercanos a cero en la escala de grises) y los valles con los píxeles claros (valores cercanos a n-1 en la escala de grises).

Los sistemas de reconocimiento por huella digital se aprovechan de la estructura de la huella para su diseño; pues, en el momento del registro en la base de datos, muchos sistemas solo extraen características de la imagen de la huella digital para su almacenamiento; aunque también existen los que comparan imágenes de la huella a través de métodos de correlación (comparan la huella completa).

Las características extraídas a menudo son de tipo física (minucias o singularidades), pero a veces no están directamente relacionados con rasgos físicos (Orientación local de la imagen o respuesta a filtros), esto hace que la información de la huella se guarde en espacio significativamente pequeños en comparación a guardar la imagen completa de la huella.

1.5.2.2 ESQUEMA DE CAPTURA DE LA HUELLA

Para la captura de la imagen de una huella digital, en general, se usa la estructura mostrada en la figura 1.11: un sensor lee la superficie del dedo y convierte la lectura analógica a digital a través de un convertidor A/D; un modulo interfaz es responsable de la comunicación (envío de imágenes, recepción de comandos de control) con dispositivos externos (como una PC o PDA).

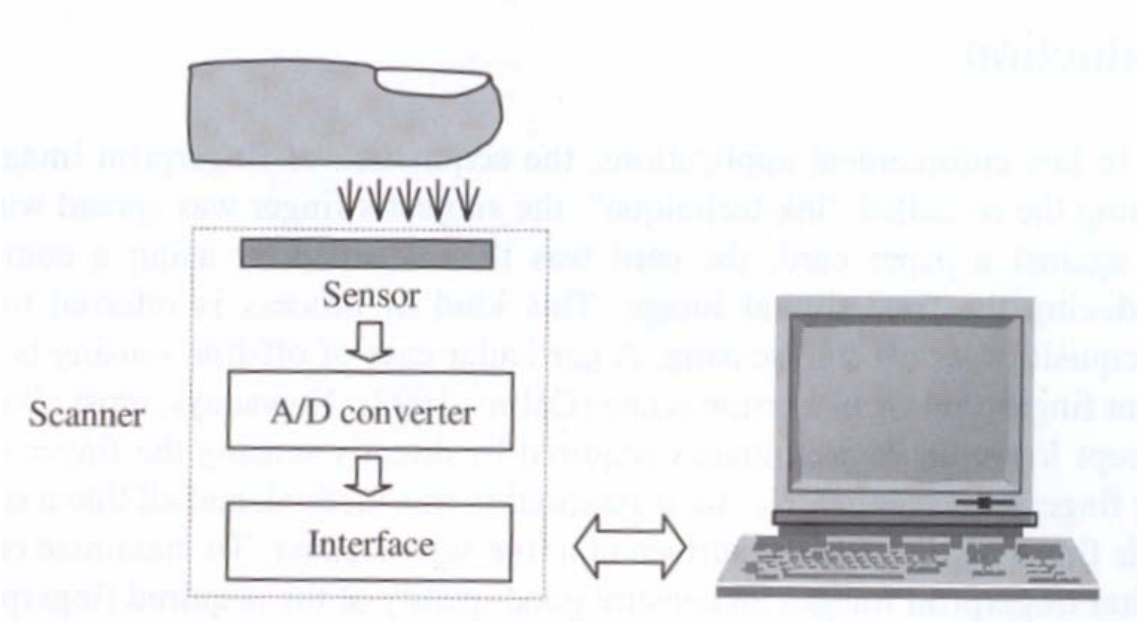


Figura 1.11: **Diagrama de bloques de un escáner de huella digital.**

1.5.2.2.1 Tipos de sensores

En el mercado existen diferente tipos de sensores de huella, como:

- Sensores de fibra óptica
- Sensores electro ópticos
- Sensores capacitivos
- Sensores térmicos
- Sensores de campo eléctrico
- Sensores piezo eléctricos
- Sensores de ultra sonido
- Etc.

1.5.2.2.2 Parámetros de calidad de los sensores

Para medir el nivel de calidad de las imágenes capturadas por los sensores se toman en cuenta los siguientes parámetros:

- *Resolución:* Indica el número de dots o píxeles por pulgada usado en la captura de la imagen
- *Área:* El tamaño del área rectangular de la imagen capturada por el escáner de huella digital.
- Número de píxeles: Es el número de píxeles en una imagen de huella digital
- Rango dinámico: Denota el número de bits usado para codificar el valor de la intensidad de la huella digital
- Precisión geométrica: Especificada como la máxima distorsión geométrica introducida por el sensor.

1.6 OBJETIVOS DEL PROYECTO

Entre los objetivos que se plantean en la tesis están:

- Puesta en marcha del equipo de desarrollo de Texas Instruments (DSK TMS320C6000 y Daughter Board FADT).

- Análisis teórico de diversos algoritmos de compresión de imágenes.

- Implementación de los diversos algoritmos estudiados en el equipo de desarrollo.

- Estudio comparativo de las tasas de error de identificación biométrica mediante huella digital, sobre las imágenes comprimidas.

Capítulo 2

ANÁLISIS TEÓRICO DE LAS TÉCNICAS DE COMPRESIÓN DE IMÁGENES

2.1 INTRODUCCIÓN

Existen muchos factores que influyen en la rapidez con la cual una aplicación puede almacenar, procesar y comunicar información, una de las más importantes es la cantidad de datos que contienen esta información.

En una imagen digital se crean gran cantidad de datos, lo cual causa problemas en su almacenamiento y transmisión.

La compresión de imágenes permite reducir la cantidad de datos para representar una imagen, a través de la eliminación de datos redundantes; es decir, se encarga de transformar una distribución de píxeles en un conjunto de datos sin correlacionar.

La redundancia de datos es la clave de la compresión, es más, es una cantidad cuantificable:

$$R_D = 1 - \frac{1}{C_R} \qquad (2.1a)$$

$$C_R = \frac{N_1}{N_2} \qquad (2.1b)$$

R_D: Redundancia relativa de datos.

C_R: Relación de compresión.

N_1 y N_2: Unidades de información de dos conjuntos de datos que representan la misma imagen.

Por ejemplo, si N1=100 datos; N2=10 datos representa la misma información; entonces la relación de compresión seria de 10 y la redundancia de 0.9; es decir el 90% de los datos de N1 son redundantes y no son necesario para transmitir la misma información que tendrá N2.

2.2 TIPOS DE REDUNDANCIAS

En la compresión digital de imágenes existen tres tipos de redundancias:

- Redundancia de codificación.
- Redundancia entre píxeles.
- Redundancia psicovisual.

2.2.1 REDUNDANCIA DE CÓDIGO[1]

Se presenta cuando los niveles de gris de una imagen están codificados de forma que se emplean más símbolos que los estrictamente necesarios para representar a cada uno de ellos.

[1] Un código es un conjunto de símbolos (letras, números, bits) utilizados para representar un cuerpo de información o un conjunto de sucesos. A cada pieza de información o suceso se le asigna una secuencia de símbolos codificados llamado **palabra código**. El número de símbolo de cada palabra código es su **longitud.**

Casi siempre esto ocurre cuando los niveles de gris de la imagen se representan mediante un código binario natural[2], puesto que este asigna el mismo número de bits tanto a los valores más probables como a los menos probables.

Para eliminar esta redundancia y de paso obtener la compresión de la imagen se utiliza la **codificación de longitud variable**, la cual consiste en asignar menos bits a los niveles de gris más probables y más bits a los menos probables.

2.2.2 REDUNDANCIA ENTRE PÍXELES

Este tipo de redundancia tiene su origen de las correlaciones existentes entre los píxeles de una imagen. La mayor parte de la contribución visual de un único píxel a una imagen es redundante, por lo cual es posible predecir razonablemente el valor de un determinado píxel a partir del valor de sus vecinos.

2.2.3 REDUNDANCIA PSICOVISUAL

Se produce porque el ojo humano no responde con la misma sensibilidad a toda información visual, en el proceso visual normal cierta información simplemente tiene menor importancia relativa que otra.

La información redundante se puede eliminar sin que se altere significativamente la calidad de la percepción de la imagen.

[2] Un código binario natural es aquel en el que a cada pieza de información o suceso a codificar (como el valor del nivel de gris) se le asigna un valor de la forma 2^m, en un código binario de m bits.

Como la eliminación de datos psicovisualmente redundantes se traduce en una pérdida de información cuantitativa, a menudo se denomina con el nombre de **Cuantificación.**

2.3 MODELOS DE COMPRESIÓN DE IMÁGENES

La compresión de imágenes se aprovecha de los tres tipos de redundancias definidos anteriormente. Un sistema de compresión consta de dos bloques estructurales distintos: un **codificador** y un **decodificador**.

Una imagen de entrada f(x,y) alimenta el **codificador**, que crea un conjunto de símbolos a partir de los datos de entrada. Después de la transmisión a través del **canal**, la representación codificada alimenta al **decodificador**, donde se genera una imagen F(x,y) de salida reconstruida, la cual, puede ser (o no) una replica exacta de f(x,y). Si lo es, el sistema esta libre de error, es decir, preserva la información; si no lo es, el sistema presenta algún nivel de distorsión en la imagen construida.

Tanto el codificador como el decodificador consta de de dos funciones, o sub-bloques relativamente independientes. El codificador esta formado por un **codificador de fuente**, que elimina las redundancias de entrada, y un **codificador de canal**, que aumenta la inmunidad al ruido de la salida del codificador de fuente.

El decodificador incluye un decodificador de canal seguido de un decodificador de fuente.

Si el canal entre el codificador y decodificador esta libre de ruido (sin error), se omite el codificador y decodificador de canal.

2.3.1 EL CODIFICADOR Y DECODIFICADOR DE FUENTE

El codificador de fuente sirve para reducir o eliminar de la imagen de entrada redundancias de codificación, entre píxeles y psicovisuales. La aplicación especifica y los requisitos de fidelidad asociados determinan cual es el mejor método de codificación que se debe utilizar en cada situación. Normalmente, este método se puede modelar mediante la secuencia de tres operaciones independientes, estos son:

- Conversor.
- Cuantificador.
- Codificador de símbolos.

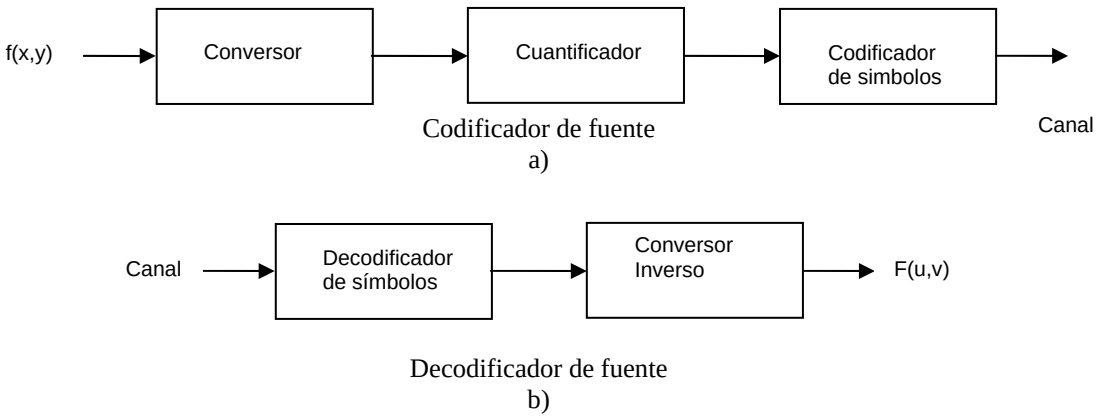


Figura 2.1: **a) Modelo del codificador de fuente. b) y de decodificador de fuente.**

En la primera etapa del proceso de codificación de fuente, el **conversor** transforma los datos de la imagen de entrada en un formato (habitualmente no visualizable) diseñado para reducir las redundancias entre los píxeles de la imagen. Esta operación generalmente es reversible y puede reducir directamente la cantidad de datos necesarios para representar la imagen.

La segunda etapa, o bloque **cuantificador** reduce la precisión de la salida del conversor de acuerdo con algún criterio de fidelidad preestablecido. Esta etapa reduce las redundancias psicovisuales de la imagen de entrada. Esta operación es irreversible por lo cual debe omitirse cuando se desee una compresión libre de errores.

En la tercera y última etapa, el **codificador de símbolos** crea un código de longitud fija o variable que representa la salida cuantificada y transforma la salida de acuerdo con el código.

Para reducir la redundancia de codificación, en la mayoría de los casos el código es de longitud variable. La operación es reversible.

En resumen la aplicación de las tres etapas elimina cada una de las tres redundancias, es importante recordar que si se desea una compresión sin errores se omita la etapa del cuantificador.

El decodificador de fuente solamente contiene dos componentes: **un decodificador de símbolos** y un **conversor inverso**. Estos bloques realizan, en orden opuesto, las operaciones inversas de los bloques del codificador de símbolos y del conversor presentes en el codificador de fuente.

2.3.2 EL CODIFICADOR Y DECODIFICADOR DE CANAL

Cuando el canal de comunicación contiene ruido o es propenso a errores, el codificador y decodificador de canal es una etapa muy importante en el proceso de compresión de imágenes, puesto que reducen el impacto del ruido del canal insertando una forma controlada de redundancia en los datos fuente codificados. Puesto que si no se inserta esta redundancia, la salida del codificador de fuente seria muy sensible al ruido introducido en la transmisión.

Una de las técnicas más útiles de codificación de canal fue desarrollada por R.W. Hamming (1950). Consiste en añadir suficientes bits a los datos que se codifican para asegurar que las palabras código validas difieran en un número mínimo de bits.

2.4 TIPOS DE COMPRESIÓN

Existen dos grandes grupos de compresiones, que se clasifican por la cantidad de datos perdidos:

- Compresión sin pérdidas
- Compresión con pérdidas

2.4.1 COMPRESIÓN SIN PERDIDAS

La compresión sin errores es aplicada en la gestión de documentos médicos o de negocios, en los que la compresión con pérdidas esta prohibida por cuestiones

legales; otra aplicación es el procesamiento de imágenes de los satélites LANDSAT, donde la utilización y el coste de obtención de los datos hacen indeseable cualquier perdida. Otra aplicaron podría ser la radiografía digital, en la que la pérdida de información puede comprometer la precisión de un diagnostico.

Estas técnicas normalmente ofrecen relaciones de compresión de 2 a 10. Se pueden aplicar tanto a imágenes de escala de grises como a binarias. Estas técnicas de compresión se componen de dos etapas relativamente independientes:

- Representación alternativa de la imagen de entrada en la que se reduzca la redundancia entre píxeles.
- Codificación de la representación para eliminar la redundancia de código.

Las técnicas de compresión sin pérdidas se pueden clasificar en:

- Codificación por longitud variable.
- Codificación de plano de bits.
- Codificación predictiva sin pérdidas.

2.4.1.1 CODIFICACIÓN POR LONGITUD VARIABLE

Se basa en la reducción de la redundancia de código. Esta tipo de codificación asigna las palabras código más pequeñas a los niveles de gris más probables.

Entre este tipo de técnicas de codificación tenemos:

- Codificación de Huffman
- Codificación Aritmética

2.4.1.2 CODIFICACIÓN DE PLANOS DE BITS

Esta técnica de compresión reduce la redundancia entre píxeles, para lo cual, se basa en la *descomposición de planos de bits* y en la compresión de cada una de estas imágenes utilizando uno de los métodos de compresión binaria.

Los métodos de compresión binaria usados pueden ser:

- Codificación por zonas constantes.
- Codificación por longitudes de series unidimensionales.
- Codificación por longitudes de series bidimensionales.
- Codificación y trazado de contornos.

Descomposición de plano de bits

La *descomposición en planos de bits* consiste en descomponer una imagen multinivel monocroma o en color) en una serie de imágenes binarias.

Los niveles de gris de una imagen con una escala de grises de *m* bits se pueden representar utilizando el polinomio de base 2:

$$A_{m-1}2^{m-1}+ A_{m-2}2^{m-2}+ .....+A_12^1+A_02^0 \qquad (2.2)$$

Un método sencillo de descomposición, basado en esta propiedad, consiste en separar los *m* coeficientes del polinomio en *m* planos de 1 bit.

2.4.1.3 CODIFICACIÓN PREDICTIVA SIN PÉRDIDAS

La codificación predictiva sin pérdidas se basa en la eliminación de las redundancias entre píxeles muy próximos, extrayendo y codificando únicamente la nueva información que aporta cada píxel.

La nueva información de un píxel se define como la diferencia entre el valor real y el valor estimado de ese píxel.

2.4.2 COMPRESIÓN CON PÉRDIDAS

Estructuralmente se diferencia del otro tipo de compresión, al incluir en su esquema el bloque cuantificador; estas técnicas de compresión elimina la redundancia psicovisual. Este tipo de compresión es capaz de reproducir imágenes monocromas reconocibles a partir de datos que se han comprimido con un factor de compresión de 30, además de reproducir imágenes virtualmente indistinguibles de las originales con factores de 10:1 y 20:1.

Entre las técnicas que aplican la compresión con pérdidas, tenemos:

- Codificación predictiva con perdidas
- Codificación por transformación

2.4.2.1 CODIFICACIÓN PREDICTIVA CON PERDIDAS

Es una versión con pérdida de la codificación predictiva mencionada en la sección compresión sin errores, la diferencia se basa en la adición en su esquema de un cuantificador (la cual causa la pérdida de datos).

2.4.2.2 CODIFICACIÓN POR TRANSFORMACIÓN

En esta técnica de codificación se utiliza una transformación lineal, reversible para hacer corresponder la imagen con un conjunto de coeficientes de la transformada, que después se cuantifican y codifican. Se puede utilizar transformaciones como: Hadamard, Coseno Discreta DCT, Karhunen_Loève, Fourier Discreta, etc.

El esquema del codificador por transformación se realiza en cuatro pasos: descomposición de sub-imágenes, transformación, cuantificación y codificación.

Una imagen de NxN de la entrada se subdivide en sub-imágenes de tamaño nxn, que después se transforman para generar $(N/n)^2$ matrices de transformadas de sub-imágenes nxn. El objetivo del proceso de transformación es deshacer la correlación de los píxeles de cada sub-imagen, o empaquetar tanta información como resulte posible en el menor número de coeficientes de la transformada. La etapa de cuantificación elimina selectivamente, o cuantifica con menor precisión, los coeficientes que lleven la menor cantidad de información, los cuales, son los que producen menor impacto en la calidad de la sub-imagen reconstruida. La ultima

etapa codifica (en la mayoría de los casos, se usa la codificación de longitud variable) los coeficientes cuantificados.

Luego de dar un repaso sobre las diferentes técnicas de compresión que existen en la actualidad, nos centraremos a dar una explicaron mas explicita sobre los métodos que se han implementado o se hubieran implementado en el proyecto.

Estos son:

- Compresión JPEG
- Compresión Hadamard
- Compresión Binaria

2.5 COMPRESIÓN JPEG

La compresión JPEG es un estándar que fue implementado bajo los auspicios de la Organización Internacional para la Estandarización, ISO y el Comité Consultivo de la Telefonía y Telegrafía Internacional, CCITT.

Es utilizado en la compresión de imágenes monocromas y en color, se definen tres sistemas diferentes de codificación:

- *Un sistema de codificación secuencial*: Con pérdidas, que se basa en la transformada DCT y es apropiada para la mayoría de las aplicaciones.
- *Un sistema de codificación extendida*: Para aplicaciones de mayor compresión, mayor precisión, o de reconstrucción progresiva.

- *Un sistema de codificación independiente sin perdidas*: Para la compresión reversible.

Para poder ser compatible con el estándar JPEG, un producto o sistema debe admitir el sistema básico.

Compresión JPEG de codificación básico.

La compresión se realiza en tres etapas: cálculo de DCT, cuantificación y asignación de un código de longitud variable.

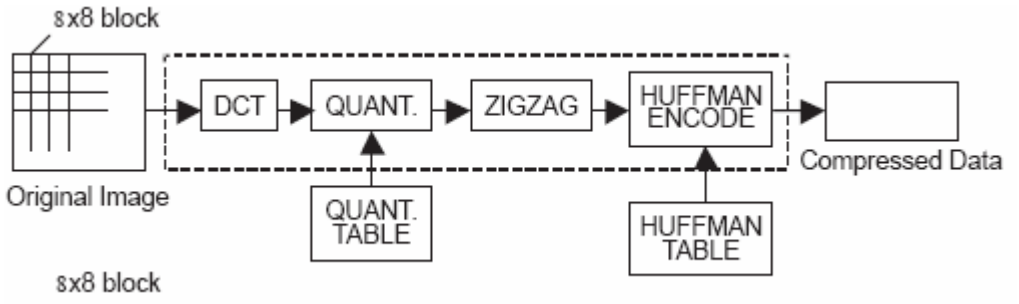
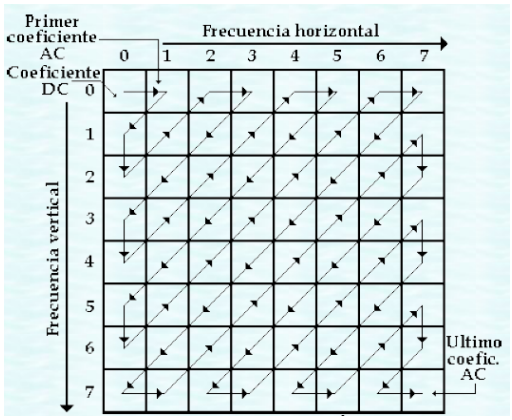


Figura 2.2: **Esquema de bloques de la compresión JPEG.**

La imagen se divide en bloques de píxeles de tamaño 8x8, que se procesan de izquierda a derecha y de arriba abajo. Según se va encontrando cada bloque o sub-imagen de 8x8, se cambian los niveles de sus 64 píxeles, sustrayendo de los mismos la cantidad 128, siendo 256 el máximo número de niveles de gris. Luego, se calcula la transformada del coseno discreta bidimensional del bloque, se cuantifica según la tabla estándar de la figura 2.3b y se reordenada utilizando el patrón en zigzag de la figura 2.3a para construir una secuencia unidimensional de coeficientes cuantificados.



a)

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

b)

Figura 2.3: **(a) Barrido en zigzag de la sub-imagen cuantificada. b) Tabla estándar de cuantificación.**

C*(u,v)=Redondeo$\left(\frac{F(u,v)}{N(u,v)}\right)$, donde N(u,v) es la matriz de normalización de la figura 2.3b, F(u,v) es la sub-imagen transformada y C*(u,v) es resultado de la cuantificación.

Ya que la matriz unidimensional reordenada según el patrón zigzag esta distribuida cualitativamente según una frecuencia creciente, el proceso de codificación JPEG ha sido diseñado de modo que se beneficia de la existencia de largas series de ceros que se producen normalmente en la reordenación. En particular, los coeficientes AC[3] no nulos se codifican utilizando un código de longitud variable (se usa una tabla de codificación Huffman AC) que define el valor del coeficiente y el número de ceros que le preceden. El coeficiente DC se codifica como la diferencia con respecto al coeficiente DC de la sub-imagen anterior (se usa una tabla de codificación Huffman DC).

[3] El termino AC se refiere a todos los coeficientes de la transformada, a excepción del coeficiente de orden cero, llamado coeficiente DC.

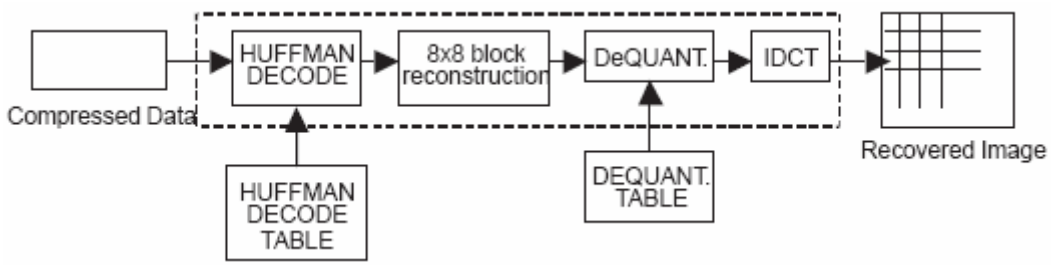


Figura 2.4: **Esquema de bloques de la descompresión JPEG.**

Para descomprimir una sub-imagen comprimida con JPEG el decodificador debe reproducir, en primer lugar, los coeficientes normalizados de la transformada de los se obtuvo el flujo de bits de la sub-imagen comprimida. Puesto que una secuencia binaria codificada por Huffman se decodifica instantáneamente y de forma única, este paso se realiza fácilmente con solo tablas de equivalencias. Después de deshace la normalización usando la ecuación:

$$F^*(u,v) = C^*(u,v) * N(u,v). \qquad (2.3)$$

donde N es la matriz de normalización de la figura 2.3b, C* es la matriz decodificada y F* es la matriz de-cuantificada que posteriormente se le aplicara transformada inversa. La imagen totalmente reconstruida se obtiene aplicando la transformada inversa DCT a la imagen resultante del paso anterior, luego se desplaza los niveles de los píxeles de la transformada inversa la cantidad 128, obteniendo de esta manera una sub-imagen con valores cercanos a la sub-imagen original antes de aplicar la compresión.

2.5.1 TRANSFORMADA COSENO DISCRETA (DCT).

Es una transformada real y ortogonal, tiene unas características óptimas en cuanto a compactación de energía y a decorrelacionar coeficientes.

Su aplicación más importante es la compresión de imágenes (para lo cual la imagen se subdivide en matrices de 8x8).

Esta transformación esta definida por la ecuación 2.4:

$$F_{v,u} = \frac{1}{4}\, c_u\, c_v \sum_{x=0}^{7} \sum_{y=0}^{7} f_{y,x} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16} \tag{2.4}$$

$$C_U = C_V = \begin{cases} \frac{1}{\sqrt{2}} & \text{para u, v} = 0 \\ 1 & \text{para otro caso.} \end{cases}$$

Esta transformada llevada a forma matricial se expresa por la ecuación 2.5:

$$T_{8x8} = M_{8x8} * F_{8x8} * M^{T}_{8x8} \tag{2.5}$$

Donde T es el resultado de la transformada de la sub-imagen, F es la matriz de las sub-imagen y M es la matriz base de la Transformada, donde M_{8x8} se define a través ecuación 2.6:

$$M = \begin{matrix} 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} & 1/\sqrt{8} \\ \phi(1) & \phi(3) & \phi(5) & \phi(7) & \phi(9) & \phi(11) & \phi(13) & \phi(15) \\ \phi(2) & \phi(6) & \phi(10) & \phi(14) & \phi(18) & \phi(22) & \phi(26) & \phi(30) \\ \phi(3) & \phi(9) & \phi(15) & \phi(21) & \phi(27) & \phi(33) & \phi(39) & \phi(45) \\ \phi(4) & \phi(12) & \phi(20) & \phi(28) & \phi(36) & \phi(44) & \phi(52) & \phi(60) \\ \phi(5) & \phi(15) & \phi(25) & \phi(35) & \phi(45) & \phi(55) & \phi(65) & \phi(75) \\ \phi(6) & \phi(18) & \phi(30) & \phi(42) & \phi(54) & \phi(66) & \phi(78) & \phi(90) \\ \phi(7) & \phi(21) & \phi(35) & \phi(49) & \phi(63) & \phi(77) & \phi(91) & \phi(105) \end{matrix} \tag{2.6}$$

$$\phi(u) = \frac{1}{2}\cos\left(\frac{u\pi}{16}\right)$$

La transformada inversa del Coseno Discreto se define a través de la ecuación 2.7

$$f_{y,x} = \frac{1}{4}\sum_{u=0}^{7}\sum_{y=0}^{7} c_u\, c_v\, F_{v,u}\, \cos\frac{(2x+1)u\pi}{16}\, \cos\frac{(2y+1)v\pi}{16} \tag{2.7}$$

$$C_U = C_V = \begin{cases} \frac{1}{\sqrt{2}} & \text{para u, v} = 0 \\ 1 & \text{para otro caso.} \end{cases}$$

2.6 COMPRESIÓN HADAMARD.

La compresión Hadamard, no es de usado de manera estándar como la compresión JPEG; el método de compresión Hadamard se basa en dos etapas: Transformación Hadamard y Cuantificación.

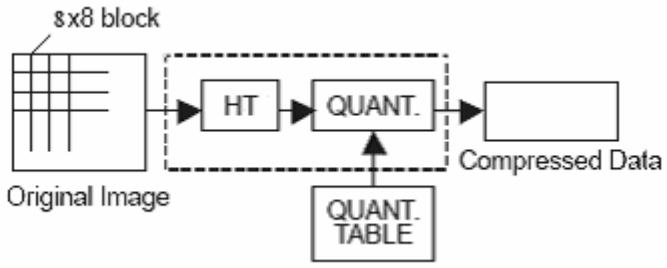


Figura 2.5: **Esquema de bloques de la compresión Hadamard.**

La imagen original a comprimir se la divide en sub-imágenes de tamaño 8x8, se le aplica la transformada de Hadamard para eliminar la redundancia entre píxeles, y luego se le aplica la matriz normalizada indicada en la figura 2.6 para proceder a la cuantificación de la salida de la transformación. Para esto se multiplica cada elemento de la matriz a cuantificar por el correspondiente elemento de la matriz normalizada.

$$\begin{bmatrix} 11110000 \\ 11110000 \\ 11110000 \\ 11110000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \end{bmatrix} \qquad \begin{bmatrix} 11111100 \\ 11111100 \\ 11111100 \\ 11111100 \\ 11111100 \\ 11111100 \\ 00000000 \\ 00000000 \end{bmatrix}$$

a) b)

Figura 2.6: **(a) Matriz cuantificadora para una compresión 4:1. b) Matriz cuantificadora para una compresión 1,78:1.**

Si la compresión es de 4: 1 cada sub-imagen será almacenada en 16 bytes, y si la compresión es de 1,78: 1 cada sub-imagen será almacenada en 36 bytes.

La descompresión de los datos se realiza convirtiendo cada una de las sub-imágenes transformadas comprimidas en sub-imágenes de 8x8, luego se le aplica la Transformada Inversa de Hadamard, el resultado de esta operación da una sub-imagen que visiblemente posee un parecido con la imagen original.

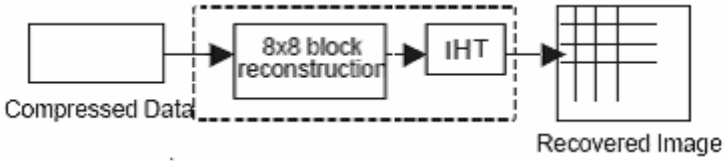


Figura 2.7: **Esquema de bloques de la descompresión Hadamard.**

2.6.1 TRANSFORMADA HADAMARD

Es una transformación real, simétrica y ortogonal, de cálculo rápido, posee una buena compactación de energía para imágenes altamente correlacionadas.

Esta definida a través de la ecuación 2.8

$$H(u,v) = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x[m,n]\cdot(-1)^{\sum_{i=0}^{t-1}[b_i(m)b_i(u)+b_i(n)b_i(v)]} \tag{2.8}$$

$$\boxed{a_{u,v}(m,n) = \frac{1}{N}\cdot(-1)^{\sum_{i=0}^{t-1}[b_i(m)b_i(u)+b_i(n)b_i(v)]}}$$

Esta transformada, puede ser modelada en forma de matriz

$$T_{nxn} = H_{nxn} * F_{nxn} * H_{nxn} \tag{2.9}$$

en donde: T es el resultado de la transformada, F es la imagen original y H es la matriz base de Hadamard (ecuación 2.10). Para un n de valor tres, la matriz H_{3x3}, queda de la siguiente manera:

$$H_3 = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \tag{2.10}$$

La transformada inversa de Hadamard se define a través de la ecuación 2.11, en donde se puede observar que usa la misma función base de la transformada directa de Hadamard.

$$x[m,n] = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} H(u,v)\cdot(-1)^{\sum_{i=0}^{t-1}[b_i(m)b_i(u)+b_i(n)b_i(v)]} \tag{2.11}$$

2.7 COMPRESIÓN BINARIA.

Esta compresión se basa en la binarización de las imágenes de entrada. En el caso de la implementada en el proyecto, se elige un valor de umbral, si el valor del píxel es mayor al de umbral entonces se le da el valor de 255 (blanco en la escala de grises) y si el valor del píxel es menor al de umbral entonces se le da el valor de cero (negro en la escala de grises).

Los valores de 0 y 255 siguen teniendo 8 bits para su descripción, por eso estos valores se convierten en 0 y 1, respectivamente, para describir cada píxel con un solo bit. Esto hace que la compresión sea fija, de ocho a uno; inmediatamente se encripta los datos, de tal manera que en un byte el espacio que antes ocupaba 1 píxeles, ahora lo ocupan 8 píxeles.

Para la descompresión se usa el mismo procedimiento indicado anteriormente, pero de manera inversa.

Además del método mencionado, también existen métodos estandarizados que no fueron usados en el proyecto, pero vale mencionarlos, estos son:

- Compresión Unidimensional o Estándar grupo 3.
- Compresión Bidimensional o Estándar grupo 4.

2.7.1 COMPRESIÓN UNIDIMENSIONAL O ESTÁNDAR GRUPO 3

En este método cada línea de una imagen se codifica como una serie de palabras código de longitud variable que representa las longitudes de las series blancas y negras alternativas observadas en una exploración de izquierda a derecha de cada línea. Las propias palabras son de dos tipos. Si la longitud de la serie es menor que 63, se utiliza un código de finalización. Si la longitud de la serie es mayor a 63, se utiliza el código de construcción junto con un código de finalización que representa la diferencia entre el código constructor y la longitud real de la serie.

2.7.2 COMPRESIÓN BIDIMENSIONAL O ESTÁNDAR GRUPO 4

Es un método línea a línea en el que se codifica la posición de cada transición de series blancas a negras, o de negras a blancas con respecto a la posición de un elemento de referencia a_0 situado en la línea de codificación actual. A la línea codificada se la denomina línea de referencia; la línea de referencia de la primera línea de una nueva imagen es una línea blanca imaginaria.

Capítulo 3

IMPLEMENTACIÓN DEL SISTEMA

3.1 ENTORNO DE DESARROLLO

La implementación del proyecto se realizará en base a herramientas de la marca Texas Instrument, marca de la cual se usará como hardware una placa de desarrollo TMS320C6713 DSK. Esta servirá como interfaz entre el PC y el sensor.

Un daughter card FPC1010 que servirá como sensor para la captura de las huellas digitales, en él viene incorporado un convertidor analógico-digital.

Fig. 3.1: **Entorno de trabajo.**

Como software para el PC, usaremos la aplicación Code Composer Studio Versión 2.20, el cual se encargará de programar al DSK 6713, almacenar las imágenes y

patrones en el disco duro del PC; adicionalmente junto con el sensor de huellas viene incorporado las librerías que nos permitirán manejar la daughter card desde el Code Composer Studio.

En el CCS tendremos las siguientes extensiones de archivos:
*.dat: Archivo de datos, *.out: Archivo ejecutable, *.c: Archivo código fuente, *.wks: Archivo de contiene el ejecutable y todos los códigos fuente, y *.gel: Archivo que contiene una interfaz entre el usuario y el código fuente del programa.

El FPC 1010 y los drivers que lo controlarán viene en un solo paquete llamado FADT[4]

En lo que respecta a los métodos de compresión, se han implementado tres métodos: La *Compresión JPEG* de codificación secuencial o básico, un método de compresión basado en la transformada Hadamard al cual llamaremos *Compresión Hadamard* y un método de compresión basado en la binarización de la imagen de escala de grises y posterior encriptamiento, al cual llamaremos *Compresión Binaria*.

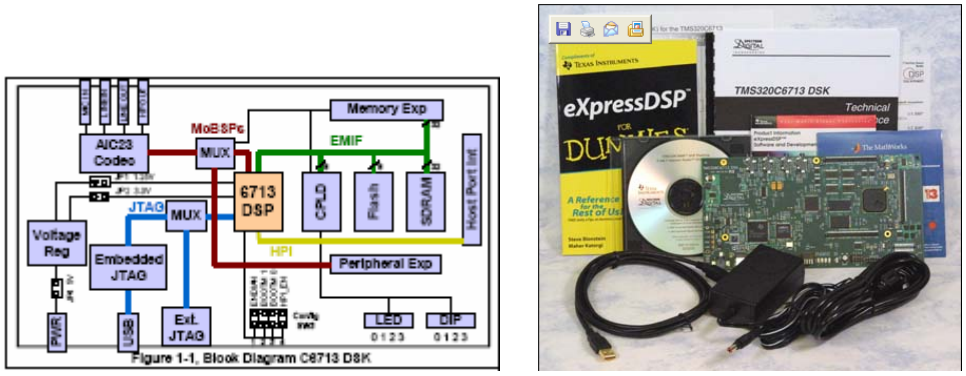
3.1.1 TARJETA DE ADQUISICIÓN DE DATOS TMS320C6713DSK

Para el desarrollo del proyecto se ha usado como interfaz entre el PC y el sensor de huella al equipo de desarrollo del DSP TMS320C6713 diseñado por la Texas Instrument; este DSK (DSP Started Kit) posee las siguientes características:

- Un Procesador de Señales Digitales TMS320C6713 de la marca Texas Instrument.

[4] FADT son las siglas de Fingerprint Authentication Development Tool

- Un codec stereo AIC23.
- 8 Mbytes SDRAM (Synchronous Dinamic Random Access Memory).
- 512 Kbytes de memoria FLASH no volátil (256 Kbytes usado en configuración por default).
- 4 LEDs e interruptores DIP accesibles por el usuario.
- Configuración de la placa por software a través de registros implementados en CPLD.
- Opción “boot” configurable.
- Conectores de expansión estándar para el uso del daughter card.
- Emulación JTAG a través del emulador JTAG sobre la tarjeta con interfaz servidor USB o emulador externo.
- Fuente de poder voltaje simple (+5V).



a) b)

Figura 3.2: **a) Diagrama de bloques del 6713 DSK b) Imagen del DSK 6713 completo.**

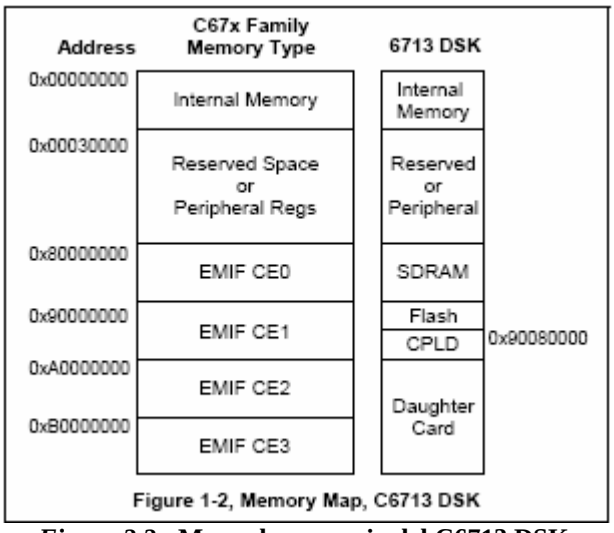


Figura 3.3: **Mapa de memoria del C6713 DSK.**

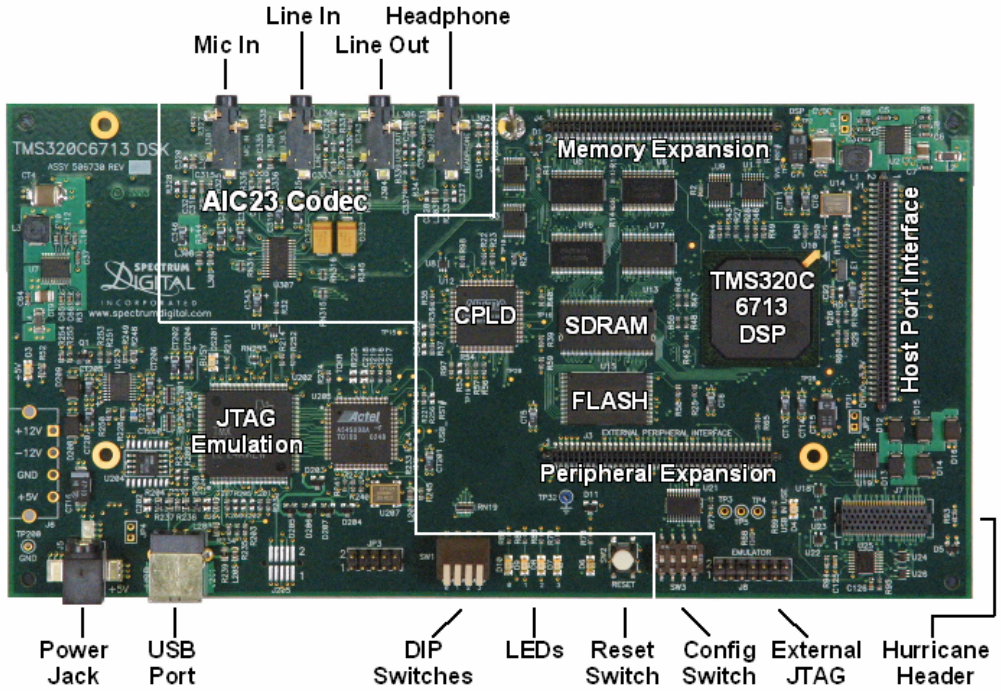


Figura 3.4: **Ubicación de los elementos en el C6713 DSK.**

3.1.1.1 PROCESADOR DIGITAL DE SEÑALES DSP TMS320C6713

Es un procesador de señales digital que trabaja en coma flotante y posee una velocidad de procesado de 225 Mhz. Posee una arquitectura de dos niveles de memoria caché: L1 y L2; en la cual, el nivel 1 se divide en L1P y L1D. L1P es una memoria caché de programa de 4 Kbytes y L1D es una memoria caché de datos con 2 vías de acceso asociados de 4 Kbytes, en ambas, la CPU accede directamente. L2 es una memoria de 256 Kbytes, esta se divide en 64 Kbytes (son 4 bancos de 16 Kbytes) que puede trabajar como nivel 2 de memoria caché o como memoria RAM, y 192 Kbytes de memoria RAM; la RAM actúa como memoria interna del DSP.

Tiene en total un espacio lógico de memoria direccionable de 4 Gigabytes. Puede trabajar con palabras de datos de 8, 16, 32 y 64 bits.

En su interior posee los siguientes periféricos:

- **EMIF: External Memory Interface**; el cual permite la interacción entre la DSP y sus periféricos externos, tiene un bus de datos de 32 bits (ED 31:0) y un bus de dirección de 20 bits (EA 21:2), tiene 4 espacios de memoria cada uno con 128 Mbytes que en total conforman 512 Mbytes de espacio direccionable.

- **EDMA: Enhanced Direct Memory Access**; se encarga de transferir datos entre la memoria interna del DSP (L2) y sus periféricos internos. Posee 16 canales, que permite sincronizar los eventos que realicen los periféricos.

- **HPI: Host Port Interface**; es un puerto paralelo de 16 bits que permite a un procesador servidor acceder directamente al espacio de memoria de la CPU del DSP y al EDMA.

- **McASP: Multichannel Audio Serial Port**; son 2 unidades.

- **I2C: Inter-Integrated Circuit**; son 2 unidades.

- **McBSP: Multichannel Buffered Serial Port**; son 2 puertos seriales que poseen comunicación full-duplex. El modo selección multicanal le permite seleccionar canales independientes para transmitir-recibir en una misma fase. Tiene 128 canales, de los cuales 32 canales pueden ser habilitados en un tiempo dado.

- **Timers de 32 Bits**; son 2 unidades.

- **GPIO: General Purpose Input/Output**; son 16 Puertos (GP 0:15) que pueden funcionar como entradas o salidas.

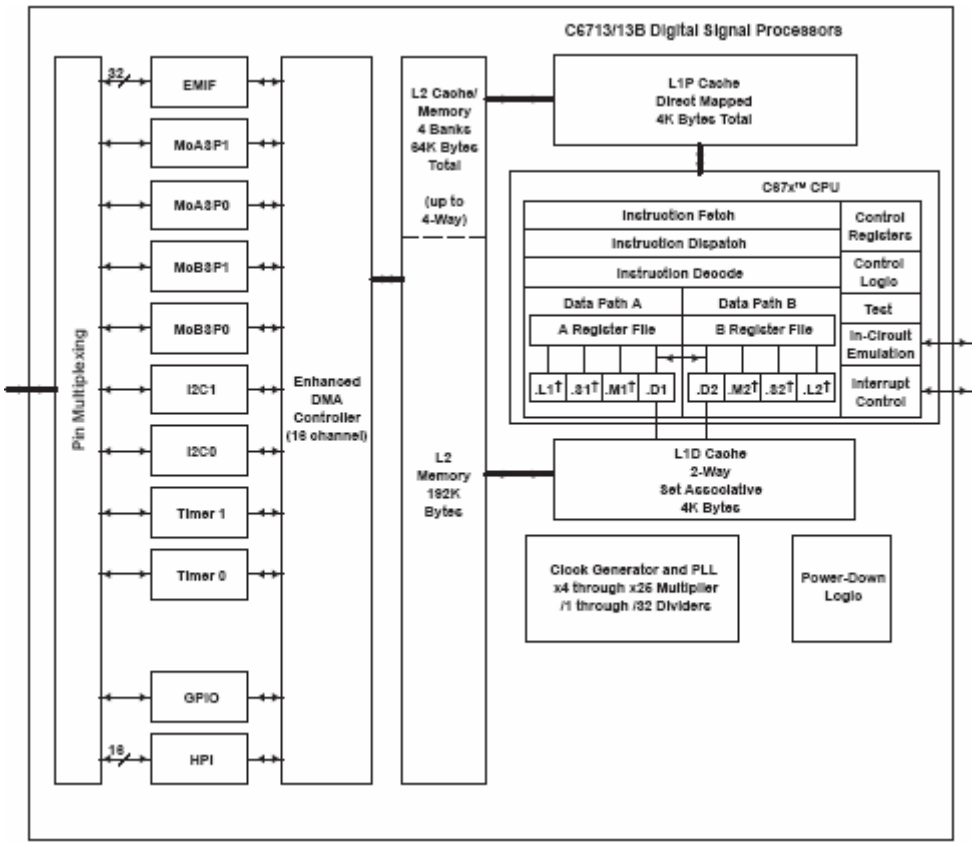


Figura 3.5: **Diagrama de bloques del DSP C6713.**

3.1.2 EQUIPO DE VERIFICACIÓN E IDENTIFICACIÓN DE HUELLAS FADT

El Fingerprint Authentication Development Tool (FADT) es un equipo que permite el desarrollo de sistemas embebidos de autentificación de huellas digitales basados en DSP de la Texas Instrument. Consiste de un daughter card (FPC 1010) con un software asociado que incluye controladores del sensor para captura de imágenes de huella digital a través del DSP 6713 y una versión de evaluación de algoritmos de verificación y/o reconocimiento de huellas digitales llamada FPCore Demo Software.

Figura 3.6: **Imagen del sensor y software del FADT.**

3.1.2.1 SENSOR DE HUELLA DIGITAL FPC1010

El FPC1010 FingerPrint Sensor Daughter Card es un sensor del tipo capacitivo, el cual incluye en su empaquetamiento: un convertidor Analógico Digital y una interfaz de puerto serial (SPI), para interactuar con el McBSP del DSP 6713. El McBSP del DSP actúa como maestro y el SPI del sensor actúa como esclavo.

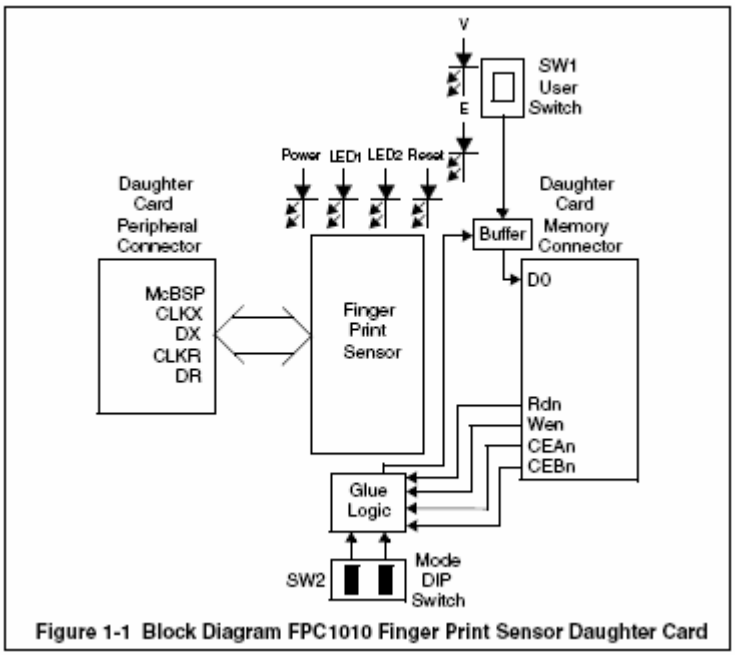


Figura 3.7: **Diagrama de Bloque del Daughter Card FPC1010.**

El sensor tiene las siguientes características:

- Sensor de 200x152 píxeles con resolución de 8 bits, imagen a escala de grises.
- Interruptor Selección de Modo.
- Interruptor Selección de Usuario.
- Seis LED's indicadores de estado.
- Dos conectores para comunicarse con la DSK:
    - Conector Periférico: Para transmitir y recibir datos, y recibir órdenes de la DSP C6713.
    - Conector Memoria: Para que la DSP le asigne un espacio en la memoria, a través del EMIF, y poder habilitar la lectura de los datos capturados por el sensor.

3.1.3 SOFTWARE CODE COMPOSER STUDIO CCS

El CCS es el software que será utilizado para la programación del DSP 6713, posee un entorno de desarrollo integrado que suministra aplicaciones tipo ventana en la cual se ejecuta todo el código de programa, desde las variables de entrada y edición de tu código de programa, hasta la compilación y construcción de un archivo ejecutable, y finalmente depurar el código de programa.

El código de programa se puede editar utilizando lenguaje: C, C++ o ensamblador.

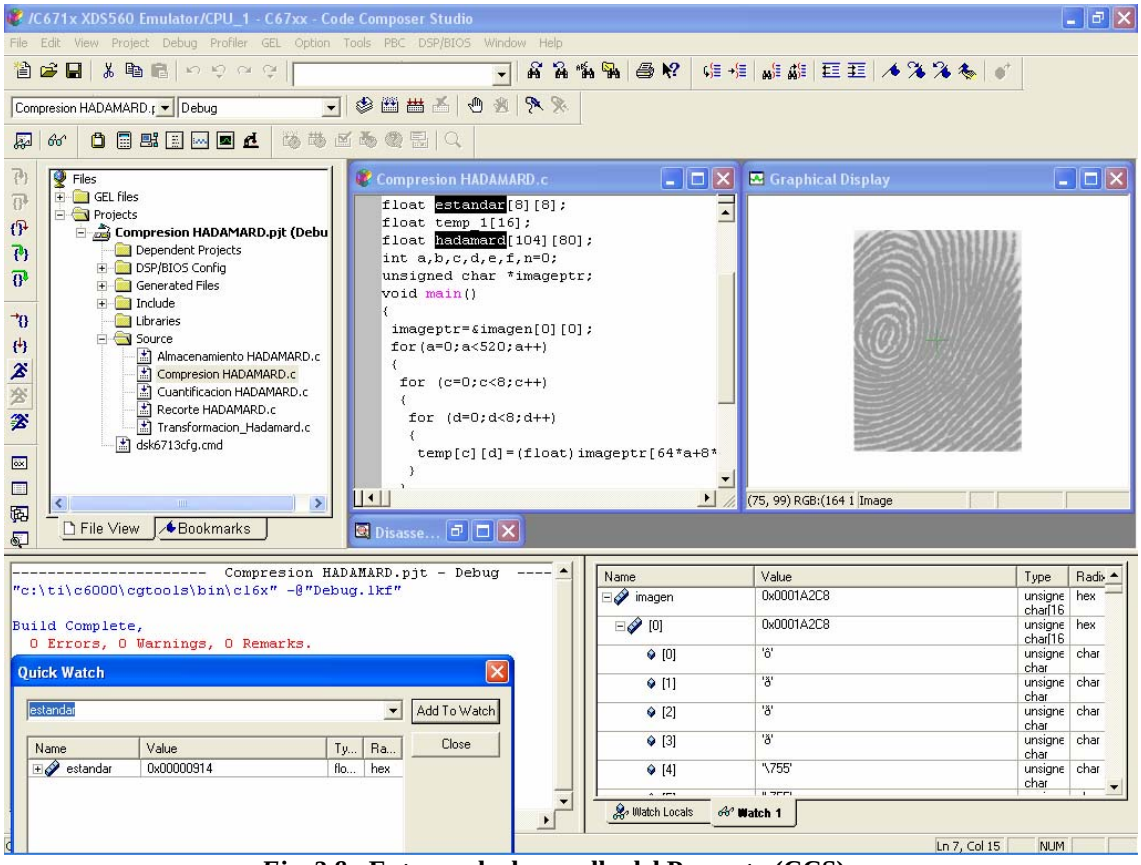


Fig. 3.8: **Entorno de desarrollo del Proyecto (CCS).**

3.2 JPEG DE CODIFICACIÓN SECUENCIAL

3.2.1 COMPRESIÓN JPEG

Para implementar la compresión JPEG dentro del CCS, el código fuente se lo dividió en programa principal y en funciones. En el anexo E encontramos el código fuente completo.

En el programa principal, la imagen se divide en sub-imágenes 8x8. Las funciones se llaman:

Resta: En la cual, a cada valor de la sub-imagen 8x8, se le resta 128.

Transformación DCT: La sub-imagen 8x8 es decorrelacionada usando la Transformada Coseno Discreta.

Cuantificación: La sub-imagen 8x8 es cuantificada usando la matriz estándar de cuantificación.

Zigzag: La sub-imagen es reordenada usando el patrón zigzag mostrado en la figura 2.3a y se forma un vector de 64 elementos.

Inversión: El orden de los elementos del vector se invierte.

Ceros: Lee el número de ceros que contiene el vector hasta llegar al primer valor diferente de cero.

Valores DC: Almacena en un vector el valor del primer coeficiente de las sub-imágenes 8x8.

Huffman AC: Se codifica todos los valores de la sub-imagen 8x8 (excepto el primer coeficiente) usando el procedimiento de Huffman

Huffman DC: Se codifica usando Huffman todos los valores DC de la imagen completa.

Huffman: Almacena todos los valores, tanto DC como AC codificados, sin empaquetarlos.

Almacenamiento: Almacena todos los valores codificados empaquetados, culminando el proceso de compresión.

3.2.2 PROCEDIMIENTO PARA LA COMPRESIÓN JPEG

1) Cargar el archivo **compresión2.out,** que esta ubicado en C:\ti\Proyectos de TESIS\Compresion JPEG\Compresion JPEG 2\Debug\, usando **File→Load Program.**

2) Cargar la imagen original que se va a comprimir, usando **File→Data→Load**, en el espacio Address escriba **imagen.**

3) Ejecute la compresión, usando **Debug→Run.**

4) Guarde la imagen comprimida usando **File→Data→Save,** dialogo **Storing Memory into File**, en la caja **Address** escriba **compresión_JPEG** y en **Length** el número de bytes de la imagen comprimida.

3.2.3 DESCOMPRESIÓN JPEG

Para implementar la descompresión JPEG dentro del CCS, el código fuente se lo dividió en programa principal y en funciones. En el anexo F encontramos el código fuente completo.

Las funciones que se usan en el programa se llaman:

Número bytes: Permite calcular la cantidad de bytes diferentes de cero que contiene la imagen a descomprimir.

Decodificación Huffman: Permite revertir la codificación Huffman realizada a la imagen en el proceso de compresión**.**

Valores DC: Permite recuperar el valor original de los valores DC antes de la codificación de la imagen completa.

Reconstrucción Sub-imagen: Desde esta función hasta la última, a la matriz se la va a procesar por sub-matrices de tamaño 8x8. Permite la construcción de las sub-imágenes antes del reordenamiento en zigzag en el proceso de compresión.

Zigzag: Permite el reordenamiento de las sub-imágenes antes de la etapa de cuantificación en el proceso de compresión.

Cuantificación: Permite la decuantificación de las sub-imágenes antes de la etapa de transformación en el proceso de compresión.

Transformada Inversa: Permite la transformación inversa Coseno Discreta de las sub-imágenes a descomprimir.

Suma: Contiene la función que añade a cada uno de los elementos de la sub-imagen el valor de 128, además convierte sus cantidades de valores con decimales a enteros.

Almacenamiento: Contiene la función que reordena a cada sub-imagen y luego la almacena en una matriz, la cual tendrá temporalmente la imagen total.

Reordenamiento: Contiene la función que permite reordenar los valores de la imagen completa descomprimida.

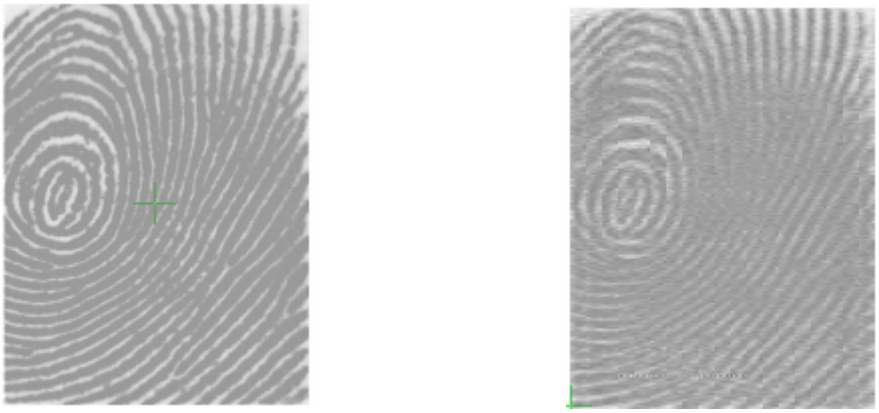
3.2.4 PROCEDIMIENTO PARA LA DESCOMPRESIÓN JPEG

1) Cargar el archivo **Descompresión JPEG.out,** que esta ubicado en C:\ti\Proyectos de TESIS\Compresion JPEG\Descompresion JPEG\Debug\, usando **File→Load Program.**

2) Cargar la imagen comprimida que se va a comprimir, usando **File→Data→Load**. En el espacio Address escriba **dato;** el archivo esta ubicado en C:\ti\Datos de Proyecto\imagenes\imagenes comprimidas\JPEG\Usuario_x\comprimidas\ (el valor x es el usuario, hay 10)**.**

3) Ejecute la compresión usando **Debug→Run.**

4) Guarde la imagen comprimida usando **File→Data→Save** en C:\ti\Datos de Proyecto\imagenes\imagenes comprimidas\JPEG\Usuario_x\descomprimidas**,** en el dialogo **Storing Memory into File**, en la caja **Address** escriba **imagen** y en **Length** el número 33280.



a) b)

Figura 3.9: **Comparación entre a) imagen original b) Imagen JPEG descomprimida.**

3.3 HADAMARD

3.3.1 COMPRESIÓN Hadamard

Para implementar la compresión Hadamard, tanto la compresión 4:1 como la 1,78:1 dentro del CCS, el código fuente se divide en programa principal y en funciones. En el anexo G encontramos el código fuente de la compresión 1,78:1 y en el anexo I encontramos el de la compresión 4:1.

En el programa principal, la imagen se divide en sub-imágenes 8x8. Las funciones se llaman:

Transformada Hadamard: En esta función se aplica a cada una de las sub-imágenes 8x8 la transformada de Hadamard.

Cuantificación: Contiene la función que permite cuantificar cada una de las imágenes, la matrices a usar para cuantificar son las de la figura 2.6.

Recorte: En esta función se almacenan temporalmente de la compresión 4:1, 16 valores de la sub-imagen cuantificada y de la compresión 1,78, 36 valores de la imagen cuantificada.

Almacenamiento: Contiene la función que permite almacenar en una matriz la imagen total comprimida.

3.3.2 PROCEDIMIENTO PARA LA COMPRESIÓN HADAMARD

1) Cargar el archivo **compresión hadamard xx.out;** donde xx depende de cual nivel de compresión se desea elegir; esta ubicado en C:\ti\Proyectos de TESIS\Compresion HADAMARD\4-1\Compresion hadamard 4-1\Debug (para la compresión 4:1) y C:\ti\Proyectos de TESIS\Compresion HADAMARD\2-1\COMPRESION HADAMARD 2-1\Debug (para la compresión 1,78:1) usando **File→Load Program.**

2) Cargar la imagen original que se va a comprimir usando **File→Data→Load**, en el espacio Address escriba **imagen.**

3) Ejecute la compresión usando **Debug→Run.**

4) Guarde la imagen comprimida usando **File→Data→Save,** dialogo **Storing Memory into File**, en la caja **Address** escriba **hadamard** y en **Length** el número de bytes de la imagen comprimida; para 4:1, 8320 y para 1,78, 18720.

3.3.3 DESCOMPRESIÓN HADAMARD

Para implementar la descompresión Hadamard, tanto la compresión 4:1 como la 1,78:1 dentro del CCS, el código fuente se divide en programa principal y en funciones, el proceso se realiza dividiendo la imagen comprimida en vectores. En el anexo H encontramos el código fuente de la descompresión 1,78:1 (aquí se toma un vector de 36 valores) y en el anexo J encontramos el de la descompresión 4:1 (aquí se toma un vector de 16 valores).

En el programa principal, la imagen se divide en sub-imágenes 8x8. Las funciones se llaman:

Conversión: Aquí se realiza la cuantificación inversa de cada una de las sub-imágenes.

Transformación: Se aplica a cada una de las sub-imágenes la transformada inversa de Hadamard.

Almacenamiento: Aquí se almacenan todas las sub-imágenes, se las agrupa y se obtiene la imagen descomprimida.

3.3.4 PROCEDIMIENTO PARA LA DESCOMPRESIÓN HADAMARD

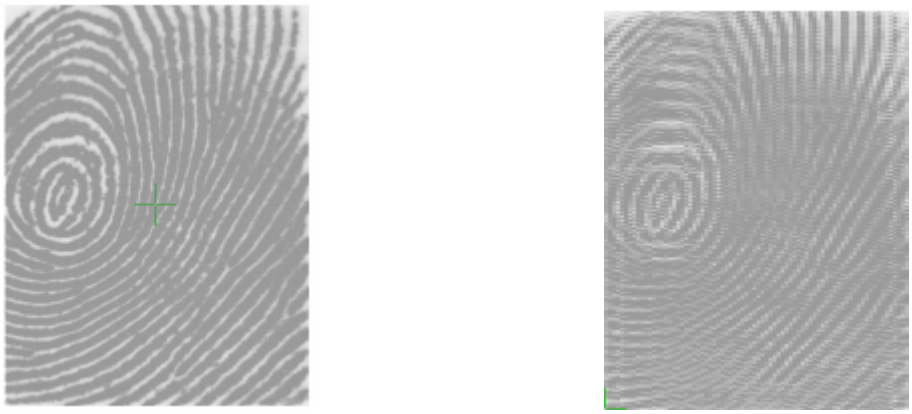
Procedimiento para la descompresión Hadamard 4:1

1) Cargar el archivo **Descompresión Hadamard3.out,** que esta ubicado en C:\ti\Proyectos de TESIS\Compresion HADAMARD\4-1\Descompresion Hadamard 4-1\Debug\, usando **File→Load Program.**

2) Cargar la imagen comprimida que se va a comprimir usando **File→Data→Load**, en el espacio Address escriba **dato,** el archivo esta ubicado en C:\Datos de Proyecto\imagenes\imagenes comprimidas\HADAMARD\4-1\Usuario_x\comprimidas (el valor x es el usuario, hay 10)**.**

3) Ejecute la compresión usando **Debug→Run.**

4) Guarde la imagen comprimida usando **File→Data→Save** en C:\Datos de Proyecto\imagenes\imagenes-comprimidas\HADAMARD\4-1\Usuario_x\descomprimidas\, en el dialogo **Storing Memory into File**, en la caja **Address** escriba **imagen** y en **Length** el número 33280.



a) b)

Figura 3.10: **Comparación entre a) imagen original b) Imagen Hadamard 4:1 descomprimida.**

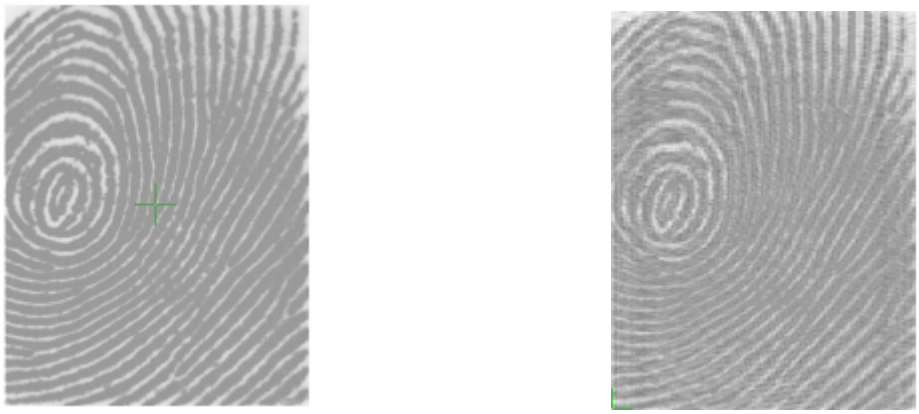
Procedimiento para la descompresión Hadamard 1,78:1

1) Cargar el archivo **Descompresión Hadamard3.out,** que esta ubicado en C:\ti\Proyectos de TESIS\Compresion HADAMARD\2-1\Descompresion Hadamard 2-1\Debug\, usando **File→Load Program.**

2) Cargar la imagen comprimida que se va a comprimir usando **File→Data→Load**, en el espacio **Address** escriba **dato.** El archivo esta ubicado en C:\Datos de Proyecto\imagenes\imagenes-comprimidas\HADAMARD\2-1\Usuario_x\comprimidas (el valor x es el usuario, hay 10)**.**

3) Ejecute la compresión usando **Debug→Run.**

4) Guarde la imagen comprimida usando **File→Data→Save** en C:\Datos de Proyecto\imagenes\imagenes-comprimidas\HADAMARD\2-1\Usuario_x\descomprimidas\, en el dialogo **Storing Memory into File**, en la caja **Address** escriba **imagen** y en **Length** el número 33280.



a) b)

Figura 3.11: **Comparación entre a) imagen original b) Imagen Hadamard 1.78:1 descomprimida.**

3.4 BINARIA

3.4.1 COMPRESIÓN BINARIA

Para implementar esta compresión en el CCS, se usaron las siguientes funciones (se muestran en el anexo K), además del código fuente principal:

Binarización: En esta función se binariza la imagen a comprimir.

Compresión: Contiene la función que empaqueta los bits que conforman la imagen Binarizada, guarda 8 píxeles por cada byte, almacena la imagen comprimida.

3.4.2 PROCEDIMIENTO PARA LA COMPRESIÓN BINARIA

1) Cargar el archivo **compresión binaria.out** que esta ubicado en C:\ti\Proyectos de TESIS\Compresion binaria\Compresion\Debug usando **File→Load Program.**

2) Cargar la imagen original que se va a comprimir usando **File→Data→Load**, en el espacio Address escriba **imagen.**

3) Ejecute la compresión usando **Debug→Run.**

4) Guarde la imagen comprimida usando **File→Data→Save,** dialogo **Storing Memory into File**, en la caja **Address** escriba **comp** y en **Length** el número 4160.

3.4.3 DESCOMPRESIÓN BINARIA

La descompresión Binaria implementada e indicada en el anexo L, usa un archivo principal y dos funciones, los cuales son:

Descompresión: Contiene la función que desempaqueta cada uno de los bits de la imagen comprimida y los convierte en bytes.

Decodificación: Contiene la función que permite asignar a cada píxel de la imagen un valor: blanco o negro para visualizar la imagen binaria.

3.4.4 PROCEDIMIENTO PARA LA DESCOMPRESIÓN BINARIA

1) Cargar el archivo **Descompresión.out,** que esta ubicado en C:\ti\Proyectos de TESIS\Compresion binaria\Descompresion\Debug\, usando **File→Load Program.**

2) Cargar la imagen comprimida que se va a comprimir usando **File→Data→Load**, en el espacio Address escriba **dato.** El archivo esta ubicado en C:\ti\Datos de Proyecto\imagenes\imagenes binarizadas\Usuario_x\comprimidas\ (el valor x es el usuario, hay 10)**.**

3) Ejecute la compresión usando **Debug→Run.**

4) Guarde la imagen comprimida usando **File→Data→Save** en C:\ti\Datos de Proyecto\imagenes\imagenes binarizadas\Usuario_x\descomprimidas\, en el dialogo **Storing Memory into File**, en la caja **Address** escriba **imagen** y en **Length** el número 33280.

a) b)

Figura 3.12: **Comparación entre a) imagen original b) Imagen Binaria descomprimida.**

3.5 PROCESADO DE IMAGEN DE HUELLA ORIGINAL

3.5.1 PROCESO DE CAPTURA DE LA IMAGEN ORIGINAL.

Para la captura de la imagen se procede usando los siguientes pasos:

1) Se enciende el Code Composer Studio desde el menú de Inicio o su icono en el escritorio.

2) Desde el menú **File**, se elige **Workspace**, luego se selecciona **Load Workspace,** y elijo el archivo fpc1010_dsk6713.wks desde la carpeta: C:\FADT\FPC1010_Sensor_Driver\dsk6713_fpc1010\demo\

3) Desde el menú **File**, elijo **Load Program** y selecciono fpc_1010_dsk6713.out.

4) Desde el menú **File**, elijo **Load GEL**, y selecciono el archivo gel llamado demo.gel

5) Desde el menú **Debug Menú**, elijo **Run**.

6) Para capturar la imagen, coloco el dedo a ser capturado sobre el sensor, luego desde el menú GEL, elijo **Demo Menú →CaptureFingerPrint.** Repite este paso para captura imágenes de huellas digitales adicionales.

7) El programa se parará temporalmente después de la captura de la imagen, y la imagen de la huella digital deberá mostrarse en la ventana de la pantalla

de gráficos. Si no, refresque la ventana de la pantalla mediante el procedimiento **Configuración de la pantalla de gráficos** (Anexo B).

8) Para salir de la demostración, elija **Gel→Demo Menú→Exit Demo.**

3.4.5 PROCEDIMIENTO PARA EL REGISTRO Y VERIFICACIÓN DE LAS HUELLAS

Para el registro y verificación de imágenes, se usará el demo embebido del FADT.

1) Cargue el archivo workspace **dad_fpc_demo.wsk** que se encuentra en C:\FADT\FPC_DAD_Verification_Algorithm_Demos\Embedded_Demo_6713DSK\fpc1010_embedded_demo dentro del Code Composer Studio usando **File→Workspace→Load Workspace**.

Esto automáticamente carga el proyecto **dad_fpc_demo.pjt**, así también cargara la ventana de pantalla de gráficos con la pre configuración del sensor FPC1010 y que habilita al usuario a ver la imagen de huella digital capturada dentro del CCS.

2) Cargar el archivo **dad_fpc_demo.out** usando **File→Load Program.**

3) Ejecute el demo embebido usando **Debug→Run**.

4) Para iniciar una nueva operación de captura/registro/verificación, elija una de las opciones en **Demo Menú (GEL→Demo Menú→*command).***

Las opciones para command son **CaptureFingerPrint**, **Enrol** y **Verify**. Cuando se elige estos comandos, el DSP se ejecuta en un lazo hasta que un dedo es colocado en el sensor. La imagen capturada puede ser vista en la ventana de pantalla de gráficos. Use el comando refresh desde menú que aparece apretando el clic derecho entre capturas para ver la ultima imagen de huella digital capturada sobre la ventana de gráficos.

Figura 3.13: **Led's indicadores de registro o verificación exitosa en el sensor.**

Una captura, registro o verificación satisfactoria es indicada al alumbrarse el led verde (LED 2) y un registro o verificación fallida es indicada al iluminarse el led rojo (LED 1) de la Daughter Card.

3.5.3 PROCEDIMIENTO PARA ALMACENAR UNA HUELLA REGISTRADA O CAPTURADA.

Para poder almacenar una huella en el disco duro del PC a través del CCS, se realizan los siguientes pasos:

1) Desde el menú **File**, elijo **Data→Save** y selecciono el lugar dentro del disco duro donde guardare los datos, colocando un nombre para su identificación y el tipo de dato que guardo, el archivo se guardara en el formato .dat.

2) Relleno el dialogo **Storing Memory into File**, en la caja **Address** con la dirección en donde se ubica la imagen (también puedo colocar el nombre de arreglo que contiene la imagen) y en la caja **Length** el tamaño de la imagen, y oprimo **OK.**

3.5.4 PROCEDIMIENTO PARA USAR UNA IMAGEN DE HUELLA PREVIAMENTE ALMACENADA

Para poder usar una imagen de huella previamente almacenada en el disco duro del PC a través del CCS, se realizan los siguientes pasos:

1) Desde el menú **File**, elijo **Data→Load** y selecciono el archivo que deseo observar colocando su nombre en la caja **Name**.

Relleno el dialogo **Loading Memory into File**; en la caja **Address,** con la dirección en donde deseo ubicar la imagen (también puedo colocar el nombre de arreglo que contiene la imagen) y en la caja **Length** aparecerá automáticamente el tamaño de la imagen, y oprimo **OK.**

Capítulo 4

ANÁLISIS Y RESULTADOS

Esta es la etapa más importante del proyecto, puesto que en si, es su principal objetivo. Para el análisis estadístico de las compresiones implementadas, no vamos a utilizar los criterios de fidelidad comunes para ellas, como lo que se desea es saber que tan fiable son estos métodos de compresión para poder realizar la identificación de personas usaremos las variables estadísticas: Tasa de Falsa Aceptación y Tasa de Falso Rechazo.

4.1 BASE DE DATOS

La base de datos permite poner en funcionamiento cada una de los métodos de compresión implementadas.

En este proyecto se han almacenado un total de 50 huellas digitales tomadas de 10 personas diferentes, tomando de cada unos cinco huellas al azar. La edad comprendida de estas personas oscila entre 22 y 41 años; mientras que por sexo, la cantidad de hombres y mujeres es la misma, es decir, cinco para cada grupo.

Cada huella fue capturada dos veces, la primera para obtener un patrón de la huella (proceso de registro) y la otra para capturar la imagen de la huella de manera total,

esta imagen es llamada Imagen Original, y es la que servirá para las compresiones que se llevarán a cabo en el siguiente paso del proyecto.

4.2 PROCEDIMIENTO PARA LA VERIFICACIÓN DE HUELLAS DESCOMPRIMIDAS

1) Cargar el archivo **Registra_verifica2.out,** que esta ubicado en C:\ti\Proyectos de TESIS\REGISTRA_VERIFICA_2\Debug\, usando **File→Load Program.**

2) Cargar el archivo **Registra_verifica2.gel,** que esta ubicado en C:\ti\Proyectos de TESIS\REGISTRA_VERIFICA_2\, usando **File→Load GEL.**

3) Para la verificación se necesita dos entradas, el patrón y la imagen de la huella, el patrón esta ubicado en C:\ti\Datos de Proyecto\patrones\Patron_x\, en el espacio **Address** escriba **p**. La imagen a cargar dependerá del archivo descomprimido que se desee verificar, en este caso, en el espacio **Address** escriba **i.**

4) Ejecute la compresión usando **Debug→Run,** luego **GEL→Demo Menú→verifica** y por ultimo coloque cualquier dedo sobre el sensor.

5) Si la verificación es correcta, entonces se ilumina el LED verde 0 y si es incorrecta se ilumina el LED verde 3, ambos led's forman parte de la placa del DSK.

4.3 COMPARACIÓN O ESTUDIO REALIZADOS

Para ser objetivo en nuestro análisis comparativo debemos tomar alguna referencia. Para el caso del proyecto, tomaremos como referencia la tasa de falsa aceptación y la tasa de falso rechazo de las imágenes originales. Un dato importante de resaltar es que las verificaciones que se realicen con las imágenes originales se realizarán con un nivel de seguridad o sensibilidad de 1/1000 o 0,1% de Falsa Aceptación (es preciso recordar que el nivel de máxima seguridad es de 1/100000 o 0,00001% de Falsa Aceptación). Todas las imágenes originales serán comprimidas usando los tres tipos de compresión implementados, luego cada una de estas imágenes serán descomprimidas, para inmediatamente ser verificadas. Todos los resultados serán agrupados por compresión y almacenados para luego calcular la Tasa de Falsa Aceptación y Tasa de Falso Rechazo de cada uno de estos grupos y poder definir una conclusión apropiada.

4.4 RESULTADO DE LA VERIFICACIÓN DE HUELLAS PROCESADAS

Las 50 imágenes originales fueron comprimidas en los tres métodos de compresión implementados, la descompresión de estas imágenes dio como resultado cuatro grupos de imágenes de huellas digitales descomprimidas (recordar, que se implementó la compresión Hadamard con dos tipo de niveles de compresión diferentes: 4:1 y 1,78:1).

Cada uno de estos grupos fueron independientemente verificados. Cada uno de los patrones de huella fueron comparados con las cincuenta huellas originales y las descomprimidas, lo que da como resultado **2500 comparaciones** por cada tipo de huellas (entre original y descomprimida).

Para el cálculo de la tasa de falsa aceptación, se toma como referencia la cantidad de comparaciones en donde teóricamente debe existir rechazo, en el caso de este análisis, seria de **2450 comparaciones**.

Para el cálculo de la tasa de falso rechazo, se toma como referencia la cantidad de comparaciones en donde teóricamente debe existir aceptación, en el caso de este análisis, seria de **50 comparaciones**.

Esto hace que en conjunto, entre verificación de huellas originales y descomprimidas, en comparación con los patrones de las huellas se llegue a un total de **12500 comparaciones**.

4.4.1 COMPRESIÓN JPEG

Este método de compresión alcanzo niveles de compresión que van de 6,2 a 15,2.

La aplicación de la etapa de verificación en donde comparamos huellas con los patrones que, teóricamente deben coincidir, da como resultado que 18 huellas digitales descomprimidas no satisfacen la verificación; lo cual indica que este método posee una tasa de falso rechazo de 36%.

Mientras que la comparación de huellas que, teóricamente no coinciden con el patrón de la huella, da como resultado cero huellas digitales que satisfacen la verificación; lo cual indica una tasa de falsa aceptación de 0%.

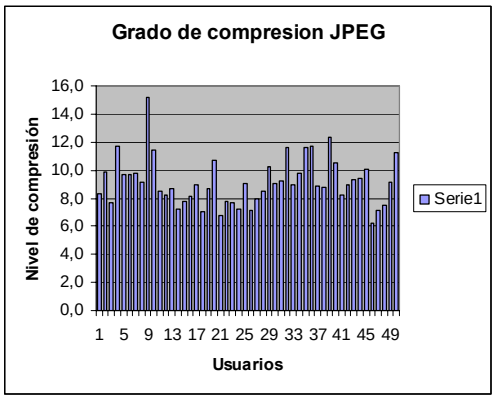


Figura 4.1: **Niveles de compresión en el método JPEG.**

4.4.2 COMPRESIÓN BINARIA

Este método de compresión tuvo un nivel de compresión de 8:1.

La aplicación de la etapa de verificación en donde comparamos huellas con los patrones que, teóricamente deben coincidir, da como resultado que una huella digital descomprimida no satisface la verificación; lo cual indica que este método posee una tasa de falso rechazo de 2%.

Mientras que la comparación de huellas que, teóricamente no coinciden con el patrón de la huella, da como resultado cero huellas digitales que satisfacen la verificación; lo cual indica una tasa de falsa aceptación de 0%.

4.4.3 COMPRESIÓN HADAMARD (PRIMERA IMPLEMENTACIÓN)

Este método de compresión tuvo un nivel de compresión de 4:1.

La aplicación de la etapa de verificación en donde comparamos huellas con los patrones que, teóricamente deben coincidir, da como resultado que 27 huellas digitales descomprimidas no satisfacen la verificación; lo cual indica que este método posee una tasa de falso rechazo de 54%.

Mientras que la comparación de huellas que, teóricamente no coinciden con el patrón de la huella, da como resultado cero huellas digitales que satisfacen la verificación; lo cual indica una tasa de falsa aceptación de 0%.

4.4.4 COMPRESIÓN HADAMARD (SEGUNDA IMPLEMENTACIÓN)

Este método de compresión tuvo un nivel de compresión de 1,78:1.

La aplicación de la etapa de verificación en donde comparamos huellas con los patrones que, teóricamente deben coincidir, da como resultado que 5 huellas digitales descomprimidas no satisfacen la verificación; lo cual indica que este método posee una tasa de falso rechazo de 10%.

Mientras que la comparación de huellas que, teóricamente no coinciden con el patrón de la huella, da como resultado cero huellas digitales que satisfacen la verificación; lo cual indica una tasa de falsa aceptación de 0%.

4.5 COMPARACIÓN DE LAS TASAS DE FALSA ACEPTACIÓN Y FALSO RECHAZO

MÉTODO DE COMPRESIÓN	Nivel de compresión	Tasa de falsos rechazos (FRR)	Tasa de falsas aceptaciones (FAR)
Hadamard 1º implementación.	*4*	*54%*	*0%*
JPEG	*6,2 – 15,2*	*36%*	*0%*
Binaria	*8*	*2%*	*0%*
Hadamard 2º implementación.	*1,78*	*10%*	*0%*

Tabla IV.I: Resumen de los resultados en relación a niveles de compresión, porcentajes FAR y FRR.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES.

- La compresión Binaria posee mayor nivel de seguridad, pues el FAR es del 2% y el FRR del 0%.

- La compresión JPEG contiene el mayor nivel de compresión, puesto que 36 de las 50 imágenes tienen un nivel de compresión mayor a ocho.

- Si el nivel de compresión de Hadamard disminuye a un 44,5% entonces el valor de FAR disminuye al 18, 5%, aumenta el nivel de seguridad del sistema.

- Para implementar un sistema de verificación remoto se usaría la compresión Binaria, por su nivel de seguridad.

RECOMENDACIONES

- Aumentar el tamaño de la base de datos para mejorar la eficiencia del cálculo de FAR y FRR.

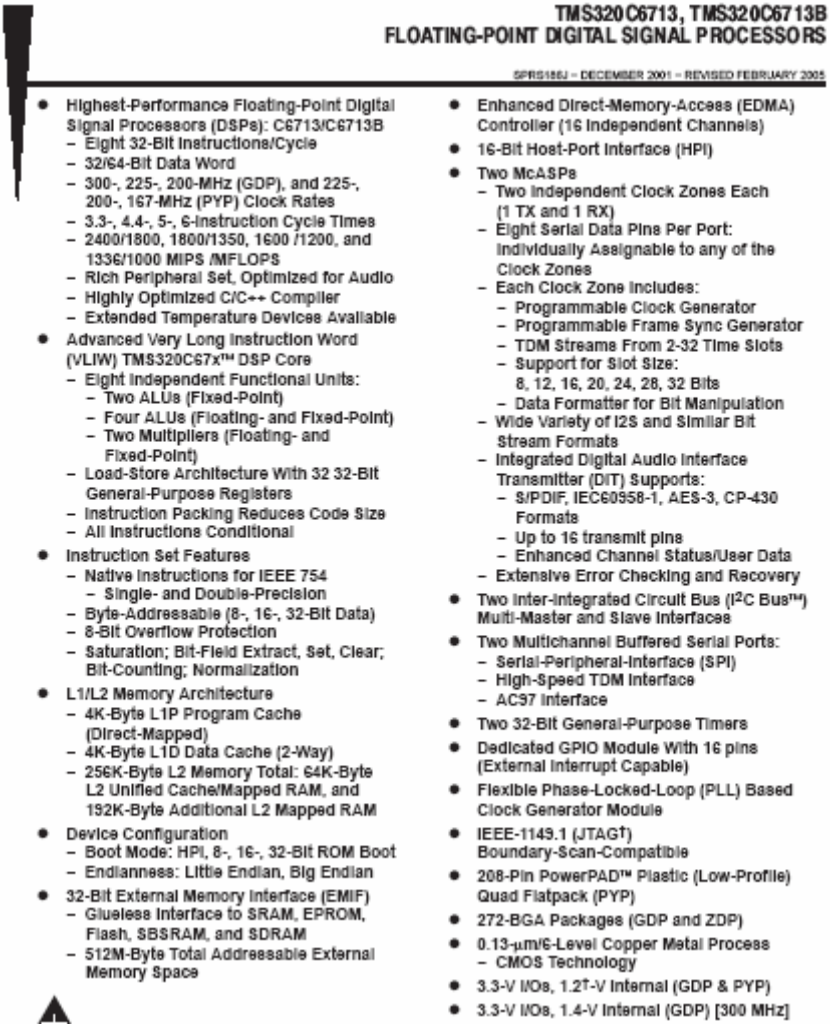
- Para mejorar el estudio, realizar la captura de los patrones de las imágenes descomprimidas pues es probable que las tasa de FAR bajen.
- Implementar la compresión binaria con el estándar Grupo 4 del CCITT.

- Futura implementación de este sistema de verificación e identificación a través de tecnología GPRS, en la cual un PDA seria un elemento de captura remota y el servidor, ubicado en un centro de información de una institución o empresa tendría almacenado los patrones para verificaciones o identificaciones.

ANEXOS

ANEXO A

INFORMACIÓN DEL DSP TMS320C6713.



TMS320C6713, TMS320C6713B
FLOATING-POINT DIGITAL SIGNAL PROCESSORS

SPRS186J – DECEMBER 2001 – REVISED FEBRUARY 2005

- Highest-Performance Floating-Point Digital Signal Processors (DSPs): C6713/C6713B
  - Eight 32-Bit Instructions/Cycle
  - 32/64-Bit Data Word
  - 300-, 225-, 200-MHz (GDP), and 225-, 200-, 167-MHz (PYP) Clock Rates
  - 3.3-, 4.4-, 5-, 6-Instruction Cycle Times
  - 2400/1800, 1800/1350, 1600 /1200, and 1336/1000 MIPS /MFLOPS
  - Rich Peripheral Set, Optimized for Audio
  - Highly Optimized C/C++ Compiler
  - Extended Temperature Devices Available
- Advanced Very Long Instruction Word (VLIW) TMS320C67x™ DSP Core
  - Eight Independent Functional Units:
    - Two ALUs (Fixed-Point)
    - Four ALUs (Floating- and Fixed-Point)
    - Two Multipliers (Floating- and Fixed-Point)
  - Load-Store Architecture With 32 32-Bit General-Purpose Registers
  - Instruction Packing Reduces Code Size
  - All Instructions Conditional
- Instruction Set Features
  - Native Instructions for IEEE 754
    - Single- and Double-Precision
  - Byte-Addressable (8-, 16-, 32-Bit Data)
  - 8-Bit Overflow Protection
  - Saturation; Bit-Field Extract, Set, Clear; Bit-Counting; Normalization
- L1/L2 Memory Architecture
  - 4K-Byte L1P Program Cache (Direct-Mapped)
  - 4K-Byte L1D Data Cache (2-Way)
  - 256K-Byte L2 Memory Total: 64K-Byte L2 Unified Cache/Mapped RAM, and 192K-Byte Additional L2 Mapped RAM
- Device Configuration
  - Boot Mode: HPI, 8-, 16-, 32-Bit ROM Boot
  - Endianness: Little Endian, Big Endian
- 32-Bit External Memory Interface (EMIF)
  - Glueless Interface to SRAM, EPROM, Flash, SBSRAM, and SDRAM
  - 512M-Byte Total Addressable External Memory Space
- Enhanced Direct-Memory-Access (EDMA) Controller (16 Independent Channels)
- 16-Bit Host-Port Interface (HPI)
- Two McASPs
  - Two Independent Clock Zones Each (1 TX and 1 RX)
  - Eight Serial Data Pins Per Port: Individually Assignable to any of the Clock Zones
  - Each Clock Zone Includes:
    - Programmable Clock Generator
    - Programmable Frame Sync Generator
    - TDM Streams From 2-32 Time Slots
    - Support for Slot Size: 8, 12, 16, 20, 24, 28, 32 Bits
    - Data Formatter for Bit Manipulation
  - Wide Variety of I2S and Similar Bit Stream Formats
  - Integrated Digital Audio Interface Transmitter (DIT) Supports:
    - S/PDIF, IEC60958-1, AES-3, CP-430 Formats
    - Up to 16 transmit pins
    - Enhanced Channel Status/User Data
  - Extensive Error Checking and Recovery
- Two Inter-Integrated Circuit Bus (I²C Bus™) Multi-Master and Slave Interfaces
- Two Multichannel Buffered Serial Ports:
  - Serial-Peripheral-Interface (SPI)
  - High-Speed TDM Interface
  - AC97 Interface
- Two 32-Bit General-Purpose Timers
- Dedicated GPIO Module With 16 pins (External Interrupt Capable)
- Flexible Phase-Locked-Loop (PLL) Based Clock Generator Module
- IEEE-1149.1 (JTAG†) Boundary-Scan-Compatible
- 208-Pin PowerPAD™ Plastic (Low-Profile) Quad Flatpack (PYP)
- 272-BGA Packages (GDP and ZDP)
- 0.13-µm/6-Level Copper Metal Process
  - CMOS Technology
- 3.3-V I/Os, 1.2†-V Internal (GDP & PYP)
- 3.3-V I/Os, 1.4-V Internal (GDP) [300 MHz]

Hoja de datos del TMS320C6713 DSP

MEMORY BLOCK DESCRIPTION	BLOCK SIZE (BYTES)	HEX ADDRESS RANGE
Internal RAM (L2)	192K	0000 0000 – 0002 FFFF
Internal RAM/Cache	64K	0003 0000 – 0003 FFFF
Reserved	24M – 256K	0004 0000 – 017F FFFF
External Memory Interface (EMIF) Registers	256K	0180 0000 – 0183 FFFF
L2 Registers	128K	0184 0000 – 0185 FFFF
Reserved	128K	0186 0000 – 0187 FFFF
HPI Registers	256K	0188 0000 – 018B FFFF
McBSP 0 Registers	256K	018C 0000 – 018F FFFF
McBSP 1 Registers	256K	0190 0000 – 0193 FFFF
Timer 0 Registers	256K	0194 0000 – 0197 FFFF
Timer 1 Registers	256K	0198 0000 – 019B FFFF
Interrupt Selector Registers	512	019C 0000 – 019C 01FF
Device Configuration Registers	4	019C 0200 – 019C 0203
Reserved	256K – 516	019C 0204 – 019F FFFF
EDMA RAM and EDMA Registers	256K	01A0 0000 – 01A3 FFFF
Reserved	768K	01A4 0000 – 01AF FFFF
GPIO Registers	16K	01B0 0000 – 01B0 3FFF
Reserved	240K	01B0 4000 – 01B3 FFFF
I2C0 Registers	16K	01B4 0000 – 01B4 3FFF
I2C1 Registers	16K	01B4 4000 – 01B4 7FFF
Reserved	16K	01B4 8000 – 01B4 BFFF
McASP0 Registers	16K	01B4 C000 – 01B4 FFFF
McASP1 Registers	16K	01B5 0000 – 01B5 3FFF
Reserved	160K	01B5 4000 – 01B7 BFFF
PLL Registers	8K	01B7 C000 – 01B7 DFFF
Reserved	264K	01B7 E000 – 01BB FFFF
Emulation Registers	256K	01BC 0000 – 01BF FFFF
Reserved	4M	01C0 0000 – 01FF FFFF
QDMA Registers	52	0200 0000 – 0200 0033
Reserved	16M – 52	0200 0034 – 02FF FFFF
Reserved	720M	0300 0000 – 2FFF FFFF
McBSP0 Data Port	64M	3000 0000 – 33FF FFFF
McBSP1 Data Port	64M	3400 0000 – 37FF FFFF
Reserved	64M	3800 0000 – 3BFF FFFF
McASP0 Data Port	1M	3C00 0000 – 3C0F FFFF
McASP1 Data Port	1M	3C10 0000 – 3C1F FFFF
Reserved	1G + 62M	3C20 0000 – 7FFF FFFF
EMIF CE0†	256M	8000 0000 – 8FFF FFFF
EMIF CE1†	256M	9000 0000 – 9FFF FFFF
EMIF CE2†	256M	A000 0000 – AFFF FFFF
EMIF CE3†	256M	B000 0000 – BFFF FFFF
Reserved	1G	C000 0000 – FFFF FFFF

† The number of EMIF address pins (EA[21:2]) limits the maximum addressable memory (SDRAM) to 128MB per CE space.

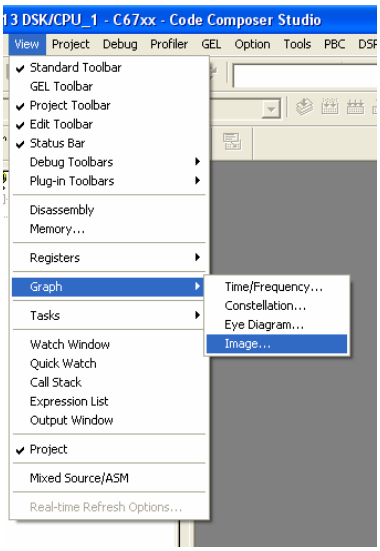
Resumen del mapa de memoria del TMS320C6713 DSP

ANEXO B

CONFIGURACIÓN DE LA PANTALLA DE GRÁFICOS

Si la imagen no logra verse de manera automática, existe la opción de tipo manual, este sigue los siguientes pasos:

1) Desde el menú **View**, elija **Graph→Image** para abrir la ventana de pantalla de gráficos.

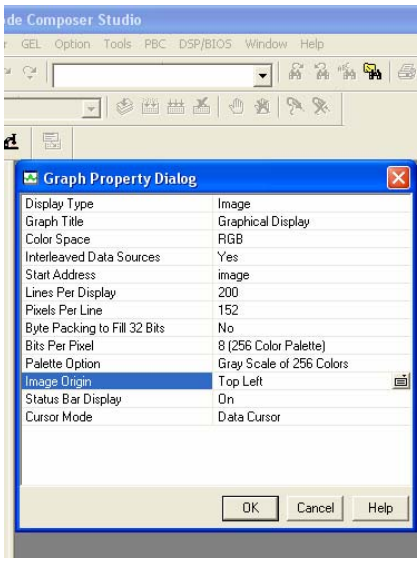


Selección para la configuración de pantalla de gráficos.

2) Rellene las cajas del **Graph Property Dialog** con la siguiente configuración para el sensor FPC1010.

Graph Property Dialog Box Settings:	
Color space:	**RGB**
Interleaved data:	**YES**
Start address:	**image** (image array defined in code)
Lines per display:	**200**
Pixels per line:	**152**
Byte packing to fill 32 bits:	**NO**
Bits per pixel:	**8**
Palette Option:	**Gray scale of 256 colors**
Image Origin:	**Top Left**
Everything else:	**Default**

Datos del sensor FPC1010 para la configuración de la pantalla.



Configuración del CCS con los datos del sensor.

ANEXO C

CÓDIGO FUENTE PARA CAPTURA DE LA IMAGEN ORIGINAL.

NOMBRE: main.c
DESCRIPCIÓN: Archivo principal para leer la imagen completa desde el sensor FPC1010.

```
#include <csl.h>
#include <csl_mcbsp.h>
#include <stdio.h>
#include "dsk6713.h"
#include "fpc1010.h"
#include "spi.h"
#pragma DATA_SECTION (image, ".picture");
unsigned char image[208][160];
unsigned int captureFP=0,exitDemo=0;

void dummyFunc(void)
{
   captureFP=0;
}

void main(void)
{
 unsigned char        *imageptr;
 DSK6713_init();
 CSL_init();
 DSK6713_rset(DSK6713_MISC , MCBSP1SEL);
 SPI_init(2884615,0);
 printf (" To capture fingerprint: \n");
 printf (" Place your finger on the sensor \n");
 printf (" Under Menú GEL , choose>> DemoMenú->CaptureFingerPrint \n");
 printf (" ================================================== \n");*/
 imageptr = &image[0][0];
 waitForFp:
   while (!captureFP)
   {
    if(exitDemo)
    {
     printf (" Exiting the Demo for FPC1010 on 6713DSK \n");
     exit();
    }
   }
   sensorInit();
   printf (" Capturing Image...\n");
   readImage(imageptr);
   printf (" Reading Done. You may remove your finger.\n");
```

```
    printf (" ****************************************************** \n");
    printf (" To recapture fingerprint: \n");
    printf (" Place your finger on the sensor \n");
    printf (" GEL>> DemoMenú->CaptureFingerPrint \n");
    printf (" ======================================================= \n");
    printf (" To Exit Demo: \n");
    printf (" Under Menú GEL , choose>> DemoMenú->ExitDemo \n");
    printf (" ======================================================= \n");
    dummyFunc();
    goto waitForFp;
}
```

NOMBRE: fpc1010.c
DESCRIPCION: El archivo describe diferentes funciones para leer la imagen desde el sensor FPC1010.

```
#include <csl.h>
#include <csl_mcbsp.h>
#include "fpc1010.h"
#include "spi.h"

/* Nombre Función: Sensor_init
 * Descripción       : Inicializa el Sensor */

void sensorInit(void)
{
 /* Hardware se reinicia por 1 micro seg.*/
 SPI_RESET;
 SPI_SET;
 /* Inicialización (parte1) */
 SPI_ENABLE;
 SPI_write(RESET);
 SPI_DISABLE;
 SPI_ENABLE;
 SPI_write(SENSX);
 SPI_write(0x01);
 SPI_DISABLE;
 /* Inicialización (parte2) */
 SPI_ENABLE;
 SPI_write(SENSY);
 SPI_write(0x01);
 SPI_DISABLE;
 SPI_ENABLE;
 SPI_write(COMMAND);
 SPI_write(0x08);
 SPI_DISABLE;
}

/* Nombre Función: Read_Image
 * Descripción   : Leer la imagen completa */

void readImage(unsigned char *image_ptr)
{
 int numRows,numCols,i,j;
 unsigned char pix1, temp;
 unsigned volatile char dummy[6];
 unsigned char *temp_ptr;
 /* Read all 200 Rows */
 for(numRows=0;numRows<MAXROW  ;numRows++)
 {
  temp_ptr = image_ptr;
  for(j=0;j<3;j++)
  {
   image_ptr = temp_ptr;
   SPI_ENABLE;
```

```
  SPI_write(READ_SHFTX);
  /* Ignore 32 clocks */
  for(i=0;i<5;i++)
  dummy[i]=SPI_read();
  /* Capture 152 8bit pixel in each row */
  for(numCols=0; numCols<MAXCOLUMN;numCols++)
  {
   pix1 = SPI_read();
   if(j==0)
   {
    *image_ptr++ = pix1;
   }
   else if(j==1)
   {
    *image_ptr++ += pix1;
   }
   if(j==2)
   {
    temp = *image_ptr + pix1;
    *image_ptr++ = ~temp;
   }
  }
  /* Ignore 16 clocks */
  for(i=0;i<2;i++)
  dummy[i]=SPI_read();
  SPI_DISABLE;
 }
 SPI_ENABLE;
 SPI_write(READ_SHFTY);
 SPI_DISABLE;
}
}
```

ANEXO D

CÓDIGO FUENTE PARA REGISTRO Y VERIFICACIÓN DE HUELLAS.

NOMBRE: main.c
DESCRIPCIÓN: El archivo describe como llamar a las diferentes funciones disponibles en la Fingerprint Authentical Algorithm API.
También suministra ejemplos para usar los LEDs en la daughter card etc.

```
#include <std.h>
#include <csl.h>
#include <csl_mcbsp.h>
#include <csl_cache.h>
#include "dad.h"
#include "dad_fpc.h"
#include "dsk6713.h"
#include "dsk6713_led.h"
#include "fpc1010.h"
#include "spi.h"

#define HIGH_TRANSLATION_TOLERANCE                  0
#define NORMAL_TRANSLATION_TOLERANCE 1
#define NORMAL_SECURITY             0
#define HIGH_SECURITY               1
#define VERY_HIGH_SECURITY          2

#pragma DATA_SECTION (image, "picture");
static XDAS_UInt8 image[208][160];
#pragma DATA_SECTION (template, "picture");
#pragma DATA_ALIGN (template, 4);
static unsigned char template[1040];
unsigned volatile int captureFP=0, enrollFP=0, verifyFP=0, exitDemo=0;

void dummyCapture(void)
{
   captureFP=0;
}
void dummyEnroll(void)
{
   enrollFP=0;
}
void dummyVerify(void)
{
   verifyFP=0;
}
```

```
void main()
{
 DAD_Handle  handle;
 IDAD_Fxns   fxns;
 DAD_Params  params;

 XDAS_UInt8         *imageptr;
 XDAS_Bool          Reg_OK;
 XDAS_Bool          Ver_OK;
 XDAS_Bool          firstAttempt;
 XDAS_Bool   fingerpresent;

 CACHE_L2Mode OldMode;
 DSK6713_init();
 DSK6713_LED_init();
 CSL_init();
 OldMode = CACHE_setL2Mode(CACHE_16KCACHE);
 CACHE_enableCaching(CACHE_CE00);

 imageptr = &image[0][0];

 fxns = DAD_FPC_IDAD;
 params = DAD_PARAMS;
 params.securitylevel = NORMAL_SECURITY;

 DSK6713_rset(DSK6713_MISC , MCBSP1SEL);
 SPI_init(2884615,0);
 DAD_init();

 if((handle = DAD_create(&fxns, &params)) != NULL)
 {
  DSK6713_rset(DSK6713_DC_REG,3);

waitForFp:
  while (!captureFP && !enrollFP && !verifyFP && !exitDemo);
    DSK6713_rset(DSK6713_DC_REG,3);
    imageptr = &image[0][0];
    fingerpresent = FALSE;
    while(!fingerpresent && !exitDemo)
    {
     sensorInit();
     readImage(imageptr);
     fingerpresent = DAD_checkIfFinger(handle,imageptr);
    }
    DSK6713_rset(DSK6713_DC_REG,3);
    imageptr = &image[0][0];

    if(captureFP)
    {
     DSK6713_rset(DSK6713_DC_REG,1);
     dummyCapture();
     goto waitForFp;
    }

    if(enrollFP)
    {
     Reg_OK = DAD_enroll(handle, imageptr, template );
     if(Reg_OK)
     {
      DSK6713_rset(DSK6713_DC_REG,1);
     }
     else
     {
      DSK6713_rset(DSK6713_DC_REG,2);
     }
     dummyEnroll();
```

```
    goto waitForFp;
   }

   if(verifyFP)
   {
    firstAttempt = TRUE;
    Ver_OK = DAD_verify(handle, imageptr, template, HIGH_TRANSLATION_TOLERANCE,firstAttempt);
    if(Ver_OK)
    {
     DSK6713_rset(DSK6713_DC_REG,1);
    }
    else
    {
     DSK6713_rset(DSK6713_DC_REG,2);
    }
    dummyVerify();
    goto waitForFp;
   }
   DSK6713_rset(DSK6713_DC_REG,3);
   DAD_delete(handle);
   DAD_exit();
 exit();
 }
}
```

ANEXO E

CÓDIGO FUENTE PARA COMPRESIÓN JPEG.

NOMBRE: compresión JPEG.c
DESCRIPCIÓN: Archivo principal que contiene la implementación de la compresión JPEG.

```
#pragma DATA_SECTION(imagen,"picture");
#pragma DATA_SECTION(numero_bits_AC,"picture");
#pragma DATA_SECTION(compresion_JPEG,"picture");
#pragma DATA_SECTION(huffman,"picture");
#pragma DATA_SECTION(numero_bits,"picture");
#pragma DATA_SECTION(codigo1,"picture");
unsigned char imagen [208][160];
float Transfor[8][8];
float estandar[8][8];
float temp[8][8];
float temporal[8][8];
int codigo[64];
int valores_DC[520];
int temp_2[64];
int huffman_AC [18320];
int huffman_DC[520];
int huffman [18320];
int numero_bits_AC[18320];
int numero_bits[18320];
int numero_bits_DC[520];
int salida1,temp3;
unsigned char compresion_JPEG[18320];
int a,b,c,k=0,l=0,n=0,p,s=0;
unsigned char *color;

float DCT [8][8]={0.3536,0.3536,0.3536,0.3536,0.3536,0.3536,0.3536,0.3536,
                  0.4904,0.4157,0.2778,0.0975,-0.0975,-0.2778,-0.4157,-0.4904,
                  0.4619,0.1913,-0.1913,-0.4619,-0.4619,-0.1913,0.1913,0.4619,
                  0.4157,-0.0975,-0.4904,-0.2778,0.2778,0.4904,0.0975,-0.4157,
                  0.3536,-0.3536,-0.3536,0.3536,0.3536,-0.3536,-0.3536,0.3536,
                  0.2778,-0.4904,0.0975,0.4157,-0.4157,-0.0975,0.4904,-0.2778,
                  0.1913,-0.4619,0.4619,-0.1913,-0.1913,0.4619,-0.4619,0.1913,
                  0.0975,-0.2778,0.4157,-0.4904,0.4904,-0.4157,0.2778,-0.0975};
float DCT_t[8][8]={0.3536,0.4904,0.4619,0.4157,0.3536,0.2778,0.1913,0.0975,
                   0.3536,0.4157,0.1913,-0.0975,-0.3536,-0.4904,-0.4619,-0.2778,
                   0.3536,0.2778,-0.1913,-0.4904,-0.3536,0.0975,0.4619,0.4157,
                   0.3536,0.0975,-0.4619,-0.2778,0.3536,0.4157,-0.1913,-0.4904,
                   0.3536,-0.0975,-0.4619,0.2778,0.3536,-0.4157,-0.1913,0.4904,
                   0.3536,-0.2778,-0.1913,0.4904,-0.3536,-0.0975,0.4613,-0.4157,
                   0.3536,-0.4157,0.1913,0.0975,-0.3536,0.4904,-0.4613,0.2778,
                   0.3536,-0.4904,0.4619,-0.4157,0.3536,-0.2778,0.1913,-0.0975};
```

```
void main()
{
 color=&imagen[0][0];
 for(a=0;a<520;a++)
 {
  for (b=0;b<8;b++)
  {
   for (c=0;c<8;c++)
   {
    temp[b][c]=(float)color[64*a+8*b+c];
   }
  }
  Resta(&temp[0][0],&temporal[0][0]);
  Transformacion_DCT(&DCT[0][0],&temporal[0][0],&DCT_t[0][0],&Transfor[0][0]);
  Cuantificacion(&Transfor[0][0],&estandar[0][0]);
  Zig_zag(&estandar[0][0],codigo);
  Inversion(codigo,temp_2);
  p=Ceros(temp_2);
  k=codigo[0]-l;
  Valores_DC(&k,valores_DC,&a);
  Huffman_AC(codigo,huffman_AC,numero_bits_AC,&p,&n);
  l=codigo[0];
 }
 Huffman_DC(valores_DC,huffman_DC,numero_bits_DC);
 Huffman(huffman_DC,huffman_AC,numero_bits_DC,numero_bits_AC,&n,huffman,numero_bits);
 Relleno_ceros(compresion_JPEG);
 Almacenamiento(huffman,numero_bits,compresion_JPEG,&n,&s);
 exit();
}
```

NOMBRE: Resta 128.c
DESCRIPCION: El archivo contiene a la funcion que permite resta 128 a cada valor de la imagen a la cual se aplicara la compresión JPEG.

```
void Resta (float*in,float*out)
{
 int i,j;
 for(i=0;i<8;i++)
 {
  for(j=0;j<8;j++)
  {
   out[8*i+j]=in[8*i+j]-128;
  }
 }
}
```

NOMBRE: Transformación Coseno Discreta.c
DESCRIPCION: El archivo contiene a la función que permite la aplicación de la DCT.

```
void Transformacion_DCT(float*DCT,float*temp,float*DCT_t,float*Transfor)
{
 unsigned int i,j,l,x,y,z;
 float D[8][8],sum,suma;
 for (i=0;i<8;i++)
 {
  for (j=0;j<8;j++)
  {
   sum=0;
   for (l=0;l<8;l++)
   {
    sum += DCT[l+i*8] * temp[j+l*8];
    D[i][j]=sum;
   }
  }
```

```
 }
 for (x=0;x<8;x++)
 {
  for (y=0;y<8;y++)
  {
   suma=0;
   for (z=0;z<8;z++)
   {
    suma +=D[x][z] * DCT_t[y+z*8];
    Transfor[y+x*8]=suma;
   }
  }
 }
}
```

NOMBRE: Cuantificación JPEG.c
DESCRIPCION: El archivo contiene a la función que permite la cuantificación de la imagen a comprimir.

```
#include <math.h>
#define value 8
float normal [value][value]={ 8, 5, 5, 8,12,20,25,30,
                               6, 6, 7, 9,13,29,30,27,
                               7, 6, 8,12,20,28,34,28,
                               7, 8,11,14,25,43,40,31,
                               9,11,18,28,34,54,51,38,
                              12,17,27,32,40,52,56,46,
                              24,32,39,43,51,60,60,50,
                              36,46,47,49,56,50,51,49};
void Cuantificacion(float *entrada, float *estandar)
{
 int d,e;
 for(d=0;d<value;d++)
 {
  for(e=0;e<value;e++)
  {
   estandar[8*d+e] =((entrada[8*d+e])/(normal[d][e]));
   if(estandar[8*d+e] <0)
   {
    estandar[8*d+e] = ceil(estandar[8*d+e]);
   }
   else
   {
    estandar[8*d+e] = floor(estandar[8*d+e]);
   }
  }
 }
}
```

NOMBRE: Zig zag.c
DESCRIPCION: El archivo contiene a la función que permite el reordenamiento en zigzag de la imagen anteriormente cuantificada, en este paso la imagen de una matriz 8x8 se convierte en una matriz 64x1.

```
void Zig_zag(float*in,int*out)
{
 int posicion[64]={0,1,8,16,9,2,3,10,
                   17,24,32,25,18,11,4,5,
                   12,19,26,33,40,48,41,34,
                   27,20,13,6,7,14,21,28,
                   35,42,49,56,57,50,43,36,
                   29,22,15,23,30,37,44,51,
                   58,59,52,45,38,31,39,46,
                   53,60,61,54,47,55,62,63};
```

```
 int i,j=0;
 for(i=0;i<64;i++)
 {
  j=posicion[i];
  out[i]=in[j];
 }
}
```

NOMBRE: Inversión.c
DESCRIPCION: El archivo contiene a la función que permite que el ultimo elemento del vector de 64 elementos pase a ser el primero, y el segundo elemento pasa a ser el penúltimo y así se sigue la secuencia con los demás elementos.

```
void Inversion (int*in,int*out)
{
 int i;
 for(i=0;i<64;i++)
 {
  out[63-i]=in[i];
 }
}
```

NOMBRE: Ceros.c
DESCRIPCION: El archivo contiene a la función que cuenta la cantidad de ceros presente en el vector de 64 elementos.

```
int Ceros(int*in)
{
 int i,n=0;
 for(i=0;i<64;i++)
 {
  if(in[i]==0)
  {
   n=n+1;
  }
  else
  {
   return(n);
  }
 }
}
```

NOMBRE: Valores_DC.c
DESCRIPCION: El archivo contiene a la función que almacena todos los coeficientes DC de cada una de las submatrices 8x8, lo guarda en un vector de 520 elementos.

```
void Valores_DC(int*in,int*out,int*n)
{
 int k;
 k=*n;
 out[k]=*in;
}
```

NOMBRE: Algoritmo Huffman_AC.c
DESCRIPCION: El archivo contiene a la función que codifica todos los valores AC de cada una de las submatrices 8x8 con el procedimiento del Algoritmo de Huffman.

```
void Huffman_AC(int*in,int*out,int*bits,int*p,int*s)
{
 int salida;
 int b=0,i,j=0,k=0,m=0,d=0,temp,a,l=-1,aux=0,bit=0,numero;
```

```
 m=*s;
 for(i=1;i<=64-(*p);i++)
 {
  if(i>0&&i<(64-(*p)))        //CODIGO AC JPEG
(luminancia)
  {
   k=in[i];
   if(k==0)
   {
    if(j<16)
    {
     j=j+1;
     goto salto;
    }
    if(j==16)
    {
     temp=4087;
     bit=12;
     j=0;
     l=l+1;
     d=d+1;
    }
   }
   if(k!=0)
   {
    if(j==0)
    {
     if(k==-1||k==1)
     {
      temp=0;
      aux=~-2;
      bit=3;
     }
     if(k==-3||k==-2||k==2||k==3)
     {
      temp=4;
      aux=~-4;
      bit=4;
     }
     if((k>=-7&&k<=-4)||(k>=4&&k<=7))
     {
      temp=32;
      aux=~-8;
      bit=6;
     }
     if((k>=-15&&k<=-8)||(k>=8&&k<=15))
     {
      temp=176;
      aux=~-16;
      bit=8;
     }
     if((k>=-31&&k<=-16)||(k>=16&&k<=31))
     {
      temp=832;
      aux=~-32;
      bit=10;
     }
     if((k>=-63&&k<=-32)||(k>=32&&k<=63))
     {
      temp=3584;
      aux=~-64;
      bit=12;
     }
     if((k>=-127&&k<=-64)||(k>=64&&k<=127))
     {
      temp=15360;
      aux=~-128;
      bit=14;
     }
     if((k>=-255&&k<=-128)||(k>=128&&k<=255))
     {
      temp=259584;
      aux=~-256;
      bit=18;
     }
     if((k>=-511&&k<=-256)||(k>=256&&k<=511))
     {
      temp=33489920;
      aux=~512;
      bit=25;
     }
     if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
     {
      temp=66980864;
      aux=~-1024;
      bit=26;
     }
    }
    if(j==1)
    {
     if(k==-1||k==1)
     {
      temp=24;
      aux=~-2;
      bit=5;
     }
     if(k==-3||k==-2||k==2||k==3)
     {
      temp=228;
      aux=~-4;
      bit=8;
     }
     if((k>=-7&&k<=-4)||(k>=4&&k<=7))
     {
      temp=968;
      aux=~-8;
      bit=10;
     }
     if((k>=-15&&k<=-8)||(k>=8&&k<=15))
     {
      temp=8032;
      aux=~-16;
      bit=13;
     }
     if((k>=-31&&k<=-16)||(k>=16&&k<=31))
     {
      temp=65216;
      aux=~-32;
      bit=16;
     }
     if((k>=-63&&k<=-32)||(k>=32&&k<=63))
     {
      temp=4186368;
      aux=~-64;
      bit=22;
     }
     if((k>=-127&&k<=-64)||(k>=64&&k<=127))
     {
      temp=8372864;
      aux=~-128;
      bit=23;
     }
```

```
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16745984;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33492480;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=133963776;
  aux=~-1024;
  bit=26;
 }
}
if(j==2)
{
 if(k==-1||k==1)
 {
  temp=54;
  aux=~-2;
  bit=6;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=992;
  aux=~-4;
  bit=10;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=8120;
  aux=~-8;
  bit=13;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1046672;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2093376;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4186816;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8373760;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16747776;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33496064;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=66993152;
  aux=~-1024;
  bit=26;
 }
}
if(j==3)
{
  if(k==-1||k==1)
 {
  temp=116;
  aux=~-2;
  bit=7;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=2012;
  aux=~-4;
  bit=11;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=16312;
  aux=~-8;
  bit=14;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1046784;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2093600;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4187264;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8374656;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16749568;
  aux=~-256;
  bit=24;
 }
```

```
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33499648;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67000320;
  aux=~-1024;
  bit=26;
 }
}
if(j==4)
{
 if(k==-1||k==1)
 {
  temp=118;
  aux=~-2;
  bit=7;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=4064;
  aux=~-4;
  bit=12;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=523448;
  aux=~-8;
  bit=19;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1046912;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2093856;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4187776;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8375680;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16751616;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33503744;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67008512;
  aux=~-1024;
  bit=26;
 }
}
if(j==5)
{
 if(k==-1||k==1)
 {
  temp=244;
  aux=~-2;
  bit=8;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=4068;
  aux=~-4;
  bit=12;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=523512;
  aux=~-8;
  bit=19;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1047040;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2094112;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4188288;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8376704;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16753664;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33507840;
  aux=~-512;
  bit=25;
 }
```

```
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67016704;
  aux=~-1024;
  bit=26;
 }
}
if(j==6)
{
 if(k==-1||k==1)
 {
  temp=246;
  aux=~-2;
  bit=8;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=8160;
  aux=~-4;
  bit=13;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=523576;
  aux=~-8;
  bit=19;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1047168;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2094368;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4188800;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8377728;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16755712;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33511936;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67024896;
  aux=~-1024;
  bit=26;
 }
}
if(j==7)
{
 if(k==-1||k==1)
 {
  temp=498;
  aux=~-2;
  bit=9;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=8164;
  aux=~-4;
  bit=13;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=523640;
  aux=~-8;
  bit=19;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1047296;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2094624;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4189312;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8378752;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16757760;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33516032;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67033088;
  aux=~-1024;
  bit=26;
 }
```

```
}
if(j==8)
{
 if(k==-1||k==1)
 {
  temp=500;
  aux=~-2;
  bit=9;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=130816;
  aux=~-4;
  bit=17;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=523704;
  aux=~-8;
  bit=19;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1047424;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2094880;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4189824;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8379776;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16759808;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33520128;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67041280;
  aux=~-1024;
  bit=26;
 }
}
if(j==9)
{
 if(k==-1||k==1)
 {
  temp=1008;
  aux=~-2;
  bit=10;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=261884;
  aux=~-4;
  bit=18;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=523776;
  aux=~-8;
  bit=19;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1047568;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2095168;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4190400;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8380928;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16762112;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33524736;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67050496;
  aux=~-1024;
  bit=26;
 }
}
if(j==10)
{
 if(k==-1||k==1)
 {
  temp=1010;
```

```
 aux=~-2;
 bit=10;
}
if(k==-3||k==-2||k==2||k==3)
{
 temp=261920;
 aux=~-4;
 bit=18;
}
if((k>=-7&&k<=-4)||(k>=4&&k<=7))
{
 temp=523848;
 aux=~-8;
 bit=19;
}
if((k>=-15&&k<=-8)||(k>=8&&k<=15))
{
 temp=1047712;
 aux=~-16;
 bit=20;
}
if((k>=-31&&k<=-16)||(k>=16&&k<=31))
{
 temp=2095456;
 aux=~-32;
 bit=21;
}
if((k>=-63&&k<=-32)||(k>=32&&k<=63))
{
 temp=4190976;
 aux=~-64;
 bit=22;
}
if((k>=-127&&k<=-64)||(k>=64&&k<=127))
{
 temp=8382080;
 aux=~-128;
 bit=23;
}
if((k>=-255&&k<=-128)||(k>=128&&k<=255))
{
 temp=16764416;
 aux=~-256;
 bit=24;
}
if((k>=-511&&k<=-256)||(k>=256&&k<=511))
{
 temp=33529344;
 aux=~-512;
 bit=25;
}
if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
{
 temp=67059712;
 aux=~-1024;
 bit=26;
}
}
if(j==11)
{
 if(k==-1||k==1)
 {
  temp=1012;
  aux=~-2;
  bit=10;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=261956;
  aux=~-4;
  bit=18;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=523920;
  aux=~-8;
  bit=19;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1047856;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2095744;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4191552;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8383232;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16766720;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33533952;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67068928;
  aux=~-1024;
  bit=26;
 }
}
if(j==12)
{
 if(k==-1||k==1)
 {
  temp=2036;
  aux=~-2;
  bit=11;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=261992;
```

```
 aux=~-4;
 bit=18;
}
if((k>=-7&&k<=-4)||(k>=4&&k<=7))
{
 temp=523992;
 aux=~-8;
 bit=19;
}
if((k>=-15&&k<=-8)||(k>=8&&k<=15))
{
 temp=1048000;
 aux=~-16;
 bit=20;
}
if((k>=-31&&k<=-16)||(k>=16&&k<=31))
{
 temp=2096032;
 aux=~-32;
 bit=21;
}
if((k>=-63&&k<=-32)||(k>=32&&k<=63))
{
 temp=4192128;
 aux=~-64;
 bit=22;
}
if((k>=-127&&k<=-64)||(k>=64&&k<=127))
{
 temp=8384384;
 aux=~-128;
 bit=23;
}
if((k>=-255&&k<=-128)||(k>=128&&k<=255))
{
 temp=16769024;
 aux=~-256;
 bit=24;
}
if((k>=-511&&k<=-256)||(k>=256&&k<=511))
{
 temp=33538560;
 aux=~-512;
 bit=25;
}
if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
{
 temp=67078144;
 aux=~-1024;
 bit=26;
}
}
if(j==13)
{
 if(k==-1||k==1)
 {
  temp=4084;
  aux=~-2;
  bit=12;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=262028;
  aux=~-4;
  bit=18;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=524064;
  aux=~-8;
  bit=19;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1048144;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2096320;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4192704;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8385536;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16771328;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33543168;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67087360;
  aux=~-1024;
  bit=26;
 }
}
if(j==14)
{
 if(k==-1||k==1)
 {
  temp=8172;
  aux=~-2;
  bit=13;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=262064;
  aux=~-4;
  bit=18;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=524136;
```

```
 aux=~-8;
 bit=19;
}
if((k>=-15&&k<=-8)||(k>=8&&k<=15))
{
 temp=1048288;
 aux=~-16;
 bit=20;
}
if((k>=-31&&k<=-16)||(k>=16&&k<=31))
{
 temp=2096608;
 aux=~-32;
 bit=21;
}
if((k>=-63&&k<=-32)||(k>=32&&k<=63))
{
 temp=4193280;
 aux=~-64;
 bit=22;
}
if((k>=-127&&k<=-64)||(k>=64&&k<=127))
{
 temp=8386688;
 aux=~-128;
 bit=23;
}
if((k>=-255&&k<=-128)||(k>=128&&k<=255))
{
 temp=16773632;
 aux=~-256;
 bit=24;
}
if((k>=-511&&k<=-256)||(k>=256&&k<=511))
{
 temp=33547776;
 aux=~-12;
 bit=25;
}
if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
{
 temp=67096576;
 aux=~-1024;
 bit=26;
}
}
if(j==15)
{
 if(k==-1||k==1)
 {
  temp=131050;
  aux=~-2;
  bit=17;
 }
 if(k==-3||k==-2||k==2||k==3)
 {
  temp=262104;
  aux=~-4;
  bit=18;
 }
 if((k>=-7&&k<=-4)||(k>=4&&k<=7))
 {
  temp=524216;
  aux=~-8;
  bit=19;
 }
 if((k>=-15&&k<=-8)||(k>=8&&k<=15))
 {
  temp=1048448;
  aux=~-16;
  bit=20;
 }
 if((k>=-31&&k<=-16)||(k>=16&&k<=31))
 {
  temp=2096928;
  aux=~-32;
  bit=21;
 }
 if((k>=-63&&k<=-32)||(k>=32&&k<=63))
 {
  temp=4193920;
  aux=~-64;
  bit=22;
 }
 if((k>=-127&&k<=-64)||(k>=64&&k<=127))
 {
  temp=8387968;
  aux=~-128;
  bit=23;
 }
 if((k>=-255&&k<=-128)||(k>=128&&k<=255))
 {
  temp=16776192;
  aux=~-256;
  bit=24;
 }
 if((k>=-511&&k<=-256)||(k>=256&&k<=511))
 {
  temp=33552896;
  aux=~-512;
  bit=25;
 }
 if((k>=-1023&&k<=-512)||(k>=512&&k<=1023))
 {
  temp=67106816;
  aux=~-1024;
  bit=26;
 }
}
if(k>=0)
{
 temp=temp|k;
}
if(k<0)
{
 a=(k&aux)-1;
 temp=temp|a;
}
j=0;
l=l+1;
d=d+1;
}
salida=temp;
numero=bit;
}
if(i==(64-(*p)))
{
salida=10;
numero=4;
l=l+1;
d=d+1;
j=0;
```

```
 }
 out[m+l]=salida;
 bits[m+l]=numero;
salto:
 b+=1;
}
```

NOMBRE: Algoritmo Huffman_DC.c
DESCRIPCION: El archivo contiene a la función que codifica todos los valores DC de cada una de las submatrices 8x8 con el procedimiento del Algoritmo de Huffman.

```
void Huffman_DC(int*in,int*out,int*bits)
{
 int i,n=0,temp,a,aux=0,bit;
 for(i=0;i<520;i++)//CODIGO DC JPEG (luminancia)
 {
  n=in[i];
  if(n==0)
  {
   if(i==519)
   {
    temp=10;
    bit=4;
   }
   else
   {
    temp=2;
    bit=3;
   }
  }
  if(n==-1||n==1)
  {
   if(i==519)
   {
    temp=22;
    bit=5;
   }
   else
   {
    temp=6;
    bit=4;
   }
   aux=~-2;
  }
  if(n==-3||n==-2||n==2||n==3)
  {
   if(i==519)
   {
    temp=48;
    bit=6;
   }
   else
   {
    temp=16;
    bit=5;
   }
   aux=~-4;
  }
  if((n>=-7&&n<=-4)||(n>=4&&n<=7))
  {
   if(i==519)
   {
    temp=32;
    bit=6;
   }
   else
```

```
  m=m+d;
  *s=m;
 }
```

```
 {
  temp=0;
  bit=5;
 }
 aux=~-8;
}
if((n>=-15&&n<=-8)||(n>=8&&n<=15))
{
 if(i==519)
 {
  temp=208;
  bit=8;
 }
 else
 {
  temp=80;
  bit=7;
 }
 aux=~-16;
}
if((n>=-31&&n<=-16)||(n>=16&&n<=31))
{
 if(i==519)
 {
  temp=448;
  bit=9;
 }
 else
 {
  temp=192;
  bit=8;
 }
 aux=~-32;
}
if((n>=-63&&n<=-32)||(n>=32&&n<=63))
{
 if(i==519)
 {
  temp=1920;
  bit=11;
 }
 else
 {
  temp=896;
  bit=10;
 }
 aux=~-64;
}
if((n>=-127&&n<=-64)||(n>=64&&n<=127))
{
 if(i==519)
 {
  temp=7936;
  bit=13;
 }
 else
 {
  temp=3840;
  bit=12;
 }
 aux=~-128;
}
if((n>=-255&&n<=-128)||(n>=128&&n<=255))
{
 if(i==519)
 {
  temp=32256;
  bit=15;
 }
 else
 {
  temp=15872;
  bit=14;
 }
 aux=~-256;
}
if((n>=-511&&n<=-256)||(n>=256&&n<=511))
{
 if(i==519)
 {
  temp=130048;
  bit=17;
 }
 else
 {
  temp=64512;
  bit=16;
 }
 aux=~-512;
}
if((n>=-1023&&n<=-512)||(n>=512&&n<=1023))
{
 if(i==519)
 {
  temp=522240;
  bit=19;
 }
 else
 {
  temp=260096;
  bit=18;
 }
 aux=~-1024;
}
if((n>=-2047&&n<=-1024)||(n>=1024&&n<=2047))
{
 if(i==519)
 {
  temp=2093056;
  bit=21;
 }
 else
 {
  temp=1044480;
  bit=20;
 }
 aux=~-2048;
}
if(n>=0)
{
 temp=temp|n;
}
if(n<0)
{
 a=(n&aux)-1;
 temp=temp|a;
}
out[i]=temp;
bits[i]=bit;
}
}
```

NOMBRE: Huffman.c
DESCRIPCION: El archivo contiene a la función que almacena todos los valores codificados de la imagen completa, tanto DC como AC, sin empaquetamiento.

```
void Huffman(int*in,int*in2,int*bit,int*bit2,int*n,int*out,int*bitout)
{
 int i,j=0,k;
 k=*n;
 for(i=0;i<(k+521);i++)
 {
  if(i<k)
  {
   out[i]=in2[i];
   bitout[i]=bit2[i];
  }
  if(i>=k)
  {
   out[i]=in[j];
   bitout[i]=bit[j];
   j+=1;
  }
 }
}
```

NOMBRE: Relleno_ceros.c
DESCRIPCION: El archivo contiene a la función que rellena de ceros la matriz en donde se guardara la imagen empaquetada (es decir, sin existencia de ceros).

```
void Relleno_ceros(unsigned char*in)
{
 int i;
 for(i=0;i<18320;i++)
 {
  in[i]=0;
 }
}
```

NOMBRE: Almacenamiento JPEG.c
DESCRIPCION: El archivo contiene a la función que almacena la imagen completa comprimida empaquetada.

```
#include<math.h>
void Almacenamiento(int*in,int*num_bit,unsigned char*out,int*n,int*s)
{
 int a=0,i,j,g=0,x=0,m=0,temp=0,tempo=0,numero=0;
 unsigned char salida=0,temporal=0;
 m=*n;
 m=m+520;
 x=*s;
 for(i=0;i<m;i++)
 {
  temp=in[i];
  numero=num_bit[i];
  for(j=0;j<numero;j++)
  {
   a=pow(2,j);
   tempo=temp&a;
   tempo=tempo>>j;
   temporal=(unsigned char)tempo;
   temporal=temporal<<g;
   salida=temporal|salida;
   g+=1;
   if(g==8)
   {
```

```
    out[x]=salida;
    salida=0;
    x+=1;
    g=0;
   }
  }
  if(i==m-1)
  {
   out[x]=salida;
   salida=0;
   x+=1;
   g=0;
  }
 }
 *s=x;
}
```

ANEXO F

CÓDIGO FUENTE PARA DESCOMPRESIÓN JPEG.

NOMBRE: Descompresión JPEG.c
DESCRIPCIÓN: Archivo principal que contiene la implementación de la descompresión JPEG.

```
#pragma DATA_SECTION(imagen,"picture");
#pragma DATA_SECTION(decodificado,"picture");
#pragma DATA_SECTION(dato,"picture");
#pragma DATA_SECTION(imagen_temp,"picture");
unsigned char imagen[208][160];
unsigned char imagen_temp[208][160];
int decodificado[33280];
unsigned char dato[11000];
unsigned char temp[11000];
int num_val_subimagenes[520];
int num_subimagen;
int subimagen[64];
int estandar[64];
int transfor_inv[64];
float temporal[64];
float tempo[64];

float DCT [8][8]={0.3536,0.3536,0.3536,0.3536,0.3536,0.3536,0.3536,0.3536,
                  0.4904,0.4157,0.2778,0.0975,-0.0975,-0.2778,-0.4157,-0.4904,
                  0.4619,0.1913,-0.1913,-0.4619,-0.4619,-0.1913,0.1913,0.4619,
                  0.4157,-0.0975,-0.4904,-0.2778,0.2778,0.4904,0.0975,-0.4157,
                  0.3536,-0.3536,-0.3536,0.3536,0.3536,-0.3536,-0.3536,0.3536,
                  0.2778,-0.4904,0.0975,0.4157,-0.4157,-0.0975,0.4904,-0.2778,
                  0.1913,-0.4619,0.4619,-0.1913,-0.1913,0.4619,-0.4619,0.1913,
                  0.0975,-0.2778,0.4157,-0.4904,0.4904,-0.4157,0.2778,-0.0975};
float DCT_t[8][8]={0.3536,0.4904,0.4619,0.4157,0.3536,0.2778,0.1913,0.0975,
                   0.3536,0.4157,0.1913,-0.0975,-0.3536,-0.4904,-0.4619,-0.2778,
                   0.3536,0.2778,-0.1913,-0.4904,-0.3536,0.0975,0.4619,0.4157,
                   0.3536,0.0975,-0.4619,-0.2778,0.3536,0.4157,-0.1913,-0.4904,
                   0.3536,-0.0975,-0.4619,0.2778,0.3536,-0.4157,-0.1913,0.4904,
                   0.3536,-0.2778,-0.1913,0.4904,-0.3536,-0.0975,0.4613,-0.4157,
                   0.3536,-0.4157,0.1913,0.0975,-0.3536,0.4904,-0.4613,0.2778,
                   0.3536,-0.4904,0.4619,-0.4157,0.3536,-0.2778,0.1913,-0.0975};

int i,j=0,k,m=0,n=0,cant_decod=0;

void main()
{
 n=Numero_bytes(dato);
 k=5500-n;
 Decodificacion_Huffman(dato,decodificado,num_val_subimagenes,&k,&cant_decod);
 Valores_DC(decodificado);
```

```
 for(i=0;i<520;i++)
 {
  Reconstruccion_subimagen(decodificado,num_val_subimagenes,&i,&j,subimagen);
  Zig_Zag(subimagen,estandar);
  Cuantificacion(estandar,transfor_inv);
  Transformada_inversa(&DCT_t[0][0],transfor_inv,&DCT[0][0],temporal);
  Suma(temporal,tempo);
  Almacenamiento(tempo,imagen_temp,&m);
  Reordenamiento(imagen_temp,imagen);
 }
 exit();
}
```

NOMBRE: Número_bytes.c
DESCRIPCION: Archivo que contiene la funcion que permite calcular la cantidad de bytes diferentes de cero que contiene la imagen a descomprimir.

```
int Numero_bytes(unsigned char*in)
{
 int i,k,n=0;
 for(i=0;i<5500;i++)
 {
  k=5499-i;
  if(in[k]==0)
  {
   n=n+1;
  }
  else
  {
   goto salir;
  }
 }
salir: return(n);
}
```

NOMBRE: Decodificación Huffman.c
DESCRIPCION: Archivo que contiene la funcion que permite revertir la codificacion Huffman realizada a la imagen en el proceso de compresion.

```
#include<math.h>
void Decodificacion_Huffman(unsigned char*in,int*out,int*num_subimagen,int*n,int*num)
{
 unsigned char temp,tempo;
 int AC=0,a=0,b,c=0,d=0,e=-1,i=0,j=0,k=0,m=0,p=0,r=0,q,s=0,ceros=0,signo=0,numero_bits=0,almacena=0,aux=0;
 unsigned int alterna=0,alterno=0,temporal=0,inicio=0;
 int salida=0;
 k=*n;
 b=*num;
 for(i=1;i<=k;i++)
 {
  temp=in[k-i];
  for(j=1;j<=8;j++)
  {
   a=pow(2,8-j);
   tempo=temp&a;
   tempo=tempo>>8-j;
   temporal=(unsigned int)tempo;
   salida=salida<<c;
   salida=temporal|salida;
   c=1;
   numero_bits+=1;
   if(inicio==1)
   {
    goto start;
   }
```

```
  if(salida==1)
  {
   salida=0;
   alterno=1;
   numero_bits=0;
   goto start;
  }
  if(salida==0)
  {
   numero_bits=0;
   goto reinicio;
  }
start:
  inicio=1;
  if(AC==1)
  {
   goto codigo_AC;
  }
  if(m==520)
  {
   AC=1;
   goto codigo_AC;
  }
  if(d==2)
  {
   goto decod_DC;
  }
  if(numero_bits==2&&salida==0)
  {
   alterna=3;
   numero_bits=0;
   d=2;
  }
  if(numero_bits==3)
  {
   if(salida==2)
   {
    alterna=0;
    numero_bits=0;
    d=2;
   }
   if(salida==3)
   {
    alterna=1;
    numero_bits=0;
    d=2;
   }
   if(salida==4)
   {
    alterna=2;
    numero_bits=0;
    d=2;
   }
   if(salida==5)
   {
    alterna=4;
    numero_bits=0;
    d=2;
   }
   if(salida==6)
   {
    alterna=5;
    numero_bits=0;
    d=2;
   }
  }
  if(numero_bits==4&&salida==14)
  {
   alterna=6;
   numero_bits=0;
   d=2;
  }
  if(numero_bits==5&&salida==30)
  {
   alterna=7;
   numero_bits=0;
   d=2;
  }
  if(numero_bits==6&&salida==62)
  {
   alterna=8;
   numero_bits=0;
   d=2;
  }
  if(numero_bits==7&&salida==126)
  {
   alterna=9;
   numero_bits=0;
   d=2;
  }
  if(numero_bits==8&&salida==254)
  {
   alterna=10;
   numero_bits=0;
   d=2;
  }
  if(numero_bits==9&&salida==510)
  {
   alterna=11;
   numero_bits=0;
   d=2;
  }
decod_DC:
  if(d==2)
  {
   if(numero_bits==0&&alterna==0)
   {
    salida=0;
    almacena=1;
   }
   if(numero_bits==1&&alterna==1)
   {
    salida=salida&1;
    if(salida!=1)
    {
     signo=1;
     aux=-2;
    }
    almacena=1;
   }
   if(numero_bits==2&&alterna==2)
   {
    salida=salida&3;
    if(salida!=2&&salida!=3)
    {
     signo=1;
     aux=-4;
    }
    almacena=1;
   }
   if(numero_bits==3&&alterna==3)
   {
```

```
 salida=salida&7;
 if(salida<4||salida>7)
 {
  signo=1;
  aux=-8;
 }
 almacena=1;
}
if(numero_bits==4&&alterna==4)
{
 salida=salida&15;
 if(salida<8||salida>15)
 {
  signo=1;
  aux=-16;
 }
 almacena=1;
}
if(numero_bits==5&&alterna==5)
{
 salida=salida&31;
 if(salida<16||salida>31)
 {
  signo=1;
  aux=-32;
 }
 almacena=1;
}
if(numero_bits==6&&alterna==6)
{
 salida=salida&63;
 if(salida<32||salida>63)
 {
  signo=1;
  aux=-64;
 }
 almacena=1;
}
if(numero_bits==7&&alterna==7)
{
 salida=salida&127;
 if(salida<64||salida>127)
 {
  signo=1;
  aux=-128;
 }
 almacena=1;
}
if(numero_bits==8&&alterna==8)
{
 salida=salida&255;
 if(salida<128||salida>255)
 {
  signo=1;
  aux=-256;
 }
 almacena=1;
}
if(numero_bits==9&&alterna==9)
{
 salida=salida&511;
 if(salida<256||salida>511)
 {
  signo=1;
  aux=-512;
 }
  almacena=1;
 }
 if(numero_bits==10&&alterna==10)
 {
  salida=salida&1023;
  if(salida<512||salida>1023)
  {
   signo=1;
   aux=-1024;
  }
  almacena=1;
 }
 if(numero_bits==11&&alterna==11)
 {
  salida=salida&2047;
  if(salida<1024||salida>2047)
  {
   signo=1;
   aux=-2048;
  }
  almacena=1;
 }
}
goto almacena;
codigo_AC:
 if(numero_bits==2)
 {
  if(salida==0)
  {
   alterno=1;
   numero_bits=0;
   d=1;
  }
  if(salida==1)
  {
   alterno=2;
   numero_bits=0;
   d=1;
  }
 }
 if(numero_bits==3&&salida==4)
 {
  alterno=3;
  numero_bits=0;
  d=1;
 }
 if(numero_bits==4)
 {
  if(salida==10)
  {
   r=r+p;
   p=b-520-r;
   num_subimagen[e]=p;
   e+=1;
   salida=0;

   numero_bits=0;
  }
  if(salida==11)
  {
   alterno=4;
   numero_bits=0;
   d=1;
  }
  if(salida==12)
  {
```

```
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
}
if(numero_bits==5)
{
 if(salida==26)
 {
  alterno=5;
  numero_bits=0;
  d=1;
 }
 if(salida==27)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
}
if(numero_bits==6)
{
 if(salida==56)
 {
  alterno=6;
  numero_bits=0;
  d=1;
 }
 if(salida==57)
 {
  alterno=2;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
 if(salida==58)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=3;
  d=1;
 }
 if(salida==59)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=4;
  d=1;
 }
}
if(numero_bits==7)
{
 if(salida==120)
 {
  alterno=7;
  numero_bits=0;
  d=1;
 }
 if(salida==121)
 {
  alterno=3;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
 if(salida==122)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=5;
  d=1;
 }
 if(salida==123)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=6;
  d=1;
 }
}
if(numero_bits==8)
{
 if(salida==248)
 {
  alterno=2;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
 if(salida==249)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=7;
  d=1;
 }
 if(salida==250)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=8;
  d=1;
 }
}
if(numero_bits==9)
{
 if(salida==502)
 {
  alterno=4;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
 if(salida==503)
 {
  alterno=2;
  numero_bits=0;
  ceros=1;
```

```
 s=3;
 d=1;
}
if(salida==504)
{
 alterno=1;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
}
if(salida==505)
{
 alterno=1;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==506)
{
 alterno=1;
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
}
if(numero_bits==10)
{
 if(salida==1014)
 {
  alterno=8;
  numero_bits=0;
  d=1;
 }
 if(salida==1015)
 {
  alterno=3;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
 if(salida==1016)
 {
  alterno=2;
  numero_bits=0;
  ceros=1;
  s=4;
  d=1;
 }
 if(salida==1017)
 {
  alterno=2;
  numero_bits=0;
  ceros=1;
  s=5;
  d=1;
 }
 if(salida==1018)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=12;
  d=1;
 }
}
if(numero_bits==11)
{
 if(salida==2038)
 {
  alterno=5;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
 if(salida==2039)
 {
  alterno=3;
  numero_bits=0;
  ceros=1;
  s=3;
  d=1;
 }
 if(salida==2040)
 {
  alterno=2;
  numero_bits=0;
  ceros=1;
  s=6;
  d=1;
 }
 if(salida==2041)
 {
  alterno=2;
  numero_bits=0;
  ceros=1;
  s=7;
  d=1;
 }
 if(salida==2042)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=13;
  d=1;
 }
}
if(numero_bits==12)
{
 if(salida==4086)
 {
  alterno=1;
  numero_bits=0;
  ceros=1;
  s=14;
  d=1;
 }
 if(salida==4087)
 {
  salida=0;
  ceros=1;
  s=15;
  d=1;
  almacena=1;
  goto almacena;
 }
}
```

```
if(numero_bits==15)
{
 if(salida==32704)
 {
  alterno=2;
  numero_bits=0;
  ceros=1;
  s=8;
  d=1;
 }
}
if(numero_bits==16)
{
 if(salida==65410)
 {
  alterno=9;
  numero_bits=0;
  d=1;
 }
 if(salida==65411)
 {
  alterno=10;
  numero_bits=0;
  d=1;
 }
 if(salida==65412)
 {
  alterno=6;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
 if(salida==65413)
 {
  alterno=7;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
 if(salida==65414)
 {
  alterno=8;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
 if(salida==65415)
 {
  alterno=9;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
 if(salida==65416)
 {
  alterno=10;
  numero_bits=0;
  ceros=1;
  s=1;
  d=1;
 }
 if(salida==65417)
 {
  alterno=4;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
 if(salida==65418)
 {
  alterno=5;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
 if(salida==65419)
 {
  alterno=6;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
 if(salida==65420)
 {
  alterno=7;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
 if(salida==65421)
 {
  alterno=8;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
 if(salida==65422)
 {
  alterno=9;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
 if(salida==65423)
 {
  alterno=10;
  numero_bits=0;
  ceros=1;
  s=2;
  d=1;
 }
 if(salida==65424)
 {
  alterno=4;
  numero_bits=0;
  ceros=1;
  s=3;
  d=1;
 }
 if(salida==65425)
 {
  alterno=5;
```

```
 numero_bits=0;
 ceros=1;
 s=3;
 d=1;
}
if(salida==65426)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=3;
 d=1;
}
if(salida==65427)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=3;
 d=1;
}
if(salida==65428)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=3;
 d=1;
}
if(salida==65429)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=3;
 d=1;
}
if(salida==65430)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=3;
 d=1;
}
if(salida==65431)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=4;
 d=1;
}
if(salida==65432)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=4;
 d=1;
}
if(salida==65433)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=4;
 d=1;
}
if(salida==65434)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=4;
 d=1;
}
if(salida==65435)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=4;
 d=1;
}
if(salida==65436)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=4;
 d=1;
}
if(salida==65437)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=4;
 d=1;
}
if(salida==65438)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=4;
 d=1;
}
if(salida==65439)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=5;
 d=1;
}
if(salida==65440)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=5;
 d=1;
}
if(salida==65441)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=5;
 d=1;
```

```
}
if(salida==65442)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=5;
 d=1;
}
if(salida==65443)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=5;
 d=1;
}
if(salida==65444)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=5;
 d=1;
}
if(salida==65445)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=5;
 d=1;
}
if(salida==65446)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=5;
 d=1;
}
if(salida==65447)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=6;
 d=1;
}
if(salida==65448)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=6;
 d=1;
}
if(salida==65449)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=6;
 d=1;
}
if(salida==65450)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=6;
 d=1;
}
if(salida==65451)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=6;
 d=1;
}
if(salida==65452)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=6;
 d=1;
}
if(salida==65453)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=6;
 d=1;
}
if(salida==65454)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=6;
 d=1;
}
if(salida==65455)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=7;
 d=1;
}
if(salida==65456)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=7;
 d=1;
}
if(salida==65457)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=7;
 d=1;
}
if(salida==65458)
{
 alterno=6;
```

```
 numero_bits=0;
 ceros=1;
 s=7;
 d=1;
}
if(salida==65459)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=7;
 d=1;
}
if(salida==65460)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=7;
 d=1;
}
if(salida==65461)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=7;
 d=1;
}
if(salida==65462)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=7;
 d=1;
}
if(salida==65463)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=8;
 d=1;
}
if(salida==65464)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=8;
 d=1;
}
if(salida==65465)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=8;
 d=1;
}
if(salida==65466)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=8;
 d=1;
}
if(salida==65467)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=8;
 d=1;
}
if(salida==65468)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=8;
 d=1;
}
if(salida==65469)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=8;
 d=1;
}
if(salida==65470)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=8;
 d=1;
}
if(salida==65471)
{
 alterno=2;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
}
if(salida==65472)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
}
if(salida==65473)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
}
if(salida==65474)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
```

```
}
if(salida==65475)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
}
if(salida==65476)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
}
if(salida==65477)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
}
if(salida==65478)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
}
if(salida==65479)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=9;
 d=1;
}
if(salida==65480)
{
 alterno=2;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==65481)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==65482)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==65483)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==65484)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==65485)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==65486)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==65487)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==65488)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=10;
 d=1;
}
if(salida==65489)
{
 alterno=2;
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
if(salida==65490)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
if(salida==65491)
{
 alterno=4;
```

```
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
if(salida==65492)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
if(salida==65493)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
if(salida==65494)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
if(salida==65495)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
if(salida==65496)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
if(salida==65497)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=11;
 d=1;
}
if(salida==65498)
{
 alterno=2;
 numero_bits=0;
 ceros=1;
 s=12;
 d=1;
}
if(salida==65499)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=12;
 d=1;
}
if(salida==65500)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=12;
 d=1;
}
if(salida==65501)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=12;
 d=1;
}
if(salida==65502)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=12;
 d=1;
}
if(salida==65503)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=12;
 d=1;
}
if(salida==65504)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=12;
 d=1;
}
if(salida==65505)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=12;
 d=1;
}
if(salida==65506)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=12;
 d=1;
}
if(salida==65507)
{
 alterno=2;
 numero_bits=0;
 ceros=1;
 s=13;
 d=1;
```

```
}
if(salida==65508)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=13;
 d=1;
}
if(salida==65509)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=13;
 d=1;
}
if(salida==65510)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=13;
 d=1;
}
if(salida==65511)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=13;
 d=1;
}
if(salida==65512)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=13;
 d=1;
}
if(salida==65513)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=13;
 d=1;
}
if(salida==65514)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=13;
 d=1;
}
if(salida==65515)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=13;
 d=1;
}
if(salida==65516)
{
 alterno=2;
 numero_bits=0;
 ceros=1;
 s=14;
 d=1;
}
if(salida==65517)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=14;
 d=1;
}
if(salida==65518)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=14;
 d=1;
}
if(salida==65519)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=14;
 d=1;
}
if(salida==65520)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=14;
 d=1;
}
if(salida==65521)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=14;
 d=1;
}
if(salida==65522)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=14;
 d=1;
}
if(salida==65523)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=14;
 d=1;
}
if(salida==65524)
{
 alterno=10;
```

```
 numero_bits=0;
 ceros=1;
 s=14;
 d=1;
}
if(salida==65525)
{
 alterno=1;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
if(salida==65526)
{
 alterno=2;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
if(salida==65527)
{
 alterno=3;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
if(salida==65528)
{
 alterno=4;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
if(salida==65529)
{
 alterno=5;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
if(salida==65530)
{
 alterno=6;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
if(salida==65531)
{
 alterno=7;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
if(salida==65532)
{
 alterno=8;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
if(salida==65533)
{
 alterno=9;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
if(salida==65534)
{
 alterno=10;
 numero_bits=0;
 ceros=1;
 s=15;
 d=1;
}
}
if(d==1)
{
 if(numero_bits==1&&alterno==1)
 {
  salida=salida&1;
  if(salida!=1)
  {
   signo=1;
   aux=-2;
  }
  almacena=1;
 }
 if(numero_bits==2&&alterno==2)
 {
  salida=salida&3;
  if(salida!=2&&salida!=3)
  {
   signo=1;
   aux=-4;
  }
  almacena=1;
 }
 if(numero_bits==3&&alterno==3)
 {
  salida=salida&7;
  if(salida<4||salida>7)
  {
   signo=1;
   aux=-8;
  }
  almacena=1;
 }
 if(numero_bits==4&&alterno==4)
 {
  salida=salida&15;
  if(salida<8||salida>15)
  {
   signo=1;
   aux=-16;
  }
  almacena=1;
 }
 if(numero_bits==5&&alterno==5)
 {
  salida=salida&31;
  if(salida<16||salida>31)
```

```
    {
     signo=1;
     aux=-32;
    }
    almacena=1;
   }
   if(numero_bits==6&&alterno==6)
   {
    salida=salida&63;
    if(salida<32||salida>63)
    {
     signo=1;
     aux=-64;
    }
    almacena=1;
   }
   if(numero_bits==7&&alterno==7)
   {
    salida=salida&127;
    if(salida<64||salida>127)
    {
     signo=1;
     aux=-128;
    }
    almacena=1;
   }
   if(numero_bits==8&&alterno==8)
   {
    salida=salida&255;
    if(salida<128||salida>255)
    {
     signo=1;
     aux=-256;
    }
    almacena=1;
   }
   if(numero_bits==9&&alterno==9)
   {
    salida=salida&511;
    if(salida<256||salida>511)
    {
     signo=1;
     aux=-512;
    }
    almacena=1;
   }
   if(numero_bits==10&&alterno==10)
   {
    salida=salida&1023;
    if(salida<512||salida>1023)
    {
     signo=1;
     aux=-1024;
    }
    almacena=1;
   }
  }
almacena:
  if(almacena==1)
  {
   b+=1;
   numero_bits=0;
   if(signo==1)
   {
    signo=0;
    salida=(salida+1)|aux;
    out[m]=salida;
   }
   if(signo==0)
   {
    out[m]=salida;
   }
   if(ceros==1)
   {
    for(q=1;q<=s;q++)
    {
     m+=1;
     out[m]=0;
     b+=1;
    }
    ceros=0;
    c=0;
   }
   m+=1;
   if(e==519)
   {
    num_subimagen[e]=63-(m-520-r);
   }
   salida=0;
   d=0;
  }
reinicio:
  almacena=0;
 }
}
 *num=b;
}
```

NOMBRE: Valores_DC.c
DESCRIPCION: Archivo que contiene la funcion que permite recuperar el valor original de los valores DC antes de la codificacion de la imagen total.

```
void Valores_DC(int*in)
{
 int i,j,k;
 int temp[520],temp1[520];

 for(i=0;i<520;i++)
 {
  temp[519-i]=in[i];
  temp1[518-i]=in[i];
 }
 for(j=0;j<520;j++)
```

```
 {
  temp[j+1]=temp[j]+temp1[j];
 }
 for(k=0;k<520;k++)
 {
  in[k]=temp[519-k];
 }
}
```

NOMBRE: Reconstruccion_subimagen.c
DESCRIPCION: Archivo que contiene la funcion que permite la reconstruccion de las sub-imagenes antes del reordenamiento en zigzag en el proceso de compresión.

```
void Reconstruccion_subimagen(int*in,int*in_2,int*n,int*p,int*out)
{
 int i,j,k,m;
 int out1[64];
 j=*n;
 m=*p;
 k=in_2[j];
 m=m+520;
 for(i=0;i<64;i++)
 {
  if(i==0)
  {
   out[i]=in[j];
  }
  if(i>0&&i<=k)
  {
   out1[i]=in[m+i-1];
   out[k-i+1]=out1[i];
  }
  if(i>k)
  {
   out[i]=0;
  }
 }
 m=m+k-520;
 *p=m;
}
```

NOMBRE: Zig_zag.c
DESCRIPCION: Archivo que contiene la funcion que permite el reordenamiento de las sub-imagenes antes de la etapa de cuantificación en la compresión.

```
void Zig_Zag(int*in,int*out)
{
 int posicion[64]={0,1,5,6,14,15,27,28,
                   2,4,7,13,16,26,29,42,
                   3,8,12,17,25,30,41,43,
                   9,11,18,24,31,40,44,53,
                   10,19,23,32,39,45,52,54,
                   20,22,33,38,46,51,55,60,
                   21,34,37,47,50,56,59,61,
                   35,36,48,49,57,58,62,63};
 int i,j=0;
 for(i=0;i<64;i++)
 {
  j=posicion[i];
  out[i]=in[j];
 }
}
```

NOMBRE: Cuantificación JPEG (descompresion).c
DESCRIPCION: Archivo que contiene la funcion que permite la decuantificacion de las sub-imagenes antes de la etapa de transformación en la compresión.

```
void Cuantificacion(int*in,int*out)
{
 int normal [64]={ 8, 5, 5, 8,12,20,25,30,
                   6, 6, 7, 9,13,29,30,27,
                   7, 6, 8,12,20,28,34,28,
                   7, 8,11,14,25,43,40,31,
                   9,11,18,28,34,54,51,38,
                   12,17,27,32,40,52,56,46,
                   24,32,39,43,51,60,60,50,
                   36,46,47,49,56,50,51,49};
 int i;
 for(i=0;i<64;i++)
 {
  out[i]=in[i]*normal[i];
 }
}
```

NOMBRE: Transformada inversa.c
DESCRIPCION: Archivo que contiene la funcion que permite la transformación inversa de las sub-imagenes a descomprimir.

```
void Transformada_inversa(float*in_1,int*in_2,float*in_3,float*out)
{
 unsigned int i,j,l,x,y,z;
 float D[8][8],sum,suma;
 for (i=0;i<8;i++)
 {
  for (j=0;j<8;j++)
  {
   sum=0;
   for (l=0;l<8;l++)
   {
    sum += in_1[l+i*8] * (float)in_2[j+l*8];
    D[i][j]=sum;
   }
  }
 }
 for (x=0;x<8;x++)
 {
  for (y=0;y<8;y++)
  {
   suma=0;
   for (z=0;z<8;z++)
   {
    suma +=D[x][z] * in_3[y+z*8];
    out[y+x*8]=suma;
   }
  }
 }
}
```

NOMBRE: Suma.c
DESCRIPCION: Archivo que contiene la funcion que añade a cada uno de los elementos de la sub-imagen el valor de 128, ademas convierte sus cantidades de valores con decimales a enteros.

```
#include<math.h>
void Suma(float*in,float*out)
{
```

```
 int i,j;
 for(i=0;i<64;i++)
 {
  if(in[i]<0)
  {
   in[i]=ceil(in[i]);
  }
  else
  {
   in[i]=floor(in[i]);
  }
 }
 for(j=0;j<64;j++)
 {
  out[j]=in[j]+128;
  if(out[j]>255)
  {
   out[j]=255;
  }
 }
}
```

NOMBRE: Almacenamiento.c
DESCRIPCION: Archivo que contiene la función que reordena a cada sub-imagen y luego la almacena en una matriz, la cual tendra temporalmente la imagen total.

```
void Almacenamiento(float*in,unsigned char*out,int*n)
{
 int i,m;
 unsigned char temp[64];
 m=*n;
 for(i=0;i<64;i++)
 {
  temp[
63-i]=in[i];
 }
 for(i=0;i<64;i++)
 {
  out[m+i]=(unsigned char)temp[i];
 }
 m=m+64;
 *n=m;
}
```

NOMBRE: Reordenamiento.c
DESCRIPCION: Archivo que contiene la funcion que permite reordenar los valores de la imagen completa descomprimida.

```
void Reordenamiento(unsigned char*in,unsigned char*out)
{
 int i;
 for(i=0;i<33280;i++)
 {
  out[i]=in[33279-i];
 }
}
```

ANEXO G

CÓDIGO FUENTE PARA LA COMPRESIÓN HADAMARD (NIVEL 1,78:1).

NOMBRE: Compresión HADAMARD 2.c
DESCRIPCIÓN: Archivo principal donde se implementa la compresión Hadamard nivel 1,78 a 1 de compresión.

```
#pragma DATA_SECTION(imagen,"picture");
unsigned char imagen[208][160];
float temp [8][8];
float transfor[8][8];
float estandar[64];
float temp_1[36];
float hadamard[18720];
int a,b,c,d,e,f,n=0;
unsigned char *imageptr;
void main()
{
 imageptr=&imagen[0][0];
 for(a=0;a<520;a++)
 {
  for (c=0;c<8;c++)
  {
   for (d=0;d<8;d++)
   {
    temp[c][d]=(float)imageptr[64*a+8*c+d];
   }
  }
  transformada_Hadamard(&temp[0][0],&transfor[0][0]);
  Cuantificacion(&transfor[0][0],estandar);
  Recorte(estandar,temp_1);
  Almacenamiento_HADAMARD(temp_1,hadamard,&n);
 }
 exit();
}
```

NOMBRE: Transformación_HADAMARD 2.c
DESCRIPCION: Archivo que contiene la función que aplica la transformada Hadamard a cada una de las sub-imagenes de la huella digital.

```
void transformada_Hadamard(float*temp,float*Transfor)
{
 float D[8][8];
 float Hadamard[64]= {1, 1, 1, 1, 1, 1, 1, 1,
                      1, 1, 1, 1,-1,-1,-1,-1,
                      1, 1,-1,-1,-1,-1, 1, 1,
                      1, 1,-1,-1, 1, 1,-1,-1,
                      1,-1,-1, 1, 1,-1,-1, 1,
                      1,-1,-1, 1,-1, 1, 1,-1,
                      1,-1, 1,-1,-1, 1,-1, 1,
                      1,-1, 1,-1, 1,-1, 1,-1};
 unsigned int i,j,l,x,y,z;
 float sum,suma;
 for (i=0;i<8;i++)
 {
  for (j=0;j<8;j++)
  {
   sum=0;
   for (l=0;l<8;l++)
   {
    sum +=Hadamard[l+i*8] * temp[j+l*8];
    D[i][j]=sum;
   }
  }
 }
 for (x=0;x<8;x++)
 {
  for (y=0;y<8;y++)
  {
   suma=0;
   for (z=0;z<8;z++)
   {
    suma +=D[x][z] * Hadamard[y+z*8];
    Transfor[y+x*8]=suma/8;
   }
  }
 }
}
```

NOMBRE: Cuantificación HADAMARD 2.c
DESCRIPCION: Archivo que contiene la función que cuantifica cada una de las sub-imagenes de la huella digital.

```
#include <math.h>
#define value 8
float normal [value][value]={1,1,1,1,1,1,0,0,
                             1,1,1,1,1,1,0,0,
                              1,1,1,1,1,1,0,0,
                              1,1,1,1,1,1,0,0,
                              1,1,1,1,1,1,0,0,
                              1,1,1,1,1,1,0,0,
                              0,0,0,0,0,0,0,0,
                              0,0,0,0,0,0,0,0};

void Cuantificacion(float *entrada, float *estandar)
{
 int d,e;
 for(d=0;d<value;d++)
 {
  for(e=0;e<value;e++)
  {
```

```
   estandar[8*d+e] = entrada[8*d+e]*normal[d][e];
   if(estandar[8*d+e] <0)
   {
    estandar[8*d+e] = ceil(estandar[8*d+e]);
   }
   else
   {
    estandar[8*d+e] = floor(estandar[8*d+e]);
   }
  }
 }
}
```

NOMBRE: Recorte HADAMARD 2.c
DESCRIPCION: Archivo que contiene la función toma, para luego almacenarla, todos los valores de las sub-imagenes diferentes de cero.

```
void Recorte(float*in,float*out )
{
 int i,j;
 int a[36]={ 0, 1, 2, 3, 4, 5,
             8, 9,10,11,12,13,
             16,17,18,19,20,21,
             24,25,26,27,28,29,
             32,33,34,35,36,37,
             40,41,42,43,44,45};
 for(i=0;i<36;i++)
 {
  j=a[i];
  out[i]=in[j];
 }
}
```

NOMBRE: Almacenamiento HADAMARD 2.c
DESCRIPCION: Archivo que contiene la función almacena la imagen de la huella digital comprimida.

```
void Almacenamiento_HADAMARD(float*in,float*out,int*n)
{
 int i,k;
 k=*n;
 for(i=0;i<36;i++)
 {
  out[k+i]=in[i];
 }
 k=k+36;
 *n=k;
}
```

ANEXO H

CÓDIGO FUENTE PARA LA DESCOMPRESIÓN HADAMARD (NIVEL 1,78:1)

NOMBRE: Descompresión HADAMARD 2.c
DESCRIPCIÓN: Archivo que contiene el algoritmo para realizar la descompresión de la imagen de la huella digital.

```
#pragma DATA_SECTION(imagen,"picture");
unsigned char imagen[208][160];

float dato[18720];
float temp[36];
float estandar[64];
float transfor_inv[8][8];
float *TEMP,*DATO;
int n,i,x=0,y=0;
void main()
{
 TEMP=&temp[0];
 DATO=&dato[0];
 for(n=0;n<520;n++)
 {
  for(i=0;i<36;i++)
  {
   TEMP[i]=DATO[n*36+i];
  }
  Conversion(temp,estandar);
  Transformacion(estandar,&transfor_inv[0][0]);
  Almacenamiento(&transfor_inv[0][0],&imagen[0][0],&x);
  x=x+1;
 }
 exit();
}
```

NOMBRE: Conversión 2.c
DESCRIPCION: Archivo que contiene la función que permite la decuantificación de cada una de las sub-imagenes a descomprimir.

```
void Conversion(float*in,float*out)
{
 int i,j;
 int a[36]={ 0, 1, 2, 3, 4, 5,
             8, 9,10,11,12,13,
             16,17,18,19,20,21,
             24,25,26,27,28,29,
             32,33,34,35,36,37,
             40,41,42,43,44,45};
```

```
 for(i=0;i<64;i++)
 {
  out[i]=0;
 }
 for(i=0;i<36;i++)
 {
  j=a[i];
  out[j]=in[i];
 }
}
```

NOMBRE: Transformación Inversa HADAMARD 2.c
DESCRIPCION: Archivo que contiene la función que aplica la transformada inversa Hadamard a cada una de las sub-imagenes.

```
float D[8][8];
void Transformacion(float*in,float*out)
{
 float Hadamard[64]= {1, 1, 1, 1, 1, 1, 1, 1,
                      1, 1, 1, 1,-1,-1,-1,-1,
                      1, 1,-1,-1,-1,-1, 1, 1,
                      1, 1,-1,-1, 1, 1,-1,-1,
                      1,-1,-1, 1, 1,-1,-1, 1,
                      1,-1,-1, 1,-1, 1, 1,-1,
                      1,-1, 1,-1,-1, 1,-1, 1,
                      1,-1, 1,-1, 1,-1, 1,-1};
 unsigned int i,j,l,x,y,z;
 float sum,suma;
 for (i=0;i<8;i++)
 {
  for (j=0;j<8;j++)
  {
   sum=0;
   for (l=0;l<8;l++)
   {
    sum +=Hadamard[l+i*8] * in[j+l*8];
    D[i][j]=sum;
   }
  }
 }
 for (x=0;x<8;x++)
 {
  for (y=0;y<8;y++)
  {
   suma=0;
   for (z=0;z<8;z++)
   {
    suma +=D[x][z] * Hadamard[y+z*8];
    out[y+x*8]=suma/8;
   }
  }
 }
}
```

NOMBRE: Almacenamiento HADAMARD inversa 2.c
DESCRIPCION: Archivo que contiene la función que almacena las sub-imagenes descomprimdas y permite obtener un aproximado de la imagen original de la huella.

```
void  Almacenamiento(float*in,unsigned char*out,int*x)
{
 int b,c;
 for (b=0;b<8;b++)
 {
  for (c=0;c<8;c++)
  {
   out[64*(*x)+8*b+c]=(unsigned char)in[8*b+c];
  }
 }
}
```

ANEXO I

CÓDIGO FUENTE PARA LA COMPRESIÓN HADAMARD (NIVEL 4:1).

NOMBRE: Compresión HADAMARD 4.c
DESCRIPCIÓN: Archivo principal donde se implementa la compresión Hadamard nivel 4 a 1 de compresión.

```
#pragma DATA_SECTION(imagen,"picture");
unsigned char imagen[208][160];
float temp [8][8];
float transfor[8][8];
float estandar[8][8];
float temp_1[16];
float hadamard[104][80];
int a,b,c,d,e,f,n=0;
unsigned char *imageptr;
void main()
{
 imageptr=&imagen[0][0];
 for(a=0;a<520;a++)
 {
  for (c=0;c<8;c++)
  {
   for (d=0;d<8;d++)
   {
    temp[c][d]=(float)imageptr[64*a+8*c+d];
   }
  }
  transformada_Hadamard(&temp[0][0],&transfor[0][0]);
  Cuantificacion(&transfor[0][0],&estandar[0][0]);
  Recorte(&estandar[0][0],temp_1);
  Almacenamiento_HADAMARD(temp_1,&hadamard[0][0],&n);
 }
 exit();
}
```

NOMBRE: Transformación_HADAMARD 4.c
DESCRIPCION: Archivo que contiene la función que aplica la transformada Hadamard a cada una de las sub-imagenes de la huella digital.

```
void transformada_Hadamard(float*temp,float*Transfor)
{

 float D[8][8];
 float Hadamard[64]= {1, 1, 1, 1, 1, 1, 1, 1,
                      1, 1, 1, 1,-1,-1,-1,-1,
                      1, 1,-1,-1,-1,-1, 1, 1,
                      1, 1,-1,-1, 1, 1,-1,-1,
                      1,-1,-1, 1, 1,-1,-1, 1,
                      1,-1,-1, 1,-1, 1, 1,-1,
                      1,-1, 1,-1,-1, 1,-1, 1,
                      1,-1, 1,-1, 1,-1, 1,-1};
 unsigned int i,j,l,x,y,z;
 float sum,suma;
 for (i=0;i<8;i++)
 {
  for (j=0;j<8;j++)
  {
   sum=0;
   for (l=0;l<8;l++)
   {
    sum +=Hadamard[l+i*8] * temp[j+l*8];
    D[i][j]=sum;
   }
  }
 }
 for (x=0;x<8;x++)
 {
  for (y=0;y<8;y++)
  {
   suma=0;
   for (z=0;z<8;z++)
   {
    suma +=D[x][z] * Hadamard[y+z*8];
    Transfor[y+x*8]=suma/8;
   }
  }
 }
}
```

NOMBRE: Cuantificación HADAMARD 4.c
DESCRIPCION: Archivo que contiene la función que cuantifica cada una de las sub-imagenes de la huella digital.

```
#include <math.h>
#define value 8
float normal [value][value]={1,1,1,1,0,0,0,0,
                             1,1,1,1,0,0,0,0,
                             1,1,1,1,0,0,0,0,
                              1,1,1,1,0,0,0,0,
                              0,0,0,0,0,0,0,0,
                              0,0,0,0,0,0,0,0,
                              0,0,0,0,0,0,0,0,
                              0,0,0,0,0,0,0,0};

void Cuantificacion(float *entrada, float *estandar)
{
 int d,e;
 for(d=0;d<value;d++)
 {
```

```
  for(e=0;e<value;e++)
  {
   estandar[8*d+e] = entrada[8*d+e]*normal[d][e];
   if(estandar[8*d+e] <0)
   {
    estandar[8*d+e] = ceil(estandar[8*d+e]);
   }
   else
   {
    estandar[8*d+e] = floor(estandar[8*d+e]);
   }
  }
 }
}
```

NOMBRE: Recorte HADAMARD 4.c
DESCRIPCION: Archivo que contiene la función toma, para luego almacenarla, todos los valores de las sub-imagenes diferentes de cero.

```
void Recorte(float*in,float*out )
{
 int i,j;
 for(i=0;i<4;i++)
 {
  for(j=0;j<4;j++)
  {
   out[4*i+j]=in[8*i+j];
  }
 }
}
```

NOMBRE: Almacenamiento HADAMARD 4.c
DESCRIPCION: Archivo que contiene la función almacena la imagen de la huella digital comprimida.

```
void Almacenamiento_HADAMARD(float*in,float*out,int*n)
{
 int i,j,k;
 k=*n;
 for(i=0;i<4;i++)
 {
  for(j=0;j<4;j++)
  {
   out[k+(4*i)+j]=in[4*i+j];
  }
 }
 k=k+16;
 *n=k;
}
```

ANEXO J

CÓDIGO FUENTE PARA LA DESCOMPRESIÓN HADAMARD (NIVEL 4:1).

NOMBRE: Descompresión HADAMARD 4.c
DESCRIPCIÓN: Archivo que contiene el algoritmo para realizar la descompresión de la imagen de la huella digital.

```
#pragma DATA_SECTION(imagen,"picture");
unsigned char imagen[208][160];

float dato[104][80];
float temp[16];
float estandar[8][8];
float transfor_inv[8][8];
float *TEMP,*DATO;
int n,i,x=0,y=0;
void main()
{
 TEMP=&temp[0];
 DATO=&dato[0][0];
 for(n=0;n<520;n++)
 {
  for(i=0;i<16;i++)
  {
   TEMP[i]=DATO[n*16+i];
  }
  Conversion(temp,&estandar[0][0]);
  Transformacion(&estandar[0][0],&transfor_inv[0][0]);
  Almacenamiento(&transfor_inv[0][0],&imagen[0][0],&x);
  x=x+1;
 }
 exit();
}
```

NOMBRE: Conversión 4.c
DESCRIPCION: Archivo que contiene la función que permite la decuantificación de cada una de las sub-imagenes a descomprimir.

```
void Conversion(float*in,float*out)
{
 int i,j;
 for(i=0;i<8;i++)
 {
  for(j=0;j<8;j++)
  {
   out[8*i+j]=0;
  }
```

```
 }
 for(i=0;i<4;i++)
 {
  for(j=0;j<4;j++)
  {
   out[8*i+j]=in[4*i+j];
  }
 }
}
```

NOMBRE: Transformación Inversa HADAMARD 4.c
DESCRIPCION: Archivo que contiene la función que aplica la transformada inversa Hadamard a cada una de las sub-imagenes.

```
float D[8][8];
void Transformacion(float*in,float*out)
{
 float Hadamard[64]= {1, 1, 1, 1, 1, 1, 1, 1,
                      1, 1, 1, 1,-1,-1,-1,-1,
                      1, 1,-1,-1,-1,-1, 1, 1,
                      1, 1,-1,-1, 1, 1,-1,-1,
                      1,-1,-1, 1, 1,-1,-1, 1,
                      1,-1,-1, 1,-1, 1, 1,-1,
                      1,-1, 1,-1,-1, 1,-1, 1,
                      1,-1, 1,-1, 1,-1, 1,-1};
 unsigned int i,j,l,x,y,z;
 float sum,suma;
 for (i=0;i<8;i++)
 {
  for (j=0;j<8;j++)
  {
   sum=0;
   for (l=0;l<8;l++)
   {
    sum +=Hadamard[l+i*8] * in[j+l*8];
    D[i][j]=sum;
   }
  }
 }
 for (x=0;x<8;x++)
 {
  for (y=0;y<8;y++)
  {
   suma=0;
   for (z=0;z<8;z++)
   {
    suma +=D[x][z] * Hadamard[y+z*8];
    out[y+x*8]=suma/8;
   }
  }
 }
}
```

NOMBRE: Almacenamiento HADAMARD inversa 4.c
DESCRIPCION: Archivo que contiene la función que almacena las sub-imagenes descomprimdas y permite obtener un aproximado de la imagen original de la huella.

```
void  Almacenamiento(float*in,unsigned char*out,int*x)
{
 int b,c;
 for (b=0;b<8;b++)
 {
  for (c=0;c<8;c++)
  {
   out[64*(*x)+8*b+c]=(unsigned char)in[8*b+c];
  }
 }
}
```

ANEXO K

CÓDIGO FUENTE PARA LA COMPRESIÓN BINARIA.

NOMBRE: main.c
DESCRIPCIÓN: Archivo que contiene el algoritmo que permite comprimir en forma binaria las imágenes.

```
#pragma DATA_SECTION (imagen,"picture")
#pragma DATA_SECTION (binario,"picture")

unsigned char imagen[208][160];
unsigned char binario[208][160];
unsigned char comp [4160];

void main()
{
 Binarizacion(imagen,binario);
 Compresion(binario,comp);
 exit();
}
```

NOMBRE: Binarización.c
DESCRIPCION: Archivo que contiene la función que binariza la imagen de la huella.

```
void Binarizacion(unsigned char*in,unsigned char*out )
{
 int i;
 for(i=0;i<33280;i++)
 {
  if(in[i]<=180)
  {
   out[i]=0;
  }
  else
  {
   out[i]=255;
  }
 }
}
```

NOMBRE: Compresión.c
DESCRIPCION: Archivo que contiene la función que empaqueta los bit que conforman la imagen Binarizada, guarda 8 píxeles por cada byte, almacena la imagen comprimida.

```
void Compresion(unsigned char*in,unsigned char*out)
{
 unsigned char temp=0,salida=0;
 int i,j=0,k=0;
 for(i=0;i<33280;i++)
 {
  temp=in[i];
  temp=temp<<j;
  salida=salida|temp;
  j+=1;
  if(j==8)
  {
   out[k]=salida;
   salida=0;
   k+=1;
   j=0;
  }
 }
}
```

ANEXO L

CÓDIGO FUENTE PARA LA DESCOMPRESIÓN BINARIA.

NOMBRE: main.c
DESCRIPCIÓN: Archivo que contiene el algoritmo que permite descomprimir la imagen Binarizada.

```
#pragma DATA_SECTION (imagen,"picture")
#pragma DATA_SECTION (dato,"picture")
unsigned char imagen[208][160];
unsigned char dato [4160];

void main()
{
 descompresion(dato,imagen);
 decodificacion(imagen);
 exit();
}
```

NOMBRE: descompresión.c
DESCRIPCION: Archivo que contiene la función que desempaqueta cada uno de los bits de la imagen comprimida y los convierte en bytes.

```
#include <math.h>
void descompresion(unsigned char*in,unsigned char*out)
{
 unsigned char temp,tempo;
 int a,i,j,k;
 for(i=0;i<4160;i++)
 {
  temp=in[i];
  for(j=0;j<8;j++)
  {
   a=pow(2,j);
   tempo=temp&a;
   tempo=tempo>>j;
   out[k]=tempo;
   k+=1;
  }
 }
}
```

NOMBRE: decodificación.c
DESCRIPCION: Archivo que contiene la función que permite asignar a cada píxel de la imagen un valor: blanco o negro para visualizar la imagen binaria.

```
void decodificacion(unsigned char*in)
{
 int i;
 for(i=0;i<33280;i++)
 {
  if(in[i]==1)
  {
   in[i]=255;
  }
 }
}
```

ANEXO M

TABLAS RESUMEN
RESULTADOS DE LA VERIFICACIÓN
DE LAS HUELLAS PROCESADAS.

Tabla de aceptaciones y rechazos de la huella digital 1_1

A: Aceptada R: Rechazada

	Original	Hadamard 1:4	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	A	A	A	A	A
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 1_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	A	R	R	A	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 1_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	A	A	A	A	A
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 1_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	A	R	R	A	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 1_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	A	R	R	A	A
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 2_1

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	A	R	A	A	A
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 2_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	A	R	A	A	A
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 2_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	A	R	A	A	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 2_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	A	R	R	A	A
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 2_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	A	A	R	A	A
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 3_1

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	A	R	R	A	A
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 3_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	A	R	R	A	A
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 3_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	A	R	A	A	A
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 3_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	A	A	A	A	A
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 3_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	A	A	A	A	A
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 4_1

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	A	A	A	A	A
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 4_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	A	A	A	A	A
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 4_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	A	R	A	A	A
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 4_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	A	A	A	A	A
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 4_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	A	R	R	A	A
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 5_1

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	A	A	A	A	A
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 5_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	A	R	R	A	A
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 5_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	A	A	A	A	A
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 5_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	A	R	A	A	A
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 5_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	A	R	R	A	A
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 6_1

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	A	R	A	A	A
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 6_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	A	A	A	A	A
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 6_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	A	A	A	A	A
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 6_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	A	R	A	A	A
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 6_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	A	A	A	A	A
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 7_1

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	A	A	A	A	A
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 7_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	A	R	R	R	A
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 7_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	A	A	A	A	A
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 7_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	A	A	A	A	A
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 7_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	A	R	R	A	A
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 8_1

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	A	R	R	A	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 8_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	A	A	A	A	A
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 8_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	A	A	A	A	A
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 8_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	A	A	A	A	A
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 8_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	A	R	A	A	A
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 9_1

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	A	R	A	A	A
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 9_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	A	A	A	A	A
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 9_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	A	A	R	A	A
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 9_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	A	R	R	A	A
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 9_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	A	R	R	A	A
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 10_1

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	A	A	A	A	A
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 10_2

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	A	R	A	A	A
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 10_3

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	A	R	R	A	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 10_4

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	A	A	A	A	A
Usuario 10_5	R	R	R	R	R

Tabla de aceptaciones y rechazos de la huella digital 10_5

A: Aceptada R: Rechazada

	Original	Hadamard	JPEG	Binarizada	Hadamard 1:1,78
Usuario 1_1	R	R	R	R	R
Usuario 1_2	R	R	R	R	R
Usuario 1_3	R	R	R	R	R
Usuario 1_4	R	R	R	R	R
Usuario 1_5	R	R	R	R	R
Usuario 2_1	R	R	R	R	R
Usuario 2_2	R	R	R	R	R
Usuario 2_3	R	R	R	R	R
Usuario 2_4	R	R	R	R	R
Usuario 2_5	R	R	R	R	R
Usuario 3_1	R	R	R	R	R
Usuario 3_2	R	R	R	R	R
Usuario 3_3	R	R	R	R	R
Usuario 3_4	R	R	R	R	R
Usuario 3_5	R	R	R	R	R
Usuario 4_1	R	R	R	R	R
Usuario 4_2	R	R	R	R	R
Usuario 4_3	R	R	R	R	R
Usuario 4_4	R	R	R	R	R
Usuario 4_5	R	R	R	R	R
Usuario 5_1	R	R	R	R	R
Usuario 5_2	R	R	R	R	R
Usuario 5_3	R	R	R	R	R
Usuario 5_4	R	R	R	R	R
Usuario 5_5	R	R	R	R	R
Usuario 6_1	R	R	R	R	R
Usuario 6_2	R	R	R	R	R
Usuario 6_3	R	R	R	R	R
Usuario 6_4	R	R	R	R	R
Usuario 6_5	R	R	R	R	R
Usuario 7_1	R	R	R	R	R
Usuario 7_2	R	R	R	R	R
Usuario 7_3	R	R	R	R	R
Usuario 7_4	R	R	R	R	R
Usuario 7_5	R	R	R	R	R
Usuario 8_1	R	R	R	R	R
Usuario 8_2	R	R	R	R	R
Usuario 8_3	R	R	R	R	R
Usuario 8_4	R	R	R	R	R
Usuario 8_5	R	R	R	R	R
Usuario 9_1	R	R	R	R	R
Usuario 9_2	R	R	R	R	R
Usuario 9_3	R	R	R	R	R
Usuario 9_4	R	R	R	R	R
Usuario 9_5	R	R	R	R	R
Usuario 10_1	R	R	R	R	R
Usuario 10_2	R	R	R	R	R
Usuario 10_3	R	R	R	R	R
Usuario 10_4	R	R	R	R	R
Usuario 10_5	A	R	R	A	A

BIBLIOGRAFÍA

1. **GONZÁLEZ RAFAEL C., WOODS RICHARD E.**; Tratamiento digital de imágenes; Addison - Wesley Iberoamérica; Estados Unidos 1996; Capítulo 3: páginas 149,156-157, y Capítulo 6: páginas 335-352,373-439.

2. **MALTONI DAVIDE, MAIO DARIO, JAIN ANIL K. PRABHAKAR SALIL**; Handbook of Fingerprint Recognition; Springer Science+Bussiness Media, Inc; Estados Unidos 2003; Capítulo 1: páginas 3-4, Capítulo 2: páginas 53-57, 59-65, y Capítulo 3: páginas 83-87.

3. **HTTP:// WWW.FUAC.EDU.CO/AUTONOMA/PREGRADO/INGENIERIA/INGELEC/PROYECTOSGRADO/COMPRESVIDEO/INDEX.HTM;** Compresión de Video Digital: Bajo los Estándares MPEG; Capítulo 6: páginas 1, 3 y 6.

4. **TEXAS INSTRUMENTS**; FPC1010 Sensor Drivers Quick Start Guide; Estados Unidos, Julio 2003; página 2.

5. **FRINGERPRINT CARDS AB**; User Manual FPCore Demo For Texas Instruments TMS320C6713 and TMS320C5510 DSP Started Kits (Revisión D); Suecia, Noviembre 2003; páginas 3 y 12.

6. **TEXAS INSTRUMENTS**; Product Bulletin Fringerprint Authentication Development Tool; Estados Unidos, 2004; páginas 1 y 3.

7. **SPECTRUM DIGITAL INC**; FPC1010 Finger Print Sensor Daughter Card Technical Reference; Estados Unidos, Julio 2003; Capítulo 1: páginas 1_2 y 1_3.

8. **SPECTRUM DIGITAL INC**; TMS320C6713 DSK Technical Reference; Estados Unidos, Mayo 2003; Capítulo 1: páginas 1_2 y 1_5.

9. **TEXAS INSTRUMENTS**; TMS320C6000 DSP External Memory Interface (EMIF), Reference Guide; Estados Unidos, Marzo 2003; Capítulo 1: página 14.

10. **TEXAS INSTRUMENTS**; TMS320C6000 Peripherals Reference Guide; Estados Unidos, Febrero 2001; Capítulo 7: página 7_2 y Capítulo 17: página 17_2.

11. **TEXAS INSTRUMENTS**; TMS320C6713, TMS320C6713B Floating- Point Digital Signal Processor; Estados Unidos, Febrero 2005; páginas 14 y 17.

12. **TEXAS INSTRUMENTS**; TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Controller Reference Guide; Estados Unidos, Marzo 2005; Capítulo 1: páginas 1_2 y 1_19.

13. **TEXAS INSTRUMENTS**; TMS320C6000 DSP Multichannel Buffered Serial Port (McBSP) Reference Guide; Estados Unidos, Septiembre 2004; Capítulo 1: página 11 y Capítulo 8: página 69.