



Universidad Católica de Salta

Facultad de Ingeniería

Carrera: Ingeniería en Informática

Sistema de Vigilancia basado en Microcontrolador con
Notificaciones Móviles en Tiempo Real y Enfoque de
Bajo Costo

Alumno: Ezequiel Barrios

Alumno: Nicolas Oris

Profesor Guía: Ing. Mauricio Guaimas

Año: 2025

Director:

Director de Tesis
Ing. Mauricio Guaimas

Tribunal evaluador:

Presidente del Tribunal Evaluador:

Firma: _____

Miembro del Tribunal Evaluador:

Firma: _____

Miembro del Tribunal Evaluador:

Firma: _____

Fecha de exposición:

____/____/____

Dedicatorias

Dedicamos este proyecto de grado a nuestras familias, por su amor incondicional, paciencia y apoyo constante a lo largo de este camino académico.

A nuestros profesores, quienes con su vocación, enseñanzas y guía nos han inspirado a superarnos día a día.

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a todas las personas que hicieron posible la realización de este proyecto.

A nuestras familias, por ser nuestro mayor soporte emocional y brindarnos siempre palabras de aliento.

A nuestro profesor guía, el Ing. Mauricio Guaimas, por su orientación profesional, sus observaciones valiosas y por motivarnos a llevar adelante el proyecto.

A nuestros amigos y compañeros, por su compañía, sus consejos y por compartir con nosotros este proceso.

Índice

Abstract	10
1. Introducción	11
1.1. Contexto y Justificación.....	11
1.2. Motivación del Proyecto	11
1.3. Objetivos	11
Objetivo General	11
Objetivos Específicos.....	12
1.4 Alcance	12
Alcance Técnico.....	12
Alcance de Usuario	12
Limitaciones del Sistema	13
1.5. Metodología	13
2. Estado de la Cuestión.....	16
2.1. Antecedentes	16
2.2. Soluciones alternativas y alternativa propuesta	17
2.3. Marco Teórico.....	19
2.4. Norma ISO 9001	20
3. Definición del Problema	21
4. Solución propuesta.....	22
4.1. Análisis Comparativo de Tecnologías para el Desarrollo de Backend en Aplicaciones de Procesamiento de Imágenes	22

4.2. Descripción del Sistema a Desarrollar	35
4.3 Seguridad del Sistema.....	38
4.4. Gestión del Proyecto	39
4.4.1. División de Tareas	39
4.4.2. Diagrama de Gantt	41
4.5. Cumplimiento de Objetivos	43
5. Análisis de Factibilidad.....	44
5.1 Recursos Humanos.....	44
5.2 Hardware.....	44
Requerimientos de Hardware.....	44
5.3 Software	45
Requerimientos de Software	45
5.4 Sumatoria de Costos	46
5.5 Factibilidad Legal	46
Observación	47
5.6 Factibilidad Técnica.....	47
5.7 Factibilidad Operativa.....	48
6. Análisis de Riesgos	49
6.1 Matriz de Riesgos	49
6.2 Identificación de Riesgos.....	51
6.3 Análisis FODA	52
7. Resultados	53
7.1. Funcionamiento del Sistema	53

7.2. Pruebas y Validación	54
Pruebas de Funcionamiento	54
Análisis de Resultados	56
7.3. Conclusión	56
7.4. Futuras Líneas de Investigación.....	57
Referencias Bibliográficas	59
Anexos	63
ANEXO A.....	64
ANEXO B.....	83
ANEXO C	93
ANEXO D.....	94

Índice de Tablas

Tabla 1 – Historia de Usuario	24
Tabla 2 - HU001	25
Tabla 3 – HU002.....	25
Tabla 4 – HU003.....	26
Tabla 5 – HU004.....	26
Tabla 6 – HU005.....	26
Tabla 7 – HU006.....	26
Tabla 8 – HU007.....	27
Tabla 9 – HU008.....	27
Tabla 10 – Referencia de Diagrama de Flujo	29
Tabla 11 – Tabla Fotos	37
Tabla 12 – Tabla Usuarios	38
Tabla 13 – Costos de Recursos Humanos.....	44
Tabla 14 – Costos de Hardware	45
Tabla 15 – Costos de Software	46
Tabla 16 – Sumatoria de Costos	46
Tabla 17 – Niveles de Riesgo	50
Tabla 18 – Matriz de Riesgo	50
Tabla 19 – identificación de Riesgos	51
Tabla 20 – Análisis FODA	53
Tabla 21 – Prueba de Detección	55
Tabla 22 – Prueba de envío y recepción de imágenes	55

Tabla 23 – Prueba de notificación	55
---	----

Índice de Ilustraciones

Ilustración 1 - Metodología Cascada	13
Ilustración 2 -Tablero Kanban	28
Ilustración 3 - Diagrama de flujo de captura y envío de fotos.....	30
Ilustración 4 - Diagrama de flujo de Análisis de fotos y envío de notificación.....	32
Ilustración 5 - Diagrama de flujo de visualizar fotos en la aplicación.....	34
Ilustración 6 - Estructura del sistema.....	35
Ilustración 7 - Gestión del Proyecto.....	42
Ilustración 8 - Gestión del Proyecto.....	42
Ilustración 9 - Representación del Funcionamiento del Sistema	54
Ilustración 10 - Microcontrolador ESP32-CAM.....	64
Ilustración 11 - Barra lateral de Arduino IDE	66
Ilustración 12 - Bocetos	67
Ilustración 13 - Administrador de Placas.....	68
Ilustración 14 - Administrador de bibliotecas.....	69
Ilustración 15 - Lista de ejemplos	70
Ilustración 16 - Niveles de la IA	79
Ilustración 17 - Instalador del controlador.....	83
Ilustración 18 - Gestor de URLs	84
Ilustración 19 - Gestor de tarjetas	85
Ilustración 20 - Tarjeta de ESP32	85
Ilustración 21 - Selección de placa	86
Ilustración 22 - Carga de ejemplo	87

Ilustración 23 - Código de Arduino IDE.....	88
Ilustración 24 - Subida de código	89
Ilustración 25 - Mensaje del ESP32.....	90
Ilustración 26 - Configuración de cámara.....	91
Ilustración 27 - Cámara funcionando.....	92
Ilustración 28 - Login.....	95
Ilustración 29 - Registro de usuario	97
Ilustración 30 - Página de Inicio	99
Ilustración 31 - Lista de fotos	100
Ilustración 32 - Botón de llamar a emergencias.....	101
Ilustración 33 - Notificación de alerta	102

Abstract

Este proyecto presenta el desarrollo e implementación de un sistema de vigilancia en tiempo real basado en un microcontrolador de bajo costo con cámara integrada, diseñado para detectar presencia humana y enviar notificaciones móviles de forma instantánea. La solución propuesta utiliza el módulo ESP32-CAM para la captura de imágenes, un servidor backend desarrollado con Python, Flask y OpenCV para el procesamiento y la detección, y una aplicación móvil multiplataforma construida con Ionic para la interacción con el usuario. Las alertas en tiempo real se transmiten mediante WebSockets, permitiendo a los usuarios monitorear su entorno de forma remota y eficiente desde cualquier dispositivo con acceso a internet.

El sistema fue diseñado con un enfoque en la accesibilidad y la economía, resultando ideal para entornos domésticos o pequeños comercios con recursos limitados.

1. Introducción

1.1. Contexto y Justificación

En la actualidad, los sistemas de seguridad domésticos desempeñan un papel crucial al proporcionar protección y tranquilidad en los hogares. Este proyecto se orienta a democratizar el acceso a tecnologías de seguridad, priorizando soluciones económicas, accesibles y eficientes. Tras un análisis exhaustivo de microcontroladores, se eligió el ESP32 por su balance entre costo y capacidades técnicas, así como su compatibilidad con sistemas de inteligencia artificial. Además, se empleará la red neuronal MobileNet para la detección de personas, una herramienta gratuita y de alto rendimiento en aplicaciones de visión por computadora. Complementando el sistema, se desarrollará una Progressive Web App (PWA) con Ionic, permitiendo a los usuarios recibir notificaciones en sus dispositivos móviles de forma intuitiva y accesible. Al aprovechar tecnologías open-source y accesibles, el proyecto fomenta la adopción masiva de soluciones de seguridad tecnológica, maximizando el impacto social al minimizar riesgos y mejorar la calidad de vida en los hogares con un enfoque en el bajo costo.

1.2. Motivación del Proyecto

En el contexto actual de Argentina, la inseguridad es una de las principales preocupaciones de la población. Según estudios recientes del Grupo MIDE, este problema se posiciona como una de las mayores inquietudes nacionales, superando incluso aspectos económicos como la inflación. Este proyecto busca responder a esta problemática desarrollando un sistema de seguridad doméstico avanzado pero accesible, aprovechando tecnologías económicas como el microcontrolador ESP32 y una red neuronal MobileNet para la detección de personas. Complementado por una Progressive Web App (PWA) desarrollada en Ionic, los usuarios podrán recibir notificaciones en tiempo real, aumentando la eficacia del sistema. El enfoque en tecnologías modernas, económicas y de código abierto asegura que esta solución pueda ser adoptada ampliamente, promoviendo la protección y el bienestar en los hogares argentinos.

1.3. Objetivos

Objetivo General

Desarrollar e implementar un sistema de vigilancia en tiempo real, basado en un microcontrolador con cámara integrada, capaz de detectar personas en las imágenes capturadas y enviar notificaciones a dispositivos móviles, utilizando componentes de bajo costo y fácil acceso para facilitar su implementación en entornos domésticos o con recursos limitados. La solución propuesta tendrá un costo total inferior a los 200 USD, ya que se basa en software de

código abierto y solo requiere la adquisición del hardware necesario para su implementación, lo que facilita su implementación en entornos domésticos o con recursos económicos limitados.

Objetivos Específicos

- Diseñar la arquitectura del sistema de vigilancia utilizando el microcontrolador ESP32, integrando el módulo de cámara y conectividad a internet mediante componentes de bajo costo.
- Programar el ESP32 para capturar y transmitir datos en tiempo real al servidor web cada vez que se detecte movimiento.
- Implementar una API REST que permita recibir, almacenar las capturas, detectar personas y enviar notificaciones al usuario en tiempo real.
- Desarrollar una aplicación web progresiva (PWA) que permita al usuario visualizar las capturas de la cámara, consultar el historial de eventos y gestionar el sistema de vigilancia.
- Evaluar la precisión, eficiencia y viabilidad del sistema mediante pruebas en distintos entornos, considerando la conectividad, el tiempo de respuesta y la relación costo-beneficio de la solución propuesta.

1.4 Alcance

Alcance Técnico

- El sistema será capaz de detectar movimiento en una zona específica y capturar una imagen en el momento del evento.
- Utilizará el ESP32 como plataforma central de control para la cámara.
- Se enviará una notificación al usuario a través del servidor flask a una aplicación móvil.
- Se desarrollará una base de datos para registrar el historial de detecciones, incluyendo las imágenes y los datos correspondientes a las mismas.

Alcance de Usuario

- El usuario tendrá acceso a una aplicación móvil donde podrá monitorear el sistema de vigilancia, visualizar alertas en tiempo real, acceder a la totalidad de las fotos y realizar llamadas de emergencias.
- La interfaz de la aplicación será diseñada para ser intuitiva, permitiendo que usuarios sin experiencia técnica puedan utilizarla.

Limitaciones del Sistema

- La calidad de las imágenes y la velocidad de respuesta dependerá de la conexión a internet en el área de implementación.
- El sistema estará limitado a un área de detección específica, determinada por la posición de la cámara.
- Las notificaciones en tiempo real y el acceso a la aplicación móvil dependen de la estabilidad de la red WiFi.

1.5. Metodología

Se utilizará la metodología “Waterfall” (o “desarrollo en cascada”). Esta metodología consiste en desarrollar el proyecto de forma secuencial.

Este paradigma exige un enfoque sistemático y secuencial del desarrollo de software. Abarca las siguientes actividades:

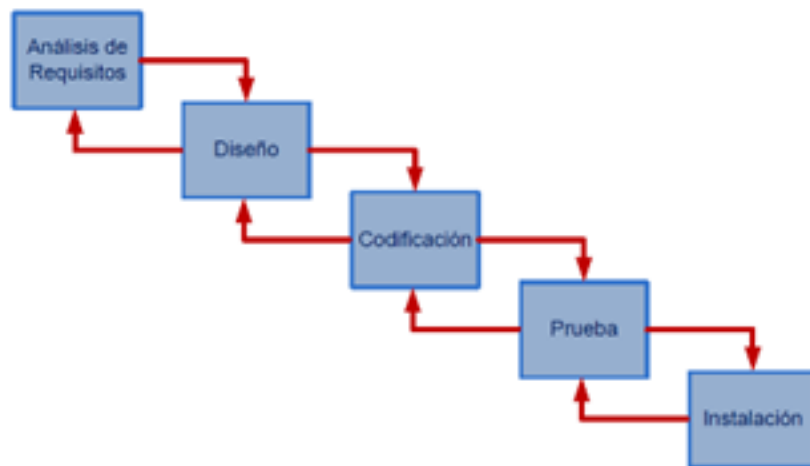


Ilustración 1 - Metodología Cascada

1. Recolección de Requisitos (Inicio del Proyecto)

Se recopila toda la información necesaria sobre el problema a resolver. Se identifican los actores involucrados, se definen claramente los objetivos del proyecto, y se especifica qué se espera lograr con el sistema (funcionalidades mínimas, desempeño, tecnología, usuarios finales, etc.).

- El sistema debe capturar imágenes mediante una cámara.
- El sistema debe detectar la presencia de personas en las imágenes captadas por la cámara.

- El sistema debe notificar en tiempo real al usuario sobre la presencia de personas en el lugar vigilado.

2. Análisis (Estudio del Estado del Arte y Problemática)

Se investigan tecnologías existentes y se analiza el problema desde el punto de vista técnico y práctico. Se profundiza en los componentes que se usarán, su viabilidad y cómo se relacionan con los objetivos planteados.

- Revisar las capacidades del ESP32, cómo se programa con Arduino IDE, cómo se conectará al servidor.
- Analizar el uso de OpenCV para detección de personas.
- Justificar el uso de Ionic para una PWA debido a su compatibilidad multiplataforma.

3. Diseño del Sistema

Se definen las estructuras, interfaces, bases de datos y flujos de trabajo. Se plantean diagramas, historias de usuario, y el diseño general del software y hardware.

- Diagrama de flujo donde el ESP32 captura imágenes → las envía al servidor Flask → Flask procesa con OpenCV → notifica vía WebSocket a la PWA.
- Diseño de la interfaz de usuario de la PWA.
- Tablero Kanban para gestionar tareas del proyecto.

4. Implementación (Desarrollo del Sistema)

Se lleva a cabo la codificación de los distintos componentes del sistema: firmware del ESP32, servidor backend con Flask, procesamiento con OpenCV y desarrollo de la PWA en Ionic.

- Programar el ESP32 para capturar imágenes y enviarlas por HTTP/MQTT.
- Implementar un servidor Flask que recibe las imágenes, las procesa con OpenCV y detecta personas.
- Creas una PWA que se conecta a través de WebSocket y recibe notificaciones cuando se identifica a una persona.

5. Pruebas y Validación

Se realizan pruebas funcionales, de rendimiento, y validación de que el sistema cumple con los requerimientos. Se revisa cada componente por separado y luego el sistema en conjunto.

- Pruebas de precisión de la detección de personas.

- Pruebas de conectividad del ESP32.
- Verificación de que las notificaciones lleguen a tiempo a la PWA.

6. Mantenimiento / Evaluación de Impacto

Se evalúa la viabilidad del sistema a largo plazo, su impacto social, económico o técnico, y se proponen mejoras. También se documentan las lecciones aprendidas.

- Analizar si es viable escalar este sistema para vigilancia en hogares o empresas.
- Sugerir mejoras como agregar reconocimiento de gestos o integración con otros sistemas domóticos.

2. Estado de la Cuestión

2.1. Antecedentes

En los últimos años, los sistemas de videovigilancia han evolucionado significativamente gracias a los avances en el campo del Internet de las Cosas (IoT), la visión por computadora y las aplicaciones móviles. Tradicionalmente, estos sistemas dependían de la observación humana constante o del almacenamiento pasivo de grabaciones. Sin embargo, la integración de algoritmos inteligentes para la detección de eventos y la capacidad de enviar notificaciones en tiempo real ha permitido un monitoreo más eficiente, proactivo y autónomo. Este tipo de soluciones, conocidas como sistemas de vigilancia inteligentes, son cada vez más comunes tanto en entornos residenciales como industriales, debido a su capacidad para reducir los tiempos de respuesta ante incidentes y disminuir la intervención manual.

Uno de los componentes de hardware más utilizados en este tipo de sistemas es el ESP32-CAM, un microcontrolador de bajo costo que incluye una cámara integrada y conectividad Wi-Fi. Su versatilidad y eficiencia energética lo convierten en una herramienta idónea para aplicaciones de videovigilancia de bajo presupuesto. Diversos trabajos y proyectos han explorado su uso en sistemas de monitoreo, particularmente en aquellos que requieren la captura y transmisión remota de imágenes. Además, su capacidad para interactuar con servicios en la nube o con servidores locales permite la implementación de arquitecturas distribuidas y escalables.

El procesamiento de las imágenes capturadas por estos dispositivos se realiza generalmente mediante técnicas de visión por computadora, siendo OpenCV una de las bibliotecas más utilizadas en este ámbito. Esta herramienta permite realizar tareas como la detección de movimiento, el seguimiento de objetos y, en particular, la detección de personas. Con el auge de la inteligencia artificial, se ha extendido el uso de modelos precisos y robustos, como los basados en redes neuronales convolucionales, entre ellos YOLO (You Only Look Once) y MobileNet. Estas técnicas permiten discriminar entre personas y otros objetos, reduciendo la cantidad de falsos positivos en las notificaciones.

Uno de los aspectos clave en los sistemas de vigilancia modernos es la capacidad de emitir notificaciones en tiempo real al usuario final. El desarrollo de aplicaciones móviles juega un papel fundamental en la accesibilidad y control remoto del sistema. En particular, frameworks como Ionic han ganado popularidad debido a su enfoque multiplataforma, lo que permite desplegar aplicaciones tanto en Android como en iOS a partir de una sola base de código, optimizando el tiempo y los recursos de desarrollo.

Finalmente, la arquitectura cliente-servidor empleada en este tipo de sistemas permite una separación clara entre los componentes del sistema: el dispositivo capturador (ESP32-

CAM), el servidor de análisis (basado en frameworks como Flask y bibliotecas como OpenCV) y la interfaz del usuario (aplicación móvil). Esta separación no solo mejora la modularidad del sistema, sino que también facilita su mantenimiento, escalabilidad y adaptación a diferentes contextos. De este modo, se puede construir una solución integral que combina hardware económico, procesamiento inteligente y comunicación eficaz para responder a las necesidades actuales en materia de seguridad y vigilancia.

2.2. Soluciones alternativas y alternativa propuesta

Sistemas de vigilancia existentes

Actualmente, existen diversos sistemas de videovigilancia que incorporan reconocimiento facial y envían notificaciones al teléfono móvil. A continuación, se presentan algunos ejemplos destacados:

1. Prosegur Argentina – Cámara con Reconocimiento Inteligente

Esta cámara permite gestionar la seguridad del hogar desde una aplicación móvil, ofreciendo funciones como visualización en tiempo real, acceso a grabaciones, filtrado de eventos y personalización de notificaciones. Es útil para monitorear accesos, proteger zonas específicas, y vigilar a niños, mascotas o personas mayores.[32]

2. Movistar Prosegur Alarmas – Alarma con Reconocimiento Facial Inteligente

Este sistema utiliza una cámara móvil 360° que detecta la presencia de personas, mascotas y sonidos específicos, enviando notificaciones en tiempo real al móvil del cliente con clips de video del evento. Funciona independientemente de que la alarma esté activada o desactivada.[33]

3. VIVOTEK – Solución de Reconocimiento Facial

VIVOTEK ofrece soluciones que convierten cámaras de vigilancia en sistemas de reconocimiento facial, permitiendo monitorear espacios públicos, proteger puntos de acceso e infraestructura crítica, y agilizar procesos de gestión. Estas soluciones proporcionan autenticación facial precisa y gestión de perfiles.[34]

4. Innovatrics – SmartFace para la Seguridad Pública

SmartFace es una plataforma que mejora el procesamiento de video mediante reconocimiento facial en tiempo real, permitiendo detectar y reconocer identidades humanas y objetos. Emite notificaciones y alertas instantáneas, registra incidentes y facilita su consulta y procesamiento.[35]

5. SISTEMA CCTV FULL HD LITE – Detectar - Seguridad & Informática

Detectar Seguridad es una empresa que se dedica a brindar servicios de seguridad en la Ciudad de Salta. Ofrece un Sistema CCTV Full HD lite 1080N de 1 cámara.[36]

Análisis de Alternativas de Microcontroladores

A continuación, se presentarán algunos microcontroladores, de entre los cuales se escogió el ESP32 para desarrollar el presente trabajo.

ESP8266

Es un microcontrolador de bajo costo con capacidad Wi-Fi, ideal para proyectos simples de IoT. Sin embargo, presenta limitaciones en términos de potencia de procesamiento y funcionalidades adicionales.

Raspberry Pi Pico (RP2040)

Basado en el chip RP2040, es un microcontrolador de bajo costo con procesador de doble núcleo y una variedad de interfaces de E/S. Sin embargo, carece de conectividad inalámbrica integrada.

STM32

Una familia de microcontroladores de 32 bits basada en ARM Cortex-M, conocida por su rendimiento y variedad de periféricos. Sin embargo, no cuenta con conectividad inalámbrica integrada.

Arduino Uno

Basado en el microcontrolador ATmega328P de 8 bits, es ampliamente utilizado por principiantes debido a su simplicidad y extensa documentación. Sin embargo, presenta limitaciones en términos de potencia de procesamiento y carece de conectividad inalámbrica integrada.

Conclusión

El ESP32 se presenta como una opción robusta y versátil para proyectos que requieren alta capacidad de procesamiento y conectividad inalámbrica integrada. Su combinación de características lo hace especialmente adecuado para aplicaciones en el ámbito del Internet de las Cosas (IoT), superando en varios aspectos a otros microcontroladores populares como el ESP8266, Raspberry Pi Pico, STM32 y Arduino Uno.

2.3. Marco Teórico

El desarrollo de sistemas de vigilancia inteligentes ha experimentado una transformación significativa con el avance de las tecnologías digitales, permitiendo soluciones más accesibles, eficientes y escalables. En este contexto, el presente proyecto propone la implementación de un sistema de vigilancia que integra diversos componentes tecnológicos clave como microcontroladores, Internet de las Cosas (IoT), Aplicaciones Web Progresivas (PWA), bases de datos, API REST, y técnicas de Inteligencia Artificial, todo orquestado bajo una arquitectura cliente-servidor.

El uso de microcontroladores, como Arduino o ESP32, permite la interacción con sensores de movimiento, cámaras y otros dispositivos físicos que conforman el sistema de monitoreo. Estos dispositivos actúan como nodos de recolección de datos dentro de una red IoT, facilitando la comunicación entre el entorno físico y el entorno digital. Mediante protocolos de comunicación como MQTT, HTTP o WebSockets, los datos capturados son enviados al servidor en tiempo real para su procesamiento y almacenamiento.

La integración con el paradigma de Internet de las Cosas (IoT) habilita una comunicación constante entre dispositivos y servicios en la nube, permitiendo la supervisión remota del sistema desde cualquier lugar con conexión a Internet. Para facilitar el acceso y la interacción del usuario, se ha optado por el desarrollo de una Aplicación Web Progresiva (PWA), la cual combina lo mejor de las aplicaciones web y móviles: es ligera, rápida, puede funcionar sin conexión, y ofrece soporte para notificaciones en tiempo real, cruciales para alertar a los usuarios ante eventos de seguridad.

El sistema se estructura bajo una arquitectura cliente-servidor, donde el cliente (la PWA) se comunica con el servidor a través de una API bien definida. Esta interfaz facilita la manipulación de datos almacenados en una base de datos relacional o NoSQL, que guarda registros de eventos, usuarios, dispositivos y análisis generados. Además, se hace uso de Client-Side Rendering (CSR) para optimizar la experiencia de usuario, mejorando la velocidad de carga y la interactividad de la aplicación.

La Inteligencia Artificial cumple un rol fundamental al permitir la detección inteligente de patrones o comportamientos sospechosos mediante algoritmos de visión por computadora o aprendizaje automático. Esta capacidad añade una capa de análisis predictivo al sistema, mejorando la precisión de las alertas y reduciendo falsos positivos.

En conjunto, estos elementos conforman una solución robusta de vigilancia que no solo permite la supervisión en tiempo real de espacios físicos, sino que también ofrece escalabilidad, accesibilidad multiplataforma y una experiencia de usuario moderna y eficiente.

2.4. Norma ISO 9001

La **ISO 9001:2015** es una norma internacional que establece los requisitos para un sistema de gestión de calidad (SGC). Esta norma está diseñada para ayudar a las organizaciones a garantizar que sus productos y servicios cumplan con las expectativas de calidad de los clientes y se adhieran a la normativa vigente. A través de su implementación, se busca mejorar la satisfacción del cliente, la eficiencia operativa y el cumplimiento de las expectativas legales.

La versión 2015 de esta norma incorpora un enfoque basado en riesgos y un sistema más flexible y adaptable a distintos tipos de organizaciones.

Principales características de la ISO 9001:2015

1. **Enfoque en el cliente:** La norma pone un fuerte énfasis en la satisfacción del cliente, buscando cumplir con sus requisitos y mejorar la relación con los mismos.
2. **Liderazgo:** Promueve el compromiso y liderazgo de la alta dirección para crear un ambiente que apoye el sistema de gestión de calidad.
3. **Compromiso de las personas:** La norma subraya la importancia de involucrar a todos los miembros de la organización para mejorar continuamente los procesos.
4. **Enfoque basado en procesos:** Se enfoca en la gestión de los procesos de la organización para lograr resultados más eficaces y eficientes.
5. **Mejora continua:** Promueve la mejora constante de los procesos, productos y servicios para mantener la calidad y competitividad.
6. **Toma de decisiones basada en la evidencia:** La norma fomenta el uso de datos e información precisa para tomar decisiones que mejoren los procesos.
7. **Gestión de las relaciones:** Enfocada en el establecimiento y mantenimiento de relaciones mutuamente beneficiosas con proveedores y otras partes interesadas.
8. **Enfoque de riesgos:** Se introduce la gestión de riesgos para identificar y minimizar potenciales problemas y para mejorar la resiliencia de la organización.[31]

3. Definición del Problema

En la actualidad, la seguridad es un aspecto crítico en múltiples entornos, como hogares, empresas, instituciones educativas e infraestructuras públicas. Sin embargo, los sistemas de vigilancia tradicionales presentan diversas limitaciones que reducen su efectividad y accesibilidad.

Uno de los principales problemas es la falta de vigilancia continua y accesible desde cualquier lugar. Muchos sistemas de monitoreo dependen de hardware propietario y requieren acceso a software específico, lo que dificulta la supervisión remota y puede retrasar la respuesta ante eventos de seguridad.

Además, los sistemas comerciales suelen implicar altos costos de implementación y mantenimiento, ya que requieren cámaras especializadas, licencias de software y servidores centralizados. Esta barrera económica limita su adopción, especialmente en pequeñas y medianas empresas o usuarios domésticos con presupuestos ajustados.

Otro desafío importante es la dependencia de soluciones cerradas y de terceros, lo que genera problemas de privacidad, falta de personalización y dificultades para la integración con otros sistemas. En muchos casos, los datos de vigilancia se almacenan en servidores externos, lo que puede representar un riesgo si no se cumplen con estándares adecuados de protección de datos.

Para abordar estas deficiencias, se propone el desarrollo de un sistema de vigilancia en tiempo real basado en una ESP32-CAM y una Progressive Web App (PWA) desarrollada con Ionic.

La ESP32-CAM es un módulo de bajo costo con conectividad Wi-Fi y capacidad para capturar imágenes y transmitir video en tiempo real. Su uso permite implementar un sistema de vigilancia económico y escalable, eliminando la necesidad de hardware costoso y propietario.

Por otro lado, la aplicación móvil desarrollada con Ionic permitirá a los usuarios recibir notificaciones instantáneas ante eventos sospechosos, visualizar imágenes en tiempo real, acceder al historial de capturas y tomar acciones inmediatas, como llamar a emergencias. Al ser una PWA, la aplicación podrá ejecutarse en cualquier dispositivo sin necesidad de instalación, brindando una mayor accesibilidad y compatibilidad con distintos sistemas operativos.

Esta solución permite a los usuarios monitorear su entorno desde cualquier ubicación y reaccionar rápidamente ante posibles amenazas. Al estar basada en tecnologías de código

abierto, el sistema ofrece mayor transparencia, flexibilidad y capacidad de personalización en comparación con soluciones comerciales cerradas.

4. Solución propuesta

Las soluciones comerciales disponibles en el mercado suelen presentar costos elevados, requerimientos técnicos complejos o dependencia de servicios de terceros, lo que las hace inaccesibles para muchos usuarios, especialmente en entornos con recursos limitados.

En este sentido, el desarrollo de un Sistema de Vigilancia basado en microcontrolador, con capacidad de notificaciones móviles en tiempo real y un enfoque de bajo costo, se presenta como una alternativa viable, accesible y eficiente. El uso de microcontroladores como el ESP32-CAM permite reducir significativamente el costo del hardware sin sacrificar funcionalidades esenciales como la captura de imágenes, conectividad Wi-Fi y procesamiento básico en el borde, aunque es importante aclarar que para la correcta detección de personas es necesario que la iluminación sea buena, ya que esta cámara al ser de bajo costo, no posee visión nocturna. Además, la integración con plataformas móviles para el envío de alertas instantáneas mejora la capacidad de respuesta ante eventos, sin necesidad de monitoreo constante o infraestructura costosa.

Este tipo de sistema es especialmente adecuado para aplicaciones donde se requiere una solución de vigilancia básica pero funcional, como viviendas particulares, pequeñas y medianas empresas (PyMEs), establecimientos educativos o instituciones públicas con limitaciones presupuestarias. Asimismo, su carácter modular y adaptable permite futuras expansiones, como el reconocimiento facial o el almacenamiento en la nube, lo que incrementa su utilidad y escalabilidad.

4.1. Análisis Comparativo de Tecnologías para el Desarrollo de Backend en Aplicaciones de Procesamiento de Imágenes

La selección de la tecnología adecuada para el desarrollo de aplicaciones backend es un aspecto crucial en el diseño de sistemas de software. En aplicaciones de procesamiento de imágenes y detección de personas mediante inteligencia artificial (IA), la eficiencia en el manejo de cálculos intensivos y la disponibilidad de bibliotecas especializadas juegan un papel determinante. En este estudio, se realiza un análisis comparativo entre diversas tecnologías backend, evaluando sus ventajas y limitaciones con respecto a Python y Flask, los cuales han sido seleccionados para la implementación del presente proyecto.

Se han considerado cuatro tecnologías de desarrollo backend que representan opciones viables para el proyecto:

- Python (Flask)
- Node.js
- Java
- C# (.NET)

Cada una de estas tecnologías se analizará en función de los siguientes criterios:

- Disponibilidad de bibliotecas para IA y procesamiento de imágenes
- Rendimiento y escalabilidad
- Facilidad de desarrollo
- Comunidad y soporte

Conclusión

Tras analizar las tecnologías disponibles, se concluye que Python y Flask representan la mejor opción para el desarrollo del backend en aplicaciones de procesamiento de imágenes y detección de personas. La disponibilidad de bibliotecas especializadas, la simplicidad en el desarrollo y el fuerte respaldo de la comunidad son factores determinantes que favorecen su elección.

Aunque tecnologías como Node.js, Java y C# ofrecen ventajas en ciertos escenarios, su eficiencia en procesamiento de imágenes y aprendizaje automático es inferior en comparación con Python.

Este análisis respalda la selección de Flask como framework para el backend del sistema propuesto, asegurando un desarrollo eficiente y alineado con los objetivos del proyecto.

Requerimientos Funcionales y No Funcionales

Los requerimientos funcionales describen el comportamiento del sistema a implementar, es decir, definen cuáles son los servicios o funcionalidades que deberán incluirse en el desarrollo del sistema. Por lo tanto, se definieron los siguientes:

- Iniciar sesión
- Al momento de ingresar nombre de usuario y contraseña se debe validar que sea correcta.
- Los mensajes de notificación por campos vacíos se realizan a través de Toast. Mostrar un mensaje al tener una autenticación fallida.
- El usuario podrá visualizar las fotos de la cámara de seguridad en la aplicación desarrollada.
- El usuario deberá configurar su perfil en la aplicación.

- Para el ingreso de claves a través de la aplicación o teclado digital serán validados para que sean solo números.
- El usuario deberá recibir notificaciones.
- El usuario deberá poder filtrar las fotos por fecha.
- El usuario deberá poder ver las fotos de forma paginada.
- El usuario deberá poder llamar a emergencias.
- El usuario deberá poder ver el historial de eventos.

Los requerimientos no funcionales son aquellos que definen las propiedades del sistema: rendimiento, seguridad, disponibilidad, es decir, el “cómo” se cumplirá el funcionamiento del sistema integrado. Se definieron:

- Cerrar sesión
- El tiempo de respuesta debe ser casi inmediato.
- La autenticación debe ser por medio de JWT
- La notificación desde el servidor a la aplicación será por medio de websockets

Historia de Usuario

Las Historias de Usuario son utilizadas para describir lo que el usuario es capaz de hacer. Son herramientas utilizadas para especificar las funcionalidades del Sistema y describen los requerimientos del cliente a un nivel más fino. Para realizar las Historias de Usuario se tomará el formato siguiente.

Historia de Usuario	
ID:	Usuario:
Programador Responsable:	
Nombre Historia:	Tipo:
Descripción:	
Comentario:	

Tabla 1 – Historia de Usuario

Las leyendas son:

- ID: identificador de la Historia de Usuario.
- Usuario: Indica el usuario que realizará las funcionalidades.

- Programador Responsable: nombre de la persona encargada de programar o realizar esta Historia de Usuario.
- Nombre de historia: Nombre o título que describe la funcionalidad.
- Tipo: Indica si es un requerimiento funcional o no funcional.
- Descripción: se describe de manera concisa la funcionalidad.
- Comentario: información adicional y detallada de la descripción.

Estas son las Historias de Usuario:

ID: HU001	
Usuario: Cliente	
Nombre Historia: Iniciar Sesión	Tipo: Requerimiento Funcional
Descripción: Autenticación de usuario.	
Comentario: El usuario podrá autenticarse a través de nuestro sistema.	

Tabla 2 - HU001

ID: HU002	
Usuario: Cliente	
Nombre Historia: Cerrar Sesión	Tipo: Requerimiento No Funcional
Descripción: El usuario podrá cerrar sesión.	
Comentario: El usuario podrá salir del sistema.	

Tabla 3 – HU002

ID: HU003	
Usuario: Cliente	
Nombre Historia: Visualizar últimas fotos	Tipo: Requerimiento Funcional
Descripción: El usuario podrá ver las últimas fotos tomadas por la cámara.	

Comentario: El usuario podrá ver las últimas fotos tomadas por la cámara.

Tabla 4 – HU003

ID: HU004	
Usuario: Cliente	
Nombre Historia: Ver fotos paginadas	Tipo: Requerimiento Funcional
Descripción: El usuario podrá ver las fotos de forma paginada.	
Comentario: El usuario podrá ver las fotos de forma paginada.	

Tabla 5 – HU004

ID: HU005	
Usuario: Cliente	
Nombre Historia: Recibir notificaciones de detección	Tipo: Requerimiento Funcional
Descripción: El usuario recibirá notificaciones de movimiento.	
Comentario: El usuario recibirá una notificación si el sistema detecta a una persona.	

Tabla 6 – HU005

ID: HU006	
Usuario: Cliente	
Nombre Historia: Filtrar fotos por fecha	Tipo: Requerimiento Funcional
Descripción: El usuario podrá filtrar fotos por fecha.	
Comentario: El usuario podrá filtrar fotos por fecha.	

Tabla 7 – HU006

ID: HU007	
Usuario: Cliente	
Nombre Historia: Visualizar historia de eventos	Tipo: Requerimiento Funcional
Descripción: El usuario podrá ver el historial de eventos.	
Comentario: El usuario podrá ver todas las veces que se le notificó y la hora de la notificación.	

Tabla 8 – HU007

ID: HU008	
Usuario: Cliente	
Nombre Historia: Llamar a emergencias	Tipo: Requerimiento Funcional
Descripción: El usuario podrá llamar a emergencias.	
Comentario: El usuario podrá llamar al 911 desde la aplicación.	

Tabla 9 – HU008

Tablero Kanban

Una vez creadas las historias de usuario, se dividen las tareas del proyecto usando el tablero Kanban. Dentro de cada tarjeta se desarrollan tareas específicas. Aquí se pueden ver las tareas pendientes, en progreso y finalizadas del proyecto.



Ilustración 2 -Tablero Kanban

Estructura General del Sistema

El Sistema fue desarrollado con una estructura cliente, Web API, base de datos, controladores y actuadores. Se lo describe de la siguiente manera:

- Cliente: aplicación móvil que gestiona el cliente con las opciones indicadas dentro del sistema integrado.
- Web API: donde se procesan las peticiones que realiza el cliente proveniente de la aplicación móvil.
- Base de Datos: donde se almacenan los datos.
- Cámara: dispositivo que se encarga de tomar fotos que posteriormente serán enviadas al servidor.

Diagrama de Flujo

Estos diagramas representan los distintos flujos de actividades que realizan los diferentes procesos del sistema como ser: Captura y envío de imágenes, visualizar imágenes, llamar a emergencias. Antes de mostrar los flujos se mostrará una referencia.

Símbolo	Nombre	Descripción
	Tarea.	Es una actividad atómica.
	Compuerta exclusiva.	Representa una toma de decisión.
	Flujo de secuencia.	Muestra el orden en el que las actividades son ejecutadas.
	Inicio.	Indica donde inicia el proceso.
	Fin.	Indica donde termina el proceso.

Tabla 10 – Referencia de Diagrama de Flujo

Captura y envío de fotos

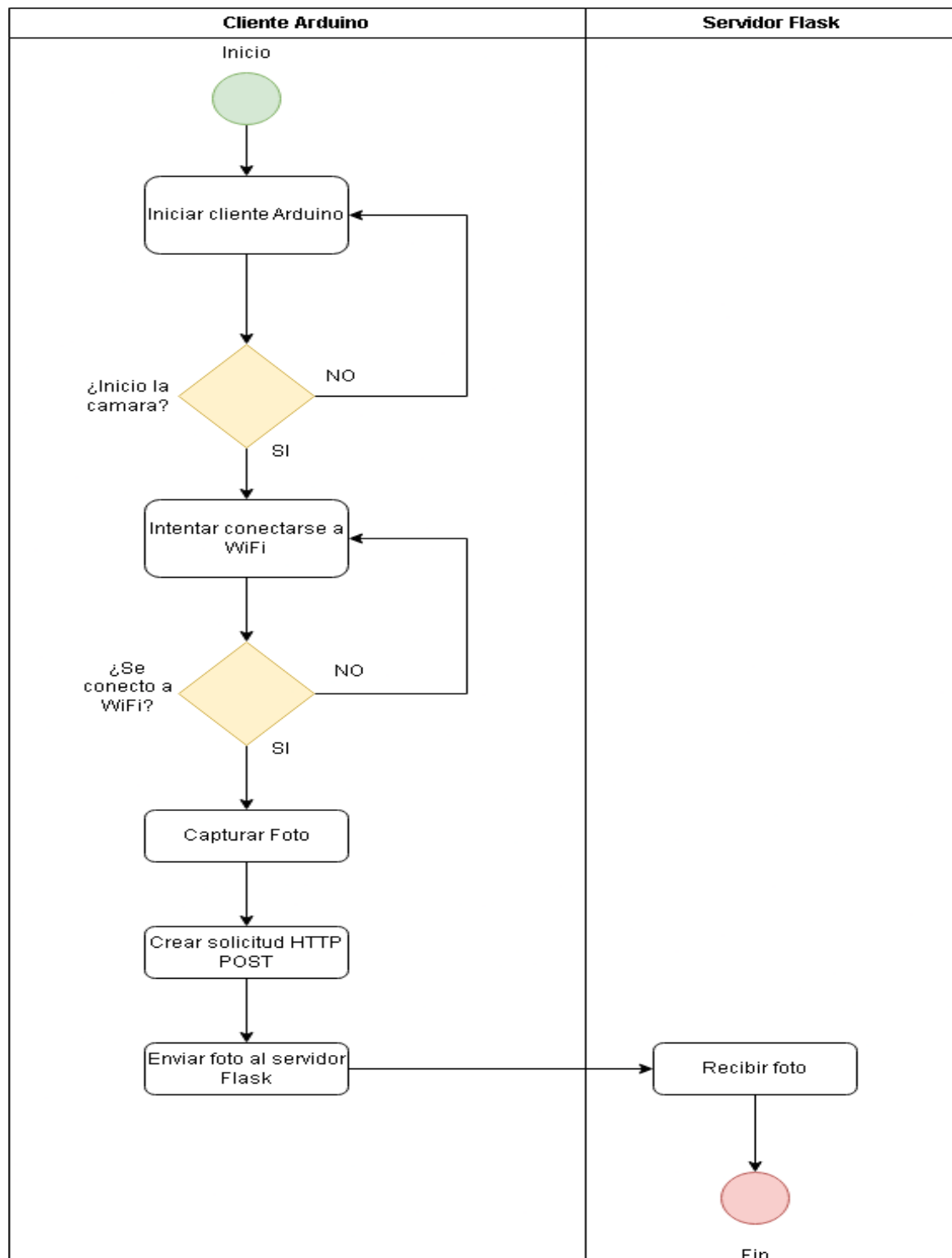


Ilustración 3 - Diagrama de flujo de captura y envío de fotos

El diagrama de flujo presentado ilustra el proceso de captura y envío de imágenes desde el cliente Arduino, específicamente un módulo ESP32-CAM, hacia el servidor backend desarrollado con Flask. Este flujo constituye una de las piezas fundamentales del sistema de vigilancia, ya que permite iniciar el monitoreo y la detección de personas a través del análisis de las imágenes capturadas.

El proceso se inicia en el dispositivo cliente, donde se realiza la inicialización del módulo Arduino. A continuación, se verifica si la cámara ha sido correctamente iniciada. En caso de falla, el sistema permanece en espera, intentando inicializar nuevamente. Una vez que la cámara está activa, el dispositivo procede a conectarse a una red Wi-Fi.

Si la conexión Wi-Fi no se establece, el proceso no avanza hasta que se logre la conectividad. Cuando se confirma la conexión, el ESP32-CAM captura una imagen del entorno. Esta imagen es entonces empaquetada en una solicitud HTTP de tipo POST, la cual es enviada al servidor Flask alojado localmente. En el servidor, se recibe la imagen y finaliza el proceso.

Análisis de fotos y envío de notificación

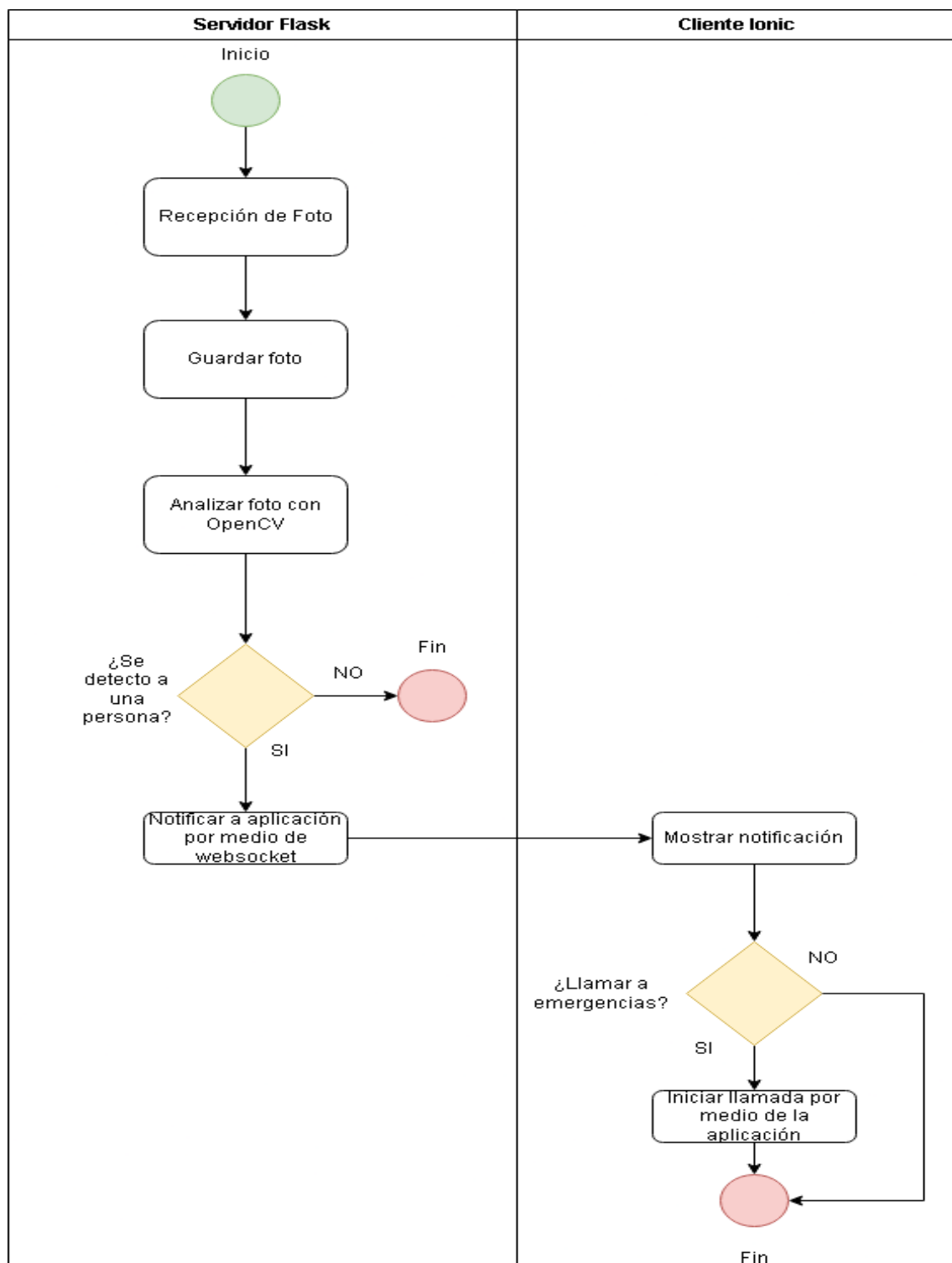


Ilustración 4 - Diagrama de flujo de Análisis de fotos y envío de notificación

El siguiente diagrama de flujo describe el proceso que se lleva a cabo una vez que el servidor Flask ha recibido una imagen desde el cliente Arduino. Este flujo representa el núcleo del análisis inteligente del sistema, así como la emisión de alertas hacia el usuario a través de la aplicación móvil.

El proceso inicia con la recepción de la imagen enviada por el ESP32-CAM. Una vez recibida, la imagen es almacenada en el servidor para su posterior procesamiento o consulta. Luego, se utiliza la biblioteca OpenCV para analizar la imagen con el objetivo de detectar si en ella aparece una persona.

Si no se detecta la presencia de una persona, el flujo finaliza sin tomar más acciones. En cambio, si la detección es positiva, el servidor procede a emitir una notificación hacia la aplicación móvil utilizando comunicación en tiempo real mediante WebSockets. Esto permite que la alerta llegue de forma inmediata al usuario.

Del lado del cliente móvil, la aplicación recibe y muestra la notificación visualmente al usuario. A partir de allí, se le ofrece la opción de iniciar una llamada de emergencia. Si el usuario decide llamar, la aplicación activa la función correspondiente del dispositivo para realizar la llamada al número de emergencia. Si no se desea contactar a emergencias, el proceso finaliza tras la notificación.

Este flujo garantiza la automatización del monitoreo, permitiendo una respuesta inmediata del sistema ante eventos críticos, y brinda al usuario una herramienta de acción directa en situaciones que lo requieran, sin necesidad de interacción previa con el servidor.

Visualizar fotos en la aplicación

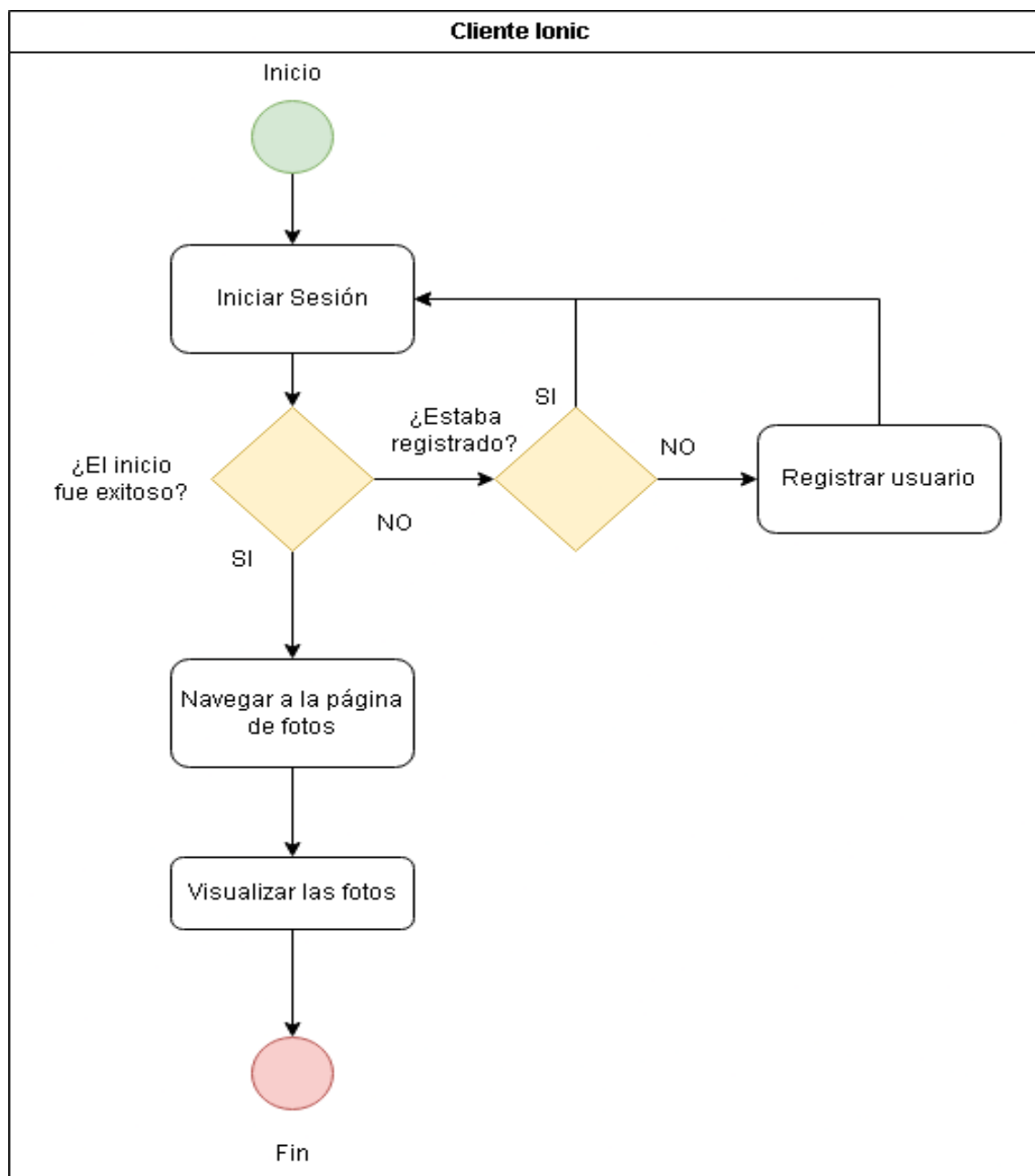


Ilustración 5 - Diagrama de flujo de visualizar fotos en la aplicación

El siguiente diagrama de flujo representa el proceso de inicio de sesión y navegación dentro de la aplicación móvil desarrollada con el framework Ionic. Esta funcionalidad es fundamental para garantizar que el acceso a la información registrada por el sistema de vigilancia esté restringido únicamente a usuarios autorizados.

El proceso comienza con la interfaz de inicio de sesión, donde el usuario debe ingresar sus credenciales. Una vez completado este paso, se realiza una verificación sobre si el usuario está registrado en el sistema. En caso de que no lo esté, se le redirige a una pantalla para proceder con el registro de usuario, asegurando así el cumplimiento de una autenticación previa antes de acceder a las funcionalidades de la aplicación.

Si el usuario ya se encuentra registrado y el inicio de sesión es exitoso, se lo redirige a la página principal de visualización de fotos, donde podrá observar las imágenes captadas por el sistema de vigilancia. Esta sección se presenta en forma paginada, permitiendo una navegación ordenada y eficiente entre las distintas capturas almacenadas en el servidor.

Este flujo garantiza un acceso seguro, estructurado y enfocado en la usabilidad, permitiendo que el usuario interactúe con el sistema de forma clara, solo tras haberse autenticado correctamente.

4.2. Descripción del Sistema a Desarrollar

Diseño de la Arquitectura del Sistema

El sistema contará con una cámara ESP32 para la obtención de imágenes, luego se enviarán a una API en flask para almacenarla y realizar el reconocimiento. Si detecta algo inusual entonces enviará una notificación a la aplicación móvil.

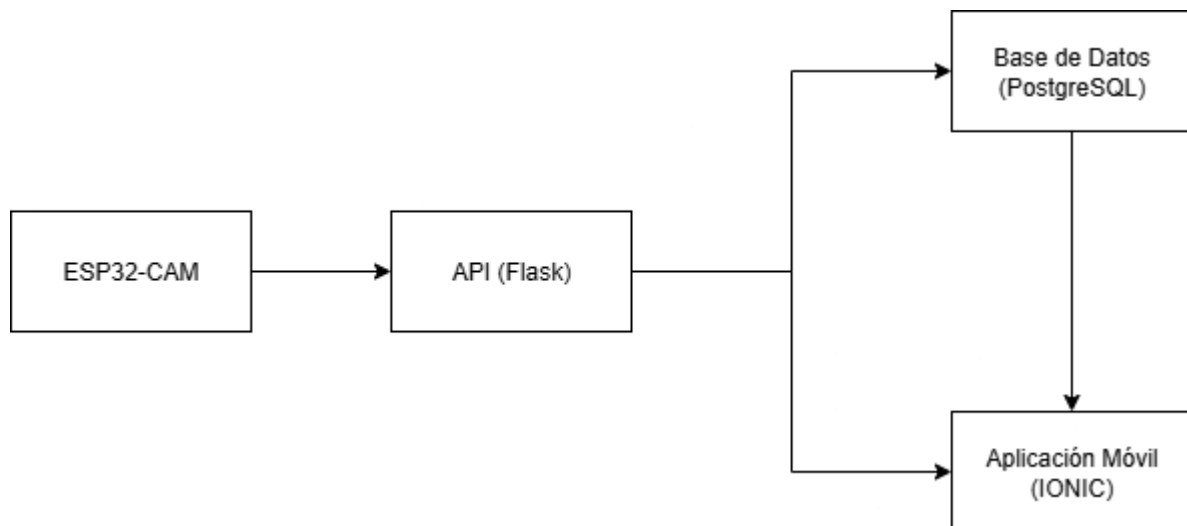


Ilustración 6 - Estructura del sistema

Configuración de Hardware

Debido a la cantidad de explicación que requiere la configuración de Arduino estará en el Anexo A.

Programación del Microcontrolador

El código está diseñado para un ESP32-CAM y tiene la función de capturar imágenes con su cámara y enviarlas a un servidor Flask a través de una solicitud HTTP POST. Lo primero es importar las librerías y configurar la cámara. Luego se captura la imagen y si falla, se imprime un mensaje de error y se termina la ejecución de ese ciclo. Después se prepara una solicitud HTTP POST con el header adecuado para enviar archivos y se envía la imagen al servidor Flask. Por último, espera 10 segundos antes de capturar la siguiente imagen y repetir el proceso.

Desarrollo del Backend en Python con Flask

El código comienza importando las librerías necesarias, como Flask para crear el servidor web, flask_cors.CORS para permitir solicitudes de diferentes orígenes (lo que es útil si el cliente y servidor están en dominios diferentes), y cv2 de OpenCV para procesamiento de imágenes. Además, se utiliza psycopg2 para interactuar con la base de datos PostgreSQL, y dotenv para cargar variables de entorno desde un archivo .env, lo que ayuda a mantener configuraciones y credenciales de forma segura.

Una vez importadas las librerías, se carga la configuración del servidor Flask, incluyendo las variables de entorno necesarias para configurar los puertos, las rutas del modelo de detección de personas y las credenciales de la base de datos. En cuanto a la configuración de CORS, se permite que el servidor acepte solicitudes de un cliente específico, en este caso, una aplicación Ionic, lo que es útil para evitar restricciones de seguridad de los navegadores cuando el cliente y servidor se ejecutan en dominios distintos.

En cuanto a la gestión de archivos, el servidor configura una carpeta llamada uploads donde se almacenarán las imágenes que se reciban. Si la carpeta no existe, se crea automáticamente. Además, se carga el modelo de detección de objetos MobileNet SSD utilizando OpenCV. Este modelo es capaz de detectar una serie de objetos en imágenes, y en este caso, está configurado para identificar si hay personas en las fotos recibidas.

Una de las rutas principales es /latests-images, que permite obtener las últimas 20 imágenes almacenadas en el servidor. El servidor consulta la base de datos para obtener los registros más recientes y los devuelve en formato JSON, incluyendo la URL de cada imagen, la fecha de subida y si se detectó una persona en la imagen.

Otra ruta importante es /uploads/<filename>, que permite servir las imágenes almacenadas en el servidor cuando se hace una solicitud GET. Esta ruta permite a los usuarios acceder a las imágenes mediante sus nombres de archivo.

Finalmente, la ruta /upload es donde la ESP32-CAM envía las imágenes capturadas. El servidor verifica que el archivo esté presente y sea válido, luego lo guarda en la carpeta de

subidas con un nombre único basado en la fecha y hora actual. Después, la imagen se procesa para detectar si hay personas utilizando el modelo MobileNet SSD cargado previamente. Si se detecta una persona en la imagen, el servidor guarda la información en la base de datos PostgreSQL, incluyendo el nombre del archivo, la ruta, la marca de tiempo y el estado de detección de personas.

Al final del código, el servidor Flask se inicia en el puerto configurado y comienza a escuchar solicitudes HTTP.

Desarrollo de la Base de Datos

La base de datos contará con una tabla que se encargará de almacenar las imágenes, así como su hora y si detecta actividad. Por comodidad, se usará el sistema de archivos del servidor y se almacenarán los metadatos de las imágenes en la base de datos. También contaremos con la tabla usuarios que almacenará datos del registro e inicio de sesión en la aplicación.

Tabla Fotos

Columna	Descripción
id	Identificador único
file_path	Ruta del archivo
upload_time	Fecha y hora del registro
activity_detected	Si se detectó actividad

Tabla 11 – Tabla Fotos

Tabla Usuarios

Columna	Descripción
id	Identificador único
name	Nombre del usuario
lastname	Apellido del usuario
email	Email del usuario
password	Contraseña del usuario
phone	Teléfono del usuario

role	Define si el usuario es cliente o administrador.
status	1 = Activo, 0 = Deshabilitado.
created_at	Fecha en la que se registró el usuario.

Tabla 12 – Tabla Usuarios

Desarrollo de la Aplicación Móvil

La aplicación móvil fue desarrollada como una Aplicación Web Progresiva (PWA) utilizando el framework Ionic, permitiendo su ejecución tanto en navegadores como en dispositivos móviles, manteniendo una experiencia de usuario fluida y adaptable. Esta aplicación cumple la función principal de notificar al usuario en tiempo real cuando se detecta la presencia de una persona en las imágenes captadas por el módulo ESP32-CAM, cuya información es procesada por el servidor backend desarrollado en Flask con soporte de OpenCV.

Además de recibir notificaciones, el usuario puede visualizar las imágenes capturadas por la cámara en una galería con paginación, lo que facilita la navegación y revisión del historial de eventos. Para proteger el acceso al sistema, se implementaron mecanismos de autenticación, incluyendo guards que restringen el ingreso únicamente a usuarios válidos.

La aplicación también incorpora una funcionalidad de llamada directa a emergencias, permitiendo al usuario actuar rápidamente en caso de una intrusión o evento sospechoso. Las notificaciones en tiempo real se gestionan mediante comunicación WebSocket, lo que garantiza una entrega inmediata y eficiente de los eventos relevantes.

4.3 Seguridad del Sistema

La seguridad constituye un aspecto fundamental en el diseño e implementación del sistema de vigilancia, ya que se trata del manejo de imágenes potencialmente sensibles y de la comunicación entre dispositivos distribuidos en una red. Por ello, se incorporaron distintas medidas para garantizar la integridad, autenticidad y confidencialidad de los datos transmitidos y accedidos por los distintos componentes del sistema.

Seguridad en la comunicación entre la cámara y el backend

Para evitar que cualquier dispositivo no autorizado acceda al endpoint del servidor y envíe datos falsos, se implementó un mecanismo de autenticación mediante clave secreta. Cada vez que el ESP32-CAM realiza una solicitud HTTP al servidor Flask, incluye en los encabezados una clave única que es verificada en el backend. De esta manera, solo se aceptan conexiones provenientes de la cámara configurada previamente, lo que previene intentos externos de inyección de datos o imágenes.

Seguridad en la aplicación móvil

La aplicación desarrollada con Ionic utiliza autenticación basada en tokens JWT (JSON Web Tokens). Cuando el usuario inicia sesión con sus credenciales, el backend genera un token que es utilizado en las siguientes solicitudes para verificar la identidad del usuario. Este mecanismo permite que el acceso a recursos protegidos —como la visualización del historial de imágenes— esté restringido únicamente a usuarios autenticados.

Además, se implementaron guards de navegación en la aplicación, los cuales impiden que se acceda a rutas internas si no existe un token válido en la sesión actual. Esto refuerza la seguridad desde el lado del cliente, evitando accesos indebidos mediante manipulación de la interfaz.

Almacenamiento de datos e imágenes

Las imágenes capturadas por la cámara son almacenadas en el servidor local de forma organizada y sin exposición pública directa. El acceso a estas imágenes se realiza exclusivamente a través de la API del backend, que requiere autenticación. De esta forma, no es posible acceder a las imágenes directamente mediante URL sin pasar por los controles de acceso definidos.

Buenas prácticas aplicadas

Además de los mecanismos mencionados, se aplicaron prácticas recomendadas como la validación de datos en los formularios, la limitación de intentos de inicio de sesión, y la exclusión de información sensible del lado del cliente. Esto contribuye a reducir riesgos comunes como inyecciones de código, ataques de fuerza bruta o exposición de credenciales.

4.4. Gestión del Proyecto

4.4.1. División de Tareas

El desarrollo del proyecto fue realizado en equipo, permitiendo una distribución eficiente de las tareas conforme a las habilidades y conocimientos de cada integrante. La organización del trabajo jugó un papel fundamental para cumplir con los objetivos propuestos y asegurar la calidad del sistema final.

1. Asignación de Responsabilidades

Durante las primeras etapas del proyecto, se identificaron los principales módulos del sistema y se procedió a dividir las tareas de forma equitativa y estratégica:

Programación del microcontrolador (ESP32)

La responsabilidad de programar el microcontrolador fue asumida por Nicolás Oris, utilizando el entorno de desarrollo Arduino IDE. Este trabajo incluyó el diseño de la lógica de adquisición de datos de sensores, el control de periféricos, la conexión a redes Wi-Fi y la comunicación con el servidor backend mediante protocolos HTTP y WebSocket según el diseño del sistema.

Desarrollo del Backend

También bajo responsabilidad de Nicolás Oris, se desarrolló el backend utilizando Flask. Esta parte del sistema permitió la recepción de datos desde el ESP32, su procesamiento, almacenamiento y la gestión de las notificaciones en tiempo real dirigidas hacia la aplicación móvil.

Desarrollo de la Aplicación Móvil

Ezequiel Barrios se encargó del desarrollo de la aplicación móvil utilizando el framework Ionic, permitiendo recibir notificaciones en tiempo real, mostrar el estado del sistema de vigilancia y brindando al usuario una interfaz amigable para la interacción con el sistema.

2. Colaboración y Trabajo Conjunto

Además de las tareas individuales, hubo componentes clave del proyecto que fueron desarrollados en conjunto:

Fase de diseño y planificación

Ambos integrantes participaron en la definición de los objetivos, la arquitectura del sistema, la selección de componentes y tecnologías, y en la elaboración del cronograma de actividades.

Pruebas de funcionamiento

Las pruebas de integración entre los distintos módulos (microcontrolador, backend y aplicación móvil) fueron realizadas de forma conjunta. Esto incluyó pruebas de conectividad, validación de envío y recepción de notificaciones y análisis de latencia.

Redacción de conclusiones y validación del sistema

Las conclusiones del trabajo, así como la evaluación final del sistema, fueron redactadas colaborativamente, asegurando que reflejaran de manera objetiva los resultados obtenidos y el cumplimiento de los objetivos.

3. Coordinación y Herramientas Utilizadas

Para mantener una coordinación efectiva, se utilizaron diversas herramientas de colaboración como ser:

- Git y GitHub, para el control de versiones del código fuente y el desarrollo colaborativo;
- Google Drive, para la redacción compartida de la tesis y almacenamiento de documentos técnicos;
- Draw.io, para la confección de diagramas de flujo;
- Mendeley, para la gestión de referencias bibliográficas;
- Reuniones periódicas, tanto presenciales como virtuales (mediante Discord y Meet), para revisar avances y resolver bloqueos técnicos.

4. Reflexión sobre el Trabajo en Equipo

La experiencia de trabajo colaborativo fue enriquecedora tanto en lo técnico como en lo personal. Permitió no solo la consolidación de conocimientos en áreas específicas (como programación, desarrollo móvil y backend), sino también el fortalecimiento de habilidades blandas como la comunicación, la organización y la resolución conjunta de problemas. El enfoque multidisciplinario y la buena coordinación fueron factores clave en la ejecución exitosa del proyecto

4.4.2. Diagrama de Gantt

A continuación, se presenta el diagrama de Gantt correspondiente a la planificación temporal del desarrollo del sistema de vigilancia propuesto. Este cronograma contempla las distintas etapas del proyecto, desde el análisis inicial hasta la finalización del mismo, incluyendo el desarrollo de los módulos hardware, backend y aplicación móvil, así como su integración, pruebas y cierre. La distribución de tareas fue realizada considerando una duración total de 103 días, con tiempos estimados para cada subtarea, de forma que se garantice una gestión eficiente de los recursos disponibles y el cumplimiento de los objetivos planteados.

		Nombre	Duracion	Inicio	Terminado
1		ANÁLISIS Y PLANIFICACIÓN	12 days?	03/02/25 08:00	18/02/25 17:00
2		Planteamiento del problema	4 days?	03/02/25 08:00	06/02/25 17:00
3		Revisión de trabajos previos	4 days?	06/02/25 08:00	11/02/25 17:00
4		Formulación de objetivos	3 days?	11/02/25 08:00	13/02/25 17:00
5		Recolección de requerimientos	3 days?	13/02/25 08:00	17/02/25 17:00
6		Planificación del cronograma	2 days?	17/02/25 08:00	18/02/25 17:00
7		DESARROLLO HARDWARE (ESP32-CAM)	10 days?	18/02/25 08:00	03/03/25 17:00
8		Configuración inicial de ESP32-CAM	5 days?	18/02/25 08:00	24/02/25 17:00
9		Captura y prueba de imágenes	4 days?	24/02/25 08:00	27/02/25 17:00
10		Conexión Wi-Fi y envío HTTP	3 days?	27/02/25 08:00	03/03/25 17:00
11		DESARROLLO BACKEND (FLASK + OPENCV)	17 days?	03/03/25 08:00	25/03/25 17:00
12		Estructura base del servidor Flask	4 days?	03/03/25 08:00	06/03/25 17:00
13		Recepción de imágenes vía POST	3 days?	06/03/25 08:00	10/03/25 17:00
14		Procesamiento con OpenCV (detección)	6 days?	10/03/25 08:00	17/03/25 17:00
15		Envío de notificaciones (Websocket)	5 days?	17/03/25 08:00	21/03/25 17:00
16		Gestión de imágenes (almacenamiento)	3 days?	21/03/25 08:00	25/03/25 17:00
17		DESARROLLO APP MÓVIL (IONIC)	19 days?	25/03/25 08:00	18/04/25 17:00
18		Diseño de interfaces (login, galería)	5 days?	25/03/25 08:00	31/03/25 17:00
19		Conexión con backend	6 days?	31/03/25 08:00	07/04/25 17:00
20		Recepción de notificaciones	4 days?	07/04/25 08:00	10/04/25 17:00
21		Visualización de imágenes + paginado	4 days?	10/04/25 08:00	15/04/25 17:00
22		Función de llamada al 911	4 days?	15/04/25 08:00	18/04/25 17:00
23		PRUEBAS E INTEGRACIÓN	29 days?	18/04/25 08:00	28/05/25 17:00
24		Integración de componentes	9 days?	18/04/25 08:00	30/04/25 17:00
25		Pruebas funcionales	8 days?	30/04/25 08:00	09/05/25 17:00
26		Pruebas de usuario (experiencia)	11 days?	09/05/25 08:00	23/05/25 17:00
27		Documentación de resultados de pruebas	4 days?	23/05/25 08:00	28/05/25 17:00
28		FINALIZACIÓN	16 days?	28/05/25 08:00	18/06/25 17:00
29		Resultados y Conclusiones	6 days?	28/05/25 08:00	04/06/25 17:00
30		Finalización del proyecto	11 days?	04/06/25 08:00	18/06/25 17:00

Ilustración 7 - Gestión del Proyecto

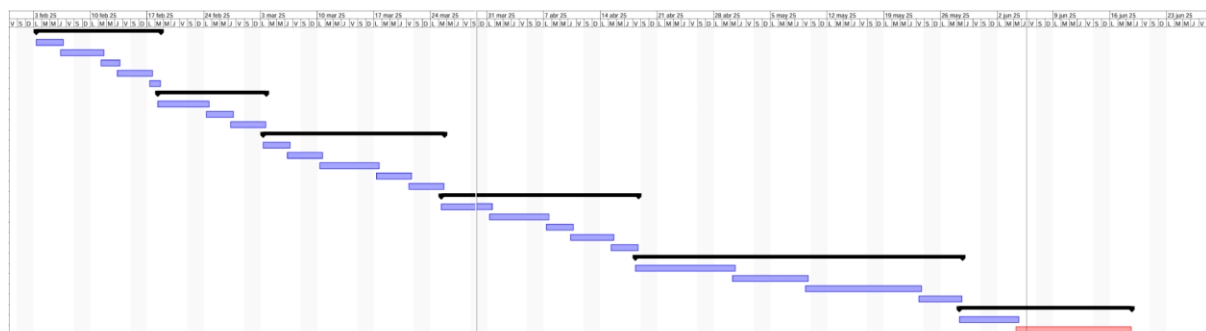


Ilustración 8 - Gestión del Proyecto

4.5. Cumplimiento de Objetivos

El desarrollo e implementación del Sistema de Vigilancia basado en Microcontrolador con Notificaciones Móviles en Tiempo Real y Enfoque de Bajo Costo ha permitido alcanzar satisfactoriamente tanto el objetivo general como los objetivos específicos propuestos en este proyecto.

En primer lugar, se logró desarrollar un sistema funcional en tiempo real utilizando el microcontrolador ESP32-CAM, el cual integra cámara y conectividad Wi-Fi, permitiendo la captura y transmisión de imágenes de forma autónoma. Gracias al uso exclusivo de componentes de bajo costo y fácil acceso, así como software de código abierto, el costo total del sistema se mantuvo por debajo de los 200 USD, cumpliendo con el enfoque económico planteado y demostrando su viabilidad en entornos domésticos o con recursos limitados.

Respecto a los objetivos específicos, se cumplieron de la siguiente manera:

Se diseñó la arquitectura del sistema utilizando el ESP32-CAM como núcleo, integrando su módulo de cámara y conectividad a internet con componentes económicos.

Se programó el microcontrolador para detectar movimiento y capturar imágenes, las cuales son transmitidas en tiempo real a un servidor web.

Se implementó una API REST, la cual recibe las capturas, ejecuta la detección de personas mediante algoritmos de visión por computadora, y envía notificaciones inmediatas al usuario a través de dispositivos móviles.

Se desarrolló una aplicación web progresiva (PWA) que permite a los usuarios visualizar las imágenes capturadas, acceder al historial de eventos registrados y gestionar el sistema de forma remota, mejorando la accesibilidad y la experiencia de uso.

Finalmente, se evaluó el desempeño del sistema mediante pruebas en distintos entornos, analizando variables como conectividad, tiempo de respuesta y confiabilidad de las detecciones. Los resultados demostraron una buena relación costo-beneficio, así como una respuesta adecuada para su aplicación en escenarios reales.

En conclusión, el sistema desarrollado cumple con los lineamientos propuestos, ofreciendo una solución efectiva, accesible y replicable para necesidades básicas de vigilancia, y posicionándose como una alternativa viable frente a opciones comerciales más costosas.

5. Análisis de Factibilidad

5.1 Recursos Humanos

El costo estimado de recursos humanos para el desarrollo del sistema, basados en el diagrama de Gantt, es de \$18.128.659,20 pesos argentinos.

Rol	Cantidad	Costo por hora	Total de horas	Subtotal
Analista Funcional	1	\$12.000	824	\$9.888.000,00
Analista Programador	1	\$10.000,80	824	\$8.240.659,20
Total				\$18.128.659,20

Tabla 13 – Costos de Recursos Humanos

Honorarios obtenidos de COPAIPA[37].

5.2 Hardware

Para la puesta en producción del sistema del sistema de vigilancia no hace falta invertir demasiado puesto que los componentes necesarios no requieren que sean de alta gama.

Requerimientos de Hardware

A continuación se muestran los requerimientos de hardware con sus respectivos costos.

Dispositivo	Cantidad	Voltaje	Descripción	Costo
ESP32-CAM[38]	1	3.3V - 5V	Captura imágenes y las envía al backend en Flask.	\$22.402,00
Módulo FTDI[38]	1	5V	Se usa para programar el ESP32-CAM.	
Módulo WiFi ESP32-CAM[38]	1	3.3V - 5V	Conecta el ESP32-CAM al servidor Flask	

			para transmitir imágenes en tiempo real.	
Servidor Flask Intel Celeron (o AMD equivalente), 2GB RAM, 320GB almacenamiento [39]	1	12V	Ejecuta el backend en Flask, recibe imágenes del ESP32-CAM y las almacena y reenvía a la PWA.	\$150.000,00
Cable Usb A Micro Usb 3 Mts [40]	1	5V	Conecta el ESP32-CAM con el servidor	\$4.160,00
Prolongador Usb 2.0 Activo Reforzado Y Filtro Extensor 10mts [41]	1	5V	Permite extender el cable que conecta el ESP32-CAM con el servidor	\$17.699,00
Total				\$194.261,00

Tabla 14 – Costos de Hardware

5.3 Software

A continuación, se presenta el software requerido para poner en marcha este sistema.

Requerimientos de Software

Los requerimientos de software son los siguientes.

Software	Costo
Python 3	\$0,00
Git	\$0,00
Arduino IDE	\$0,00
NodeJS	\$0,00

Visual Studio Code	\$0,00
Ionic	\$0,00
Total	\$0,00

Tabla 15 – Costos de Software

5.4 Sumatoria de Costos

Con los costos anteriores se puede calcular el costo final del proyecto el cual es de \$18.128.659,20 pesos argentinos.

Costo	Monto
Recursos Humanos	\$18.128.659,20
Hardware	\$194.261,00
Software	\$0,00
Total	\$18.128.659,20

Tabla 16 – Sumatoria de Costos

El costo total de implementación del presente proyecto es de \$194.261,00 pesos argentinos, dado que únicamente se requiere la adquisición del hardware necesario para su puesta en marcha. Esta solución puede clasificarse como de bajo costo, especialmente si se la compara con alternativas ofrecidas por proveedores locales, como es el caso de la empresa Detectar Seguridad & Informática, cuya propuesta para satisfacer la misma necesidad asciende a \$406.800,00 pesos argentinos. [36].

5.5 Factibilidad Legal

La factibilidad legal de un proyecto nos permite confirmar que se cumpla con los requisitos jurídicos necesarios para su funcionamiento y además garantizar que el uso del sistema no violara ninguno de los mismos. Realizar este estudio es esencial ya que de este depende que se pueda aplicar o no.

Para el presente proyecto se siguió el marco legal existente, es decir la Ley N° 25.326 que es la llamada Ley de Protección de Datos Personales, la cual establece normas para el tratamiento de datos personales que en este caso serían las fotos que toma el microcontrolador ESP32-CAM.

Si bien el sistema de vigilancia propuesto en este proyecto está originalmente orientado al uso privado en viviendas particulares, su aplicación no se limita exclusivamente a este ámbito. También resulta viable su implementación en otros entornos, tanto del sector público (como instituciones estatales) como del sector privado (como PyMEs), siempre que su utilización se mantenga dentro de espacios controlados por el titular del sistema.

El sistema no está destinado a una implementación comercial ni masiva, y su funcionamiento está restringido a contextos donde se pueda garantizar el control sobre su operación. Bajo estas condiciones, no se vulneran derechos de terceros ni se invade la privacidad de personas ajenas al entorno autorizado.

En función de lo expuesto, se concluye que el sistema es legalmente factible bajo el marco normativo argentino, tanto para su uso en ámbitos privados como institucionales, siempre que se respeten los principios legales vigentes y se garantice la adecuada protección de los datos personales capturados.

Observación

En caso de que el sistema sea implementado en un escenario real, se contemplan los aspectos legales establecidos en la Ley N.º 25.326 de Protección de los Datos Personales, vigente en la República Argentina. Esta legislación establece que toda información vinculada a personas físicas identificadas o identificables constituye un dato personal, y por lo tanto, debe ser protegida adecuadamente. Las imágenes capturadas por el sistema pueden ser consideradas como tales si contienen rostros u otros elementos identificatorios. En consecuencia, se recomienda que el uso del sistema se mantenga restringido al ámbito privado, sin apuntar hacia espacios públicos, veredas o propiedades ajenas, salvo que exista el consentimiento explícito de las personas potencialmente afectadas.

5.6 Factibilidad Técnica

El proyecto es técnicamente factible, ya que se apoya en tecnologías ampliamente conocidas, de bajo costo y con una gran comunidad de soporte. El uso del módulo ESP32-CAM permite implementar un sistema de captura de imágenes eficiente sin requerir hardware complejo ni costoso. Por su parte, el servidor backend desarrollado con Flask en Python permite manejar la recepción de imágenes, el procesamiento mediante OpenCV y el envío de notificaciones en tiempo real, lo cual puede realizarse incluso en computadoras de prestaciones modestas.

La aplicación móvil está desarrollada en Ionic, un framework multiplataforma que permite una rápida implementación para sistemas Android, lo que amplía la accesibilidad del sistema. Además, las tecnologías empleadas son compatibles entre sí y no presentan

requerimientos técnicos elevados, lo que garantiza la viabilidad de su desarrollo e integración dentro de un entorno controlado y doméstico.

5.7 Factibilidad Operativa

Desde el punto de vista operativo, el sistema puede ser instalado y utilizado por usuarios finales con conocimientos básicos, dado que está diseñado para operar de forma automatizada y con una interfaz móvil sencilla. La instalación del hardware en un entorno doméstico no requiere conocimientos técnicos avanzados, y la interfaz móvil permite visualizar imágenes, recibir alertas y tomar decisiones como llamar a un número de emergencia en un solo clic.

Además, el sistema está pensado como una solución no masiva ni comercial, orientada a la vigilancia hogareña o de pequeños espacios privados, lo cual reduce la complejidad operativa y los requerimientos legales asociados. La dependencia mínima de internet (solo necesaria para el envío de datos y notificaciones) y el bajo consumo energético del hardware también contribuyen positivamente a su operatividad.

6. Análisis de Riesgos

El análisis de riesgos es una etapa fundamental en el desarrollo de cualquier sistema tecnológico, ya que permite identificar posibles amenazas que podrían afectar su funcionamiento, seguridad o sostenibilidad, así como establecer estrategias de mitigación. A continuación, se presentan los principales riesgos identificados durante el diseño, desarrollo e implementación del sistema de vigilancia, clasificados según su naturaleza.

6.1 Matriz de Riesgos

Durante el desarrollo e implementación de cualquier sistema tecnológico, es fundamental identificar y evaluar los posibles riesgos que puedan afectar su funcionamiento, seguridad o viabilidad. El análisis de riesgos permite anticiparse a problemas potenciales y establecer estrategias de mitigación que reduzcan su impacto.

En este proyecto se realizó una identificación de los principales riesgos asociados al sistema de vigilancia basado en microcontrolador, considerando tanto aspectos técnicos como operativos. Para su evaluación se utilizó una matriz de riesgos, donde cada evento fue clasificado según su probabilidad de ocurrencia (alta, media o baja) y el impacto que tendría en el sistema (alto, medio o bajo). A partir de esta combinación, se determinó un nivel de riesgo global (bajo, medio o alto) y se propusieron medidas para mitigarlo o reducir su efecto en caso de ocurrencia.

La matriz de riesgos se estructura mediante una escala que combina dos dimensiones:

- Probabilidad (P): Qué tan probable es que el riesgo ocurra. Se clasifica como Baja (1), Media (2) o Alta (3).
- Impacto (I): Qué consecuencias tendría ese riesgo para el proyecto. También se clasifica como Bajo (1), Medio (2) o Alto (3).

Multiplicando estos dos factores ($P \times I$), se obtiene un nivel de riesgo que puede clasificarse como:

Nivel de Riesgo	Rango	Categoría
1 – 2	Bajo	Riesgo aceptable
3 – 4	Medio	Riesgo moderado
6 – 9	Alto	Riesgo crítico

Tabla 17 – Niveles de Riesgo

A continuación, se presenta la matriz de riesgos correspondiente:

Impacto / Probabilidad	Baja(1)	Media(2)	Alta(3)
Alto(3)	3(Medio)	6(Alto)	9(Alto)
Medio(2)	2(Bajo)	4(Medio)	6(Alto)
Bajo(1)	1(Bajo)	2(Bajo)	3(Medio)

Tabla 18 – Matriz de Riesgo

6.2 Identificación de Riesgos

Riesgo	Probabilidad	Impacto	Medida de Mitigación
Inestabilidad del ESP32-CAM	Media	Alta	Implementar reinicio automático y control de errores
Pérdida de conexión/red	Alta	Media	Reintentos automáticos, compresión de imágenes
Acceso no autorizado al backend	Media	Alta	Uso de clave secreta y autenticación JWT
Interceptación de datos en tránsito	Baja	Alta	Sugerencia de implementar HTTPS
Sobrecarga del servidor backend	Media	Alta	Optimización del flujo de datos y pruebas de carga
Fallos durante la instalación	Media	Media	Manual de instalación detallado
Dificultades del usuario con la interfaz	Media	Baja	Diseño PWA intuitivo y simplificado
Daño físico del ESP32-CAM o fuente	Media	Media	Elección de hardware económico y fácilmente reemplazable

Tabla 19 – Identificación de Riesgos

6.3 Análisis FODA

El análisis FODA (Fortalezas, Oportunidades, Debilidades y Amenazas) es una herramienta estratégica que permite evaluar de forma integral los factores internos y externos que influyen en el desarrollo de un proyecto. Esta metodología facilita la toma de decisiones al identificar aspectos positivos que deben potenciarse y elementos negativos que deben corregirse o gestionarse con precaución.

En el contexto de este sistema de vigilancia basado en microcontrolador, el análisis FODA ayuda a comprender el posicionamiento del proyecto respecto a sus capacidades tecnológicas, su entorno de aplicación, sus limitaciones y sus posibilidades de mejora y expansión futura. A continuación, se presenta el análisis FODA correspondiente al sistema desarrollado:

Interno	Fortalezas	Debilidades
	1. Bajo costo de implementación, accesible para usuarios domésticos. 2. Utilización de tecnologías modernas (WebSockets, PWA, JWT). 3. Facilidad de instalación y operación básica. 4. Modularidad para futuras ampliaciones. 5. Notificaciones en tiempo real y funcionalidad multiplataforma.	1. Dependencia de la calidad de conexión Wi-Fi. 2. Limitaciones en resolución y procesamiento del ESP32-CAM. 3. Sistema de seguridad básico (JWT y clave compartida, sin HTTPS). 4. Requiere conocimientos técnicos mínimos para instalación inicial.
Externo	Oportunidades	Amenazas
	1. Escalabilidad hacia soluciones para pequeños comercios o comunidades. 2. Integración futura con plataformas de domótica (Smart Home). 3. Uso de almacenamiento en la nube y paneles administrativos. 4. Mejora en la calidad de imagen con mejor hardware. 5. Aplicación de IA para análisis	1. Posibles vulnerabilidades de seguridad si no se protege la red. 2. Obsolescencia de componentes de bajo costo o discontinuidad del soporte. 3. Competencia con soluciones comerciales más sofisticadas. 4. Cambios en APIs o frameworks (Ionic, Flask, etc.) que generen incompatibilidades.

	más avanzados (detección de intrusos, reconocimiento facial, etc.).	
--	---	--

Tabla 20 – Análisis FODA

7. Resultados

7.1. Funcionamiento del Sistema

En esta sección se describe el funcionamiento completo e integrado del sistema de vigilancia desarrollado, abarcando tanto el componente de hardware (ESP32-CAM), como el backend (servidor Flask con OpenCV) y la aplicación móvil (Ionic).

El proceso comienza con el encendido del módulo ESP32-CAM, el cual inicializa la cámara y establece conexión con una red Wi-Fi configurada previamente. Una vez conectado, el dispositivo captura imágenes periódicamente o en respuesta a un evento, y las envía mediante solicitudes HTTP POST al servidor Flask.

El servidor Flask recibe estas imágenes, las almacena y las procesa utilizando la biblioteca OpenCV. El objetivo del procesamiento es detectar la presencia de personas en las imágenes. En caso de una detección positiva, el sistema genera una notificación en tiempo real que se transmite a la aplicación móvil utilizando un canal WebSocket previamente establecido.

La aplicación móvil, desarrollada como una Aplicación Web Progresiva (PWA) con Ionic, está diseñada para recibir estas notificaciones instantáneamente. El usuario es alertado visualmente sobre la posible intrusión, y desde la propia aplicación puede visualizar una galería con todas las imágenes captadas, organizada de manera paginada. Además, se incluye una funcionalidad crítica que permite realizar una llamada directa a los servicios de emergencia desde la misma interfaz.

Este funcionamiento integrado demuestra la viabilidad técnica del sistema propuesto y su utilidad como solución económica y funcional para la vigilancia doméstica.

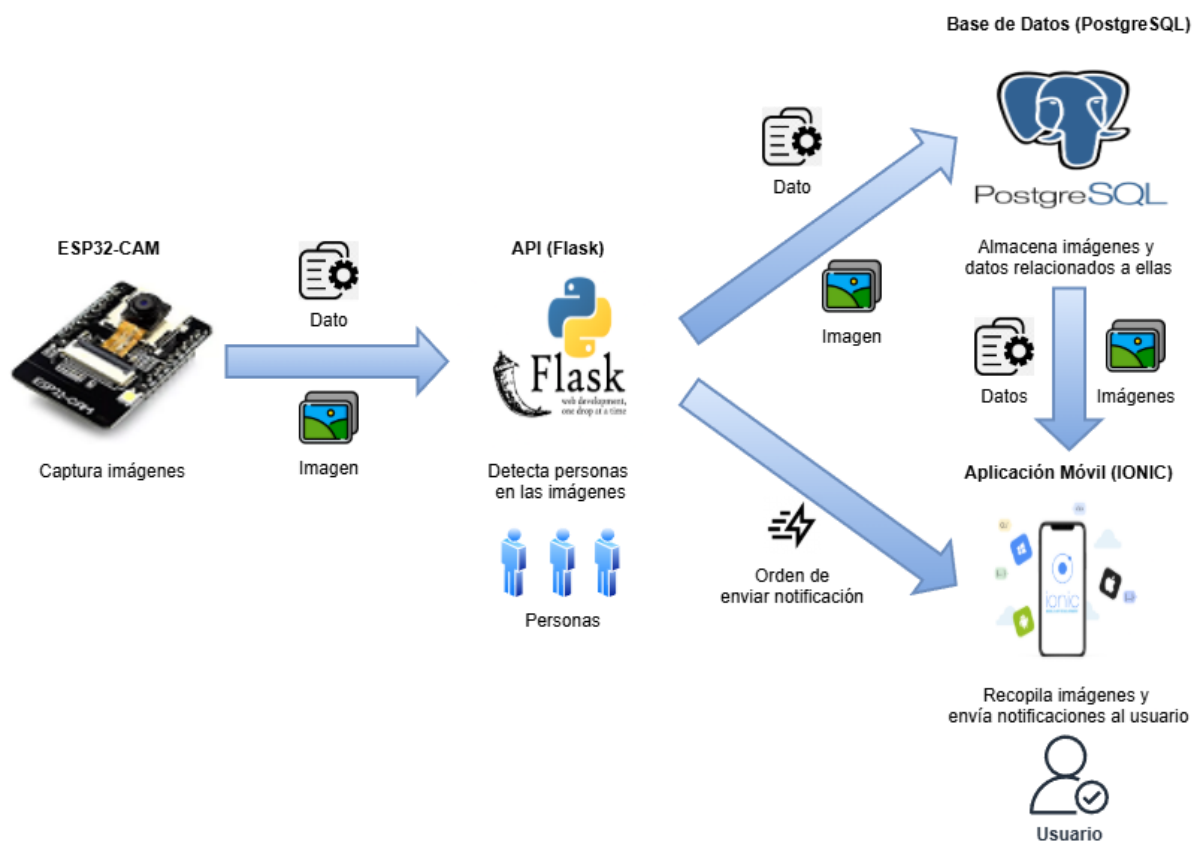


Ilustración 9 - Representación del Funcionamiento del Sistema

7.2. Pruebas y Validación

Pruebas de Funcionamiento

Detección de personas (OpenCV)

Partiendo de una muestra de 100 repeticiones, se obtuvieron los siguientes resultados:

En 92 ocasiones, se obtuvo una detección correcta de la persona, mientras que en 8 ocasiones no fue detectada. Se esperaba que la detección fuese correcta en al menos 90 ocasiones.

En 3 ocasiones se obtuvo una falsa detección de una persona. Se esperaba que se detectara erróneamente una persona en 5 o menos ocasiones.

Prueba	Resultado Esperado	Resultado Obtenido	Observaciones
Precisión detección	$\geq 90\%$	92%	Detectó correctamente en 92 ocasiones
Falsos positivos	$\leq 5\%$	3%	Solo 3 falsas detecciones

Tabla 21 – Prueba de Detección

Se concluye que la detección de personas es precisa.

Envío y recepción de imágenes

Partiendo de una muestra de 100 repeticiones, se obtuvieron los siguientes resultados:

Se obtuvo en promedio una latencia de envío de 0.2mseg, mientras que se esperaba una latencia de envío menor o igual a 1 seg.

Prueba	Resultado Esperado	Resultado Obtenido	Observaciones
Latencia de envío y recepción	≤ 1 seg	0.2 mseg	Tiempo aceptable

Tabla 22 – Prueba de envío y recepción de imágenes

Se concluye que el tiempo de envío y recepción de imágenes es aceptable.

Notificaciones en tiempo real

Partiendo de una muestra de 50 repeticiones, se obtuvieron los siguientes resultados:

Prueba	Resultado Esperado	Resultado Obtenido	Observaciones
Latencia de notificación	≤ 5 seg	0.5 seg	Tiempo correcto

Tabla 23 – Prueba de notificación

Se concluye que el tiempo de notificación en tiempo real es correcto.

Análisis de Resultados

Luego de la implementación y puesta en marcha del sistema, se llevaron a cabo diversas pruebas destinadas a evaluar su funcionamiento, efectividad y capacidad de respuesta ante distintas condiciones de uso. Estas pruebas se realizaron en entornos controlados y reales, simulando escenarios comunes de vigilancia en el hogar, como la detección de personas en pasillos, entradas o habitaciones con diferentes niveles de iluminación.

Durante las pruebas, se observó que el sistema fue capaz de identificar con éxito la presencia de personas en la mayoría de los casos. El modelo de detección utilizado, junto con el procesamiento de imágenes mediante OpenCV en el servidor Flask, permitió una respuesta adecuada dentro de márgenes aceptables de tiempo. Una vez realizada la detección, el sistema envió notificaciones en tiempo real a la aplicación móvil, cumpliendo con el objetivo de alertar al usuario de forma inmediata.

La recepción de imágenes por parte del servidor se realizó sin interrupciones relevantes, y la comunicación entre los distintos componentes del sistema (ESP32-CAM, servidor Flask y app móvil) demostró ser estable bajo una conexión de red doméstica estándar. El almacenamiento y la paginación de imágenes funcionaron correctamente, permitiendo al usuario consultar eventos pasados de forma ordenada.

En cuanto a la experiencia de uso, la aplicación móvil fue percibida como intuitiva y accesible, y las funcionalidades clave —como el acceso al historial de imágenes y la llamada rápida a emergencias— respondieron adecuadamente durante todas las pruebas realizadas.

Finalmente, cabe destacar que el sistema logró cumplir con sus objetivos propuestos no solo desde lo funcional, sino también desde lo económico. La elección de componentes de bajo costo como el ESP32-CAM, junto con tecnologías de desarrollo open source, permitió construir una solución accesible y viable, sin comprometer el rendimiento esperado.

7.3. Conclusión

El desarrollo del sistema de vigilancia con notificación en tiempo real demostró ser una solución viable, funcional y de bajo costo para entornos domésticos. A través de la integración de un módulo ESP32-CAM, un servidor backend desarrollado en Python con Flask y OpenCV, y una aplicación móvil basada en el framework Ionic, se logró implementar un sistema capaz de capturar imágenes, detectar presencia humana y enviar alertas instantáneas al usuario final.

Este proyecto se destacó no solo por su económica implementación, sino también por su accesibilidad tecnológica y facilidad de uso, permitiendo que incluso usuarios sin conocimientos técnicos puedan beneficiarse de una solución de seguridad eficiente. El uso de tecnologías modernas como WebSockets para notificaciones en tiempo real, junto con una

arquitectura basada en PWA, contribuyó significativamente a ofrecer una experiencia de usuario fluida y compatible con múltiples plataformas.

Durante el proceso de desarrollo, surgieron diversos desafíos técnicos que debieron ser abordados para garantizar el correcto funcionamiento del sistema. Uno de los principales inconvenientes fue la inestabilidad inicial de la transmisión de imágenes desde el ESP32-CAM al servidor, lo cual fue solucionado mediante ajustes en la frecuencia de captura, el manejo de conexiones HTTP y el control de errores en la red. También se presentaron dificultades en la integración de las notificaciones en tiempo real, que fueron superadas mediante el uso de sockets persistentes y la correcta configuración del servidor Flask. Además, se trabajó en la mejora del rendimiento del backend, optimizando el procesamiento de imágenes y ajustando el flujo de trabajo entre captura, análisis y notificación.

A pesar de sus limitaciones, el sistema logró cumplir con los objetivos planteados. No obstante, se identificó que la calidad de las imágenes capturadas está sujeta a las limitaciones del hardware utilizado, particularmente del módulo ESP32-CAM. Por tanto, el empleo de microcontroladores o cámaras con mejores capacidades técnicas en futuras versiones permitiría mejorar significativamente la resolución, claridad y fiabilidad del análisis visual.

La validación mediante métricas específicas, como la precisión en la detección de personas y el tiempo de respuesta del sistema, confirmó que la solución cumple con los objetivos planteados al inicio del proyecto. De este modo, el sistema constituye una base sólida para futuras mejoras, como la incorporación de inteligencia artificial más avanzada, almacenamiento en la nube o funcionalidades orientadas a la domótica y el hogar inteligente.

En definitiva, este trabajo refleja cómo el uso inteligente y estratégico de herramientas tecnológicas de bajo costo puede convertirse en soluciones concretas con impacto real, reafirmando el valor de la ingeniería aplicada a mejorar la seguridad y el bienestar de las personas en sus espacios cotidianos.

7.4. Futuras Líneas de Investigación

A partir de los resultados obtenidos y del alcance definido en esta tesis, se identifican diversas posibilidades para continuar desarrollando y mejorando el sistema de vigilancia con notificación en tiempo real. Estas futuras líneas de investigación permitirían ampliar las funcionalidades, optimizar el rendimiento y adaptar el sistema a nuevos escenarios de uso.

Una primera línea de mejora consiste en la integración de sensores adicionales, como sensores de movimiento (PIR), sensores ultrasónicos o magnéticos, que podrían trabajar de forma complementaria con la detección por cámara. Esto permitiría reducir la cantidad de

imágenes procesadas, disminuir los falsos positivos y optimizar el consumo de recursos, especialmente en escenarios donde la cámara no es necesaria de forma continua.

Otra dirección prometedora es la migración del sistema hacia una infraestructura basada en la nube. Esto permitiría centralizar el almacenamiento de imágenes, gestionar múltiples dispositivos desde diferentes ubicaciones, y acceder al sistema desde cualquier lugar con conexión a internet. Además, la nube facilitaría la escalabilidad del sistema, tanto en términos de almacenamiento como de procesamiento y monitoreo en tiempo real.

También podría considerarse la incorporación de técnicas más avanzadas de inteligencia artificial y aprendizaje automático, como el uso de modelos de detección de objetos entrenados específicamente para ambientes domésticos, que mejorarían la precisión en la identificación de personas y permitirían reconocer comportamientos anómalos o patrones inusuales.

En cuanto al hardware, una línea clara de investigación y desarrollo apunta a sustituir el ESP32-CAM por microcontroladores o cámaras más avanzadas que permitan capturar imágenes en mayor resolución y con mejor rendimiento en condiciones de baja iluminación. Esto incrementaría la calidad de las detecciones y abriría la puerta a análisis visuales más complejos, incluso con técnicas de reconocimiento facial o de contexto. Del mismo modo, se considera la optimización del equipo de procesamiento, particularmente la computadora que actúa como servidor Flask, reemplazándola por una unidad más robusta, con el fin de mejorar la velocidad de procesamiento, la estabilidad del sistema y la escalabilidad de la solución.

Otra mejora significativa sería la incorporación del uso de contenedores mediante tecnologías como Docker, lo cual permitiría simplificar el despliegue del sistema, facilitar la instalación en distintos entornos, garantizar la consistencia entre ambientes de desarrollo y producción, y reducir problemas de compatibilidad entre dependencias. Esto contribuiría a una distribución más eficiente y accesible del sistema, especialmente para usuarios o desarrolladores con menos experiencia técnica.

Por otro lado, una evolución lógica del sistema es su integración con plataformas de domótica o automatización del hogar, lo que abriría la posibilidad de activar alarmas, encender luces o bloquear accesos en función de los eventos detectados por el sistema de vigilancia.

Finalmente, sería interesante desarrollar una versión multiusuario y multi-dispositivo, con distintos niveles de acceso y control, orientada a pequeños comercios o instalaciones compartidas, manteniendo el bajo costo como característica principal.

Estas líneas de investigación no solo permitirían enriquecer el sistema actual, sino también extender su aplicación a nuevos contextos, manteniendo el enfoque en la accesibilidad, la simplicidad y la eficiencia de los recursos tecnológicos empleados.

Referencias Bibliográficas

- [1] M. A. Mazidi, S. Naimi, and S. Naimi, “The AVR Microcontroller and Embedded Systems: Using Assembly and C,” 2014, Accedido: Apr. 19, 2025. [Online]. Disponible: www.pearsoned.co.uk
- [2] “ESP-IDF Programming Guide - ESP32 - — ESP-IDF Programming Guide v5.4.1 documentation.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html>
- [3] “Getting Started with Arduino IDE 2 | Arduino Documentation.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2/>
- [4] L. J. Fennelly, “Effective Physical Security - Google Libros.” Accedido: Apr. 19, 2025. [Online]. Disponible: [https://books.google.com.ar/books?hl=es&lr=&id=v0y0DAAAQBAJ&oi=fnd&pg=PP1&dq=Fennelly,+L.+J.+\(2016\).+Effective+Physical+Security.+Butterworth-Heinemann.&ots=yqesdP8OAu&sig=qmMjjMIU2UEchfRL27J5YHzZH9Q#v=onepage&q=Fennelly%2C%20L.%20J.%20\(2016\).%20Effective%20Physical%20Security.%20Butterworth-Heinemann.&f=false](https://books.google.com.ar/books?hl=es&lr=&id=v0y0DAAAQBAJ&oi=fnd&pg=PP1&dq=Fennelly,+L.+J.+(2016).+Effective+Physical+Security.+Butterworth-Heinemann.&ots=yqesdP8OAu&sig=qmMjjMIU2UEchfRL27J5YHzZH9Q#v=onepage&q=Fennelly%2C%20L.%20J.%20(2016).%20Effective%20Physical%20Security.%20Butterworth-Heinemann.&f=false)
- [5] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey”, doi: 10.1016/j.comnet.2010.05.010.
- [6] Michael. Miller, “The Internet of things : how smart TVs, smart cars, smart homes, and smart cities are changing the world,” p. 319, 2015.
- [7] “Open-Source UI Toolkit to Create Your Own Mobile Apps.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://ionicframework.com/docs>
- [8] “Introducción a las aplicaciones web progresivas (PWA) - Microsoft Edge Developer documentation | Microsoft Learn.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://learn.microsoft.com/es-es/microsoft-edge/progressive-web-apps-chromium/>
- [9] Abraham. Silberschatz, Greg. Gagne, and P. B. . Galvin, “Operating System Concepts, Enhanced Edition,” 2018, Accedido: Apr. 19, 2025. [Online]. Disponible: <https://www.wiley.com/en-us/Operating+System+Concepts%2C+10th+Edition-p-9781119320913>
- [10] A. S. Tannenbaum, “MODERN OPERATING SYSTEMS FOURTH EDITION”.
- [11] R. S. Pressman, “Software Engineering: A Practitioner’s Approach”, Accedido: Apr. 19, 2025. [Online]. Disponible: www.mhhe.com/pressman.

- [12] C. J. . Date, “Database in Depth: Relational Theory for Practitioners - C.J. Date - Google Libros.” Accedido: Apr. 19, 2025. [Online]. Disponible: https://books.google.com.ar/books?id=sZc7B6wnmNcC&printsec=copyright&redir_esc=y#v=onepage&q&f=false
- [13] C. J. . Date and S. L. M. Ruiz Faudón, “Introducción a los sistemas de bases de datos,” p. 936, 2001.
- [14] “PostgreSQL: The world’s most advanced open source database.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://www.postgresql.org/>
- [15] I. Sommerville *et al.*, “SOFTWARE ENGINEERING Ninth Edition,” 2011.
- [16] “Welcome to Flask — Flask Documentation (3.1.x).” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://flask.palletsprojects.com/en/stable/>
- [17] A. S. , & V. S. M. Tanenbaum, “DISTRIBUTED SYSTEMS Second Edition”, Accedido: Apr. 19, 2025. [Online]. Disponible: www.minix3.org.This
- [18] J. F. , & R. K. W. Kurose, “Computer Networking: a Top Down Approach.” Accedido: Apr. 19, 2025. [Online]. Disponible: https://gaia.cs.umass.edu/kurose_ross/online_lectures.htm
- [19] R. T. , et al. Fielding, “RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://datatracker.ietf.org/doc/html/rfc2616>
- [20] I. Fette and A. Melnikov, “The WebSocket Protocol,” Dec. 2011, doi: 10.17487/RFC6455.
- [21] Steve. Souders, “High performance web sites : essential knowledge for frontend engineers,” p. 146, 2007, Accedido: Apr. 19, 2025. [Online]. Disponible: <https://www.oreilly.com/library/view/high-performance-web/9780596529307/>
- [22] G. Hohpe and B. Woolf, “Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions (Addison-Wesley Signature Series (Fowler)) : Gregor, Hohpe, Bobby, Woolf.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://www.amazon.es/Enterprise-Integration-Patterns-Designing-Deploying/dp/0321200683>
- [23] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature* 2015 521:7553, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
- [24] N. Aramayo, “INTELIGENCIA ARTIFICIAL – Machine Learning – Deep Learning conceptos generales”, presentación de PowerPoint, Cátedra de Sistemas Inteligentes, Universidad Católica de Salta, Salta, Argentina, agosto 2021.

- [25] Y. Bengio, “Learning deep architectures for AI,” *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–27, 2009, doi: 10.1561/22000000006.
- [26] “AI Pioneers John Hopfield and Geoffrey Hinton Win Nobel Prize in Physics - WSJ.” Accedido: Apr. 19, 2025. [Online]. Disponible: https://www.wsj.com/science/nobel-prize-in-physics-awarded-to-duo-for-machine-learning-c9e4b0d7?utm_source=chatgpt.com
- [27] “OpenCV - Open Computer Vision Library.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://opencv.org/>
- [28] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017, Accedido: Apr. 21, 2025. [Online]. Disponible: <https://arxiv.org/abs/1704.04861v1>
- [29] “¿Qué es la metodología Kanban y cómo funciona? [2025] • Asana.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://asana.com/es/resources/what-is-kanban>
- [30] B. John and B. John, “JSON Web Token (JWT),” *tools.ietf.org*, Accedido: Apr. 19, 2025. [Online]. Disponible: <https://tools.ietf.org/html/rfc7519>
- [31] “ISO - ISO 9000 family — Quality management.” Accedido: Apr. 19, 2025. [Online]. Disponible: <https://www.iso.org/standards/popular/iso-9000-family>
- [32] “Cámara con reconocimiento inteligente | Prosegur Argentina.” Accedido: Apr. 21, 2025. [Online]. Disponible: <https://www.prosegur.com.ar/seguridad-hogar/Reconocimiento-inteligente>
- [33] “Alarma con reconocimiento facial inteligente de Movistar Prosegur Alarmas.” Accedido: Apr. 21, 2025. [Online]. Disponible: <https://www.movistar.es/blog/alarmas/alarma-reconocimiento-facial-inteligente/>
- [34] “Solución de reconocimiento facial.” Accedido: Apr. 21, 2025. [Online]. Disponible: <https://www.vivotek.com/es/solutions/application/67>
- [35] “Innovatrics – SmartFace para la Seguridad Pública” Accedido: Jul. 12, 2025. [Online]. Disponible: <https://www.innovatrics.com/es/soluciones-de-reconocimiento-facial/>
- [36] “SISTEMA CCTV FULL HD LITE – Detectar - Seguridad & Informática” Accedido: Jul. 12, 2025. [Online]. Disponible: <https://drive.google.com/file/d/1OA6ygobjRfXHQvLryicPejfKVd0kXVfv/view?usp=sharing>
- [37] “Honorarios Mínimos Indicativos de Informática.” Accedido: May 11, 2025. [Online]. Disponible: <https://copaipa.org.ar/informatica/>

- [38] “Esp32 Cam Ov2640 + Placa Base Programador Esp32-cam-mb Hobb | MercadoLibre.” Accedido: Apr. 21, 2025. [Online]. Disponible: https://articulo.mercadolibre.com.ar/MLA-1138395462-esp32-cam-ov2640-placa-base-programador-esp32-cam-mb-hobb-_JM#polycard_client=search-nordic&position=20&search_layout=grid&type=item&tracking_id=7518ee11-9962-40bf-9462-7c61fde63dff&wid=MLA1138395462&sid=search
- [39] “Notebook G4 10.1" color turquesa 2GB de Ram - 320GB HDD - Celeron” Accedido: Jul. 09, 2025. [Online]. Disponible: https://www.mercadolibre.com.ar/notebook-g4-101-color-turquesa-2gb-de-ram-320gb-hdd-celeron/p/MLA26697687#polycard_client=search-nordic&searchVariation=MLA26697687&wid=MLA2074896278&position=26&search_layout=stack&type=product&tracking_id=11245906-41be-40fa-9871-071ce54fa0df&sid=search
- [40] “Cable USB a Micro USB 3 mts - Mscompu10,” Accedido: Jul. 12, 2025. [Online]. Disponible: https://articulo.mercadolibre.com.ar/MLA-619041444-cable-usb-a-micro-usb-3-mts-mscompu10-_JM?searchVariation=180934560631#polycard_client=search-nordic&searchVariation=180934560631&position=32&search_layout=stack&type=item&tracking_id=8216c1d2-1277-4590-a04d-38742b67fde9
- [41] “Prolongador USB 2.0 activo reforzado con filtro - Extensor 10 mts,” Accedido: Jul. 12, 2025. [Online]. Disponible: https://articulo.mercadolibre.com.ar/MLA-676418829-prolongador-usb-20-activo-reforzado-y-filtro-extensor-10mts-_JM#polycard_client=search-nordic&position=13&search_layout=stack&type=item&tracking_id=affd64ef-54b0-4c4a-b747-9387d23011f0&wid=MLA676418829&sid=search

Anexos

Anexo A: Definiciones.

Anexo B: Configuración de Arduino.

Anexo C: Guía de Instalación.

Anexo D: Manual de Usuario.

ANEXO A

DEFINICIONES Y HERRAMIENTAS UTILIZADAS

Microcontrolador

Un microcontrolador es un circuito integrado que funciona como una pequeña computadora en un solo chip. Integra un procesador, memoria y periféricos de entrada/salida (E/S) en una sola unidad, lo que permite controlar dispositivos electrónicos de manera autónoma sin necesidad de componentes externos complejos. Los microcontroladores son ampliamente utilizados en aplicaciones de sistemas embebidos, donde es necesario realizar tareas específicas en tiempo real, como el control de motores, sensores, pantallas, y otros elementos en electrodomésticos, automóviles y dispositivos portátiles.

El diseño de un microcontrolador incluye:

- Unidad Central de Procesamiento (CPU): Ejecuta las instrucciones y gestiona las operaciones lógicas y aritméticas.
- Memoria: Puede ser ROM para almacenar el programa, y RAM para almacenar datos temporales durante la operación.
- Periféricos de entrada y salida: Facilitan la comunicación con otros dispositivos, como sensores, motores, o sistemas de comunicación.

Los microcontroladores son esenciales en dispositivos de bajo costo y bajo consumo energético, donde se necesita realizar tareas específicas de forma continua o en respuesta a eventos externos.[1]

ESP32



Ilustración 10 - Microcontrolador ESP32-CAM

El ESP32 es un microcontrolador de bajo costo y bajo consumo desarrollado por Espressif Systems. Incluye un procesador de doble núcleo Tensilica Xtensa LX6, junto con módulos de conectividad Wi-Fi y Bluetooth, lo que lo hace ideal para aplicaciones de Internet de las Cosas (IoT). Además de sus capacidades de conectividad, el ESP32 cuenta con múltiples interfaces, como ADCs, DACs, PWM, SPI, I2C, y UART, lo que permite una fácil integración con sensores, actuadores y otros dispositivos. Gracias a su bajo consumo de energía, el ESP32 es adecuado para aplicaciones móviles y portátiles que requieren operaciones eficientes y en tiempo real.[2]

Características clave del ESP32

- Doble núcleo: Integra dos núcleos de procesamiento Xtensa LX6, lo cual permite ejecutar múltiples tareas de forma simultánea.
- Conectividad inalámbrica: Incorpora Wi-Fi de 2.4 GHz y Bluetooth (Bluetooth clásico y BLE), lo que permite conectarse a redes y dispositivos de manera flexible.
- Memoria integrada: Cuenta con RAM de 520 KB y soporta almacenamiento en Flash externo.
- Bajo consumo de energía: Tiene diferentes modos de ahorro energético, ideales para aplicaciones que requieren funcionamiento en tiempo real pero con ahorro de energía, como sensores y dispositivos portátiles.
- Amplia gama de periféricos: Soporta interfaces como SPI, I2C, I2S, UART, ADC, DAC y PWM, lo cual facilita la conexión con sensores y otros componentes electrónicos.
- Facilidad de programación: Se puede programar con múltiples entornos, como el IDE de Arduino, PlatformIO y ESP-IDF (el framework de desarrollo oficial de Espressif).

Arduino IDE

El Arduino Integrated Development Environment contiene un editor de texto para escribir código, una consola, una barra de herramientas con botones para diferentes funciones y una serie de menús.

Los programas de Arduino son denominados bocetos, los cuales se escriben en el editor de texto y se guardan con una extensión “.ino”.

Arduino IDE es una mejora del IDE clásico, con mayor rendimiento, interfaz de usuario mejorada y muchas características nuevas, como autocompletado, un depurador incorporado y sincronización de bocetos con Arduino Cloud.

Descripción General

Arduino IDE presenta una nueva barra lateral, lo que hace que las herramientas más utilizadas sean más accesibles.



Ilustración 11 - Barra lateral de Arduino IDE

- Verificar / Cargar: compila y carga tu código en tu placa Arduino.
- Seleccionar placa y puerto: las placas Arduino detectadas se muestran automáticamente aquí, junto con el número de puerto.
- Cuaderno de bocetos: aquí encontrarás todos tus bocetos almacenados localmente en tu ordenador. Además, puedes sincronizarlos con Arduino Cloud y obtenerlos del entorno en línea.
- Administrador de placas: explora los paquetes de Arduino y de terceros que se pueden instalar. Por ejemplo, para usar una placa MKR WiFi 1010 requiere Arduino SAMD Boards instalado.
- Administrador de bibliotecas: explore miles de bibliotecas Arduino, creadas por Arduino y su comunidad.
- Depurador: prueba y depura programas en tiempo real.
- Buscar: busca palabras clave en tu código.
- Abrir Monitor Serial: abre la herramienta Monitor Serial, como una nueva pestaña en la consola.

Características

Arduino IDE es un editor versátil con numerosas funciones. Puedes instalar bibliotecas directamente, sincronizar tus bocetos con Arduino Cloud, depurarlos y mucho más. En esta sección, se enumeran algunas de las funciones principales, junto con un enlace a un artículo más detallado.

Bocetos

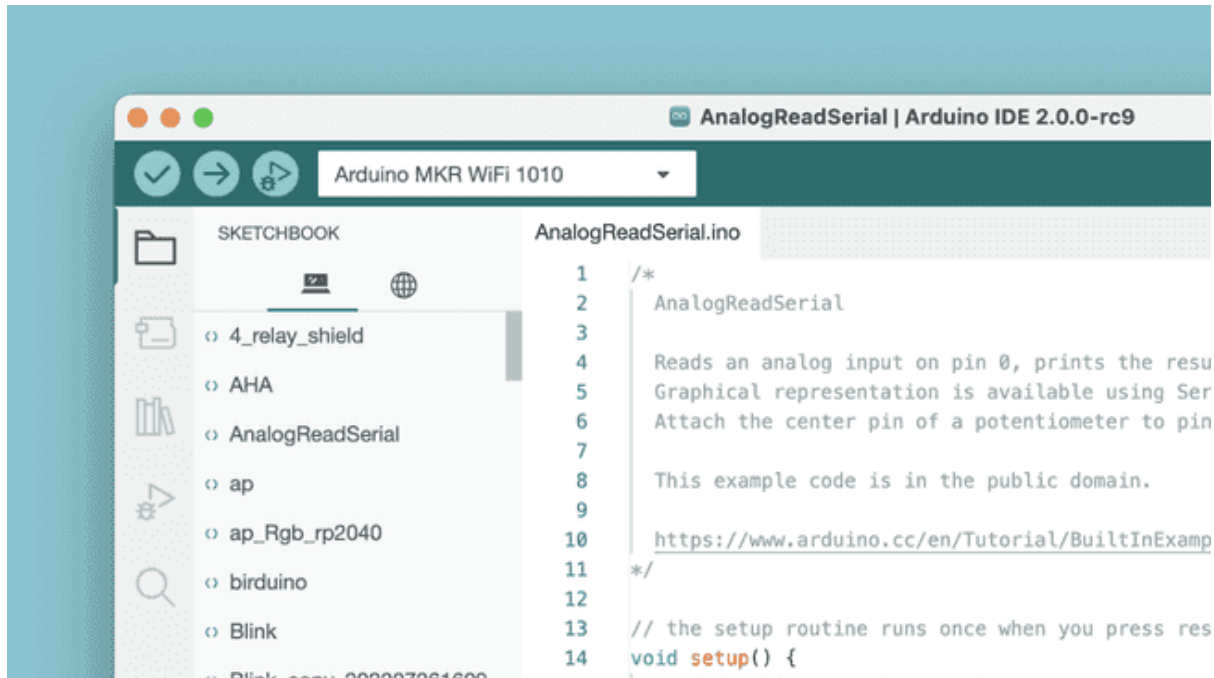


Ilustración 12 - Bocetos

El cuaderno de bocetos es donde se almacenan tus archivos de código. Los bocetos de Arduino se guardan como archivos “.ino” y deben almacenarse en una carpeta con el mismo nombre. Por ejemplo, un boceto llamado my_sketch.ino debe almacenarse en una carpeta llamada my_sketch.

Normalmente, sus bocetos se guardan en una carpeta llamada Tus Documentos.

Para acceder a su cuaderno de bocetos, haga clic en el ícono de carpeta ubicado en la barra lateral.

Administrador de placas



Ilustración 13 - Administrador de Placas

Con el Administrador de Placas, puede explorar e instalar paquetes de placas. Cada paquete de placa contiene las instrucciones para compilar su código en las placas incluidas en él.

Existen varios paquetes de placas Arduino disponibles, como avr, samd, megaavr y más.

Administrador de bibliotecas

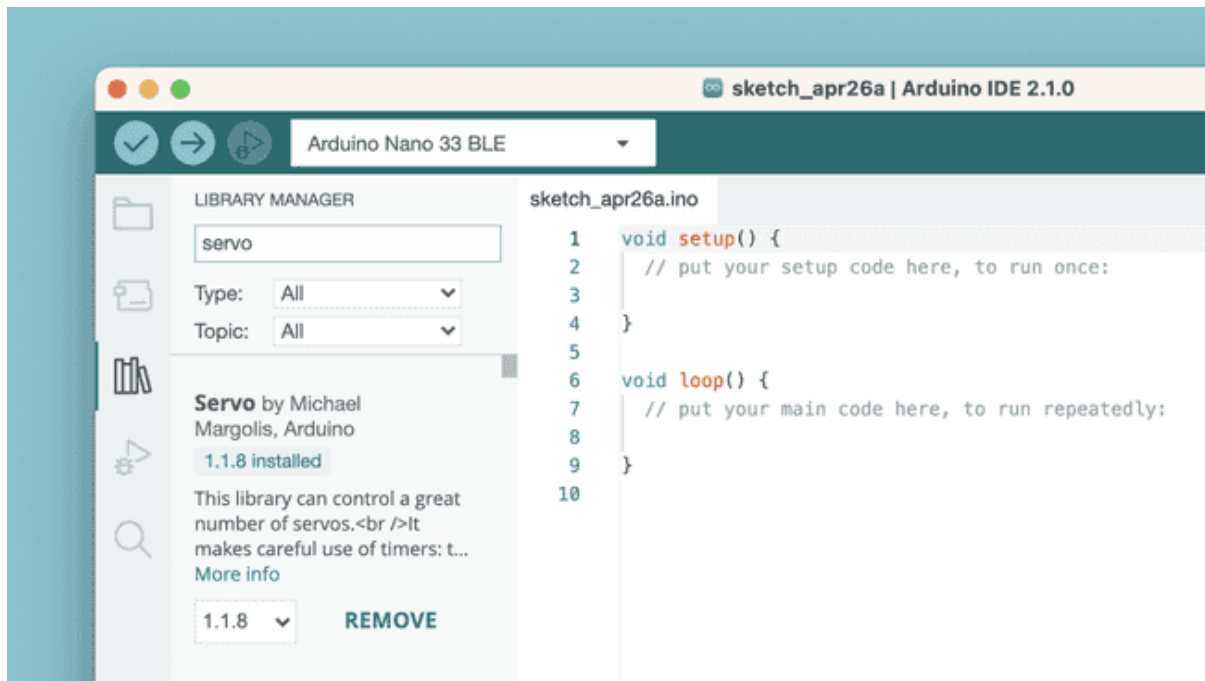


Ilustración 14 - Administrador de bibliotecas

Con el administrador de bibliotecas puedes explorar e instalar miles de bibliotecas. Las bibliotecas son extensiones de la API de Arduino y facilitan, por ejemplo, el control de un servomotor, la lectura de sensores específicos o el uso de un módulo Wi-Fi.

Monitor Serial

El Serial Monitor es una herramienta que le permite ver la transmisión de datos desde su placa, por ejemplo, a través de `Serial.print()`.

Anteriormente, esta herramienta se encontraba en una ventana independiente, pero ahora está integrada con el editor. Esto facilita tener varias instancias ejecutándose simultáneamente en el equipo.

Bocetos de ejemplo

Una parte importante de la documentación de Arduino son los bocetos de ejemplo que vienen con las bibliotecas. Estos muestran ejemplos de las funciones utilizadas en la práctica, ilustrando el uso previsto y las características de cada biblioteca.

Las bibliotecas que vienen incluidas como parte de un paquete de placas también pueden incluir bibliotecas, y esas bibliotecas a menudo incluyen bocetos de ejemplo.

Para abrir los bocetos de ejemplo incluidos en las bibliotecas que ha instalado manualmente o que vienen incluidos en los paquetes de placas, navegue a Archivo > Ejemplos y busque la biblioteca que está buscando en la lista que aparece.

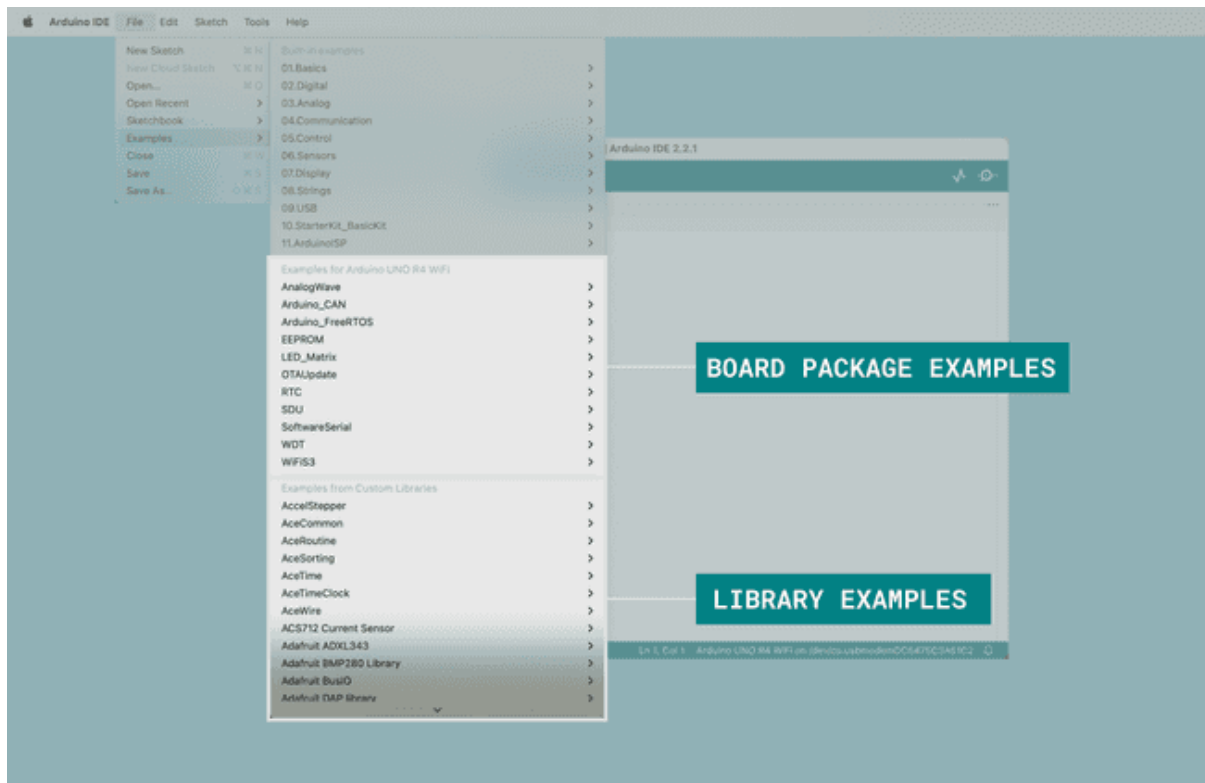


Ilustración 15 - Lista de ejemplos

En la imagen de arriba, puedes ver cómo se ve la lista de ejemplos cuando una placa UNO R4 WiFi está conectada a tu computadora.

Depuración

La herramienta de depuración se utiliza para probar y depurar programas, de ahí su nombre. Permite navegar por la ejecución de un programa de forma controlada.[3]

Sistema de Seguridad

Un sistema de seguridad es un conjunto de medidas y tecnologías diseñadas para proteger los activos físicos, información y personas contra amenazas y vulnerabilidades, ya sean internas o externas. Estos sistemas pueden incluir dispositivos de monitoreo, control de acceso, alarmas, cámaras de vigilancia y protocolos de respuesta ante incidentes. El objetivo de un sistema de seguridad es minimizar riesgos y proporcionar una respuesta rápida ante

cualquier evento que pueda comprometer la integridad o seguridad de las instalaciones o información.[4]

Internet de las Cosas

El Internet de las Cosas (IoT, por sus siglas en inglés de “Internet Of Things”) se refiere a la interconexión de dispositivos físicos a través de internet, permitiendo que recojan y compartan datos. Esta tecnología permite que objetos cotidianos, como electrodomésticos, vehículos y sensores, se conecten y comuniquen entre sí, facilitando la automatización y el monitoreo en tiempo real. Algunas aplicaciones del IoT incluyen:

1. **Hogares inteligentes:** Dispositivos como termostatos y luces que pueden ser controlados de manera remota.
2. **Salud:** Monitores de salud que envían datos a médicos para un seguimiento continuo.
3. **Agricultura:** Sensores que optimizan el riego y monitorean la salud de los cultivos.
4. **Transporte:** Vehículos conectados que mejoran la gestión del tráfico y la seguridad vial.[5]

Componentes clave del Internet de las Cosas (IoT):

1. **Dispositivos y sensores:** Equipos que recopilan datos del entorno.
2. **Conectividad:** Métodos de comunicación como Wi-Fi, Bluetooth y redes móviles.
3. **Plataformas de gestión:** Infraestructuras que almacenan y gestionan los datos.
4. **Análítica y procesamiento de datos:** Herramientas que analizan la información recolectada.
5. **Seguridad:** Medidas para proteger los datos y dispositivos.
6. **Interfaz de usuario:** Aplicaciones que permiten la interacción con los dispositivos.[6]

Framework

Un framework es un conjunto de herramientas, bibliotecas y reglas que proporciona una estructura predefinida para desarrollar software. Facilita la creación de aplicaciones al ofrecer funcionalidades comunes, como manejo de bases de datos, enrutamiento y gestión de interfaces gráficas, lo que ahorra tiempo y esfuerzo a los desarrolladores.

Los frameworks se usan en diversos ámbitos, como el desarrollo web (Django, Ruby on Rails), aplicaciones móviles (Flutter, React Native) y desarrollo de software en general (Spring para Java, Angular para frontend).

Ionic

Ionic es un kit de herramientas de interfaz de usuario de código abierto para crear aplicaciones móviles de alto rendimiento y alta calidad utilizando tecnologías web (HTML, CSS y JavaScript) con integraciones para frameworks populares como Angular, React y Vue. Crea e implementa aplicaciones que funcionan en múltiples plataformas, como iOS nativo, Android y la web como una aplicación web progresiva, todo con una base de código.[7]

Aplicación Web Progresiva (PWA)

Las aplicaciones web progresivas (PWA) son aplicaciones que se compilan mediante tecnologías web y que se pueden instalar y ejecutar en todos los dispositivos, desde un código base.

Las PWA proporcionan experiencias nativas a los usuarios en dispositivos auxiliares. Se adaptan a las funcionalidades admitidas por cada dispositivo y también se pueden ejecutar en exploradores web, como sitios web.

Ventajas

Experiencias nativas

Cuando se instalan en un dispositivo, las PWA funcionan igual que otras aplicaciones. Por ejemplo:

- Las PWA tienen sus propios iconos de aplicación que se pueden agregar a la pantalla principal o barra de tareas de un dispositivo.
- Las PWA se pueden iniciar automáticamente cuando se abre un tipo de archivo asociado.
- Las PWA se pueden ejecutar cuando el usuario inicia sesión.
- Las PWA se pueden enviar a almacenes de aplicaciones, como Microsoft Store.

Funcionalidades avanzadas

Las PWA también tienen acceso a funcionalidades avanzadas. Por ejemplo:

- Las PWA pueden seguir funcionando cuando el dispositivo está sin conexión.
- Las PWA admiten notificaciones push.
- Las PWA pueden realizar actualizaciones periódicas incluso cuando la aplicación no se está ejecutando.
- Los PWA pueden acceder a las características de hardware.

Ventajas relacionadas con la web

Por último, las PWA se pueden ejecutar en exploradores web, al igual que los sitios web. Ejecutar como sitios web proporciona ventajas a las PWA, como:

- Los motores de búsqueda pueden indexar las PWA.
- Las PWA se pueden compartir e iniciar desde un vínculo web estándar.
- Las PWA son seguras para los usuarios porque usan puntos de conexión HTTPS seguros y otras medidas de seguridad de usuario.
- Los PWA se adaptan al tamaño o la orientación de la pantalla del usuario y al método de entrada.
- Las PWA pueden usar API web avanzadas como WebBluetooth, WebUSB, WebPayment, WebAuthn o WebAssembly.

Menor coste de desarrollo

Las PWA tienen un coste de desarrollo multiplataforma mucho menor que las aplicaciones compiladas que requieren un código base específico e independiente para cada plataforma, como Android, iOS y cada sistema operativo de escritorio.

Con una PWA, puede usar un único código base que se comparte entre su sitio web, aplicación móvil y aplicación de escritorio (entre sistemas operativos).

Compatibilidad entre dispositivos

Las aplicaciones web progresivas (PWA) se compilan mediante código HTML, CSS y JavaScript hospedado en servidores web y que se ejecuta en motores de explorador web. Se pueden usar directamente en un explorador web, como un sitio web, o instalarse en un dispositivo desde una tienda de aplicaciones o mediante la característica de instalación de la aplicación de un explorador compatible.

La experiencia del usuario de una PWA se escala o reduce verticalmente con las funcionalidades del dispositivo. En concreto, en Windows con Microsoft Edge, puede crear experiencias envolventes que se integren profundamente con el sistema operativo. El uso de solo tecnologías web garantiza que la aplicación funcionará en otros exploradores, sistemas operativos y dispositivos.

La informática de escritorio sigue siendo el entorno de productividad principal para muchos usuarios. Las PWA pueden proporcionar experiencias totalmente personalizadas e instalables en un sistema operativo de escritorio que heredan las ventajas de ese entorno. Sin embargo, las PWA también se pueden instalar en dispositivos móviles o usarse directamente en exploradores web.

Cerrando las diferencias entre la web y lo nativo

No hay mucho que no se pueda hacer hoy en día con las tecnologías web. Muchas características que antes solo eran posibles mediante lenguajes específicos del dispositivo ahora se pueden lograr mediante tecnologías web estándar. Por ejemplo:

- Gestionar archivos.
- Compartir contenido con otras aplicaciones.
- Obtener acceso al Portapapeles.
- Sincronizar los datos y capturar los recursos en segundo plano.
- Acceder al hardware del dispositivo, como Bluetooth y USB.
- Almacenar contenido en bases de datos.
- Aprovechar las ventajas de los gráficos acelerados por hardware.
- Usar diseños, animaciones y filtros CSS para crear diseños avanzados.
- Ejecutar código de rendimiento casi compilado con WebAssembly.

Las PWA en el escritorio pueden aprovechar al máximo estas funcionalidades para ofrecer lo que los usuarios esperan que las aplicaciones de escritorio puedan hacer. Estas funcionalidades proporcionan experiencias más atractivas en plataformas web y de escritorio.

Casos de éxito

El uso de tecnologías de aplicación web progresiva (PWA) es una excelente manera de hacer que la aplicación sea segura, reconocible, enlazable, fácil de instalar y de actualizar, con capacidad de respuesta e independiente de la red. Muchas empresas han tenido éxito con las PWA. Por ejemplo:

- La PWA de Starbucks ha duplicado el número de usuarios activos diarios. Los pedidos en el escritorio se hacen casi con la misma velocidad que en el móvil.
- Trivago ha experimentado un aumento del 150 % para las personas que agregan su PWA a la pantalla principal. El aumento de la participación provocó un aumento del 97 % en los registros de las ofertas de hotel.
- Tinder ha reducido los tiempos de carga de 11,91 segundos a 4,68 segundos con su PWA. La aplicación es un 90 % más pequeña que la aplicación de Android compilada.[8]

Android

Android es un sistema operativo basado en el kernel de Linux, desarrollado principalmente para dispositivos móviles como teléfonos inteligentes y tabletas. Creado por Google, Android proporciona una plataforma abierta para desarrolladores, permitiéndoles crear aplicaciones que aprovechen las funciones y sensores de los dispositivos. El sistema operativo incluye una arquitectura modular, facilitando la interacción entre la interfaz de usuario y el

hardware, así como la administración eficiente de memoria y energía, características esenciales para el funcionamiento en dispositivos portátiles.[9]

IOS

IOS es un sistema operativo móvil desarrollado por Apple para sus dispositivos móviles, incluyendo el iPhone, iPad y iPod Touch. Basado en macOS (sistema operativo desarrollado también por Apple para sus computadoras de escritorio y portátiles, basado a su vez en Unix), este sistema operativo está diseñado para ser eficiente en el uso de recursos y optimizado para pantallas táctiles. iOS está basado en un núcleo de microkernel que ofrece estabilidad, seguridad y un rendimiento optimizado. Su arquitectura permite una interacción fluida con las aplicaciones y servicios del ecosistema de Apple.[10]

Aplicaciones

Una aplicación es un tipo de software diseñado específicamente para ejecutar tareas o funciones particulares, facilitando a los usuarios la realización de actividades específicas en dispositivos como computadoras, teléfonos inteligentes o tabletas. Las aplicaciones pueden ser programas independientes o formar parte de un sistema operativo, y abarcan una variedad de funciones, desde procesamiento de texto hasta gestión de datos o entretenimiento.[11]

Base de datos

Un "sistema de bases de datos" es un conjunto organizado de datos almacenados y accesibles de manera que se pueda manipular la información de manera eficiente. En su definición, destaca la importancia de las reglas de integridad y la estructura organizada de los datos, que permiten no solo almacenar, sino también garantizar que los datos sean precisos y estén disponibles para ser consultados y modificados.[12]

Una base de datos relacional, es un tipo de base de datos que utiliza el modelo relacional para almacenar datos. En este modelo, los datos se organizan en tablas (también llamadas relaciones), que están formadas por filas (tuplas) y columnas (atributos). Estas tablas están interrelacionadas entre sí mediante claves primarias y claves foráneas. Este modelo se basa en la teoría matemática de relaciones, lo que permite que las bases de datos relacionales sean extremadamente flexibles y eficaces para realizar consultas complejas a través del lenguaje SQL.

Las bases de datos relacionales, por tanto, ofrecen ventajas como la consistencia, la integridad de los datos y la capacidad de realizar operaciones de manera eficiente gracias a su estructura organizada en tablas.[13]

PostgreSQL

PostgreSQL es un potente sistema de base de datos relacional de objetos de código abierto con más de 35 años de desarrollo activo que le ha valido una sólida reputación de confiabilidad, robustez de características y rendimiento.[14]

API

Una API (Application Programming Interface o Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permite que distintos programas o sistemas se comuniquen entre sí. Las APIs especifican cómo los componentes de software deben interactuar, proporcionando una capa de abstracción que facilita el intercambio de datos y funcionalidades entre aplicaciones de manera estructurada y eficiente.[15]

Flask

Flask es un marco de desarrollo web escrito en Python que permite crear aplicaciones web de manera rápida y sencilla.[16]

Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo de diseño de software en el cual las tareas se dividen entre proveedores de recursos o servicios, llamados servidores, y solicitantes de servicios, llamados clientes. Los clientes realizan peticiones al servidor a través de una red, y el servidor procesa estas solicitudes y devuelve una respuesta adecuada. Este enfoque permite la centralización de los recursos y facilita la escalabilidad del sistema.[17]

Protocolos de Comunicación

Los protocolos de comunicación son un conjunto de reglas que permiten el intercambio de datos entre dispositivos en una red. Estos protocolos definen cómo se establecen, mantienen y finalizan las conexiones, además del formato y la sincronización de los mensajes transmitidos. En sistemas cliente-servidor, los protocolos más comunes incluyen HTTP y WebSocket.[18]

HTTP

El Protocolo de Transferencia de Hipertexto (HTTP) es un protocolo de nivel de aplicación utilizado para la transferencia de información en la World Wide Web. Funciona bajo un esquema cliente-servidor, donde el cliente realiza solicitudes (requests) y el servidor responde con los recursos solicitados (responses). HTTP es sin estado, lo que significa que no mantiene información sobre conexiones anteriores.[19]

WebSocket

WebSocket es un protocolo de comunicación que proporciona un canal de comunicación full-duplex sobre una única conexión TCP. A diferencia de HTTP, que es unidireccional, WebSocket permite la interacción bidireccional entre cliente y servidor en tiempo real, lo que lo hace ideal para aplicaciones que requieren actualizaciones constantes como sistemas de vigilancia, mensajería o videojuegos en línea.[20]

Client-Side Rendering (Renderizado del lado del cliente)

El Client-Side Rendering (CSR) o renderizado del lado del cliente es una técnica en la cual la mayor parte del procesamiento y generación del contenido visual de una aplicación web ocurre en el navegador del usuario. En este enfoque, el servidor envía principalmente una estructura HTML básica junto con archivos JavaScript, que luego se encargan de generar dinámicamente el contenido de la página. CSR es común en aplicaciones web modernas, especialmente las desarrolladas con frameworks como React, Angular o Vue.js, y permite una experiencia de usuario más fluida y dinámica.[21]

Notificaciones en Tiempo Real

Las notificaciones en tiempo real son alertas enviadas instantáneamente a los usuarios para informarles sobre eventos o cambios importantes, incluso si no están usando activamente una aplicación. Estas notificaciones permiten mantener informados a los usuarios sobre actividades relevantes, como mensajes nuevos, actualizaciones, o recordatorios. Las notificaciones en tiempo real suelen implementarse mediante tecnologías como el envío de notificaciones push y WebSockets, y son esenciales en aplicaciones de mensajería, redes sociales y monitoreo en tiempo real.[22]

Deep Learning

El aprendizaje profundo pretende conseguir que los ordenadores sean capaces de construir automáticamente conceptos complejos a partir de conceptos sencillos utilizando grandes redes neuronales y gran cantidad de datos.

Deep Learning es una subárea del aprendizaje automático que utiliza redes neuronales artificiales con múltiples capas (redes neuronales profundas) para aprender representaciones jerárquicas de datos, permitiendo la resolución de tareas complejas como reconocimiento de voz, visión por computadora y procesamiento de lenguaje natural.[23]

Red Neuronal: Una red neuronal es básicamente una función matemática definida por nodos en capas donde cada nodo es una función más simple que depende de los nodos de la capa anterior.[24]

Representation Learning

La representación de los datos es crucial para el éxito de los algoritmos de Machine Learning (Lo que también ocurre en la vida real; imaginemos la dificultad de hacer operaciones aritméticas en números romanos, por ejemplo).[24]

Representation Learning es un enfoque dentro del aprendizaje automático que se centra en aprender representaciones eficientes de los datos. Estas representaciones, o características, son transformaciones de los datos originales que facilitan la realización de tareas de aprendizaje, como clasificación, predicción o agrupamiento, sin necesidad de intervención manual para extraer características. El objetivo es permitir que el modelo aprenda automáticamente las características más relevantes de los datos para mejorar su rendimiento en diversas tareas.[25]

Machine Learning

El aprendizaje automático o machine learning es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos que permiten a las computadoras aprender y mejorar a partir de datos, sin ser programadas explícitamente para cada tarea. El aprendizaje automático se utiliza en diversas aplicaciones, desde motores de búsqueda y diagnósticos médicos hasta reconocimiento de voz y sistemas de recomendación. Su capacidad para analizar grandes volúmenes de datos y extraer patrones ha transformado múltiples industrias y sigue siendo un área de investigación activa con un impacto significativo en la sociedad.[26]

Inteligencia Artificial

“La inteligencia artificial es la disciplina derivada de los conocimientos científicos y de ingeniería que estructura las metodologías, técnicas y algoritmos que apoyan la resolución de problemas, intentando emplear mecanismos sofisticados de razonamiento y usando de modo eficiente recursos y conocimientos”.[24]

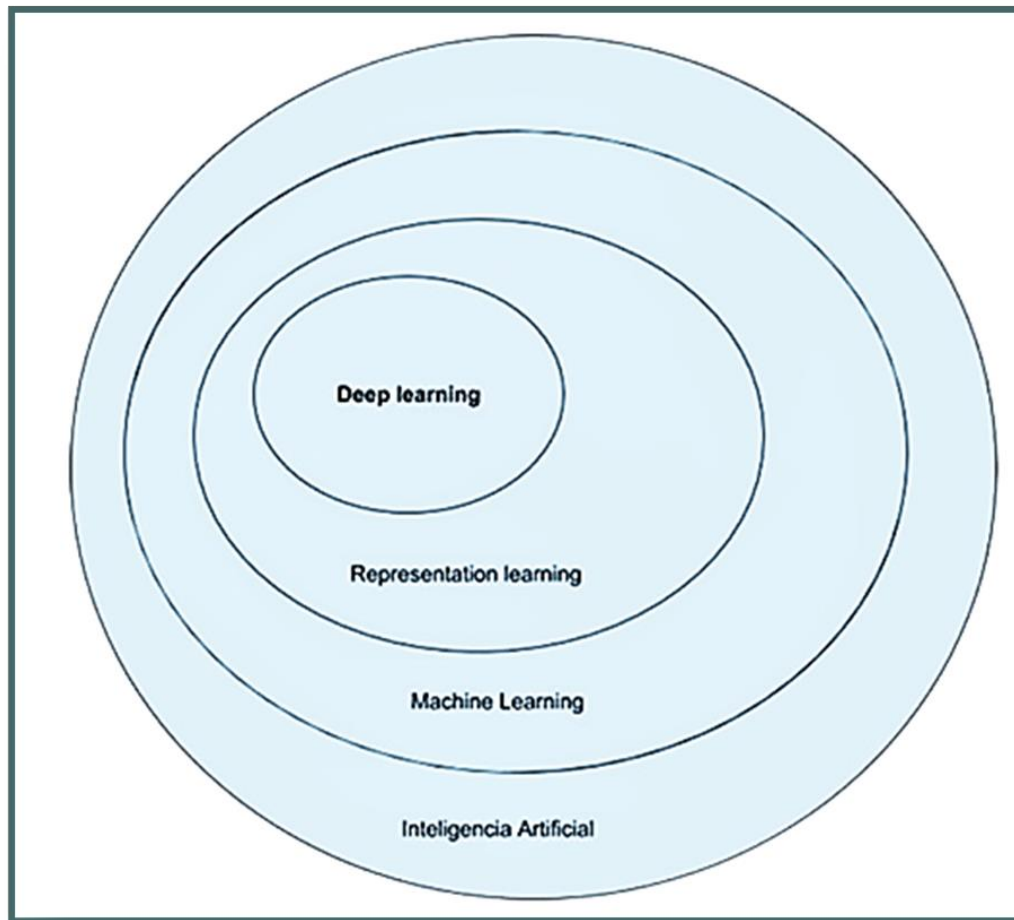


Ilustración 16 - Niveles de la IA

OpenCV

OpenCV es la biblioteca de visión por computadora más grande del mundo. OpenCV es de código abierto, contiene más de 2500 algoritmos y es operado por la Fundación Open Source Vision sin fines de lucro. Las áreas de aplicación de OpenCV incluyen:[27]

- Reconocimiento facial
- Reconocimiento de gestos
- Interacción persona-computadora
- Robótica móvil
- Comprensión de movimientos
- Detección y reconocimiento de objetos
- Segmentación

- Visión estéreo
- Reconstrucción de estructura a partir del movimiento
- Seguimiento de objetos
- Realidad aumentada

MobileNet

MobileNet es una familia de arquitecturas de redes neuronales convolucionales optimizadas para dispositivos móviles y sistemas con recursos computacionales limitados. Fue desarrollada por Google con el objetivo de ofrecer modelos livianos que mantuvieran una alta precisión en tareas de clasificación, detección y segmentación de imágenes, pero con un bajo requerimiento de cómputo y memoria. Su principal innovación radica en el uso de convoluciones separables en profundidad (depthwise separable convolutions), una técnica que reduce significativamente el número de parámetros y operaciones requeridas en comparación con las convoluciones tradicionales.

Gracias a esta arquitectura eficiente, MobileNet se ha convertido en una opción popular para aplicaciones de visión por computadora en entornos embebidos y móviles, incluyendo sistemas de vigilancia, donde el procesamiento debe realizarse en tiempo real y con restricciones de hardware. Existen varias versiones de esta red (MobileNetV1, V2, y V3), cada una con mejoras en velocidad, precisión y eficiencia energética.[28]

Tablero Kanban

Un tablero Kanban es una herramienta de gestión visual que permite a los equipos representar gráficamente las tareas en diferentes etapas de un proceso, facilitando la visualización del flujo de trabajo y la identificación de posibles cuellos de botella. Cada columna del tablero representa una fase específica del proceso, y las tarjetas dentro de las columnas simbolizan tareas individuales que se desplazan a medida que avanzan en el flujo de trabajo. [29]

Autenticación por JWT(JSON Web Token)

La autenticación mediante JSON Web Tokens (JWT) es un método que permite la transmisión segura de información entre partes en formato JSON. Según la especificación RFC 7519, un JWT es un estándar abierto que define una forma compacta y autocontenida de representar reclamaciones (claims) entre dos partes. Estas reclamaciones pueden ser verificadas y confiables porque están firmadas digitalmente, lo que garantiza la integridad y autenticidad de la información transmitida.

En el contexto de la autenticación, cuando un usuario inicia sesión exitosamente, el servidor genera un JWT que incluye información sobre la identidad del usuario y sus

privilegios. Este token es enviado al cliente y debe ser adjuntado en cada solicitud posterior a recursos protegidos, generalmente en el encabezado de autorización HTTP. De esta manera, el servidor puede validar el token y autorizar el acceso sin necesidad de mantener el estado de la sesión en el servidor, lo que hace que este método sea eficiente y escalable.[30]

Python (Flask)

Python es ampliamente reconocido por su versatilidad en el desarrollo de aplicaciones de inteligencia artificial y procesamiento de imágenes. Específicamente, Flask es un microframework minimalista que permite el desarrollo rápido de aplicaciones web. Sus principales ventajas incluyen:

- Amplia gama de bibliotecas: Python cuenta con bibliotecas especializadas como OpenCV, TensorFlow y Keras, que simplifican el desarrollo de aplicaciones de IA.
- Simplicidad y legibilidad del código: La sintaxis clara y concisa de Python reduce la complejidad en el desarrollo y mantenimiento del software.
- Comunidad y soporte: Python posee una comunidad extensa, lo que facilita la solución de problemas y la actualización constante de herramientas.
- Desempeño optimizado para IA: Aunque Python es un lenguaje interpretado, sus bibliotecas subyacentes están optimizadas en C/C++, lo que permite un rendimiento eficiente en tareas de procesamiento intensivo.

Node.js

Node.js es un entorno de ejecución basado en JavaScript, diseñado para aplicaciones web escalables y asincrónicas. Sus ventajas incluyen:

- Alto rendimiento en operaciones de entrada/salida: Gracias a su arquitectura no bloqueante y asincrónica, es altamente eficiente en aplicaciones con numerosas solicitudes concurrentes.
- Uso unificado de JavaScript: Permite emplear JavaScript tanto en el frontend como en el backend, lo que facilita la integración y reutilización del código.
- Extensa comunidad: Su popularidad ha permitido la creación de numerosos paquetes y bibliotecas.

Sin embargo, presenta desventajas para aplicaciones de procesamiento de imágenes:

- Limitada disponibilidad de bibliotecas especializadas para IA: A diferencia de Python, Node.js no cuenta con un ecosistema maduro para procesamiento de imágenes y detección de personas.

- Menor eficiencia en cálculos intensivos: Debido a su modelo basado en un solo hilo, no es la opción más eficiente para tareas que requieren alto procesamiento computacional.

Java

Java es un lenguaje compilado ampliamente utilizado en aplicaciones empresariales. Sus principales ventajas son:

- Alto rendimiento y escalabilidad: Gracias a la máquina virtual de Java (JVM), permite una ejecución optimizada en diferentes plataformas.
- Soporte para multihilo: Facilita la ejecución de procesos concurrentes, lo que puede ser beneficioso para ciertas tareas de procesamiento.
- Seguridad y estabilidad: Es una opción confiable para sistemas a gran escala.

Desventajas:

- Menor disponibilidad de bibliotecas especializadas para IA: Aunque existen opciones como Deeplearning4j, el ecosistema de aprendizaje automático en Java es menos desarrollado que en Python.
- Complejidad y curva de aprendizaje: La sintaxis de Java puede ser más verbosa, lo que aumenta la complejidad del desarrollo.

C# (.NET)

C# es un lenguaje desarrollado por Microsoft, ideal para aplicaciones empresariales y sistemas integrados con tecnologías de Microsoft. Sus ventajas incluyen:

- Alto rendimiento y seguridad: Ofrece herramientas avanzadas para el desarrollo de aplicaciones seguras y eficientes.
- Buena integración con el ecosistema de Microsoft: Especialmente útil si se utilizan servicios en la nube de Azure.

Desventajas:

- Menor adopción en IA y procesamiento de imágenes: Aunque existen bibliotecas para IA en C#, el soporte y la comunidad en esta área son limitados en comparación con Python.
- Curva de aprendizaje pronunciada: Su sintaxis y estructura pueden requerir un mayor tiempo de aprendizaje y desarrollo.

ANEXO B

CONFIGURACIÓN DE ARDUINO

En esta sección se detallarán los pasos para poner en marcha el software Arduino IDE.

Para poder usar la cámara ESP32, primero se debe realizar una configuración previa que consiste en:

1. Instalar el controlador

Debemos instalar el Driver correspondiente al chip CH340, que permite, por un lado, utilizar el puerto USB del PC y, por otro, convertir en comunicación serie los datos enviados al ESP32.



Ilustración 17 - Instalador del controlador

2. Configurar el IDE de Arduino

En el IDE de Arduino debemos ingresar al gestor de URLs de tarjetas, mediante la ruta: “Archivo/Preferencias/Gestor de URLs adicionales de tarjetas” y añadimos el siguiente enlace:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

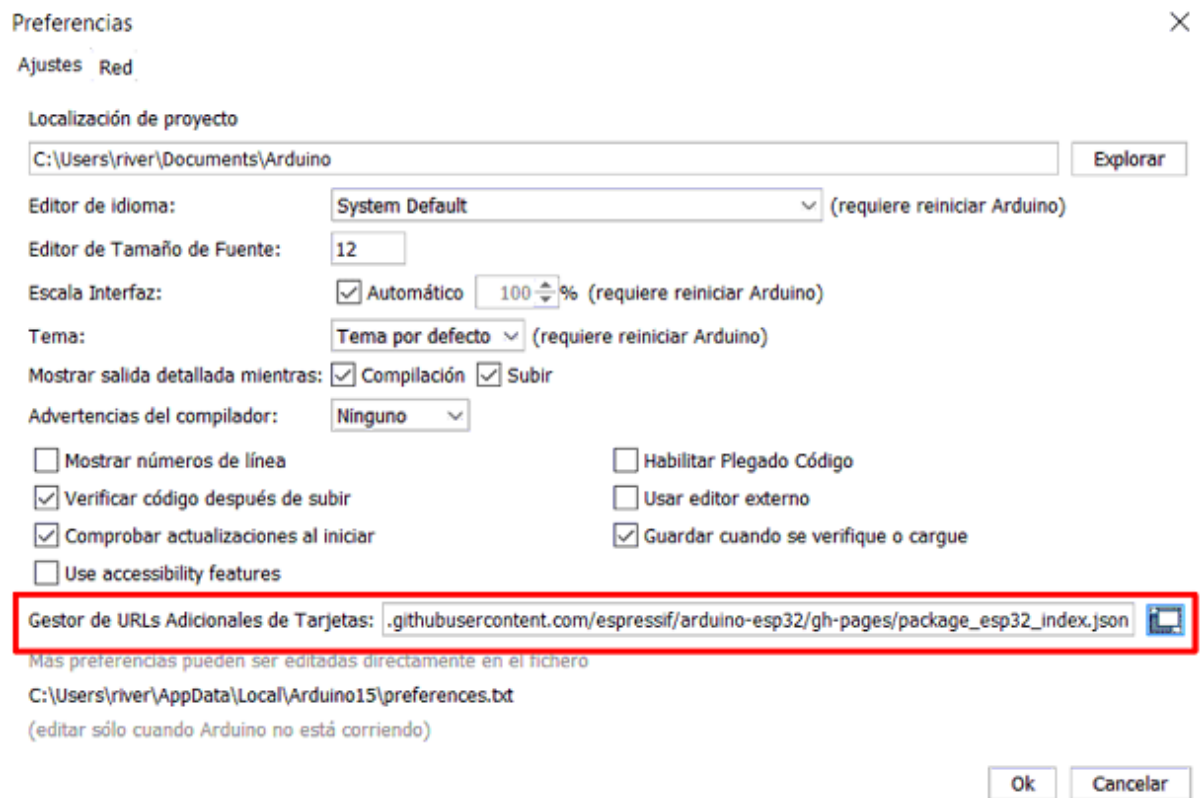


Ilustración 18 - Gestor de URLs

3. Instalar la librería de ESP32

Debemos acceder a: “Herramientas/Placas/Gestor de tarjetas” y escribimos: “esp32”.

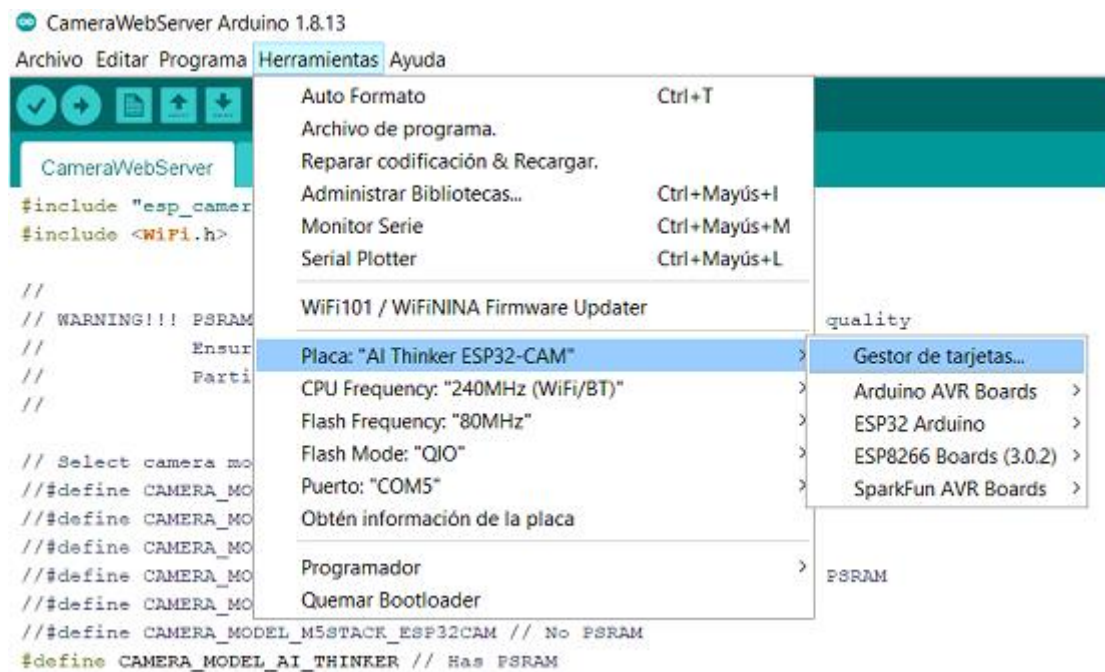


Ilustración 19 - Gestor de tarjetas

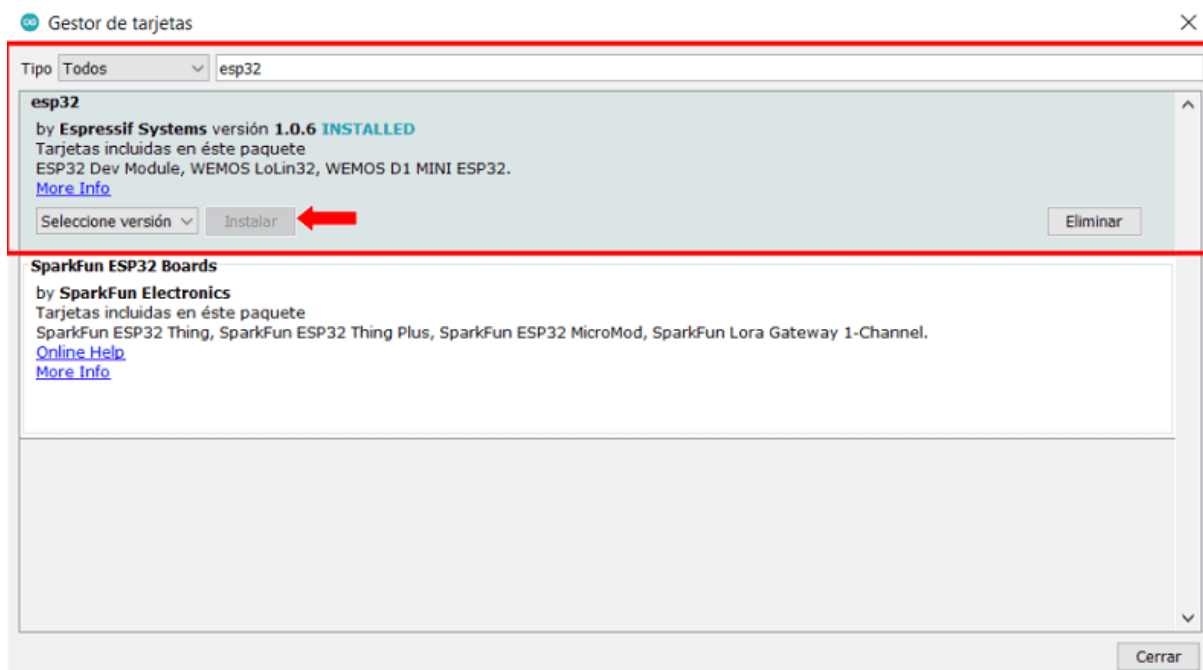


Ilustración 20 - Tarjeta de ESP32

4. Elegir la Placa

Seleccionar AI Thinker ESP32-CAM

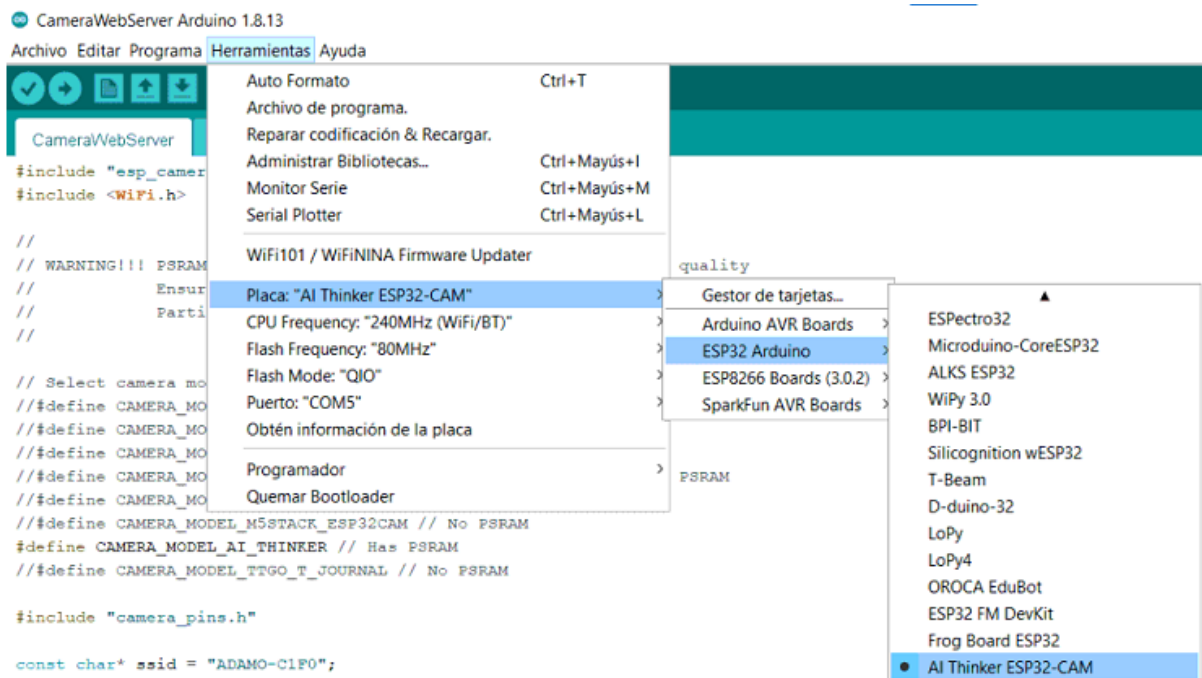


Ilustración 21 - Selección de placa

5. Cargar ejemplo

Seleccionar Ejemplo ESP32 Camera CameraWebServer

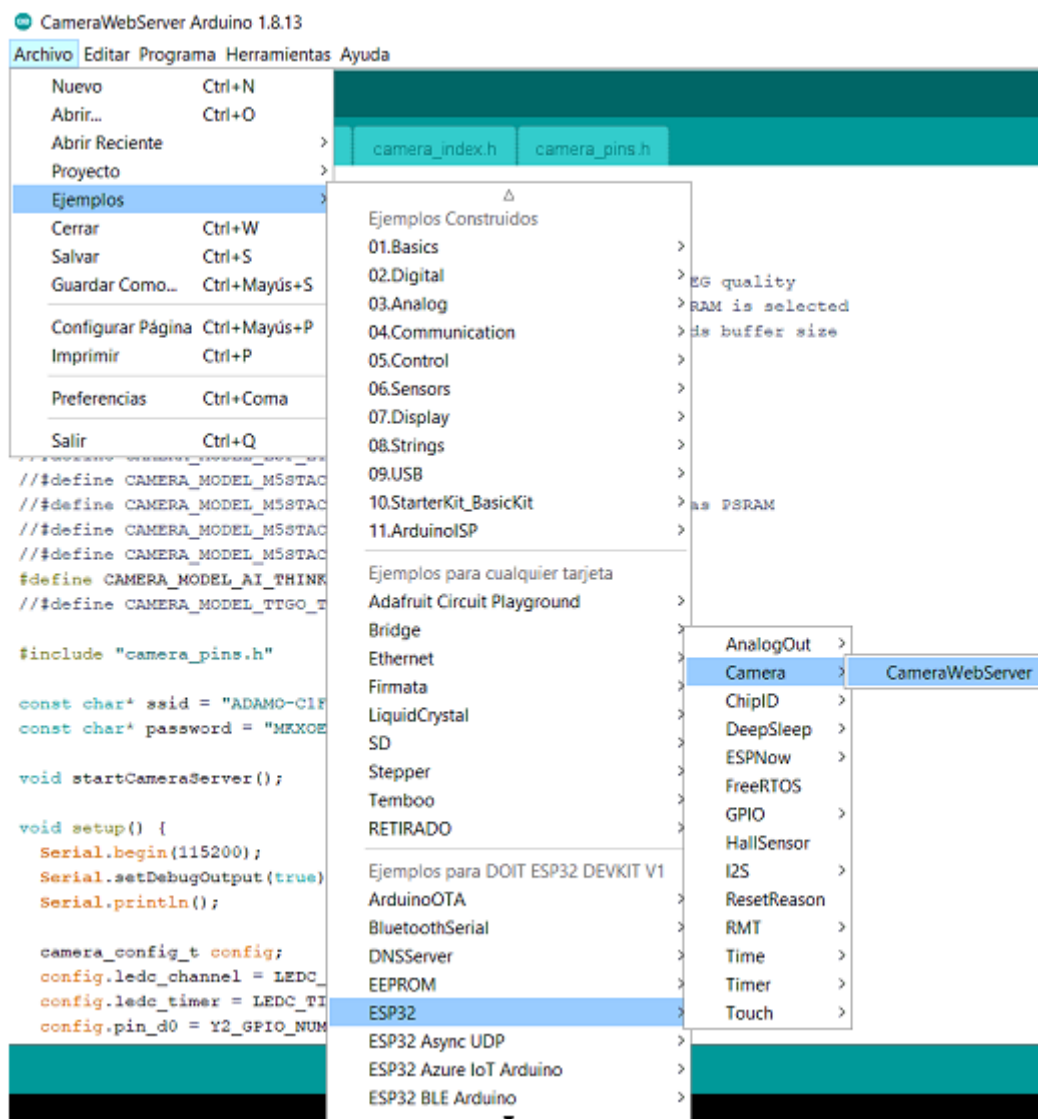


Ilustración 22 - Carga de ejemplo

6. Modificar el Código

Descomentamos la línea del modelo de cámara que vamos a utilizar. En este caso, descomentamos `CAMERA_MODEL_AI_THINKER`.

Asignamos a las variables “ssid” y “password” los valores correspondientes a la red WIFI.

Configuramos la velocidad del puerto serie a 115200



```
CameraWebServer Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda

CameraWebServer app_httpd.cpp camera_index.h camera_pins.h

#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//           Ensure ESP32 Wrover Module or other board with PSRAM is selected
//           Partial images will be transmitted if image exceeds buffer size
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM

#include "camera_pins.h"

const char* ssid = "Pon tu red ";
const char* password = "Pon tu clave ";

void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();
}
```

Ilustración 23 - Código de Arduino IDE

7. Subir el programa a la ESP32CAM

Con la ESP32CAM conectada, pulsaremos el ícono de la flecha hacia arriba, para subir el programa. En ese momento se comenzará a conectar con la placa y se mostrará el porcentaje de subida. Una vez alcanzado el 100%, nos mostrará el mensaje “*Hard resetting via RTS pin...*”.



Ilustración 24 - Subida de código

Abrimos el monitor serie, mediante el ícono de la lupa que se encuentra situado en el vértice superior derecho del IDE de Arduino y, pulsando el botón de “reset” de la ESP32CAM, nos aparecerá la dirección IP que se le ha asignado a la cámara.

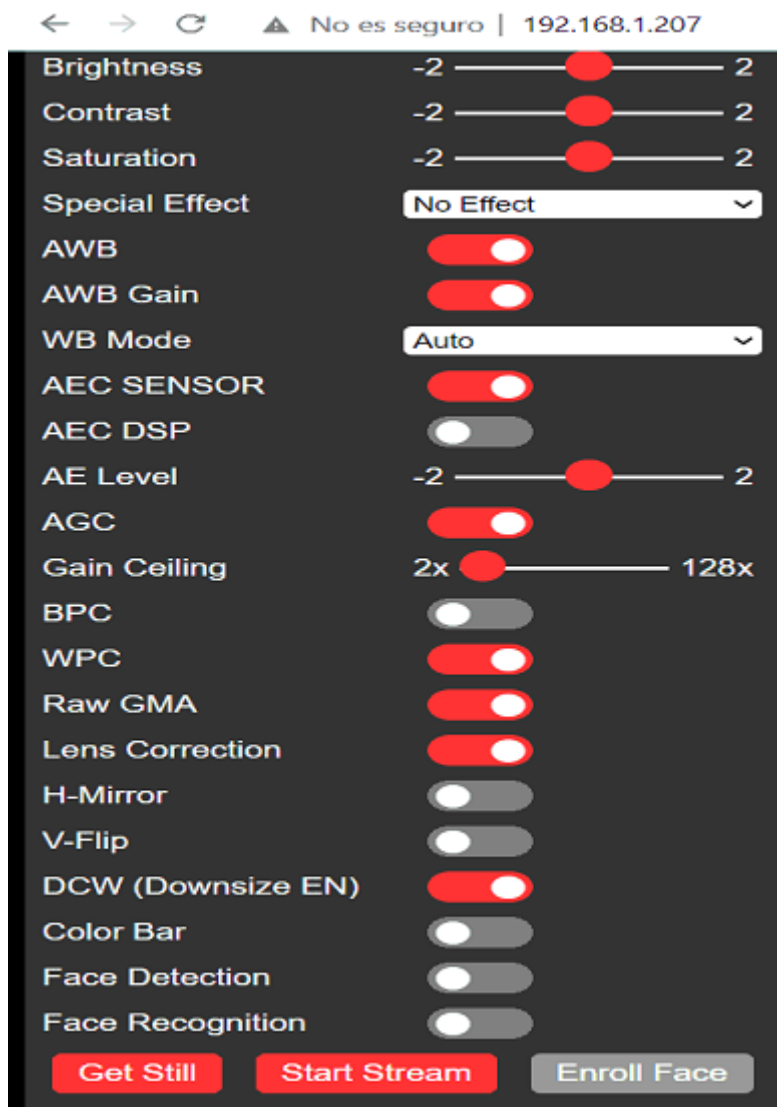


Ilustración 26 - Configuración de cámara

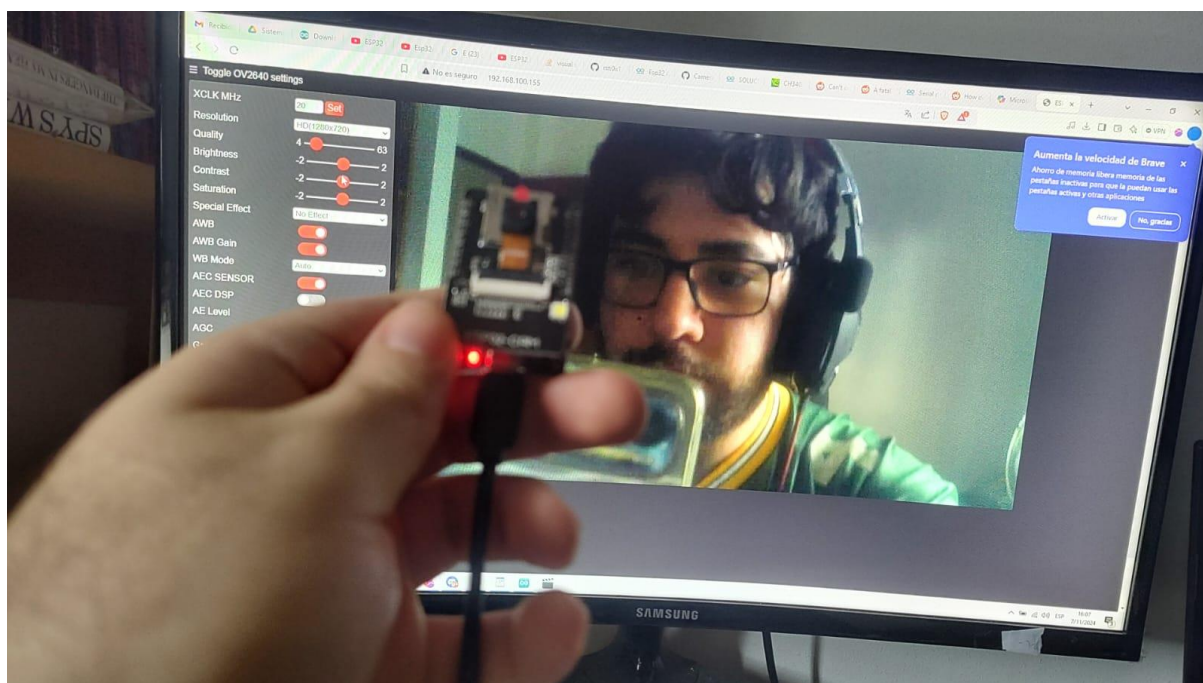


Ilustración 27 - Cámara funcionando

ANEXO C

GUÍA DE INSTALACIÓN

Instalación y configuración del entorno Arduino para ESP32-CAM

1. Descargar e instalar Arduino IDE
 - a. Descargar desde: <https://www.arduino.cc/en/software>
 - b. Instalar el IDE con los valores predeterminados.
2. Agregar soporte para placas ESP32
 - a. Abrir Arduino IDE y dirigirse a:
 - b. Archivo > Preferencias
 - c. En el campo “Gestor de URLs Adicionales de Tarjetas”, agregar la siguiente URL: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
 - d. Ir a Herramientas > Placa > Gestor de tarjetas, buscar ESP32 e instalar.
3. Cargar el código en la ESP32-CAM
 - a. Conectar el dispositivo mediante un adaptador USB-TTL.
 - b. Seleccionar la placa: AI Thinker ESP32-CAM.
 - c. Cargar el sketch proporcionado en el repositorio del sistema.

Instalación del backend en Python (Servidor Flask)

1. Instalar Python 3
 - a. Descargar desde: <https://www.python.org/downloads/>
 - b. Marcar la opción "Add Python to PATH" al instalar.
2. Clonar el repositorio del backend
 - a. URL: <https://github.com/nicooris98/Backend-Tesis>
3. Seguir guía de instalación dentro del archivo README.md
4. Ejecutar el servidor
 - a. El servidor quedará escuchando en localhost con el puerto seleccionado.

Instalación del frontend móvil (Aplicación Ionic)

1. Clonar el repositorio de la aplicación
 - a. URL: <https://github.com/nicooris98/mobile-ionic-tesis>
2. Seguir guía de instalación dentro del archivo README.md
3. Ejecutar la aplicación

ANEXO D

MANUAL DE USUARIO

Este manual tiene como objetivo guiar al usuario final en la utilización del sistema de vigilancia desarrollado. Está orientado a personas sin conocimientos técnicos avanzados, proporcionando una explicación paso a paso sobre el uso de la aplicación móvil, el monitoreo remoto de imágenes capturadas, la recepción de notificaciones ante detección de personas y otras funcionalidades disponibles.

El sistema está diseñado para ser intuitivo, accesible y fácil de usar. A través de este documento, el usuario podrá familiarizarse con las distintas pantallas, botones y opciones del sistema, así como entender el flujo de funcionamiento general. Además, se detallan aspectos importantes como los requisitos para su correcto funcionamiento, procedimientos básicos de inicio de sesión, visualización de eventos, y acciones disponibles en caso de emergencias.

Este manual también incluye capturas de pantalla y descripciones visuales para facilitar la comprensión del sistema, asegurando que cualquier usuario pueda aprovechar de forma efectiva todas sus capacidades.

Login

La pantalla de inicio de sesión permite al usuario acceder al sistema ingresando su nombre de usuario y contraseña. El formulario cuenta con validaciones que requieren un mínimo de 6 caracteres y no permite el envío de campos vacíos, garantizando así la seguridad y la integridad de los datos ingresados.

Login

Iniciar Sesión

Usuario

Contraseña

INGRESAR

IR A REGISTRO

Ilustración 28 - Login

Registro

En la pantalla de registro, el usuario puede crear una nueva cuenta. Para ello, debe ingresar un nombre de usuario único y completar los campos de contraseña y confirmación de contraseña, los cuales también están sujetos a validaciones. Una vez registrado, podrá iniciar sesión con sus credenciales.

Registro

Registrar usuario

Usuario

Contraseña

Repetir Contraseña

REGISTRAR

IR A LOGIN

Ilustración 29 - Registro de usuario

Página de Inicio

Esta es la pantalla principal de la aplicación. En ella se muestran las últimas 20 imágenes capturadas por la cámara del sistema, permitiendo al usuario tener una visión rápida y actualizada de los eventos recientes registrados.

Inicio

Ultimas 20 fotos



LLAMAR A EMERGENCIAS

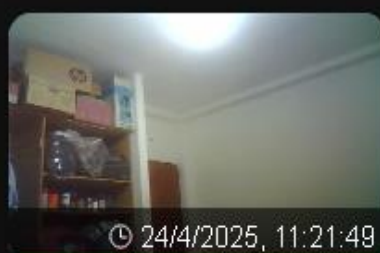


Ilustración 30 - Página de Inicio

Vista Paginada de Fotos

A través de esta vista, el usuario puede acceder al historial completo de imágenes captadas por la cámara. Las fotos se presentan en un formato paginado, lo que permite una navegación ordenada y eficiente por las capturas almacenadas.

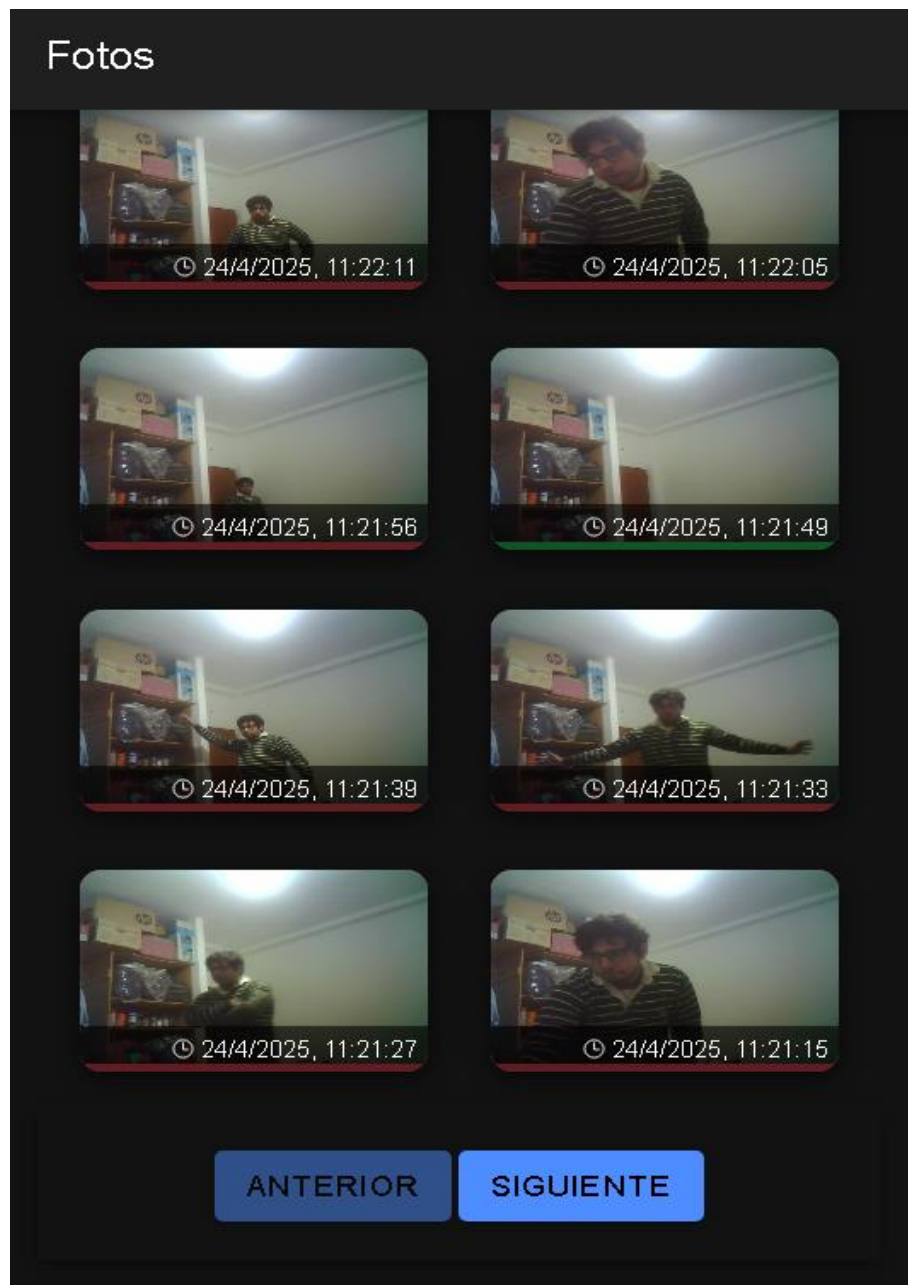


Ilustración 31 - Lista de fotos

Llamada a Emergencia

Esta función permite al usuario realizar una llamada inmediata al número de emergencias preconfigurado. Está representada mediante un botón claramente visible, diseñado para actuar rápidamente ante una situación sospechosa o de peligro.

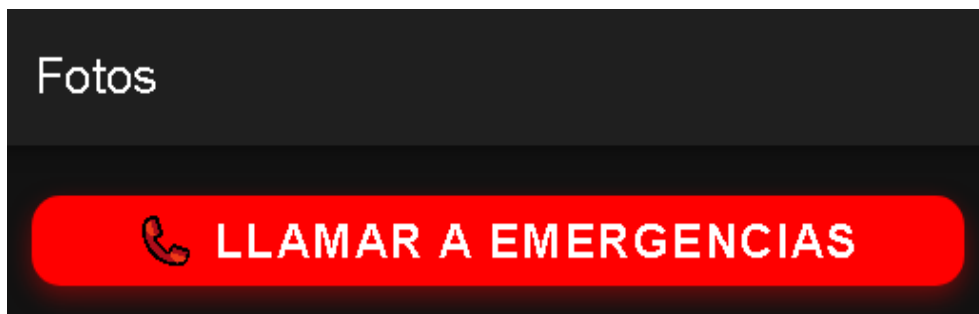


Ilustración 32 - Botón de llamar a emergencias

Notificación en Tiempo Real

El sistema emite notificaciones automáticas al dispositivo móvil del usuario cada vez que se detecta una persona en las imágenes. Estas alertas permiten actuar de manera oportuna, incluso cuando el usuario no se encuentra utilizando activamente la aplicación.

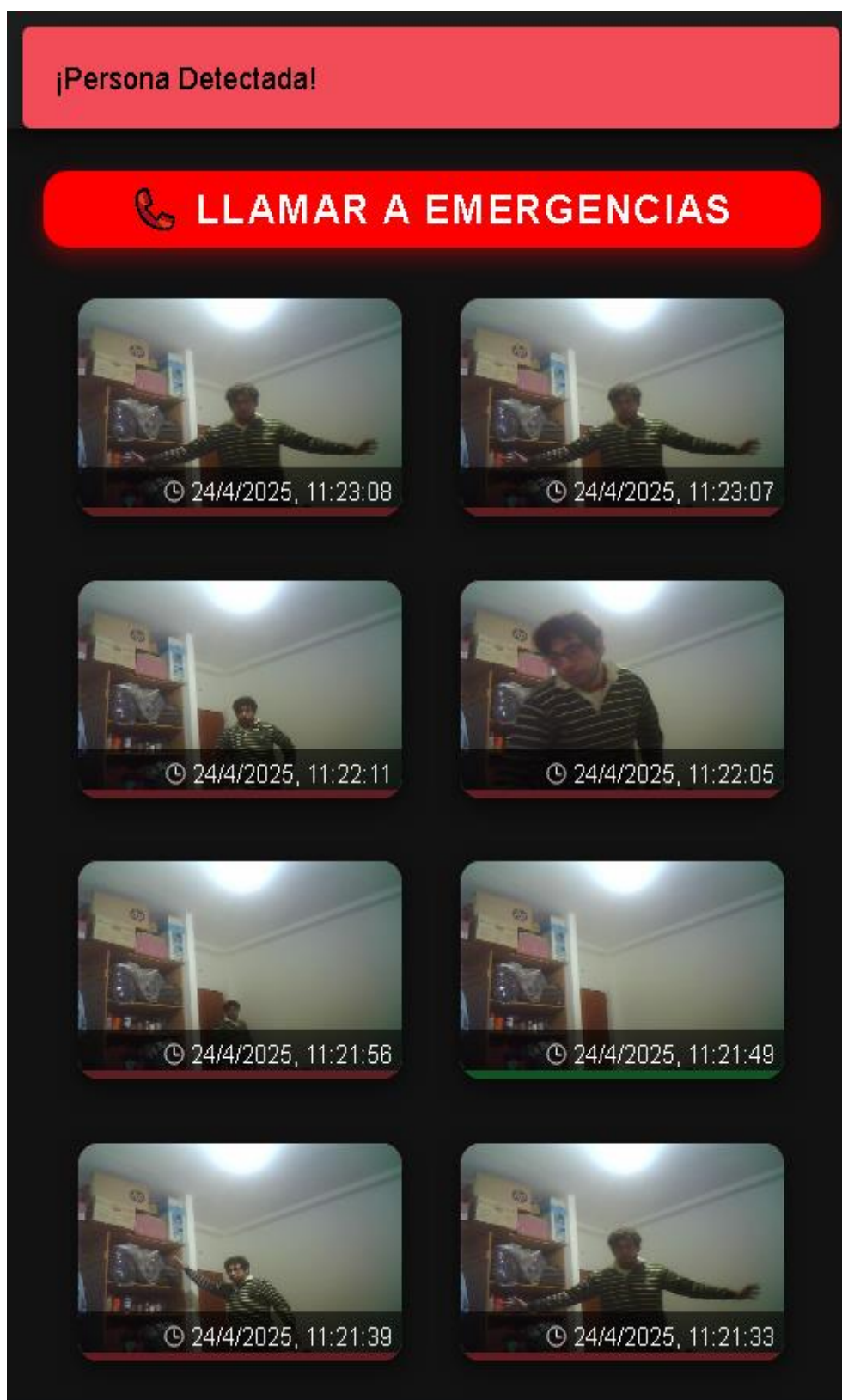


Ilustración 33 - Notificación de alerta