Sistema de Monitoreo y Alerta Temprana basado en Inteligencia Artificial para Áreas Protegidas

Autores:
Fabrizio Martin Contigiani
Gabriel Orlando Da Silva Schmies

Tutor:
Dr. Ing. Sergio Eduardo Moya

27 de diciembre de 2025

Resumen

El presente trabajo describe el diseño e implementación de un sistema de monitoreo y alerta temprana para áreas protegidas, que combina tecnologías de Internet de las Cosas (IoT) con Inteligencia Artificial (IA) para la detección automática de fauna silvestre, personas y vehículos.

El sistema está compuesto por una red de nodos de captura basados en microcontroladores ESP32 equipados con cámaras, los cuales se comunican mediante el protocolo ESP-MESH para transmitir imágenes hacia un nodo raíz. Este nodo actúa como puerta de enlace, reenviando las imágenes a un servidor remoto a través de Transmission Control Protocol (TCP)/IP.

En el servidor, las imágenes son procesadas por un servicio de inferencia basado en SpeciesNet, un modelo de detección de objetos desarrollado por Google que utiliza YOLOv5. Este modelo permite identificar y clasificar especies de fauna silvestre, así como detectar la presencia de humanos y vehículos, generando alertas automáticas ante posibles intrusiones.

La arquitectura del servidor incluye una aplicación web desarrollada en Django para la gestión de imágenes, un bot de Telegram para el envío de notificaciones en tiempo real, y una base de datos PostgreSQL para el almacenamiento persistente. Todo el sistema está contenedorizado mediante Docker para facilitar su despliegue.

Los resultados demuestran la viabilidad de implementar un sistema de vigilancia inteligente de bajo costo para áreas protegidas, capaz de operar de manera autónoma y alertar a los administradores ante eventos relevantes.

Palabras Clave - Cámaras Trampa, Internet de las Cosas, Inteligencia Artificial, Monitoreo de Fauna, Detección de Intrusos, ESP-MESH, SpeciesNet, YOLO

Índice general

Índice de figuras **5**

Índice de cuadros **6**

Glosario **7**

Siglas **8**

1. Introducción **9**
1.1. Contexto y motivación 9
1.2. Estructura del documento 9

2. Antecedentes **10**
2.1. Trabajos relacionados 10
2.2. Soluciones comerciales existentes 10
2.3. Estado del arte 10
2.4. Análisis comparativo 10

3. Planteamiento del Problema **11**
3.1. Áreas protegidas y conservación de fauna silvestre 11
3.2. Problemática de la vigilancia en áreas remotas 11
3.3. Sistemas de monitoreo tradicionales 11
3.3.1. Cámaras trampa convencionales 11
3.3.2. Limitaciones actuales 11
3.4. Necesidad de detección de intrusos 11
3.5. Justificación del proyecto 11

4. Objetivos y Alcance **12**
4.1. Objetivo general 12
4.2. Objetivos específicos 12
4.3. Alcance del proyecto 12
4.4. Limitaciones 12

5. Marco Teórico **13**
5.1. Internet de las Cosas (IoT) 13
5.1.1. Arquitecturas IoT 13
5.1.2. Protocolos de comunicación inalámbrica 13

5.2. Redes Mesh . . . 13
5.2.1. Topologías de red . . . 13
5.2.2. ESP-MESH y Mwifi . . . 13
5.3. Inteligencia Artificial aplicada a visión por computadora . . . 13
5.3.1. Redes neuronales convolucionales (CNN) . . . 13
5.3.2. Detección de objetos con YOLO . . . 13
5.3.3. SpeciesNet de Google . . . 13
5.4. Tecnologías de desarrollo . . . 13
5.4.1. Microcontroladores ESP32 . . . 13
5.4.2. ESP-IDF y ESP-MDF . . . 13
5.4.3. Contenedorización con Docker . . . 13
5.4.4. Framework Django . . . 13

6. Metodología **14**
6.1. Enfoque metodológico . . . 14
6.2. Etapas del desarrollo . . . 14
6.3. Herramientas y tecnologías utilizadas . . . 14
6.4. Métricas de evaluación . . . 14

7. Diseño del Sistema **15**
7.1. Arquitectura general . . . 16
7.2. Diseño del hardware . . . 16
7.2.1. Selección de componentes . . . 16
7.2.2. Nodo de captura con cámara . . . 16
7.2.3. Nodo raíz . . . 16
7.2.4. Alimentación y consumo energético . . . 16
7.3. Diseño de la red mesh . . . 16
7.3.1. Topología de la red . . . 16
7.3.2. Protocolo de comunicación . . . 16
7.3.3. Formato de datos . . . 16
7.4. Diseño del servicio de detección . . . 16
7.4.1. Servidor de inferencia con SpeciesNet . . . 16
7.4.2. Detección de animales, humanos y vehículos . . . 16
7.4.3. Anotación de imágenes con bounding boxes . . . 16
7.5. Diseño del servidor de aplicación . . . 16
7.5.1. Arquitectura de servicios . . . 16
7.5.2. Gestión de imágenes . . . 16
7.5.3. Interfaz web . . . 16
7.5.4. Bot de Telegram y sistema de alertas . . . 16

8. Implementación **17**
8.1. Nodo mesh (mesh-node) . . . 18
8.1.1. Firmware del nodo de captura . . . 18
8.1.2. Captura de imágenes . . . 18
8.1.3. Compresión y transmisión . . . 18
8.2. Nodo raíz (root-node) . . . 18
8.2.1. Firmware del nodo raíz . . . 18

8.2.2. Conexión con servidor TCP . . . 18
8.2.3. Gestión de la red mesh . . . 18
8.3. Servicio de detección (wildlife-detection) . . . 18
8.3.1. Contenedor Docker con SpeciesNet . . . 18
8.3.2. API de inferencia con LitServe . . . 18
8.3.3. Procesamiento de imágenes . . . 18
8.4. Servidor de aplicación (server) . . . 18
8.4.1. Aplicación Django . . . 18
8.4.2. Integración con SpeciesNet . . . 18
8.4.3. Bot de Telegram y sistema de alertas . . . 18
8.4.4. Base de datos PostgreSQL . . . 18
8.4.5. Despliegue con Docker Compose . . . 18

9. Pruebas y Resultados **19**
9.1. Ambiente de pruebas . . . 19
9.2. Pruebas de conectividad y red mesh . . . 19
9.2.1. Alcance de la red . . . 19
9.2.2. Latencia de transmisión . . . 19
9.2.3. Estabilidad de la conexión . . . 19
9.3. Pruebas de detección . . . 19
9.3.1. Detección de fauna silvestre . . . 19
9.3.2. Detección de humanos . . . 19
9.3.3. Detección de vehículos . . . 19
9.4. Evaluación del modelo de IA . . . 19
9.4.1. Precisión y recall . . . 19
9.4.2. Tiempo de inferencia . . . 19
9.5. Pruebas de consumo energético . . . 19
9.6. Pruebas del sistema de alertas . . . 19
9.6.1. Tiempo de respuesta . . . 19
9.7. Análisis de resultados . . . 19

10. Conclusiones **20**
10.1. Conclusiones generales . . . 20
10.2. Aportes del trabajo . . . 20
10.3. Trabajos futuros . . . 20
10.4. Recomendaciones . . . 20

Bibliografía **21**

A. Esquemáticos del hardware **22**

B. Código fuente relevante **23**
B.1. Firmware del nodo mesh . . . 23
B.2. Firmware del nodo raíz . . . 23
B.3. Servidor de detección . . . 23
B.4. Aplicación Django . . . 23

C. Manual de instalación y configuración **24**
C.1. Configuración del firmware 24
C.2. Despliegue del servidor 24
C.3. Configuración del bot de Telegram 24

D. Manual de usuario **25**

E. Especificaciones técnicas **26**

F. Análisis de viabilidad económica **27**

Índice de figuras

Índice de cuadros

Glosario

bot de Telegram Programa automatizado que interactúa con usuarios a través de la plataforma de mensajería Telegram. 1

Django Framework de desarrollo web de alto nivel escrito en Python, que sigue el patrón modelo-vista-plantilla. 1

Docker Plataforma de contenedorización que permite empaquetar aplicaciones junto con sus dependencias para facilitar el despliegue. 1

ESP-MESH Protocolo de red mesh desarrollado por Espressif para microcontroladores ESP32, basado en Wi-Fi. 1

ESP32 Microcontrolador de bajo costo y bajo consumo con Wi-Fi y Bluetooth integrados, fabricado por Espressif Systems. 1

inferencia Proceso de utilizar un modelo de aprendizaje automático entrenado para realizar predicciones sobre nuevos datos. 1

nodo Dispositivo individual que forma parte de una red mesh. 1

nodo raíz Nodo principal de una red mesh que actúa como puerta de enlace hacia redes externas. 1

PostgreSQL Sistema de gestión de bases de datos relacional de código abierto. 1

SpeciesNet Modelo de detección y clasificación de fauna silvestre desarrollado por Google, basado en YOLOv5. 1

YOLOv5 Versión 5 del modelo You Only Look Once, arquitectura de red neuronal para detección de objetos en tiempo real. 1

Siglas

IA Inteligencia Artificial. 1

IoT Internet de las Cosas. 1

TCP Transmission Control Protocol. 1

Capítulo 1

Introducción

[1]

1.1. Contexto y motivación

1.2. Estructura del documento

Capítulo 2

Antecedentes

2.1. Trabajos relacionados

2.2. Soluciones comerciales existentes

2.3. Estado del arte

2.4. Análisis comparativo

Capítulo 3

Planteamiento del Problema

3.1. Áreas protegidas y conservación de fauna silvestre

3.2. Problemática de la vigilancia en áreas remotas

3.3. Sistemas de monitoreo tradicionales

3.3.1. Cámaras trampa convencionales

3.3.2. Limitaciones actuales

3.4. Necesidad de detección de intrusos

3.5. Justificación del proyecto

Capítulo 4

Objetivos y Alcance

4.1. Objetivo general

4.2. Objetivos específicos

4.3. Alcance del proyecto

4.4. Limitaciones

Capítulo 5

Marco Teórico

5.1. Internet de las Cosas (IoT)

5.1.1. Arquitecturas IoT

5.1.2. Protocolos de comunicación inalámbrica

5.2. Redes Mesh

5.2.1. Topologías de red

5.2.2. ESP-MESH y Mwifi

5.3. Inteligencia Artificial aplicada a visión por computadora

5.3.1. Redes neuronales convolucionales (CNN)

5.3.2. Detección de objetos con YOLO

5.3.3. SpeciesNet de Google

5.4. Tecnologías de desarrollo

5.4.1. Microcontroladores ESP32

5.4.2. ESP-IDF y ESP-MDF

5.4.3. Contenedorización con Docker

5.4.4. Framework Django

Capítulo 6

Metodología

6.1. Enfoque metodológico

6.2. Etapas del desarrollo

6.3. Herramientas y tecnologías utilizadas

6.4. Métricas de evaluación

Capítulo 7

Diseño del Sistema

7.1. Arquitectura general

7.2. Diseño del hardware

7.2.1. Selección de componentes

7.2.2. Nodo de captura con cámara

7.2.3. Nodo raíz

7.2.4. Alimentación y consumo energético

7.3. Diseño de la red mesh

7.3.1. Topología de la red

7.3.2. Protocolo de comunicación

7.3.3. Formato de datos

7.4. Diseño del servicio de detección

7.4.1. Servidor de inferencia con SpeciesNet

7.4.2. Detección de animales, humanos y vehículos

7.4.3. Anotación de imágenes con bounding boxes

7.5. Diseño del servidor de aplicación

7.5.1. Arquitectura de servicios

7.5.2. Gestión de imágenes

7.5.3. Interfaz web

7.5.4. Bot de Telegram y sistema de alertas

Capítulo 8

Implementación

8.1. Nodo mesh (mesh-node)

8.1.1. Firmware del nodo de captura

8.1.2. Captura de imágenes

8.1.3. Compresión y transmisión

8.2. Nodo raíz (root-node)

8.2.1. Firmware del nodo raíz

8.2.2. Conexión con servidor TCP

8.2.3. Gestión de la red mesh

8.3. Servicio de detección (wildlife-detection)

8.3.1. Contenedor Docker con SpeciesNet

8.3.2. API de inferencia con LitServe

8.3.3. Procesamiento de imágenes

8.4. Servidor de aplicación (server)

8.4.1. Aplicación Django

8.4.2. Integración con SpeciesNet

8.4.3. Bot de Telegram y sistema de alertas

8.4.4. Base de datos PostgreSQL

8.4.5. Despliegue con Docker Compose

Capítulo 9

Pruebas y Resultados

9.1. Ambiente de pruebas

9.2. Pruebas de conectividad y red mesh

9.2.1. Alcance de la red

9.2.2. Latencia de transmisión

9.2.3. Estabilidad de la conexión

9.3. Pruebas de detección

9.3.1. Detección de fauna silvestre

9.3.2. Detección de humanos

9.3.3. Detección de vehículos

9.4. Evaluación del modelo de IA

9.4.1. Precisión y recall

9.4.2. Tiempo de inferencia

9.5. Pruebas de consumo energético

9.6. Pruebas del sistema de alertas

9.6.1. Tiempo de respuesta

9.7. Análisis de resultados

Capítulo 10

Conclusiones

10.1. Conclusiones generales

10.2. Aportes del trabajo

10.3. Trabajos futuros

10.4. Recomendaciones

Bibliografía

[1] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, págs. 779-788, 2016.

Apéndice A

Esquemáticos del hardware

Apéndice B

Código fuente relevante

B.1. Firmware del nodo mesh

B.2. Firmware del nodo raíz

B.3. Servidor de detección

B.4. Aplicación Django

Apéndice C

Manual de instalación y configuración

C.1. Configuración del firmware

C.2. Despliegue del servidor

C.3. Configuración del bot de Telegram

Apéndice D

Manual de usuario

Apéndice E

Especificaciones técnicas

Apéndice F

Análisis de viabilidad económica