**Esquema de Proyecto de Investigación de Gestión de IoT**

1. **Introducción**
   1. **Planteamiento del problema y motivación.**

En la actualidad, la seguridad de los hogares es una de las principales preocupaciones de las familias. Los robos, incendios y otras situaciones de riesgo pueden ocurrir en cualquier momento, incluso cuando la vivienda está desocupada. Muchos sistemas de seguridad existentes en el mercado son costosos o requieren instalaciones complejas, lo que limita su adopción en hogares con presupuesto reducido.

El proyecto busca ofrecer una solución económica y accesible para monitorear el movimiento y la temperatura en el hogar, alertando de manera inmediata al usuario mediante un sistema de alarma sonora (buzzer) y la visualización de datos en tiempo real a través de una aplicación móvil.

La motivación principal es proporcionar una herramienta de bajo costo, fácil instalación y de código abierto que mejore la seguridad doméstica y brinde tranquilidad a los usuarios.

* 1. ***Alcance del proyecto (qué se cubre y qué queda fuera).***

Lo que se cubre:

* Desarrollo de un sistema basado en ESP32 que detecte movimiento y mida la temperatura ambiente.
* Activación de una alarma sonora (buzzer) en caso de detección de movimiento o temperatura fuera de rango.
* Transmisión de datos en tiempo real a través de Wi-Fi para su visualización en una aplicación móvil.
* Interfaz de la aplicación móvil para ver el estado del sensor y activar/desactivar la alarma de forma remota.

Lo que queda fuera:

* + Integración con cámaras de seguridad o grabación de video.
  + Notificaciones vía SMS o llamadas telefónicas.
  + Integración con asistentes inteligentes (Alexa, Google Home).
  + Alimentación mediante baterías de larga duración (se utilizará alimentación por corriente).
  + Certificaciones oficiales de producto para comercialización.
  1. ***Objetivos (general y específicos).***

**Objetivo general:**

Diseñar e implementar un sistema de monitoreo doméstico basado en ESP32 que detecte movimiento y temperatura, emita una alerta sonora y permita visualizar y controlar el estado del sistema mediante una aplicación móvil.

**Objetivos específicos:**

* + Configurar y programar el ESP32 para recibir datos de un sensor de movimiento (PIR) y un sensor de temperatura.
  + Implementar la activación de un buzzer cuando se detecten condiciones anómalas (movimiento no esperado o temperatura elevada).
  + Desarrollar una aplicación móvil sencilla para mostrar en tiempo real los datos de los sensores y permitir activar/desactivar la alarma.
  + Establecer comunicación inalámbrica (Wi-Fi) entre el ESP32 y la aplicación móvil.
  + Realizar pruebas en un entorno doméstico simulado para verificar el correcto funcionamiento del sistema.
  1. ***Usuarios y beneficios esperados.***

**Usuarios:**

* + Propietarios de viviendas que buscan un sistema de seguridad económico y fácil de instalar.
  + Estudiantes y entusiastas de electrónica/IoT que deseen experimentar con proyectos de monitoreo de hogar.
  + Pequeños negocios que necesiten una solución de alerta básica.

**Beneficios esperados:**

* + Mayor seguridad en el hogar mediante la detección temprana de intrusos o anomalías de temperatura.
  + Tranquilidad para los usuarios al poder monitorear el estado de su vivienda en tiempo real desde cualquier lugar.
  + Bajo costo de implementación en comparación con sistemas de seguridad comerciales.
  + Portabilidad y posibilidad de escalar el sistema agregando nuevos sensores en el futuro.

1. **Marco teórico**
   1. **Breve revisión de soluciones similares (sistemas comerciales y proyectos académicos).**

Con base en el estudio realizado por Raúl Álvarez Salvador y Carlos Alberto Bazán Prieto en 2022, se planteó la creación de un sistema de videovigilancia de bajo costo utilizando hardware y software de código abierto. Para ello, los autores emplearon un módulo ESP32-CAM y analizaron las herramientas disponibles en el entorno de desarrollo Arduino, así como las librerías necesarias para el reconocimiento facial. Como resultado, lograron una configuración mínima viable que, en su contexto, puede ser utilizada para el monitoreo de centros aislados, demostrando que esta tecnología es una solución accesible para la videovigilancia. En consecuencia, esta investigación se relaciona directamente con mi trabajo, ya que aborda la viabilidad técnica y económica de un sistema de seguridad basado en el reconocimiento facial, proporcionando un marco de referencia sólido sobre los componentes y métodos que se pueden aplicar en un contexto similar.

Según la tesis, elaborada por García y Mancheno (2023), el propósito principal fue desarrollar un sistema de control de acceso para dispositivos de seguridad utilizando tecnología IoT. Para ello, los autores emplearon un método descriptivo y de enfoque cualitativo. Como resultado, lograron implementar un sistema de control de acceso que permite la gestión y el monitoreo de dispositivos de seguridad, destacando el uso de tecnología IoT para mejorar la eficiencia y la seguridad. En conclusión, su trabajo demostró la viabilidad de una solución de seguridad basada en IoT, lo cual se relaciona directamente con mi investigación al proporcionar una base sólida y un punto de referencia sobre la aplicación de tecnologías similares en el ámbito de la seguridad y el control de acceso.

En relación con los antecedentes sobre sistemas de videovigilancia y detección de movimiento, López (2023) en su trabajo, se propuso como objetivo la creación de un sistema de videovigilancia con un microcontrolador ESP32-S3 que fuera capaz de capturar, procesar y transmitir video en tiempo real al detectar movimiento. El método empleado se centró en el desarrollo de una conexión bidireccional entre una PC y el microcontrolador, además del procesamiento de imágenes a través del software OpenCV. Como resultado, se logró implementar un sistema con una interfaz de usuario que permite el control del usuario y ofrece cuatro modos de funcionamiento, con la capacidad de realizar el análisis de movimiento tanto en el PC como en el microcontrolador. En conclusión, el autor demostró la viabilidad de desarrollar un sistema de videovigilancia de bajo costo y eficiente utilizando hardware y software accesibles. Este antecedente se relaciona directamente con mi investigación, ya que establece una base sólida sobre el uso de tecnologías como el microcontrolador ESP32 y el análisis de video para la detección de movimiento, elementos clave que también se abordarán en mi proyecto.

* 1. **Tecnologías relevantes (protocolos, plataformas, frameworks).**

**ESP32:** Microcontrolador de bajo costo con conectividad Wi-Fi y Bluetooth, ideal para proyectos IoT.

**Sensores:**

* + **PIR (Passive Infrared):** Para detección de movimiento.
  + **DHT11:** Para medir temperatura y humedad.

**Firebase (Google):** Plataforma en la nube para almacenamiento de datos en tiempo real (Realtime Database o Firestore), lo que permite que los datos sean accesibles desde cualquier lugar.

**Protocolo HTTP/HTTPS o REST API:** Para enviar y recibir datos entre el ESP32 y Firebase de manera segura.

**Aplicación móvil:** Puede desarrollarse usando **Flutter** (Dart) para ser multiplataforma (Android/iOS).

**IDE de programación para ESP32:**

* + **Arduino IDE** para cargar el firmware al ESP32.

**Buzzer:** Actuador que emite señal sonora al detectar eventos de alerta.

* 1. **Lecciones aprendidas y brechas a cubrir por el proyecto.**

**Lecciones aprendidas:**

* + La conectividad a Internet es esencial para sistemas IoT, pero debe ser estable para evitar falsas alarmas o pérdida de datos.
  + Firebase es una solución eficiente para almacenar y sincronizar datos en tiempo real sin necesidad de administrar servidores propios.
  + La interacción entre hardware (ESP32, sensores) y software (aplicación móvil + Firebase) requiere pruebas de integración para evitar inconsistencias.
  + La simplicidad de la interfaz de usuario es crucial: un diseño intuitivo permite que cualquier persona use el sistema sin conocimientos técnicos.

**Brechas a cubrir por el proyecto:**

* + Implementar un sistema de autenticación seguro para que solo el usuario autorizado pueda acceder y controlar la alarma.
  + Minimizar la latencia entre la detección del evento y la notificación en la app móvil.
  + Mejorar la eficiencia energética del sistema para que pueda usarse con baterías en futuras versiones.
  + Diseñar una solución escalable que permita agregar más sensores (gas, humo, puertas/ventanas) sin reestructurar todo el sistema.
  + Garantizar la resiliencia ante fallas de conexión (guardar datos en memoria temporal y enviarlos cuando vuelva la conexión).

1. **Requisitos**
   1. **Requisitos funcionales**
      1. *Listado de funciones (p. ej. lectura de sensor X cada n segundos, envío por MQTT, activación de actuador, panel de control).*

**Gestión de conectividad Wi-Fi:**

* + Activación automática de conexión Wi-Fi al inicio.
  + Activación de **WiFiManager** cuando no haya red disponible para permitir la configuración manual de la conexión desde un dispositivo móvil.

**Lectura de sensores:**

* + **Sensor PIR:** Lectura de detección de movimiento cada 500 ms (o tiempo configurable).
  + **Sensor DHT11:** Lectura de temperatura y humedad cada n segundos (por ejemplo, cada 10 s).

**Control de sensores desde la app:**

* + Activación y desactivación del sensor PIR de forma remota desde la aplicación móvil.
  + Configuración del umbral de temperatura para disparar la alarma desde la aplicación (por ejemplo, que suene a partir de 30 °C).

**Almacenamiento y sincronización en la nube:**

* + Envío de datos de sensores a **Firebase Realtime Database** o Firestore en tiempo real.
  + Actualización de estado del sistema (sensor PIR activado/desactivado, estado del buzzer) en Firebase para reflejarlo en la app.

**Activación de actuadores:**

**Buzzer:**

* + - Sonará cuando el PIR detecte movimiento (si está activado desde la app).
    - Sonará cuando la temperatura supere el umbral definido.
  + Permitir el apagado del buzzer desde la aplicación móvil.

**Interfaz en aplicación móvil:**

* + Pantalla para visualizar en tiempo real:
    - Estado del sensor PIR (activado/desactivado).
    - Última detección de movimiento.
    - Temperatura y humedad actuales.
    - Estado de conexión (online/offline).
  + Botones para:
    - Activar/desactivar el sensor PIR.
    - Configurar el umbral de temperatura para la alarma.
    - Silenciar el buzzer.

**Notificación de eventos en la app:**

* + Mostrar alertas visuales en la app cuando se detecte movimiento o temperatura fuera de rango.
  + Indicar si el ESP32 perdió conexión a Internet.

**Manejo de errores y resiliencia:**

* + Almacenar temporalmente los datos en la memoria del ESP32 si no hay conexión y enviarlos cuando se restablezca.
  + Reintentos automáticos de conexión Wi-Fi antes de activar WiFiManager.
  1. **Requisitos no funcionales**

**Rendimiento (latencia, throughput).**

* + Latencia máxima permitida:
    - La detección de movimiento debe reflejarse en la app y activar el buzzer en menos de 1 segundo.
    - La lectura de temperatura debe actualizarse en Firebase al menos cada 10 segundos.
  + Throughput:
    - El sistema debe poder manejar hasta 10 lecturas por minuto sin saturar el microcontrolador ni la conexión Wi-Fi.
    - La app debe ser capaz de recibir datos en tiempo real sin interrupciones perceptibles para el usuario.

**Disponibilidad y fiabilidad.**

* + El sistema debe funcionar de forma continua 24/7 mientras esté conectado a la red eléctrica.
  + En caso de pérdida de conexión a Internet, el ESP32 debe:
    - Reintentar la conexión automáticamente.
    - Activar el WiFiManager si no logra reconectarse después de n intentos.
    - Almacenar los datos de sensores en memoria temporal y enviarlos a Firebase una vez que se restablezca la conexión.

**Consumo energético (si aplica).**

* + El ESP32 debe operar en modo de bajo consumo cuando no haya eventos de movimiento y el envío de datos esté en espera (deep sleep opcional en futuras versiones).
  + El consumo total del sistema (ESP32 + sensores + buzzer) no debe exceder 500 mA bajo carga normal para ser alimentado por un cargador USB estándar de 5V.

**Escalabilidad**.

* + El diseño debe permitir agregar sensores adicionales (por ejemplo, de humo, gas, apertura de puertas) sin requerir una reestructuración completa del firmware.
  + La base de datos en Firebase debe ser capaz de manejar múltiples dispositivos (más de un ESP32 en distintas habitaciones) reportando datos de manera simultánea.

**Seguridad y privacidad.**

* + La comunicación entre el ESP32 y Firebase debe realizarse mediante HTTPS para garantizar la integridad y confidencialidad de los datos.
  + Implementar autenticación con API key o tokens de Firebase para evitar accesos no autorizados.
  + La app debe requerir inicio de sesión (usuario/contraseña) para activar o desactivar el sistema.

**Restricciones (presupuesto, hardware disponible, normativa).**

* + Presupuesto: El costo total del sistema no debe superar los 100 PEN considerando ESP32, sensores, buzzer y fuente de alimentación.
  + Hardware disponible:
    - ESP32 (DevKit V1 o similar)
    - Sensor PIR HC-SR501
    - Sensor DHT11
    - Buzzer activo
    - Fuente de alimentación 5V
  1. **Historias de usuario / Casos de uso**

Usuario A quiere... (1–5 historias priorizadas).

**HU-1 (Alta prioridad)**

Usuario A quiere:

Activar o desactivar el sensor de movimiento desde la aplicación móvil para decidir cuándo el sistema debe detectar intrusos.

Criterios de aceptación:

* + Cuando el sensor esté activado y detecte movimiento, debe sonar el buzzer y mostrarse la alerta en la app.
  + Cuando el sensor esté desactivado, no debe sonar el buzzer ni mostrar alertas de movimiento.

**HU-2 (Alta prioridad)**

Usuario A quiere:

Recibir alertas en tiempo real cuando la temperatura exceda un umbral configurado para poder tomar medidas preventivas (por ejemplo, evitar sobrecalentamiento o riesgo de incendio).

Criterios de aceptación:

* + La app debe permitir configurar el umbral de temperatura.
  + Si la temperatura supera el umbral, el buzzer debe sonar y la app mostrar una alerta en tiempo real.
  + El usuario debe poder silenciar la alarma desde la app.

**HU-3 (Prioridad media)**

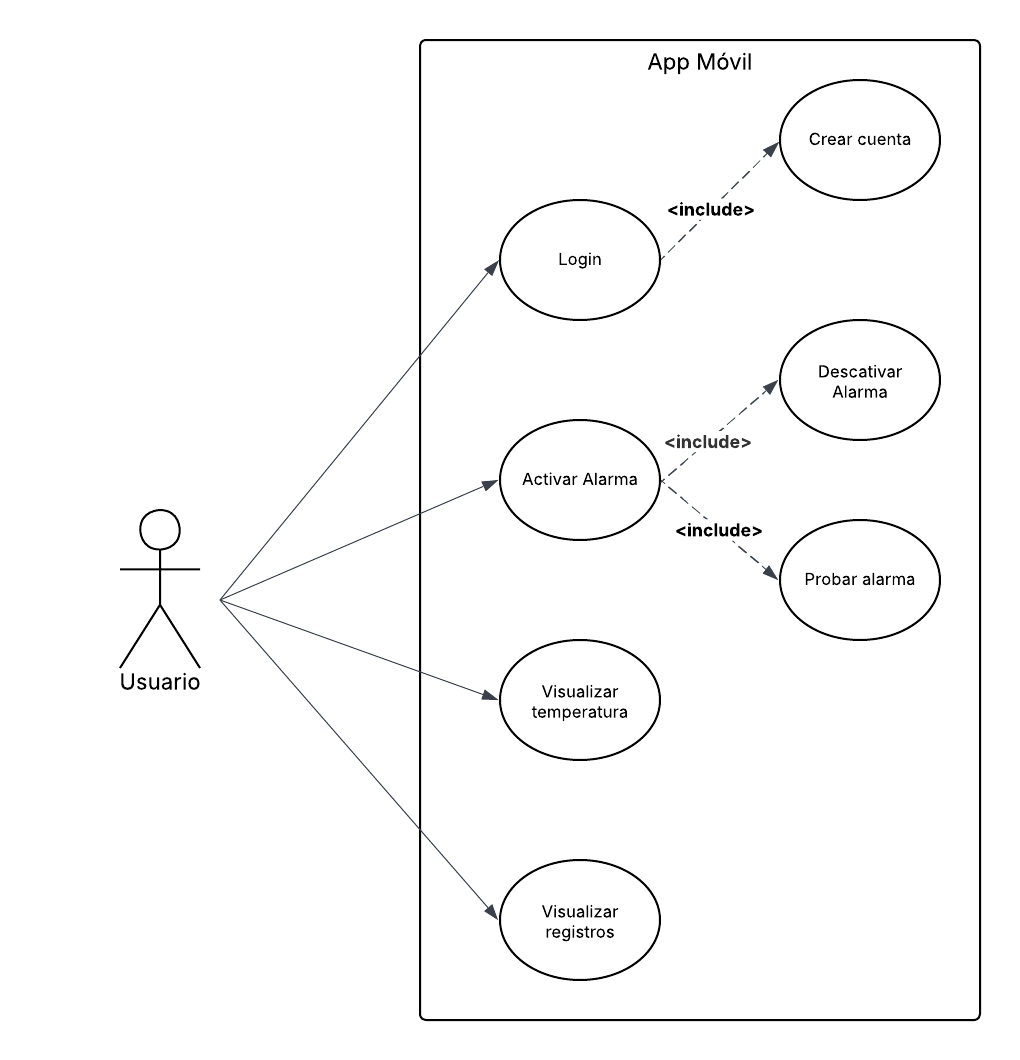
Usuario A quiere:

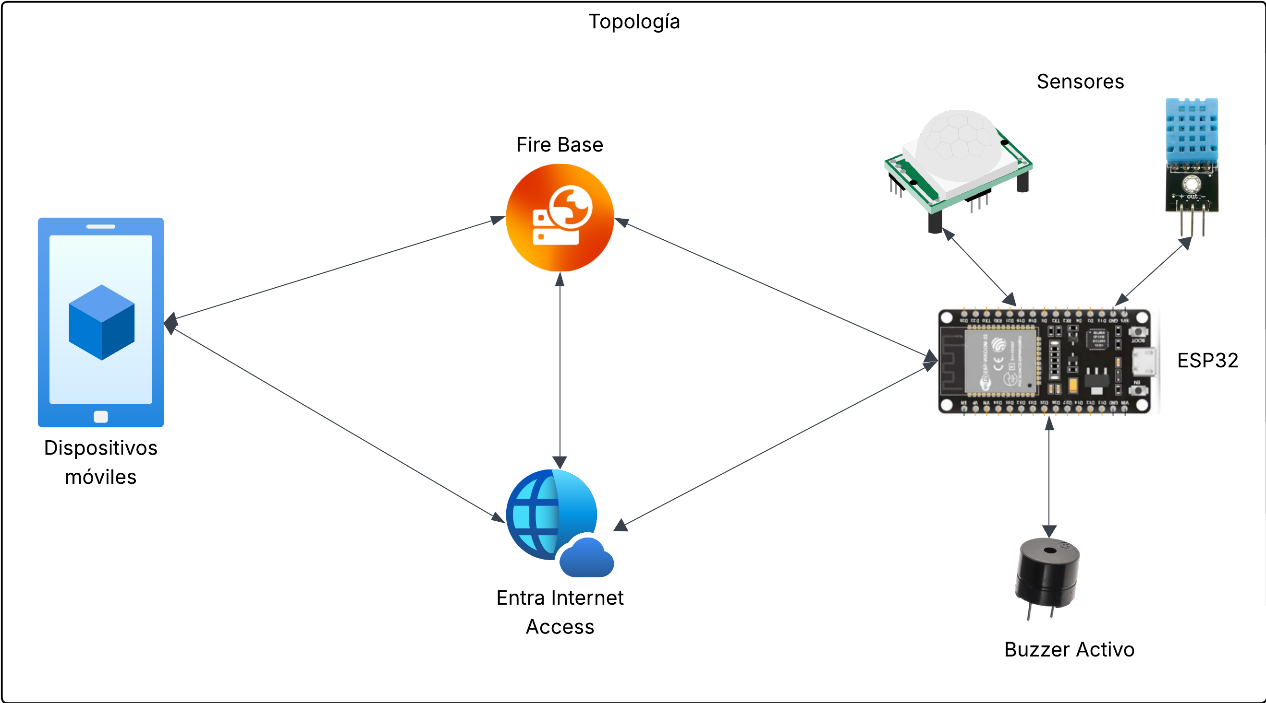
Configurar la conexión Wi-Fi manualmente cuando el dispositivo no pueda conectarse automáticamente, para asegurarse de que el sistema funcione, aunque cambie la red.

Criterios de aceptación:

* + Si el ESP32 no logra conectarse tras varios intentos, debe activar el WiFiManager y crear un punto de acceso.
  + El usuario debe poder conectarse a ese AP y configurar las credenciales de su red Wi-Fi.
  + Una vez configurado, el ESP32 debe reconectarse automáticamente y reanudar el envío de datos a Firebase.

**Diagramas sencillos de casos de uso.**





**4. Diseño de la solución / Arquitectura**

**4.1 Visión general (Diagrama de alto nivel)**

* Diagrama: dispositivos → red → plataforma → aplicación.
* Componentes y responsabilidades.
  1. **Capa de dispositivo (hardware y firmware)**

Selección del microcontrolador:

* + **Modelo:** ESP-WROOM-32 (ESP32)
  + **Características relevantes:** Doble núcleo, conectividad Wi-Fi y Bluetooth integradas, bajo consumo, pines GPIO digitales y analógicos.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Componente** | **Modelo** | **Función** | **Rango / Resolución** | **Notas** |
| Sensor PIR | HC-SR501 | Detectar movimiento | 3–7 m | |  | | --- | |  |   Sensibilidad ajustable |
| Sensor de temperatura y humedad | |  | | --- | |  |   DHT11 | Medir condiciones ambientales | Temp: 0–50C / ± 2C;  Hum: 20–90% / ±5% | Requiere tiempo de muestreo de 2s |
| Actuador | Buzzer activo | Emitir alerta sonora |  | Activado por GPIO cuando se detecta movimiento |

**Gestión de energía:**

El ESP32 puede entrar en modo de ahorro **(deep sleep)** cuando no hay eventos, activándose mediante la interrupción del PIR.

Fuente de energía: adaptador de 5 V o batería Li-ion recargable.

**Organización del firmware:**

El firmware está programado en Arduino IDE, dividido en módulos:

* + **sensors.cpp:** Lectura del DHT11 y PIR.
  + **wifi\_manager.cpp:** Conexión y reconexión a Wi-Fi.
  + **firebase.cpp:** Envío de datos a Firebase Realtime Database.
  + **main.cpp:** Bucle principal y lógica de activación del buzzer.

**4.3 Capa de red / conectividad**

**Tecnologías usadas:**

* + Wi-Fi 2.4 GHz, conexión directa a internet mediante el ESP32.

**Topología de red:**

* + Estrella: Todos los dispositivos (ESP32) se conectan a un punto de acceso Wi-Fi que comunica con la nube.

**Protocolos de comunicación:**

* + Transporte: TCP/IP.
  + Aplicación: HTTP/REST para envío de datos hacia Firebase.

**Estrategia de QoS y tolerancia a fallos:**

* + El firmware valida la conexión Wi-Fi y reintenta automáticamente en caso de desconexión.
  + Si la red falla, los datos se almacenan temporalmente en memoria y se reenvían al reconectarse.
  + Se usan confirmaciones (ACK) desde Firebase para verificar la recepción correcta.

**4.4 Capa de plataforma / middleware**

**Broker / Servicio IoT:**

* + Se utiliza Firebase Realtime Database como plataforma en la nube para la comunicación en tiempo real entre el dispositivo y la aplicación.

**Base de datos:**

* + Firebase Realtime Database (NoSQL): guarda lecturas de sensores y estados de alerta.
  + Estructura:

**Procesamiento de datos:**

* + Procesamiento en la nube (Cloud).
  + Firebase ejecuta la sincronización automática con la app Flutter.

**APIs y flujo de datos:**

1. El ESP32 envía lecturas mediante petición HTTP POST.
2. Firebase almacena y actualiza los valores en tiempo real.
3. La app Flutter consulta y muestra los datos en dashboards.
4. Si se detecta movimiento, Firebase dispara una notificación al usuario.

**4.5 Capa de aplicación (front-end / back-end)**

**Arquitectura del back-end:**

* + Firebase cumple las funciones de backend (autenticación, base de datos y notificaciones push).
  + Arquitectura monolítica basada en la nube (BaaS), sin necesidad de microservicios.

**Interfaz de usuario (App Flutter):**

* + Pantalla de inicio de sesión (correo o Google).
  + Dashboard con:
    - Lecturas actuales de temperatura y humedad.
    - Indicador de movimiento detectado.

**Modelo de Datos**

**5. Seguridad y privacidad (detalle técnico)**

* Modelo de amenaza (ej. STRIDE o similar) y principales riesgos identificados.
* Autenticación de dispositivo (certificados, tokens, mTLS).
* Gestión y almacenamiento de claves (HSM/TPM/software keystore).
* Cifrado en tránsito (TLS) y en reposo (AES).
* OTA y verificación de firmware (firmas digitales).
* Hardening de endpoints y principio de menor privilegio (RBAC).
* Registro y auditoría (logs, retención, análisis).
* Política de privacidad y cumplimiento (datos personales, consentimiento).
* Plan de respuesta a incidentes (procedimiento corto).

**6. Implementación**

**6.1 Hardware: montaje y pruebas iniciales**

* Fotos/esquemáticos.
* Procedimiento de montaje y validación de sensores.

**6.2 Firmware y software embarcado**

* Organización del código, tecnologías, librerías clave.
* Mecanismos de telemetría y tolerancia a fallos.

**6.3 Backend y servicios**

* Estructura del repositorio, contenedores (Docker) y despliegue.
* Endpoints, modelos de datos y ejemplos de payload (JSON/CBOR).

**6.4 Interfaz de usuario**

* Mockups, consistencia UX, accesibilidad.
* Capturas y explicación de funcionalidades.

**7. Plan de pruebas y resultados**

**7.1 Estrategia de pruebas**

* Unit tests, integración, pruebas end-to-end.
* Pruebas de conectividad (pérdida de red, reconexión).
* Pruebas de estrés / carga (número de dispositivos simulados).
* Pruebas de seguridad (scans, fuzzing, pruebas de autenticación).
* Pruebas ambientales (si aplica: temperatura, humedad, altitud).

**7.2 Casos de prueba y logs**

* Tabla de casos de prueba (ID, objetivo, procedimiento, resultado).
* Métricas recogidas: latencia promedio, tasa de muestreo lograda, consumo energético, tasa de paquetes perdidos.

**7.3 Resultados y análisis**

* Gráficos/tabla comparativa antes/después de optimizaciones.
* Lecciones aprendidas y limitaciones.

**8. Conclusiones y recomendaciones**

* Resumen de resultados frente a los objetivos.
* Recomendaciones para mejora, escalado o continuidad.
* Trabajos futuros (extensiones posibles).

**9. Referencias bibliográficas**

* Formato APA / IEEE (obligatorio).
* Enlaces a repositorios y documentación externa.

Salvador, R. A., & Prieto, C. B. (2022). Sistema de seguridad con reconocimiento facial en módulo ESP32. Mare Ingenii, 4(1), 43-52. <https://doi.org/10.52948/mare.v4i1.684>

García Yagual, B. R., & Mancheno Poveda, L. A. (2023). Diseño e implementación de un sistema de control de acceso para dispositivos de seguridad basado en tecnología IoT. ESPOL. FIEC. <http://www.dspace.espol.edu.ec/handle/123456789/57584>

López López, A. (2025). Diseño e implementación de un Sistema de Detección de Movimiento y Transmisión de Video en Tiempo Real con ESP32 (Doctoral dissertation, Universitat Politècnica de València). <https://riunet.upv.es/handle/10251/214946>

**10. Anexos (entregables y material adicional)**

* A. Lista de materiales detallado, incluyendo costos (hoja de cálculo).
* B. Esquemas eléctricos / PCB / Gerbers (si aplica).
* C. Código: estructura de carpetas y archivos clave (ejemplos).
* D. Scripts y comandos para reproducir el despliegue.
* E. Logs completos de pruebas y raw data (CSV).
* F. Manual de usuario y guía de instalación.
* G. Video de demostración (enlace) y diapositivas.