UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo e implementación de un sistema completo de comunicación y gestión de alarmas a través de internet mediante el protocolo SIA DC-09-2021

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Moreno Maynero, Vicente

Tutor/a: Monserrat del Río, José Francisco

Director/a Experimental: LOPEZ MUÑOZ, PABLO

CURSO ACADÉMICO: 2023/2024

Resumen

Este trabajo final de grado desarrolla un módulo transmisor y un módulo receptor de mensajes de alarma empleando el protocolo SIA DC-09-2021, centrándose específicamente en la integración dentro de un sistema de detección de drones mediante cámaras de vigilancia. El proyecto se plantea como una solución completa, desarrollando tanto el envío y recepción del mensaje de alarma como el envío y recepción del video asociado a la misma. Así mismo, las funciones de receptor se amplían crear un entorno accesible para el usuario final mediante el envío de mensajes de texto para la notificación de las alarmas y la creación de una interfaz web con todos los datos de las alertas asociadas a cada usuario. En este contexto, el documento analiza en profundidad el protocolo de alarmas SIA DC-09-2021, haciendo hincapié en las funciones que permiten establecer una comunicación segura, así como la manera en la que el protocolo permite el envío de los datos, tanto de posición del objeto detectado como de video. Por otro lado, el proyecto analiza en detalle los protocolos de nivel de transporte soportados por el protocolo y se centra en UDP para el desarrollo de este. Además, incidiendo en lo crítico de la seguridad en el manejo de mensajes de alarma, este documento analiza en detalle e implementa el protocolo DTLS con el objetivo de crear una comunicación segura y resistente a todo tipo de ataques. Finalmente, se valida la funcionalidad del sistema mediante pruebas en un entorno real, asegurando la transmisión y recepción de todos los mensajes y la resistencia frente a posibles pérdidas de paquetes.

Palabras Clave: SIA; SIA DC-09-2021; Alarma; DTLS

Resum

Aquest treball final de grau desenvolupa un mòdul transmissor i un mòdul receptor de missatges d'alarma emprant el protocol SIA DC-09-2021, centrant-se específicament en la integració dins d'un sistema de detecció de drons mitjançant càmeres de vigilància. El projecte es planteja com una solució completa, desenvolupant tant l'enviament i recepció del missatge d'alarma com l'enviament i recepció del vídeo associat a aquesta. Així mateix, les funcions de receptor s'amplien per a crear un entorn accessible per a l'usuari final mitjançant l'enviament de missatges de text per a la notificació de les alarmes i la creació d'una interfície web amb totes les dades de les alertes associades a cada usuari. En aquest context, el document analitza en profunditat el protocol d'alarmes SIA DC-09-2021, posant l'accent en les funcions que permeten establir una comunicació segura, així com la manera en la qual el protocol permet l'enviament de les dades, tant de posició de l'objecte detectat com de vídeo. D'altra banda, el projecte analitza detalladament els protocols de nivell de transport suportats pel protocol i es centra en UDP per al desenvolupament d'aquest. A més, incidint en el crític de la seguretat en el maneig de missatges d'alarma, aquest document analitza detalladament i implementa el protocol DTLS amb l'objectiu de crear una comunicació segura i resistent a tota mena d'atacs. Finalment, es valida la funcionalitat del sistema mitjançant proves en un entorn real, assegurant la transmissió i recepció de tots els missatges i la resistència enfront de possibles pèrdues de paquets.

Paraules clau: SIA; SIA DC-09-2021; Alarma; DTLS

Abstract

This final degree project develops a transmitter module and a receiver module for alarm messages using the SIA DC-09-2021 protocol, focusing specifically on integration within a drone detection system using surveillance cameras. The project is proposed as a complete solution, developing both the sending and receiving of the alarm message and the sending and receiving of the associated video. Likewise, the receiver functions are extended to create an accessible environment for the end user by sending text messages for alarm notification and creating a web interface with all the alert data associated with each user. In this context, the document analyses in depth the SIA DC-09-2021 alarm protocol, emphasising the functions that allow secure communication to be established, as well as the way in which the protocol allows data to be sent, both the position of the detected object and the video. On the other hand, the project analyses in detail the transport level protocols supported by the protocol and focuses on UDP for the development of the protocol. In addition, stressing the criticality of security in the handling of alarm messages, this document analyses in detail and implements the DTLS protocol with the aim of creating a secure communication that is resistant to all types of attacks. Finally, the functionality of the system is validated by testing in a real environment, ensuring the transmission and reception of all messages and the resilience against possible packet loss.

Key words: SIA; SIA DC-09-2021; Alarm; DTLS

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la ingeniería de telecomunicación

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1-2
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	3-8
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	2
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	9-22
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	8,10,13,15, 21-22
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	23-28
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	29-30

Índice

Capítulo 1. Introducción y objetivos

1.1 Introducción

La evolución en la industria del presente siglo conduce a que la tecnología esté cada vez más presente en las vidas de todos los ciudadanos. Elementos que hace menos de veinte años parecían ciencia ficción, se han abierto paso hasta estar al alcance de cualquier persona. Un ejemplo de estos elementos son los drones, vehículos aéreos tripulados a control remoto que pueden contar con cámaras de video, micrófonos u otros dispositivos que permiten la vigilancia y el control. Aunque estos vehículos suponen un importante apoyo en sectores tan esenciales como el control de incendios o el rescate de personas, también son utilizados para burlar la seguridad de instalaciones públicas o privadas de todo tipo debido a su pequeño tamaño y fácil manejo. En este sentido, resulta fundamental que los elementos de seguridad se adapten a este entorno en constante evolución.

Al tratarse de información sensible, la industria y el cliente buscan tanto fiabilidad como seguridad en las comunicaciones. Es indispensable que cualquier sistema de comunicación de alarmas no pueda ser interceptado por ataques tipo Man-in-the-Middle. Si este punto falla, resulta inútil contar con el sistema de seguridad más avanzado ante intrusiones, ya que un segundo atacante podría suplantar al sistema de seguridad y hacer creer a la central receptora de alarmas que no hay ningún peligro. Este hecho resulta especialmente importante cuando la comunicación se desarrolla a través de internet, el medio de comunicación más importante actualmente. Este medio, aunque resulta indiscutible que es verdaderamente útil tanto por sus características como por su casi total implantación, es objeto de ataques de todo tipo y es crucial impedir que estos sean capaces de suponer una brecha en un sistema de seguridad.

Para la comunicación a través de internet se puede optar por dos protocolos de capa de transporte, User Datagram Protocol (UDP) y Transmission Control Protocol (TCP). La diferencia entre ambos está en el grado de control sobre la transmisión de cada paquete, así como en la complejidad. El protocolo UDP permite una comunicación sencilla, en la que no se establece una conexión permanente pero tampoco hay ningún control de la recepción de los paquetes. El protocolo TCP, en cambio, sí realiza un control de la recepción de cada paquete de datos, sin embargo, es más lento y pesado en términos de congestión de un servidor. En este proyecto, debido a los requerimientos técnicos del proyecto, se hace uso del protocolo UDP.

Como se ha mencionado, en este proyecto es indispensable la seguridad en las comunicaciones. Por ello, se hace uso de Datagram Transport Layer Security (DTLS). Este protocolo de seguridad de la capa de transporte es el encargado de que el envío y recepción de paquetes mediante el protocolo UDP no pueda ser ni interceptado, ni sustituido por otro mensaje malicioso. Gracias a esto, se puede garantizar que la comunicación entre el sistema de seguridad y la central receptora de alarmas no se vea comprometida.

En cuanto al envío de mensajes, es necesario seguir un protocolo a la hora de comunicar las alertas, de forma que el sistema pueda adaptarse a los requerimientos de una empresa de seguridad. Por ello, en el presente Trabajo Final de Grado se emplean los protocolos Contact ID y SIA DC-09-2021 para dar formato a los mensajes de alarma. Además, también se transmitirá un pequeño video en formato mp4 con la detección asociada a la alerta.

Por otro lado, la arquitectura en la que se va a basar el diseño de la comunicación es cliente-servidor. Esto es así debido a que la central receptora es un punto único en la red, conocido por múltiples sistemas de alarma los cuales pretenden establecer una comunicación con ella.

Finalmente, resulta conveniente que los usuarios del sistema de seguridad puedan acceder a las alertas de manera sencilla y eficiente. Para ello, en el presente trabajo se creará una interfaz web para poder acceder a las alarmas de cada usuario y los videos de detección asociados a las mismas. Además, se desarrollará un sistema de envío de mensajes de texto vía WhatsApp donde el usuario recibirá las alertas en tiempo real.

1.2 Objetivos

Este Trabajo Final de Grado tiene una serie de objetivos orientados a la realización de un sistema completo de comunicación de mensajes de alarma. En línea con esto, el objetivo principal de este TFG es desarrollar un sistema de comunicación cliente – servidor usando el protocolo SIA DC-09-2021.

Como objetivos secundarios destacamos los siguientes:

- Integrar el cliente dentro de un sistema de detección de drones mediante Inteligencia Artificial.
- Implementar un entorno para la recepción de las alarmas por parte del usuario final.
- Validar el funcionamiento completo del sistema en un entorno real.

Además, otra meta de este Trabajo Final de Grado es contribuir al desarrollo sostenible de la sociedad. Por tanto, en el Anexo I se profundizará acerca de la contribución de este proyecto a los Objetivos de Desarrollo Sostenible y la Agenda 2030.

1.3 Estructura del documento

Este Trabajo Final de Grado está formado por cinco capítulos que se detallan a continuación:

- **Capítulo 1. Introducción y objetivos:** en este capítulo se introducen las motivaciones y tecnologías empleadas en el proyecto.
- **Capítulo 2. Marco teórico:** en este capítulo se detallan las herramientas teóricas que sirven de base para el desarrollo de la solución.
- **Capítulo 3. Metodología:** este capítulo describe todos los pasos seguidos desde el establecimiento de los requisitos hasta la validación del sistema.
- **Capítulo 4. Desarrollo y resultados del trabajo:** en este capítulo se detalla la creación tanto del cliente como del servidor, así como del entorno para los usuarios.
- **Capítulo 5. Conclusiones y propuesta de trabajo futuro:** este capítulo final engloba las enseñanzas y resultados obtenidos con este proyecto, así como futuras ampliaciones posibles.

Capítulo 2. Marco teórico

Este capítulo proporciona los conocimientos teóricos empleados para llevar a cabo el presente Trabajo Final de Grado. Se abordarán en profundidad los conceptos de Contact ID, SIA DC-09-2021, UDP, DTLS y arquitectura cliente-servidor.

2.1 Protocolo Contact ID

El protocolo Contact ID es un protocolo desarrollado por Ademco, una empresa proveedora de soluciones de seguridad. Este protocolo ha sido cedido a la Security Industry Association (SIA) para su uso libre. Gracias a esto, está ampliamente estandarizado en la industria de la seguridad [1]. Este protocolo establece tanto la estructura de los mensajes como los diferentes mensajes de alarma posibles.

La composición de un mensaje de alarma según el protocolo Contact ID es "ACCT MT QXYZ GG CCC S". Las cuatro primeras letras (ACCT) representan el código de cuenta, el cual se compone de 4 cifras hexadecimales (0-9, A-F). Las dos siguientes (MT) representan que el mensaje sigue el protocolo Contact ID, estas pueden ser un 18 o un 98. La siguiente letra (Q) representa el calificador de evento, este puede ser 1, para eventos nuevos; 3, para eventos ya solucionados o 6, para eventos notificados todavía presentes. Las tres siguientes letras (XYZ) representan el código de evento en sí. Este se compone por tres cifras hexadecimales, pero no utiliza la A (0-9, B-F). Los dos grupos de letras siguientes (GG y CCC) representan la partición y zona respectivamente. Siguen el mismo formato que el bloque anterior (0-9, B-F). Finalmente, S es el checksum del mensaje. Una comprobación de que el mensaje enviado y recibido no ha sufrido alteraciones [1].

Hay una gran cantidad de códigos de evento, los cuales se agrupan en seis bloques según su tipo. Los códigos de evento del grupo 100 son alarmas de seguridad, el grupo más extenso y principal en este protocolo. Los códigos de evento del grupo 200 son alarmas de supervisión. Los códigos de evento del grupo 300 son alertas de problemas en el sistema. Las alertas del grupo 400 son alarmas de apertura, cierre o acceso remoto. Las alertas del grupo 500 indican la indisponibilidad de algún elemento del sistema, como una campana. Finalmente, las alertas del grupo 600 son alertas de prueba o programación. La Figura 1 muestra los grupos de alertas mencionados.

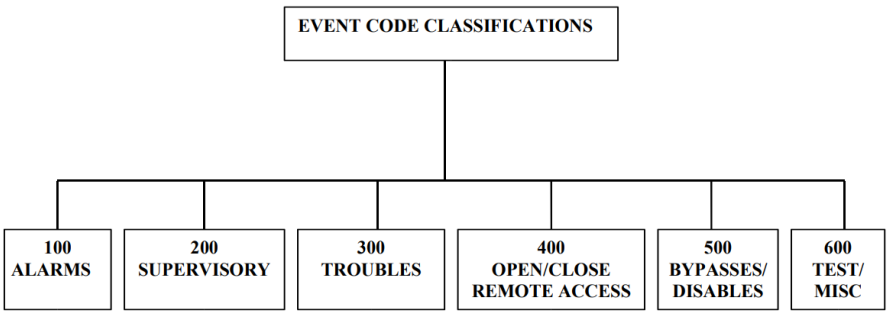

Figura 1. Esquema de códigos de evento.
Fuente: **[1]**

En el contexto de este proyecto, se empleará el código 133 (24h safe) para indicar la activación del sistema. Se empleará el código 131 (Perimeter) para indicar una intrusión. También se empleará el código 300 (System trouble) para indicar que el sistema no funciona.

2.2 SIA DC-09-2021

El protocolo Contact ID data de 1999 y su principal problema es que este protocolo no recoge el envío de mensajes por internet, sino que indica que los mensajes de alarma se deben transmitir mediante tonos DTMF, es decir, por línea telefónica. Por ello, y por agregar más funcionalidades como otros datos relevantes de la alerta, se debe emplear el protocolo SIA DC-09-2021.

El protocolo SIA DC-09-2021 se utiliza para reportar eventos desde las instalaciones donde se encuentre el equipamiento hasta la central receptora de alarmas empleando el protocolo de internet (IP) [2]. Analizando la compatibilidad de este protocolo con los protocolos de capa de transporte, SIA DC-09-2021 es compatible tanto con UDP como con TCP [2]. En este proyecto se utiliza UDP debido a que es un requerimiento previo del usuario final de este sistema. El protocolo estudiado en este apartado es muy completo y también completa los mensajes de alarma de Contact ID con información relevante.

Analizando el formato de los mensajes, este protocolo absorbe y completa lo ya mencionado en el Apartado 2.1, añadiendo los mensajes de mantenimiento y los datos extendidos. Los mensajes de mantenimiento son mensajes recurrentes que se mandan sin información. Solo sirven para verificar que el sistema funciona correctamente y que no ha sufrido un corte. Por otro lado, los datos extendidos son datos que completan un mensaje de alarma. Hay varios tipos de datos extendidos, cada uno se identifica por una letra, pero en este proyecto se van a utilizar tres. X, Y y Z se enviarán junto a un mensaje de alerta para indicar la posición de un objeto detectado. V se enviará como un segundo mensaje de alerta indicando los principales datos del vídeo que va a acompañar la alerta. La plantilla de mensaje de cada evento es:

<LF><crc><0LLL><"id"><seq><Rrcvr><Lpref><#acct>[<pad>|...data...][x…data…]<timestamp><CR>

Explicando por partes, <LF> representa el carácter ASCII de avance de línea (0x0A). <crc> es un código de comprobación, similar a un hash, que sirve de comprobación de la integridad del mensaje. <0LLL> es un 0 seguido de la longitud del mensaje. <"id"> es el protocolo usado para el mensaje al cual se está envolviendo. Podrá ser "ADM-CID" si se trata de un mensaje de alarma usando el protocolo Contact ID o "NULL" si se envía un mensaje de mantenimiento. Si el mensaje está cifrado, se incluirá un asterisco (*) justo después de las primeras comillas. <seq> es un número de secuencia de cuatro cifras propio de cada mensaje de alarma. Si llega a 9999, empieza a contar desde 0001 otra vez. <Rrcvr>, <Lpref> y <#acct> son la identificación de la cuenta del cliente. Número de receptor, prefijo de la cuenta y número de la cuenta (el mismo que el indicado en la sección anterior). Todos en hexadecimal e incluyen la R, la L y la almohadilla (#) al inicio de cada uno respectivamente. <pad> sirve de relleno para que los mensajes cifrados tengan una longitud múltiplo de 16. Además, siempre se repite <#acct> antes del | en los mensajes tipo Contact ID. …data… es el mensaje tipo Contact ID tal cual como se ha explicado en el apartado anterior. x…data… es un tipo de datos extendido que completa el mensaje. <timestamp> es la fecha en formato epoch y <CR> representa el carácter ASCII para el retorno de carro (0x0D). En resumen, un mensaje completo sería:

98BB0099"ADM-CID"0210RA0L23A#111A[#111A|111A 18 1131 0B 0004][X2.45Y3Z8.443]_19:59:02,05-13-2024

Por último, este protocolo permite el cifrado del mensaje, más allá de que el paquete se cifre o no para su transmisión por internet. Este cifrado se realiza mediante AES y requiere de una clave compartida de 128, 192 o 256 bits.

Como conclusión, el protocolo SIA DC-09-2021 permite el envío de mensajes de alarma más completos para añadir información relevante a cada alerta. Además, este protocolo permite utilizar internet para la transmisión de mensajes de alarma, lo que resulta de gran utilidad si se guardan las debidas medidas de seguridad.

2.3 User Datagram Protocol (UDP)

Como se ha mencionado en la sección anterior, el protocolo de capa de transporte empleado en este Trabajo Final de Grado es UDP. Esto es así debido a los requerimientos por parte del usuario que utilizará el resultado de este proyecto. Por tanto, resulta conveniente entender en profundidad los detalles de este protocolo. UDP se puede entender como una versión reducida de TCP y se explica de manera precisa haciendo la comparación entre ambos. En la Figura 2 se puede observar una comparación entre UDP y TCP.

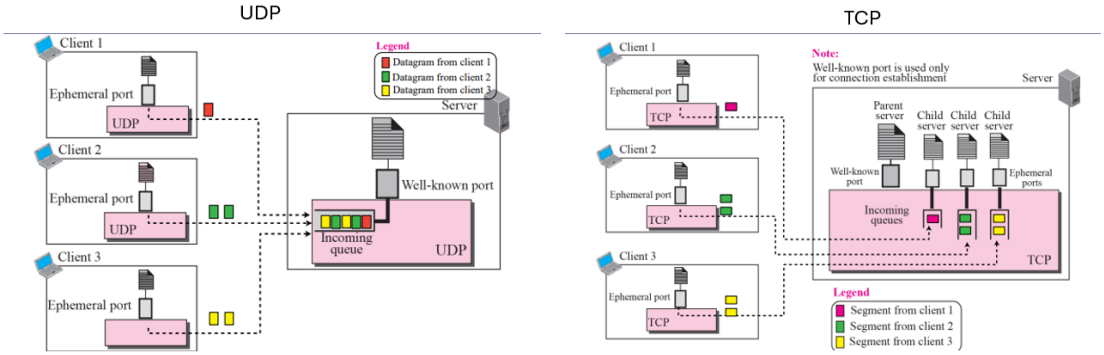

Figura 2. Comparación entre UDP y TCP.
Fuente: [3]

UDP es un protocolo de capa de transporte sin conexión, es decir, no se establece un circuito virtual entre los dos puntos de la red que se intentan comunicar. Por ello, no se puede garantizar la entrega de todos los paquetes que se envían, al contrario de lo que sucede con TCP. Por otro lado, UDP utiliza muchos menos recursos que TCP, lo que supone una ventaja en entornos en los que la calidad de señal es débil. Además, debido a que UDP no hace ningún seguimiento de la llegada de los paquetes de datos, este protocolo de capa de transporte genera menos retrasos en la comunicación, por lo que se utiliza en aplicaciones sensibles al retardo como las llamadas de voz sobre IP [4].

La cabecera de un paquete UDP está compuesta por ocho bytes, separados en cuatro grupos de dos. Estos grupos son: puerto de origen, puerto de destino, longitud del paquete y checksum del paquete [4].

2.4 Datagram Transport Security Layer (DTLS)

Una vez analizados los principales rasgos de UDP, se observa que en ningún momento se hace referencia a la seguridad de las comunicaciones. Si se envía a través de internet un mensaje empleando solamente UDP, resulta imposible asegurar que no esté habiendo algún tipo de ataque de tipo Man-in-the-Middle, es decir, que un atacante se sitúe en algún punto del camino del datagrama con el objetivo de interceptarlo o modificarlo. Por ello, y más si cabe en un proyecto tan dependiente de la seguridad como es este Trabajo Final de Grado, resulta indispensable hacer uso de DTLS.

Datagram Transport Security Layer (DTLS) es un protocolo de seguridad en el transporte de paquetes UDP a través de internet. Sus principales utilidades son asegurar que los mensajes enviados se mantengan confidenciales, íntegros y autenticados. En el contexto del proyecto, DTLS es el protocolo encargado de que el sistema de alarmas pueda confiar en la central receptora y viceversa. Para ello, tanto cliente como servidor hacen uso de claves y certificados TLS/SSL [5].

Los certificados TLS/SSL, son las claves que hacen segura la conexión. Para establecer una conexión segura con un servidor, es necesario que este tenga una clave privada y un certificado público. El servidor transmitirá su certificado público al cliente que generará una clave llamada "secreto premaestro". Esta clave volverá a ser transmitida al servidor, el cual será el único capaz de resolver dicho secreto e iniciar la conexión. Gracias a este proceso, llamado intercambio de claves, el cliente y el servidor se identifican y reconocen sin revelar en ningún momento la clave privada del servidor. Por tanto, se establece una comunicación segura. Además, a partir de este proceso ya se pueden transmitir mensajes cifrados. Hay que tener en cuenta que, para garantizar la seguridad, las credenciales del servidor deben estar expedidas por una autoridad certificadora de confianza. No sirve de nada que una web se certifique si la autoridad certificadora no es de confianza. Por ello, existen diversas autoridades certificadoras ampliamente reconocidas [6].

Sin embargo, existen herramientas para generar autoridades certificadoras y claves tanto públicas como privadas de manera libre [7]. Estas se utilizan en entornos de pruebas, donde la seguridad no es crítica, aunque sí lo vaya a ser en la versión definitiva, y en entornos en los que una empresa quiere expedir sus propios certificados. En este proyecto se busca que el servidor sea capaz de autenticar y cifrar las comunicaciones con sus clientes, por tanto, resulta favorable el hecho de que dicho servidor cuente con una autoridad certificadora propia que otorgue los certificados a los usuarios. Para ello, en este proyecto se hace uso de OpenSSL.

2.5 Arquitectura cliente-servidor

En este Trabajo Final de Grado se emplea la arquitectura cliente-servidor. Esto es así debido a la facilidad y simpleza de manejar a múltiples clientes desde un único servidor, lo cual hace a esta arquitectura la más común de las utilizadas en todo tipo de servicios. Se llama arquitectura cliente-servidor a toda arquitectura en la que participan dos componentes, uno que utiliza unos servicios (cliente) y otro que los proporciona (servidor). En general, cualquier aplicación web usa esta arquitectura. En la Figura 3 se muestra una representación gráfica de una arquitectura cliente-servidor.

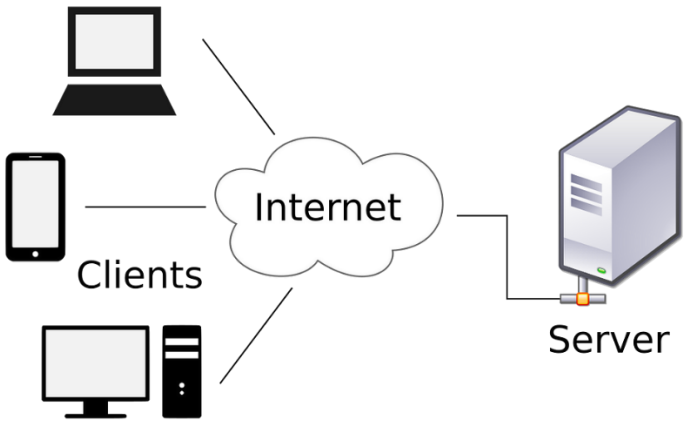

Figura 3. Representación de la arquitectura cliente-servidor.
Fuente: [8]

Analizando ambos componentes más detalladamente, el cliente es la parte que inicia las peticiones de servicio, en el contexto de este proyecto, es el que envía los mensajes de alarma. Además, los clientes no son capaces de comunicarse entre sí, al menos directamente, ya que su único contacto es el servidor. Por otro lado, no tienen que estar siempre conectados y el funcionamiento del servicio no se verá comprometido por que un cliente deje de estar presente. Generalmente, hay múltiples clientes y sobre estos no recae la complejidad de mantener el tráfico de la red por muy grande que sea esta.

Por su parte, el servidor es el núcleo central del servicio. Sobre este recae la mayor parte de la complejidad del mismo. El servidor es el encargado de atender las peticiones de todos los clientes, por lo que es la parte más sensible del sistema. Si el número de clientes aumenta sin mejorar las prestaciones del servidor, este se saturará. Además, el servidor está en una dirección IP conocida por todos los clientes, ya que estos deben conectarse a él siempre que lo necesiten. Este componente debe estar siempre conectado, ya que sin él, el sistema no puede funcionar [8].

Capítulo 3. Metodología

En este capítulo se detallan todos los pasos llevados a cabo desde el establecimiento de los requisitos de este sistema de comunicación de alarmas por parte del usuario final hasta su efectiva implantación en un entorno real. Se analizan todas las funciones requeridas para el cliente y las requeridas para el servidor, así como la manera en la que se ha llevado a cabo su implementación. Además, se detalla la manera en la que se ha desarrollado el entorno para la visualización de alarmas por parte de los usuarios.

3.1 Requisitos iniciales del sistema

El primer paso para el desarrollo del sistema de comunicación de alarmas es la definición de los requisitos iniciales exigidos para el mismo. Para ello, es necesario contar con la información proporcionada por el usuario final del sistema y ampliar esta información con los requisitos mínimos indispensables para hacer un producto final seguro y completo.

Ordenando dichos requisitos por orden de importancia, el primero de ellos es desarrollar un sistema de comunicación empleando los protocolos Contact ID y SIA DC-09-2021. Este requisito establece la base para el desarrollo de todo el sistema de comunicación, pues ambos protocolos definen el formato de los mensajes enviados y el segundo define también qué protocolos de capa de transporte son de aplicación, como se ha explicado en el Capítulo 2. El siguiente requisito inicial es el uso del protocolo de capa de transporte UDP. Pese a que se podrían utilizar tanto TCP como UDP, el usuario exige que se emplee este último, por lo que en el desarrollo tanto del cliente como del servidor solo se trabajará con UDP. El último de los requisitos iniciales definidos por el usuario final es relativo al envío de video para la notificación de las alarmas, se define inicialmente que los videos comunicados deben estar en formato mp4.

Una vez considerados los requisitos iniciales establecidos por el usuario final, estos se amplían para considerar aspectos relativos a la seguridad y a la compatibilidad con el sistema de detección de drones con el que se integrará este proyecto. En primer lugar, es necesario que el uso de UDP vaya acompañado del uso de DTLS para conseguir seguridad e integridad de los mensajes. Por otro lado, se establece que se enviarán las coordenadas donde se haya detectado al dron que cause la alarma como datos extendidos al propio mensaje de alarma. Además, se establece que el cliente deberá programarse de manera que sea integrable en el sistema de detección de drones como un objeto. Por ello, el desarrollo completo del sistema se realizará por medio del lenguaje de programación Python, ya que es el mismo lenguaje de programación que emplea el sistema de detección de drones.

Por último, se establece la manera en la que se deberán comunicar y mostrar las alarmas a los usuarios. Se establece una distinción entre la notificación de las alarmas y la visualización de estas. Respecto a la notificación, se establece que se enviará un mensaje por medio de la aplicación de mensajería instantánea WhatsApp cada vez que se detecte una intrusión. Esta notificación deberá decir en lenguaje natural cuando y donde se ha detectado dicha intrusión. Además, mostrará una prueba de la misma con una imagen. Respecto a la visualización de las alarmas, se define la creación de una interfaz web que muestre una lista con las alarmas relativas a cada usuario, así como el video asociado a las mismas.

3.2 Características del lado del cliente

En esta sección se detalla al completo el desarrollo del cliente, desde la recepción de los datos necesarios para general un mensaje de alarma hasta su transmisión por internet hasta el receptor. A lo largo de las distintas subsecciones, se explicará la manera en la que se ha desarrollado para que sea fácilmente integrable con el sistema de detección de alarmas, la creación de los diferentes mensajes de alarma, el cifrado y el envío de los datos por internet.

3.2.1 Características generales del cliente

El desarrollo de la parte del cliente de este sistema de comunicación de alarmas se ha realizado considerando la necesaria integración dentro de un sistema de detección de drones. Por ello, ha resultado conveniente que esta parte del sistema de comunicación se desarrolle de manera que pueda integrarse como un objeto. Así, la iniciación del sistema de comunicación se realiza al crear el objeto y cada acción se realiza como una sencilla llamada a una función. De esta manera, la complejidad de la comunicación no se transforma en una complejidad mayor del sistema donde se pretende integrar.

Los datos necesarios para la creación del cliente corresponden a dos aspectos fundamentales: dirección IP del servidor e identificación del cliente. El primer aspecto es debido a que el cliente debe comunicarse con un servidor ubicado en una dirección conocida, como se ha explicado de manera general para las arquitecturas cliente – servidor en el Capítulo 2, y el segundo es debido a que el protocolo SIA DC-09-2021 establece una identificación por tres valores de cada cliente. Estos son: número de receptor, prefijo de cliente y número de cliente. Con esta información, el sistema ya cuanta con todos los parámetros necesarios para la comunicación de mensajes de alarma con el servidor.

Para la comunicación por internet de manera segura es necesaria la creación de certificados con el fin de que servidor y cliente se autentiquen mutuamente. Por ello, el cliente contará con una clave pública y privada propias, así como con la clave pública de la autoridad certificadora que validará a todos los partícipes de la comunicación. De esta manera, podrá asegurarse de que se comunica con el servidor con el que pretende hacerlo y también podrá certificar su autenticidad al servidor.

Respecto a la creación y envío de los mensajes, previamente se ha detallado la forma de los mismos y se ha podido comprobar como todos tienen el mismo formato independientemente del mensaje que envíen. Por ello, la creación y envío de los mensajes, tanto de alarma como de mantenimiento, se hará reutilizando la mayor cantidad de código posible. De esta manera se reduce la cantidad de código, reduciendo la complejidad del mismo en su desarrollo y facilitando futuras ampliaciones. En la Figura 4 se muestra un esquema general del funcionamiento de esta parte del sistema.

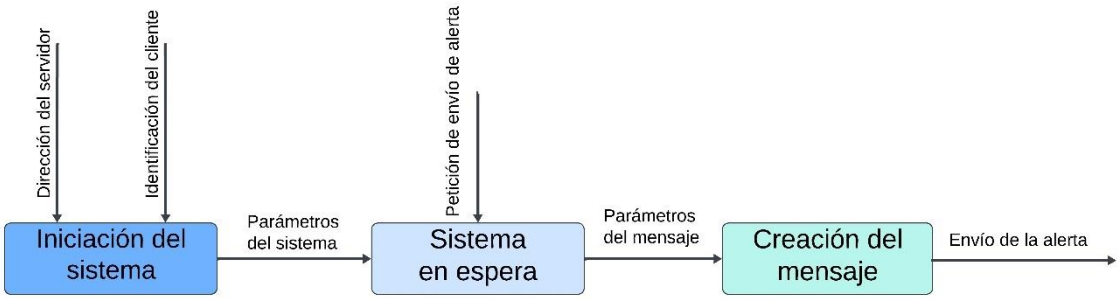

Figura 4. Diagrama del funcionamiento general del cliente.
Fuente: Elaboración propia

3.2.2 Desarrollo e implementación de la configuración inicial del cliente

La configuración inicial del cliente es el primer paso para establecer la comunicación entre cliente y servidor. En este paso, el cliente no va a llevar a cabo ninguna comunicación con el servidor. Sin embargo, sí va a gestionar toda la información relativa a las conexiones, con el objetivo de que se realicen de manera sencilla con una sola llamada a la función que desencadene el envío de los mensajes de alarma. En esta configuración inicial, será necesario aportar la dirección IP pública y el puerto del servidor al que se pretende enviar los mensajes. Gracias a la arquitectura cliente – servidor, no será necesario modificar este ajuste en ningún momento. Por otro lado, en la configuración inicial del cliente también se detalla la identificación del cliente y las configuraciones de mantenimiento y cifrado de los mensajes.

Esta configuración inicial se lleva a cabo al crear el objeto. Para crearlo, es necesario llamar a la función TransmisorSIA. Esta función acepta un número variable de parámetros, puesto que solo la dirección IP y el puerto de destino son esenciales para la comunicación, pero hay otros que también son de interés. En la Tabla 1 se muestran las diferentes maneras de crear el objeto.

Parámetros	Descripción
IPServidor, puertoServidor	Forma más sencilla de crear el objeto. La identificación del cliente se llenará de "0" y las configuraciones también.
IPServidor, puertoServidor, numeroCliente, prefijoCliente, numeroReceptor	Se configura la dirección del servidor y la identificación del cliente, esta se compone de 4, 3 y 2 números hexadecimales respectivamente.
IPServidor, puertoServidor, flagMantenimiento, flagCifrado	Se configura la dirección del servidor y la configuración del cliente. Cada uno de los parámetros es un valor booleano.
IPServidor, puertoServidor, numeroCliente, prefijoCliente, numeroReceptor, flagMantenimiento, flagCifrado	Se configuran todos los parámetros. Es la forma más compacta de configurar el sistema.

Tabla 1. Formas de iniciar el objeto TransmisorSIA.
Fuente: Elaboración propia

numeroCliente, prefijoCliente y numeroReceptor son los tres valores de identificación del cliente que se establecen en el protocolo SIA DC-09-2021. numeroCliente también es el identificador del cliente del protocolo Contact ID. flagMantenimiento establece si se deben enviar mensajes periódicos al receptor con el objetivo de que este compruebe que la conexión no se ha perdido. Dicha periodicidad es de veinte segundos. flagCifrado establece si los mensajes deben ir cifrados según lo establecido en el protocolo SIA DC-09-2021. En ambos casos, un valor de "1" representa la activación de ese ajuste.

Salvo la dirección IP y el puerto del servidor, el resto de parámetros se pueden configurar después de la creación del objeto, ya que no son estrictamente esenciales. De esta manera, resulta más sencillo para el ingeniero encargado de la integración realizar una configuración completa en tres pasos en vez de hacerlo todo en una misma llamada. Para ello, se hace uso de diferentes setters. En primer lugar, el setter setCifrado permite cambiar la configuración del cifrado de los mensajes. En segundo lugar, el setter setMantenimiento permite elegir si se envían mensajes de mantenimiento periódicos o no. Por otro lado, setCliente permite introducir los tres valores de

identificación del cliente. Por último, existe la posibilidad de cambiar la dirección del servidor con setServer en caso de ser necesario. La Tabla 2 muestra los distintos setters disponibles.

setter	Descripción
setCifrado(flagCifrado)	Permite configurar el cifrado de los mensajes.
setMantenimiento(flagMantenimiento)	Permite configurar el envío de mensajes de mantenimiento.
setCliente(numeroCliente, prefijoCliente, numeroReceptor)	Permite configurar el identificador del cliente.
setServer(IPServidor, puertoServidor)	Permite configurar la dirección del servidor.

Tabla 2. Setters del objeto TransmisorSIA.
Fuente: Elaboración propia.

3.2.3 Desarrollo e implementación de los diferentes tipos de mensaje

Para el desarrollo de los diferentes tipos de mensaje, es necesario considerar que la base de todos ellos es un mensaje bajo el formato Contact ID (o uno vacío en el caso de los mensajes de mantenimiento) envuelto sobre un mensaje con el formato SIA DC-09-2021. Por ello, la mejor forma de desarrollar el código es seguir esa misma estructura lógica. De esta manera, el mensaje se construirá y se enviará de la misma forma independientemente de su contenido y, por tanto, la complejidad del código disminuirá.

Siguiendo esta lógica, el primer paso es generar un mensaje bajo el formato Contact ID. Para ello, se define la función generar_mensaje_CID. Esta función recibe como parámetros de entrada todos los componentes de un mensaje que sigue dicho formato salvo el checksum, estos son: número de cuenta, tipo de mensaje, calificador de evento, código de evento, grupo de alarma y código de zona. Todos los parámetros son comunes en el desarrollo de este proyecto salvo código de evento, que depende de la alerta enviada como se detalla en el Capítulo 2. Una vez recibidos todos los parámetros, se define un mensaje previo que sigue la estructura de un mensaje Contact ID. Con este mensaje previo definido, se calcula el checksum como se define en la Ecuación 3.1, considerando al valor “0” como un “10”. Este checksum se añade como último valor del mensaje bajo el formato Contact ID.

$$s = \sum_{i=0}^{n}(valorCifra_i) \ \% \ 15 \qquad \textbf{(3.1)}$$

Una vez construido el mensaje Contact ID, o habiéndose saltado ese paso previo si se trata de un mensaje de mantenimiento, el siguiente paso es integrarlo en un mensaje SIA DC-09-2021. Debido a las diferentes posibilidades de dicho formato, la construcción del mensaje se vuelve más compleja y debe realizarse por pasos. La construcción de este mensaje se realiza con la función generar_mensaje_sia. Los parámetros de entrada de la misma son: flagCifrado, protocolo, numeroSecuencia, numeroReceptor, prefijoCuenta, numeroCuenta, mensajeContactID, xdata y timestamp. El parámetro flagCifrado indica la configuración de cifrado del mensaje. El parámetro protocolo indica si se trata de un mensaje vacío, en el caso de mensajes de mantenimiento, o de un mensaje de alarma bajo el protocolo Contact ID. El parámetro numeroSecuencia indica el número de secuencia que lleva asociado el mensaje a enviar, este se incrementará en uno cuando se envíe el mensaje hasta 9999 donde volverá a 0001. Es cero en el caso de mensajes de mantenimiento. Los parámetros numeroReceptor, prefijoCuenta y numeroCuenta son los valores que identifican a un cliente, detallados anteriormente. El parámetro mensajeContactID es el mensaje según el protocolo Contact ID que se ha detallado en el párrafo anterior. El parámetro xdata es el que recoge los datos extendidos donde se incluyen las coordenadas del avistamiento o los datos del video de la detección.

Una vez considerados los parámetros de entrada, queda la construcción del mensaje bajo el formato SIA DC-09-2021 en sí mismo. Para ello, el primer paso es establecer un mensaje previo con los elementos comunes a todos los mensajes independientemente de si están cifrados o no. Por tanto, se crea un primer mensaje previo con la estructura <"id"><seq><Rrcvr><Lpref><#acct> donde el significado de cada elemento es el indicado en el Capítulo 2. Posteriormente, si el mensaje no debe ir cifrado se añaden el mensaje Contact ID, los datos extendidos y la fecha. Sin embargo, si el mensaje debe ir cifrado se añade un padding antes del mensaje Contact ID de un número de caracteres calculado según la Ecuación 3.2. Este padding sirve para que el cifrado actúe sobre un número de caracteres múltiplo de 16. Una vez insertado el padding al mensaje, este se cifra empleando cifrado AES, usando claves que debe conocer también el receptor.

$$\boldsymbol{padding} = \left(\mathbf{16} - (\boldsymbol{longitudMensajeCodificar}\ \%\ \mathbf{16})\right)\ \%\ \mathbf{16} \qquad \textbf{(3.2)}$$

Una vez elaborado todo le mensaje según el protocolo SIA DC-09-2021 salvo el crc y la longitud, es necesario incorporar ambos campos al mensaje. En primer lugar, se calcula la longitud del mensaje ya generado, es decir, que ninguno de los dos campos mencionados se tiene en cuenta para este cálculo. Después, se calcula el CRC usando el polinomio CRC-16, este valor lo empleará el receptor para asegurar la integridad del mensaje recibido. Una vez calculados ambos valores, estos se incorporan al mensaje, creando el mensaje completo con la estructura <LF><crc><0LLL><"id"><seq><Rrcvr><Lpref><#acct>[<pad>|...data...][x…data…]<timestamp><CR> como se ha detallado en el Capítulo 2. En la Figura 5 se representa la manera en la que se crean los mensajes de alarma.

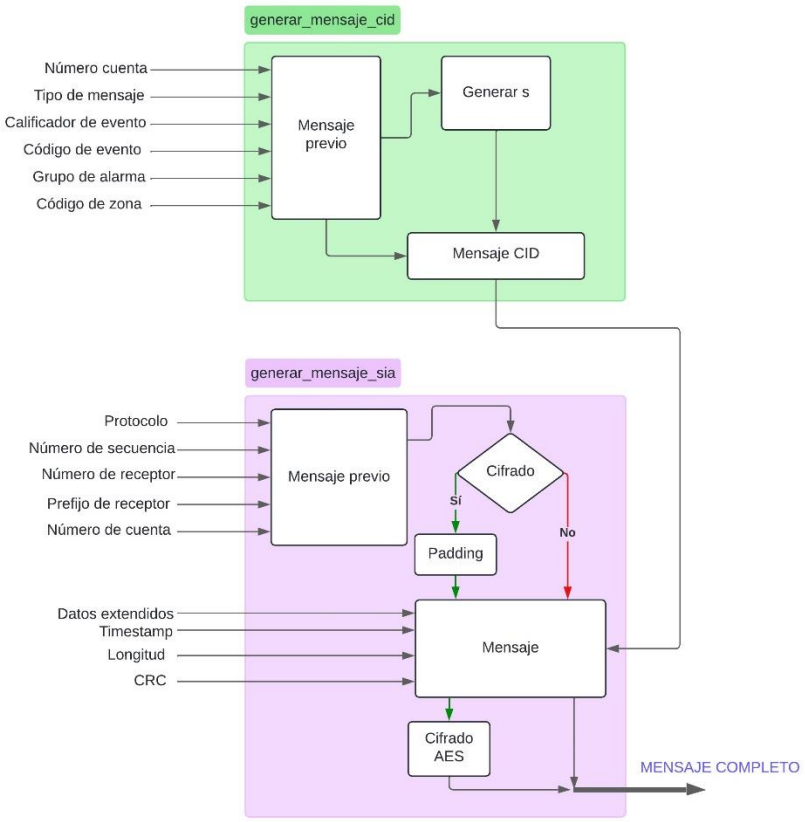

Figura 5. Diagrama de flujo de la creación de los mensajes SIA DC-09-2021.
Fuente: Elaboración propia

Para facilitar la integración de las funciones relativas al envío de los mensajes en el código de detección de drones, se ha optado por definir una función para cada tipo de mensaje. De esta manera, la integración se lleva a cabo insertando una intuitiva llamada a una función según cual sea el objetivo de la alerta. Así, las distintas funciones de alerta disponibles se detallan en la Tabla 3.

Función	Descripción
enviarAlarma(videoPath, x, y, z)	Función que permite el envío de mensajes que notifican de una intrusión. Tanto incluir video como incluir coordenadas es opcional.
enviarNull()	Función que permite en envío de mensajes de mantenimiento. Se puede integrar manualmente, pero generalmente será el propio sistema el que los envíe periódicamente si se configura.
enviarInicioSistema()	Función que permite el envío de un mensaje inicial por parte del sistema.
enviarCaidaSistema()	Función que permite el envío de un mensaje de error del sistema.

Tabla 3. Funciones de alerta del sistema.

Detallando los elementos mostrados en la Tabla 3, la primera función llamada enviarAlarma es la que se empleará de manera más habitual. Es la función empleada para enviar los mensajes de alarma, independientemente de si la alerta debe incluir video adjunto o no. Para ello, esta función define primeramente el protocolo, Contact ID, y un parámetro que define el cifrado o no del mensaje según se haya definido en la configuración. Después, establece los valores necesarios para la construcción del mensaje Contact ID. El único que varía según si es un mensaje de alarma o de inicio y caída del sistema es el código de evento, el cual es 131 en este caso. Por tanto, se definirán los valores propios de un mensaje Contact ID tal y como se han detallado anteriormente. También se llamará a la función iniciarSecuencia como ya se ha detallado y se obtendrá la fecha y hora actuales en formato epoch con la función time.time(). A partir de aquí, el comportamiento de esta función difiere si se han introducido parámetros de video o de coordenadas. Si se pretenden enviar coordenadas, se crearán datos extendidos bajo la estructura X{x}Y{y}Z{z}. Después, se enviará el mensaje de alarma como se detalla en la Sección 3.2.4. Si se pretende enviar un archivo de video, se llamará a la función privada __enviarVideo, que se detalla en la Subsección 3.2.5, después de enviar el mensaje de alarma y esta mandará otro mensaje de alarma idéntico al primero, pero con otros datos extendidos que proporcionarán datos relativos al video.

Siguiendo con las funciones detalladas en la Tabla 3, las funciones enviarInicioSistema y enviarCaidaSistema tienen comportamientos muy similares. Ambas siguen el mismo procedimiento que enviarAlarma con la diferencia de que no incluyen datos extendidos en ningún caso ni se envía video. Además, en código de evento es distinto al que emplea dicha función. En el caso de enviarInicioSistema, el código de evento es 133 y en el caso de enviarCaidaSistema el código de evento es 300.

Por último, la función enviarNull envía un mensaje de mantenimiento. Este se envía si se llama a la función como se indica en la Tabla 3 o si se configura como se ha detallado anteriormente. El mensaje de mantenimiento utiliza el protocolo SIA DC-09-2021, pero es un mensaje vacío y no usa el protocolo Contact ID ni datos extendidos. Por tanto, el campo dedicado al mensaje Contact ID estará vacío y, además, en ningún caso estará cifrado y el número de secuencia será 0. Sin embargo, sí obtendrá la fecha y hora actuales en formato epoch con la función time.time() para

enviarla en el mensaje, puesto que es un requerimiento del protocolo SIA DC-09-2021. La Tabla 4 representa las características propias de cada mensaje.

Función	Código de protocolo	Código de evento	Tipos de datos extendidos	Funciones extra
enviarAlarma	ADM-CID	131	X,Y,Z	Puede llamar a enviarVideo.
enviarVideo	ADM-CID	131	V	No la llama el usuario directamente.
enviarNull	NULL	-	-	-
enviarInicioSistema	ADM-CID	133	-	-
enviarCaidaSistema	ADM-CID	300	-	-

Tabla 4. Diferencias entre funciones de alerta.

3.2.4 Desarrollo e implementación del envío de mensajes de alarma

Una vez generado el mensaje que se pretende hacer llegar al servidor, es necesario desarrollar todo el procedimiento a través del cual el mensaje generado se envía mediante sockets UDP protegido con el protocolo DTLS. Para ello, será necesario obtener certificados TLS/SSL, establecer la conexión segura y enviar los mensajes.

En primer lugar, para la creación de los certificados TLS/SSL se hace uso de la herramienta OpenSSL, detallada en el Capítulo 2. Gracias a ella, es posible generar una entidad certificadora que valide a los partícipes de la comunicación, en este caso, cliente y servidor. Para crear dicha entidad certificadora, se genera una clave privada como un archivo llamado ca.key y, posteriormente, una clave pública como un archivo llamado ca.crt.

Una vez creada la entidad certificadora, esta se emplea para crear los certificados del cliente siguiendo unos pasos muy similares. Primero se crea la clave privada como un archivo llamado client.key y posteriormente una clave pública como un archivo llamado client.csr. Finalmente, para que la entidad certificadora valide al cliente, se emplean las claves pública y privada de dicha entidad para validar la clave pública del cliente. Con esto, cuando el servidor busque validar el certificado del cliente, sabrá que está validado por una autoridad de confianza. Es importante considerar que el certificado del cliente no está asociado a una IP en concreto, por lo que el certificado será válido independientemente de la red a la que esté conectado. Esto no será así con el servidor, el cual debe ser una dirección conocida. Después de obtener los certificados del cliente, se crea un contexto SSL con el que se introduce tanto la clave pública de la entidad certificadora como las claves púbica y privada del cliente. Esta es la información que utilizará para establecer la conexión segura entre el cliente y el servidor.

Posteriormente, se crea el socket UDP y se envuelve con DTLS. Con este socket protegido, se inicia el handshake por parte del cliente que tratará de establecer la conexión inicial tres veces antes de dar error. Si se completa correctamente, el siguiente paso es enviar el mensaje generado según el protocolo SIA DC-09-2021.

Después de enviar el mensaje, el cliente espera un mensaje de confirmación. Si se ha recibido correctamente, se recibirá un ACK y habrá acabado la comunicación del mensaje de alarma. Si se ha recibido con algún error, se recibirá un NACK y el cliente volverá a crear y enviar el mensaje de alarma con la fecha actualizada hasta un total de tres intentos. Si no se recibe respuesta en cinco segundos, se volverá a enviar el mismo mensaje hasta un total de tres intentos. Si el receptor no ha sido capaz de procesar el mensaje o de entender que se trata de un mensaje de alarma, se recibirá un DUH y se dará por fallido el intento de comunicación. La Figura 6 representa la manera en la que el cliente gestiona el envío de mensajes.

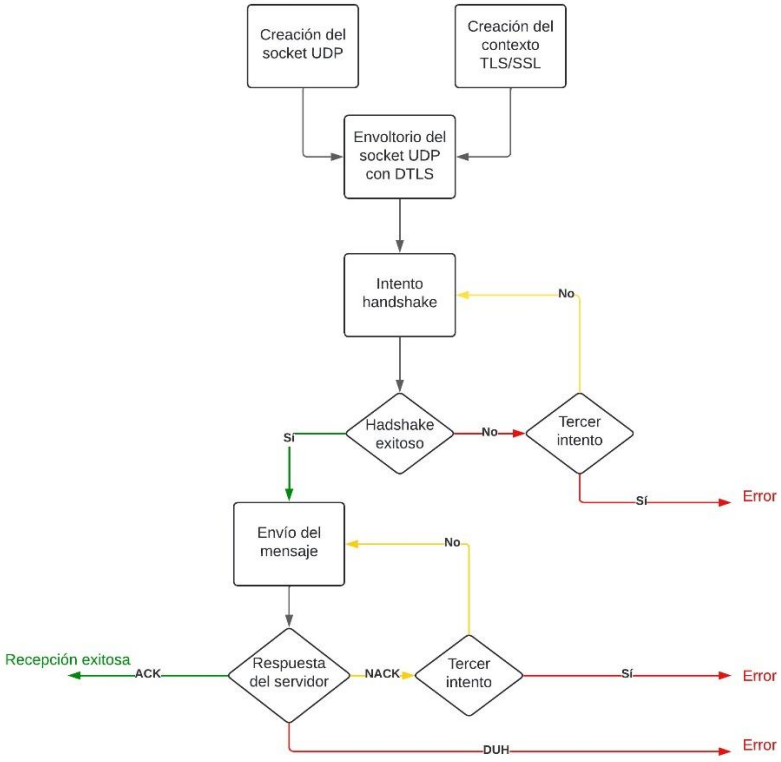

Figura 6. Diagrama de flujo del envío de mensajes mediante UDP.
Fuente: Elaboración propia.

3.2.5 Desarrollo e implementación del envío de archivos de video

En el caso de los mensajes de detección de una intrusión, la llamada a la función de envío de la alarma puede recibir un video en formato .mp4 como parámetro de entrada. Si esto sucede, el sistema enviará dos mensajes de alarma y un video. El primero será un mensaje que notificará la intrusión que podrá llevar, o no, las coordenadas de la misma como datos extendidos. El segundo será el mismo mensaje de alarma, pero con otros datos extendidos. En este caso, los datos extendidos llevarán los metadatos fundamentales del video y seguirán el siguiente formato: VF{fps video}W{ancho video}H{altura video}.

Por tanto, cuando la función enviarAlarma tenga como parámetro de entrada un video, esta llamará a otra función llamada __enviarVideo después de enviar el mensaje de alarma con las coordenadas. En ella, primero se extraerán los datos de fotogramas por segundo y resolución del video empleando la librería cv2. Después, se mandará un mensaje de alarma con estos datos como se ha mencionado en el párrafo anterior y, finalmente, se mandará el video.

El envío del video no puede realizarse con un solo paquete UDP debido a que la longitud de los mismos está limitada a 1500 bytes. Por tanto, se emplea la función imencode de la librería cv2 para separar el video en imágenes JPG. La calidad de las mismas se deja al 75% puesto que es una calidad suficiente que hace que los videos reconstruidos no pierdan calidad aparentemente si las resoluciones están por debajo de HD. Cada frame se comprime usando la librería zlib. Se usa esta librería porque evita la corrupción del video si se pierde algún paquete. Si se usa base64, los videos son altamente susceptibles a corromperse más de lo deseado. Una vez codificado cada frame, este se envía por paquetes de longitud igual a 1024 bytes. Después de cada frame se manda un mensaje de ‘END’ y al enviar el video completo otro idéntico. De esta manera, el receptor puede reconstruir el video sin problemas. Finalmente, el cliente espera un mensaje de ACK como

respuesta al último 'END' o lo vuelve a enviar hasta recibirlo. La Figura 7 muestra la forma en la que los videos son tratados para su transmisión por parte del cliente.

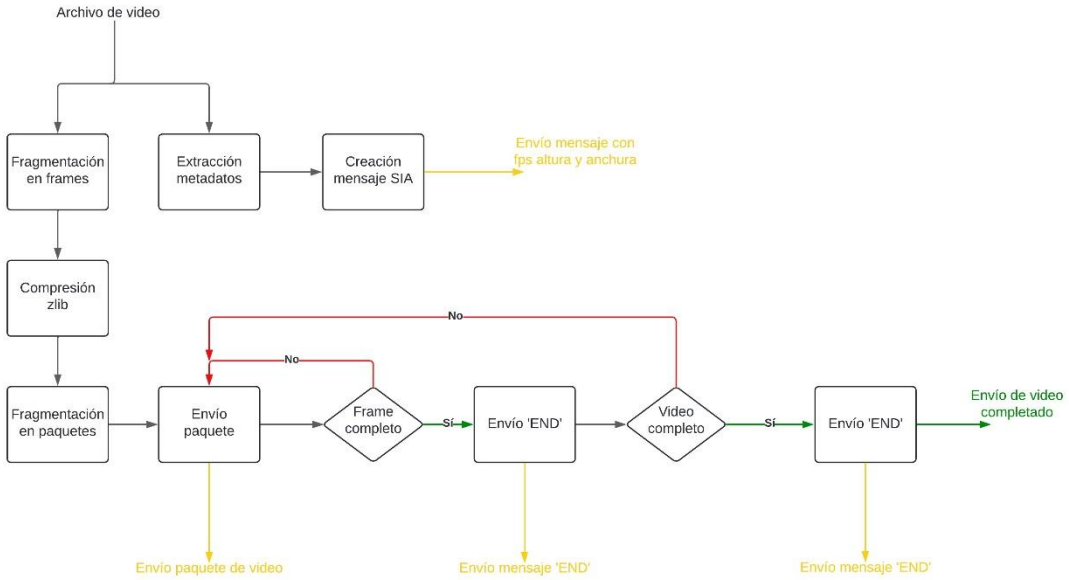

Figura 7. Diagrama de flujo del envío de archivos de video.
Fuente: Elaboración propia.

3.3 Características del lado del servidor

En esta sección se detalla el desarrollo del servidor receptor de mensajes de alarma desde la recepción de los mensajes de alerta y videos asociados a ellos hasta su almacenamiento en una base de datos para su notificación al usuario final. Se describe como se aprovecha el formato de los mensajes SIA DC-09-2021 para extraer la información de cada alerta y como se reconstruyen los videos recibidos por fragmentos. Además, se detalla la manera en la que se almacenan las alertas en una base de datos para su posterior visualización por parte del usuario final.

3.3.1 Características generales del servidor

El desarrollo del lado del servidor receptor de alarmas se ha realizado considerando que este debe funcionar como un sistema autónomo al cliente. Este trabajará en un equipo dedicado y su dirección IP y puerto serán conocidos por los clientes. Además, siempre serán estos últimos los que realizarán la conexión inicial. Por otro lado, el servidor debe actuar como un sistema completo en sí mismo y funcionar permanentemente a la espera de conexiones.

Los únicos datos necesarios para la configuración del servidor son su dirección IP y su puerto. A diferencia del cliente, el servidor no debe conocer a los clientes para que la comunicación sea exitosa, serán los propios mensajes de alarma los que proporcionarán al servidor la identificación del cliente que se comunica con él.

Para la verificación mutua entre clientes y servidor, este debe disponer, al igual que los clientes, de las claves públicas y privadas expedidas por la autoridad certificadora, así como la clave pública de dicha autoridad que permitirá verificar la confianza en los certificados de los clientes. De esta manera, el servidor aceptará conexiones de clientes legítimos y rechazará las conexiones de todos los que no tengan certificados firmados por la autoridad certificadora.

Respecto a la recepción de mensajes de alarma, el servidor aprovecha el formato fijo y conocido de los mensajes SIA DC-09-2021 para obtener la información contenida en ellos. Así mismo, verificará la fecha, el CRC y la longitud para asegurar la integridad de los mensajes recibidos. Respecto a los videos, el servidor reconstruirá los fragmentos enviados por el cliente y guardará los videos recibidos en formato mp4. Finalmente, guardará las alertas y los videos para notificarlos y mostrarlos al usuario final. En la Figura 8 se muestra el funcionamiento general del servidor.

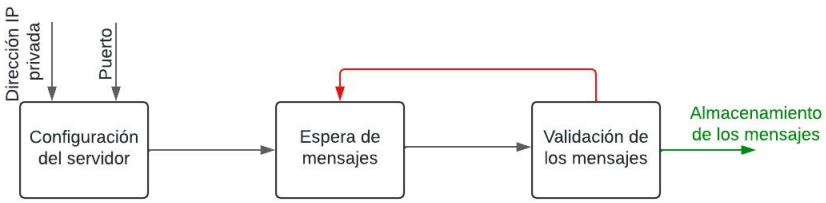

Figura 8. Esquema general del servidor.
Fuente: Elaboración propia

3.3.2 Desarrollo e implementación de la configuración inicial del servidor

La configuración inicial del servidor es mucho más sencilla que la del cliente. Esto es así debido a que todos los parámetros relativos a la identificación del cliente, así como los relativos a la configuración del cifrado y los mensajes de mantenimiento se realizan en el lado del cliente. De esta manera, solo será necesario configurar la dirección IP privada y el número de puerto al que está asignado el servidor.

Sin embargo, sí es necesario configurar la red a la cual se conecta el servidor de forma que el router permita recibir conexiones en un equipo en particular desde el exterior de la red. Para ello, es necesario entrar a la configuración del router a la cual se accede generalmente al conectarse al mismo en la dirección '192.168.0.1' desde cualquier navegador. Posteriormente, se busca la configuración llamada 'Redirección de puertos' y se añade una asignación de puertos. Se debe indicar la dirección IP privada del terminal que hace de servidor, el protocolo empleado para las comunicaciones, que en este caso será UDP, y los puertos, privado y público, empleados. Estos últimos pueden ser el mismo y resulta más práctico que sea así. En la Figura 9 se muestra la configuración del router empleada en este proyecto.

Figura 9. Configuración de la redirección de puertos.
Fuente: Elaboración propia.

Por otro lado, es necesario iniciar la base de datos en la que se guardarán los mensajes de alarma. Para iniciarla, así como para manejarla, se utiliza la librería sqlite3. Esta es una librería que permite un manejo sencillo de SQL para realizar acciones de poca complejidad. En este caso, se inicia la base de datos llamada alarmas.db y se crea si esta no existe con los parámetros que se definirán en una subsección siguiente.

3.3.3 Desarrollo e implementación de la recepción de mensajes de alarma

Para la recepción de los mensajes de alarma, es necesario considerar que el servidor debe escuchar permanentemente a la espera de que un cliente establezca una conexión para llevar a cabo la transmisión de un mensaje. Para ello, el primer paso es la creación del contexto DTLS. Al crear este contexto, se carga la clave pública y la clave privada del servidor, así como la clave pública de la autoridad certificadora que se utilizará para validar a los clientes. Además, se establece que deberá rechazar la conexión y generar un error si el certificado no existe o no es válido.

Posteriormente, el servidor queda a la escucha de recibir un primer mensaje e intentar el handshake con aquel cliente que pretenda enviar un mensaje de alarma. Si el handshake es exitoso, el servidor quedará vinculado temporalmente al cliente que pretende enviar una alerta. De esta manera, se establecerá una conexión virtual. Esta conexión no viene determinada por el protocolo UDP, el cual no establece estas conexiones; sino por DTLS. Este protocolo hace que, aunque los mensajes se vayan a enviar como mensajes UDP en los que no hay confirmación de envío ni un circuito virtual en sentido estricto, el servidor no aceptará mensajes de otro cliente hasta que se cierre la comunicación cifrada por medio de DTLS.

Dicha conexión solo estará establecida el tiempo que dure la transmisión del mensaje de alarma, independientemente de si el mensaje se recibe correctamente o no. En el caso de los mensajes de alarma que notifican la posterior recepción de un video, esta conexión DTLS perdura hasta la recepción del video. Posteriormente se cierra. En la Figura 10 se muestra la manera en la que el servidor gestiona la recepción de los mensajes de alarma.

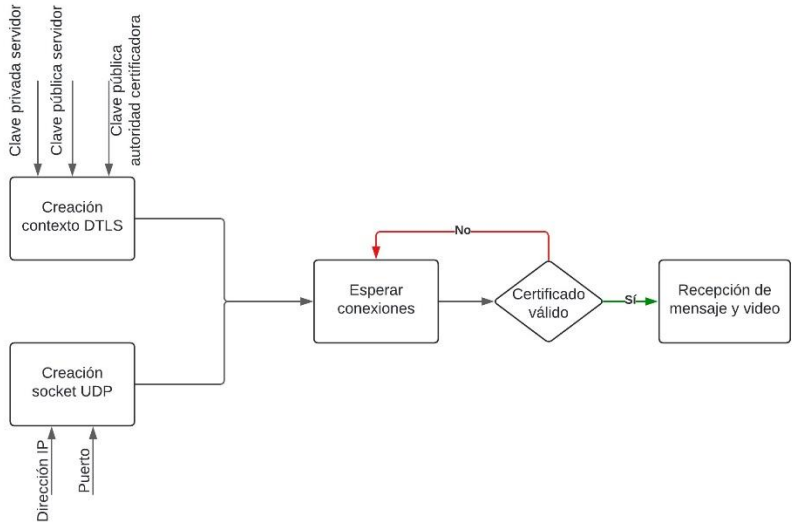

Figura 10. Esquema de la recepción de mensajes por parte del servidor.
Fuente: Elaboración propia.

3.3.4 Desarrollo e implementación de la extracción de datos y validación de mensajes de alarma

Una vez recibido un mensaje de alarma por parte del cliente, es necesario extraer toda la información que este contiene y verificar su integridad. Para ello, se aprovechará el hecho de que el formato del mensaje es conocido y se extraerán los datos del mensaje. Posteriormente, se verificará que el CRC y el checksum del mensaje son correctos y que la fecha del mismo está dentro del margen que acepta el protocolo SIA DC-09-2021.

En primer lugar, se busca el primer carácter del mensaje que es un salto de línea. El CRC serán los cuatro dígitos siguientes. Después, se busca el siguiente carácter "0" ya que este indica el inicio del campo de longitud del mensaje. El campo longitud del mensaje son las tres siguientes cifras. Posteriormente, el texto entre comillas contiene el protocolo utilizado. Este campo contendrá, o no, un asterisco que identifica si va a haber cifrado. El número de secuencia son las cuatro siguientes cifras después de las comillas. El número de receptor estará entre la siguiente R y la siguiente L. El prefijo de cuenta está entre dicha L y la siguiente almohadilla. El número de cuenta se encuentra entre la siguiente almohadilla y el siguiente corchete. A partir de aquí, los datos que se extraen y la manera en la que esto sucede depende del tipo de mensaje que se transmita y de si está cifrado o no.

En el caso de los mensajes de mantenimiento, identificados con el protocolo "NULL", solo se extraerá la fecha en formato epoch. Esta se encuentra entre el carácter "_" y el final del mensaje. En el caso de los mensajes de alarma, identificados con el protocolo "ADM-CID", dependerá de si el mensaje está cifrado o no. En el caso de mensajes cifrados, que se habrán identificado por el asterisco, estos se desencriptarán de manera sencilla al conocer el cifrado empleado (AES) y las claves empleadas para ello. Estas son fijas y conocidas de antemano, no se transmiten en ningún momento. Una vez desencriptado, los siguientes pasos son comunes para todos los mensajes que usan el protocolo ADM-CID. El mensaje Contact ID se encuentra entre el carácter "|" y el cierre del corchete. Los datos extendidos se encuentran entre los siguientes corchetes. Finalmente, la fecha en formato epoch se encuentra entre el carácter "_" y el final del mensaje.

Para extraer la información de los datos extendidos, que contiene coordenadas de la detección que se alerta o información de video, se buscan los prefijos de los datos extendidos entre los que se conocen como posibles. Para distinguir los datos que contiene el campo de datos extendidos, se utiliza el primer carácter. Si este es una "V" se sabe que es información de video y, además, que después de este mensaje se recibirá un video. Si el primer carácter de los datos extendidos es una "X", se sabe que se reciben coordenadas.

En el caso de las coordenadas, se sabe que la estructura es X{x}Y{y}Z{z}. La coordenada X estará entre la "X" y la "Y", la coordenada Y estará entre la "Y" y la "Z" y la coordenada Z estará entre la "Z" y el final del mensaje. En el caso de la información de video, se sabe que esta sigue la estructura VF{fps video}W{ancho video}H{altura video}, por lo que se extraen los fotogramas por segundo del video como la información entre la "F" y la "W", el ancho del video como la información entre la "W" y la "H" y la altura del video como la información entre la "W" y el final.

Una vez extraídos todos los datos, el servidor recrea el mensaje a partir de los datos extraídos y vuelve a calcular tanto el CRC como la longitud del mensaje de la misma manera que lo hace el cliente. Así, compara los valores recibidos con los generados por él mismo y considera correcto el mensaje si coinciden. Por otro lado, también compara la fecha actual con la recibida y acepta como válido el mensaje si la diferencia es menor a veinte segundos. Esto lo define el protocolo SIA DC-09-2021.

Si el mensaje se considera correcto, el servidor envía un ACK con la estructura <LF><crc><0LLL><"ACK"><seq><Rrcvr><Lpref><#acct>[]<CR> donde el número de secuencia y todos los identificadores de la cuenta del cliente son los recibidos en el mensaje al que se le da respuesta. El CRC y la longitud se calculan de la misma manera que en los mensajes

del cliente. Por el contrario, si alguna comprobación falla, se manda un NACK con la estructura <LF><crc><0LLL><"NACK"><0000><R0><L0><#0>[]<timestamp><CR> donde las diferencias con el mensaje de ACK se encuentran en que este último manda la fecha actualizada en formato epoch y deja tanto la identificación del cliente como el número de secuencia relleno de valores "0". En la Figura 11 se representa el proceso de extracción y validación de datos a partir del mensaje recibido.

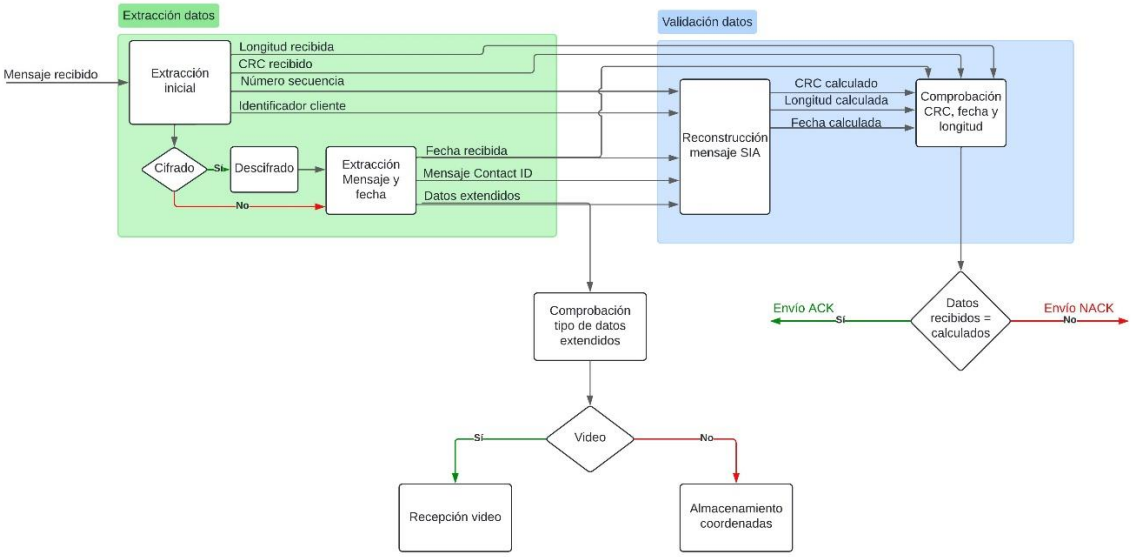

Figura 11. Diagrama de flujo de la extracción y validación de datos de un mensaje.
Fuente: Elaboración propia.

3.3.5 Desarrollo e implementación de la recepción de archivos de video

Como se ha mencionado anteriormente, si se recibe un mensaje con el prefijo de datos extendidos "V", el servidor espera recibir un archivo de video con la intrusión que detecte el cliente. Para realizar la recepción del video y reconstruirlo, el servidor debe obtener los paquetes que conforman cada frame, unirlos y repetir este proceso hasta que se haya recibido el video completo. Posteriormente, decodificarlo y guardarlo como un archivo mp4.

En primer lugar, se utiliza la función VideoWriter de la librería cv2 para generar objeto donde se va a escribir el video recibido. El códec empleado es mp4v. A este objeto se le pasan como parámetros de entrada el códec, la resolución y el nombre que se le da al video. El nombre de cada video tiene la estructura "video_{numCuenta}_{prefCuenta}_{numReceptor}__{año}-{mes}-{día}_{hora}-{min}-{seg}". Una vez creado el objeto, se espera recibir un mensaje UDP y se comprueba si este es el que indica el final de un frame con el mensaje 'END'. Si no es el final del frame, se reciben más paquetes hasta completarlo. Una vez completo el frame, se descomprime usando la librería zlib, ya que se ha comprimido con la misma librería, y se escribe en el archivo de video. Este proceso se repite hasta recibir el último frame, que se detecta al recibir dos mensajes de terminación de frame seguidos. En la Figura 12 se representa el proceso que siguen los paquetes recibidos hasta guardar el video completo.

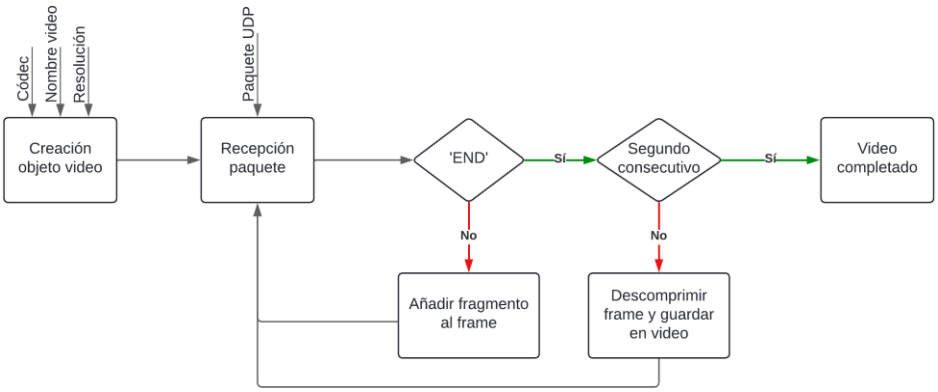

Figura 12. Diagrama de flujo de la recepción de archivos de video.
Fuente: Elaboración propia.

3.3.6 Almacenamiento de los mensajes recibidos

Para almacenar las alertas recibidas, hay que considerar que puede recibirse tanto un mensaje con coordenadas, como un mensaje de video seguido del video al que hace referencia. Por ello, es necesario idear un sistema que contemple enlazar un mensaje de alarma con un video, así como que contemple las coordenadas recibidas como una información importante. Para ello, se hace uso de la librería sqlite3 y se guardan las alertas en una base de datos con la información relevante.

Para guardar la alerta recibida, se crea una fila en la base de datos iniciada en la configuración inicial del servidor. En ella, se guardan los siguientes campos por orden de aparición: receptor, prefijo, cuenta, protocolo, fecha, hora, video, x, y, z. La fecha y la hora se obtienen a partir de la fecha en formato epoch recibida en el mensaje. Así, no solo se obtiene una fecha entendible para el usuario final, sino que también se podrá ordenar posteriormente la visualización de alarmas por orden de llegada. En el campo 'video' no se guarda el video en sí mismo, sino su nombre. De esta manera, el campo es un simple texto que enlazará al video para su descarga por parte del usuario final.

3.4 Desarrollo del entorno de notificación y visualización de alarmas

En esta sección se detalla el desarrollo del entorno de notificación y visualización de alarmas. En ella, se describe la manera en la que la notificación de las intrusiones se realiza al usuario final por medio de un mensaje de texto vía WhatsApp acompañado de un frame del video de la intrusión. Además, se explica la creación de una interfaz web donde usuarios finales y administradores pueden ver las alertas generadas y descargarse los videos de las intrusiones.

3.4.1 Notificación de las intrusiones detectadas vía WhatsApp

La notificación de las intrusiones se realiza por medio de la aplicación de mensajería instantánea WhatsApp. Esto permite que el usuario final reciba de manera sencilla en su teléfono un breve mensaje con la fecha y lugar de la intrusión, así como un fotograma de la misma. El mensaje se envía desde el servidor cada vez que recibe un mensaje con video por medio de la librería pywhatkit.

En primer lugar, cuando el servidor acaba de manejar la recepción del video, este extrae el frame central del video y lo guarda como imagen. Después, crea el mensaje "Intrusión detectada en el receptor {numeroReceptor} de la casa con número de cuenta {prefijoCliente}_{numeroCliente} en la fecha {fecha}. Para ver el video completo entre en la aplicación de seguridad.". Con la

imagen y el texto creados, utiliza la función sendwhats_image de la librería pywhatkit para mandar el mensaje con la imagen a un número de teléfono dado.

Esta manera de notificar la intrusión utilizando WhatsApp no es la óptima, ya que utiliza el navegador, abre la versión web de WhatsApp y envía el mensaje controlando el propio ordenador. Esto puede generar errores y retardos. Sin embargo, es una alternativa gratuita frente a servicios profesionales que tienen un coste por mensaje como Twilio.

3.4.2 Creación de la interfaz web para la visualización de intrusiones

La interfaz web para la visualización de intrusiones es un sistema aparte del receptor. Este no necesita del receptor para funcionar en ningún caso y su único contacto con el mismo es la base de datos que crea el servidor y los videos de intrusión. Por tanto, esta interfaz podría ser alojada por cualquier servidor siempre que disponga de conexión remota con la base de datos y los videos de las intrusiones. La interfaz web permite la creación de usuarios con distintos privilegios para ver las alarmas de la base de datos parcial o totalmente.

En primer lugar, se crea una aplicación Flask por medio de Python que permite la creación de modelos y rutas. Los modelos que se crean son dos. Un primer modelo llamado ‘Alarmas’ que hace referencia a la tabla de alarmas creadas por el receptor y un segundo modelo llamado ‘Usuario’ que almacena otra tabla con usuario, contraseña y rol. A esta segunda tabla solo se accede desde la interfaz web, ya que en dicha interfaz es donde se definen los usuarios que pueden acceder a las alarmas.

Después, se definen las rutas. La primera de las rutas a mencionar es ‘/registro’. En ella, se crea una vista con tres campos a rellenar los cuales son usuario, contraseña y un campo que permite elegir el rol de dicho usuario, estos son ‘admin’ y ‘cliente’. Gracias a este último campo, un usuario ‘admin’ podrá ver todas las alarmas de la tabla y un usuario ‘cliente’ solo aquellas cuyo numero de cliente coincida con su nombre de usuario. Cabe destacar que la contraseña se almacena como un hash, no como la propia contraseña, usando la función generate_password_hash. Esto es así por seguridad. La siguiente ruta a considerar es ‘/login’ que no es más que un formulario donde se introduce el usuario y la contraseña. Si el usuario existe y el hash de la contraseña introducida coincide con el hash de la contraseña guardad, se accede a la ruta por defecto ‘/’. A esta ruta solo se puede acceder si se ha realizado el login. Si el usuario es admin, hace una query a la base de datos mostrando todas las alarmas por orden de llegada descendente, es decir, primero las más recientes. Si el usuario es ‘cliente’ filtra aquellas alarmas con número de cuenta igual al nombre de usuario que está en la sesión.

Para cada ruta, se ha realizado una página web usando HTML y CSS. En las rutas de registro e inicio de sesión, la programación de las páginas web es muy sencilla, destinada simplemente a mostrar por pantalla el contenido. Sin embargo, en el caso de la página donde se muestran las alarmas generadas, se sustituye el campo que recoge el nombre del video asociado a las detecciones por “Descargar video”. Se inserta un enlace con la etiqueta href al nombre del video y se fuerza la descarga de un fichero con ese mismo nombre al pulsar sobre él. De esta manera, aunque en la base de datos solo se recoge el nombre del fichero, se permite la descarga del archivo con ese mismo nombre alojado en la máquina que alberga el servidor web.

Capítulo 4. Desarrollo y resultados del trabajo

En este capítulo se muestran los resultados del trabajo. A lo largo del mismo, se mostrará el funcionamiento de cada una de las partes que componen el sistema desarrollado en este Trabajo Final de Grado. Además, detallarán las pruebas realizadas para comprobar el adecuado funcionamiento de todas las partes y la interacción entre ellas. Es necesario considerar que las pruebas representadas en este documento son solo una representación de las mismas. A lo largo del desarrollo del proyecto, se han llevado a cabo pruebas cada vez que se incluía una nueva función.

4.1 Desarrollo y resultados del cliente

Para comprobar el funcionamiento del cliente, se crea un código en Python llamado testClase.py que hará, a modo de prueba, las funciones del sistema de detección de drones donde se pretende integrar el cliente. De esta manera, se pueden hacer comprobaciones de manera sencilla sin las limitaciones de emplear el sistema de detección completo. Además, utilizar un sistema de pruebas permite mandar mensajes y cambiar configuraciones de manera más rápida, forzando al sistema a trabajar bajo condiciones más exigentes que en su implementación final.

Para poder comprobar todas las funciones del cliente y su adecuada integración con el servidor, el código de prueba ejecuta todas las funciones posibles y, además, modifica las configuraciones. De esta manera, se verifica el funcionamiento integral del cliente y sus resultados pueden servir para verificar posteriormente el funcionamiento de la interfaz de visualización.

La función de pruebas crea un primer objeto transmisor con unos datos de cliente arbitrarios, sin cifrar los mensajes ni enviar mensajes de mantenimiento. Posteriormente, envía un mensaje de inicio de sistema. Después, manda tres mensajes de alarma. Cada uno con una combinación posible de datos extendidos. En los mensajes que implicarán el envío de un mensaje de texto por parte del servidor, se establece una espera que se explicará en una sección siguiente. Sigue enviando un mensaje de caída de sistema. Posteriormente, activa el cifrado, envía otro mensaje de alarma y desactiva el cifrado. Luego, manda tres mensajes de alarma modificando la identificación del cliente parcialmente antes de enviar cada uno de ellos. Primero el número de receptor, luego el prefijo de cuenta y, finalmente, el número de cuenta. Finalmente, activa el envío de mensajes de mantenimiento y espera un minuto para comprobarlos. En la Figura 13 se muestra un diagrama de flujo de estas pruebas.

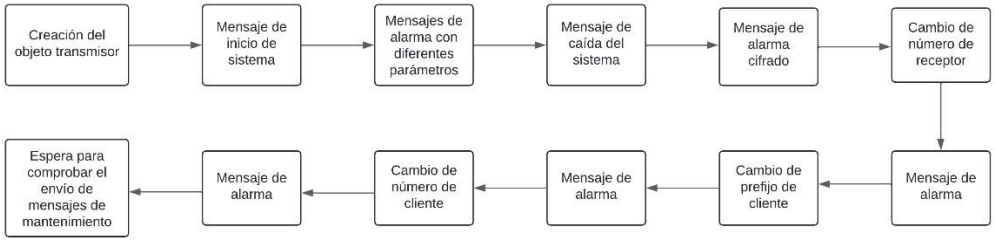

Figura 13. Diagrama de flujo que representa las pruebas del cliente.
Fuente: Elaboración propia.

Dichas pruebas se efectúan desde la terminal del ordenador. En las Figuras 14, 15 y 16, se visualizan los mensajes que aparecen a medida que se efectúan dichas pruebas. En ellas, se aprecia como el segundo mensaje, que se envía desde un socket recién creado sin ningún certificado, no recibe respuesta. Además, cuando se transmite un video se envía un primer mensaje con coordenadas y un segundo mensaje con datos del video a transmitir. Posteriormente, se envía el video. Además, el mensaje número siete va cifrado y parte del texto enviado no es legible sin descifrarlo.

```
PS C:\Users\vicen\OneDrive - UPV\Escritorio\Carpetas\Cuatri_10\Móviles\Proyecto_Alarmas> & C:/Users/vicen/AppData/
Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/vicen/OneDrive - UPV/Escritorio/Carpetas/Cuatri_10/Móviles/Pr
oyecto_Alarmas/testClase.py"
Enviando mensaje 1/10
Mensaje enviado:
5CDF0072"ADM-CID"0009R3AL22B#111A[#111A|111A 18 1133 0B 000 6]_1716988465.393489
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Enviando mensaje 2/10:
Mensaje de ataque enviado
Enviamos mensaje 3/10
Mensaje enviado:
E6C00088"ADM-CID"0010R3AL22B#111A[#111A|111A 18 1131 0B 000 4][X2.45Y3Z8.443]_1716988470.6964142
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Enviando video...
Mensaje enviado:
24860089"ADM-CID"0011R3AL22B#111A[#111A|111A 18 1131 0B 000 4][VF25.0W640H360]_1716988471.9644752
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Leidos todos los frames.
Enviamos mensaje 4/10
Mensaje enviado:
28E00087"ADM-CID"0012R3AL22B#111A[#111A|111A 18 1131 0B 000 4][X1.5Y2.5Z3.5]_1716988495.6585286
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
```

Figura 14. Prueba del cliente. Mensajes 1 a 4.
Fuente: Elaboración propia.

```
Enviamos mensaje 5/10
Mensaje enviado:
BA260073"ADM-CID"0013R3AL22B#111A[#111A|111A 18 1131 0B 000 4]_1716988497.8233244
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Enviando video...
Mensaje enviado:
73420089"ADM-CID"0014R3AL22B#111A[#111A|111A 18 1131 0B 000 4][VF25.0W640H360]_1716988498.8797126
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Leidos todos los frames.
Enviando mensaje 6/10
Mensaje enviado:
83050073"ADM-CID"0015R3AL22B#111A[#111A|111A 18 1300 0B 000 7]_1716988523.0804482
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Enviando mensaje 7/10
Mensaje enviado:
B8B60155"*ADM-CID"0016R3AL22B#111A[92F77D8224F15CFCFBE7E8372B870F6AE3E911A85D690ABA9E858721C5FC1ABC50278BAA787C981
0F7738630206664A9CC48AF7EEBB91B9D2073DCBA723F9AE9
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Enviando video...
Mensaje enviado:
D3EC0187"*ADM-CID"0017R3AL22B#111A[D2CC1BF606398CE7EEDE8DE9BCF8C4458169A87201A5A5B20BBB5511C89F464F20BB5B2BE63CAD9
3042413E27937686590004EECDAD6CD079817A092C2FCE8BBD01009708D80F79C4C18F04512CBD366
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Leidos todos los frames.
```

Figura 15. Prueba del cliente. Mensajes 5 a 7.
Fuente: Elaboración propia.

```
Enviando mensaje 8/10
Mensaje enviado:
E9550084"ADM-CID"0018R9CL22B#111A[#111A|111A 18 1131 0B 000 4][X10Y20Z30]_1716988552.4846709
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Enviando video...
Mensaje enviado:
7FB60088"ADM-CID"0019R9CL22B#111A[#111A|111A 18 1131 0B 000 4][VF25.0W640H360]_1716988553.744711
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Leidos todos los frames.
Enviando mensaje 9/10
Mensaje enviado:
C5370087"ADM-CID"0020R9CL33D#111A[#111A|111A 18 1131 0B 000 4][X100Y200Z300]_1716988577.4892075
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Enviando video...
Mensaje enviado:
B6C20089"ADM-CID"0021R9CL33D#111A[#111A|111A 18 1131 0B 000 4][VF25.0W640H360]_1716988579.0384681
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Leidos todos los frames.
Enviando mensaje 10/10
Mensaje enviado:
1AFC0043"NULL"0000R9CL33D#234F[]_1716988625.4548967
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
Mensaje enviado:
210C0043"NULL"0000R9CL33D#234F[]_1716988646.4963684
Mensaje enviado, intento número 1
El mensaje fue recibido correctamente
```

Figura 16. Prueba del cliente. Mensajes 8 a 10.
Fuente: Elaboración propia.

4.2 Desarrollo y resultados del servidor

Para comprobar el funcionamiento del servidor, solo hay que activar el servidor para que reciba los mensajes de alarma enviados en las pruebas del cliente. De esta manera, se comprueba que los mensajes enviados son iguales a los mensajes recibidos. Su correcta recepción, desencriptación en caso necesario y archivo de alertas y videos.

Las pruebas del receptor se realizan utilizando el propio receptor. Para ello, se añaden salidas de texto que muestran los mensajes recibidos y los datos extraídos de los mismos. Además, una vez ejecutadas las pruebas se verifica la presencia de los videos con la identificación adecuada y la presencia de las alertas en la base de datos. En la Figura 17, se visualiza la salida por pantalla que muestra el receptor al recibir los mensajes del cliente. En ella, se aprecia como el receptor toma como dos conexiones independientes la recepción del mensaje de alarma con las coordenadas y la recepción del mensaje con los datos de video. Además, se muestra como extrae los datos extendidos de interés y que, cuando se reciben datos de video, el servidor espera y almacena el video. Puesto que la información mostrada es muy amplia, solo se muestra la recepción de los mensajes siete y ocho enviados por el cliente.

```
Datos recibidos de ('192.168.0.24', 52771): b'Hello'
Handshake completado con: ('192.168.0.24', 52771)
Mensaje recibido
El mensaje recibido es:  '\nB8B60155"*ADM-CID"0016R3AL22B#111A[92F77D8224F15CFCFBE7E8372B870F6AE3E911A85D690ABA9E858721C5FC1ABC50278BAA787C9810F7738630206664A9CC48AF7EEBB91B9D2073DCBA72
El mensaje está cifrado
ExtendedData:  X5Y6Z7
Las coordenadas que se extraen son: x: 5, y: 6, z: 7
La fecha recibida es:  1716988526.239953
El mensaje fue recibido correctamente
Al entrar a guardar alarma, x: 5, y: 6, z: 7
Al adaptar los datos x: 5, y: 6, z: 7
Datos recibidos de ('192.168.0.24', 57707): b'Hello'
Handshake completado con: ('192.168.0.24', 57707)
Mensaje recibido
El mensaje recibido es:  '\nD3EC0187"*ADM-CID"0017R3AL22B#111A[D2CC1BF606398CE7EEDE8DE9BCF8C4458169A87201A5A5B20BBB5511C89F464F20BB5B2BE63CAD93042413E27937686590004EECDAD6CD079817A092C2
09708D80F79C4C18F04512CBD366\r'
El mensaje está cifrado
ExtendedData:  VF25.0W640H360
Las coordenadas que se extraen son: x: None, y: None, z: None
La fecha recibida es:  1716988527.2945538
El mensaje fue recibido correctamente
Recibiendo video F25.0W640H360
fps: 25.0, width: 640, height: 360
Cliente iniciado, esperando datos...
Fin del stream de video
Datos recibidos de ('192.168.0.24', 60283): b'Hello'
Handshake completado con: ('192.168.0.24', 60283)
Mensaje recibido
El mensaje recibido es:  '\nE9550084"ADM-CID"0018R9CL22B#111A[#111A|111A 18 1131 0B 000 4][X10Y20Z30]_1716988552.4846709\r'
El mensaje no está cifrado
ExtendedData:  X10Y20Z30
Las coordenadas que se extraen son: x: 10, y: 20, z: 30
La fecha recibida es:  1716988552.4846709
El mensaje fue recibido correctamente
Al entrar a guardar alarma, x: 10, y: 20, z: 30
Al adaptar los datos x: 10, y: 20, z: 30
Datos recibidos de ('192.168.0.24', 62413): b'Hello'
Handshake completado con: ('192.168.0.24', 62413)
Mensaje recibido
El mensaje recibido es:  '\n7FB60088"ADM-CID"0019R9CL22B#111A[#111A|111A 18 1131 0B 000 4][VF25.0W640H360]_1716988553.744711\r'
El mensaje no está cifrado
ExtendedData:  VF25.0W640H360
Las coordenadas que se extraen son: x: None, y: None, z: None
La fecha recibida es:  1716988553.744711
El mensaje fue recibido correctamente
Recibiendo video F25.0W640H360
fps: 25.0, width: 640, height: 360
Cliente iniciado, esperando datos...
Fin del stream de video
```

Figura 17. Prueba del servidor. Mensajes 7 y 8.
Fuente: Elaboración propia.

Respecto al almacenamiento de las alertas, en la Figura 18 se muestra cómo se almacenan en la base de datos.

SQL ▾ 1 / 1 1 - 6 of 6

id	receptor	prefijo	cuenta	protocolo	fecha	hora	video	x	y	z
1	3A	22B	111A	ADM-CID	2024-05-29	14:57:15	static/videos/video_111A_3A_22B__2024-05-29_16-57-15.mp4	2.45	3.0	8.443
2	3A	22B	111A	ADM-CID	2024-05-29	14:57:41	static/videos/video_111A_3A_22B__2024-05-29_16-57-41.mp4	1.5	2.5	3.5
3	3A	22B	111A	ADM-CID	2024-05-29	14:58:08	static/videos/video_111A_3A_22B__2024-05-29_16-58-08.mp4	5.0	6.0	7.0
4	9C	22B	111A	ADM-CID	2024-05-29	14:58:34	static/videos/video_111A_9C_22B__2024-05-29_16-58-34.mp4	10.0	20.0	30.0
5	9C	33D	111A	ADM-CID	2024-05-29	14:58:58	static/videos/video_111A_9C_33D__2024-05-29_16-58-58.mp4	100.0	200.0	300.0
6	9C	33D	234F	ADM-CID	2024-05-29	14:59:23	static/videos/video_234F_9C_33D__2024-05-29_16-59-23.mp4	1000.0	2000.0	3000.0

Figura 18. Base de datos con mensajes de alarma.
Fuente: Elaboración propia.

Respecto al almacenamiento de los videos, en la Figura 19 se muestra la manera en la que se almacenan en una carpeta dedicada a ellos.

Nombre	Estado	Fecha	Tipo	Tamaño	Duración
video_234F_9C_33D__2024-05-29_16-59-23.mp4		29/05/2024 16:59	Archivo MP4	2.070 KB	00:00:12
video_111A_9C_33D__2024-05-29_16-58-58.mp4		29/05/2024 16:59	Archivo MP4	2.070 KB	00:00:12
video_111A_9C_22B__2024-05-29_16-58-34.mp4		29/05/2024 16:58	Archivo MP4	2.070 KB	00:00:12
video_111A_3A_22B__2024-05-29_16-58-08.mp4		29/05/2024 16:58	Archivo MP4	2.070 KB	00:00:12
video_111A_3A_22B__2024-05-29_16-57-41.mp4		29/05/2024 16:57	Archivo MP4	2.070 KB	00:00:12
video_111A_3A_22B__2024-05-29_16-57-15.mp4		29/05/2024 16:57	Archivo MP4	2.070 KB	00:00:12

Figura 19. Carpeta donde se almacenan los videos recibidos por el servidor.
Fuente: Elaboración propia.

4.3 Desarrollo y resultados del entorno de notificación y visualización

Una vez comprobado el servidor receptor de mensajes de alarma, falta por comprobar el entorno de notificación y visualización de las alertas. Este se compone, por un lado, de un sistema de notificaciones vía WhatsApp y, por otro lado, de una interfaz web que muestra todas o parte de las alarmas.

La notificación de las alertas mediante un mensaje de texto es una característica incluida dentro del servidor. Por tanto, las pruebas de funcionamiento se han realizado a la vez que se recibían los mensajes, sin embargo, se detallan en esta sección por tener un enfoque muy diferente a la propia recepción de alarmas. Durante la recepción de mensajes de alarma, cada vez que un mensaje incluía un video se llamaba a una función encargada de mandar mensajes vía WhatsApp. Al utilizar la librería pywhatkit en vez de opciones de pago más sofisticadas, esta era la razón por la que se dejaba un tiempo de espera después de cada envío de alarma con video adjunto por parte del cliente. Si esto no se hace, la notificación con video de dos alertas falla. En la Figura 20 se muestran los últimos mensajes asociados a las alarmas emitidas en las pruebas.

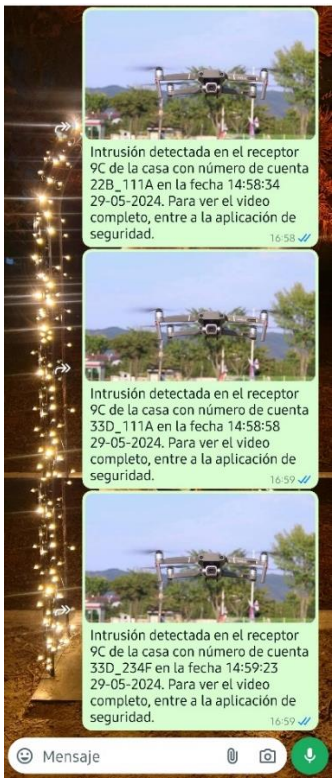

Figura 20. Mensajes de WhatsApp asociados a las alarmas 8, 9 y 10.
Fuente: Elaboración propia.

Respecto a la interfaz web para la visualización de las alarmas, esta sí es un sistema aparte del receptor. En este caso, se trata de una interfaz web que permite visualizar alarmas generadas y descargar los videos asociados a las mismas. En primer lugar, la Figura 21 muestra la ventana de registro donde se pueden crear usuarios con privilegios de administrador o de cliente. En el caso de la imagen, se trata de un administrador.

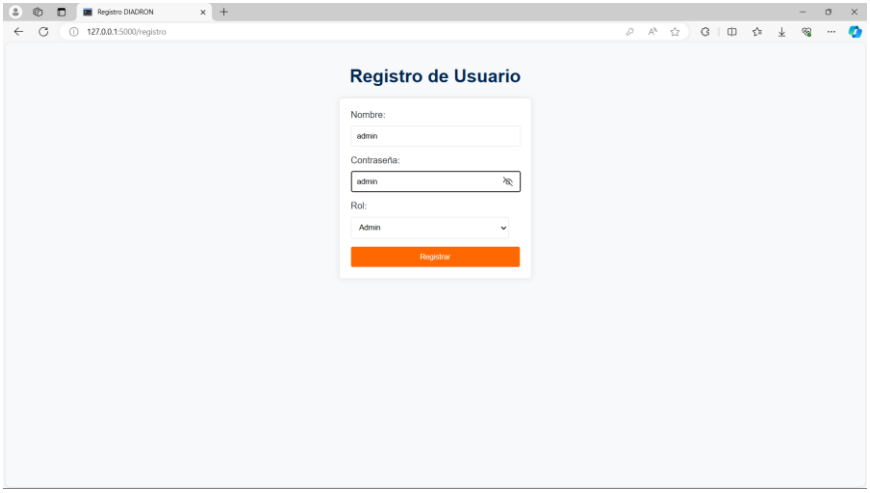

Figura 21. Ventana de registro de usuario de la interfaz web.
Fuente: Elaboración propia.

Una vez registrado un usuario, la siguiente ventana en orden de importancia es la ventana de inicio de sesión. En ella, un usuario registrado puede acceder a visualizar las alarmas. La apariencia es prácticamente idéntica a la ventana de registro, ya que los campos son los mismos, salvo por el hecho de que no aparece el rol del usuario debido a que este es un parámetro del usuario y no un identificador de este. La Figura 22 muestra esta ventana.

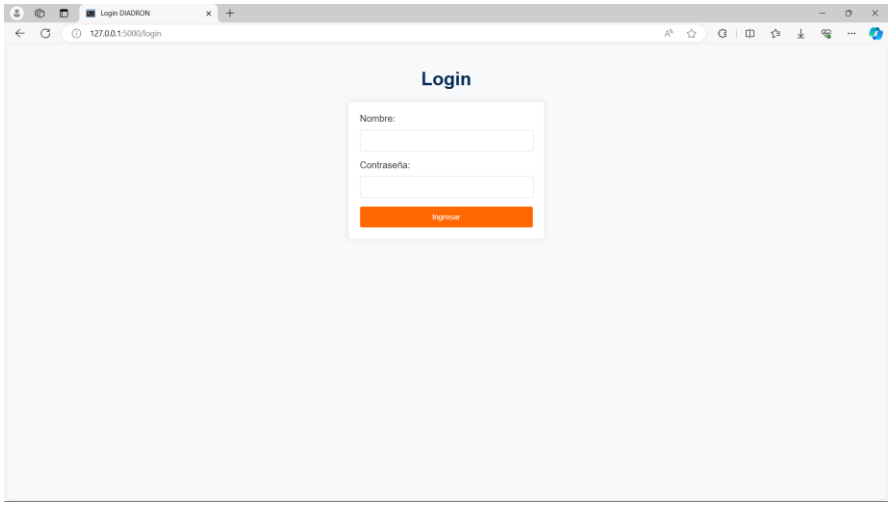

Figura 22. Ventana de inicio de sesión de la interfaz web.
Fuente: Elaboración propia.

Finalmente, la ventana principal y más completa del entorno de visualización de alarmas se compone de una tabla con aquellas alertas que tiene permiso para ver el usuario. Esta tabla está ordenada por fecha de llegada, las alarmas más recientes aparecen las primeras. Si el usuario tiene el rol de administrador, este podrá ver todas las alertas de intrusión generadas. Sin embargo, si el usuario tiene rol de cliente, solo podrá ver aquellas alertas cuyo número de cliente coincida con el nombre de usuario. Los campos visibles son los datos más relevantes de las alertas: identificación del cliente, protocolo del mensaje, fecha, coordenadas de la intrusión y enlace al

video asociado a la intrusión. Al pulsar sobre el texto “Descargar Video”, se descarga el video asociado a la alerta si este existe. Así, la Figura 23 muestra las alertas visibles para el usuario con privilegios de administrador, la Figura 24 muestra las alertas visibles para el usuario 111A y la Figura 25 muestra la alerta visible para el usuario 234F.

Figura 23. Ventana de alertas para el usuario con privilegios de administrador.
Fuente: Elaboración propia.

Figura 24. Ventana de alertas para el usuario 111A.
Fuente: Elaboración propia.

Figura 25. Ventana de alertas para el usuario 234F.
Fuente: Elaboración propia.

Capítulo 5. Conclusiones y propuesta de trabajo futuro

5.1 Conclusiones

En este trabajo final de grado se ha desarrollado un sistema de envío y recepción de mensajes por internet. Más en concreto, se ha desarrollado una solución que se integra dentro de un sistema de detección de alarmas para transmitirlas y comunicarlas a uno o varios usuarios finales. En el proyecto, se ha aprovechado el conocimiento adquirido en los estudios de grado en Ingeniería de Telecomunicaciones para implementar una solución basada en la arquitectura cliente - servidor, utilizando el protocolo de capa de transporte UDP y se ha reforzado la seguridad en la comunicación de los mensajes empleando DTLS.

Para la creación de los mensajes, se han empleado los protocolos Contact ID y SIA DC-09-2021 para formar mensajes de alarma a partir de los datos introducidos por el sistema de detección de drones. Estos mensajes contienen la identificación del cliente que crea la alerta, así como la información más relevante asociada a la misma. En este caso, al tratarse de un sistema de detección de drones, dicha información son datos de posición y video asociado a la detección. Además, el cliente también envía video asociado a las intrusiones detectadas, comprimiéndolo y fragmentándolo para permitir el envío mediante sockets UDP.

Respecto a la recepción de los mensajes, se ha aprovechado el formato fijo de los mensajes enviados siguiendo los protocolos arriba mencionados para poder extraer la información transmitida. Además, se han aprovechado los campos de longitud y CRC de los mensajes para validar la correcta recepción de estos. Así pues, el receptor también reconstruye y descomprime los videos enviados a partir de los fragmentos recibidos. También, el receptor creado en este proyecto guarda las alarmas recibidas en una base de datos junto al video asociado a las mismas.

En cuanto a la seguridad en la transmisión, este proyecto ha incorporado el protocolo de seguridad en la capa de transporte DTLS. Además, se ha utilizado OpenSSL para crear una autoridad certificadora con el objetivo de generar claves que permiten al servidor y a los diferentes clientes autenticarse y establecer una comunicación segura y cifrada evitando suplantaciones de identidad y ataques de tipo Man-in-the-Middle.

Por otro lado, se ha creado un entorno de notificación y visualización de alarmas para que los usuarios finales del sistema puedan acceder a estas. En este contexto, se ha desarrollado un sistema de notificación de alarmas vía WhatsApp. En él, se envía por medio de la aplicación de mensajería instantánea un mensaje de indica que se ha detectado una intrusión, el identificador de cliente asociado a esta, la fecha y un fotograma del video asociado. Además, se ha creado una interfaz web que permite la visualización y descarga de las alarmas. En dicha interfaz, es posible registrar usuarios con diferentes niveles de privilegios, según si deben ver todas las alarmas o solo las asociadas a ellos mismos; también es posible ver, después de iniciar sesión, las alarmas generadas en una tabla. Dicha tabla muestra el identificador de cliente asociado a cada alarma, la fecha y hora de creación, así como las coordenadas y un enlace al video asociado a la intrusión. Dicho video es descargable a voluntad del usuario.

Finalmente, se han llevado a cabo pruebas que confirman el funcionamiento del sistema completo. En relación con los resultados, se ha podido comprobar como el cliente es capaz de transmitir varios tipos de mensaje con diferentes datos asociados a los mismos. Así mismo, se ha comprobado como el receptor recibe y guarda estos mensajes en una base de datos y envía mensajes de notificación al usuario final. Finalmente, la interfaz web permite la visualización y descarga de los mensajes de alerta recibidos por parte de los usuarios. Además, se comprueba como, dependiendo del rol de dichos usuarios, las alertas mostradas son todas las existentes o solamente las asociadas al cliente que inicia sesión.

5.2 Propuesta de trabajo futuro

Como futuros proyectos que puedan surgir a partir de este trabajo final de grado, se proponen una serie de mejoras al presente sistema con el objetivo de ampliar su funcionalidad.

En primer lugar, se propone actualizar el sistema de notificación de alertas por WhatsApp por una solución de carácter profesional. Así, el manejo de estos mensajes por parte del servidor será mejor complejo y, además, será posible ampliar las funcionalidades del sistema de notificación. Como posibles ampliaciones, se propone el envío de videos completos en vez de fotogramas y el acceso directo a la interfaz web para la visualización de las alertas generadas.

En segundo lugar, se propone un trabajo de ampliación de la interfaz web para la visualización de las alarmas. Más en detalle, se propone establecer un sistema de roles con una mayor jerarquía que permita un mayor control de las alarmas que puede ver cada usuario.

Finalmente, se propone permitir el uso de TCP para la transmisión y recepción de paquetes. Esto, aunque no se ha implementado debido a las exigencias del usuario que empleará este proyecto, permitiría un mejor manejo de las conexiones y de la relación entre varios mensajes relativos a una misma intrusión.

Igualmente, faltaría validar con los usuarios finales, empresas de seguridad, sobre la funcionalidad llevada a cabo. A partir de la realimentación con el cliente, se podría mejorar la funcionalidad y dotarla de elementos adicionales que puedan dar lugar a una experiencia de usuario mejorada.

Bibliografía

[1] ADEMCO, «Digital Communication Standard - Ademco ® Contact ID Protocol - for Alarm System Communications,» Security Industry Association, 1999. [En línea]. Available: https://proseriesapi.resideo.com/Resources/PDFs/SIA-ContactIDCodes_Protocol.pdf. [Último acceso: 05 22 2024].

[2] Security Industry Association, «SIA Digital Communication Standard – Internet Protocol Event Reporting,» Security Industry Association, 2020. [En línea]. Available: https://www.securityindustry.org/wp-content/uploads/2017/10/dc09_r2021_20201027.pdf. [Último acceso: 22 05 2024].

[3] Universitat Politècnica de València, «Asignatura Diseño de Servicios telemáticos,» 07 09 2023. [En línea]. Available: https://poliformat.upv.es/access/content/group/GRA_12413_2023/Transparencias/1_Introducci%C3%B3n.pdf. [Último acceso: 23 05 2024].

[4] study-ccna.com, «UDP (User Datagram Protocol) Explained,» study-ccna.com, 2024. [En línea]. Available: https://study-ccna.com/udp-explained/. [Último acceso: 22 05 2024].

[5] P. Mampel, «¿Qué es DTLS - Datagram Transport Layer Security?,» Ringover, 02 08 2023. [En línea]. Available: https://www.ringover.es/blog/dtls. [Último acceso: 22 05 2024].

[6] Digicert, «Qué son los certificados TLS/SSL,» Digicert, 2024. [En línea]. Available: https://www.digicert.com/es/tls-ssl/tls-ssl-certificates. [Último acceso: 22 05 2024].

[7] OpenSSL, «Welcome to OpenSSL!,» OpenSSL, 16 05 2024. [En línea]. Available: https://www.openssl.org/. [Último acceso: 22 05 2024].

[8] G. Sampaio, «Introdução a Arquitetura Cliente-Servidor,» Medium, 21 08 2021. [En línea]. Available: https://medium.com/@kaisergui258/introdu%C3%A7%C3%A3o-a-arquitetura-cliente-servidor-dfae4c0218bd. [Último acceso: 23 05 2024].

[9] F. García de Zúñiga, «Todo sobre la arquitectura cliente-servidor,» Arsys, 08 03 2024. [En línea]. Available: https://www.arsys.es/blog/todo-sobre-la-arquitectura-cliente-servidor. [Último acceso: 22 05 2024].

[10] Naciones Unidas, «La Agenda para el Desarrollo Sostenible,» Naciones Unidas, 2023. [En línea]. Available: https://www.un.org/sustainabledevelopment/es/development-agenda/. [Último acceso: 30 05 2024].

Anexo I. Relación del trabajo con los Objetivos de Desarrollo Sostenible

Los objetivos de desarrollo sostenible son una iniciativa de la ONU que busca que el avance científico y social conduzca a una sociedad más justa para todos [10]. Los llamados ODS son 17 objetivos que han sido apoyados por los más de 200 Estados miembro de las Naciones Unidas y se sitúan dentro de la Agenda 2030.

En línea con estas metas, el presente trabajo final de grado se puede enmarcar en los siguientes objetivos de desarrollo sostenible:

- **ODS 9: Industria, Innovación e Infraestructuras**
- **ODS 11: Ciudades y Comunidades Sostenibles**

Cada uno de los objetivos arriba listados se compone de unas metas que lo definen en profundidad. La aportación del presente proyecto a dichos objetivos se explica con las metas a las que aporta valor. En primer lugar, respecto al ODS 9, este trabajo final de grado contribuye a la siguiente meta:

Meta 9.5 "Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales de todos los países, en particular los países en desarrollo, entre otras cosas fomentando la innovación y aumentando considerablemente, de aquí a 2030, el número de personas que trabajan en investigación y desarrollo por millón de habitantes y los gastos de los sectores público y privado en investigación y desarrollo"

Esto es así ya que se trata de un proyecto que da trabajo a varios ingenieros, los cuales, algunos pertenecen a una empresa privada y otros a la universidad. Por tanto, el desarrollo de este trabajo final de grado permite que se siga avanzando en este campo y se puedan mantener los puestos de trabajo para integrar y mantener el sistema.

En segundo lugar, respecto al ODS 11, este proyecto contribuye a la siguiente meta:

Meta 11.4 "Redoblar los esfuerzos para proteger y salvaguardar el patrimonio cultural y natural del mundo"

En este caso, el presente trabajo final de grado se desarrolla como una herramienta para la comunicación de brechas de seguridad debido a la intrusión de drones. Por tanto, el sistema se puede implementar como medida de seguridad en bienes de interés cultural de todo tipo susceptibles a padecer brechas de seguridad por la presencia de vehículos aéreos.

En definitiva, este proyecto contribuye al avance de la sociedad de manera sostenible. Favorece el crecimiento económico a través de la investigación científica y la seguridad de bienes de interés que deban ser conservados y protegidos frente a todo tipo de intrusiones.