

**UNIVERSIDAD NACIONAL DE ASUNCIÓN**

**Facultad de Ingeniería  
Ingeniería Mecatrónica**



**Aplicación de Deep Learning y Modelado Basado en Agentes  
para el Desarrollo de una Trampa Inteligente de Mosquitos**

Lucas Adrián Pin Ferreira

Matteo Nicolas Martinez Goetting

San Lorenzo - Paraguay

2026

### **Tutores del Presente Trabajo Final de Grado**

Dr. Ing. Diego Herbin Stalder Díaz

Dr. Ing. Christian Emilio Schaerer Serra

En última instancia, todo esfuerzo por imponer orden sobre lo complejo conlleva una cierta ilusión de dominio. Este trabajo no es ajeno a ello, y se inscribe en ese intento, tan necesario como inevitablemente insuficiente.

A nuestros padres, por sostener en silencio este intento, inevitablemente insuficiente, y aun así permanecer en él, incluso cuando su sentido y su tiempo no eran claros.

## **Agradecimientos y Reconocimientos**

La realización de este trabajo no habría sido posible sin el acompañamiento, la guía y la confianza de muchas personas e instituciones que, de distintas maneras, dejaron una huella en este recorrido. A todas ellas, expresamos nuestro más sincero agradecimiento.

En primer lugar, a nuestros tutores, Diego Stalder y Christian Schaerer, por haber guiado y moldeado este proyecto desde sus primeras ideas hasta su forma final. Más allá de la orientación académica y técnica, nos enseñaron el valor del rigor, la paciencia y la búsqueda constante de calidad en cada etapa del trabajo. Cada uno, desde perspectivas distintas pero complementarias, aportó una visión fundamental que enriqueció profundamente este proceso y dejó una marca duradera en nuestra formación.

A Alejandro Reckziegel, por su permanente predisposición y por acompañar el nacimiento de las primeras bases del modelo basado en agentes. Sus aportes fueron esenciales para transformar una idea inicial en una línea de investigación con dirección, estructura y propósito.

A Nilsa González, María Ferreira y Cinthya Rodríguez, del Instituto de Investigaciones en Ciencias de la Salud, por su invaluable apoyo desde el área biológica y experimental. Su generosidad al compartir conocimientos, facilitar las muestras necesarias para las pruebas y acompañar nuestras consultas técnicas permitió dar sustento y validación al prototipo desarrollado. En un trabajo que buscó tender puentes entre distintas áreas del conocimiento, su colaboración fue fundamental.

Nuestro más sincero reconocimiento a Claudia Ferreira y a Ana Pontes, de la Universidade Estadual Paulista, quienes acompañaron este proyecto incluso a



**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

la distancia, contribuyendo de manera decisiva a la maduración de los modelos desarrollados. Desde las primeras versiones hasta el modelo final expuesto en este trabajo, su orientación científica ayudó a convertir intuiciones iniciales en una propuesta mucho más sólida, consistente y consciente de sus propios alcances.

Al Laboratorio de Mecánica y Energía, agradecemos por habernos acogido, por permitirnos formar parte de una línea activa de investigación y por brindarnos acceso a sus instalaciones y recursos. Su apoyo creó el espacio donde muchas de las ideas aquí desarrolladas pudieron crecer, equivocarse, reformularse y finalmente tomar forma.

Finalmente, expresamos nuestra profunda gratitud a las instituciones y fuentes de financiamiento que sostienen e impulsan la investigación científica. A CONACYT Paraguay, en el marco del proyecto FEEI-CONACYT-PROCIENCIA – ESTR01-23 ARASY, y al Consortium Wolbloc – Wellcome Trust (Project No. 226166/Z/22/Z), cuyo apoyo hace posible no solo el desarrollo de este trabajo, sino también el crecimiento de múltiples iniciativas orientadas al estudio, monitoreo y control de enfermedades transmitidas por mosquitos.

Esta tesis representa apenas una pequeña rama dentro de un esfuerzo científico mucho mayor, construido colectivamente por personas comprometidas con la generación de conocimiento y su impacto en la sociedad. Detrás de cada experimento, cada modelo y cada resultado, existe también una red de confianza, colaboración y trabajo compartido que hizo posible llegar hasta aquí.

**Lucas y Matteo**

## Agradecimientos Personales

Cerrar esta etapa me deja un sentimiento difícil de describir. Agradezco las infinitas variables que se configuraron para traerme hasta aquí. Siento una enorme fortuna: la extraña constante de que las cosas simplemente salían bien, sin comprender el porqué ni si realmente lo merecía. Hubo momentos donde el universo parecía intervenir, Dios o los Dioses me otorgaban una epifanía o los duendes hacían su magia para corregir el rumbo, y admito que no siempre fui consciente de esa gracia.

Hoy reconozco algunas de las principales variables: mis padres, mi hermana, mi familia, mis amigos, mis mascotas, por el vital hecho de ser mi soporte silencioso; y mi compañero de trabajo, a quien agradezco su dedicación exclusiva en este desafío compartido. Pero también están las variables indirectas, como el ir y venir de rostros anónimos en la calle o los pajaritos en la ventana. El simple cantar de un ave podía cambiar mi ánimo en un instante e impulsarme a algo distinto; sutiles desviaciones que podrían terminar por recalibrar todo el camino. Y debo admitir que no puedo parar de sentirme como un hereje con este trabajo; siempre cuestioné el intentar comprender la complejidad de la realidad con métodos inherentemente imperfectos, y ahora estoy aquí, tratando de ofrecer una más de estas herramientas imperfectas.

Estas páginas presentan ese intento de aporte; uno que, sin la ayuda de todas las personas, animales, instituciones y entidades que me rodearon, no habría sido posible. Un trabajo imperfecto, como imperfecta es toda herramienta que intenta tocar la realidad. Quisiera haber seguido hasta alcanzar esa perfección inalcanzable, pero este límite se ha cumplido. Y los límites, cuando se alcanzan, no dejan más opción que seguir.

**Ne reflectas ubi ad terminos perveneris.**

**Lucas**

## Agradecimientos Personales

A mi familia, por ser mi pilar fundamental y mi refugio constante. A mis padres, por su amor incondicional y por haberme enseñado con el ejemplo el valor del esfuerzo, la honestidad y la perseverancia. A mis hermanas, por su apoyo y por celebrar conmigo cada pequeño logro. A mis mascotas que me acompañan desde la tierra y el cielo. Todo lo que soy y todo lo que he construido se sostiene sobre la confianza que han depositado en mí.

A mi gran amigo y compañero de tesis Lucas, por haber estado conmigo en la mayoría de las batallas de esta carrera y, sobre todo, en este último gran reto. Tu inteligencia, resiliencia y lealtad han sido fundamentales en todo este proceso; no habría podido pedir un mejor equipo para sacar este proyecto adelante.

A mis amigos de la FIUNA, por haberme acompañado durante el camino hasta aquí, por todas esas jornadas de estudio, los trabajos entregados a última hora y por cuidar siempre de mí en el proceso universitario.

A mis amigos de la vida, esa familia que uno tiene la fortuna de elegir. Gracias por estar siempre ahí, por las risas compartidas, las charlas sinceras y los consejos que me ayudaron a mantener los pies en la tierra. Su apoyo incondicional ha sido mi ancla en los momentos de duda y mi mayor celebración en los momentos de éxito; gracias por recordarme siempre lo que realmente importa.

***"No todo el que vaga está perdido, y el hogar está detrás, el mundo por delante, y hay muchos caminos por andar a través de las sombras hasta el borde de la noche, hasta que las estrellas se enciendan."*** - J.R.R. Tolkien

**Matteo**

## Índice general

|                                                                                                 |           |
|-------------------------------------------------------------------------------------------------|-----------|
| <b>1. Introducción</b>                                                                          | <b>20</b> |
| 1.1. Planteamiento del problema . . . . .                                                       | 21        |
| 1.2. Justificación . . . . .                                                                    | 23        |
| 1.3. Objetivos . . . . .                                                                        | 25        |
| 1.3.1. Objetivo General . . . . .                                                               | 25        |
| 1.3.2. Objetivos Específicos . . . . .                                                          | 25        |
| 1.4. Alcance y limitaciones . . . . .                                                           | 26        |
| 1.4.1. Alcance . . . . .                                                                        | 26        |
| 1.4.2. Limitaciones . . . . .                                                                   | 27        |
| <b>2. Revisión de la literatura</b>                                                             | <b>28</b> |
| 2.1. Marco Teórico . . . . .                                                                    | 28        |
| 2.1.1. Conceptos para el diseño de sistemas inteligentes de cap-<br>tura de mosquitos . . . . . | 28        |
| 2.1.2. Conceptos para el Modelado Basado en Agentes (ABM) . .                                   | 40        |
| 2.2. Estado del Arte . . . . .                                                                  | 43        |
| 2.2.1. Antecedentes en Trampas Inteligentes para Mosquitos . . .                                | 44        |
| 2.2.2. Antecedentes en Modelos Espacio-Temporales de Mosquitos                                  | 47        |
| 2.2.3. Trabajos Relacionados en la FIUNA . . . . .                                              | 49        |
| <b>3. Metodología</b>                                                                           | <b>52</b> |
| 3.1. Arquitectura General del Proyecto . . . . .                                                | 53        |
| 3.2. Fases de Desarrollo . . . . .                                                              | 54        |
| 3.2.1. Fases de Desarrollo de la Trampa Inteligente . . . . .                                   | 54        |
| 3.2.2. Fases de Desarrollo del Modelo Basado en Agentes . . . .                                 | 56        |
| 3.3. Proyección de Integración Futura . . . . .                                                 | 57        |

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

|                                                                |           |
|----------------------------------------------------------------|-----------|
| <b>4. Diseño y Hardware de la trampa</b>                       | <b>59</b> |
| 4.1. Materiales y Preparación del Cebo . . . . .               | 59        |
| 4.1.1. Materiales . . . . .                                    | 60        |
| 4.1.2. Proceso de Elaboración . . . . .                        | 60        |
| 4.1.3. Liberación de CO <sub>2</sub> del Gel . . . . .         | 62        |
| 4.2. Diseño y Modelo de la Trampa . . . . .                    | 62        |
| 4.2.1. Mecanismo de la Trampa y su Funcionamiento . . . . .    | 63        |
| 4.2.2. Modelo 3D . . . . .                                     | 67        |
| 4.3. Componentes Utilizados . . . . .                          | 70        |
| 4.3.1. Lista de Componentes . . . . .                          | 70        |
| 4.3.2. Diagrama de Bloques: Arquitectura de Hardware . . . . . | 75        |
| 4.3.3. Diagrama de Flujo: Lógica y Flujo del Proceso . . . . . | 76        |
| 4.3.4. Diagrama Eléctrico . . . . .                            | 78        |
| 4.4. Lista Final de Precios . . . . .                          | 80        |
| <b>5. Modelo de Detección y Clasificación</b>                  | <b>83</b> |
| 5.1. Modelo de Detección (YOLO) . . . . .                      | 84        |
| 5.1.1. Dataset utilizado . . . . .                             | 84        |
| 5.1.2. Búsqueda de hiperparámetros . . . . .                   | 86        |
| 5.1.3. Proceso de entrenamiento . . . . .                      | 89        |
| 5.1.4. Resultados y métricas del modelo Pytorch . . . . .      | 90        |
| 5.1.5. Exportación del modelo a ONNX . . . . .                 | 94        |
| 5.1.6. Comparación del modelo PyTorch vs ONNX . . . . .        | 96        |
| 5.2. Modelo de Clasificación (MobileNet) . . . . .             | 98        |
| 5.2.1. Dataset utilizado . . . . .                             | 99        |
| 5.2.2. Proceso de entrenamiento . . . . .                      | 103       |
| 5.2.3. Resultados y métricas del modelo PyTorch . . . . .      | 105       |
| 5.2.4. Exportación del modelo a ONNX . . . . .                 | 106       |

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

|                                                                   |            |
|-------------------------------------------------------------------|------------|
| 5.2.5. Comparación del modelo PyTorch vs ONNX . . . . .           | 109        |
| 5.3. Pipeline del modelo en cascada . . . . .                     | 111        |
| 5.3.1. Flujo de inferencia . . . . .                              | 112        |
| 5.3.2. Contexto y metodología de evaluación . . . . .             | 113        |
| 5.3.3. Evaluación del sistema en cascada . . . . .                | 114        |
| 5.3.4. Análisis de resultados . . . . .                           | 115        |
| 5.3.5. Tiempos de inferencia en el dispositivo de borde . . . . . | 117        |
| <b>6. Arquitectura IoT Implementada</b>                           | <b>119</b> |
| 6.1. Arquitectura Cliente-Servidor . . . . .                      | 119        |
| 6.2. Capa de Dispositivo de Borde . . . . .                       | 120        |
| 6.2.1. Pipeline de procesamiento en tiempo real . . . . .         | 121        |
| 6.2.2. Máquina de estados de la trampa . . . . .                  | 121        |
| 6.2.3. Consideraciones de diseño . . . . .                        | 121        |
| 6.3. Capa de Servidor (Backend) . . . . .                         | 122        |
| 6.3.1. Gestión de configuración y servicios externos . . . . .    | 122        |
| 6.3.2. Estructura de la API REST . . . . .                        | 123        |
| 6.3.3. Ingesta de datos IoT . . . . .                             | 123        |
| 6.3.4. Diseño concurrente y escalabilidad . . . . .               | 124        |
| 6.4. Protocolos de Comunicación del Cliente de Borde . . . . .    | 124        |
| 6.4.1. Nivel 1 — Comunicación Interna . . . . .                   | 126        |
| 6.4.2. Nivel 2 — Comunicación Serie (UART) . . . . .              | 127        |
| 6.4.3. Nivel 3 — Comunicación HTTP . . . . .                      | 127        |
| 6.5. Capa de Presentación (Frontend) . . . . .                    | 128        |
| 6.6. Servicios para la Base de Datos . . . . .                    | 130        |
| 6.6.1. Persistencia Relacional (PostgreSQL) . . . . .             | 131        |
| 6.6.2. Almacenamiento de Objetos . . . . .                        | 132        |
| 6.7. Despliegue en la Nube . . . . .                              | 133        |

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

|                                                                        |            |
|------------------------------------------------------------------------|------------|
| 6.7.1. Despliegue del Servidor (Backend) . . . . .                     | 134        |
| 6.7.2. Despliegue del Frontend . . . . .                               | 135        |
| 6.8. Consideraciones de Seguridad . . . . .                            | 136        |
| <b>7. Generación del Grid Map a partir de datos GIS</b>                | <b>137</b> |
| 7.1. Definición del área de estudio . . . . .                          | 137        |
| 7.2. Descarga de datos desde OpenStreetMap . . . . .                   | 139        |
| 7.2.1. OpenStreetMap . . . . .                                         | 139        |
| 7.2.2. Acceso a los datos mediante OSMnx . . . . .                     | 139        |
| 7.3. Conversión del mapa continuo a una grilla discreta . . . . .      | 141        |
| 7.3.1. Estructura de la grilla . . . . .                               | 141        |
| 7.3.2. Rasterización de polígonos . . . . .                            | 141        |
| 7.3.3. Rasterización de la red vial . . . . .                          | 142        |
| 7.4. Clasificación del entorno urbano . . . . .                        | 143        |
| 7.5. Integración de las trampas reales . . . . .                       | 144        |
| 7.5.1. Origen de los datos . . . . .                                   | 144        |
| 7.5.2. Preprocesamiento y limpieza de coordenadas . . . . .            | 145        |
| 7.5.3. Transformación de coordenadas a índices de grilla . . . . .     | 145        |
| 7.5.4. Validación y registro . . . . .                                 | 146        |
| 7.5.5. Archivos de salida . . . . .                                    | 147        |
| 7.6. Resultado: comparación entre imagen satelital y grid generado . . | 147        |
| <b>8. Modelado y Simulación de Poblaciones de Mosquitos (ABM)</b>      | <b>149</b> |
| 8.1. Entorno de simulación . . . . .                                   | 151        |
| 8.2. Primer modelo . . . . .                                           | 151        |
| 8.2.1. Base matemática: sistema ODE de referencia . . . . .            | 152        |
| 8.2.2. Temperatura ambiental y parámetros biológicos . . . . .         | 153        |
| 8.2.3. Campo de potencial . . . . .                                    | 154        |
| 8.2.4. Dinámica espacial de adultos . . . . .                          | 157        |

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

|                                                                     |            |
|---------------------------------------------------------------------|------------|
| 8.2.5. Dinámica temporal de juveniles y adultos . . . . .           | 158        |
| 8.2.6. Generación de juveniles . . . . .                            | 158        |
| 8.3. Segundo modelo . . . . .                                       | 160        |
| 8.3.1. Campo de potencial . . . . .                                 | 161        |
| 8.3.2. Descripción de los agentes . . . . .                         | 162        |
| 8.3.3. Dinámica de juveniles . . . . .                              | 163        |
| 8.3.4. Dinámica de adultos y movimiento estocástico . . . . .       | 165        |
| 8.3.5. Dinámica de apareamiento y generación de juveniles . . . . . | 166        |
| 8.3.6. Orquestador temporal de la simulación . . . . .              | 167        |
| <b>9. Resultados</b>                                                | <b>171</b> |
| 9.1. Montaje Final de la Trampa . . . . .                           | 171        |
| 9.1.1. Prototipo Final de la trampa . . . . .                       | 171        |
| 9.1.2. Pruebas de Semi-Campo . . . . .                              | 175        |
| 9.1.3. Prueba de Campo . . . . .                                    | 185        |
| 9.2. Modelado Basado en Agentes . . . . .                           | 189        |
| 9.2.1. Simulación y Resultados con el Primer Modelo . . . . .       | 189        |
| 9.2.2. Simulación y Resultados con el Segundo Modelo . . . . .      | 194        |
| 9.2.3. Análisis de Sensibilidad del Segundo Modelo . . . . .        | 198        |
| <b>10. Conclusiones y Recomendaciones</b>                           | <b>204</b> |
| 10.1. Conclusiones . . . . .                                        | 204        |
| 10.1.1. Síntesis de Aportes . . . . .                               | 205        |
| 10.1.2. Reflexiones sobre el proceso de desarrollo . . . . .        | 206        |
| 10.2. Recomendaciones para Trabajo Futuro . . . . .                 | 209        |
| 10.2.1. Recomendaciones para la Trampa . . . . .                    | 209        |
| 10.2.2. Recomendaciones para el Modelo Basado en Agentes . . . . .  | 210        |
| 10.2.3. Integración de Ambas Partes . . . . .                       | 212        |



## Índice de figuras

|                                                                                                                                                                                                                                                                                                                                                                                                 |    |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1. Dispositivo BG-Counter (Biogents AG). <b>Fuente:</b> <i>Biogents AG, BG-Counter: Innovative and Autonomous Mosquito Trap Station.</i> . . . .                                                                                                                                                                                                                                              | 29 |
| 2.2. Sistema generador de CO <sub>2</sub> basado en gel utilizado como atrayente químico en trampas para mosquitos. <b>Fuente:</b> <i>Gel CO<sub>2</sub>, Jerrican de 5L, Levure. Borne Lami Anti-Moustiques.</i> . . . . .                                                                                                                                                                     | 30 |
| 2.3. Esquema conceptual simplificado del funcionamiento de YOLO para detección de objetos. La imagen de entrada es dividida en una grilla espacial, donde la red predice simultáneamente cajas delimitadoras, probabilidades de clase y niveles de confianza para cada región de la escena. <b>Fuente:</b> <i>How to (Actually) Easily Detect Objects with Deep Learning, Medium.</i> . . . . . | 34 |
| 2.4. Esquema de interoperabilidad ONNX: flujo desde frameworks de entrenamiento hacia plataformas de despliegue y hardware. <b>Fuente:</b> <i>ONNX and AI at the Edge Documentation, Microsoft AI at Edge.</i>                                                                                                                                                                                  | 36 |
| 3.1. Arquitectura general del proyecto, mostrando las dos líneas de desarrollo paralelas (Trampa Inteligente y Modelo ABM) y su convergencia hacia la integración futura del sistema. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                                | 53 |
| 4.1. Modelo 3D de la trampa con descripción de sus partes principales. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                                                                                                                                               | 68 |
| 4.2. Vista trasera del modelo 3D de la trampa, donde se aprecian el soporte de montaje del mecanismo de clasificación y la salida del canal de ventilación. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                                                          | 69 |

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

---

|                                                                                                                                                                                                                                                                                                           |    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.3. Modelo 3D de la trampa integrada dentro de la caja protectora, mostrando la disposición del mecanismo de captura, el sistema de visión artificial y el soporte para la Raspberry Pi 5. <b>Fuente:</b> Elaboración propia. . . . .                                                                    | 70 |
| 4.4. Diagrama de bloques de la arquitectura de hardware de la trampa, organizado en cuatro bloques funcionales: procesamiento maestro (Raspberry Pi 5), control esclavo (Arduino UNO), gestión de potencia (LM2596) e interfaz física (sensores y actuadores). <b>Fuente:</b> Elaboración propia. . . . . | 76 |
| 4.5. Diagrama de flujo físico del mosquito en sincronía con el software. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                                                       | 77 |
| 4.6. Diagrama eléctrico de los componentes de la trampa. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                                                                       | 80 |
| 5.1. Vista general del dataset de detección en la plataforma Roboflow, compuesto por 22.309 imágenes anotadas (clase única: <i>mosquito</i> ). <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                 | 85 |
| 5.2. Historial de optimización de hiperparámetros con Optuna sobre 100 trials. El eje vertical representa la métrica mAP@50 sobre el conjunto de validación; el mejor resultado ( $\text{mAP@50} = 0,9633$ ) se alcanzó en el trial 43. <b>Fuente:</b> Elaboración propia. . . . .                        | 88 |
| 5.3. Importancia relativa de los hiperparámetros estimada por Optuna a partir de los 100 trials de búsqueda. Los valores representan la contribución de cada parámetro a la variación de la métrica mAP@50 sobre el conjunto de validación. <b>Fuente:</b> Elaboración propia. . . . .                    | 88 |
| 5.4. Curvas de pérdida y métricas de desempeño del modelo YOLOv8 durante 200 épocas de entrenamiento. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                          | 91 |

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

---

|                                                                                                                                                                                                                                                                                                                   |     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.5. Matriz de confusión del modelo de detección. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                                                                                      | 92  |
| 5.6. Ejemplo de predicciones del modelo YOLOv8 sobre un lote del conjunto de validación. Las cajas delimitadoras muestran las detecciones de la clase <i>mosquito</i> junto con el nivel de confianza asociado, sobre imágenes no utilizadas durante el entrenamiento. <b>Fuente:</b> Elaboración propia. . . . . | 93  |
| 5.7. Comparación de tiempos de inferencia sobre una muestra entre YOLOv8 (GPU) y el modelo exportado a ONNX (CPU). <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                     | 98  |
| 5.8. Vista general del dataset de clasificación en la plataforma Roboflow, compuesto por 2.571 imágenes. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                               | 100 |
| 5.9. Comparación de tiempos de inferencia entre el modelo PyTorch (ejecutado en GPU) y ONNX (ejecutado en CPU). <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                        | 111 |
| 5.10. Diagrama de bloques del pipeline en cascada: la imagen de entrada es procesada primero por YOLOv8 (detección), y cada región detectada pasa al clasificador MobileNetV3 (clasificación por sexo). <b>Fuente:</b> Elaboración propia. . . . .                                                                | 112 |
| 5.11. Ejemplo de salida del modelo en cascada con etiquetas de clase ( <i>Female/Male</i> ) y nivel de confianza asociado. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                             | 117 |
| 6.1. Diagrama de flujo del protocolo de comunicación IoT. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                                                                              | 126 |
| 6.2. Página de inicio de la aplicación web del sistema, con acceso a las secciones de dashboard, modelo ABM y documentación del proyecto. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                              | 129 |

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

---

|                                                                                                                                                                                                                                                                                                                                              |     |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 6.3. Dashboard de monitoreo IoT mostrando detecciones en tiempo casi real, clasificación por sexo y variables ambientales registradas por el sensor BME280. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                       | 130 |
| 6.4. Vista de la tabla <code>yolo_data</code> en la base de datos PostgreSQL gestionada por Neon, con registros de detección, metadatos ambientales y URL de imagen asociada. <b>Fuente:</b> Elaboración propia. . .                                                                                                                         | 132 |
| 6.5. Bucket <code>Mosquitoes</code> en Supabase Storage con las imágenes capturadas por el dispositivo de borde, cada una vinculada a su registro en la base de datos mediante URL pública. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                       | 133 |
| 6.6. Panel de control del entorno de despliegue del backend en Railway, mostrando el estado operativo del servicio durante las pruebas del sistema. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                               | 135 |
| 6.7. Panel de control del entorno de despliegue del frontend en Vercel, mostrando el estado del último despliegue exitoso durante las pruebas del sistema. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                        | 136 |
| 7.1. Comparación entre la imagen satelital y el Grid Map discreto generado a partir de datos de OpenStreetMap. Cada celda del grid representa una zona de $3 \times 3$ m del territorio real. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                     | 148 |
| 8.1. Del territorio real al campo de potencial. <b>(a)</b> imagen satelital de referencia; <b>(b)</b> Grid Map discreto georreferenciado generado desde OpenStreetMap; <b>(c)</b> campo de potencial $\Pi(x)$ resultante, que guía el movimiento de los agentes adultos hacia los sitios de cría. <b>Fuente:</b> Elaboración propia. . . . . | 156 |
| 8.2. Probabilidades de transición de cada celda en la vecindad para un agente en un paso de tiempo. <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                                                                               | 157 |

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

- 9.1. Trampa en la caja protectora con la puerta abierta, mostrando la disposición interna de los componentes. **Fuente:** Elaboración propia. 172
- 9.2. Trampa en la caja protectora con la puerta cerrada durante su operación normal. **Fuente:** Elaboración propia. . . . . 172
- 9.3. Mecanismo de clasificación rotativo encargado de la separación de los mosquitos detectados. **Fuente:** Elaboración propia. . . . . 173
- 9.4. Módulos electrónicos de control, procesamiento y comunicación integrados en la trampa, incluyendo los circuitos de alimentación, control de actuadores y el computador embebido para la ejecución del modelo de visión artificial. **Fuente:** Elaboración propia. . . . . 173
- 9.5. Cebo utilizado para la atracción de mosquitos, dispuesto en su contenedor dentro de la trampa. **Fuente:** Elaboración propia. . . . . 174
- 9.6. Trampa funcionando con un mosquito de prueba. El monitor conectado al Raspberry Pi permitió visualizar la detección en tiempo real. **Fuente:** Elaboración propia. . . . . 175
- 9.7. Esquema del recinto utilizado para las pruebas de semi-campo, con la disposición de la trampa y el área de liberación de los mosquitos. **Fuente:** Elaboración propia. . . . . 176
- 9.8. Vista del recinto de experimentación con la trampa colocada en una esquina del laboratorio durante la prueba de semi-campo. **Fuente:** Elaboración propia. . . . . 177
- 9.9. Trampa posicionada en la esquina del recinto de experimentación junto al agua azucarada dispuesta como fuente de alimentación. **Fuente:** Elaboración propia. . . . . 177
- 9.10. Mosquitos de la especie *Aedes aegypti* proporcionados por el IICS para las pruebas de semi-campo, en el contenedor de transporte con agua azucarada. **Fuente:** Elaboración propia. . . . . 178

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

---

|                                                                                                                                                                                                                                                                                                |     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 9.11. Distribución temporal de las detecciones registradas durante la primera prueba de semi-campo (36 horas). <b>Fuente:</b> Elaboración propia.                                                                                                                                              | 182 |
| 9.12. Distribución temporal de las detecciones registradas durante la segunda prueba de semi-campo (36 horas). <b>Fuente:</b> Elaboración propia. . . . .                                                                                                                                      | 183 |
| 9.13. Localización satelital del punto de muestreo en el Barrio Ycuá Satí. <b>Fuente:</b> Google Maps (2026). . . . .                                                                                                                                                                          | 186 |
| 9.14. Dashboard de monitoreo IoT durante la prueba de campo (1–7 de mayo), mostrando detecciones automáticas con clasificación por sexo, variables ambientales (temperatura y humedad, sensor BME280) e imagen asociada a cada evento de detección. <b>Fuente:</b> Elaboración propia. . . . . | 187 |
| 9.15. Capturas de la simulación en cuatro instantes temporales representativos. Los cuadrados magenta representan los 15 sitios de cría sintéticos; los puntos negros indican la posición de los adultos en cada instante. <b>Fuente:</b> Elaboración propia. . . . .                          | 191 |
| 9.16. Evolución de la población de juveniles en los 15 sitios de cría a lo largo de 5 000 pasos de tiempo. Las coordenadas (en celdas) de cada sitio se indican en la leyenda. <b>Fuente:</b> Elaboración propia. .                                                                            | 192 |
| 9.17. Media $\pm$ desviación estándar de la dinámica poblacional global a lo largo del tiempo, calculadas sobre seis corridas independientes con agentes y sitios de cría distribuidos aleatoriamente en cada ejecución. <b>Fuente:</b> Elaboración propia. . . . .                            | 193 |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                     |     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 9.18. Dinámica poblacional de <i>Aedes aegypti</i> a lo largo de la simulación. Cada curva representa la evolución temporal del número de agentes en cada estado: Juvenil ( $J$ ), Macho adulto ( $M$ ), Hembra no apareada ( $F_U$ ) y Hembra grávida ( $F_G$ ). Los parámetros del modelo corresponden a la Tabla 8.3 (Capítulo 8); los parámetros del entorno corresponden a la Tabla 9.11. <b>Fuente:</b> Elaboración propia.                   | 196 |
| 9.19. Distribución espacial final de los agentes sobre la grilla del entorno urbano. Cada punto representa un agente individual codificado por color según su estado ( $J$ , $M$ , $F_U$ , $F_G$ ). El fondo corresponde al Grid Map clasificado a partir de datos de OpenStreetMap ( $333 \times 333$ celdas, $\Delta s = 3$ m). Los puntos magenta indican las posiciones de las 7 trampas físicas integradas. <b>Fuente:</b> Elaboración propia. | 197 |
| 9.20. Heatmap de la matriz PRCC completa (13 parámetros $\times$ 5 QoI). Los valores en rojo indican efectos negativos (el parámetro reduce la QoI); en azul, efectos positivos. La intensidad del color es proporcional a $ \text{PRCC} $ . <b>Fuente:</b> Elaboración propia.                                                                                                                                                                     | 201 |
| 9.21. Diagrama de tornado de los índices PRCC sobre $N_{\text{final}}$ . Las barras rojas indican efectos negativos; las barras azules, efectos positivos. Las líneas verticales punteadas marcan los umbrales $ \text{PRCC}  = 0,5$ (influencia alta). <b>Fuente:</b> Elaboración propia.                                                                                                                                                          | 201 |
| 10.1. Proyección de integración futura entre el sistema de trampas inteligentes y el modelo basado en agentes. Los pasos naranja corresponden al componente IoT/trampa; los pasos verdes, al componente ABM. <b>Fuente:</b> Elaboración propia.                                                                                                                                                                                                     | 212 |
| A.1. Código QR de acceso al repositorio del proyecto. <b>Fuente:</b> Elaboración propia.                                                                                                                                                                                                                                                                                                                                                            | 222 |

## Índice de tablas

|                                                                                                                                                                                    |    |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1. Comparativa de trabajos relacionados en trampas inteligentes para mosquitos. . . . .                                                                                          | 46 |
| 2.2. Comparativa de trabajos en modelado espacio-temporal de poblaciones de mosquitos. . . . .                                                                                     | 48 |
| 2.3. Comparativa de trabajos relacionados desarrollados en la Facultad de Ingeniería de la Universidad Nacional de Asunción (FIUNA).<br><b>Fuente:</b> Elaboración propia. . . . . | 51 |
| 4.1. Materiales e ingredientes para un volumen de 1,5 L. . . . .                                                                                                                   | 60 |
| 4.2. Especificaciones técnicas de la Raspberry Pi 5. . . . .                                                                                                                       | 71 |
| 4.3. Especificaciones técnicas de la Raspberry Pi Camera Module 3<br>NoIR. . . . .                                                                                                 | 72 |
| 4.4. Especificaciones técnicas del Arduino UNO R3 . . . . .                                                                                                                        | 72 |
| 4.5. Especificaciones técnicas del servomotor . . . . .                                                                                                                            | 72 |
| 4.6. Especificaciones técnicas del Regulador Buck LM2596 . . . . .                                                                                                                 | 73 |
| 4.7. Especificaciones técnicas del Sensor BME280 . . . . .                                                                                                                         | 73 |
| 4.8. Especificaciones técnicas del Relé 5V de 1 Canal . . . . .                                                                                                                    | 74 |
| 4.9. Especificaciones técnicas del Sensor Magnético Hall A3144 . . . . .                                                                                                           | 74 |
| 4.10. Presupuesto de materiales clasificado por categoría . . . . .                                                                                                                | 81 |
| 5.1. Estadísticas del dataset utilizado. . . . .                                                                                                                                   | 85 |
| 5.2. Métricas de desempeño del modelo YOLOv8. . . . .                                                                                                                              | 91 |
| 5.3. Resultados derivados de la matriz de confusión. . . . .                                                                                                                       | 92 |
| 5.4. Información general del modelo ONNX. . . . .                                                                                                                                  | 95 |
| 5.5. Estructura de entrada del modelo ONNX. . . . .                                                                                                                                | 95 |
| 5.6. Estructura de salida del modelo ONNX. . . . .                                                                                                                                 | 95 |
| 5.7. Resultados del modelo ONNX. . . . .                                                                                                                                           | 96 |



**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

|                                                                                                                 |     |
|-----------------------------------------------------------------------------------------------------------------|-----|
| 5.8. Comparación de desempeño entre modelos YOLOv8 en PyTorch y ONNX. . . . .                                   | 97  |
| 5.9. Estadísticas del dataset original de clasificación. . . . .                                                | 101 |
| 5.10. Distribución del dataset de clasificación generado. . . . .                                               | 102 |
| 5.11. Análisis de dimensiones de las imágenes del dataset de clasificación. . . . .                             | 102 |
| 5.12. Reporte de clasificación del modelo MobileNetV3. . . . .                                                  | 105 |
| 5.13. Métricas globales del modelo de clasificación. . . . .                                                    | 105 |
| 5.14. Matriz de confusión del modelo de clasificación. . . . .                                                  | 106 |
| 5.15. Información general del modelo ONNX de clasificación. . . . .                                             | 108 |
| 5.16. Estructura de entrada del modelo ONNX de clasificación. . . . .                                           | 108 |
| 5.17. Estructura de salida del modelo ONNX de clasificación. . . . .                                            | 108 |
| 5.18. Comparación de desempeño entre los modelos MobileNetV3 en PyTorch y ONNX ejecutados en CPU y GPU. . . . . | 109 |
| 5.19. Resultados del modelo en cascada sobre el conjunto de prueba del clasificador. . . . .                    | 115 |
| 5.20. Tiempos de inferencia de los modelos ONNX medidos en la Raspberry Pi 5 (CPU). . . . .                     | 118 |
| 6.1. Endpoints de la API REST del sistema. . . . .                                                              | 123 |
| 6.2. Módulos concurrentes y mecanismos de comunicación interna. . . . .                                         | 126 |
| 6.3. Parámetros de la comunicación serie. . . . .                                                               | 127 |
| 6.4. Campos enviados al servidor. . . . .                                                                       | 128 |
| 6.5. Estructura del frontend. . . . .                                                                           | 128 |
| 7.1. Parámetros de entrada del pipeline — definen la ubicación y resolución del Grid Map . . . . .              | 138 |
| 7.2. Parámetros ajustables del pipeline — umbrales y buffers de clasificación . . . . .                         | 138 |
| 7.3. Tipos de celda definidos en el Grid Map . . . . .                                                          | 143 |

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

|                                                                                                                                                                                                                                        |     |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 8.1. Tipos de terreno en la retícula y su rol en los modelos ABM . . . . .                                                                                                                                                             | 151 |
| 8.2. Parámetros del sistema ODE de Jing Chen [1]. . . . .                                                                                                                                                                              | 153 |
| 8.3. Parámetros del modelo ABM y valores por defecto . . . . .                                                                                                                                                                         | 163 |
| 9.1. Períodos de realización de las pruebas de semi-campo. . . . .                                                                                                                                                                     | 176 |
| 9.2. Distribución de los sujetos de prueba por sexo. . . . .                                                                                                                                                                           | 178 |
| 9.3. Balance de especímenes durante la primera prueba de semi-campo.                                                                                                                                                                   | 181 |
| 9.4. Comparativa entre Detección IA y Realidad Biológica ( $N = 16$ ) —<br>Primera prueba. . . . .                                                                                                                                     | 181 |
| 9.5. Balance de especímenes durante la segunda prueba de semi-<br>campo. . . . .                                                                                                                                                       | 183 |
| 9.6. Comparativa entre Detección IA y Realidad Biológica — Segunda<br>prueba. . . . .                                                                                                                                                  | 183 |
| 9.7. Comparación de ratios de desempeño entre ambas pruebas de<br>semi-campo. . . . .                                                                                                                                                  | 184 |
| 9.8. Resultados de captura diaria y condiciones ambientales durante la<br>prueba de campo (1–7 de mayo de 2026). . . . .                                                                                                               | 187 |
| 9.9. Parámetros de entrada del Grid Map — Primer modelo . . . . .                                                                                                                                                                      | 189 |
| 9.10. Condiciones iniciales y parámetros de simulación — Primer modelo                                                                                                                                                                 | 190 |
| 9.11. Parámetros de entrada del Grid Map — Segundo modelo . . . . .                                                                                                                                                                    | 194 |
| 9.12. Condiciones iniciales y parámetros de simulación — Segundo mo-<br>delo . . . . .                                                                                                                                                 | 195 |
| 9.13. Coeficientes Partial Rank Correlation Coefficient (PRCC) sobre $N_{\text{final}}$<br>( $N = 100$ muestras Latin Hypercube Sampling (LHS), $T_{\text{máx}} = 80$ d).<br>El símbolo $\star$ indica $ \text{PRCC}  > 0,3$ . . . . . | 200 |

## **Índice de anexos**

|                                          |            |
|------------------------------------------|------------|
| <b>APÉNDICE A. Repositorio de GitHub</b> | <b>222</b> |
|------------------------------------------|------------|

## Lista de siglas

**ABM** Agent-Based Model

**API** Application Programming Interface

**ASGI** Asynchronous Server Gateway Interface

**BLDC** Brushless DC electric motor

**CA** Cellular Automata

**CDN** Content Delivery Network

**CI/CD** Continuous Integration and Continuous Deployment

**CNN** Convolutional Neural Network

**CPU** Central Processing Unit

**DEVS** Discrete Event System Specification

**FIUNA** Facultad de Ingeniería de la Universidad Nacional de Asunción

**FPN** Feature Pyramid Network

**FPS** Frames Per Second

**GIS** Geographic Information System

**GPU** Graphics Processing Unit

**GT** Ground truth

**HTTP** Hypertext Transfer Protocol

**I2C** Inter-Integrated Circuit

**IA** Inteligencia Artificial

**IICS** Instituto de Investigaciones en Ciencias de la Salud

**IoT** Internet of Things

**IoU** Intersection over Union

**JWT** JSON Web Token

**LHS** Latin Hypercube Sampling

**LME** Laboratorio de Mecánica y Energía

**mAP** mean Average Precision

**NMS** Non-Maximum Suppression

**ODE** Ordinary Differential Equation

**ONNX** Open Neural Network Exchange

**OSM** OpenStreetMap

**PaaS** Platform as a Service

**PAN** Path Aggregation Network

**PCB** Printed Circuit Board

**PDAF** Phase Detection Autofocus

**PDE** Partial Differential Equation

**PLA** Polylactic Acid

**PRCC** Partial Rank Correlation Coefficient

**PWM** Pulse Width Modulation

**QoI** Quantity of Interest

**R-CNN** Region-based Convolutional Neural Networks

**REST** Representational State Transfer

**ROI** Region of Interest

**SENPA** Servicio Nacional de Erradicación del Paludismo

**SIT** Sterile Insect Technique

**SMD** Surface-Mount Device

**SQL** Structured Query Language

**STFT** Short-Time Fourier Transform

**TPE** Tree-structured Parzen Estimator

**UART** Universal Asynchronous Receiver-Transmitter

**WGS84** World Geodetic System 1984

**YOLO** You Only Look Once

## **CAPÍTULO 1**

### **Introducción**

El control de enfermedades transmitidas por vectores representa uno de los principales desafíos en salud pública, especialmente en regiones tropicales y subtropicales donde las condiciones ambientales favorecen la proliferación de mosquitos. En este contexto, el avance de tecnologías emergentes, como la Inteligencia Artificial (IA), el Internet of Things (IoT) y el modelado computacional, abre nuevas oportunidades para el desarrollo de herramientas orientadas al monitoreo y control de estas enfermedades.

El presente trabajo se enmarca en esta línea, proponiendo el desarrollo paralelo e independiente de dos herramientas orientadas al mismo objetivo de largo plazo: por un lado, un sistema inteligente de monitoreo basado en dispositivos de borde, con capacidades de inferencia local mediante modelos de inteligencia artificial y conectividad IoT; por otro, un modelo computacional espacio-temporal de la población de mosquitos basado en agentes. Ambas herramientas fueron concebidas con una lógica de complementariedad —los datos generados por la trampa constituyen el insumo natural para la calibración del modelo computacional— pero se desarrollaron de forma enteramente independiente en el presente trabajo, estableciendo las bases técnicas para su integración y validación rigurosa en etapas futuras.

## 1.1. Planteamiento del problema

El dengue constituye una de las principales enfermedades endémicas en Paraguay, con brotes recurrentes que generan un alto impacto sanitario, social y económico, situación documentada en reportes epidemiológicos regionales recientes [2]. La transmisión del virus está asociada principalmente al mosquito *Aedes aegypti*, cuya proliferación se ve favorecida por condiciones climáticas tropicales, urbanización no planificada y deficiencias en el control de criaderos. A pesar de los esfuerzos sostenidos por parte de las autoridades sanitarias, el país continúa enfrentando epidemias periódicas, lo que evidencia la necesidad de mejorar las estrategias de monitoreo y control vectorial.

Las metodologías tradicionales de vigilancia entomológica, basadas en inspecciones manuales y conteo de larvas o huevos, presentan limitaciones significativas en términos de escalabilidad, frecuencia de muestreo y precisión. Estas limitaciones dificultan la obtención de datos en tiempo real y reducen la capacidad de respuesta ante incrementos en la población de mosquitos.

En este contexto, Paraguay ha comenzado a implementar nuevas estrategias de control biológico, destacándose la estrategia biológica *Wolbachia* [3], la cual consiste en la liberación de mosquitos infectados con la bacteria *Wolbachia*, que reduce la capacidad de transmisión del virus del dengue. Esta estrategia ha sido aprobada a nivel nacional mediante iniciativas legislativas específicas [4] y forma parte de un enfoque innovador para el control sostenible del vector. Experiencias internacionales han mostrado resultados prometedores; por ejemplo, en Indonesia se reportó una reducción del 77 % en los casos de dengue y del 86 % en hospitalizaciones [5]. Asimismo, estudios recientes han evidenciado la viabilidad económica y epidemiológica de esta estrategia en países como Brasil [6], mientras que su adopción continúa expandiéndose en la región, como lo demuestra su implementación en Perú [7].



A nivel nacional, instituciones como el Servicio Nacional de Erradicación del Paludismo (SENEPA) han participado activamente en iniciativas internacionales orientadas al control del *Aedes aegypti* mediante nuevas tecnologías [8], lo que refleja el interés creciente en la incorporación de enfoques innovadores.

No obstante, la implementación efectiva de la estrategia Wolbachia —y de cualquier intervención de control vectorial a gran escala— exige una planificación basada en evidencia de alta resolución. La liberación de mosquitos infectados con *Wolbachia* implica inversiones considerables en producción, logística y operaciones de campo; recursos que no pueden destinarse a despliegues mal dimensionados espacial o temporalmente. Para que estas intervenciones sean eficaces, es necesario, por un lado, contar con datos entomológicos precisos y actualizados —densidad poblacional, distribución espacial y variación temporal del vector— que permitan identificar las zonas y los momentos prioritarios de actuación. Por otro lado, es igualmente necesario disponer de herramientas computacionales capaces de simular distintos escenarios de intervención, de modo que sea posible anticipar el comportamiento del sistema y optimizar las decisiones antes de comprometer el presupuesto en campo. En este marco, la carencia conjunta de sistemas de monitoreo automatizado y de modelos espacio-temporales calibrados representa un obstáculo concreto para la gestión eficiente del control del dengue en el país.

Desde el punto de vista del modelado, la simulación de poblaciones de mosquitos representa un desafío complejo debido a la naturaleza estocástica del sistema, la heterogeneidad espacial del entorno urbano y la interacción local entre individuos. Si bien existen modelos basados en ecuaciones diferenciales ordinarias (Ordinary Differential Equation (ODE)) y parciales (Partial Differential Equation (PDE)), ambos enfoques suelen asumir homogeneidad espacial, lo que limita su capacidad para representar escenarios reales. Los modelos basados en agentes, en cambio, permiten incorporar heterogeneidad espacial y reglas de

comportamiento local, constituyendo una alternativa metodológica más adecuada para entornos urbanos complejos. No obstante, su calibración y validación rigurosa requieren datos experimentales de alta resolución que, en la mayoría de los contextos de vigilancia actuales, no están disponibles.

Esta carencia de datos sitúa al campo frente a una pregunta de investigación de largo alcance: ¿es posible inferir o predecir la dinámica de la población adulta de mosquitos a partir de observaciones parciales, como los conteos de huevos registrados por trampas? Responder esta pregunta de forma rigurosa exige, como condición previa, disponer tanto de herramientas capaces de generar datos de campo con suficiente resolución temporal y espacial, como de modelos computacionales validados que puedan interpretar esos datos. Abordar directamente dicha pregunta excede el alcance del presente trabajo; sin embargo, constituye la motivación de fondo que orienta el desarrollo de las dos herramientas propuestas.

En consecuencia, existe una brecha significativa entre los datos disponibles en campo, las herramientas de monitoreo actuales y los modelos matemáticos utilizados para la predicción y control de poblaciones de mosquitos. Esta brecha se manifiesta en dos dimensiones complementarias: la carencia de sistemas de adquisición capaces de generar datos entomológicos con suficiente resolución espacial y temporal, y la ausencia de modelos computacionales georreferenciados que permitan representar la dinámica del vector en entornos urbanos reales y que puedan ser calibrados con esos datos.

## **1.2. Justificación**

La presente investigación propone abordar las dos dimensiones de esta brecha mediante el desarrollo paralelo e independiente de dos herramientas complementarias: una trampa inteligente basada en tecnologías de Edge IA e IoT,

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

orientada a la adquisición automatizada de datos de campo; y un modelo basado en agentes (Agent-Based Model (ABM)) con soporte en datos de Geographic Information System (GIS), orientado a la representación computacional de la dinámica espacio-temporal del vector. Ambas herramientas fueron desarrolladas de forma independiente dentro del alcance de esta investigación, pero orientadas a una futura integración basada en datos reales de campo.

En cuanto a la trampa inteligente, su desarrollo permite automatizar el proceso de captura, identificación y clasificación de mosquitos, posibilitando la generación de datos georreferenciados con mayor detalle espacio-temporal en comparación con los métodos tradicionales. La integración de una arquitectura IoT para la transmisión y almacenamiento de datos facilita además su disponibilidad para análisis posteriores y su eventual uso en estudios epidemiológicos.

En cuanto al ABM, su desarrollo permite representar la dinámica poblacional del mosquito en entornos urbanos heterogéneos, posibilitando la exploración de escenarios y el análisis de fenómenos emergentes. El modelo está diseñado con parámetros modificables y una estructura extensible, lo que lo posiciona como una plataforma adecuada para incorporar datos reales de campo una vez que estos estén disponibles.

La contribución principal de este trabajo radica en el desarrollo de un prototipo funcional de trampa inteligente orientado a la adquisición automatizada de datos entomológicos y la formulación matemática de un modelo basado en agentes georreferenciados para la simulación espacio-temporal de la población de *Aedes aegypti*.

### 1.3. Objetivos

#### 1.3.1. Objetivo General

Desarrollar dos componentes complementarios orientados al monitoreo inteligente de poblaciones de mosquitos: una trampa inteligente basada en visión artificial e IoT para la adquisición automatizada de datos entomológicos, y un modelo basado en agentes para la simulación espacio-temporal de la población.

#### 1.3.2. Objetivos Específicos

- Desarrollar un sistema de visión artificial basado en modelos de aprendizaje profundo para la detección y clasificación de mosquitos en dispositivos embebidos.
- Diseñar e implementar el prototipo físico y la arquitectura electrónica de la trampa inteligente para la captura, procesamiento y transmisión de datos entomológicos.
- Desarrollar una arquitectura IoT para la gestión, almacenamiento y visualización de los datos generados por el sistema.
- Generar un entorno urbano georreferenciado a partir de datos GIS para su utilización en simulaciones espacio-temporales.
- Desarrollar un modelo basado en agentes para la simulación de la dinámica poblacional de *Aedes aegypti* en entornos urbanos.
- Validar experimentalmente el desempeño de la trampa inteligente y analizar el comportamiento del modelo basado en agentes mediante simulaciones exploratorias.

## **1.4. Alcance y limitaciones**

El presente trabajo se orientó al desarrollo de un prototipo funcional y la formulación de un modelo computacional que permitieran demostrar el potencial de las tecnologías emergentes aplicadas al problema epidemiológico del dengue.

### **1.4.1. Alcance**

En cuanto al alcance del sistema propuesto, se desarrolló un prototipo de trampa inteligente capaz de realizar inferencia en tiempo real sobre un dispositivo embebido, permitiendo la detección de mosquitos y su clasificación según sexo mediante técnicas de visión artificial. Adicionalmente, se implementó una arquitectura basada en IoT que permite la transmisión automática de datos e imágenes hacia una infraestructura de almacenamiento remoto, facilitando su análisis posterior.

En relación con el modelado computacional, se desarrolló un modelo basado en agentes (ABM) con capacidad de representar la dinámica espacio-temporal de la población de mosquitos. Este modelo incorporó heterogeneidad espacial mediante el uso de datos GIS, así como reglas de comportamiento local que permitieron simular procesos biológicos simples pero relevantes, tales como la interacción entre individuos y la reproducción en función de encuentros dentro de radios espaciales definidos. El modelo fue diseñado con un enfoque flexible, permitiendo la modificación de parámetros y reglas, y buscando coherencia con fundamentos matemáticos y biológicos.

Cabe destacar que la integración completa entre el sistema de adquisición de datos y el modelo computacional no se implementó dentro del alcance de este trabajo, quedando como una línea de desarrollo futuro.

### 1.4.2. Limitaciones

El alcance definido determina también los aspectos que quedan fuera del presente trabajo. Las principales limitaciones se detallan a continuación.

En cuanto a las limitaciones del sistema desarrollado, el prototipo de trampa inteligente no está orientado a su fabricación comercial, por lo que no se abordaron aspectos relacionados con la optimización de autonomía energética, resistencia a condiciones ambientales adversas ni certificaciones de uso en campo a largo plazo. Asimismo, el modelo de visión artificial implementado se considera de carácter experimental, con posibilidades de mejora tanto en términos de precisión como en la arquitectura utilizada.

Por otra parte, la arquitectura IoT fue implementada utilizando servicios basados en Application Programming Interface (API) públicas y plataformas con planes gratuitos, lo cual puede introducir restricciones en términos de escalabilidad, disponibilidad y capacidad de almacenamiento.

En relación con el modelo basado en agentes (ABM), se reconoce que el mismo constituye una aproximación simplificada del sistema real. Los parámetros del modelo no fueron calibrados exhaustivamente con datos experimentales dentro del alcance de este trabajo, por lo que su validación se considera preliminar. En consecuencia, el modelo se plantea como una herramienta exploratoria, cuyo ajuste y validación rigurosa quedan establecidos como trabajo futuro.

## **CAPÍTULO 2**

### **Revisión de la literatura**

En este Capítulo se revisan los antecedentes, conceptos y prácticas relevantes para el desarrollo del proyecto.

#### **2.1. Marco Teórico**

El marco teórico se organiza en dos subsecciones principales. Por un lado, se abordan los conceptos relacionados con el desarrollo de la trampa inteligente, incluyendo una revisión de trampas comerciales modernas, los atrayentes más utilizados, los modelos de visión artificial empleados y los mecanismos de persistencia de datos dentro de una arquitectura IoT. Por otro lado, se presentan distintos enfoques de modelado, junto con sus respectivas ventajas y desventajas en el contexto del modelado poblacional, particularmente en relación con la construcción de modelos basados en agentes (ABM).

##### **2.1.1. Conceptos para el diseño de sistemas inteligentes de captura de mosquitos**

###### **2.1.1.1. Trampas de mosquitos y atrayentes**

Las trampas de mosquitos constituyen herramientas fundamentales en programas de vigilancia epidemiológica y control vectorial, ya que permiten la captura, cuantificación y análisis de poblaciones de insectos transmisores de enfermedades. Su funcionamiento se basa en la reproducción de estímulos asociados

a los hospedadores humanos o animales, con el objetivo de atraer mosquitos en fase de búsqueda de sangre.

**Principio de funcionamiento.** Dispositivos como la trampa BG-Sentinel, ampliamente utilizada en estudios entomológicos, emplean sistemas de ventilación que generan corrientes de aire para simular la convección producida por el cuerpo humano. Este flujo transporta compuestos químicos atrayentes que imitan el olor de la piel, guiando a los mosquitos hacia el interior de la trampa, donde son capturados mediante succión [9].

En este contexto, se han desarrollado sistemas complementarios como el BG-Counter, que automatizan el monitoreo mediante sensores ópticos e infrarrojos capaces de detectar y clasificar insectos en tiempo real. Estos dispositivos permiten la transmisión remota de datos, reduciendo la necesidad de inspección manual y facilitando la vigilancia continua [10].



**Figura 2.1:** Dispositivo BG-Counter (Biogents AG). **Fuente:** *Biogents AG, BG-Counter: Innovative and Autonomous Mosquito Trap Station.*

**Uso de CO<sub>2</sub> como atrayente.** Uno de los factores más relevantes en la eficiencia de las trampas es el uso de dióxido de carbono (CO<sub>2</sub>), el cual actúa como un potente atrayente al ser un indicador natural de la presencia de un hospedador. Este gas puede suministrarse mediante cilindros presurizados o a través de



reacciones químicas controladas, como en el caso de geles generadores de  $\text{CO}_2$ , que permiten una liberación sostenida en el tiempo [11].



**Figura 2.2:** Sistema generador de  $\text{CO}_2$  basado en gel utilizado como atrayente químico en trampas para mosquitos. **Fuente:** Gel  $\text{CO}_2$ , Jerrican de 5L, Levure. Borne Lami Anti-Moustiques.

La combinación de  $\text{CO}_2$  con compuestos orgánicos volátiles, tales como ácido láctico, amoníaco y ácidos grasos, incrementa significativamente la capacidad de atracción al reproducir de manera más realista la firma química humana. Este enfoque multimodal resulta particularmente efectivo en especies del género *Aedes*, vectores de enfermedades como el dengue, zika y chikungunya.

**Monitoreo automatizado y precisión de conteo.** La incorporación de sensores y conectividad en las trampas ha permitido avanzar hacia sistemas de monitoreo automatizado. En este contexto, el dispositivo BG-Counter representa una evolución hacia sistemas capaces de proporcionar datos en tiempo real sobre la actividad de los mosquitos.

El estudio de Day et al. (2020) [12] evaluó la precisión de este dispositivo en distintos entornos, evidenciando una fuerte dependencia respecto a las condiciones ambientales.

Asimismo, se observó una relación positiva entre la abundancia relativa de

mosquitos y la precisión del sistema, lo que sugirió un mejor desempeño en entornos donde los insectos objetivo predominan. En escenarios con alta presencia de especies no objetivo de tamaño similar, la precisión disminuye debido a errores de clasificación.

No obstante, a pesar de estas limitaciones, el dispositivo presenta una correlación significativa con las tendencias poblacionales reales, lo que lo hace útil para el análisis temporal, la detección de picos de actividad y la evaluación de estrategias de control vectorial [12].

**Relevancia para sistemas de monitoreo inteligentes.** La evolución de estas tecnologías evidencia una transición desde sistemas de captura pasiva hacia plataformas de monitoreo automatizado, donde la adquisición de datos se complementa con capacidades de procesamiento y análisis.

En este marco, la integración de sensores, conectividad y modelos de visión por computador permite avanzar hacia sistemas autónomos capaces de realizar tareas de detección y clasificación en tiempo real. Este enfoque resulta particularmente relevante en el desarrollo de soluciones basadas en computación en el borde, donde se busca reducir la dependencia de infraestructura externa y mejorar la eficiencia operativa.

En consecuencia, el estudio de estas tecnologías sienta las bases conceptuales para el desarrollo de sistemas de monitoreo inteligente orientados al control de poblaciones de mosquitos.

#### **2.1.1.2. Modelos de clasificación de imágenes**

Los modelos de clasificación de imágenes constituyen una de las tareas fundamentales en visión por computador, cuyo objetivo es asignar una etiqueta a una imagen de entrada a partir de patrones aprendidos durante el entrenamiento. Este proceso implica la identificación de características discriminativas que permitan distinguir entre distintas clases.

En la práctica, estos modelos se implementan comúnmente mediante redes neuronales convolucionales (Convolutional Neural Network (CNN)), las cuales extraen representaciones jerárquicas de la imagen, desde características de bajo nivel como bordes y texturas, hasta estructuras de mayor complejidad semántica. A partir de estas representaciones, el modelo estima una distribución de probabilidad sobre las clases posibles, seleccionando aquella con mayor probabilidad como resultado.

El entrenamiento de estos modelos consiste en la optimización de sus parámetros para minimizar una función de pérdida que cuantifica el error entre las predicciones y las etiquetas reales. Este proceso permite mejorar progresivamente la capacidad del modelo para generalizar a nuevas imágenes.

Si bien la clasificación de imágenes resulta adecuada en escenarios donde el objeto de interés ocupa la mayor parte de la imagen, presenta limitaciones en contextos más complejos donde múltiples instancias pueden coexistir, lo que motiva el uso de modelos de detección de objetos.

### 2.1.1.3. Modelos de Detección de Objetos

Los modelos de detección de objetos extienden el problema de clasificación al incorporar la localización de los objetos dentro de la imagen. En este enfoque, no solo se determina qué objetos están presentes, sino también su posición, permitiendo identificar múltiples instancias en una misma escena.

En lugar de producir una única etiqueta, estos modelos generan un conjunto de predicciones, cada una compuesta por una clase, una región espacial (generalmente representada mediante cajas delimitadoras) y una medida de confianza asociada.

**Arquitectura general.** Estos modelos se basan en redes neuronales convolucionales que realizan de manera conjunta dos tareas: clasificación (identificación del objeto) y regresión (estimación de la localización del objeto dentro de la ima-

gen). Existen diferentes enfoques arquitectónicos, que van desde métodos de dos etapas, como Region-based Convolutional Neural Networks (R-CNN), hasta modelos de una sola etapa, como You Only Look Once (YOLO), los cuales priorizan la eficiencia computacional.

Los detectores de una etapa permiten procesar la imagen en una única pasada, lo que reduce significativamente el tiempo de inferencia. Esta característica resulta especialmente relevante en aplicaciones donde se requiere procesamiento en tiempo real y con recursos limitados, como sistemas embebidos o dispositivos de borde.

**Entrenamiento y evaluación.** El entrenamiento de modelos de detección implica la optimización simultánea de la precisión en la clasificación y la exactitud en la localización. Para su evaluación, se emplean métricas que integran ambos aspectos, permitiendo cuantificar el desempeño del modelo en términos de detección y posicionamiento.

Este enfoque permite dotar a los sistemas de capacidades avanzadas de percepción visual, necesarias en aplicaciones donde la interpretación espacial de la escena es un requisito fundamental.

#### 2.1.1.4. YOLO (You Only Look Once)

El modelo YOLO (*You Only Look Once*) representa un enfoque de detección de objetos en tiempo real que reformula el problema como una tarea de regresión directa desde la imagen de entrada hacia las predicciones finales. A diferencia de los métodos de múltiples etapas, YOLO procesa la imagen completa en una sola pasada, lo que permite alcanzar altas velocidades de inferencia [13].

Esta característica lo convierte en una alternativa adecuada para escenarios donde el tiempo de respuesta es un factor crítico, como en sistemas embebidos y aplicaciones de monitoreo en campo.

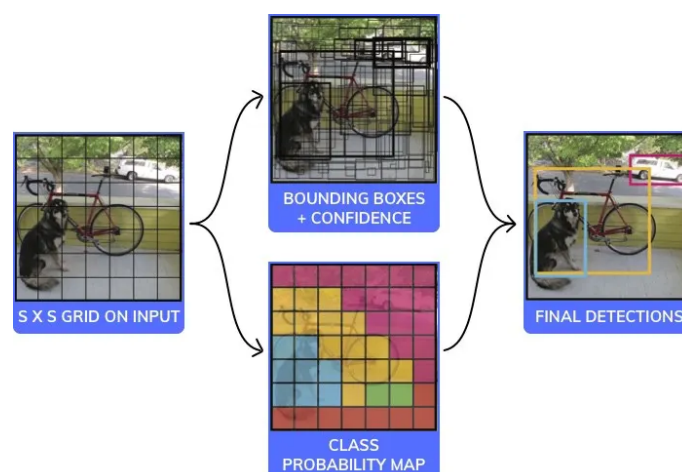
**Arquitectura general.** La arquitectura de YOLO se basa en una red neuronal

convolucional que divide la imagen en una cuadrícula y, para cada región, predice múltiples cajas delimitadoras junto con sus respectivas clases y niveles de confianza. Este diseño permite realizar simultáneamente la localización y clasificación de objetos en una única estructura.

Las versiones más recientes han incorporado mejoras orientadas a incrementar la precisión sin comprometer la eficiencia. En particular, trabajos como YOLOv7 han demostrado que es posible optimizar el rendimiento mediante estrategias de diseño estructural y entrenamiento avanzado [14].

**YOLOv8 y aplicaciones modernas.** YOLOv8 representa una evolución de esta familia de modelos, incorporando mejoras tanto en precisión como en facilidad de implementación. Su arquitectura sigue el esquema de detectores modernos, compuesta por tres bloques principales: *backbone*, *neck* y *head*.

El *backbone* se encarga de la extracción de características, mientras que el *neck* integra información multiescala mediante estructuras tipo Feature Pyramid Network (FPN) y Path Aggregation Network (PAN). Por su parte, el *head* adopta un enfoque desacoplado, separando las tareas de clasificación y regresión, lo que mejora la estabilidad del entrenamiento.



**Figura 2.3:** Esquema conceptual simplificado del funcionamiento de YOLO para detección de objetos. La imagen de entrada es dividida en una grilla espacial, donde la red predice simultáneamente cajas delimitadoras, probabilidades de clase y niveles de confianza para cada región de la escena. **Fuente:** *How to (Actually) Easily Detect Objects with Deep Learning*, Medium.

Adicionalmente, YOLOv8 emplea un esquema *anchor-free*, reduciendo la complejidad del ajuste de hiperparámetros. Estas características, junto con su eficiencia computacional, lo posicionan como una alternativa adecuada para aplicaciones de visión por computador en dispositivos con recursos limitados [15].

Diversos estudios recientes han evidenciado la robustez y adaptabilidad de esta familia de modelos en tareas complejas, manteniendo un equilibrio favorable entre precisión y velocidad [16].

En conjunto, YOLO se ha consolidado como uno de los enfoques más relevantes para la detección de objetos en tiempo real, particularmente en escenarios donde la eficiencia computacional es un requisito fundamental.

#### 2.1.1.5. MobileNetV3

MobileNetV3 es una arquitectura de red neuronal convolucional diseñada para aplicaciones de visión por computador en entornos con recursos computacionales limitados. Su objetivo es lograr un equilibrio entre precisión y eficiencia, permitiendo inferencia en tiempo real con bajo consumo energético [17].

Esta arquitectura combina técnicas de búsqueda automática de arquitecturas (*Neural Architecture Search*) con optimizaciones manuales, dando lugar a modelos altamente eficientes para distintos escenarios de despliegue.

A nivel estructural, emplea convoluciones separables en profundidad (*depth-wise separable convolutions*), reduciendo significativamente el número de parámetros y operaciones. Asimismo, incorpora módulos *Squeeze-and-Excitation*, que permiten ajustar dinámicamente la relevancia de los canales de características.

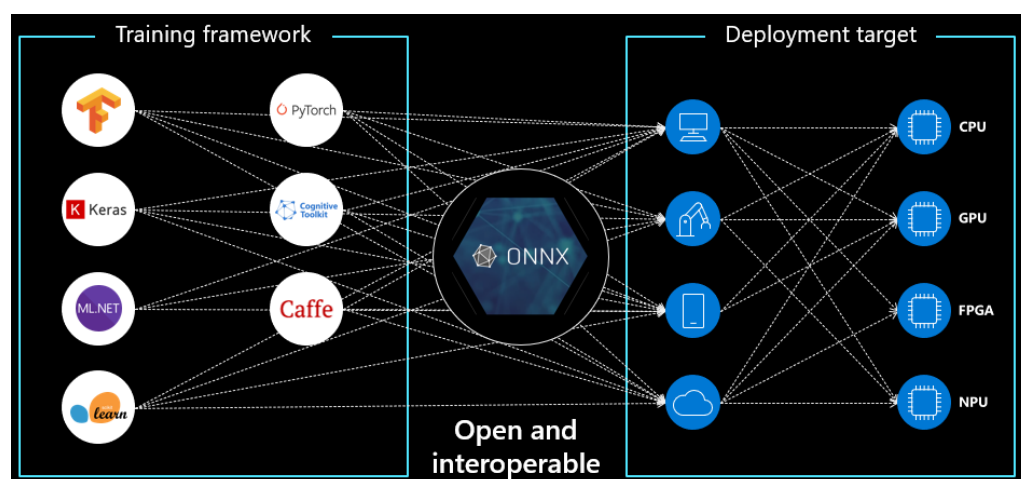
Adicionalmente, utiliza funciones de activación eficientes como *hard-swish*, que permiten aproximar comportamientos no lineales con menor costo computacional.

Estas características hacen que MobileNetV3 sea particularmente adecuado

como extractor de características o clasificador en sistemas embebidos, donde las restricciones de hardware imponen la necesidad de modelos ligeros sin comprometer significativamente el desempeño.

#### 2.1.1.6. ONNX y ONNX Runtime

Open Neural Network Exchange (ONNX) es un formato abierto que permite representar modelos de aprendizaje automático de manera interoperable entre distintos frameworks. Su principal propósito es desacoplar el proceso de entrenamiento del de despliegue, permitiendo que modelos desarrollados en entornos como PyTorch o TensorFlow puedan ser exportados y ejecutados en diversas plataformas sin modificaciones sustanciales [18].



**Figura 2.4:** Esquema de interoperabilidad ONNX: flujo desde frameworks de entrenamiento hacia plataformas de despliegue y hardware. **Fuente:** *ONNX and AI at the Edge Documentation*, Microsoft AI at Edge.

Sobre este formato se construye ONNX Runtime, un motor de inferencia optimizado que permite ejecutar modelos de manera eficiente en múltiples arquitecturas de hardware, incluyendo los Central Processing Unit (CPU), Graphics Processing Unit (GPU) y dispositivos embebidos. Este entorno de ejecución incorpora optimizaciones a nivel de grafo, lo que permite mejorar el rendimiento según las características del sistema objetivo [19].

**Relevancia en computación en el borde.** En aplicaciones de *edge computing*, donde los recursos computacionales son limitados y la latencia es un factor crítico, la posibilidad de ejecutar modelos directamente en el dispositivo resulta fundamental. En este contexto, ONNX Runtime permite realizar inferencia local, reduciendo la dependencia de servicios en la nube y mejorando los tiempos de respuesta [20].

Adicionalmente, el formato ONNX facilita la aplicación de técnicas de optimización como cuantización, poda de parámetros y simplificación de grafos, lo que permite adaptar los modelos a las restricciones de memoria y capacidad de cómputo propias de sistemas embebidos [21].

Otra característica relevante es su capacidad de ejecución eficiente en CPU, lo que resulta particularmente importante en dispositivos donde no se dispone de aceleradores especializados. Estudios recientes han demostrado que, mediante estas optimizaciones, es posible alcanzar inferencia de baja latencia manteniendo niveles adecuados de precisión en entornos locales [22].

En conjunto, ONNX y ONNX Runtime proporcionan una solución que facilita la transición desde modelos desarrollados en entornos de alto rendimiento hacia su implementación en dispositivos de borde, constituyendo un componente clave en sistemas de visión por computador desplegados en campo.

#### **2.1.1.7. Dispositivos IoT y Edge IA**

Los dispositivos IoT son sistemas físicos que integran sensores, capacidad de procesamiento y conectividad, permitiendo la adquisición y transmisión de datos desde el entorno hacia sistemas de almacenamiento o análisis. Tradicionalmente, este procesamiento se ha realizado en la nube, lo que introduce limitaciones asociadas a la latencia, la dependencia de conectividad y la gestión del ancho de banda.

Estas restricciones han impulsado el desarrollo de enfoques basados en



computación en el borde, donde parte del procesamiento se realiza directamente en el dispositivo. En este contexto, los sistemas de *Edge IA* extienden las capacidades de los dispositivos IoT al incorporar modelos de aprendizaje automático capaces de ejecutarse localmente.

Esta capacidad permite realizar tareas de percepción, como clasificación o detección de objetos, sin necesidad de transmitir los datos a un servidor externo. Como resultado, se obtiene una reducción en la latencia, un uso más eficiente del ancho de banda y una mayor robustez frente a condiciones de conectividad limitada.

En aplicaciones de monitoreo en campo, estas características resultan particularmente relevantes, ya que permiten la operación autónoma del sistema y la toma de decisiones en tiempo real. En este sentido, la integración de modelos de visión por computador en dispositivos de borde constituye un elemento fundamental para el desarrollo de sistemas inteligentes de vigilancia y control.

El interés actual en la ejecución eficiente de modelos de visión artificial sobre dispositivos de borde se refleja en investigaciones recientes enfocadas en evaluar el rendimiento energético y computacional de arquitecturas modernas de detección en hardware embebido. Estudios publicados en revistas de alto impacto, como *Scientific Reports* de Nature Portfolio, analizan el desempeño de modelos como YOLOv8 y RT-DETR sobre plataformas Edge IA, destacando la creciente relevancia de la inferencia local en aplicaciones de tiempo real [23].

En consecuencia, la combinación de arquitecturas IoT con capacidades de inferencia local habilita el desarrollo de soluciones distribuidas, donde la adquisición, procesamiento y transmisión de datos se encuentran parcialmente desacopladas, mejorando la eficiencia y escalabilidad del sistema.

#### 2.1.1.8. Arquitectura de Backend y Gestión de Datos

En sistemas de monitoreo basados en dispositivos distribuidos, la gestión de datos constituye un componente crítico, ya que debe garantizar el almacenamiento, disponibilidad y acceso eficiente a la información generada en campo. En este contexto, la naturaleza heterogénea de los datos requiere el uso de estrategias diferenciadas de almacenamiento.

En particular, es posible distinguir entre datos estructurados y no estructurados. Los primeros corresponden a información organizada en esquemas definidos, como registros de eventos, metadatos o resultados de inferencia. Para este tipo de datos, las bases de datos relacionales representan una solución adecuada, ya que permiten garantizar integridad, consistencia y eficiencia en las consultas.

Por otro lado, los datos no estructurados, como imágenes o secuencias de video, presentan requerimientos distintos en términos de almacenamiento. En estos casos, los sistemas de almacenamiento orientados a objetos (*object storage*) permiten manejar grandes volúmenes de información de manera escalable, almacenando cada elemento como una entidad independiente junto con sus metadatos asociados.

La separación entre almacenamiento estructurado y no estructurado permite optimizar tanto el acceso como la gestión de la información, evitando sobrecargar los sistemas relacionales con datos de gran tamaño y facilitando la recuperación eficiente de contenido multimedia.

Desde el punto de vista arquitectónico, estos sistemas suelen integrarse mediante servicios de **backend** que actúan como intermediarios entre los dispositivos de adquisición y los mecanismos de almacenamiento. Este enfoque permite centralizar la lógica de procesamiento, validación y acceso a los datos, facilitando la interoperabilidad entre distintos componentes del sistema.

En aplicaciones basadas en dispositivos IoT y Edge IA, esta arquitectura resulta particularmente relevante, ya que permite desacoplar la adquisición de datos en campo del almacenamiento y análisis posterior. De esta manera, los dispositivos pueden operar de forma autónoma realizando inferencia local, mientras que la infraestructura backend se encarga de la persistencia, agregación y disponibilidad de la información.

En conjunto, el uso combinado de almacenamiento estructurado y no estructurado, junto con una arquitectura de backend desacoplada, constituye una estrategia adecuada para la gestión eficiente de datos en sistemas de monitoreo distribuidos.

## **2.1.2. Conceptos para el Modelado Basado en Agentes (ABM)**

### **2.1.2.1. Datos geoespaciales (GIS)**

Los Sistemas de Información Geográfica GIS permiten capturar, almacenar, analizar y visualizar datos asociados a ubicaciones geográficas. Estos sistemas son fundamentales en aplicaciones donde la dimensión espacial influye directamente en la dinámica del fenómeno estudiado, como en modelos ambientales, urbanos o epidemiológicos.

Los datos geoespaciales se representan principalmente en formatos vectorial y raster. El formato vectorial describe entidades mediante geometrías discretas (puntos, líneas y polígonos), permitiendo representar elementos del entorno, cuerpos de agua o infraestructura vial. Por su parte, el formato raster organiza la información en una grilla regular, donde cada celda contiene un valor asociado a una variable espacial.

En el contexto de modelos basados en agentes, los datos GIS permiten construir entornos espacialmente explícitos en los cuales los agentes interactúan. Para ello, es posible transformar representaciones vectoriales en estructuras dis-

cretas, como grillas, que facilitan la simulación computacional y la definición de reglas locales. Este proceso implica la integración de múltiples capas de información geográfica, permitiendo caracterizar cada región del espacio según su uso o cobertura.

La incorporación de datos geoespaciales no solo mejora el realismo del modelo, sino que también permite capturar efectos emergentes derivados de la heterogeneidad del entorno, los cuales resultan importantes en la dinámica de poblaciones de mosquitos.

#### **2.1.2.2. Modelos basados en ecuaciones diferenciales ordinarias (ODEs)**

Las ecuaciones diferenciales ordinarias (ODEs) constituyen un enfoque clásico para el modelado de dinámicas poblacionales, describiendo la evolución temporal del sistema mediante relaciones continuas. En este marco, la tasa de cambio de la población se expresa como una función de su estado actual, lo que permite modelar fenómenos como crecimiento, interacción entre especies o propagación de enfermedades [1].

Este enfoque asume que la población está distribuida de forma homogénea y bien mezclada, es decir, que todos los individuos tienen la misma probabilidad de interactuar. Bajo esta hipótesis, los modelos presentan una alta tractabilidad analítica y bajo costo computacional, lo que facilita el estudio de tendencias globales.

Sin embargo, presentan limitaciones importantes cuando se requiere modelar heterogeneidad espacial o comportamientos individuales. En particular, las ODEs no permiten representar explícitamente la ubicación de los individuos ni las interacciones locales, lo que puede resultar insuficiente en sistemas donde la estructura espacial juega un rol determinante.

Por esta razón, si bien los modelos basados en ODEs son útiles para obtener una comprensión general de la dinámica poblacional, su aplicabilidad se ve res-

tringida en escenarios complejos donde la estructura espacial y las interacciones locales son determinantes.

### **2.1.2.3. Modelos basados en ecuaciones diferenciales parciales (PDEs)**

Las ecuaciones diferenciales parciales (PDEs) extienden el enfoque de las ODEs al incorporar variaciones espaciales, permitiendo modelar la dinámica de poblaciones distribuidas en un dominio geográfico [24]. En este caso, la población se representa como una función continua del espacio y el tiempo, lo que posibilita describir procesos como difusión, dispersión y propagación espacial.

Este tipo de modelos permite capturar fenómenos que no pueden ser representados mediante enfoques puramente temporales, como la formación de patrones espaciales o la propagación de frentes. De esta manera, las PDEs constituyen una herramienta más expresiva para el análisis de sistemas con dependencia espacial.

No obstante, esta mayor capacidad descriptiva implica un incremento significativo en la complejidad matemática y computacional. La resolución de PDEs requiere técnicas numéricas avanzadas y procesos de discretización del dominio, lo que puede resultar costoso y difícil de implementar en escenarios de alta resolución [25].

En consecuencia, si bien las PDEs permiten incorporar la dimensión espacial de forma continua, su complejidad limita su aplicabilidad en problemas donde se requiere un modelado detallado de interacciones locales y comportamiento individual.

### **2.1.2.4. Modelos basados en agentes (ABM)**

Los modelos basados en agentes (ABM, *Agent-Based Models*) representan un enfoque computacional alternativo orientado a la simulación de sistemas com-

plejos, en los cuales el comportamiento global emerge a partir de la interacción de múltiples entidades individuales [26]. A diferencia de los modelos continuos, los ABM describen el sistema mediante agentes discretos, cada uno con estados y reglas de comportamiento definidos de manera explícita.

Este enfoque permite modelar directamente la heterogeneidad entre individuos, las interacciones locales y la influencia del entorno espacial, aspectos que resultan difíciles de capturar mediante ODEs o PDEs. En particular, los ABM facilitan la incorporación de reglas no lineales, comportamientos adaptativos y estructuras espaciales complejas.

Como contrapartida, estos modelos presentan una menor tractabilidad analítica, ya que su análisis se basa principalmente en simulaciones numéricas y experimentación computacional. Asimismo, su implementación puede resultar más compleja desde el punto de vista del desarrollo.

No obstante, en escenarios donde la dinámica del sistema depende fuertemente de la interacción entre individuos y del entorno espacial, como en el caso de poblaciones de mosquitos en entornos urbanos, los ABM constituyen una herramienta especialmente adecuada, al permitir representar de manera explícita la estructura del sistema y los mecanismos que dan lugar a comportamientos emergentes.

## **2.2. Estado del Arte**

Con estos fundamentos teóricos establecidos, la siguiente sección revisa los trabajos previos más relevantes en el ámbito de las trampas inteligentes y el modelado espacio-temporal de mosquitos.

### 2.2.1. Antecedentes en Trampas Inteligentes para Mosquitos

En los últimos años, se han desarrollado múltiples enfoques para la detección, clasificación y captura selectiva de mosquitos mediante la integración de visión por computador, sistemas embebidos y tecnologías IoT.

Un enfoque destacado es el propuesto por Oliveira et al. [27], quienes desarrollaron una trampa inteligente basada en una Raspberry Pi 4 y el modelo YOLOv7. Su sistema incorpora un mecanismo de ventilación selectiva, donde la activación de distintos ventiladores permitía capturar mosquitos de interés, como *Aedes aegypti*, o liberar insectos beneficiosos como polinizadores. Además, integraron comunicación LoRaWAN para la transmisión de datos hacia la nube, implementando un sistema completo de monitoreo en tiempo real.

Por otro lado, Liu et al. [28] propusieron una trampa basada en una Raspberry Pi 3 que incorpora un mecanismo mecánico de clasificación mediante una placa rotatoria. Su sistema utiliza visión por computador para identificar especies y separar físicamente los individuos en distintos compartimientos. Destaca en este trabajo la generación de un dataset dinámico con insectos en movimiento, lo que mejora la robustez del modelo en condiciones reales. Sin embargo, su tasa de captura efectiva se mantiene limitada, evidenciando los desafíos en la integración mecánica del sistema.

Desde el punto de vista del desarrollo de modelos de clasificación, Park et al. [29] realizaron un análisis comparativo de arquitecturas profundas como VGG-16, ResNet-50 y SqueezeNet, alcanzando altas tasas de precisión mediante técnicas de *transfer learning*. Un aporte relevante de este trabajo es el uso de mapas de calor (Grad-CAM), que permiten interpretar las decisiones del modelo al identificar características morfológicas clave utilizadas en la clasificación.

En cuanto a la mejora de la calidad de los datos de entrada, Lee et al. [30] propusieron un enfoque basado en la fusión de imágenes RGB y fluorescentes

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

(UV), demostrando que la combinación de ambas modalidades mediante técnicas como la supresión de no máximos (Non-Maximum Suppression (NMS)) permite mejorar significativamente el rendimiento del sistema, alcanzando altos valores de precisión en la identificación de especies.

A nivel de hardware e identificación en campo, Wilkes et al. [31] demostraron la viabilidad de modificar sensores CMOS de bajo costo para capturar imágenes en el espectro ultravioleta, mientras que Paul et al. [32] exploraron el uso de marcadores fluorescentes para estudios de dispersión mediante captura-marcaje-recaptura. Ambos trabajos apuntan a soluciones accesibles y de bajo costo que amplían las capacidades de detección sin recurrir a equipamiento especializado.

La Tabla 2.1 sintetiza las características más relevantes de los trabajos revisados, comparando el dispositivo de inferencia utilizado, la arquitectura de IA empleada, la presencia o no de trampa física y el aporte principal para el presente trabajo.



APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

**Tabla 2.1:** Comparativa de trabajos relacionados en trampas inteligentes para mosquitos.

| Trabajo                     | Dispositivo de inferencia | Arquitectura IA                 | Trampa | Aporte relevante para este trabajo                                                                                                    |
|-----------------------------|---------------------------|---------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------|
| Wilkes et al. (2016) [31]   | Raspberry Pi              | Sin IA (imagen RAW)             | No     | Cámaras UV de bajo costo ( $\approx 25$ USD) mediante modificación química del filtro Bayer de sensores CMOS.                         |
| Park et al. (2020) [29]     | Nvidia Jetson TX2         | VGG-16 / ResNet-50              | No     | Identificó características morfológicas clave (tórax y abdomen) mediante Grad-CAM, comparables a las usadas por entomólogos expertos. |
| Lee et al. (2023) [30]      | Nvidia Jetson AGX Xavier  | Swin-Transformer + Faster R-CNN | Sí     | Fusión de imágenes RGB y UV con NMS para clasificación de mosquitos vivos bajo iluminación dual (LED blanco y UV).                    |
| Liu et al. (2023) [28]      | Raspberry Pi 3            | SqueezeNet                      | Sí     | Factibilidad de arquitecturas ligeras en el borde con mecanismo rotativo de clasificación física y conectividad IoT.                  |
| Oliveira et al. (2024) [27] | Raspberry Pi 4            | YOLOv7                          | Sí     | Detección en tiempo real con múltiples conductos de viento y conectividad LoRaWAN, capturando mosquitos y liberando polinizadores.    |
| Paul et al. (2024) [32]     | Sin dispositivo embebido  | Sin IA                          | No     | Marcadores fluorescentes de bajo costo para estudios de dispersión mediante captura-marcaje-recaptura con iluminación UV.             |

**Fuente:** Elaboración propia a partir de los trabajos citados.

Si bien existen propuestas que combinan visión artificial con mecanismos de clasificación física —Liu et al., Lee et al., Oliveira et al.—, ninguna integra simultáneamente clasificación por sexo, arquitectura en cascada exportada a ONNX y conectividad IoT sobre una Raspberry Pi. Persisten además desafíos en la robustez en condiciones reales y la eficiencia de captura, lo que justifica la exploración de nuevas arquitecturas basadas en Edge IA y modelos optimizados para entornos embebidos.

### 2.2.2. Antecedentes en Modelos Espacio-Temporales de Mosquitos

El modelado espacio-temporal de poblaciones de mosquitos ha sido abordado mediante diversos enfoques, incluyendo modelos basados en agentes (ABM), autómatas celulares (Cellular Automata (CA)) y ecuaciones diferenciales parciales (PDEs), cada uno con distintas capacidades para representar la heterogeneidad espacial y el comportamiento del sistema.

Entre los trabajos revisados destacan dos hallazgos transversales. Primero, la estructura espacial del entorno condiciona de forma determinante la dinámica poblacional: Maneerat y Daudé [33] mostraron que los edificios actúan como barreras al movimiento mediante un coeficiente de porosidad, mientras que Yamashita et al. [24] demostraron que la convección por viento genera dispersión asimétrica a lo largo de corredores urbanos. Segundo, esta heterogeneidad espacial tiene consecuencias directas para el control: Ferreira et al. [34] evidenciaron que los modelos continuos tienden a subestimar la persistencia poblacional bajo intervenciones intensivas, precisamente porque no capturan la distribución irregular de los sitios de oviposición. La Tabla 2.2 resume el conjunto de trabajos revisados.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

**Tabla 2.2:** Comparativa de trabajos en modelado espacio-temporal de poblaciones de mosquitos.

| Trabajo                      | Modelo | Enfoque                                                                                                             | ¿Distingue sexos?                                                                 | Conclusiones                                                                                                                           |
|------------------------------|--------|---------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Ferreira et al. (2008) [34]  | CA     | Evalúa la SIT para combatir <i>Aedes aegypti</i> en distintos entornos.                                             | Sí: distingue machos silvestres, machos estériles y hembras fértiles/no fértiles. | La distribución heterogénea de criaderos hace que la SIT sea difícil o impracticable.                                                  |
| García et al. (2014) [35]    | CA     | Uso de cultivos intercalados para regular la plaga <i>D. speciosa</i> .                                             | Solo hembras adultas (proporción de sexo 1:2 en el paso larva-adulto).            | Intercalar cultivos reduce la dispersión del insecto; la disponibilidad de maíz en los bordes del campo es clave para el control.      |
| Maneerat y Daudé (2017) [33] | ABM    | Efecto de edificios, calles y parques de una ciudad (Delhi) en la dispersión de <i>A. aegypti</i> .                 | Solo hembras adultas, por ser las responsables de la transmisión de virus.        | Los mosquitos son sedentarios (<100 m); los edificios actúan como barreras y los parques como corredores de dispersión.                |
| Yamashita et al. (2018) [24] | PDE    | Dinámica poblacional y dispersión de <i>A. aegypti</i> en barrios urbanos considerando viento y topografía.         | Solo hembras maduras, por ser la fase móvil y alada.                              | La población difunde lento en manzanas pero rápido por calles gracias al viento; es más efectivo fumigar uniformemente todo el barrio. |
| Chen et al. (2023) [1]       | ODE    | Proliferación y control de <i>A. aegypti</i> en Miami usando datos de trampas                                       | Solo hembras adultas, por ser las que buscan sangre y caen en las trampas.        | La lluvia afecta más los criaderos en áreas turísticas; los adulticidas son efectivos, pero lo óptimo es combinarlos con larvicidas.   |
| Ishida et al. (2023) [25]    | PDE    | Análisis matemático de la acotación de soluciones en modelos de quimiotaxis con mortalidad inducida por movimiento. | No realiza distinción de sexo en su modelo de densidad poblacional general.       | Establecieron condiciones matemáticas precisas para garantizar que las soluciones no crezcan indefinidamente en tiempo finito.         |

**Fuente:** Elaboración propia a partir de los trabajos citados.

Como muestra la Tabla 2.2, la gran mayoría de los modelos revisados considera únicamente hembras adultas, siendo Ferreira et al. [34] el único que incorpora distinción explícita de sexos, en el contexto particular de la Sterile Insect Technique (SIT). Esta omisión generalizada de la dinámica de apareamiento deja abierto un interés metodológico relevante: los ABM, al permitir codificar reglas de interacción local entre individuos, constituyen el marco más natural para incorporar encuentros entre machos y hembras como mecanismo generador de la dinámica reproductiva. En una dirección complementaria, Ishida et al. [25] demostraron que la interacción entre quimiotaxis y mortalidad asociada al movimiento produce una cota superior sobre la densidad poblacional que emerge de las reglas del sistema, sin necesidad de imponer explícitamente una capacidad de carga. Esto abre una pregunta de diseño directa para los ABM: explorar si parámetros como la capacidad de carga pueden surgir de forma emergente a partir de reglas locales —movimiento estocástico, competencia por sitios de cría y mortalidad dependiente de la densidad—, en lugar de ser fijados como constantes del modelo.

### 2.2.3. Trabajos Relacionados en la FIUNA

Identificados los antecedentes internacionales en modelos espacio-temporales y trampas inteligentes, y ante la necesidad local de contar con dispositivos automáticos de captura y clasificación sexual de *Aedes aegypti*, la FIUNA ha desarrollado recientemente diversos trabajos orientados al diseño de sistemas para la detección y clasificación de mosquitos, integrando enfoques mecatrónicos y técnicas de procesamiento de señales e inteligencia artificial. Cabe destacar que, con el último de estos trabajos y el presente, se está incursionando progresivamente en el área del modelamiento de poblaciones de mosquitos.

Honge Kudo [36] desarrolló un sistema mecatrónico para la clasificación de mosquitos, enfocado en la integración de hardware y sensores para la identifica-

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

ción de insectos basada en micrófonos y Short-Time Fourier Transform (STFT). Este trabajo sienta bases importantes en el diseño físico y la automatización de procesos de clasificación.

Por su parte, Zichner von Schmeling [37] propuso un sistema basado en la detección de la frecuencia de aleteo mediante sensores infrarrojos, automatizando por completo el diseño anterior mediante una Printed Circuit Board (PCB) personalizada y motores Brushless DC electric motor (BLDC). Este enfoque destaca por el uso de señales físicas como característica discriminativa para la identificación de especies.

Finalmente, Balbuena Duarte y Barrios Pereira [38] desarrollaron una trampa inteligente basada en inteligencia artificial para la detección de mosquitos, integrando modelos de visión por computador en un sistema automatizado.

En conjunto, estos trabajos evidencian el interés y avance en el desarrollo de soluciones tecnológicas para el monitoreo y control de mosquitos dentro de la FIUNA. Asimismo, reflejan una evolución desde enfoques basados en sensores físicos hacia la incorporación de inteligencia artificial, lo que proporcionó un contexto relevante para el desarrollo del presente trabajo.

La Tabla 2.3 resume y compara los tres trabajos según las innovaciones aportadas, las recomendaciones pendientes y la incorporación de modelamiento matemático.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

**Tabla 2.3:** Comparativa de trabajos relacionados desarrollados en la FIUNA.  
**Fuente:** Elaboración propia.

| Trabajo                                            | Innovaciones                                                                                                                                           | Recomendaciones de los autores                                                                                                                                                   | Modelo matemático                                                |
|----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|
| Honge Kudo (ago. 2025) [36]                        | Primer prototipo mecatrónico funcional. Succión por turbina de vacío, cilindro rotatorio para separación física y sensado acústico (micrófono + STFT). | Automatizar la captura (era manual por interruptor), reducir ruidos mecánicos y migrar a clasificación automática por IA.                                                        | No                                                               |
| Zichner von Schmeling (dic. 2025) [37]             | Sustitución del audio por sensor infrarrojo (OPT101) inmune al ruido ambiental. PCB personalizada, succión compacta (BLDC) e interfaz gráfica táctil.  | Lectura en tiempo real de la señal del sensor (reducción de latencia), mejora de la rugosidad interna del canal de vuelo y entrenamiento de modelos de <i>machine learning</i> . | No                                                               |
| Balbuena Duarte y Barrios Pereira (dic. 2025) [38] | Clasificación visual por sexo con YOLOv8, mecanismo de captura tipo iris servocontrolado y atrayente biológico de CO <sub>2</sub> por fermentación.    | Migrar inferencia al dispositivo embebido ( <i>edge IA</i> ), corregir desenfoque en cámara y añadir conectividad IoT.                                                           | Sí. Modelo determinístico de ODE basado en temperatura y lluvia. |

**Fuente:** Elaboración propia a partir de los trabajos citados.

En síntesis, la revisión de la literatura pone de manifiesto que el problema del monitoreo y control de *Aedes aegypti* ha sido abordado desde frentes complementarios: modelos espacio-temporales que revelan el papel determinante de la heterogeneidad espacial en la dinámica poblacional, sistemas de trampa inteligente que demuestran la viabilidad del *edge IA* para la clasificación en campo, y una línea institucional que ha ido convergiendo progresivamente hacia la integración de ambas dimensiones. Las brechas identificadas —clasificación sexual embebida, conectividad IoT y modelado con componente espacial— constituyen el punto de partida del presente trabajo. El siguiente capítulo presenta la metodología desarrollada para abordar cada una de ellas.

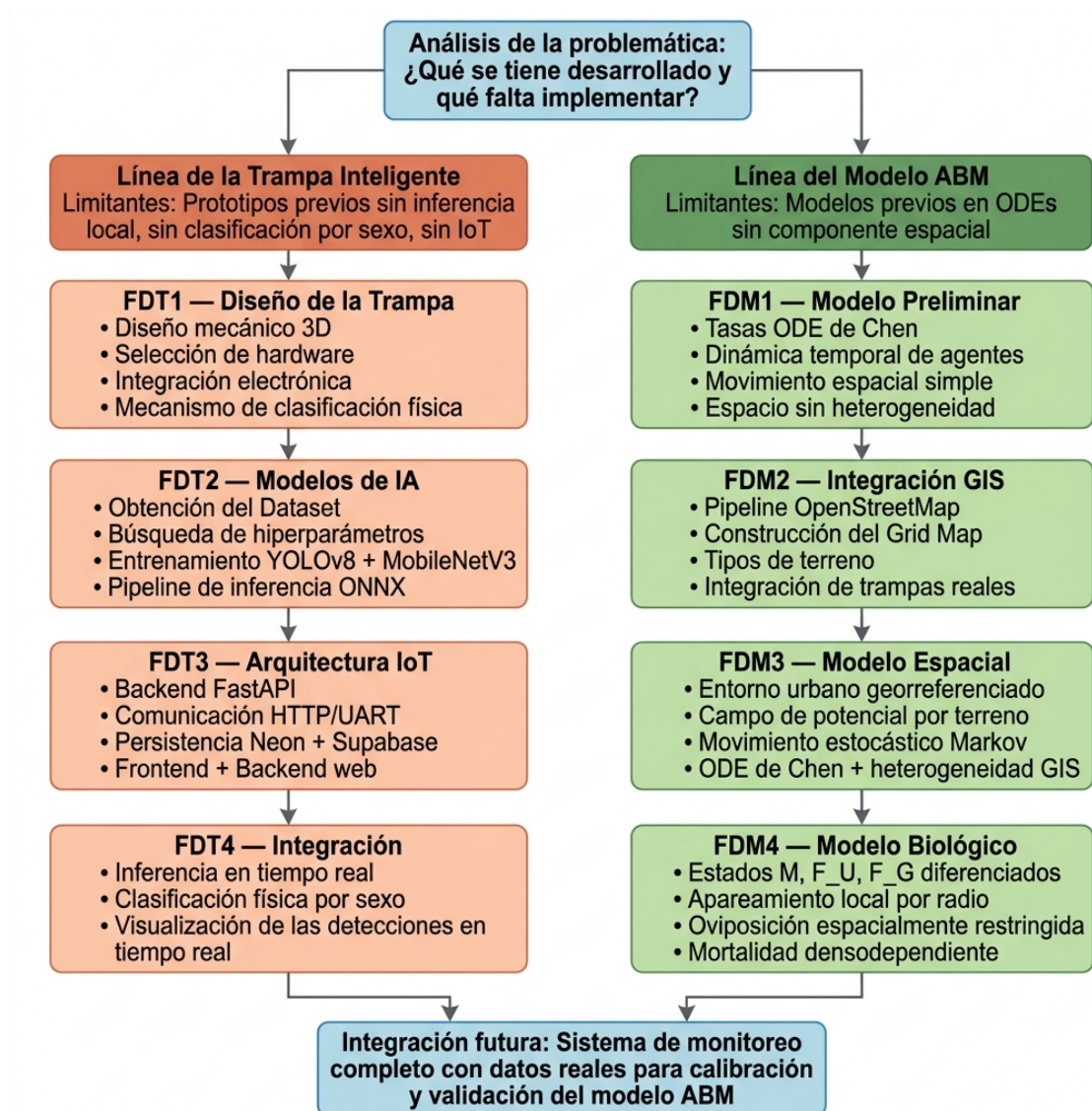
## **CAPÍTULO 3**

### **Metodología**

La metodología adoptada se organizó en torno a estas dos líneas de investigación paralelas, las cuales responden a brechas identificadas en el estado del arte y en los antecedentes desarrollados en la FIUNA: por un lado, la ausencia de un sistema de captura con clasificación física automática e inferencia local; por otro, la carencia de modelos computacionales con representación espacio-temporal del entorno urbano.

El enfoque metodológico general se basó en un desarrollo iterativo e incremental, donde cada fase construye sobre los resultados de la anterior y está orientada hacia una integración futura de ambos componentes. La Figura 3.1 presenta el esquema general de la metodología del proyecto, ilustrando el punto de partida de cada línea, sus objetivos respectivos, las fases de desarrollo y la convergencia hacia una visión integrada del sistema.

### 3.1. Arquitectura General del Proyecto



**Figura 3.1:** Arquitectura general del proyecto, mostrando las dos líneas de desarrollo paralelas (Trampa Inteligente y Modelo ABM) y su convergencia hacia la integración futura del sistema. **Fuente:** Elaboración propia.

Como se observa en la Figura 3.1, el proyecto se organizó en dos líneas de desarrollo paralelas e independientes que parten de un análisis común de la problemática:

- **Línea de la Trampa Inteligente (FDT):** orientada a resolver las limitaciones de los prototipos previos desarrollados en la FIUNA, en particular la



ausencia de inferencia local en el dispositivo de borde, la falta de clasificación física de los mosquitos según sexo, y la no implementación de una arquitectura IoT para la transmisión y gestión de datos.

- **Línea del Modelo Basado en Agentes (FDM):** orientada a superar las limitaciones de los modelos previos basados en ODEs, que carecen de componente espacial explícita, mediante la construcción de un ABM con heterogeneidad espacial urbana derivada de datos GIS.

Si bien ambas líneas se desarrollaron de forma independiente a lo largo de este trabajo, fueron concebidas desde el inicio con una lógica de complementariedad: los datos generados por la trampa inteligente fueron diseñados para alimentar y calibrar el modelo computacional en trabajos futuros, con el objetivo de cerrar el ciclo entre adquisición de datos y modelado.

### 3.2. Fases de Desarrollo

Cada línea de desarrollo se estructuró en un conjunto de fases secuenciales, definidas a partir de los objetivos específicos del trabajo (Sección 1.3, Capítulo 1). Las fases son incrementales: los resultados de cada una habilitarán los insumos de la siguiente, asegurando coherencia técnica y trazabilidad del proceso.

#### 3.2.1. Fases de Desarrollo de la Trampa Inteligente

El desarrollo de la trampa inteligente comprendió cuatro fases principales que cubren desde la inteligencia artificial hasta la integración física del sistema completo:

- **FDT1 — Diseño y construcción de la trampa:** Esta fase comprendió el diseño del modelo 3D de la trampa, la selección e integración de los componentes electrónicos y mecánicos, el diseño del diagrama eléctrico, y la

construcción del prototipo funcional. La trampa incorpora un mecanismo de clasificación física rotatorio, un sistema de visión artificial y un cebo de  $\text{CO}_2$  biológico para la atracción de mosquitos. El detalle de esta fase se presenta en el Capítulo 4.

- **FDT2 — Modelos de IA y pipeline de inferencia:** Esta fase abarcó la adquisición y preparación del conjunto de datos, la búsqueda de hiperparámetros óptimos mediante técnicas automatizadas, y el entrenamiento de los modelos de detección (*YOLOv8*) y clasificación (*MobileNetV3*). Se incluyó la exportación de los modelos al formato ONNX para su despliegue eficiente en dispositivos de borde, y el diseño del pipeline en cascada que integra ambos modelos. El detalle de esta fase se presenta en el Capítulo 5.
- **FDT3 — Diseño de la arquitectura IoT:** Esta fase definió la arquitectura cliente-servidor del sistema, que incluyó el dispositivo de borde como nodo de adquisición y procesamiento local, un backend basado en *FastAPI* desplegado en la nube, y una interfaz web para la visualización de datos en tiempo casi real. Se especifican los protocolos de comunicación, los mecanismos de persistencia de datos y las estrategias de despliegue. El detalle de esta fase se presenta en el Capítulo 6.
- **FDT4 — Integración completa de la trampa:** Esta fase consolidó los tres componentes anteriores (modelo de IA, hardware físico y arquitectura IoT) en un prototipo funcional integrado. Se realizaron pruebas de campo y semi-campo para evaluar el desempeño del sistema en condiciones reales, incluyendo la captura, identificación, clasificación física y registro de mosquitos. Los resultados de esta fase se presentan en el Capítulo 9.

### 3.2.2. Fases de Desarrollo del Modelo Basado en Agentes

El desarrollo del ABM también comprendió cuatro fases, que van desde la construcción de un modelo preliminar hasta la implementación de un modelo con reglas biológicas propias y entorno heterogéneo:

- **FDM1 — Construcción de un modelo preliminar basado en las ODEs de Chen:** Como punto de partida, se construyó un modelo ABM preliminar cuyas probabilidades de transición demográfica (mortalidad y maduración) se derivan directamente del sistema de ODE propuesto por [1] para la dinámica de *Aedes aegypti* en función de la temperatura. En este modelo preliminar, el movimiento de los adultos es guiado por un campo de potencial sin heterogeneidad de entorno. Cabe destacar que este modelo sienta las bases y no se ahondará más sobre el mismo en este trabajo, centrando el enfoque exclusivamente en los dos modelos siguientes.
- **FDM2 — Implementación del pipeline GIS para la generación del entorno urbano:** Esta fase consistió en el desarrollo de un pipeline automatizado para la construcción del entorno de simulación a partir de datos geoespaciales reales descargados de *OpenStreetMap*. El resultado es una grilla discreta (Grid Map) que codifica los tipos de terreno del entorno urbano (edificios, calles, vegetación, cuerpos de agua y trampas físicas). El detalle de esta fase se presenta en el Capítulo 7.
- **FDM3 — Construcción del primer modelo con campo de potencial en entorno heterogéneo:** Esta fase incorporó el entorno urbano georreferenciado al modelo preliminar, añadiendo un campo de potencial que pondera el tipo de terreno para guiar el movimiento de los adultos. Se mantienen las tasas de transición derivadas del sistema ODE de referencia, pero se añade la heterogeneidad espacial como elemento diferenciador. El detalle de

este modelo se presenta en el Capítulo 8. Este modelo fue presentado en el **XLIV Congreso Nacional de Matemática Aplicada e Computacional** (CNMAC 2025).

- **FDM4 — Construcción del segundo modelo con reglas propias e interacciones locales:** Esta fase introdujo un segundo modelo ABM con reglas de transición propias, independientes del sistema ODE de referencia, e incorporó estados adicionales diferenciados por sexo y estado reproductivo ( $M$ ,  $F_U$ ,  $F_G$ ). La dinámica reproductiva quedó definida íntegramente por interacciones locales: el apareamiento requiere la presencia de machos en un radio de búsqueda, la oviposición se restringe a sitios de cría dentro de un radio máximo, y la mortalidad juvenil es densodependiente dentro de un vecindario local. Se mantiene el entorno urbano heterogéneo generado en la fase FDM2. El detalle de este modelo se presenta en el Capítulo 8. Este modelo fue aceptado para su presentación en el **XLV Congreso Nacional de Matemática Aplicada e Computacional** (CNMAC 2026).

### 3.3. Proyección de Integración Futura

Las dos líneas de desarrollo —la trampa inteligente y el modelo basado en agentes— se abordaron de forma completamente independiente en el presente trabajo, sin que se implemente ninguna forma de integración entre ambas dentro de su alcance. No obstante, la arquitectura de cada componente fue diseñada con una lógica de complementariedad que habilita su integración en etapas futuras. La trampa inteligente fue concebida para generar datos georreferenciados con alta resolución temporal sobre la actividad de los mosquitos en campo, incluyendo conteos de capturas diferenciadas por sexo y variables ambientales como temperatura y humedad.

Estos datos, acumulados a lo largo del tiempo por una red de trampas desple-

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

gadas en el barrio de estudio, constituyen el insumo natural para la calibración de los parámetros del ABM. En particular, la calibración de las tasas de mortalidad, maduración y apareamiento a partir de datos reales permitiría validar el modelo de forma rigurosa, reduciendo la brecha entre la dinámica simulada y el sistema real.

En consecuencia, la integración de ambas herramientas tendría el potencial de evolucionar hacia un sistema capaz no solo de monitorear, sino también de predecir y optimizar estrategias de control vectorial. Esta proyección se establece como la principal línea de trabajo futuro del proyecto y se discute con mayor detalle en el Capítulo 10.

Los capítulos siguientes detallan cada uno de los componentes del sistema de acuerdo con las fases descritas en este Capítulo:

- **Capítulo 4:** Diseño y hardware de la trampa (FDT1).
- **Capítulo 5:** Modelo de detección y clasificación (FDT2).
- **Capítulo 6:** Arquitectura IoT implementada (FDT3).
- **Capítulo 7:** Generación del Grid Map a partir de datos GIS (FDM2).
- **Capítulo 8:** Modelado y simulación ABM (FDM3 y FDM4).
- **Capítulo 9:** Resultados y pruebas de campo de la trampa (FDT4) y resultados de simulación de los modelos ABM (FDM3 y FDM4).

En síntesis, el presente trabajo se estructuró en dos líneas paralelas y complementarias: (**FDT**) la adquisición automatizada de datos entomológicos mediante tecnologías de Edge IA e IoT; y (**FDM**) el modelado computacional espacio-temporal basado en agentes. Aunque ambas líneas fueron desarrolladas de forma independiente dentro del alcance de esta investigación, convergen conceptualmente en la futura calibración del modelo a partir de datos reales obtenidos en campo.

## **CAPÍTULO 4**

### **Diseño y Hardware de la trampa**

Este Capítulo describe el diseño y la implementación física de la trampa inteligente desarrollada como parte de la fase FDT1 del proyecto. Se abordan cuatro aspectos principales: la preparación del cebo de  $\text{CO}_2$  biológico utilizado como atrayente, el diseño mecánico tridimensional y el principio de funcionamiento del sistema de captura y clasificación, la selección e integración de los componentes electrónicos que conforman la arquitectura de hardware, y un presupuesto detallado del prototipo. En conjunto, estos elementos constituyen la base física sobre la cual opera el sistema de visión artificial y la arquitectura IoT descritos en los capítulos siguientes.

#### **4.1. Materiales y Preparación del Cebo**

La solución propuesta se fundamentó en el principio de la biomimética, que imita la exhalación de  $\text{CO}_2$  del cuerpo humano. Este gas constituye uno de los principales estímulos utilizados por los mosquitos hematófagos para la localización de hospedadores.

Al simular la presencia humana, el sistema atrayente incrementa la probabilidad de que los insectos se aproximen al dispositivo. Una vez en las proximidades, estos son capturados mediante el sistema de succión integrado en la trampa.

El mecanismo de generación de  $\text{CO}_2$  se basa en un proceso de fermentación alcohólica, en el cual la levadura metaboliza el azúcar contenido en un medio

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

gelificado. Esta reacción biológica produce una liberación sostenida de  $\text{CO}_2$  en bajas concentraciones, resultando inofensiva tanto para los seres humanos como para el entorno [11].

El uso de un sustrato en forma de gel permite controlar la velocidad de reacción y prolongar la producción del gas en el tiempo. Una vez que la levadura ha consumido completamente el gel (indicado visualmente por la pérdida de color), el sistema debe ser reemplazado para reiniciar el ciclo de generación de  $\text{CO}_2$ .

#### **4.1.1. Materiales**

**Tabla 4.1:** Materiales e ingredientes para un volumen de 1,5 L.

| <b>Material / Ingrediente</b> | <b>Cantidad</b>    |
|-------------------------------|--------------------|
| Agua (preparación inicial)    | 180 mL             |
| Azúcar blanca                 | 300 g              |
| Jarabe de menta               | 30 mL              |
| Gelatina neutra               | 13,5 g             |
| Levadura de panadería         | 2 g                |
| Agua adicional (activación)   | 600 mL             |
| Bidón plástico                | 1,5 L              |
| Manguera de $\text{CO}_2$     | 1 unidad           |
| Pegamento                     | Cantidad necesaria |

**Fuente:** Elaboración propia.

#### **4.1.2. Proceso de Elaboración**

##### **(1) Preparación del gel**

1. Calentar 180 mL de agua hasta ebullición.
2. Añadir el azúcar y remover hasta su completa disolución, formando un jarabe homogéneo.
3. Incorporar el jarabe de menta, que actuará como indicador visual del consumo del sustrato.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

4. Agregar la gelatina neutra mientras se mantiene la agitación a fuego medio hasta lograr una mezcla uniforme.
5. Verter la mezcla aún tibia en el bidón de 1,5 L antes de su solidificación.
6. Dejar reposar durante 24 horas a temperatura ambiente hasta obtener la consistencia de gel.

## **(2) Activación del gel**

1. Añadir la levadura de panadería al gel solidificado.
2. Incorporar 600 mL de agua.
3. Mezclar suavemente con movimientos circulares, evitando agitación brusca que pueda degradar la estructura del gel.

## **(3) Preparación del bidón**

1. Perforar la tapa del bidón en el centro, con un diámetro adecuado para la manguera de CO<sub>2</sub>.
2. Sellar la unión aplicando pegamento para garantizar hermeticidad.
3. Instalar el bidón dentro de la estación.
4. Guiar la manguera a través de la plataforma intermedia, ubicándola cerca de la compuerta de succión.

## **(4) Reactivación del gel**

1. Cada 4 días, retirar aproximadamente 300 mL de la solución.
2. Añadir 300 mL de agua fresca al bidón.
3. Reinstalar el bidón en la estación.



La degasificación periódica mejora la producción continua de  $\text{CO}_2$ , incrementando la eficiencia del cebo.

#### 4.1.3. Liberación de $\text{CO}_2$ del Gel

- **Duración de la liberación:** El bidón de 1,5 litros está diseñado para proporcionar una liberación ininterrumpida de  $\text{CO}_2$  durante 8 días.
- **Tasa de liberación diaria:** Los generadores biológicos de 1,5 litros estabilizados en gel suelen emitir un promedio de 4,5 a 9 gramos de  $\text{CO}_2$  al día. Este flujo bajo pero constante es suficiente para crear una pluma olfativa efectiva en un radio de 5 a 10 metros.
- **Liberación total estimada:** Al finalizar el ciclo de 8 días de fermentación, la mezcla de 1,5 litros habrá liberado un total aproximado de 40 a 70 gramos de  $\text{CO}_2$ .
- **A mayor temperatura:** El metabolismo de la levadura se acelera. Esto resulta en picos de mayor liberación de  $\text{CO}_2$  durante los días calurosos o las primeras horas de la tarde, coincidiendo precisamente con los momentos de mayor actividad metabólica de los mosquitos.
- **A menor temperatura:** Durante las noches frías, el proceso se ralentiza de forma natural, reduciendo el  $\text{CO}_2$  emitido y ayudando a conservar la mezcla durante el ciclo completo de ocho días.

#### 4.2. Diseño y Modelo de la Trampa

En esta sección se presenta el diseño y el principio de funcionamiento de la trampa inteligente desarrollada para la captura, detección y clasificación de mosquitos por sexo. El sistema propuesto integra componentes mecánicos, elec-

trónicos y de visión artificial, conformando una solución mecatrónica orientada a la operación autónoma en entornos reales.

El diseño se encuentra fuertemente inspirado en el sistema propuesto por [28], adaptando y extendiendo sus principios mediante la incorporación de un mecanismo de clasificación física, un sistema de visión basado en aprendizaje automático y una arquitectura de control distribuida.

#### **4.2.1. Mecanismo de la Trampa y su Funcionamiento**

##### **4.2.1.1. Descripción General y Estructura Física**

El sistema mecánico consiste en una trampa entomológica automatizada diseñada para la captura, clasificación (por sexo) y recolección segregada de mosquitos. La estructura principal está conformada por un prisma rectangular con dimensiones de 220 mm de alto, 160 mm de ancho y 45 mm de profundidad. La cara frontal del prisma está sellada con una placa de acrílico transparente, fijada mediante tornillos de 3 mm, la cual permite la inspección visual ininterrumpida de la cámara interna. El prototipo de la trampa fue fabricado mediante impresión 3D utilizando Polylactic Acid (PLA), seleccionado por su facilidad de manufactura, bajo peso y rigidez mecánica.

##### **4.2.1.2. Sistema de Captura y Flujo de Aire**

El principio de captura se basa en la generación de presión negativa. El conducto central es un tubo de sección cuadrada de 20 x 20 mm que atraviesa verticalmente el dispositivo, dividido funcionalmente en tres secciones:

- **Sección superior (Entrada):** Cuenta con un acople de transición cónico de 45 mm de altura, que reduce el diámetro desde una entrada circular de 72 mm hasta la sección cuadrada de 20 x 20 mm.

- **Sección inferior (Succión):** Dispone de otro acople de transición de 50 mm de altura en su extremo final, donde se aloja un ventilador de computadora de 92 mm. Este ventilador extrae el aire del interior, generando un flujo de succión constante desde la entrada superior.

#### 4.2.1.3. Cálculo de la Velocidad del Flujo de Aire

La velocidad en cada punto se determina aplicando la ecuación de continuidad para fluidos incompresibles,  $V = Q/A$ . Las áreas transversales de los dos puntos críticos son:

$$A_{\text{tubo}} = (0,020 \text{ m})^2 = 4,00 \times 10^{-4} \text{ m}^2$$

$$A_{\text{entrada}} = \pi (0,036 \text{ m})^2 \approx 4,07 \times 10^{-3} \text{ m}^2$$

El ventilador axial de 92 mm posee un caudal nominal de 40 CFM ( $Q_0 = 0,0188 \text{ m}^3/\text{s}$ ) en descarga libre. Sin embargo, bajo alta presión estática los ventiladores axiales experimentan caídas drásticas de caudal [39, 40].

Dado que no se realizó una caracterización experimental de la curva presión-caudal del ventilador dentro del conducto, se adoptó una estimación conservadora equivalente al 15 % del caudal nominal, basada en pérdidas esperadas por restricción geométrica y resistencia aerodinámica.

$$Q_{\text{real}} = 0,0188 \times 0,15 \approx 2,82 \times 10^{-3} \text{ m}^3/\text{s} \quad (4.1)$$

Sustituyendo en la ecuación de continuidad se obtiene el perfil de velocidades:

$$V_{\text{tubo}} = \frac{Q_{\text{real}}}{A_{\text{tubo}}} = \frac{2,82 \times 10^{-3}}{4,00 \times 10^{-4}} = 7,05 \text{ m/s}$$
$$V_{\text{entrada}} = \frac{Q_{\text{real}}}{A_{\text{entrada}}} = \frac{2,82 \times 10^{-3}}{4,07 \times 10^{-3}} \approx 0,69 \text{ m/s}$$

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

La velocidad en la entrada (0,69 m/s) resulta suficientemente suave para atraer al vector sin generar turbulencias disuasorias. Una vez en el interior, la reducción geométrica acelera el flujo a 7,05 m/s, superando ampliamente la velocidad típica de vuelo del mosquito (aprox. 1,0–1,5 m/s), lo que dificulta significativamente su escape una vez ingresado al conducto.

#### **4.2.1.4. Emisión Acústica**

El sistema registra un nivel de presión sonora de 75 dB en el núcleo del ventilador, el cual se atenúa hasta los 60 dB en la entrada exterior de la trampa. Esta reducción es fundamental para el diseño, ya que 60 dB equivale a un nivel de ruido ambiental moderado, el cual no se espera que produzca un efecto disuasorio significativo durante la aproximación al cebo.

#### **4.2.1.5. Sistema de Visión e Identificación**

La cámara se monta sobre un soporte desmontable ensamblado a presión, posicionándola a 100 mm del panel frontal de acrílico. Esta configuración permite enfocar directamente la región de captura del conducto central. Para garantizar una iluminación homogénea e independiente de las condiciones ambientales, el montaje incluye una barra de luces LED situada en la parte superior del soporte de la cámara.

#### **4.2.1.6. Mecanismo de Clasificación y Recolección (Sección Media)**

El núcleo operativo de la trampa es la segunda sección del conducto, compuesta por un cilindro rotatorio de 140 mm de diámetro integrado de forma perpendicular al eje vertical del flujo de aire. Las características de este mecanismo son:

- **Tubo deslizante interno:** Dentro del conducto cuadrado del cilindro giratorio, se aloja un tubo cuadrado secundario y hueco (con paredes de 2 mm de grosor y una longitud equivalente a la mitad del conducto del cilindro). Este tubo interior está dimensionado con tolerancias justas para deslizarse libremente sin fricción hacia arriba y hacia abajo. Sus extremos están provistos de una red mosquitera.
- **Actuación y Posicionamiento:** La rotación del cilindro es controlada por un servomotor de rotación continua (360°), montado en la parte posterior mediante tornillos de 3 mm. El posicionamiento exacto del cilindro se garantiza mediante un sistema de retroalimentación magnética, utilizando imanes incrustados de 4 mm y un sensor de efecto Hall.
- **Recámaras de almacenamiento:** Flanqueando la tercera sección del tubo vertical (la parte inferior), se encuentran dos recámaras de recolección aisladas, destinadas a separar a los mosquitos hembra de los machos.

#### 4.2.1.7. Ciclo de Operación

El funcionamiento de la trampa sigue una secuencia cíclica y automatizada:

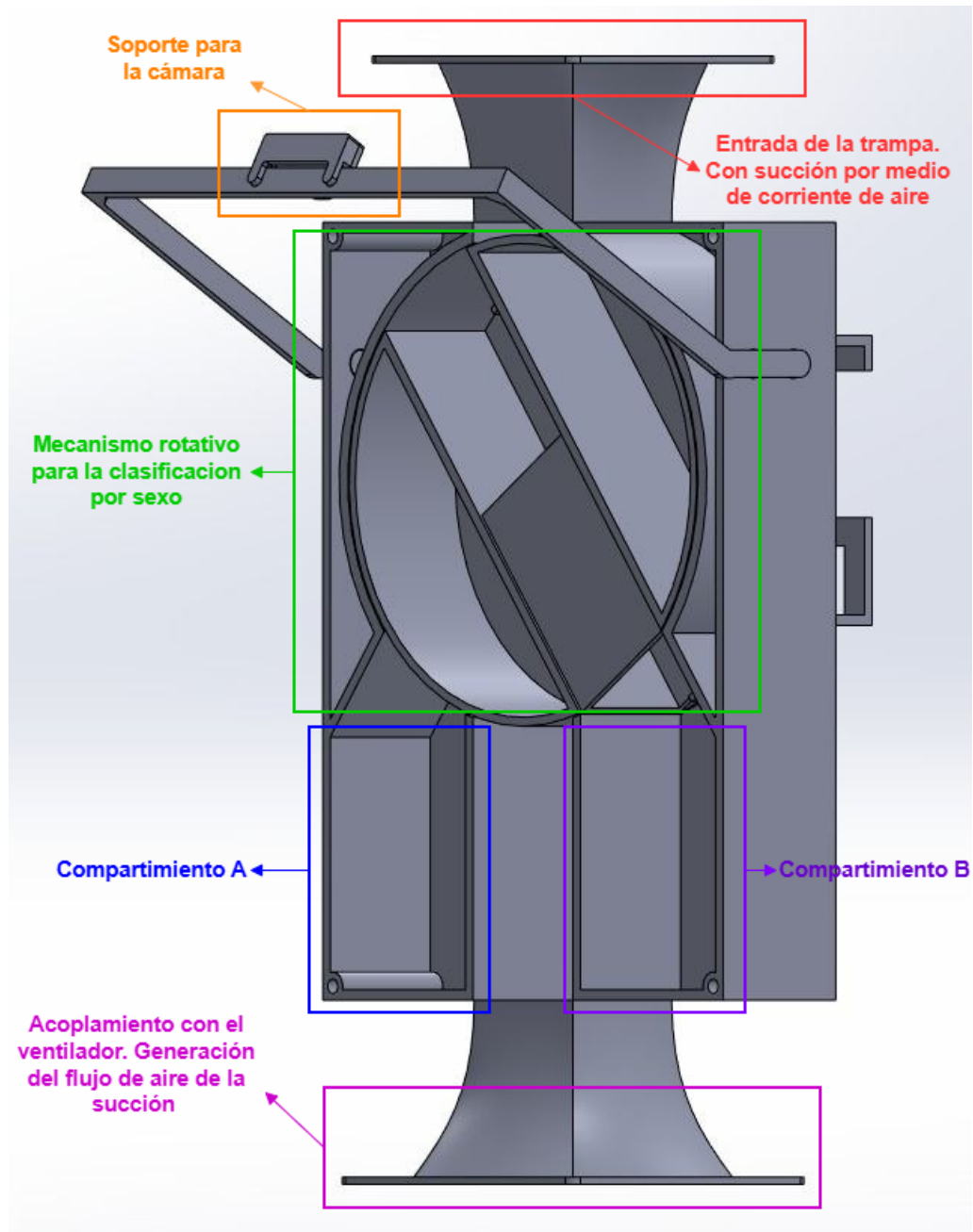
1. **Captura:** El ventilador inferior genera presión negativa. El mosquito es succionado por la toma de 72 mm, atraviesa la primera sección y entra al cilindro giratorio, donde el flujo de aire lo inmoviliza contra la red mosquitera del tubo interior deslizante.
2. **Identificación:** El espécimen, retenido contra la malla visible a través del acrílico frontal y contrastado contra el PLA blanco, es iluminado por la barra LED y fotografiado por la cámara. El sistema procesa la imagen y determina si el individuo es macho o hembra.

3. **Segregación:** Una vez clasificado, el servomotor hace girar el cilindro alineando su conducto interno con una de las dos recámaras laterales correspondientes.
4. **Expulsión por gravedad:** Al girar el cilindro hacia una posición inclinada/horizontal, la gravedad actúa sobre el tubo interior deslizante. Este cae libremente hacia el extremo inferior, actuando como un émbolo que empuja y expulsa al mosquito hacia la recámara de recolección seleccionada.
5. **Reinicio:** El servomotor devuelve el cilindro a su posición vertical inicial, el tubo deslizante vuelve a su posición de reposo permitiendo de nuevo el flujo de aire ininterrumpido, y el sistema queda listo para la siguiente captura.

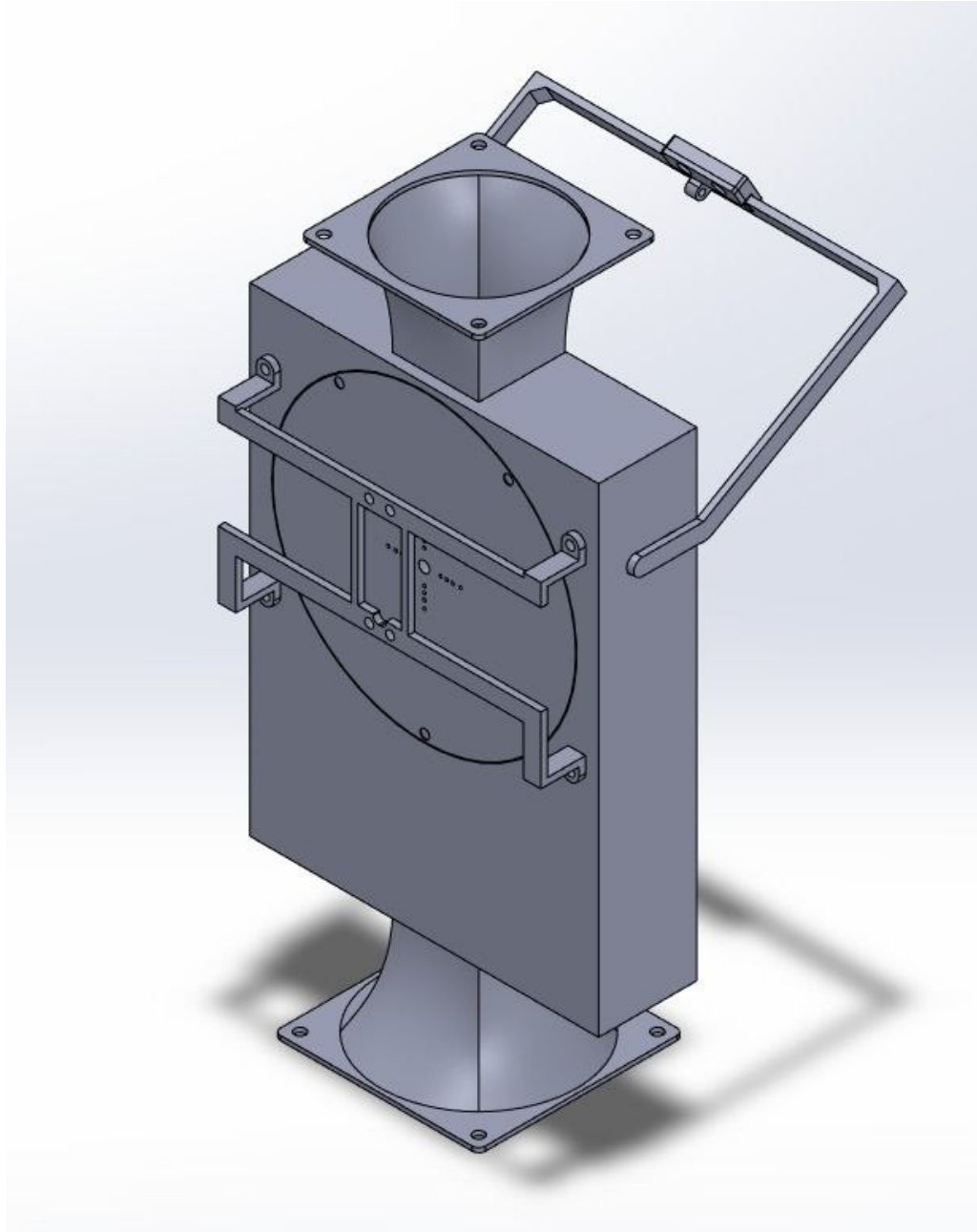
#### 4.2.2. Modelo 3D

En la Figura 4.1 y la Figura 4.2 se presentan el diseño 3D del mecanismo completo de la trampa. En la Figura 4.3 se puede visualizar la representación del mecanismo de la trampa dentro de la caja protectora con el soporte para el Raspberry Pi 5.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

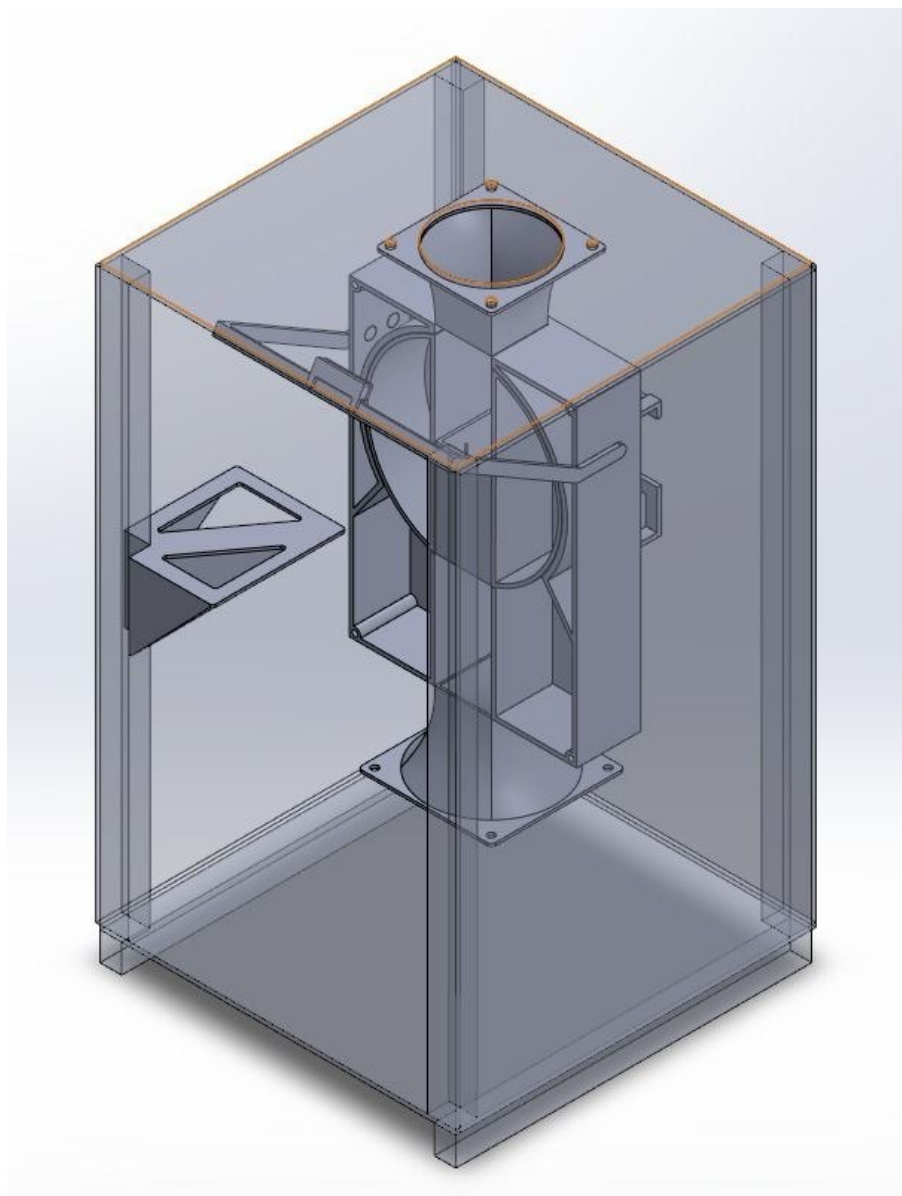


**Figura 4.1:** Modelo 3D de la trampa con descripción de sus partes principales.  
**Fuente:** Elaboración propia.



**Figura 4.2:** Vista trasera del modelo 3D de la trampa, donde se aprecian el soporte de montaje del mecanismo de clasificación y la salida del canal de ventilación.  
**Fuente:** Elaboración propia.





**Figura 4.3:** Modelo 3D de la trampa integrada dentro de la caja protectora, mostrando la disposición del mecanismo de captura, el sistema de visión artificial y el soporte para la Raspberry Pi 5. **Fuente:** Elaboración propia.

### 4.3. Componentes Utilizados

#### 4.3.1. Lista de Componentes

A continuación se describen los principales componentes del sistema y su función dentro de la arquitectura de la trampa. Se destaca que algunos dispositivos de mayor costo, como la Raspberry Pi 5 y la Raspberry Pi Camera Module

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

3 NoIR, fueron provistos por el Laboratorio de Mecánica y Energía (LME). Su selección responde tanto a la disponibilidad como a su capacidad para cumplir con los requerimientos de procesamiento e inferencia en tiempo real, así como a la calidad de adquisición de imágenes necesaria para la detección de objetos de pequeño tamaño. No obstante, es importante señalar que el sistema puede implementarse con alternativas de menor costo (por ejemplo, Raspberry Pi 4), con posibles compromisos en el rendimiento. El resto de los componentes fue seleccionado priorizando bajo costo, disponibilidad y facilidad de integración, características adecuadas para un prototipo funcional.

**Raspberry Pi 5.** La Raspberry Pi 5 actúa como unidad central de procesamiento del sistema. Se encarga de la ejecución de los modelos de visión artificial, el procesamiento de imágenes, los hilos para la subida de datos continua al servidor y la coordinación general de los demás módulos. La comunicación con el sistema de control se realiza mediante interfaz serial con el Arduino.

**Tabla 4.2:** Especificaciones técnicas de la Raspberry Pi 5.

| Característica           | Especificación                     |
|--------------------------|------------------------------------|
| CPU                      | Quad-Core ARM Cortex-A76 @ 2,4 GHz |
| Memoria RAM              | 4–16 GB LPDDR4X                    |
| GPU                      | VideoCore VII                      |
| Almacenamiento           | MicroSD / PCIe (NVMe)              |
| Conectividad             | Wi-Fi, Bluetooth, Ethernet         |
| Puertos                  | USB, HDMI, GPIO                    |
| Aplicación en el sistema | Procesamiento e inferencia         |

**Fuente:** Elaboración propia a partir de los datos oficiales del fabricante.

**Raspberry Pi Camera Module 3 NoIR.** Este módulo se utiliza para la adquisición de imágenes del sistema. Su sensor de alta resolución y enfoque automático Phase Detection Autofocus (PDAF) permiten obtener imágenes nítidas del mosquito en el punto de captura, condición fundamental para la correcta clasificación.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

**Tabla 4.3:** Especificaciones técnicas de la Raspberry Pi Camera Module 3 NoIR.

| Característica           | Especificación      |
|--------------------------|---------------------|
| Sensor                   | Sony IMX708         |
| Resolución               | ~12 MP              |
| Enfoque                  | Automático (PDAF)   |
| Aplicación en el sistema | Captura de imágenes |

**Fuente:** Elaboración propia a partir de los datos oficiales del fabricante.

**Arduino UNO R3.** El Arduino UNO se emplea como unidad de control secundaria para tareas de tiempo real, principalmente el accionamiento del servomotor y la lectura de sensores, el cual es controlado mediante Universal Asynchronous Receiver-Transmitter (UART) por el Raspberry Pi. Esta arquitectura desacopla el control mecánico del procesamiento de alto nivel.

**Tabla 4.4:** Especificaciones técnicas del Arduino UNO R3

| Característica           | Especificación                   |
|--------------------------|----------------------------------|
| Microcontrolador         | ATmega328P                       |
| Frecuencia               | 16 MHz                           |
| Entradas/Salidas         | 14 digitales, 6 analógicas       |
| Comunicación             | UART, I2C, SPI                   |
| Aplicación en el sistema | Control de actuadores y sensores |

**Fuente:** Elaboración propia a partir de los datos oficiales del fabricante.

**Servomotor de rotación continua HK15138.** El servomotor acciona el mecanismo de clasificación mediante la rotación del cilindro interno. Su control se realiza mediante señales Pulse Width Modulation (PWM) desde el Arduino.

**Tabla 4.5:** Especificaciones técnicas del servomotor

| Característica           | Especificación         |
|--------------------------|------------------------|
| Tipo                     | Rotación continua      |
| Voltaje                  | 4,8–6 V                |
| Control                  | PWM                    |
| Aplicación en el sistema | Accionamiento mecánico |

**Fuente:** Elaboración propia a partir de los datos oficiales del fabricante.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

**Regulador de Voltaje LM2596.** El módulo regulador LM2596 es utilizado para disminuir el voltaje de la fuente principal a niveles adecuados para los distintos módulos del sistema.

**Tabla 4.6:** Especificaciones técnicas del Regulador Buck LM2596

| Característica           | Especificación                                |
|--------------------------|-----------------------------------------------|
| Tipo                     | Convertidor DC-DC Buck (reductor)             |
| Voltaje de entrada       | 4,0 – 40 V DC                                 |
| Voltaje de salida        | 1,25 – 37 V DC ajustable                      |
| Corriente máxima         | 3 A (con disipación adecuada)                 |
| Eficiencia               | Hasta 92 %                                    |
| Aplicación en el sistema | Regulación de la alimentación para el Arduino |

**Fuente:** Elaboración propia a partir de los datos oficiales del fabricante.

**Sensor BME280.** El BME280 es un sensor ambiental digital de alta integración capaz de medir temperatura, humedad relativa y presión atmosférica en un solo dispositivo compacto. En este proyecto fue utilizado para monitorear las condiciones ambientales internas o externas del sistema, permitiendo correlacionar variables climáticas con la actividad de los mosquitos y el desempeño operativo del dispositivo. Su bajo consumo energético y compatibilidad con buses I<sup>2</sup>C y SPI lo convierten en una opción eficiente para plataformas embebidas.

**Tabla 4.7:** Especificaciones técnicas del Sensor BME280

| Característica           | Especificación                 |
|--------------------------|--------------------------------|
| Variables medidas        | Temperatura, humedad, presión  |
| Rango de temperatura     | -40 a 85 °C                    |
| Rango de humedad         | 0 – 100 % RH                   |
| Rango de presión         | 300 – 1100 hPa                 |
| Precisión típica         | ±1 °C, ±3 % RH                 |
| Interfaz de comunicación | I <sup>2</sup> C / SPI         |
| Voltaje de operación     | 1,8 – 3,6 V                    |
| Consumo energético       | Muy bajo (< 1 mA típico)       |
| Aplicación en el sistema | Monitoreo y registro ambiental |

**Fuente:** Elaboración propia a partir de los datos oficiales del fabricante.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

**Relé 5V 1 Canal.** El módulo relé de 5 V y 1 canal es un interruptor electro-mecánico controlado electrónicamente, utilizado para activar o desactivar cargas de mayor voltaje o corriente mediante una señal lógica de bajo nivel. Se emplea para la conmutación de cargas externas, permitiendo el control de dispositivos auxiliares.

**Tabla 4.8:** Especificaciones técnicas del Relé 5V de 1 Canal

| Característica           | Especificación                   |
|--------------------------|----------------------------------|
| Tipo                     | Relé electromecánico de 1 canal  |
| Voltaje de activación    | 5 V DC                           |
| Señal de control         | TTL digital                      |
| Capacidad de carga       | 10 A @ 250 V AC / 10 A @ 30 V DC |
| Contactos                | NO / NC / COM                    |
| Aislamiento              | Optoacoplador (según módulo)     |
| Aplicación en el sistema | Conmutación de cargas externas   |

**Fuente:** Elaboración propia a partir de los datos oficiales del fabricante.

**Sensor Magnético Hall A3144.** El A3144 es un sensor magnético digital basado en efecto Hall, diseñado para detectar la presencia de campos magnéticos generados por imanes permanentes. Se utiliza para detectar la posición del mecanismo rotatorio mediante retroalimentación magnética.

**Tabla 4.9:** Especificaciones técnicas del Sensor Magnético Hall A3144

| Característica           | Especificación                |
|--------------------------|-------------------------------|
| Tipo                     | Sensor Hall digital unipolar  |
| Voltaje de operación     | 4,5 – 24 V DC                 |
| Salida                   | Digital tipo colector abierto |
| Aplicación en el sistema | Detección de posición         |

**Fuente:** Elaboración propia a partir de los datos oficiales del fabricante.

**Ventilador PC 92 mm.** Ventilador típico usado en computadoras de mesa; genera el flujo de aire necesario para la captura de los mosquitos mediante presión negativa.

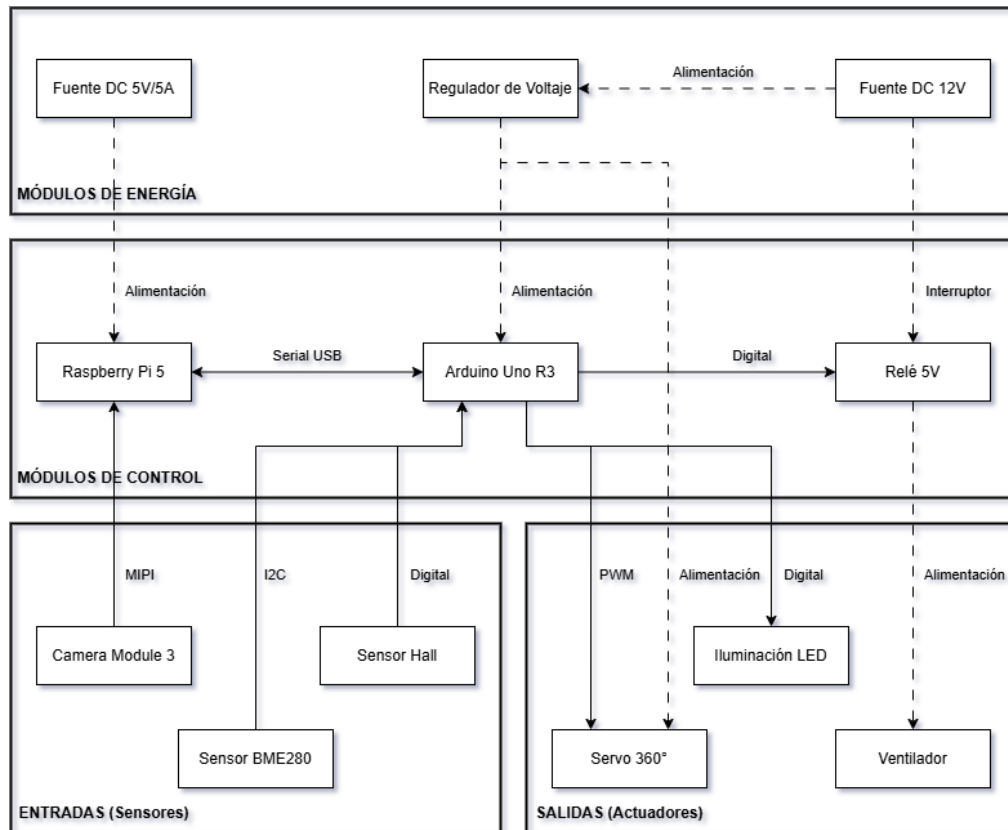
**Fuente 12V.** Proporciona la energía eléctrica al sistema, siendo distribuida y regulada según los requerimientos de cada componente.

#### 4.3.2. Diagrama de Bloques: Arquitectura de Hardware

Este esquema muestra la jerarquía de control y la distribución de energía del sistema.

- **Bloque de Procesamiento Maestro (Raspberry Pi 5):** Es el núcleo de toma de decisiones. Se encarga de capturar la imagen a través del Módulo de Cámara 3, procesar el algoritmo de visión artificial para clasificar el sexo del mosquito, y enviar la orden de movimiento (A o B) al microcontrolador. Cuenta con su propia fuente aislada para evitar reinicios por consumo de motores.
- **Bloque de Control Esclavo (Arduino Uno R3):** Actúa como la interfaz de hardware en tiempo real. Traduce las órdenes del Raspberry Pi en señales eléctricas (PWM para el servo, señales lógicas para el relé y LEDs) y recopila datos ambientales (BME280) y de posicionamiento del motor (Sensor Hall).
- **Bloque de Gestión de Potencia:** Liderado por el Regulador LM2596, este bloque toma los 12V de la fuente principal y los estabiliza a 5V para alimentar el Arduino, el servomotor y la lógica del relé. La línea de 12V pura pasa exclusivamente por el relé para alimentar el ventilador, aislando el ruido electromagnético.
- **Bloque de Interfaz Física (Sensores y Actuadores):** Comprende el mecanismo de captura (Ventilador), el mecanismo de iluminación y retroalimentación (LEDs y Sensor Hall), la medición de variables de estudio (BME280) y el actuador de clasificación (Servomotor 360°).

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**



**Figura 4.4:** Diagrama de bloques de la arquitectura de hardware de la trampa, organizado en cuatro bloques funcionales: procesamiento maestro (Raspberry Pi 5), control esclavo (Arduino UNO), gestión de potencia (LM2596) e interfaz física (sensores y actuadores). **Fuente:** Elaboración propia.

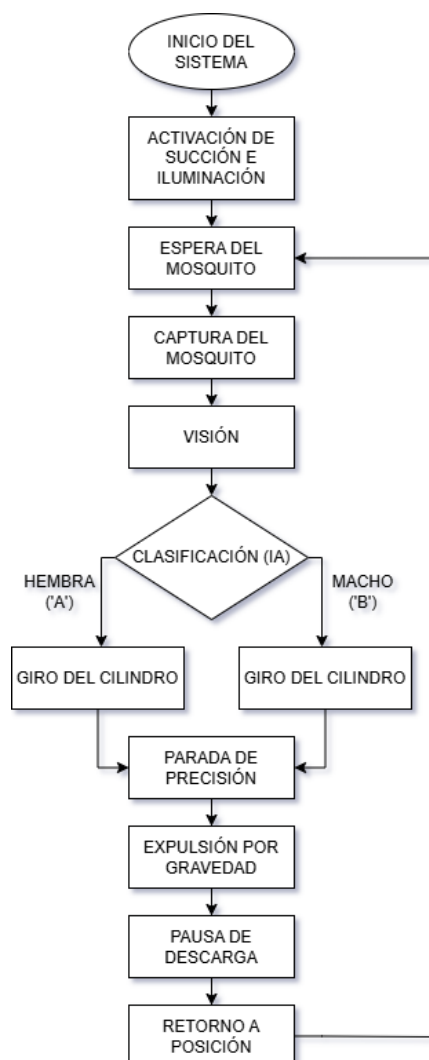
#### 4.3.3. Diagrama de Flujo: Lógica y Flujo del Proceso

Este segundo diagrama representa la máquina de estados y el flujo físico del mosquito en sincronía con el software.

- **Activación de Succión e Iluminación:** Arduino activa Relé → Ventilador ON, e Iluminación → LED ON.
- **Espera del Mosquito:** El sistema espera la captura de un espécimen.
- **Captura del Mosquito:** El mosquito queda retenido contra la malla del tubo interior.
- **Visión:** Raspberry Pi captura imagen.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

- **Clasificación (IA):** Raspberry Pi procesa imagen: ¿Macho o Hembra?
- **Giro del Cilindro:** Servo gira a Posición A o B.
- **Parada de Precisión:** Sensor Hall detecta imán de posición
- **Expulsión por Gravedad:** Tubo interior cae → Expulsa mosquito a recámara
- **Pausa de Descarga:** Espera de 2 segundos
- **Retorno a Posición:** Servo gira de regreso a posición central vertical



**Figura 4.5:** Diagrama de flujo físico del mosquito en sincronía con el software.  
**Fuente:** Elaboración propia.



#### 4.3.4. Diagrama Eléctrico

El diagrama eléctrico del sistema, mostrado en la Figura 4.6, describe la interconexión entre los módulos previamente definidos, así como la distribución de las líneas de alimentación y señales de control.

El sistema se organiza en dos dominios de alimentación claramente diferenciados: una línea de 5V para lógica y control, y una línea de 12V destinada a cargas de potencia.

**Alimentación del Sistema.** La fuente principal de 12V ingresa mediante un conector Jack DC y se distribuye en dos ramas:

- **Rama de potencia (12V):** alimenta directamente al ventilador a través del contacto del relé, evitando que esta carga afecte la electrónica de control.
- **Rama regulada (5V):** un convertidor step-down LM2596 reduce los 12V a 5V estables. Esta línea alimenta el pin de 5V del Arduino Uno, el servomotor y el módulo relé (lógica de activación).

La Raspberry Pi 5 se alimenta de forma independiente mediante una fuente dedicada de 5V a través de USB-C, evitando interferencias por variaciones de carga.

**Interconexión entre Controladores.** La comunicación entre la Raspberry Pi 5 y el Arduino Uno se realiza mediante un enlace USB, permitiendo la transmisión de comandos y datos de forma serial.

**Conexión de Sensores.** Los sensores se conectan directamente al Arduino, que actúa como unidad de adquisición:

- El sensor BME280 se conecta mediante interfaz Inter-Integrated Circuit (I2C): SDA → A4, SCL → A5.
- El sensor de efecto Hall entrega una señal digital conectada al pin 11.

- Ambos sensores comparten alimentación de 5V y referencia común a GND.

**Conexión de Actuadores.** Los actuadores son controlados por el Arduino mediante salidas digitales:

- **Servomotor:**

- Señal PWM → pin 9
- Alimentación → 5V (desde el LM2596)

- **Módulo relé:**

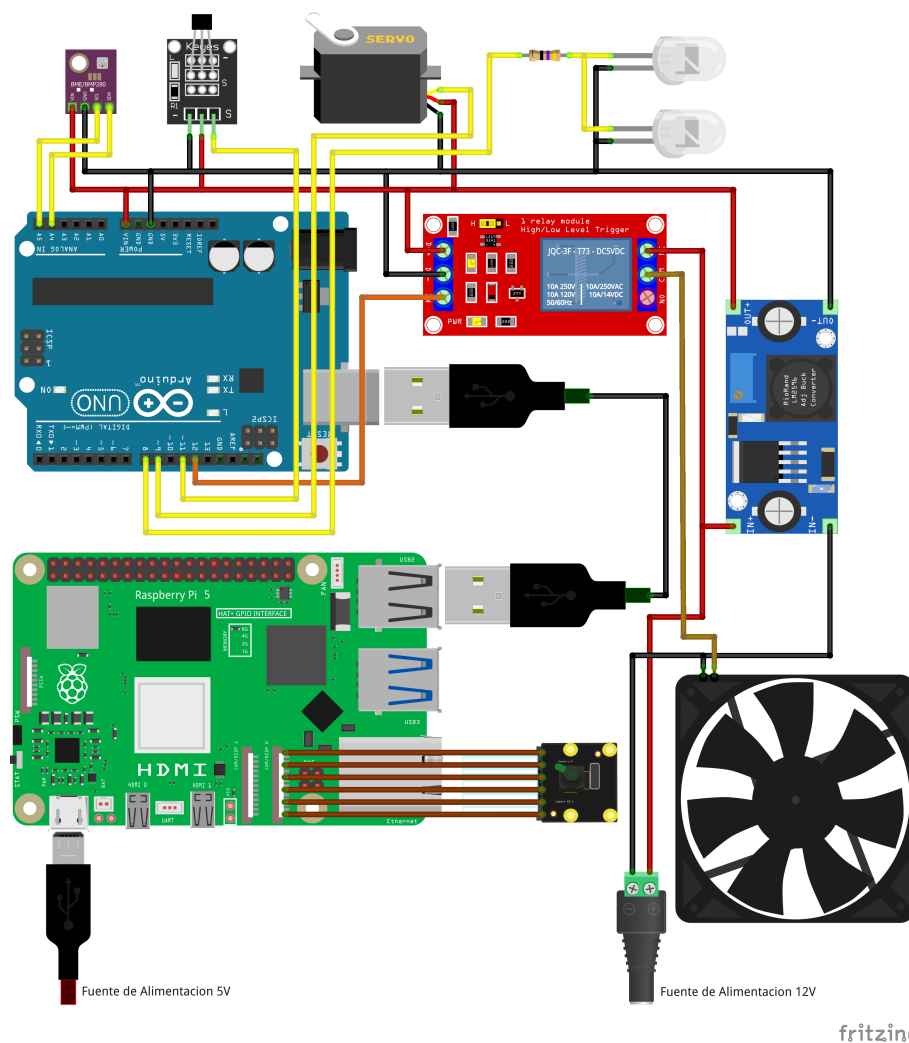
- Señal de control → pin 12
- El relé conmuta la línea de 12V que alimenta el ventilador

- **LEDs indicadores:**

- Control → pin 8
- Con resistencia limitadora en serie y retorno a GND

Todas las tierras (GND) del sistema se encuentran unificadas para garantizar una referencia común de señal.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS



**Figura 4.6:** Diagrama eléctrico de los componentes de la trampa. **Fuente:** Elaboración propia.

#### 4.4. Lista Final de Precios

A continuación se detallan los precios de referencia correspondientes al mes de mayo de 2026, junto al proveedor consultado, para todos los componentes y materiales necesarios para la construcción del prototipo de la trampa. La conversión entre guaraníes y dólares estadounidenses se realizó utilizando la cotización vigente para ese mismo período.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

**Tabla 4.10:** Presupuesto de materiales clasificado por categoría

| #                                 | Descripción                | Cant. | P. Unitario<br>(PYG) | P. Total<br>(PYG) | Proveedor                 |
|-----------------------------------|----------------------------|-------|----------------------|-------------------|---------------------------|
| <b>Componentes electrónicos</b>   |                            |       |                      |                   |                           |
| 1                                 | Servo 360 HK15138          | 1     | 85.000               | 85.000            | Electromer S.A.           |
| 2                                 | Relé 5V 1 Canal            | 1     | 35.000               | 35.000            | Electromer S.A.           |
| 3                                 | Regulador LM2596           | 1     | 40.000               | 40.000            | Electromer S.A.           |
| 4                                 | Sensor BME280              | 1     | 70.000               | 70.000            | Electromer S.A.           |
| 5                                 | Sensor Hall A3144          | 1     | 40.000               | 40.000            | Electrónica Plett<br>S.A. |
| 6                                 | Arduino UNO R3             | 1     | 185.000              | 185.000           | Electrónica Plett<br>S.A. |
| 7                                 | Ventilador PC 92 mm        | 1     | 60.000               | 60.000            | Electrónica Plett<br>S.A. |
| 8                                 | Jumpers Dupont             | 20    | 500                  | 10.000            | Electromer S.A.           |
| 9                                 | Camera Module 3 NoIR       | 1     | 173.580              | 173.580           | Cytron Technolo-<br>gies  |
| 10                                | Raspberry Pi 5             | 1     | 1.199.280            | 1.199.280         | Cytron Technolo-<br>gies  |
| 11                                | Fuente 12V                 | 1     | 40.000               | 40.000            | Electromer S.A.           |
| <b>Materiales de construcción</b> |                            |       |                      |                   |                           |
| 12                                | Impresión 3D PLA Blanco    | 1     | 300.000              | 300.000           | Grilon3                   |
| 13                                | Madera Terciada 3 mm       | 1     | 53.500               | 53.500            | Bricopar S.A.             |
| 14                                | Pintura Aerosol Negra      | 1     | 30.000               | 30.000            | Amanecer                  |
| 15                                | Pintura Acrílica Blanca    | 1     | 15.500               | 15.500            | Acrilex                   |
| 16                                | Clavos 3/4"                | 50    | 100                  | 5.000             | Ferretería                |
| 17                                | Red Mosquitero             | 1     | 15.000               | 15.000            | Ferretería                |
| 18                                | Bidón plástico 1,5 L       | 1     | 8.000                | 8.000             | SANSEI S.A.               |
| 19                                | Manguera Transparente 5 mm | 5     | 2.600                | 2.600             | SANSEI S.A.               |

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

**Tabla 4.10 – continuación**

| #                              | Descripción                | Cant. | P. Unitario<br>(PYG) | P. Total<br>(PYG) | Proveedor              |
|--------------------------------|----------------------------|-------|----------------------|-------------------|------------------------|
| 20                             | Imán Nd 4x2 mm             | 3     | 5.000                | 15.000            | Electrónica Plett S.A. |
| 21                             | Imán Nd 8x3 mm             | 2     | 6.000                | 12.000            | Porter S.A.            |
| 22                             | Acrílico Transparente 3 mm | 1     | 32.000               | 32.000            | Luxacril S.A.          |
| 23                             | Tornillo 3 mm              | 20    | 500                  | 10.000            | Electromer S.A.        |
| 24                             | Bisagra 25 mm              | 2     | 4.000                | 8.000             | Ferretería             |
| <b>Materiales para el cebo</b> |                            |       |                      |                   |                        |
| 25                             | Levadura 10 g              | 1     | 1.550                | 1.550             | Copalsa S.A.           |
| 26                             | Gelatina 24 g              | 1     | 6.500                | 6.500             | Bretzke S.A.           |
| 27                             | Menta Fresca               | 1     | 1.500                | 1.500             | Superseis S.A.         |
| 28                             | Azúcar 300 g               | 1     | 4.500                | 4.500             | Superseis S.A.         |
| <b>Cotización USD/PYG</b>      |                            |       |                      |                   | 6.312                  |
| <b>Total USD</b>               |                            |       |                      |                   | 389,50                 |
| <b>Total PYG</b>               |                            |       |                      |                   | 2.458.510              |

**Fuente:** Elaboración propia a partir del mercado global.

El costo total del prototipo fue de **USD 389,50**. Sin embargo, los componentes más costosos —la Raspberry Pi 5 y la cámara NoIR— fueron provistos por el LME, por lo que el costo podría reducirse utilizando alternativas más económicas.

Como referencia, el sistema comercial *BG-Counter* de Biogents [10] tiene un costo a la fecha de **USD 2.580,00**, más de seis veces superior al del prototipo desarrollado, lo que evidencia el potencial de la propuesta como alternativa de bajo costo para aplicaciones de vigilancia entomológica automatizada.

## CAPÍTULO 5

### Modelo de Detección y Clasificación

Este Capítulo documenta el desarrollo de los modelos de inteligencia artificial que conforman la fase FDT2 del proyecto. A diferencia de los antecedentes resumidos en la Tabla 2.1, cuyo objetivo central es la detección o discriminación de especies, el presente trabajo apunta a la **clasificación sexual automatizada de *Aedes aegypti* en dispositivos de borde**. Para ello se adopta una arquitectura en cascada: un detector YOLOv8 de clase única localiza el mosquito en la imagen —siguiendo el esquema de detección en tiempo real sobre Raspberry Pi demostrado por [27]— y un clasificador MobileNetV3 Large discrimina el sexo del espécimen detectado, aprovechando la capacidad de las redes CNN para capturar características morfológicas finas validada por [29]. Ambos modelos se exportan al formato ONNX para inferencia eficiente en el dispositivo de borde descrito en el Capítulo 4 [21, 23]. Este desacoplamiento entre detección y clasificación permite además optimizar cada componente de forma independiente y dotar al sistema de escalabilidad natural: incorporar una tercera etapa de discriminación de especie tendría un costo de entrenamiento e inferencia marginal, al operar únicamente sobre las detecciones ya filtradas por las etapas previas. Para cada modelo se describe el dataset utilizado, el proceso de entrenamiento, y los resultados obtenidos tanto en PyTorch como en ONNX. El entrenamiento y la evaluación de métricas se realizaron en una PC equipada con un procesador Intel Core i7 de 14<sup>a</sup> generación y una GPU NVIDIA GeForce RTX 5060 Laptop; los tiempos de inferencia en el dispositivo de borde se reportan al final del Capítulo.

## 5.1. Modelo de Detección (YOLO)

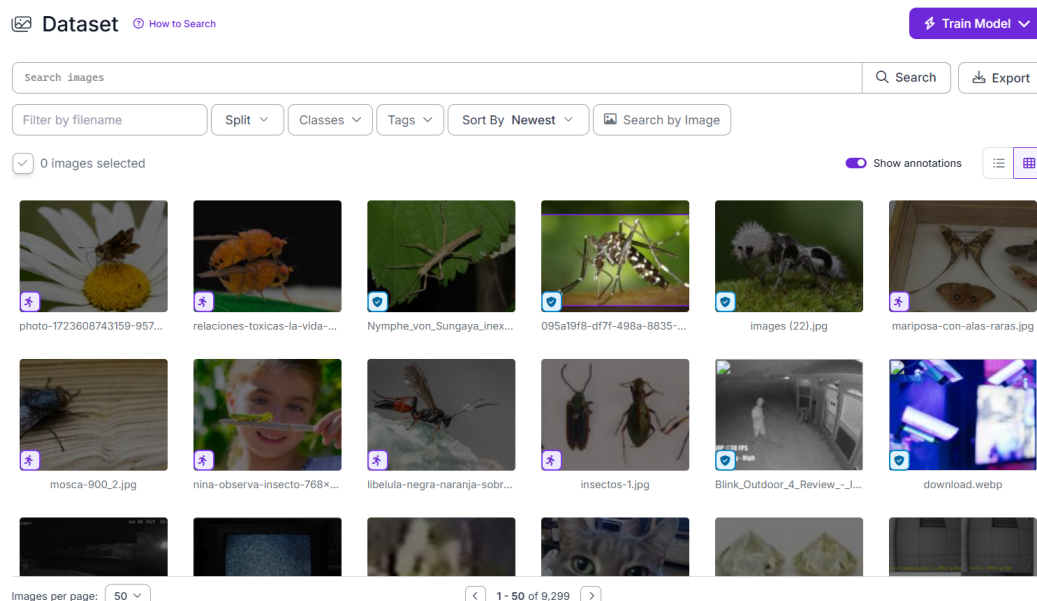
La primera etapa del sistema en cascada está a cargo de un detector YOLOv8, cuya función es localizar cada espécimen en la imagen antes de que el clasificador opere sobre él. La variante *small* (yolov8s) fue adoptada por su equilibrio entre precisión y costo computacional: incorpora mejoras en la arquitectura respecto a versiones anteriores de la familia YOLO, facilita la exportación a ONNX y ha demostrado operar en tiempo real sobre Raspberry Pi en trabajos previos de monitoreo de *Aedes aegypti* [27, 28]. Las secciones siguientes describen el dataset utilizado, la búsqueda de hiperparámetros, el proceso de entrenamiento y los resultados obtenidos en PyTorch y en ONNX.

### 5.1.1. Dataset utilizado

El dataset utilizado fue obtenido inicialmente de la plataforma Roboflow Universe, y posteriormente ampliado mediante la incorporación y etiquetado manual de nuevas imágenes. Este proceso permitió aumentar la variabilidad de las condiciones de captura, incluyendo diferentes fondos, iluminaciones y posiciones del mosquito, mejorando así la capacidad de generalización del modelo.

Con el objetivo de simplificar el pipeline de inferencia y facilitar la exportación del modelo a formato ONNX, el dataset fue definido con una única clase: *mosquito*. Esta decisión reduce la complejidad del modelo y permite optimizar su desempeño en entornos de ejecución restringidos.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**



**Figura 5.1:** Vista general del dataset de detección en la plataforma Roboflow, compuesto por 22.309 imágenes anotadas (clase única: *mosquito*). **Fuente:** Elaboración propia.

Adicionalmente, se aplicaron técnicas de *data augmentation* para enriquecer el conjunto de entrenamiento. En particular, se generaron tres variantes por cada imagen original, utilizando las siguientes transformaciones:

- Rotaciones de 90° (sentido horario y antihorario)
- Variaciones de brillo en el rango de -19 % a +19 %
- Adición de ruido en hasta el 1,45 % de los píxeles

A continuación, se presentan las estadísticas del dataset:

**Tabla 5.1:** Estadísticas del dataset utilizado.

| Descripción                   | Imágenes      | Porcentaje     |
|-------------------------------|---------------|----------------|
| Entrenamiento                 | 19.530        | 87,5 %         |
| Validación                    | 1.400         | 6,3 %          |
| Prueba                        | 1.379         | 6,2 %          |
| <b>Total</b>                  | <b>22.309</b> | <b>100,0 %</b> |
| Instancias clase 0 (mosquito) | 17.140        |                |

**Fuente:** Elaboración propia.



### 5.1.2. Búsqueda de hiperparámetros

La selección adecuada de hiperparámetros es un factor determinante en el desempeño de modelos de aprendizaje profundo, especialmente en arquitecturas de detección de objetos como YOLOv8. Estos parámetros controlan aspectos críticos del proceso de entrenamiento, tales como la dinámica de convergencia, la regularización del modelo y la capacidad de generalización frente a datos no vistos.

En aplicaciones desplegadas en dispositivos de borde, esta etapa adquiere mayor relevancia, ya que no solo se busca maximizar métricas de precisión, sino también obtener modelos estables, eficientes y robustos ante variaciones del entorno. Un ajuste inadecuado puede derivar en sobreajuste, convergencia inestable o degradación del rendimiento en condiciones reales.

Dado que el espacio de hiperparámetros es altamente no convexo y de gran dimensionalidad, se empleó una estrategia de optimización automatizada basada en el framework Optuna, el cual implementa técnicas de búsqueda eficientes como Tree-structured Parzen Estimator (TPE) para explorar configuraciones prometedoras sin necesidad de una búsqueda exhaustiva.

**Metodología de optimización.** El proceso de búsqueda se formuló como un problema de optimización donde cada conjunto de hiperparámetros constituye un *trial*, y su desempeño es evaluado mediante la métrica mean Average Precision (mAP) a un umbral de Intersection over Union (IoU) de 0,50 (mAP@50) sobre el conjunto de validación.

Para cada *trial*, se entrenó un modelo YOLOv8 de tipo *small* (yolov8s) durante 15 épocas, utilizando imágenes de tamaño  $256 \times 256$  y un tamaño de *batch* de 16, manteniendo constantes estos parámetros para aislar el efecto de los hiperparámetros optimizados.

El espacio de búsqueda incluyó las siguientes categorías:

- **Parámetros de optimización:** tasa de aprendizaje inicial ( $lr_0$ ), factor final de aprendizaje ( $lr_f$ ), momento (*momentum*) y regularización (*weight\_decay*).
- **Estrategia de calentamiento:** número de épocas de *warmup*, utilizado para estabilizar las primeras iteraciones del entrenamiento.
- **Componentes de la función de pérdida:** ponderaciones de las pérdidas de localización (*box*), clasificación (*cls*) y distribución focal (*dfl*), que influyen directamente en el balance entre precisión espacial y discriminación de clases.
- **Data augmentation:** parámetros asociados a transformaciones de color (HSV), volteos (*flip*) y técnicas avanzadas como *mosaic* y *mixup*, las cuales afectan la capacidad de generalización del modelo.

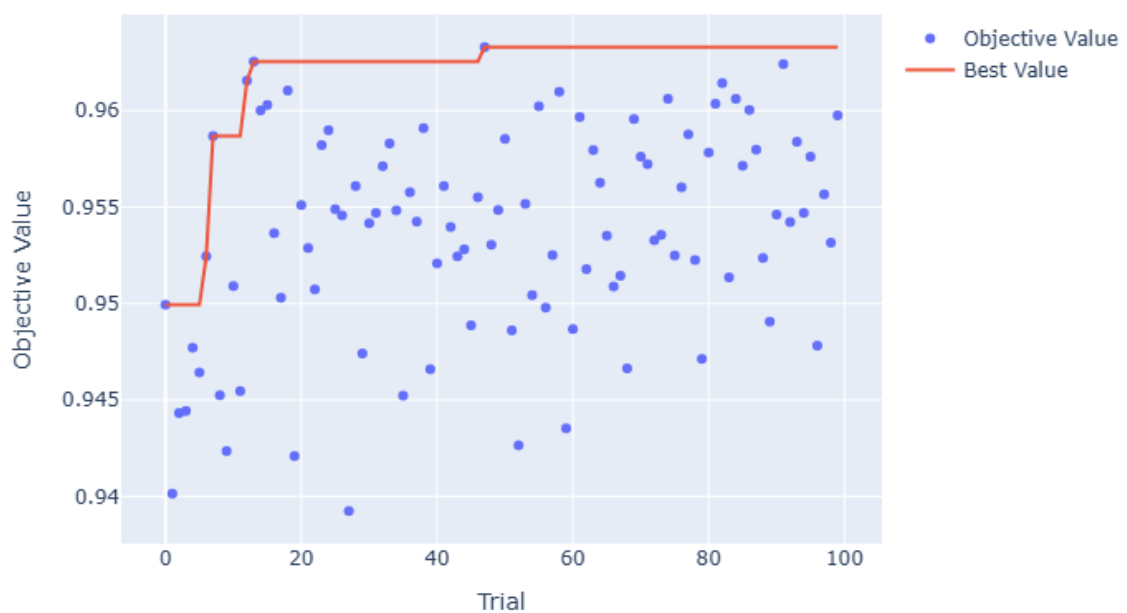
Cada entrenamiento fue seguido por una evaluación explícita sobre el conjunto de validación, utilizando la métrica mAP@50 como función objetivo a maximizar. En caso de fallos durante el entrenamiento, el *trial* era penalizado asignando un valor nulo.

Se ejecutaron un total de 100 *trials*, almacenando los resultados en una base de datos Structured Query Language (SQL) para permitir la reanudación del proceso y el análisis posterior de la importancia de cada hiperparámetro.

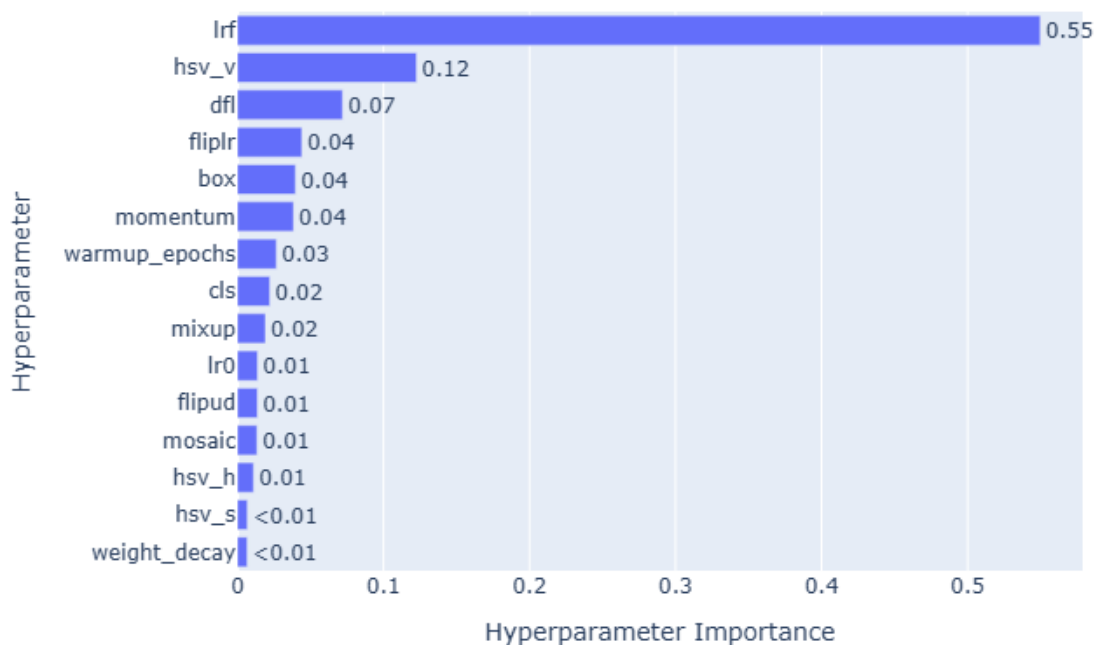
Adicionalmente, se implementaron mecanismos de liberación de memoria (tanto en CPU como en GPU) al finalizar cada *trial*, garantizando la estabilidad del proceso en ejecuciones prolongadas.

**Resultados de la optimización.** El mejor conjunto de hiperparámetros obtenido alcanzó un valor de mAP@50 de 0,9633 sobre el conjunto de validación, en el *trial* 43, como se muestra en la Figura 5.2, evidenciando un alto desempeño del modelo en la tarea de detección.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS



**Figura 5.2:** Historial de optimización de hiperparámetros con Optuna sobre 100 trials. El eje vertical representa la métrica mAP@50 sobre el conjunto de validación; el mejor resultado (mAP@50 = 0,9633) se alcanzó en el trial 43. **Fuente:** Elaboración propia.



**Figura 5.3:** Importancia relativa de los hiperparámetros estimada por Optuna a partir de los 100 trials de búsqueda. Los valores representan la contribución de cada parámetro a la variación de la métrica mAP@50 sobre el conjunto de validación. **Fuente:** Elaboración propia.

Estos resultados reflejan un equilibrio entre una tasa de aprendizaje relativamente alta y mecanismos de regularización moderados, junto con un uso intensivo de técnicas de aumento de datos, lo cual contribuyó a mejorar la capacidad de generalización del modelo sin comprometer su estabilidad.

### 5.1.3. Proceso de entrenamiento

El entrenamiento del modelo de detección se realizó utilizando la implementación de YOLOv8 proporcionada por la librería Ultralytics en Python. El proceso se basó en la carga de un modelo preentrenado (*yolov8s.pt*), el cual es posteriormente ajustado (*fine-tuning*) utilizando el dataset específico del problema.

Para el entrenamiento se cargan los hiperparámetros óptimos previamente obtenidos desde un archivo externo en formato YAML (*best\_hyp.yaml*). Esto permite desacoplar la etapa de optimización del proceso de entrenamiento final.

Posteriormente, se inicializa el modelo YOLOv8 y se ejecuta el entrenamiento. La configuración de parámetros para el entrenamiento usada fue la siguiente:

- **Número de épocas:** 200 iteraciones completas sobre el dataset.
- **Tamaño de imagen:** 256 × 256 píxeles, adecuado para balancear precisión y costo computacional.
- **Batch size:** 16 imágenes por iteración.
- **Dispositivo de ejecución:** GPU (device=0), permitiendo acelerar el entrenamiento.
- **Early stopping:** se detiene el entrenamiento si no hay mejora tras 20 épocas (*patience*).
- **Entrenamiento determinista:** se habilitaron mecanismos de reproducibilidad mediante semillas aleatorias fijas y operaciones deterministas en GPU,

asegurando que los resultados obtenidos puedan replicarse en futuras ejecuciones del experimento.

- **Mosaic augmentation:** desactivado en las últimas 10 épocas para estabilizar el aprendizaje.

Este enfoque modular (búsqueda de hiperparámetros y entrenamiento desacoplado) facilitó la reproducibilidad del entrenamiento.

#### 5.1.4. Resultados y métricas del modelo Pytorch

Los resultados del entrenamiento del modelo YOLOv8 se presentan en la Figura 5.4, donde se muestran las curvas de pérdida y métricas de desempeño a lo largo de las 200 épocas.

**Análisis de convergencia.** Las funciones de pérdida de entrenamiento (*train/box\_loss*, *train/cls\_loss* y *train/dfl\_loss*) presentan una disminución sostenida durante todo el proceso, indicando una adecuada convergencia del modelo. En particular, la pérdida de localización (*box\_loss*) se reduce de 0,016 a 0,007, evidenciando una mejora significativa en la precisión de las cajas delimitadoras.

De manera similar, las pérdidas en validación siguen una tendencia decreciente sin divergencias respecto al entrenamiento, lo que indica ausencia de sobreajuste. Aunque se observa un pico transitorio en *val/cls\_loss* durante las primeras épocas, este comportamiento se estabiliza rápidamente, siendo atribuible a la fase inicial de calentamiento del modelo.

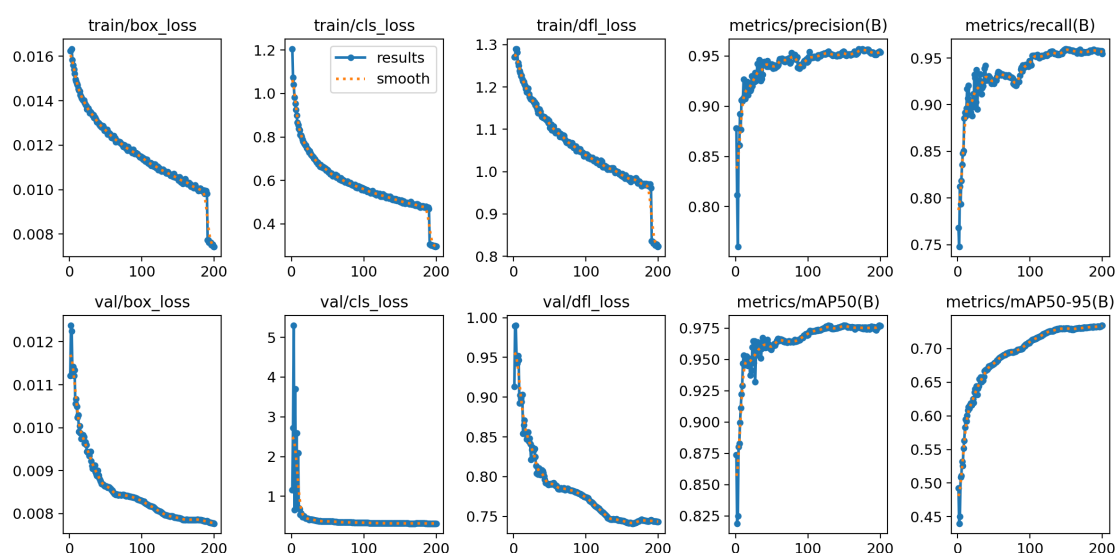
**Métricas de desempeño.** Las métricas principales muestran una mejora progresiva y una estabilización hacia las últimas épocas:

**Tabla 5.2:** Métricas de desempeño del modelo YOLOv8.

| Métrica   | Valor |
|-----------|-------|
| Precisión | 0,954 |
| Recall    | 0,954 |
| mAP@50    | 0,975 |
| mAP@50-95 | 0,734 |

**Fuente:** Elaboración propia.

Estos resultados evidencian un alto desempeño del modelo, particularmente en el umbral de IoU de 0,50, donde se alcanza un mAP@50 de 0,975. La métrica mAP@50-95, más estricta, también presenta valores elevados, lo que indica una buena capacidad de localización bajo distintos niveles de exigencia.

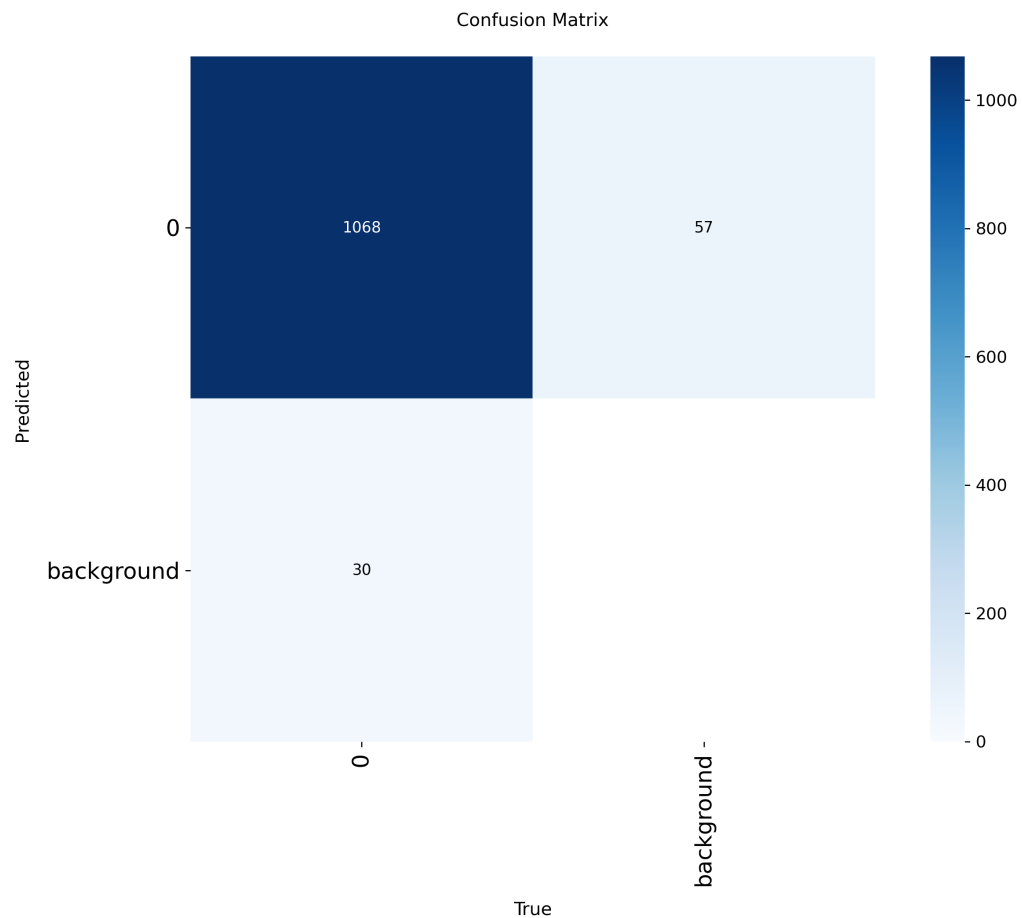


**Figura 5.4:** Curvas de pérdida y métricas de desempeño del modelo YOLOv8 durante 200 épocas de entrenamiento. **Fuente:** Elaboración propia.

Cabe destacar que las métricas presentan una evolución monótonica y una estabilización progresiva, lo cual es característico de modelos bien entrenados y con buena capacidad de generalización.

**Análisis de la matriz de confusión.** La matriz de confusión, mostrada en la Figura 5.5, permite analizar el comportamiento del modelo en términos de aciertos y errores.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**



**Figura 5.5:** Matriz de confusión del modelo de detección. **Fuente:** Elaboración propia.

A partir de esta matriz se obtienen los siguientes valores:

**Tabla 5.3:** Resultados derivados de la matriz de confusión.

| Categoría                 | Cantidad |
|---------------------------|----------|
| Verdaderos Positivos (TP) | 1068     |
| Falsos Positivos (FP)     | 57       |
| Falsos Negativos (FN)     | 30       |

**Fuente:** Elaboración propia.

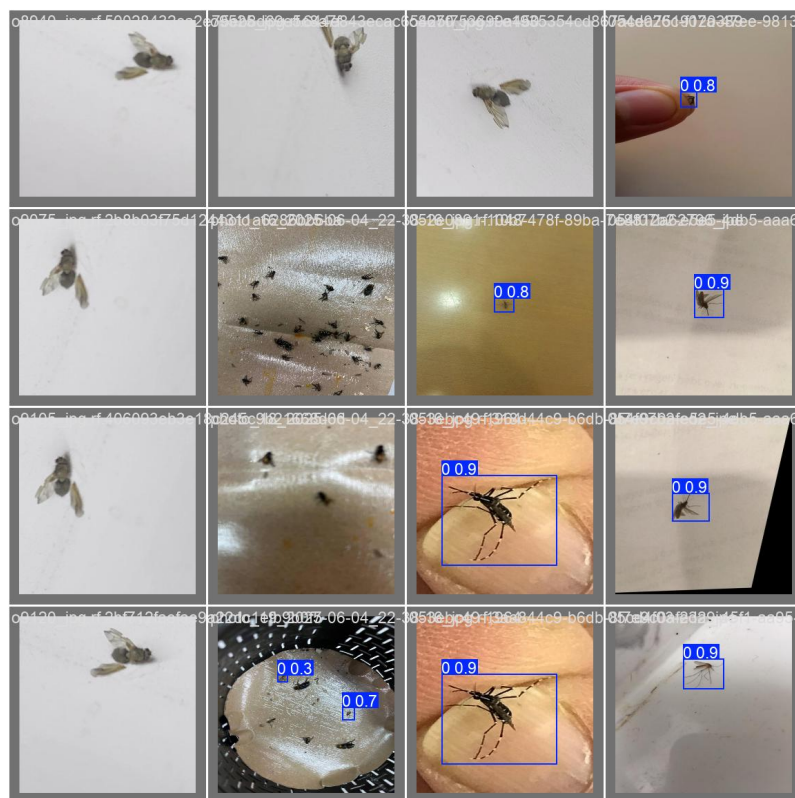
Se observa que el modelo presenta una baja tasa de falsos negativos, lo cual es especialmente relevante en aplicaciones de monitoreo, donde la omisión de detecciones puede ser crítica. Por otro lado, la tasa de falsos positivos, de aproximadamente el 5 %, es moderada y puede ajustarse mediante el umbral de

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

confianza durante la inferencia.

En conjunto, los resultados obtenidos indican que el modelo presenta una convergencia estable, sin evidencias de sobreajuste, y un desempeño elevado en la tarea de detección de mosquitos. La combinación de alta precisión y recall, junto con valores elevados de mAP, sugiere que el modelo es adecuado para su implementación en sistemas de inferencia en tiempo real.

Adicionalmente, se realizó una validación cualitativa mediante la inspección visual de las predicciones generadas sobre el conjunto de validación, como se muestra en la Figura 5.6. En esta figura se observan las cajas delimitadoras predichas por el modelo sobre imágenes no utilizadas durante el entrenamiento, permitiendo evaluar directamente su capacidad de generalización.



**Figura 5.6:** Ejemplo de predicciones del modelo YOLOv8 sobre un lote del conjunto de validación. Las cajas delimitadoras muestran las detecciones de la clase *mosquito* junto con el nivel de confianza asociado, sobre imágenes no utilizadas durante el entrenamiento. **Fuente:** Elaboración propia.

Estos resultados confirman que el modelo no solo presenta un buen desem-



peño numérico, sino también un comportamiento consistente en escenarios cercanos a condiciones reales de operación.

### 5.1.5. Exportación del modelo a ONNX

Una vez finalizado el entrenamiento del modelo en formato PyTorch (.pt), se procedió a su exportación al formato ONNX, con el objetivo de facilitar su despliegue en entornos de inferencia optimizados y dispositivos edge.

El proceso de exportación consiste en convertir el grafo computacional dinámico de PyTorch en una representación estática e interoperable, permitiendo su ejecución en distintos motores de inferencia como ONNX Runtime. Esta conversión es fundamental para desacoplar el modelo del framework de entrenamiento y habilitar optimizaciones específicas de hardware, como aceleración en CPU, GPU o dispositivos embebidos.

Durante la exportación, se definen parámetros clave como el tamaño de entrada del modelo, la versión del *opset* y la configuración del grafo. En este trabajo, se utilizó un tamaño de entrada fijo de  $256 \times 256$  píxeles, lo que permite simplificar la inferencia y mejorar la eficiencia en tiempo de ejecución.

```
import os
from ultralytics import YOLO
# CONFIG
MODEL_PATH = "best.pt"
ONNX_DIR    = "onnx_export"
IMGSZ       = 256
# LOAD MODEL
model = YOLO(MODEL_PATH)
# EXPORT TO ONNX
export_path = model.export(
    format="onnx",
    imgsz=IMGSZ,
    dynamic=False,
    simplify=False,
    opset=12
)
```

**Listing 5.1:** El proceso de exportación se implementó utilizando la librería Ultralytics.

En este proceso, se destacaron los siguientes parámetros:

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

---

- **imgsz:** define el tamaño de entrada del modelo.
- **dynamic=False:** fija las dimensiones de entrada, optimizando el rendimiento en inferencia.
- **simplify=False:** evita simplificaciones automáticas que pueden introducir incompatibilidades en ciertos entornos.
- **opset=12:** versión del estándar ONNX utilizada, compatible con la mayoría de los motores de inferencia.

**Estructura del modelo ONNX.** Una vez exportado, el modelo presenta la siguiente estructura:

**Tabla 5.4:** Información general del modelo ONNX.

| Propiedad  | Valor   |
|------------|---------|
| IR Version | 7       |
| Producer   | PyTorch |
| Opset      | 12      |

**Fuente:** Elaboración propia.

**Tabla 5.5:** Estructura de entrada del modelo ONNX.

| Entrada | Descripción             |
|---------|-------------------------|
| images  | Tensor [1, 3, 256, 256] |

**Fuente:** Elaboración propia.

**Tabla 5.6:** Estructura de salida del modelo ONNX.

| Salida  | Descripción         |
|---------|---------------------|
| output0 | Tensor [1, 5, 1344] |

**Fuente:** Elaboración propia.

La salida del modelo corresponde a un conjunto de predicciones que incluyen información de localización y confianza para cada detección, las cuales deben ser procesadas posteriormente mediante técnicas como NMS.

**Evaluación del modelo ONNX.** Para verificar la integridad del proceso de exportación, el modelo ONNX fue evaluado utilizando el mismo conjunto de datos empleado durante la validación en PyTorch. Los resultados obtenidos se presentan en la Tabla 5.7.

**Tabla 5.7:** Resultados del modelo ONNX.

| Métrica              | Valor  |
|----------------------|--------|
| Precisión            | 0,9466 |
| Recall               | 0,9390 |
| Detecciones promedio | 0,7194 |
| Desviación estándar  | 0,7542 |

**Fuente:** Elaboración propia.

Se observa que las métricas obtenidas son consistentes con las del modelo original en PyTorch, con una ligera disminución atribuible al cambio de formato y a las diferencias en la ejecución del grafo. No obstante, el desempeño se mantiene dentro de rangos aceptables para su implementación en sistemas de inferencia en tiempo real.

En conjunto, estos resultados validan la correcta conversión del modelo y su viabilidad para despliegue en entornos edge, manteniendo un equilibrio adecuado entre precisión y eficiencia computacional.

#### 5.1.6. Comparación del modelo PyTorch vs ONNX

Con el objetivo de evaluar el impacto de la exportación del modelo a formato ONNX, se realizó una comparación entre el modelo original en PyTorch y su versión optimizada ejecutada mediante ONNX Runtime. Todas las pruebas fueron realizadas utilizando el conjunto de prueba original de Roboflow, bajo las mismas condiciones experimentales.

El modelo YOLOv8 en PyTorch fue evaluado tanto en CPU como en GPU, mientras que el modelo ONNX fue ejecutado utilizando el proveedor *CPUExecu-*

tionProvider.

**Tabla 5.8:** Comparación de desempeño entre modelos YOLOv8 en PyTorch y ONNX.

| Modelo       | Tiempo (ms) | Precisión | Recall | mAP@50 |
|--------------|-------------|-----------|--------|--------|
| ONNX (CPU)   | 9,05        | 0,9466    | 0,9390 | 0,920  |
| YOLOv8 (CPU) | 20,47       | 0,9677    | 0,9574 | 0,975  |
| YOLOv8 (GPU) | 7,34        | 0,9677    | 0,9574 | 0,975  |

**Fuente:** Elaboración propia.

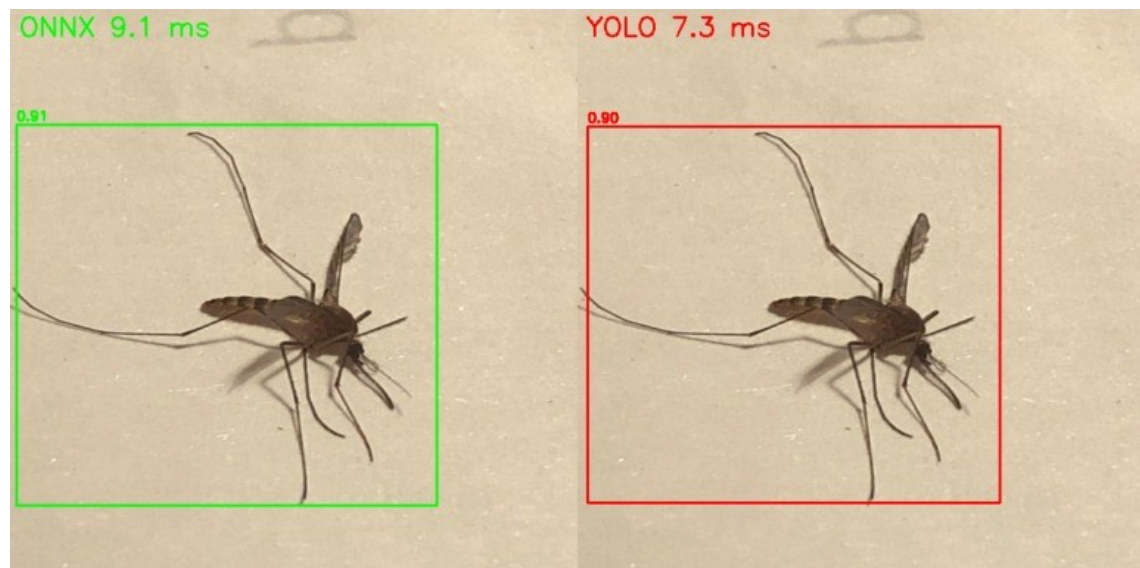
Los resultados evidencian una reducción significativa en el tiempo de inferencia al utilizar ONNX en CPU, alcanzando aproximadamente una mejora del 55 % respecto a la ejecución del modelo PyTorch en CPU. Esta optimización es especialmente relevante en escenarios de *edge computing*, donde el acceso a aceleradores como GPU puede ser limitado o inexistente.

En términos de desempeño, el modelo ONNX presenta una ligera disminución en precisión y recall respecto al modelo original en PyTorch. Sin embargo, esta diferencia es marginal y se mantiene dentro de un rango aceptable para aplicaciones prácticas, lo que confirma que la exportación no compromete significativamente la calidad de las predicciones.

Por otro lado, la ejecución en GPU del modelo PyTorch sigue siendo la opción más rápida en términos absolutos. No obstante, esta configuración no siempre es viable en dispositivos embebidos, lo que refuerza la relevancia del uso de ONNX como alternativa eficiente en CPU.

**Comparación temporal.** La Figura 5.7 presenta una comparación directa de los tiempos de inferencia por muestra entre ambos enfoques evaluados. En ella se observa que el modelo exportado a ONNX y ejecutado en CPU alcanzó un tiempo medio de 9,05 ms con una desviación estándar de 0,67 ms, mientras que el modelo YOLOv8 en PyTorch ejecutado sobre GPU registró una media de 7,34 ms y una desviación estándar de 0,99 ms.

Estos resultados evidencian que ambos sistemas presentan una latencia baja y estable, con una diferencia relativamente reducida entre la ejecución acelerada por GPU y la inferencia optimizada en CPU mediante ONNX Runtime. En términos prácticos, esto confirma que ONNX puede ofrecer un rendimiento cercano al de una GPU dedicada, incluso utilizando únicamente procesamiento por CPU.



**Figura 5.7:** Comparación de tiempos de inferencia sobre una muestra entre YOLOv8 (GPU) y el modelo exportado a ONNX (CPU). **Fuente:** Elaboración propia.

En conjunto, ONNX permite reducir significativamente la latencia en CPU manteniendo un desempeño competitivo, lo que lo convierte en una alternativa adecuada para despliegue en dispositivos edge.

## 5.2. Modelo de Clasificación (MobileNet)

La segunda etapa del sistema en cascada recibe cada recorte generado por YOLOv8 y produce la clasificación binaria *hembra/macho*. Para esta tarea se adoptó *MobileNetV3 Large*: su diseño —bloques *inverted residual* con módulos de atención *Squeeze-and-Excitation* y activaciones *h-swish*— maximiza la precisión bajo restricciones estrictas de cómputo, mientras que la variante *Large* ofrece mayor capacidad representacional que las versiones compactas sin com-

prometer su viabilidad en el dispositivo de borde. Al operar exclusivamente sobre las regiones ya detectadas, su latencia marginal dentro del pipeline es reducida, tal como confirman los tiempos medidos en la Raspberry Pi 5 presentados al final del capítulo. Las secciones siguientes describen el dataset, el proceso de entrenamiento y los resultados obtenidos en PyTorch y en ONNX.

### 5.2.1. Dataset utilizado

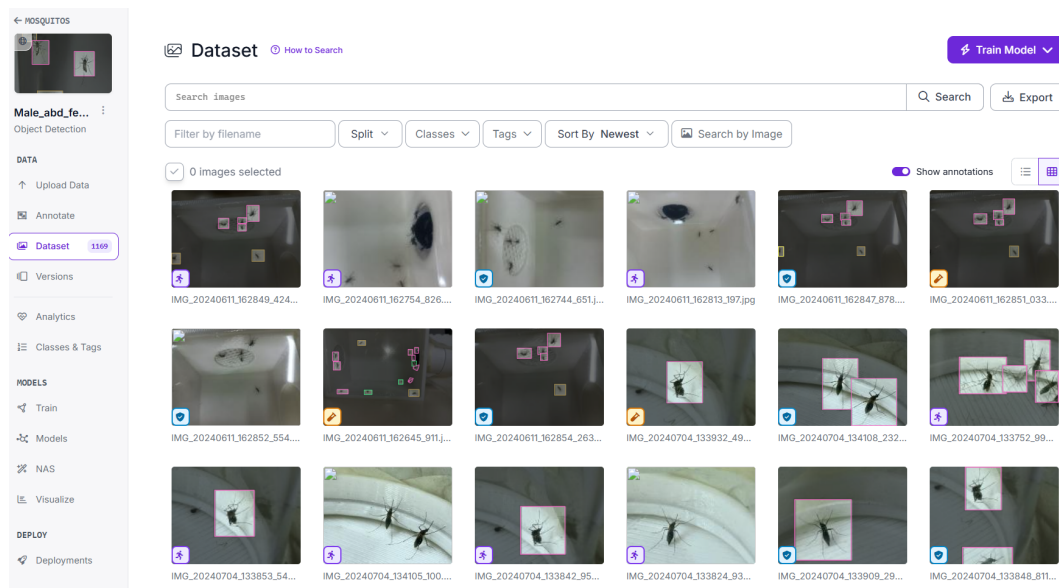
#### 5.2.1.1. Dataset original con etiquetado Roboflow

El dataset original utilizado para la tarea de clasificación fue obtenido desde la plataforma Roboflow. Está compuesto por imágenes de mosquitos capturadas y cedidas por el Instituto de Investigaciones en Ciencias de la Salud (IICS), institución con la que este trabajo colabora en el marco del mismo proyecto de investigación. Las imágenes fueron etiquetadas por integrantes del grupo en el trabajo previo de Balbuena Duarte y Barrios Pereira [38], desarrollado anteriormente dentro de la misma línea de investigación en la FIUNA.

Este conjunto de datos contiene imágenes de mosquitos clasificadas según su sexo, constituyendo dos clases: *hembra* (Clase 0) y *macho* (Clase 1). La diversidad del dataset incluye variaciones en orientación, iluminación y condiciones de captura, lo cual resulta fundamental para entrenar modelos robustos.

La Figura 5.8 muestra una vista general del dataset en la plataforma Roboflow.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**



**Figura 5.8:** Vista general del dataset de clasificación en la plataforma Roboflow, compuesto por 2.571 imágenes. **Fuente:** Elaboración propia.

Con el objetivo de mejorar la capacidad de generalización del modelo, se aplicaron técnicas de *data augmentation*, generando tres variantes por cada imagen de entrenamiento. Las transformaciones utilizadas fueron:

- Volteos horizontales y verticales
- Rotaciones de  $90^\circ$  (horario, antihorario y  $180^\circ$ )
- Rotaciones aleatorias en el rango de  $-45^\circ$  a  $+45^\circ$
- Transformaciones de corte (*shear*) de  $\pm 10^\circ$  en horizontal y vertical
- Variaciones de saturación entre  $-25\%$  y  $+25\%$
- Variaciones de brillo entre  $-15\%$  y  $+15\%$
- Ajustes de exposición entre  $-10\%$  y  $+10\%$
- Desenfoque (*blur*) de hasta 1,2 píxeles

A continuación, se presentan las estadísticas del dataset:

**Tabla 5.9:** Estadísticas del dataset original de clasificación.

| Descripción                 | Cantidad     | Porcentaje     |
|-----------------------------|--------------|----------------|
| Entrenamiento               | 2.103        | 81,8 %         |
| Validación                  | 233          | 9,1 %          |
| Prueba                      | 235          | 9,1 %          |
| <b>Total</b>                | <b>2.571</b> | <b>100,0 %</b> |
| Instancias clase 0 (hembra) | 3.343        |                |
| Instancias clase 1 (macho)  | 2.203        |                |

**Fuente:** Elaboración propia.

#### 5.2.1.2. Conversión a un dataset de clasificación

Para adaptar el dataset original al entrenamiento de un modelo de clasificación, fue necesario transformar las anotaciones en formato YOLO (cajas delimitadoras) en imágenes individuales centradas en cada mosquito. Este proceso permitió convertir un problema de detección en un conjunto de datos adecuado para clasificación supervisada.

La conversión se realizó mediante un script en Python, el cual procesa las imágenes y sus respectivas etiquetas en formato YOLO. En primer lugar, se leen las coordenadas normalizadas de cada caja delimitadora y se convierten a coordenadas en píxeles. Posteriormente, se recortan las regiones correspondientes a cada mosquito y se almacenan en una estructura de carpetas organizada por clase (*Female* y *Male*) y subconjunto (*train*, *valid*, *test*).

Adicionalmente, se implementaron mecanismos para asegurar la validez de los recortes, tales como la verificación de límites dentro de la imagen y la eliminación de regiones inválidas. También se generaron nombres únicos para cada recorte, considerando que una misma imagen puede contener múltiples instancias.

Como resultado de este proceso, se obtuvo un dataset de clasificación donde cada imagen contiene un único mosquito, eliminando la complejidad espacial y



**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

permitiendo que el modelo se enfoque exclusivamente en la tarea de discriminación entre clases.

**Tabla 5.10:** Distribución del dataset de clasificación generado.

| Conjunto      | Clase  | Cantidad | Porcentaje |
|---------------|--------|----------|------------|
| Entrenamiento | Female | 2.725    | 61,9 %     |
|               | Male   | 1.678    | 38,1 %     |
|               | Total  | 4.403    | 100,0 %    |
| Validación    | Female | 319      | 64,7 %     |
|               | Male   | 174      | 35,3 %     |
|               | Total  | 493      | 100,0 %    |
| Prueba        | Female | 227      | 50,8 %     |
|               | Male   | 220      | 49,2 %     |
|               | Total  | 447      | 100,0 %    |

**Fuente:** Elaboración propia.

Se observa un desbalance moderado en los conjuntos de entrenamiento y validación, con predominancia de la clase *Female*, mientras que el conjunto de prueba presenta una distribución más equilibrada, lo cual es adecuado para una evaluación imparcial del modelo.

**Análisis del tamaño de las imágenes.** Dado que los recortes provienen de detecciones con diferentes escalas y resoluciones, se realizó un análisis estadístico del tamaño de las imágenes resultantes:

**Tabla 5.11:** Análisis de dimensiones de las imágenes del dataset de clasificación.

| Conjunto      | Tamaño mínimo | Tamaño máximo | Tamaño promedio |
|---------------|---------------|---------------|-----------------|
| Entrenamiento | 19 × 14       | 1286 × 1276   | 491 × 510       |
| Validación    | 1 × 13        | 857 × 858     | 417 × 438       |
| Prueba        | 11 × 5        | 943 × 978     | 434 × 449       |

**Fuente:** Elaboración propia.

La variabilidad en las dimensiones refleja la diversidad de escalas en las detecciones originales. Esta heterogeneidad es beneficiosa para el entrenamiento,

ya que expone al modelo a diferentes resoluciones del objeto de interés, aunque requiere un preprocesamiento posterior (redimensionamiento) para su uso en redes neuronales convolucionales.

En conjunto, este proceso de conversión permite generar un dataset adecuado para clasificación, manteniendo la riqueza del dataset original y optimizando su estructura para el entrenamiento del modelo MobileNetV3.

### 5.2.2. Proceso de entrenamiento

El entrenamiento del modelo de clasificación se realizó utilizando la arquitectura *MobileNetV3 Large* implementada en la librería *torchvision* de PyTorch. Se partió de pesos preentrenados sobre el dataset ImageNet (*transfer learning*), lo cual permite aprovechar representaciones visuales generales previamente aprendidas y adaptarlas a la tarea específica de clasificación de mosquitos.

**Configuración del entrenamiento.** El proceso de entrenamiento fue configurado con los siguientes parámetros:

- **Tamaño de imagen:** 160 × 160 píxeles
- **Batch size:** 32
- **Tasa de aprendizaje:**  $3 \times 10^{-4}$
- **Número máximo de épocas:** 50
- **Early stopping:** 10 épocas sin mejora en validación
- **Dispositivo:** GPU (cuando está disponible)

Las imágenes fueron normalizadas utilizando los parámetros estándar de ImageNet, asegurando compatibilidad con los pesos preentrenados.

**Data augmentation y preprocesamiento.** Para mejorar la capacidad de generalización del modelo, se aplicaron transformaciones únicamente al conjunto de entrenamiento:

- Recortes aleatorios redimensionados (*RandomResizedCrop*) en el rango [85 %, 100 %]
- Volteo horizontal aleatorio
- Variaciones de color (*ColorJitter*) en brillo, contraste, saturación y tono

Para los conjuntos de validación y prueba, se utilizó únicamente un redimensionamiento fijo seguido de normalización, evitando introducir variabilidad adicional durante la evaluación.

**Manejo del desbalance de clases.** Dado el desbalance presente en el dataset, se implementó un esquema de ponderación en la función de pérdida. Los pesos de cada clase fueron calculados inversamente proporcionales a su frecuencia en el conjunto de entrenamiento, permitiendo penalizar más fuertemente los errores en la clase minoritaria.

**Función de pérdida y optimización.** Se utilizó la función de pérdida *CrossEntropyLoss* con dos modificaciones relevantes:

- **Pesos de clase:** para compensar el desbalance
- **Label smoothing** ( $\alpha = 0,1$ ): para mejorar la generalización y reducir la sobreconfianza del modelo

El optimizador seleccionado fue *AdamW*, con un factor de regularización (*weight decay*) de  $10^{-4}$ , lo que contribuye a evitar el sobreajuste.

Adicionalmente, se empleó un scheduler de tipo *Cosine Annealing*, que ajusta la tasa de aprendizaje de forma suave a lo largo de las épocas, favoreciendo una convergencia más estable.

**Estrategia de entrenamiento.** El entrenamiento se llevó a cabo mediante un bucle iterativo que alterna entre fases de entrenamiento y validación en cada época. En cada iteración:

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

---

1. Se actualizan los pesos del modelo utilizando el conjunto de entrenamiento.
2. Se evalúa el modelo sobre el conjunto de validación.
3. Se ajusta la tasa de aprendizaje mediante el scheduler.

Se utilizó como criterio de selección del mejor modelo la **precisión** sobre el conjunto de validación. Cada vez que se obtenía una mejora en esta métrica, los pesos del modelo eran almacenados.

Además, se implementó un mecanismo de *early stopping*, que detiene el entrenamiento si no se observa mejora en la precisión de validación durante un número consecutivo de épocas (patience), evitando así sobreentrenamiento y reduciendo el costo computacional.

### 5.2.3. Resultados y métricas del modelo PyTorch

El desempeño del modelo de clasificación basado en MobileNetV3 Large fue evaluado utilizando el conjunto de prueba, obteniendo una **precisión** global de 0,9731, lo que indica un alto nivel de acierto en la discriminación entre mosquitos macho y hembra.

**Tabla 5.12:** Reporte de clasificación del modelo MobileNetV3.

| Clase  | Precisión | Recall | F1-score | Soporte |
|--------|-----------|--------|----------|---------|
| Female | 0,97      | 0,98   | 0,97     | 227     |
| Male   | 0,98      | 0,96   | 0,97     | 220     |

**Fuente:** Elaboración propia.

**Tabla 5.13:** Métricas globales del modelo de clasificación.

| Métrica                 | Valor  |
|-------------------------|--------|
| Precisión               | 0,9731 |
| Macro Avg (F1-score)    | 0,97   |
| Weighted Avg (F1-score) | 0,97   |

**Fuente:** Elaboración propia.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

Los resultados muestran un desempeño equilibrado entre ambas clases, con valores de precisión y recall superiores al 96 %, lo que indica que el modelo es capaz de discriminar eficazmente entre mosquitos macho y hembra sin sesgos significativos.

**Tabla 5.14:** Matriz de confusión del modelo de clasificación.

|             | Predicción |      |
|-------------|------------|------|
|             | Female     | Male |
| Real Female | 223        | 4    |
| Real Male   | 8          | 211  |

**Fuente:** Elaboración propia.

A partir de la matriz de confusión, se observa que el modelo presenta una baja cantidad de errores en ambas clases. En particular, se registran únicamente 4 falsos negativos para la clase *Female* y 8 para la clase *Male*, lo cual confirma la robustez del modelo.

En conjunto, estos resultados evidencian que el modelo MobileNetV3 alcanza un alto nivel de precisión y generalización, siendo adecuado para su integración en el sistema en cascada propuesto.

#### 5.2.4. Exportación del modelo a ONNX

Siguiendo el mismo esquema que el detector, el modelo `best_model.pth` fue exportado a ONNX. A diferencia de YOLOv8, aquí se habilitaron ejes dinámicos en el *batch* para mayor flexibilidad, y se empleó *opset* 13 junto con `do_constant_folding=True` para optimizar el grafo antes de la inferencia.

```
import torch
import torchvision.models as models
# CONFIG
MODEL_PATH = "best_model.pth"
ONNX_PATH = "best_CLmodel.onnx"
IMAGE_SIZE = 160
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
NUM_CLASSES = 2
```

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

```
# LOAD MODEL (MATCH TRAINING!)
def build_model():
    model = models.mobilenet_v3_large(weights=None)
    in_features = model.classifier[3].in_features
    model.classifier[3] = torch.nn.Linear(in_features,
        NUM_CLASSES)
    return model
model = build_model()
# Load weights
state_dict = torch.load(MODEL_PATH, map_location=DEVICE)
model.load_state_dict(state_dict)
model.to(DEVICE)
model.eval()
# DUMMY INPUT
dummy_input = torch.randn(1, 3, IMAGE_SIZE, IMAGE_SIZE,
    device=DEVICE)
# EXPORT ONNX
torch.onnx.export(
    model,
    dummy_input,
    ONNX_PATH,
    export_params=True,
    opset_version=13,
    do_constant_folding=True,
    input_names=["input"],
    output_names=["logits"],
    dynamic_axes={
        "input": {0: "batch_size"},
        "logits": {0: "batch_size"}
    },
    dynamo=False
)
```

**Listing 5.2:** Código de exportación del modelo a ONNX.

**Descripción del proceso.** El procedimiento de exportación consiste en reconstruir la arquitectura del modelo exactamente como fue definida durante el entrenamiento, y posteriormente cargar los pesos aprendidos desde el archivo .pth. Es importante destacar que el modelo se inicializa sin pesos preentrenados (`weights=None`), ya que estos son reemplazados por los parámetros previamente entrenados.

Para realizar la exportación, se define una entrada ficticia (*dummy input*) con dimensiones `[1, 3, 160, 160]`, que representa el formato esperado por el modelo. Esta entrada permite a PyTorch trazar el grafo computacional y generar su representación estática en formato ONNX.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

El uso de `dynamic_axes` permite que el modelo soporte tamaños de *batch* variables durante la inferencia, lo cual aporta flexibilidad en su despliegue. Además, la opción `do_constant_folding=True` optimiza el grafo eliminando operaciones redundantes, mejorando el rendimiento en tiempo de ejecución.

**Estructura del modelo ONNX.** Una vez exportado, el modelo presenta la siguiente estructura:

**Tabla 5.15:** Información general del modelo ONNX de clasificación.

| Propiedad  | Valor   |
|------------|---------|
| IR Version | 7       |
| Producer   | PyTorch |
| Opset      | 13      |

**Fuente:** Elaboración propia.

**Tabla 5.16:** Estructura de entrada del modelo ONNX de clasificación.

| Entrada | Descripción                               |
|---------|-------------------------------------------|
| input   | Tensor [ <i>batch_size</i> , 3, 160, 160] |

**Fuente:** Elaboración propia.

**Tabla 5.17:** Estructura de salida del modelo ONNX de clasificación.

| Salida | Descripción                     |
|--------|---------------------------------|
| logits | Tensor [ <i>batch_size</i> , 2] |

**Fuente:** Elaboración propia.

La salida del modelo corresponde a un vector de dos valores (*logits*), los cuales representan las puntuaciones sin normalizar para cada clase (*Female* y *Male*). Para obtener probabilidades, es necesario aplicar una función *softmax* en la etapa de inferencia.

En conjunto, la exportación a ONNX permite desacoplar el modelo del framework de entrenamiento, facilitando su integración en sistemas de inferencia eficientes y portables, especialmente en aplicaciones sobre dispositivos edge.

### 5.2.5. Comparación del modelo PyTorch vs ONNX

Con el objetivo de evaluar el impacto de la exportación al formato ONNX, se comparó el modelo original en PyTorch (.pth) con su versión exportada sobre el conjunto de prueba de 447 muestras, bajo las mismas condiciones de hardware descritas en la Sección 5.1.

**Tabla 5.18:** Comparación de desempeño entre los modelos MobileNetV3 en PyTorch y ONNX ejecutados en CPU y GPU.

| Modelo        | Tiempo (ms) | Precisión | Recall | F1-score |
|---------------|-------------|-----------|--------|----------|
| ONNX (CPU)    | 5,83        | 0,9731    | 0,9814 | 0,9724   |
| PyTorch (CPU) | 20,83       | 0,9731    | 0,9814 | 0,9724   |
| PyTorch (GPU) | 11,26       | 0,9731    | 0,9814 | 0,9724   |

**Fuente:** Elaboración propia.

Los resultados demuestran que la exportación a ONNX preserva completamente el comportamiento del modelo original: no se registró ningún desacuerdo en las 447 predicciones evaluadas, y las métricas de clasificación son idénticas en ambos formatos. A diferencia del detector YOLOv8, donde la conversión introdujo una ligera caída en precisión y recall, el clasificador MobileNetV3 no exhibe degradación alguna, lo que refleja la mayor simplicidad de su grafo computacional y la ausencia de operaciones sensibles a la conversión de formato.

**Consistencia numérica.** En primer lugar, se verificó la equivalencia numérica entre ambos modelos mediante la comparación directa de sus salidas (*logits*) ante una misma entrada:

- PyTorch: [1,0988, -1,1573]
- ONNX: [1,0999, -1,1551]
- Diferencia media: 0,00165
- Diferencia máxima: 0,00220



Las diferencias observadas son mínimas y atribuibles exclusivamente a variaciones en la representación de punto flotante entre motores de ejecución, confirmando la correcta conversión del modelo.

**Comparación de tiempos de inferencia.** Como se aprecia en la Tabla 5.18, el modelo ONNX ejecutado en CPU alcanzó un tiempo medio de inferencia de 5,83 ms, superando ampliamente al modelo PyTorch ejecutado en CPU, que registró 20,83 ms. Esto representa una reducción aproximada del 72 % en el tiempo de inferencia, atribuible a las optimizaciones del grafo computacional aplicadas durante la exportación al formato ONNX.

En comparación con la ejecución acelerada por GPU, el modelo PyTorch obtuvo un tiempo medio de 11,26 ms, valor que continúa siendo superior al alcanzado por el modelo ONNX ejecutado únicamente en CPU. Este resultado evidencia que la versión exportada no solo conserva el desempeño del modelo original, sino que además logra una ejecución más eficiente incluso sin requerir aceleración por hardware dedicado.

La Figura 5.9 ilustra la distribución de tiempos de inferencia sobre una muestra del conjunto, comparando el modelo PyTorch en GPU frente al modelo ONNX en CPU, donde se aprecia con mayor detalle la estabilidad de cada motor de ejecución. En dicha muestra, el modelo ONNX en CPU registra una media de 2,5 ms con una desviación estándar de apenas 0,06 ms, mientras que el modelo PyTorch en GPU alcanza una media de 10,5 ms y una desviación estándar de 0,45 ms. La menor dispersión observada en el modelo ONNX evidencia una latencia más predecible y consistente durante la inferencia.

GT: Male  
PTH: Male (10.5 ms)  
ONNX: Male (2.5 ms)



**Figura 5.9:** Comparación de tiempos de inferencia entre el modelo PyTorch (ejecutado en GPU) y ONNX (ejecutado en CPU). **Fuente:** Elaboración propia.

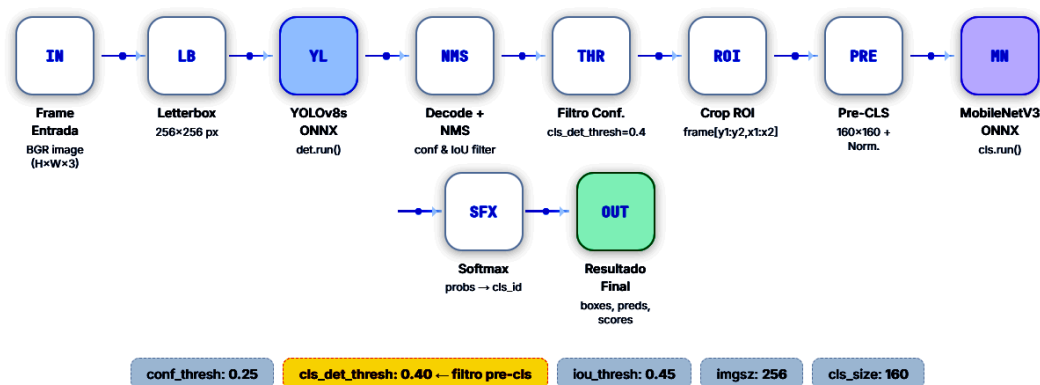
En conjunto, el modelo ONNX no solo mantiene íntegramente el desempeño del original en PyTorch, sino que además ofrece una mejora sustancial en velocidad y estabilidad temporal, consolidando su idoneidad para el despliegue como segunda etapa del pipeline en dispositivos edge.

### 5.3. Pipeline del modelo en cascada

Las Secciones 5.1 y 5.2 desarrollaron cada componente de forma aislada; esta sección describe cómo se integran en un pipeline unificado, detalla el flujo de inferencia extremo a extremo y evalúa el desempeño del sistema completo sobre el conjunto de prueba.

La Figura 5.10 ilustra el flujo de procesamiento del pipeline implementado.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS



**Figura 5.10:** Diagrama de bloques del pipeline en cascada: la imagen de entrada es procesada primero por YOLOv8 (detección), y cada región detectada pasa al clasificador MobileNetV3 (clasificación por sexo). **Fuente:** Elaboración propia.

### 5.3.1. Flujo de inferencia

El pipeline sigue una secuencia estructurada de etapas:

1. **Entrada y preprocesamiento:** La imagen de entrada es redimensionada mediante *letterbox* a un tamaño fijo de  $256 \times 256$ , preservando la relación de aspecto y agregando padding.
2. **Detección de objetos (YOLOv8 ONNX):** La imagen preprocesada es inferida por el modelo YOLOv8, generando predicciones de cajas delimitadoras junto con sus respectivas confianzas.
3. **Decodificación y filtrado:** Las predicciones son decodificadas y filtradas mediante un umbral de confianza ( $conf\_thresh = 0,125$ ), seguido de un proceso de NMS con un umbral IoU de 0,45 para eliminar detecciones redundantes.
4. **Extracción del Region of Interest (ROI):** A partir de las cajas detectadas, se recortan las regiones correspondientes a cada mosquito en la imagen original.

5. **Preprocesamiento para clasificación:** Cada recorte es redimensionado a  $160 \times 160$  píxeles, normalizado con parámetros de ImageNet y transformado al formato requerido por el modelo de clasificación.
6. **Clasificación (MobileNetV3 ONNX):** Cada región es inferida por el modelo de clasificación, obteniendo como salida un vector de probabilidades (*logits*) para las clases *Female* y *Male*.
7. **Postprocesamiento:** Se aplica una función *argmax* sobre los logits para determinar la clase final de cada detección.
8. **Visualización (para la depuración):** Finalmente, se dibujan las cajas delimitadoras sobre la imagen original, junto con la etiqueta de clase predicha.

El enfoque modular permitió optimizar cada componente de forma independiente y facilita la integración en sistemas embebidos.

### 5.3.2. Contexto y metodología de evaluación

Una consideración fundamental al interpretar los resultados del sistema en cascada es el origen del conjunto de prueba utilizado. Las 447 instancias (anotaciones individuales de mosquito) provienen del **subconjunto de prueba del dataset original usado para la clasificación pero antes del recorte y la extracción de instancias**, Tabla 5.9. Estas imágenes no son recortes pre-generados, sino imágenes completas con sus anotaciones de caja delimitadora en formato YOLO; el pipeline recorre cada una de ellas de principio a fin —detección mediante YOLOv8 seguida de clasificación mediante MobileNetV3— y las predicciones se evalúan contra las cajas de referencia Ground truth (GT). Dado que el modelo de detección (YOLOv8) **nunca fue entrenado ni validado con imágenes de esta distribución** —cuyas características de escala, iluminación y encuadre difieren de su conjunto de entrenamiento nativo—, su evaluación opera en un

régimen de dominio no visto, lo que puede explicar parte de la caída observada en el recall de detección.

La evaluación se realizó mediante un script en Python que ejecuta el pipeline completo sobre cada imagen del conjunto de prueba. Para cada imagen, el detector genera cajas predichas que son cotejadas con las etiquetas GT mediante una comparación de solapamiento geométrico (IoU). Una predicción se considera detección válida si su IoU con alguna caja de referencia no emparejada supera el umbral  $\text{IoU}_{\text{match}} = 0,50$ . Sobre cada detección válida se ejecuta el clasificador, y la predicción de clase se compara con la etiqueta de la instancia asociada. Las métricas globales se derivan de los conteos acumulados:

$$\text{Recall}_{\text{det}} = \frac{\text{Det. válidas}}{\text{GT total}}$$

$$\text{Prec}_{\text{sistema}} = \frac{\text{Clasif. correctas}}{\text{GT total}}$$

$$\text{Prec}_{\text{Cls|Det}} = \frac{\text{Clasif. correctas}}{\text{Det. válidas}}$$

### 5.3.3. Evaluación del sistema en cascada

El desempeño del sistema completo fue evaluado utilizando el conjunto de prueba bajo las mismas condiciones de hardware (Intel Core i7 de 14<sup>a</sup> generación y GPU NVIDIA RTX 5060, aunque la inferencia ONNX se ejecuta íntegramente en CPU).

**Tabla 5.19:** Resultados del modelo en cascada sobre el conjunto de prueba del clasificador.

| Métrica                                          | Valor  |
|--------------------------------------------------|--------|
| Total de instancias (GT)                         | 447    |
| Detecciones válidas ( $\text{IoU} \geq 0,50$ )   | 339    |
| Clasificaciones correctas                        | 321    |
| Recall de detección                              | 0,7584 |
| Precisión total del sistema                      | 0,7181 |
| Precisión de clasificación condicional (Cls Det) | 0,9469 |

**Fuente:** Elaboración propia.

Las implicaciones de esta asimetría de dominio se discuten en la Sección 5.3.4.

#### 5.3.4. Análisis de resultados

El comportamiento del sistema revela una asimetría que es inherente a la metodología de evaluación adoptada: ambos modelos fueron evaluados sobre el **mismo conjunto de prueba** —el conjunto de prueba nativo del clasificador, compuesto por recortes individuales de mosquito etiquetados por sexo—, pero el detector YOLOv8 **nunca fue entrenado ni validado con imágenes de esta distribución**, lo que sitúa su evaluación en un régimen de dominio no visto. El clasificador MobileNetV3, en cambio, es evaluado directamente sobre su propio conjunto de prueba, lo que hace que su métrica aislada (97,31 %) constituya el punto de referencia natural para interpretar los resultados del sistema.

El **recall de detección del 75,84 %** contrasta con el 93,9 % obtenido por el mismo detector sobre su propio conjunto de prueba (Tabla 5.7). Esta diferencia de  $\approx 18$  puntos porcentuales no debe interpretarse como una limitación estructural del modelo, sino como una consecuencia esperada de la evaluación en dominio distinto, atribuible a tres factores concretos: (i) el conjunto de prueba del clasificador está compuesto por recortes individuales de mosquito —imágenes de una sola instancia centrada—, que difieren de las imágenes multiobjeto con

las que el detector fue entrenado y validado; *(ii)* el redimensionado a  $256 \times 256$  píxeles puede introducir artefactos o pérdida de detalle en recortes de baja resolución provenientes de dicho conjunto; y *(iii)* al no haber sido expuesto durante el entrenamiento a esta distribución de imágenes, el detector carece de la especialización necesaria para maximizar su recall sobre la misma.

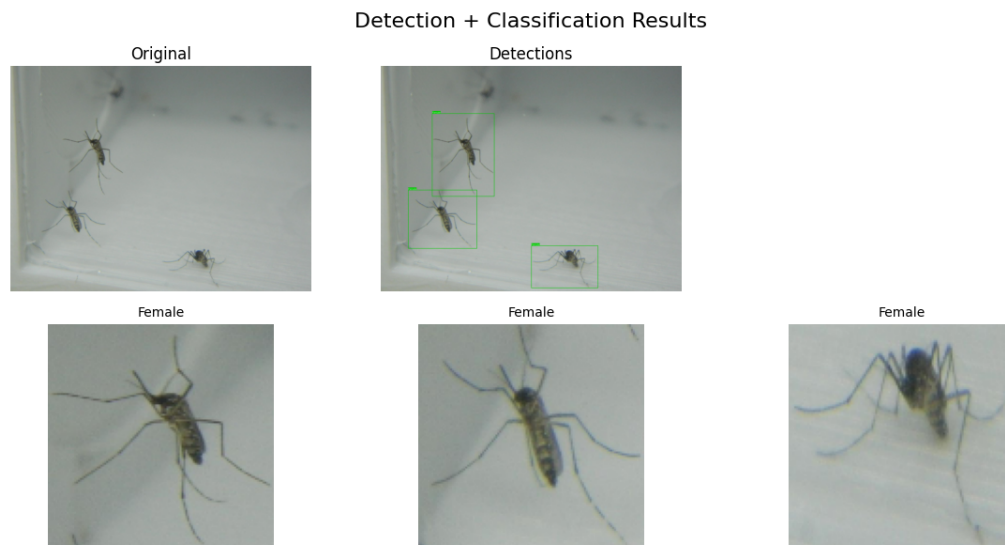
En este contexto, la **precisión de clasificación condicional del 94,69 %** (321 de 339 detecciones correctamente clasificadas) confirma que el clasificador MobileNetV3 opera de forma robusta incluso sobre recortes generados automáticamente por el detector, manteniendo un desempeño muy próximo al obtenido en su evaluación aislada (97,31 % sobre recortes perfectos). La diferencia de  $\approx 2,6$  puntos porcentuales refleja únicamente el impacto de imperfecciones geométricas en los recortes producidos por YOLO —bordes desalineados, padding asimétrico— y no una degradación intrínseca del modelo de clasificación.

La **precisión global del sistema es del 71,81 %**; este resultado evidencia que el principal margen de mejora del sistema en cascada reside en ampliar el conjunto de entrenamiento del detector con imágenes de la misma distribución que el conjunto de prueba del clasificador, lo que reduciría la brecha de dominio identificada y elevaría el recall de detección sin necesidad de modificar la arquitectura del sistema.

**Ejemplo de inferencia.** La Figura 5.11 muestra un ejemplo del resultado final del sistema, donde se observan las detecciones realizadas junto con la clasificación correspondiente de cada mosquito.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---



**Figura 5.11:** Ejemplo de salida del modelo en cascada con etiquetas de clase (*Female/Male*) y nivel de confianza asociado. **Fuente:** Elaboración propia.

En conjunto, el pipeline propuesto constituye una solución efectiva para la detección y clasificación de mosquitos en imágenes: la robustez del clasificador condicionado (94,69 % de precisión condicional) garantiza que las instancias detectadas sean correctamente etiquetadas, y el principal margen de mejora del sistema reside en ampliar el dataset del detector con imágenes de una sola instancia, lo que reduciría la brecha de dominio identificada sin necesidad de modificar la arquitectura. La modularidad de la arquitectura en cascada facilita, además, la incorporación futura de nuevas etapas —como un discriminador de especie— con un costo de entrenamiento e inferencia proporcionalmente reducido. La viabilidad de esta solución en el hardware de despliegue se completa con los tiempos de inferencia medidos directamente sobre el dispositivo de borde, presentados a continuación.

### 5.3.5. Tiempos de inferencia en el dispositivo de borde

Con el objetivo de verificar la viabilidad del sistema en el hardware de despliegue final, se midieron los tiempos de inferencia de cada componente del pipeline



**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

ejecutando los modelos ONNX directamente sobre la Raspberry Pi 5, en modo CPU. Para cada modelo se registró el tiempo promedio de inferencia sobre una muestra representativa de imágenes del conjunto de prueba.

**Tabla 5.20:** Tiempos de inferencia de los modelos ONNX medidos en la Raspberry Pi 5 (CPU).

| <b>Modelo / Etapa</b>             | <b>Tiempo de inferencia (ms)</b> | <b>FPS</b>  |
|-----------------------------------|----------------------------------|-------------|
| Detector YOLOv8 (ONNX)            | 101,11                           | 9,89        |
| Clasificador MobileNetV3 (ONNX)   | 14,77                            | 67,70       |
| <b>Sistema en cascada (total)</b> | <b>133,53</b>                    | <b>7,49</b> |

**Fuente:** Elaboración propia.

Los resultados muestran que el detector YOLOv8 concentra la mayor parte de la latencia del pipeline (101,11 ms), mientras que el clasificador MobileNetV3 añade únicamente 14,77 ms, confirmando su carácter ligero como segunda etapa. Los 17,65 ms restantes corresponden al pre y postprocesamiento intermedio: redimensionado *letterbox* de la imagen de entrada, decodificación de predicciones, NMS, extracción de recortes (ROI) y normalización de cada región antes de la clasificación. El tiempo total del sistema en cascada (133,53 ms) equivale a una frecuencia de inferencia de aproximadamente 7,5 Frames Per Second (FPS), valor compatible con la tasa de captura de la trampa y suficiente para garantizar la operación en tiempo real sobre la Raspberry Pi 5.

Con esto quedan validados los dos componentes que conforman la capa de inferencia local del sistema: un detector capaz de localizar especímenes en condiciones variables y un clasificador que discrimina el sexo con una precisión condicional del 94,69 % sobre las detecciones generadas automáticamente. Ambos modelos, exportados a ONNX y ejecutados íntegramente en CPU, constituyen el núcleo de adquisición inteligente de datos que el Capítulo 6 integra dentro de la arquitectura IoT de la trampa, articulando la inferencia local con la transmisión, persistencia y visualización de los resultados en la nube.

## CAPÍTULO 6

### Arquitectura IoT Implementada

Este Capítulo describe la arquitectura IoT de la trampa inteligente desarrollada en el marco de la fase FDT3 del proyecto. La solución integra tres capas diferenciadas: el dispositivo de borde, encargado de la adquisición de datos y la inferencia local sobre los modelos descritos en el Capítulo 5; el servidor backend, que gestiona la persistencia y exposición de datos mediante una API Representational State Transfer (REST); y la capa de presentación, implementada como una aplicación web interactiva. Se describen además los protocolos de comunicación que articulan estas capas —comunicación interna entre hilos, enlace serie con el microcontrolador y transmisión Hypertext Transfer Protocol (HTTP) al servidor—, los servicios en la nube utilizados para el despliegue, y las consideraciones de seguridad pertinentes para un sistema de estas características. Los resultados operativos del sistema en campo se presentan en el Capítulo 9.

#### 6.1. Arquitectura Cliente-Servidor

El sistema se organiza en tres capas claramente diferenciadas:

1. **Capa de dispositivo de borde (edge).** Encargada de la adquisición de datos mediante una Raspberry Pi equipada con una cámara y un sensor BME280, capaz de capturar imágenes y variables ambientales (temperatura y humedad). Estos datos son enviados al servidor mediante peticiones

HTTP POST.

2. **Capa de servidor (backend).** Implementada mediante FastAPI, esta capa recibe, procesa y almacena los datos. Gestiona la persistencia en base de datos, el almacenamiento de imágenes y la ejecución de modelos de inteligencia artificial cuando es necesario (como servicio de inferencia accesible desde la web, independiente de la inferencia local en el dispositivo de borde).
3. **Capa de presentación (frontend).** Implementada como una aplicación web estática, consume la API REST del backend y presenta la información en forma de dashboard interactivo, además de proporcionar herramientas adicionales como visualización de modelos y simulaciones, así como una interfaz de navegación y documentación del proyecto.

El desacoplamiento entre frontend y backend se gestiona mediante un archivo de configuración (`config.js`) que define dinámicamente la URL del servidor. Además, el uso de `CORSMiddleware` permite la comunicación controlada entre dominios.

Este diseño permite modificar o escalar cada capa de forma independiente, lo cual resulta crítico en sistemas IoT distribuidos.

## 6.2. Capa de Dispositivo de Borde

El dispositivo de borde está implementado sobre una Raspberry Pi y constituye el núcleo de adquisición y procesamiento en tiempo real del sistema. A diferencia de un nodo IoT convencional, este dispositivo no solo captura datos, sino que integra un pipeline de inferencia optimizado y una lógica de control reactiva para la actuación física del sistema.

El hardware del dispositivo de borde está compuesto por una Raspberry Pi 5,

Raspberry Pi Camera Module 3 NoIR y un Arduino UNO como se presentó en el Capítulo 4.

### 6.2.1. Pipeline de procesamiento en tiempo real

El dispositivo ejecuta un pipeline concurrente multihilo que separa la adquisición de imágenes, la inferencia del modelo y la transmisión de datos en módulos independientes, permitiendo mantener la tasa de captura sin bloqueos. La implementación detallada —incluyendo los mecanismos de comunicación entre módulos, el control dinámico de la frecuencia de inferencia y la estrategia de descarte de frames— se describe en la Sección 6.4.

### 6.2.2. Máquina de estados de la trampa

La coordinación entre la detección y la actuación física se implementa mediante una máquina de estados finita con dos estados principales: `IDLE` (monitoreo continuo) y `SORTING` (actuación activa). Los parámetros de transición, umbrales de confianza y mecanismos de bloqueo temporal se detallan en la Sección 6.4.

### 6.2.3. Consideraciones de diseño

El diseño del dispositivo de borde prioriza:

- **Baja latencia:** mediante procesamiento local y ejecución asíncrona.
- **Robustez:** evitando bloqueos mediante separación de hilos (cámara, inferencia, red).
- **Control determinista:** mediante máquina de estados para interacción con hardware.

- **Escalabilidad:** permitiendo migrar la inferencia entre edge y cloud sin modificar la arquitectura.

Este enfoque convierte al dispositivo en un nodo inteligente capaz de operar de forma autónoma, incluso en condiciones de conectividad limitada.

### 6.3. Capa de Servidor (Backend)

El backend del sistema está implementado en Python utilizando el framework FastAPI, ejecutado sobre el servidor Asynchronous Server Gateway Interface (ASGI) **Uvicorn**. Su función principal es actuar como núcleo de procesamiento y orquestación del sistema IoT, gestionando la persistencia de datos y la exposición de servicios mediante una API REST.

#### 6.3.1. Gestión de configuración y servicios externos

El backend utiliza variables de entorno para desacoplar la configuración del código fuente. Entre los componentes principales se encuentran:

- **Base de datos relacional (PostgreSQL):** gestionada mediante SQLAlchemy para almacenar metadatos estructurados de las detecciones. Los servicios específicos empleados se describen en la Sección 6.6.
- **Almacenamiento de objetos (bucket):** utilizado para persistir las imágenes capturadas por los dispositivos IoT.
- **Middleware CORS:** configurado para permitir comunicación entre frontend y backend en distintos dominios.

Este enfoque permite desplegar el sistema en distintos entornos (local, staging, producción) sin modificar el código.

### 6.3.2. Estructura de la API REST

El servidor expone múltiples endpoints que cubren diferentes funcionalidades del sistema:

**Tabla 6.1:** Endpoints de la API REST del sistema.

| Método | Ruta                | Descripción                                  |
|--------|---------------------|----------------------------------------------|
| GET    | /api/health         | Verificación del estado del sistema.         |
| POST   | /upload             | Ingestión de datos desde el dispositivo IoT. |
| GET    | /api/dashboard-data | Consulta de detecciones recientes.           |

**Fuente:** Elaboración propia.

Estos tres endpoints constituyen el núcleo de la gestión IoT del sistema: el endpoint `/upload` se encarga de la recepción e ingesta de datos provenientes del dispositivo de borde, mientras que `/api/dashboard-data` expone las detecciones persistidas para su visualización, y `/api/health` permite verificar la disponibilidad del servicio. En conjunto, cubren el ciclo completo de subida y persistencia de datos en el sistema.

Esta API sigue un diseño RESTful, facilitando su integración con el frontend y otros posibles clientes.

### 6.3.3. Ingesta de datos IoT

El endpoint `/upload` gestiona la recepción de datos provenientes de los dispositivos de borde. El proceso consiste en:

1. Recepción de imagen y metadatos mediante *multipart/form-data*.
2. Generación de un identificador único basado en timestamp.
3. Subida de la imagen a Supabase Storage mediante una petición HTTP.
4. Inserción de los metadatos en la base de datos PostgreSQL.

Los datos almacenados incluyen:

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

- Identificador del dispositivo (`client_id`)
- Timestamp de captura
- Detecciones (serializadas)
- Temperatura y humedad
- URL pública de la imagen

Este diseño separa almacenamiento de datos estructurados y no estructurados, optimizando escalabilidad y acceso.

#### **6.3.4. Diseño concurrente y escalabilidad**

El sistema incorpora múltiples estrategias para garantizar rendimiento y escalabilidad:

- Uso de **hilos** para tareas pesadas (carga de modelos).
- Uso de **async/await** para operaciones I/O intensivas.
- Separación de responsabilidades entre almacenamiento y otras tareas.

Este enfoque permite manejar múltiples solicitudes concurrentes, manteniendo baja latencia en la respuesta.

En conjunto, la capa de backend actúa como el núcleo inteligente del sistema, centralizando la gestión de datos y los servicios de API en una arquitectura modular y escalable.

#### **6.4. Protocolos de Comunicación del Cliente de Borde**

El cliente de borde implementa un esquema de comunicación multinivel que integra procesamiento concurrente local, interacción física y transmisión de datos hacia la nube. Este esquema se organiza en tres niveles:

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

- Comunicación interna entre módulos mediante hilos y colas.
- Comunicación serie (UART) con el microcontrolador Arduino.
- Comunicación HTTP con el servidor backend.

La Figura 6.1 resume el flujo completo de un evento de detección. El sistema ejecuta en paralelo cuatro módulos: captura de imágenes, inferencia, comunicación HTTP y gestión de comunicación serial. Esta arquitectura distribuida permite mantener un flujo continuo sin bloqueos entre adquisición, procesamiento y transmisión.

La cámara captura frames de forma continua, mientras que la frecuencia de inferencia se regula dinámicamente mediante el parámetro *SKIP*, calculado a partir de la tasa real de captura, evitando la saturación del modelo.

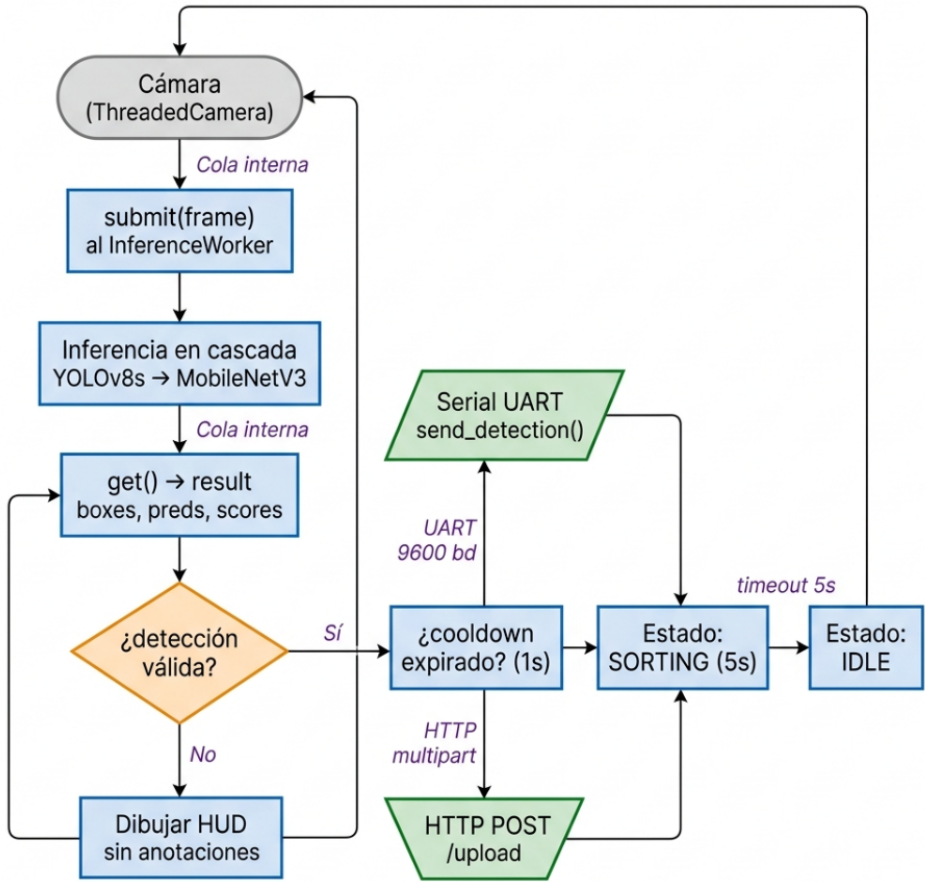
Los frames seleccionados son procesados mediante un pipeline en cascada (detección + clasificación). Si el módulo de inferencia se encuentra ocupado, los nuevos frames se descartan (*drop-if-busy*, descarte de frames cuando el módulo está ocupado), priorizando baja latencia sobre la conservación de todos los frames capturados.

Cuando una detección supera los umbrales de confianza y se cumple el intervalo mínimo entre eventos (*cooldown*), se ejecutan en paralelo:

- Envío de un comando serial al Arduino para activar el mecanismo físico.
- Envío de la imagen y metadatos al servidor mediante HTTP.

Posteriormente, el sistema entra en un intervalo temporal de bloqueo, evitando múltiples activaciones consecutivas y garantizando estabilidad en la actuación mecánica.





**Figura 6.1:** Diagrama de flujo del protocolo de comunicación IoT. **Fuente:** Elaboración propia.

6.4.1. Nivel 1 — Comunicación Interna

El pipeline de visión se implementa mediante múltiples hilos que se comunican a través de colas y métodos no bloqueantes:

**Tabla 6.2:** Módulos concurrentes y mecanismos de comunicación interna.

| Módulo     | Clase           | Rol               | Mecanismo        |
|------------|-----------------|-------------------|------------------|
| Cámara     | ThreadedCamera  | Captura de frames | Hilo productor   |
| Inferencia | InferenceWorker | Pipeline IA       | submit() / get() |
| Uploader   | Uploader        | Envío HTTP        | Cola de tareas   |
| Serial     | SerialManager   | Comunicación UART | Cola de comandos |

**Fuente:** Elaboración propia.

El control de carga se define como:

$$\text{SKIP} = \max\left(1, \left\lfloor \frac{f_{\text{cam}}}{f_{\text{target}}} \right\rfloor\right) \quad (6.1)$$

donde  $f_{\text{cam}}$  es la tasa de captura y  $f_{\text{target}}$  la frecuencia objetivo de inferencia.

#### 6.4.2. Nivel 2 — Comunicación Serie (UART)

La interacción con el sistema físico se realiza mediante un enlace UART entre la Raspberry Pi y el Arduino.

**Tabla 6.3:** Parámetros de la comunicación serie.

| Parámetro Valor |                             |
|-----------------|-----------------------------|
| Puerto          | /dev/ttyUSB0 (configurable) |
| Velocidad       | 9600 bauds                  |

**Fuente:** Elaboración propia.

Ante una detección válida, se transmite un comando de un solo carácter:

- 'A': mosquito hembra.
- 'B': mosquito macho.

Este comando activa el mecanismo de clasificación física implementado en el microcontrolador.

#### 6.4.3. Nivel 3 — Comunicación HTTP

El envío de datos al servidor se realiza mediante una petición HTTP POST con codificación multipart/form-data al endpoint /upload.

**Tabla 6.4:** Campos enviados al servidor.

| Campo       | Tipo   | Descripción                   |
|-------------|--------|-------------------------------|
| client_id   | string | Identificador del dispositivo |
| temperature | float  | Temperatura ambiental (°C)    |
| humidity    | float  | Humedad relativa (%)          |
| detections  | JSON   | Resultados de inferencia      |
| image       | file   | Imagen capturada              |

**Fuente:** Elaboración propia.

El envío se ejecuta en un hilo independiente e incorpora un mecanismo de control de frecuencia (*cooldown*) para evitar la saturación del servidor ante eventos consecutivos.

### 6.5. Capa de Presentación (Frontend)

La capa de presentación fue implementada como una aplicación web estática desarrollada utilizando **HTML5**, **CSS** y **JavaScript** nativo (*vanilla JS*). Se prescindió de frameworks de frontend para minimizar dependencias externas, reducir la complejidad del despliegue y maximizar la portabilidad del sistema.

La interfaz está estructurada en múltiples páginas independientes, cada una orientada a una funcionalidad específica dentro del sistema. La siguiente tabla resume la organización del frontend.

**Tabla 6.5:** Estructura del frontend.

| Página          | Funcionalidad                                                |
|-----------------|--------------------------------------------------------------|
| home.html       | Página principal con presentación general del sistema.       |
| about.html      | Información institucional y objetivos del proyecto.          |
| model.html      | Documentación técnica del modelo de inteligencia artificial. |
| simulation.html | Visualización e interacción con la simulación ABM.           |
| dashboard.html  | Monitoreo en tiempo casi real de las detecciones IoT.        |

**Fuente:** Elaboración propia.

La comunicación con el backend se realiza mediante peticiones HTTP asíncronas utilizando la API `fetch()`, lo que permite actualizar dinámicamente el

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

contenido sin necesidad de recargar la página. En particular, el **dashboard** implementa un mecanismo de actualización periódica basado en `setInterval`, consultando el endpoint `/api/dashboard-data` cada pocos segundos.

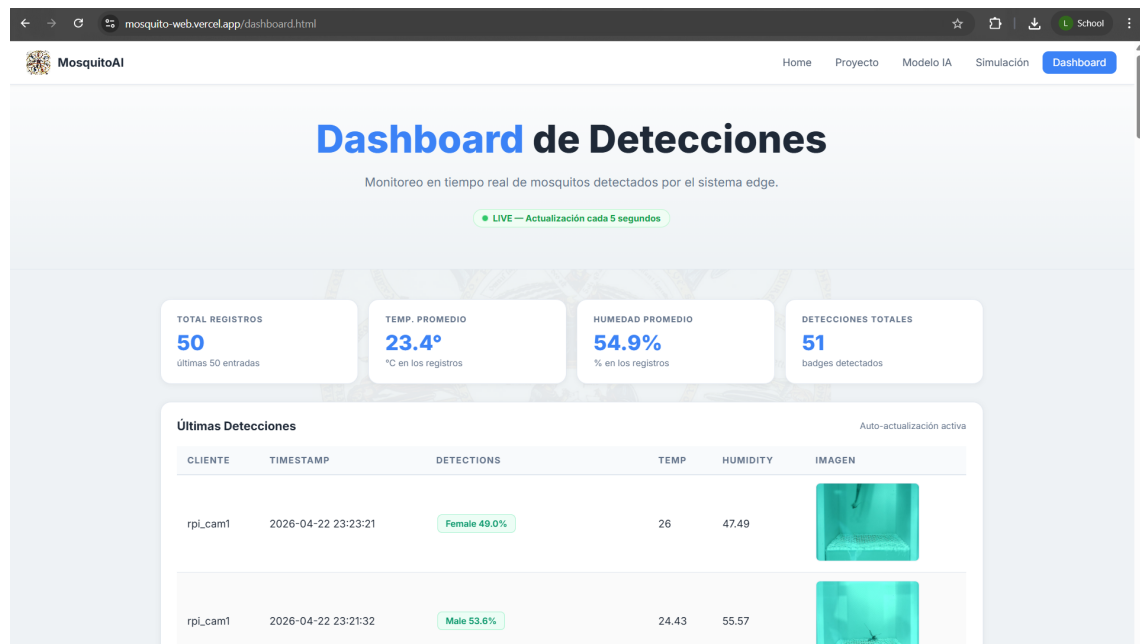
Además, la interfaz incorpora técnicas modernas de diseño web, como navegación responsiva, reutilización de componentes visuales y uso de variables CSS para mantener consistencia estética entre páginas.

En la Figura 6.2 se muestra la página principal del sistema, donde se presenta una visión general del proyecto y sus funcionalidades. Por su parte, la Figura 6.3 ilustra el dashboard de monitoreo, el cual centraliza la visualización de datos generados por los dispositivos IoT en operación.



**Figura 6.2:** Página de inicio de la aplicación web del sistema, con acceso a las secciones de dashboard, modelo ABM y documentación del proyecto. **Fuente:** Elaboración propia.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS



**Figura 6.3:** Dashboard de monitoreo IoT mostrando detecciones en tiempo casi real, clasificación por sexo y variables ambientales registradas por el sensor BME280. **Fuente:** Elaboración propia.

## 6.6. Servicios para la Base de Datos

La arquitectura de persistencia del sistema adopta un enfoque híbrido, combinando una base de datos relacional SQL para metadatos estructurados con un sistema de almacenamiento de objetos (bucket) para datos pesados como imágenes.

Esta separación responde a criterios de eficiencia y escalabilidad:

- **Base de datos SQL:** almacena información estructurada y consultable, como identificadores de dispositivo, marcas temporales, detecciones, temperatura y humedad.
- **Almacenamiento tipo bucket:** gestiona archivos binarios de gran tamaño (imágenes), evitando sobrecargar la base de datos y mejorando el rendimiento.

La consistencia entre ambos sistemas se garantiza mediante el almacenamiento de la URL pública de cada imagen dentro de la base de datos, vinculada a su registro correspondiente. De este modo, cada entrada mantiene coherencia temporal y semántica entre datos sensoriales, resultados de inferencia y evidencia visual.

### 6.6.1. Persistencia Relacional (PostgreSQL)

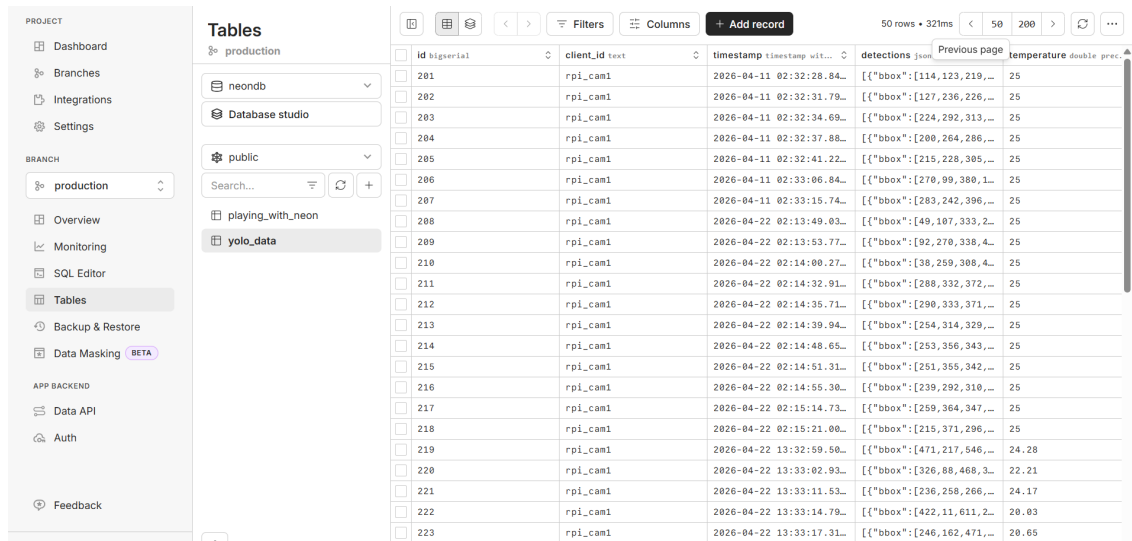
Una base de datos SQL permite almacenar datos relacionales organizados en tablas, facilitando consultas eficientes, integridad referencial y operaciones transaccionales.

En este sistema, la persistencia relacional se implementa mediante **Neon**, un servicio gestionado de PostgreSQL *serverless*, diseñado para aplicaciones modernas en la nube. Neon proporciona escalabilidad automática, suspensión por inactividad y separación entre almacenamiento y cómputo, lo que lo hace adecuado para prototipos IoT con cargas variables.

La base de datos almacena la tabla principal `yolo_data`, que contiene:

- `client_id`: identificador del dispositivo IoT.
- `timestamp`: marca temporal de la captura.
- `detections`: resultados del modelo (formato JSON).
- `temperature` y `humidity`: variables ambientales.
- `image_url`: enlace a la imagen almacenada.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS



| id  | client_id | timestamp              | detections                    | temperature |
|-----|-----------|------------------------|-------------------------------|-------------|
| 201 | rpi_cam1  | 2026-04-11 02:32:28.84 | [{"bbox": [114, 123, 219, ... | 25          |
| 202 | rpi_cam1  | 2026-04-11 02:32:31.79 | [{"bbox": [127, 236, 226, ... | 25          |
| 203 | rpi_cam1  | 2026-04-11 02:32:34.69 | [{"bbox": [224, 292, 313, ... | 25          |
| 204 | rpi_cam1  | 2026-04-11 02:32:37.88 | [{"bbox": [200, 264, 286, ... | 25          |
| 205 | rpi_cam1  | 2026-04-11 02:32:41.22 | [{"bbox": [215, 228, 305, ... | 25          |
| 206 | rpi_cam1  | 2026-04-11 02:33:06.84 | [{"bbox": [270, 99, 380, 1... | 25          |
| 207 | rpi_cam1  | 2026-04-11 02:33:15.74 | [{"bbox": [283, 242, 396, ... | 25          |
| 208 | rpi_cam1  | 2026-04-22 02:13:49.03 | [{"bbox": [49, 107, 333, 2... | 25          |
| 209 | rpi_cam1  | 2026-04-22 02:13:53.77 | [{"bbox": [92, 270, 338, 4... | 25          |
| 210 | rpi_cam1  | 2026-04-22 02:14:00.27 | [{"bbox": [338, 259, 384, ... | 25          |
| 211 | rpi_cam1  | 2026-04-22 02:14:32.91 | [{"bbox": [288, 332, 372, ... | 25          |
| 212 | rpi_cam1  | 2026-04-22 02:14:35.71 | [{"bbox": [290, 333, 371, ... | 25          |
| 213 | rpi_cam1  | 2026-04-22 02:14:39.94 | [{"bbox": [254, 314, 329, ... | 25          |
| 214 | rpi_cam1  | 2026-04-22 02:14:48.65 | [{"bbox": [253, 356, 343, ... | 25          |
| 215 | rpi_cam1  | 2026-04-22 02:14:51.31 | [{"bbox": [251, 355, 342, ... | 25          |
| 216 | rpi_cam1  | 2026-04-22 02:14:55.30 | [{"bbox": [239, 292, 310, ... | 25          |
| 217 | rpi_cam1  | 2026-04-22 02:15:14.73 | [{"bbox": [259, 364, 347, ... | 25          |
| 218 | rpi_cam1  | 2026-04-22 02:15:21.00 | [{"bbox": [215, 371, 296, ... | 25          |
| 219 | rpi_cam1  | 2026-04-22 13:32:59.50 | [{"bbox": [471, 217, 546, ... | 24.28       |
| 220 | rpi_cam1  | 2026-04-22 13:33:02.93 | [{"bbox": [326, 88, 468, 3... | 22.21       |
| 221 | rpi_cam1  | 2026-04-22 13:33:11.53 | [{"bbox": [236, 258, 286, ... | 24.17       |
| 222 | rpi_cam1  | 2026-04-22 13:33:14.79 | [{"bbox": [422, 11, 611, 2... | 20.03       |
| 223 | rpi_cam1  | 2026-04-22 13:33:17.31 | [{"bbox": [246, 162, 471, ... | 20.65       |

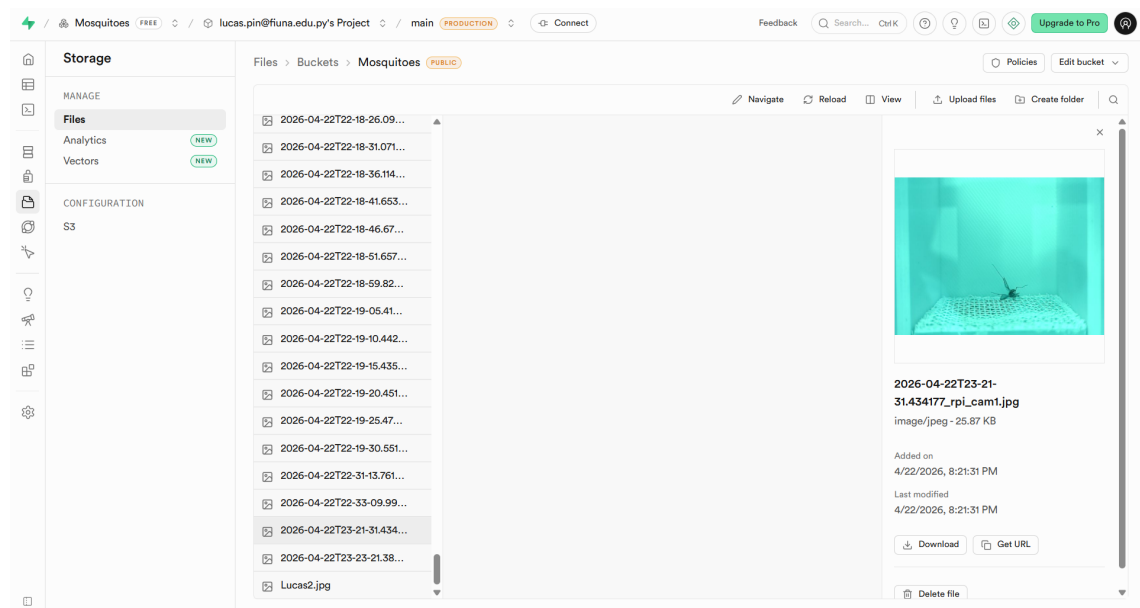
**Figura 6.4:** Vista de la tabla `yolo_data` en la base de datos PostgreSQL gestionada por Neon, con registros de detección, metadatos ambientales y URL de imagen asociada. **Fuente:** Elaboración propia.

## 6.6.2. Almacenamiento de Objetos

Para el almacenamiento de archivos binarios (imágenes), el sistema emplea **Supabase Storage**, un servicio de tipo *bucket* orientado a gestionar datos de gran tamaño mediante URLs públicas o autenticadas.

En este sistema, cada imagen capturada por el dispositivo de borde se almacena en el bucket `Mosquitoes`, generando una URL única que posteriormente se registra en la base de datos. Este enfoque separa el almacenamiento de datos pesados del procesamiento transaccional, mejorando el rendimiento global del sistema.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**



**Figura 6.5:** Bucket Mosquitoes en Supabase Storage con las imágenes capturadas por el dispositivo de borde, cada una vinculada a su registro en la base de datos mediante URL pública. **Fuente:** Elaboración propia.

En conjunto, Neon y Supabase conforman la capa de persistencia del sistema: mientras Neon garantiza la integridad y consultabilidad de los metadatos estructurados, Supabase gestiona eficientemente el almacenamiento de los datos binarios. La vinculación entre ambos sistemas, a través de la URL pública de cada imagen registrada en el bucket, asegura coherencia semántica y temporal entre cada evento de detección y su evidencia visual asociada.

## 6.7. Despliegue en la Nube

El sistema se despliega en la nube siguiendo una arquitectura distribuida entre backend y frontend, lo que permite escalar, actualizar y mantener cada componente de forma independiente.

Este enfoque presenta ventajas clave:

- Escalabilidad independiente de servicios.
- Mayor tolerancia a fallos.



- Flexibilidad para despliegues y actualizaciones continuas.
- Optimización de costos según carga de cada componente.

### 6.7.1. Despliegue del Servidor (Backend)

El servidor backend se despliega en la nube mediante una plataforma Platform as a Service (PaaS) que automatiza la construcción, configuración y ejecución de aplicaciones en contenedores. En este trabajo se utiliza **Railway** para dicha función.

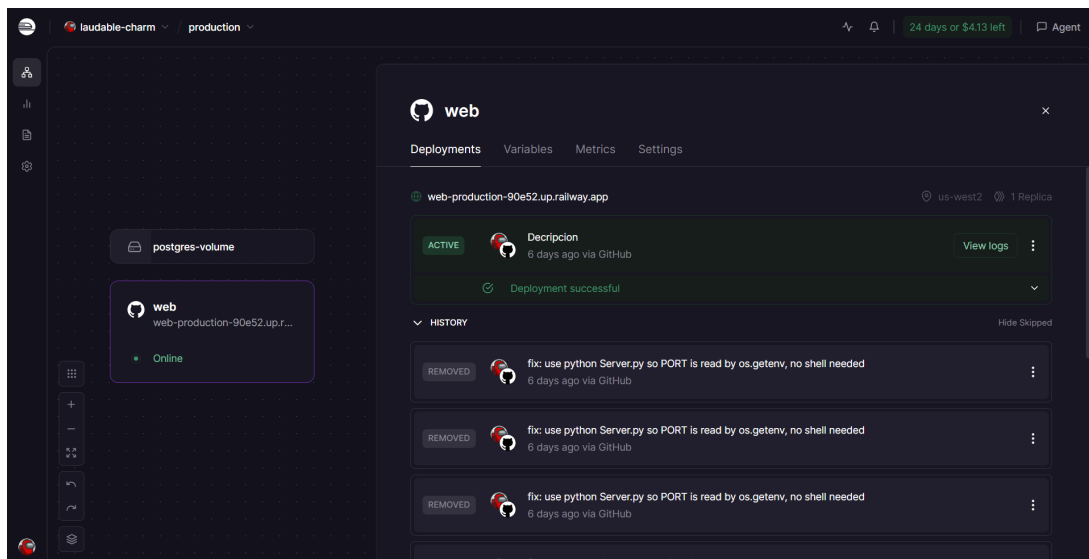
La plataforma permite desplegar directamente desde un repositorio Git, donde detecta el `Dockerfile` y construye la imagen del servicio. El contenedor incluye todas las dependencias necesarias (Python, OpenCV, librerías de IA, etc.), garantizando portabilidad y reproducibilidad.

El servidor FastAPI se ejecuta dentro del contenedor mediante Uvicorn, escuchando en el puerto asignado dinámicamente por la plataforma.

Este enfoque basado en contenedores permite:

- Aislamiento del entorno de ejecución.
- Reproducibilidad del sistema.
- Despliegues automáticos (Continuous Integration and Continuous Deployment (CI/CD)).

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**



**Figura 6.6:** Panel de control del entorno de despliegue del backend en Railway, mostrando el estado operativo del servicio durante las pruebas del sistema.  
**Fuente:** Elaboración propia.

### 6.7.2. Despliegue del Frontend

El frontend se despliega en una plataforma especializada en el alojamiento de aplicaciones web estáticas con distribución mediante un Content Delivery Network (CDN) global. En este trabajo se utiliza **Vercel** para dicha función.

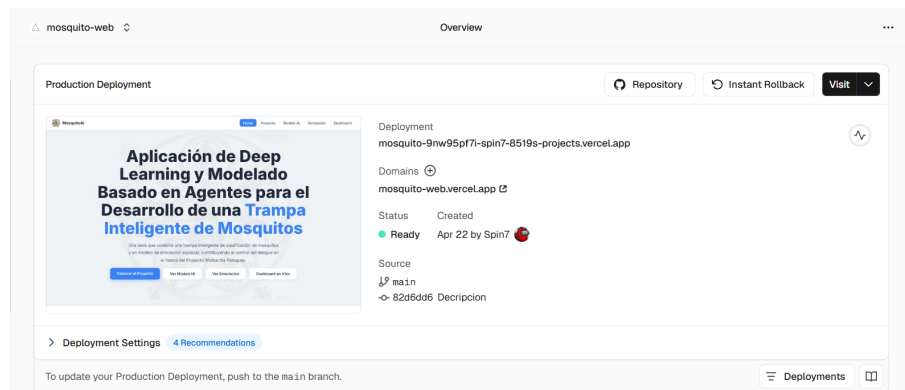
La plataforma permite desplegar directamente desde un repositorio, detectando automáticamente archivos estáticos (HTML, CSS, JavaScript) y publicándolos sin necesidad de configuración compleja.

El frontend consume la API del backend mediante peticiones HTTP, utilizando la URL configurada en el archivo `config.js`.

Las principales ventajas de este enfoque son:

- Baja latencia mediante distribución global (CDN).
- Despliegue instantáneo.
- Separación total de lógica cliente-servidor.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**



**Figura 6.7:** Panel de control del entorno de despliegue del frontend en Vercel, mostrando el estado del último despliegue exitoso durante las pruebas del sistema. **Fuente:** Elaboración propia.

## 6.8. Consideraciones de Seguridad

El sistema es un prototipo de investigación; en consecuencia, no se implementaron mecanismos de autenticación ni autorización sobre los endpoints de la API. La arquitectura adoptada incorpora, no obstante, salvaguardas implícitas: Railway y Vercel provisionan certificados TLS automáticamente (HTTPS por defecto en toda la comunicación entre cliente, backend y dispositivo de borde), las credenciales de Neon y Supabase se gestionan exclusivamente como variables de entorno en el backend y no están expuestas al frontend, y el enlace UART entre la Raspberry Pi y el Arduino es físicamente local al dispositivo.

Las extensiones necesarias para un despliegue en producción incluyen: autenticación mediante *API key* o JSON Web Token (JWT) en el endpoint `/upload` (actualmente accesible sin restricciones), restricción del origen CORS a los dominios legítimos del frontend (actualmente `allow_origins=["*"]`), validación del tipo y tamaño de los archivos recibidos, y protección del acceso a `/api/dashboard-data` mediante sesiones de usuario. En su estado actual, el sistema es adecuado para operar en entornos controlados de investigación; una transición a producción requeriría un rediseño orientado a la seguridad por defecto (*security by default*).

## CAPÍTULO 7

### Generación del Grid Map a partir de datos GIS

Este Capítulo describe el desarrollo del pipeline de construcción del entorno de simulación georreferenciado, correspondiente a la fase FDM2 del proyecto. El objetivo de esta fase es generar una representación discreta del territorio, denominada *Grid Map*. Esta representación codifica las características físicas del entorno urbano real y sirvió como espacio de actuación para los agentes de los modelos basados en agentes que se describirán en el Capítulo siguiente. Se documentan las cuatro etapas principales del pipeline: la definición paramétrica del área de estudio, la descarga de datos geoespaciales desde OpenStreetMap, la conversión del mapa vectorial continuo a una grilla discreta mediante rasterización por capas, y la integración de las posiciones reales de las trampas físicas como elementos georreferenciados dentro del entorno de simulación.

#### 7.1. Definición del área de estudio

El primer paso consiste en delimitar el área geográfica a modelar. En lugar de utilizar límites administrativos o polígonos predefinidos, se adopta un enfoque paramétrico centrado en un punto de referencia geográfico: el centroide del área de interés, expresado en grados decimales de latitud y longitud. El pipeline distingue dos categorías de parámetros. La Tabla 7.1 presenta los **parámetros de entrada**: valores que el usuario debe especificar para definir la ubicación geográfica y la resolución del área de interés. Por otro lado, la Tabla 7.2 resume los

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

**parámetros ajustables** del pipeline de clasificación, que cuentan con valores por defecto validados para el caso de estudio y pueden modificarse sin alterar la lógica del sistema.

**Tabla 7.1:** Parámetros de entrada del pipeline — definen la ubicación y resolución del Grid Map

| Símbolo     | Descripción                                | Valor empleado | Unidad  |
|-------------|--------------------------------------------|----------------|---------|
| $\varphi_c$ | Latitud del centroide del área de estudio  | -25,2637       | °       |
| $\lambda_c$ | Longitud del centroide del área de estudio | -57,5759       | °       |
| $L$         | Extensión del mapa (lado del cuadrado)     | 1000           | m       |
| $\Delta s$  | Tamaño de celda (resolución espacial)      | 3              | m/celda |

**Fuente:** Elaboración propia.

**Tabla 7.2:** Parámetros ajustables del pipeline — umbrales y buffers de clasificación

| Símbolo                | Descripción                                | Valor por defecto | Unidad |
|------------------------|--------------------------------------------|-------------------|--------|
| $\theta_{\text{edif}}$ | Umbral mínimo de cobertura para edificios  | 0,30              | —      |
| $\theta_{\text{agua}}$ | Umbral mínimo de cobertura para agua       | 0,20              | —      |
| $\theta_{\text{veg}}$  | Umbral mínimo de cobertura para vegetación | 0,45              | —      |
| $d_{\text{buf}}$       | Ancho de buffer geométrico para red vial   | 3                 | m      |

**Fuente:** Elaboración propia.

El centroide elegido  $(\varphi_c, \lambda_c)$  corresponde a un barrio de la ciudad de Asunción, Paraguay, donde se encuentran físicamente instaladas las trampas para mosquitos que serán integradas al mapa. A partir de ese punto, el sistema define un área cuadrada de  $L \times L$  metros del entorno urbano real. La resolución  $\Delta s$  se selecciona como un compromiso entre tres criterios: la capacidad de representar detalles físicos relevantes (como el ancho típico de calles y veredas), el nivel de detalle requerido por los agentes durante la simulación, y el costo computacional de construir y operar la grilla. Con estos parámetros, el número de celdas por

lado de la grilla es:

$$N = \left\lfloor \frac{L}{\Delta s} \right\rfloor \quad (7.1)$$

## 7.2. Descarga de datos desde OpenStreetMap

### 7.2.1. OpenStreetMap

OpenStreetMap (OSM) [41] es una base de datos geográfica colaborativa y de acceso libre, construida y mantenida por una comunidad global de colaboradores voluntarios. Su modelo de datos representa el mundo como un conjunto de *nodos* (puntos), *ways* (líneas y polígonos cerrados) y *relations* (agrupaciones lógicas), cada uno de los cuales puede llevar etiquetas clave-valor (*tags*) que describen su naturaleza semántica. Por ejemplo, un edificio lleva la etiqueta `building=yes`, una calle lleva `highway=residential`, y un parque puede tener `leisure=park`. A diferencia de fuentes de datos satelitales o censales que requieren licencias comerciales, OSM permite descargar de forma gratuita y programática cualquier subconjunto de datos geográficos, lo que lo convierte en una fuente adecuada para proyectos de investigación con recursos limitados.

### 7.2.2. Acceso a los datos mediante OSMnx

El acceso a los datos de OSM se realizó a través de la librería `OSMnx` [42], que proporciona una interfaz de alto nivel sobre la API Overpass de OSM. Esta librería permite solicitar elementos geográficos especificando tipo de entidad, punto central y radio de búsqueda, y devuelve los resultados como `GeoDataFrames`: colecciones vectoriales de geometrías geoespaciales gestionadas por la librería `GeoPandas`. Se realizan cuatro consultas independientes, cada una orientada a un tipo de elemento geográfico distinto:

1. **Edificios:** Se consultaron todos los polígonos con el tag `building=yes`,

capturando la huella de planta (*footprint*) de cualquier construcción, independientemente de su uso.

2. **Cuerpos de agua:** Se consultaron elementos con los tags `natural=water` y `natural=wetland`, incluyendo ríos, lagos, lagunas y humedales. Estas entidades son especialmente relevantes para el ciclo reproductivo del mosquito *Aedes aegypti*, que deposita huevos en superficies de agua estancada.
3. **Vegetación:** Se consultaron múltiples combinaciones de tags para cubrir todos los tipos de área verde presentes en entornos urbanos:
  - `landuse: forest, grass, meadow, orchard, vineyard`
  - `leisure: park, garden`
  - `natural: wood, scrub`
4. **Red vial:** Se descargó el grafo de calles completo, utilizando el parámetro `network_type=.all` para incluir calles vehiculares, peatonales y ciclovías. El grafo fue convertido a una colección de segmentos lineales (aristas del grafo) para su posterior procesamiento.

Todas las consultas se realizan con un radio de búsqueda capaz de contener al área de estudio ( $L \times L$ ), garantizando que los datos descargados cubran con margen suficiente toda la región analizada. Las geometrías resultantes están expresadas en el sistema de referencia de coordenadas World Geodetic System 1984 (WGS84) (EPSG:4326), es decir, latitud y longitud en grados decimales. Adicionalmente, se descargó una imagen de satélite de la misma zona a través del servicio de tiles Esri WorldImagery, que fue utilizada exclusivamente como referencia visual para validar la correspondencia cualitativa entre el grid generado y la realidad geográfica.

### 7.3. Conversión del mapa continuo a una grilla discreta

#### 7.3.1. Estructura de la grilla

La grilla se representa internamente como una matriz bidimensional de enteros sin signo de 8 bits (`uint8`), de dimensiones  $N \times N$ , inicializada a cero. Cada elemento  $g[r, c]$  de la matriz representa una celda cuadrada de  $\Delta_s \times \Delta_s$  metros sobre el territorio real, donde  $r$  es el índice de fila (eje norte-sur) y  $c$  es el índice de columna (eje este-oeste). La correspondencia entre índices de la matriz y posición geográfica se establece mediante un sistema de coordenadas locales centrado en el centroide del mapa. El origen local  $(0, 0)$  en metros corresponde al punto  $(\varphi_c, \lambda_c)$ , con el eje  $x$  apuntando hacia el este y el eje  $y$  apuntando hacia el norte. La celda  $[0, 0]$  corresponde a la esquina noroeste del mapa.

#### 7.3.2. Rasterización de polígonos

Para las capas de edificios, agua y vegetación se implementó un proceso de rasterización celda por celda que opera de la siguiente manera:

1. Para cada celda  $[r, c]$  de la grilla, se calcula su polígono geográfico correspondiente. Esto implica convertir el índice de la celda a coordenadas locales en metros, y luego a latitud y longitud mediante la aproximación esférica:

$$\varphi = \varphi_c + \frac{y}{R} \cdot \frac{180}{\pi}, \quad \lambda = \lambda_c + \frac{x}{R \cdot \cos(\varphi_c)} \cdot \frac{180}{\pi} \quad (7.2)$$

donde  $R = 6\,378\,137$  m corresponde al radio ecuatorial definido por el sistema WGS84,  $\varphi_c$  y  $\lambda_c$  son la latitud y longitud del centroide, e  $(x, y)$  son las coordenadas locales en metros.

2. Se consulta si algún polígono del conjunto de datos de la capa analizada intersecta geoméricamente con la celda actual.



3. En caso de intersección, se calcula la fracción de cobertura como:

$$\text{cover} = \frac{\text{Área}(\text{geom} \cap \text{celda})}{\text{Área}(\text{celda})} \quad (7.3)$$

4. Se aplica un umbral mínimo de cobertura ( $\text{min\_cover}$ ) diferenciado por clase, que actúa como filtro de asignación:

- **Edificios:**  $\theta_{\text{edif}}$  (véase Tabla 7.2) — umbral moderado para capturar tanto construcciones grandes como pequeñas.
- **Agua:**  $\theta_{\text{agua}}$  (véase Tabla 7.2) — umbral permisivo, ya que cualquier presencia significativa de agua es relevante para la dinámica del mosquito.
- **Vegetación:**  $\theta_{\text{veg}}$  (véase Tabla 7.2) — umbral más estricto para evitar clasificar como área verde celdas que solo presentan arbolado de vereda escaso.

5. Si se supera el umbral, la celda se clasifica con el identificador numérico correspondiente, siempre que dicho identificador sea mayor al valor actualmente almacenado en la celda. Esta condición implementa una **jerarquía de prioridad implícita**: una celda ya clasificada con una categoría de mayor índice no es sobrescrita por una de menor índice.

El orden de procesamiento en el pipeline respeta esta jerarquía: primero se rasterizan los edificios (clase 4), luego el agua (clase 3), y finalmente la vegetación (clase 2).

### 7.3.3. Rasterización de la red vial

Las calles son geometrías lineales (*LineStrings*), no polígonos, por lo que no es posible calcular una fracción de cobertura de área directamente. Para superar

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

---

esta limitación, se aplica un buffer geométrico a cada segmento vial antes de realizar la intersección:

- Se genera una versión expandida de cada segmento lineal aplicando un buffer geométrico de ancho  $d_{\text{buf}}$  (véase Tabla 7.2), expresado en metros. Esto convierte las líneas en bandas poligonales que representan el ancho aproximado de una vía.
- Para cada celda de la grilla, se verifica si alguno de esos polígonos interseca la celda.
- Una celda solo se clasifica como calle (clase 1) si su valor actual es 0 (*Empty*), garantizando que las calles no sobrescriban clases de mayor prioridad.

Esta lógica posiciona a la red vial como la capa de menor precedencia en el proceso de clasificación.

#### 7.4. Clasificación del entorno urbano

El sistema reconoce cinco tipos de celda, identificados por un entero en el rango  $[0, 4]$ , tal como se describe en la Tabla 7.3.

**Tabla 7.3:** Tipos de celda definidos en el Grid Map

| ID | Clase              | Color       | Descripción                                                      |
|----|--------------------|-------------|------------------------------------------------------------------|
| 0  | <i>Empty Space</i> | Crema claro | Espacio no clasificado; suelo libre sin estructura identificada. |
| 1  | <i>Road</i>        | Gris        | Vías de circulación: calles, sendas y ciclovías.                 |
| 2  | <i>Vegetation</i>  | Verde       | Áreas verdes: parques, jardines, bosques, praderas.              |
| 3  | <i>Water</i>       | Azul        | Cuerpos de agua: ríos, lagunas y humedales.                      |
| 4  | <i>Building</i>    | Marrón      | Edificaciones de cualquier tipo.                                 |

**Fuente:** Elaboración propia.

La elección de estas cinco categorías responde a criterios ecológicos y de modelado: el mosquito *Aedes aegypti* presenta preferencia por desplazarse en entornos sombreados y con cobertura vegetal, evitando en muchos casos áreas abiertas con alta exposición solar. Disponer de estas categorías diferenciadas en la retícula permite que los agentes tomen decisiones de movimiento y reproducción basadas en el tipo de celda en que se encuentran, añadiendo realismo ecológico al modelo. Además de estas cinco categorías, las celdas que contienen una trampa física son marcadas con el identificador especial **clase 5** (*Trap*, visualizada en magenta) durante la etapa de integración de trampas descrita en la sección siguiente, sobrescribiendo la clasificación previa con independencia del tipo de terreno subyacente. Esta distinción permite que el entorno de simulación registre con precisión georreferenciada la posición de cada trampa, conservando al mismo tiempo la clasificación ecológica base de las celdas circundantes.

## 7.5. Integración de las trampas reales

### 7.5.1. Origen de los datos

Las trampas para mosquitos utilizadas en este estudio son dispositivos físicos instalados en puntos conocidos del barrio analizado; se trata de ovitrampas instaladas con anterioridad a este trabajo, cuya función es la captura de huevos de *Aedes aegypti* con fines de monitoreo epidemiológico, sin contemplar el conteo de individuos adultos. Sus coordenadas geográficas —latitud y longitud— fueron registradas en campo y almacenadas en formato de tabla de datos. Sin embargo, el archivo original presentó inconsistencias de formato habituales en datos recopilados manualmente: uso de coma como separador decimal en lugar de punto, presencia de caracteres de comilla embebidos en los valores numéricos, y registros con coordenadas faltantes o fuera del rango geográfico válido.

### 7.5.2. Preprocesamiento y limpieza de coordenadas

Antes de integrar las trampas al mapa, se ejecutó una etapa de preprocesamiento automático sobre el archivo de coordenadas. Esta etapa realiza las siguientes operaciones de forma secuencial:

1. Carga del archivo como texto crudo, evitando conversiones automáticas que podrían descartar silenciosamente filas inválidas.
2. Eliminación de caracteres de comilla embebidos en los campos numéricos.
3. Reemplazo del separador decimal de coma a punto, adoptando el formato internacional estándar.
4. Conversión de los campos de texto a tipo numérico de punto flotante; los valores no convertibles se reemplazan por valores faltantes en lugar de generar un error.
5. Filtrado y descarte de los registros cuya latitud no esté en el rango  $[-90, 90]$  o cuya longitud no esté en  $[-180, 180]$ , eliminando coordenadas geográficamente inválidas.
6. Exportación del conjunto de datos limpio a un nuevo archivo, preservando el original sin modificaciones para garantizar trazabilidad y reproducibilidad.

### 7.5.3. Transformación de coordenadas a índices de grilla

Una vez saneados los datos, cada trampa es posicionada dentro del grid mediante una transformación de dos etapas. **Etapas 1: Conversión de coordenadas geográficas a métricas.** Se aplica una proyección cilíndrica equirectangular, válida para áreas pequeñas donde la curvatura terrestre es despreciable:

$$x = R \cdot \Delta\lambda \cdot \cos(\varphi_c), \quad y = R \cdot \Delta\varphi \quad (7.4)$$

donde  $\Delta\lambda = \lambda_t - \lambda_c$  y  $\Delta\varphi = \varphi_t - \varphi_c$  son las diferencias angulares (en radianes) entre la trampa  $(\varphi_t, \lambda_t)$  y el centroide  $(\varphi_c, \lambda_c)$ , y  $R = 6\,378\,137$  m es el radio ecuatorial del elipsoide WGS84. El resultado es un par  $(x, y)$  que expresa la posición de la trampa en metros desde el centro del mapa. **Etapla 2: Conversión de metros a índices de matriz.** Las coordenadas métricas se convierten a índices enteros de la grilla mediante:

$$c = \left\lfloor \frac{x + \frac{L}{2}}{\Delta s} \right\rfloor, \quad r = \left\lfloor \frac{\frac{L}{2} - y}{\Delta s} \right\rfloor \quad (7.5)$$

donde  $L = 1000$  m es el lado del mapa,  $\Delta s = 3$  m es el tamaño de celda,  $c$  es el índice de columna (eje este) y  $r$  es el índice de fila (eje norte). El signo invertido en el cálculo de  $r$  se debe a que el índice de fila 0 corresponde al borde norte de la matriz, mientras que el eje  $y$  local apunta hacia el norte.

#### 7.5.4. Validación y registro

Solo se integran al grid las trampas cuyo par de índices calculado  $(r, c)$  se encuentra dentro de los límites de la matriz, es decir:

$$0 \leq r < N \quad y \quad 0 \leq c < N \quad (7.6)$$

Las trampas que caen fuera del kilómetro cuadrado de estudio son descartadas. Para cada trampa válida, la celda correspondiente se marca con la clase 5 (*Trap*), sobrescribiendo cualquier clasificación previa. De esta manera, las trampas tienen la máxima prioridad en el sistema de clasificación. Junto con la grilla generada, se exporta un archivo de configuración del entorno en formato JSON que registra, para cada trampa integrada, las coordenadas geográficas originales, la posición en el sistema métrico local, y los índices exactos dentro de la grilla.

### 7.5.5. Archivos de salida

Como resultado del proceso completo, el pipeline genera los siguientes archivos:

- **Matriz del grid:** archivo binario `.npy` con la representación matricial de la clasificación de cada celda.
- **Visualización:** imagen en alta resolución (300 dpi) del grid con la paleta de colores de las cinco clases del entorno y la clase especial de trampa.
- **Configuración del entorno:** archivo JSON con los parámetros geográficos del mapa y la lista completa de trampas integradas.
- **Imagen satelital de referencia:** imagen del área de estudio obtenida del servicio de teselas Esri WorldImagery.
- **Comparación visual:** figura lado a lado de la imagen satelital y el grid generado, para validación cualitativa del proceso de clasificación.

### 7.6. Resultado: comparación entre imagen satelital y grid generado

La Figura 7.1 presenta una comparación directa entre la imagen satelital del área de estudio y el grid discreto generado por el proceso descrito a lo largo de este Capítulo. La imagen de la izquierda muestra la zona real tal como es captada por el servicio de imágenes Esri WorldImagery, mientras que la imagen de la derecha muestra la representación abstracta obtenida tras la descarga, rasterización y clasificación de los datos de OpenStreetMap. La correspondencia visual entre ambas imágenes permite validar cualitativamente la fidelidad del proceso de conversión: los bloques de edificaciones, la trama vial, las áreas verdes y las posiciones de las trampas se reflejan con coherencia geométrica en la grilla discreta.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

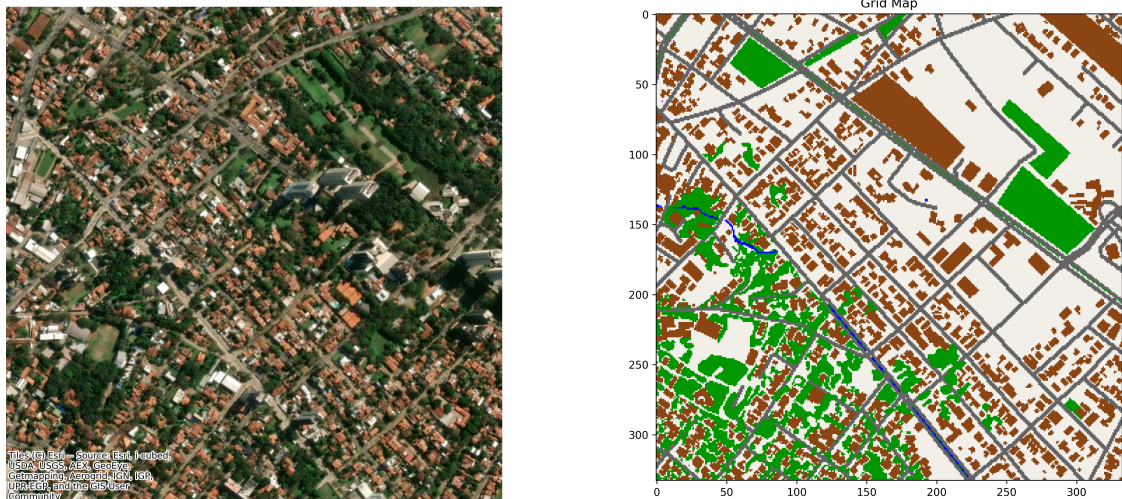


Imagen satelital de referencia  
(Esri WorldImagery)

Grid Map generado  
( $333 \times 333$  celdas,  $\Delta_s = 3$  m)

**Figura 7.1:** Comparación entre la imagen satelital y el Grid Map discreto generado a partir de datos de OpenStreetMap. Cada celda del grid representa una zona de  $3 \times 3$  m del territorio real. **Fuente:** Elaboración propia.

El Grid Map resultante constituye el entorno de simulación sobre el que operan los dos modelos basados en agentes descritos en el Capítulo 8. Su diseño paramétrico —centroide, extensión y resolución de celda como únicos insumos requeridos— permite adaptar el entorno a cualquier área urbana georreferenciada sin modificar la lógica del pipeline. La retícula discreta así obtenida, con su heterogeneidad de terreno (espacios libres, calles, vegetación, agua y edificios) y las posiciones exactas de las trampas integradas como clase 5, proporciona al simulador un anclaje geográfico real que reemplaza los entornos sintéticos homogéneos habitualmente empleados en modelos de dinámica vectorial. Cabe destacar que el identificador de trampa admite indistintamente ovitrampas ya instaladas en el área de estudio o trampas de conteo de adultos como el prototipo desarrollado en este trabajo, lo que habilita, en una etapa futura, la comparación directa entre los conteos registrados en campo y las predicciones del modelo, cerrando el ciclo de calibración de parámetros.

## CAPÍTULO 8

### Modelado y Simulación de Poblaciones de Mosquitos (ABM)

Este Capítulo documenta el desarrollo de los modelos basados en agentes correspondientes a las fases FDM3 y FDM4 del proyecto, los cuales operan sobre el entorno georreferenciado construido en el Capítulo 7. Se presentan dos modelos de simulación espacial para la dinámica poblacional del mosquito *Aedes aegypti* sobre una retícula discreta georreferenciada generada a partir de datos reales de OpenStreetMap, que captura la heterogeneidad del entorno urbano (edificios, calles, agua, vegetación y espacios libres).

Como señaló la revisión de la literatura (Tabla 2.2), la gran mayoría de los modelos espacio-temporales existentes omite la distinción de sexos: Chen et al. [1], Maneerat y Daudé [33] y Yamashita et al. [24] consideran únicamente hembras adultas, lo que limita su aplicabilidad para evaluar estrategias de control que dependen explícitamente de la dinámica reproductiva —como la SIT, cuya viabilidad Ferreira et al. [34] analizaron precisamente incorporando machos silvestres y estériles como agentes diferenciados. El presente trabajo retoma esa brecha: los ABM, al codificar reglas de interacción local entre individuos, constituyen el marco natural para modelar encuentros entre machos y hembras como mecanismo generador de la reproducción.

El **primer modelo** adopta un enfoque híbrido: las probabilidades de transición demográfica de los agentes —mortalidad y maduración— se derivan directamente del ODE propuesto por [1], que describe la dinámica de *Aedes aegypti* en función de la temperatura. La contribución propia es la incorporación de diná-



mica espacial mediante un campo de potencial georreferenciado que modula el movimiento de los adultos. Al heredar la formulación de Chen et al., este modelo comparte su limitación estructural: no distingue sexos ni modela apareamiento, lo que lo sitúa en la misma línea que los antecedentes continuos revisados y restringe su uso a escenarios donde la dinámica reproductiva no es el foco de análisis. Su principal limitación propia reside en el mecanismo de generación de juveniles: la distribución de nacimientos entre sitios de cría depende de la distancia global de cada adulto a *todos* los sitios del dominio, lo que diluye el factor local y reduce el realismo biológico de la reproducción.

El **segundo modelo** se aleja del sistema ODE de referencia para introducir reglas de transición propias. Incorpora estados más complejos —diferenciando sexo y estado reproductivo— y prioriza la **interacción estrictamente local**: la mortalidad juvenil es densodependiente dentro de un radio, el apareamiento requiere la presencia de machos en la vecindad, y la oviposición se restringe a los sitios de cría dentro de un radio máximo. Esta formulación responde directamente a la pregunta de diseño que dejaron abierta los antecedentes: explorar si una capacidad de carga emergente puede surgir de reglas locales —competencia por sitios de cría y mortalidad densodependiente— en lugar de ser fijada como constante, siguiendo la intuición analítica de Ishida et al. [25] sobre cotas emergentes en sistemas con mortalidad dependiente del movimiento. Al distinguir sexos y modelar el apareamiento como evento de vecindad, este modelo es además el más apropiado para evaluar en el futuro estrategias como la SIT, cuya efectividad —como demostró Ferreira et al. [34]— depende directamente de la distribución espacial heterogénea de los criaderos y de la proporción local de machos estériles respecto a los silvestres.

## 8.1. Entorno de simulación

Ambos modelos operan sobre el mismo entorno georreferenciado generado a partir de datos de OpenStreetMap (véase el Capítulo 7 para el proceso de construcción). El entorno es una **retícula discreta rectangular**  $\mathcal{G} \subset \mathbb{Z}^2$  de dimensiones  $N \times N$  celdas, donde cada celda  $x \in \mathcal{G}$  tiene asignado un tipo de terreno  $c(x) \in \{0, 1, 2, 3, 4\}$  según la Tabla 8.1.

**Tabla 8.1:** Tipos de terreno en la retícula y su rol en los modelos ABM

| $c(x)$ | Tipo          | Descripción                                   |
|--------|---------------|-----------------------------------------------|
| 0      | Espacio vacío | Transitable sin restricción para adultos.     |
| 1      | Calle         | Acceso restringido.                           |
| 2      | Vegetación    | Transitable; soporte de sitios de cría.       |
| 3      | Agua          | Acceso restringido.                           |
| 4      | Edificio      | Transitable para adultos (con restricciones). |

**Fuente:** Elaboración propia.

Un subconjunto finito  $\mathcal{S} = \{s_1, \dots, s_m\} \subset \mathcal{G}$  define los **sitios de cría**. Estos pueden cargarse desde coordenadas GPS reales o generarse aleatoriamente sobre celdas de vegetación ( $c(x) = 2$ ); en ningún caso se ubican sobre agua ( $c(x) = 3$ ). Las fronteras del dominio son **reflectivas**: toda posición fuera de  $\mathcal{G}$  se recorta al borde más cercano.

Cada modelo define su propio campo de potencial  $\Pi(x)$  y sus propias reglas de dinámica de agentes, descritos en las secciones correspondientes.

## 8.2. Primer modelo

El modelo presentado en esta sección constituye la primera versión del simulador basado en agentes desarrollado en el marco de este Trabajo Final de Grado. Sus resultados fueron publicados y defendidos en formato póster en el

**XLIV Congreso Nacional de Matemática Aplicada e Computacional** (CN-MAC 2025, Río de Janeiro, Brasil) y presentados en exposición oral en el **X Congreso de la Sociedad Científica Paraguaya** (octubre de 2025, Asunción, Paraguay) [43]. El segundo modelo, descrito en la Sección 8.3, constituye una extensión ampliada de este trabajo inicial. La propuesta central del modelo consiste en combinar tres componentes: (i) una dinámica temporal de los agentes derivada de un sistema de ecuaciones diferenciales ordinarias ODE tomado de la literatura, (ii) un entorno discreto georreferenciado con un campo de potencial que modula el movimiento, y (iii) un mecanismo estocástico de generación de juveniles en sitios de cría con capacidad de carga limitada.

### 8.2.1. Base matemática: sistema ODE de referencia

Las probabilidades de transición demográfica de los agentes (muerte y maduración) se derivaron del siguiente sistema de ecuaciones diferenciales ordinarias, propuesto por [1] para modelar la dinámica de poblaciones de *Aedes aegypti* en función de la temperatura:

$$\begin{aligned} J'(t) &= b(T) \left( 1 - \frac{J(t)}{K} \right) A(t) - \mu_1(T) J(t) - d(T) J(t), \\ A'(t) &= 0,5 d(T) J(t) - \mu_2(T) A(t), \end{aligned} \tag{8.1}$$

donde  $J(t)$  es la población de juveniles (larvas y pupas),  $A(t)$  la de adultos,  $K$  la capacidad de carga global del entorno, y los parámetros biológicos  $b(T)$ ,  $d(T)$ ,  $\mu_1(T)$  y  $\mu_2(T)$  dependen de la temperatura  $T$ . La Tabla 8.2 define cada parámetro.

**Tabla 8.2:** Parámetros del sistema ODE de Jing Chen [1].

| Símbolo    | Descripción                                      |
|------------|--------------------------------------------------|
| $b(T)$     | Tasa de fertilidad dependiente de la temperatura |
| $d(T)$     | Tasa de desarrollo de juvenil a adulto           |
| $\mu_1(T)$ | Tasa de mortalidad juvenil                       |
| $\mu_2(T)$ | Tasa de mortalidad adulta                        |
| $K$        | Capacidad de carga total del entorno             |
| $K_s$      | Capacidad de carga del sitio de cría $s$         |
| $\Delta t$ | Paso de tiempo de la simulación                  |

**Fuente:** Elaboración propia.

### 8.2.2. Temperatura ambiental y parámetros biológicos

Se adoptó un modelo de temperatura homogénea en todo el dominio espacial, descrita por un oscilador diurno simple:

$$T(t) = 25 + 5 \sin(2\pi t), \quad (8.2)$$

con  $t$  medido en días. Esto implica una temperatura media de  $25^\circ\text{C}$ , una amplitud de oscilación de  $\pm 5^\circ\text{C}$  y un período de 24 horas. Los parámetros biológicos del sistema (8.1) dependen de  $T$ , expresada en kelvin ( $T_k = T + 273,15$ ), a través de las siguientes expresiones tomadas de [1]:

**Mortalidad juvenil  $\mu_1(T)$ .**

$$\mu_1(T) = \begin{cases} \infty & T \notin (T_0^{(1)}, T_m^{(1)}), \\ \frac{1}{c_1 (T - T_0^{(1)}) (T_m^{(1)} - T)^2} & \text{en otro caso,} \end{cases} \quad (8.3)$$

con  $c_1 = 0,0254$ ,  $T_0^{(1)} = -2,3209$ ,  $T_m^{(1)} = 31,0033$ .

**Mortalidad adulta**  $\mu_2(T)$ .

$$\mu_2(T) = \begin{cases} \infty & T \notin (T_0^{(2)}, T_m^{(2)}), \\ \frac{1}{c_2 (T - T_0^{(2)}) (T_m^{(2)} - T)^2} & \text{en otro caso,} \end{cases} \quad (8.4)$$

con  $c_2 = 0,1037$ ,  $T_0^{(2)} = 6,4429$ ,  $T_m^{(2)} = 41,4382$ .

**Fertilidad**  $b(T)$ .

$$b(T) = \begin{cases} 0 & T \notin (T_0^{(b)}, T_m^{(b)}), \\ c_b T (T - T_0^{(b)}) \sqrt{T_m^{(b)} - T} & \text{en otro caso,} \end{cases} \quad (8.5)$$

con  $c_b = 0,0058$ ,  $T_0^{(b)} = 14,0343$ ,  $T_m^{(b)} = 39,0899$ .

**Desarrollo juvenil**  $d(T)$ .

$$d(T) = \frac{a T_k \exp(b (1/298,15 - 1/T_k))}{298,15 (1 + \exp(c (1/d_0 - 1/T_k)))}, \quad (8.6)$$

donde  $T_k = T + 273,15$  es la temperatura en Kelvin, y las constantes son  $a = 0,1666$ ,  $b = 21388$ ,  $c = -32013$ ,  $d_0 = 300,02$ .

### 8.2.3. Campo de potencial

El entorno  $\mathcal{G}$ , los tipos de terreno  $c(x)$  y los sitios de cría  $\mathcal{S}$  son los definidos en la Sección 8.1. En este primer modelo, el campo de potencial  $\Pi(x)$  combina la proximidad a los sitios de cría con un factor de ponderación dependiente del tipo de terreno:

$$\Pi(x) = w(c(x)) \Pi_0(x), \quad (8.7)$$

$$\Pi_0(x) = \begin{cases} \sum_{s \in \mathcal{S}} \Pi_b \exp\left(-\frac{\|x - s\|_2}{\lambda}\right), & \min_{s \in \mathcal{S}} \|x - s\|_2 \leq R_c, \\ 0, & \text{en otro caso,} \end{cases} \quad (8.8)$$

con  $\Pi_b = 0,5$ ,  $\lambda = 6$  y  $R_c$  el radio de influencia de cada sitio. El factor de peso penaliza o premia el tránsito según el tipo de terreno:

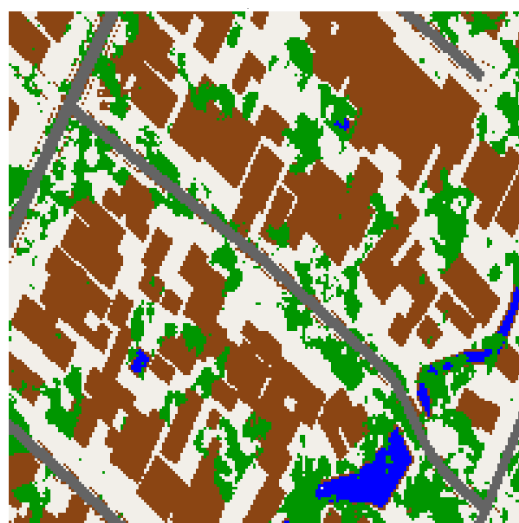
$$w(c) = \begin{cases} 1,0 & c = 0 \quad (\text{libre}), \\ 2,0 & c = 2 \quad (\text{vegetación}), \\ 0,0 & c = 1 \quad (\text{calle}), \\ 0,3 & c = 4 \quad (\text{edificio}), \\ 0,0 & c = 3 \quad (\text{agua}). \end{cases} \quad (8.9)$$

Las celdas de calle y agua reciben  $w = 0$  (en concordancia con [33]), inhibiendo el movimiento de los agentes hacia estas celdas a través del campo de potencial.

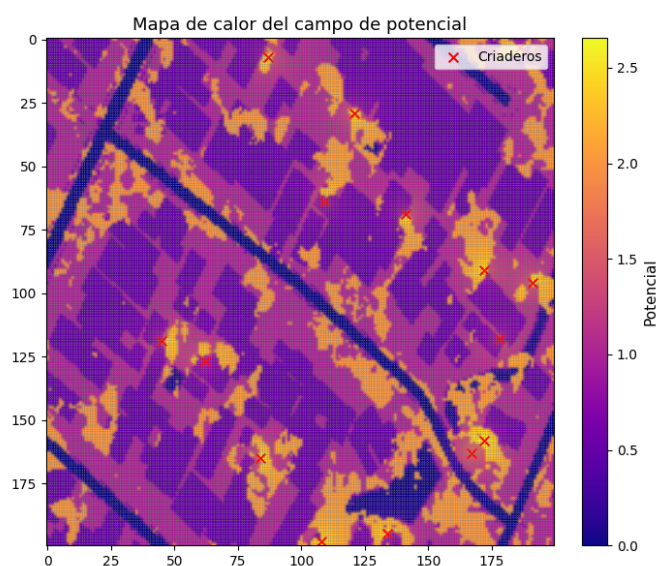
APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS



(a) Imagen satelital de la zona de estudio.



(b) Grid Map discreto generado a partir de datos GIS. Colores: libre (crema), calle (gris), vegetación (verde), agua (azul), edificio (marrón), trampa (magenta).



(c) Mapa de calor del campo de potencial  $\Pi(x)$  calculado sobre el Grid Map. Los criaderos activos se marcan con cruces rojas. Las celdas de calle y agua reciben  $w = 0$ , resultando en potencial nulo.

**Figura 8.1:** Del territorio real al campo de potencial. (a) imagen satelital de referencia; (b) Grid Map discreto georreferenciado generado desde OpenStreetMap; (c) campo de potencial  $\Pi(x)$  resultante, que guía el movimiento de los agentes adultos hacia los sitios de cría. **Fuente:** Elaboración propia.

#### 8.2.4. Dinámica espacial de adultos

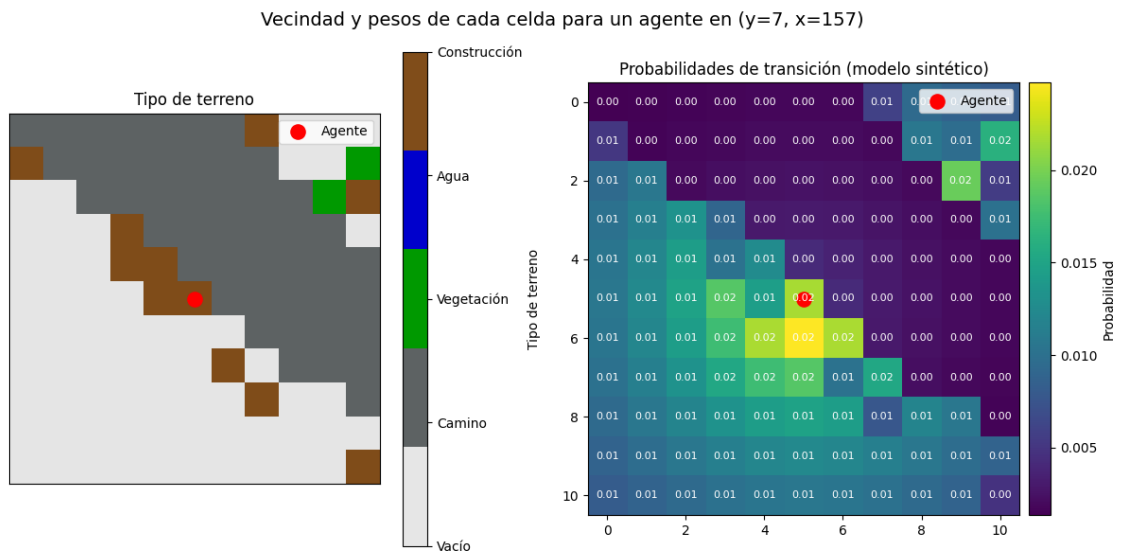
Sea  $X_i(t) = x \in \mathcal{G}$  la posición del adulto  $i$  en el tiempo  $t$ . Su vecindad de movimiento se define como un cuadrado de lado  $R$  (en celdas) centrado en  $x$ , denotado  $\mathcal{N}_R(x)$ . Los pesos de transición que combinan atracción hacia sitios de cría y penalización por distancia son:

$$\tilde{w}(y | x) = \frac{\Pi(y) + \beta_0}{0,5 \|y - x\|_2 + 1}. \quad (8.10)$$

y la probabilidad de transición al saltar a la celda  $y$  en el siguiente paso es:

$$\mathbb{P}(X_i(t + \Delta t) = y | X_i(t) = x) = \frac{\tilde{w}(y | x)}{\sum_{z \in \mathcal{N}_R(x)} \tilde{w}(z | x)}. \quad (8.11)$$

La siguiente figura permite visualizar gráficamente la ecuación anterior.



**Figura 8.2:** Probabilidades de transición de cada celda en la vecindad para un agente en un paso de tiempo. **Fuente:** Elaboración propia.



### 8.2.5. Dinámica temporal de juveniles y adultos

Las tasas de transición demográfica se convierten a probabilidades discretas mediante la aproximación exponencial estándar.

**Juveniles.** La tasa total de evento juvenil agrega maduración y mortalidad:

$$r_J(t) = d(t) + \mu_1(t), \quad p_{\text{ev},J} = 1 - e^{-r_J(t) \Delta t}. \quad (8.12)$$

dado que ocurre un evento, su tipo se selecciona de forma proporcional:

$$\mathbb{P}(\text{maduración}) = \frac{d(t)}{r_J(t)}, \quad \mathbb{P}(\text{muerte}) = \frac{\mu_1(t)}{r_J(t)}. \quad (8.13)$$

y el juvenil que madura transiciona a adulto con probabilidad 0,5 de ser macho o hembra:

$$\mathbb{P}(\text{transición a adulto} \mid \text{maduración}) = 0,5. \quad (8.14)$$

**Adultos.** La única transición posible para los adultos es la muerte, con probabilidad:

$$p_{\text{ev},A} = 1 - e^{-\mu_2(t) \Delta t}. \quad (8.15)$$

### 8.2.6. Generación de juveniles

El mecanismo de reproducción asigna a cada adulto una probabilidad de contribuir a cada sitio de cría de forma inversamente proporcional a su distancia, con limitación por capacidad de carga.

**Función de decaimiento y probabilidad de asignación.** La distancia entre el adulto  $i$  en posición  $X_i(t)$  y el sitio  $s$  es  $d_{is} = \|X_i(t) - s\|_2$ . Se define la función de

decaimiento exponencial:

$$\kappa(d_{is}) = e^{-d_{is}/\tau_{\text{rep}}}. \quad (8.16)$$

la probabilidad de que el adulto  $i$  contribuya al sitio  $s$  es:

$$\phi_s(X_i(t)) = \frac{\kappa(d_{is})}{\sum_{u \in S} \kappa(d_{iu})}. \quad (8.17)$$

**Capacidad de carga y nacimientos.** La capacidad del sitio  $s$  se controla mediante el factor de saturación:

$$\psi_s(t) = \max\left(0, 1 - \frac{J_s(t)}{K_s}\right). \quad (8.18)$$

la intensidad de nacimientos del adulto  $i$  hacia el sitio  $s$  es:

$$\lambda_{i \rightarrow s} = b(t) \phi_s(X_i(t)) \psi_s(t), \quad (8.19)$$

el número de nuevos juveniles producidos se modela como una variable aleatoria de Poisson:

$$N_{i \rightarrow s} \sim \text{Poisson}(\lambda_{i \rightarrow s} \Delta t). \quad (8.20)$$

**Regla de actualización.** La población de juveniles en cada sitio se actualiza al final del paso temporal:

$$J_s(t + \Delta t) = J_s(t) + \sum_{i \in A_t} N_{i \rightarrow s} - (\text{juveniles madurados y muertos en } s), \quad (8.21)$$

donde  $A_t$  denota el conjunto de todos los adultos vivos en el instante  $t$ .

## Síntesis del primer modelo

El primer modelo integra tres componentes arquitectónicos bien delimitados: (i) tasas demográficas derivadas del sistema ODE de [1], que vinculan mortalidad y maduración a la temperatura mediante expresiones analíticas validadas empíricamente; (ii) un campo de potencial  $\Pi(x)$  ponderado por tipo de terreno que guía el movimiento estocástico adulto hacia los sitios de cría; y (iii) un mecanismo de Poisson con decaimiento exponencial por distancia para la generación de juveniles.

La fortaleza del modelo reside en su base biológica: los parámetros demográficos heredados de [1] están validados en la literatura y su dependencia térmica incorpora realismo sin aumentar la carga de parametrización. Su limitación central es la pérdida de localidad en la reproducción: la función  $\phi_s(X_i(t))$  (ecuación 8.17) distribuye los nacimientos de cada adulto entre *todos* los sitios de  $\mathcal{S}$  sin restricción de radio, diluye el factor local y reduce el realismo biológico de la oviposición. Estas limitaciones son las que inspiran el desarrollo del segundo modelo buscando respetar el paradigma de interacciones estrictamente locales del ABM.

### 8.3. Segundo modelo

Mientras que el primer modelo deriva las transiciones demográficas a partir de tasas continuas provenientes del sistema ODE de [1], el segundo adopta una formulación completamente discreta basada en procesos de Bernoulli aplicados individualmente a cada agente y, bajo probabilidades homogéneas en cada paso temporal, en distribuciones Binomiales para describir los conteos agregados, priorizando la representación explícita de eventos individuales e interacciones estrictamente locales. A diferencia del modelo anterior, esta formulación no depende de ecuaciones diferenciales agregadas, sino de reglas estocásticas aplicadas

directamente sobre cada agente y su vecindario inmediato. En consecuencia, las dinámicas poblacionales emergen de la superposición de eventos individuales independientes.

El modelo introduce además estados biológicos adicionales —sexo y estado reproductivo— permitiendo representar de forma más detallada el ciclo de vida del mosquito. La dinámica reproductiva queda así definida exclusivamente por interacciones locales: el apareamiento requiere la presencia de machos dentro de un radio finito, la oviposición solo puede ocurrir en sitios de cría cercanos y la mortalidad juvenil depende de la densidad local de individuos.

El entorno compartido ( $\mathcal{G}$ ,  $c(x)$  y  $\mathcal{S}$ ) es el definido en la Sección 8.1; a continuación se describen las reglas específicas que gobiernan la dinámica de este segundo modelo.

### 8.3.1. Campo de potencial

A diferencia del primer modelo, donde la exclusión de calles y agua se realizaba de forma implícita mediante el factor de peso  $w(c) = 0$  (ecuación 8.9), el segundo modelo adopta un mecanismo explícito y separado: las celdas de tipo calle ( $c = 1$ ) y agua ( $c = 3$ ) se agrupan en el conjunto de celdas prohibidas y se excluyen directamente del vecindario de movimiento  $\mathcal{N}_R(x)$  (ecuación 8.33), sin intervenir en el cálculo del campo  $\Pi$ .

$$\mathcal{C}_{\text{prhb}} = \{1, 3\}, \quad (8.22)$$

Gracias a esto, el campo de potencial puede prescindir del factor de ponderación por terreno y quedar definido únicamente en función de la proximidad a los sitios de cría mediante un kernel de decaimiento exponencial:

$$\Pi(x) = \frac{1}{\Pi_{\text{máx}}} \sum_{s \in \mathcal{S}} \exp\left(-\frac{\|x - s\|_2}{\lambda}\right), \quad \lambda = 10, \quad (8.23)$$

donde  $\Pi_{\max} = \max_{x \in \mathcal{G}} \Pi(x)$  es la constante de normalización que garantiza  $\Pi(x) \in [0, 1]$  para toda celda del dominio. El efecto de las celdas prohibidas no se refleja en  $\Pi$ , sino que queda delegado íntegramente al filtro de vecindad: ningún agente puede saltar a una celda  $y \in \mathcal{C}_{\text{prhb}}$ , independientemente de su valor de potencial. Cuando  $\Pi \equiv 0$  en toda la vecindad válida, el movimiento resulta puramente exploratorio, gobernado por  $\beta_0$ .

### 8.3.2. Descripción de los agentes

Cada agente se representa como el par  $a_i(t) = (s_i(t), x_i(t))$ , donde  $s_i$  es su **estado discreto** y  $x_i \in \mathcal{G}$  su **posición**. El espacio de estados es  $\{J, M, F_U, F_G, D\}$ :

- **J — Juvenil.** Inmóvil. Existe únicamente en sitios de cría. Puede morir o madurar a adulto.
- **M — Macho adulto.** Móvil. Muere a tasa  $\mu_M$ . Disponible para apareamiento con  $F_U$  en su vecindario.
- **$F_U$  — Hembra no apareada.** Móvil. Muere a tasa  $\mu_F$ . Transiciona a  $F_G$  al encontrar un macho, condicionado a su supervivencia en el paso actual.
- **$F_G$  — Hembra grávida.** Móvil. Muere a tasa  $\mu_F$ . En cada paso temporal, condicionado a su supervivencia, oviposta con probabilidad  $1 - e^{-f\Delta t}$  en el sitio de cría más cercano dentro del radio  $r_o$ , permaneciendo en el estado  $F_G$  tras cada evento.
- **D — Muerto.** Estado absorbente terminal. No ejecuta ninguna transición ulterior.

Los juveniles son *inmóviles*; los adultos ( $M, F_U, F_G$ ) realizan un salto estocástico en la retícula en cada paso temporal.

La Tabla 8.3 resume los parámetros del modelo con sus valores por defecto.

**Tabla 8.3:** Parámetros del modelo ABM y valores por defecto

| Símbolo    | Descripción                                         | Valor por defecto    |
|------------|-----------------------------------------------------|----------------------|
| $\mu_J$    | Tasa de mortalidad juvenil basal                    | 0,05 d <sup>-1</sup> |
| $\mu_M$    | Tasa de mortalidad de machos adultos                | 0,10 d <sup>-1</sup> |
| $\mu_F$    | Tasa de mortalidad de hembras                       | 0,08 d <sup>-1</sup> |
| $\gamma$   | Tasa de maduración larval a adulto                  | 0,07 d <sup>-1</sup> |
| $\alpha$   | Coefficiente de competencia larval densodependiente | 0,001                |
| $K_c$      | Capacidad de carga del vecindario de densidad       | 100                  |
| $\beta$    | Tasa de apareamiento por macho en radio $r_m$       | 0,02 d <sup>-1</sup> |
| $f$        | Tasa de oviposición de $F_G$                        | 5,0 d <sup>-1</sup>  |
| $r_m$      | Radio de búsqueda de machos para apareamiento       | 5 celdas             |
| $r_d$      | Radio del vecindario de densidad juvenil            | 3 celdas             |
| $r_o$      | Radio máximo de oviposición al sitio de cría        | 10 celdas            |
| $R$        | Radio máximo de salto adulto en la retícula         | 3 celdas             |
| $\beta_0$  | Sesgo exploratorio del movimiento                   | 10,0                 |
| $\Delta t$ | Paso de tiempo de la simulación                     | 1,0 día              |

**Fuente:** Elaboración propia.

### 8.3.3. Dinámica de juveniles

Los juveniles son **inmóviles** y existen únicamente en los sitios de cría  $S$ . Sea  $J_k(t)$  el número de juveniles asociados al sitio  $s_k$ . La evolución poblacional en cada paso temporal se obtiene a partir de eventos individuales de nacimiento, maduración y mortalidad evaluados estocásticamente para cada agente. Los eventos de maduración y mortalidad se tratan como **mutuamente excluyentes**: en cada paso temporal, un juvenil puede morir, o bien madurar, pero no ambas cosas simultáneamente. La ecuación de evolución es:

$$J_k(t + \Delta t) = J_k(t) + X_{\text{nac},k} - X_{\text{mat},k} - X_{\text{muerte},k} \quad (8.24)$$

donde  $X_{\text{nac},k}$ ,  $X_{\text{mat},k}$  y  $X_{\text{muerte},k}$  representan respectivamente el número de nacimientos, maduraciones y muertes ocurridas en el sitio  $s_k$  durante el intervalo

$\Delta t$ , con  $X_{\text{mat},k} + X_{\text{muerte},k} \leq J_k(t)$ .

En el nivel microscópico, cada transición se evalúa mediante un ensayo de Bernoulli independiente aplicado a cada agente. Dado que las probabilidades de transición permanecen constantes para todos los individuos de una misma clase durante cada paso temporal, los conteos agregados pueden representarse equivalentemente mediante distribuciones Binomiales. Esta equivalencia vale para  $X_{\text{mat},k}$  y  $X_{\text{muerte},k}$  porque todos los juveniles en  $s_k$  comparten la misma densidad local  $\rho_k$ , y para  $X_{\text{nac},k}$  porque la tasa de oviposición  $f$  es homogénea para todas las hembras grávidas en el vecindario del sitio.

**Nacimientos.** Sea  $F_G^{(k)}$  el número de hembras grávidas cuya distancia al sitio  $s_k$  es menor o igual a  $r_o$ . Cada una realiza un ensayo de Bernoulli independiente con probabilidad  $1 - e^{-f\Delta t}$  de ovipositar durante el intervalo  $\Delta t$ ; esta probabilidad es idéntica a la de la ecuación 8.32, dado que ambas expresiones describen el mismo proceso de Poisson individual, visto desde el nivel microscópico en la dinámica adulta y desde el nivel agregado aquí. Como  $f$  es homogéneo sobre todas las hembras grávidas del vecindario, el número total de nuevos juveniles nacidos en  $s_k$  sigue:

$$X_{\text{nac},k} \sim \text{Bin}\left(F_G^{(k)}, 1 - e^{-f\Delta t}\right) \quad (8.25)$$

**Maduración.** Cada juvenil presente en  $s_k$  realiza un ensayo de Bernoulli independiente con probabilidad de maduración determinada por la tasa  $\gamma$ . Bajo esta hipótesis homogénea, el número total de maduraciones sigue:

$$X_{\text{mat},k} \sim \text{Bin}\left(J_k(t), 1 - e^{-\gamma\Delta t}\right) \quad (8.26)$$

Cada juvenil que madura transiciona posteriormente a macho  $M$  o hembra no apareada  $F_U$  con probabilidad  $\frac{1}{2}$  para cada estado.

**Mortalidad densodependiente.** La mortalidad de los juveniles en  $s_k$  incorpora un término de competencia intraespecífica proporcional a la densidad local  $\rho_k$ . Cada juvenil presente en  $s_k$  realiza un ensayo de Bernoulli independiente con probabilidad de muerte dependiente de la densidad local:

$$X_{\text{muerte},k} \sim \text{Bin}(J_k(t), 1 - e^{-(\mu_J + \alpha \rho_k) \Delta t}), \quad \rho_k = \frac{J_k(t)}{K_c \cdot A_d}, \quad A_d = (2r_d + 1)^2 \quad (8.27)$$

donde  $A_d$  es el área del vecindario de densidad (en número de celdas) y  $K_c$  la capacidad de carga local.

### 8.3.4. Dinámica de adultos y movimiento estocástico

En cada paso  $\Delta t$ , todo adulto vivo ejecuta primero su **transición demográfica** y, si permanece vivo, realiza su **salto espacial** en la retícula. Cada transición se evalúa mediante ensayos de Bernoulli independientes aplicados sobre cada agente adulto.

**Macho adulto  $M$ .** El macho muere con probabilidad:

$$P(M \rightarrow D) = 1 - e^{-\mu_M \Delta t} \quad (8.28)$$

**Hembra no apareada  $F_U$ .** La hembra no apareada es evaluada primero por mortalidad. Si sobrevive, se evalúa si encuentra un macho en su vecindario de radio  $r_m$  y transiciona al estado grávido:

$$P(F_U \rightarrow D) = 1 - e^{-\mu_F \Delta t} \quad (8.29)$$

$$M_{r_m}(x_i) = \#\{j : s_j = M, \|x_j - x_i\|_2 \leq r_m\}, \quad \lambda_m = \beta \cdot M_{r_m}(x_i) \quad (8.30)$$

$$P(F_U \rightarrow F_G \mid \text{sobrevive}) = 1 - e^{-\lambda_m \Delta t} \quad (8.31)$$



donde  $M_{r_m}(x_i)$  es el número de machos dentro del radio de apareamiento y  $\lambda_m$  es la tasa de apareamiento efectiva. La probabilidad conjunta de supervivencia y apareamiento es por tanto  $e^{-\mu_F \Delta t} (1 - e^{-\lambda_m \Delta t})$ .

**Hembra grávida  $F_G$ .** La hembra grávida es evaluada primero por mortalidad. Si sobrevive, puede ovipositar en el mismo paso:

$$P(F_G \rightarrow D) = 1 - e^{-\mu_F \Delta t}, \quad P(\text{oviposita} \mid \text{sobrevive}) = 1 - e^{-f \Delta t} \quad (8.32)$$

**Movimiento estocástico en la retícula.** Todo adulto vivo que no muere en el paso actual realiza un salto estocástico. El vecindario de salto válido excluye las celdas prohibidas:

$$\mathcal{N}_R(x) = \{ y \in \mathcal{G} \mid \|y - x\|_2 \leq R, \ c(y) \notin \mathcal{C}_{\text{prhb}} \} \quad (8.33)$$

la nueva posición se muestrea de acuerdo con un núcleo de transición de Markov que favorece celdas cercanas a sitios de cría (a través del campo  $\Pi$ ) y penaliza distancias grandes:

$$P(y \mid x) = \frac{\tilde{w}(y \mid x)}{\sum_{z \in \mathcal{N}_R(x)} \tilde{w}(z \mid x)}, \quad \tilde{w}(y \mid x) = \frac{\Pi(y) + \beta_0}{\|y - x\|_2 + 1} \quad (8.34)$$

el término  $\beta_0$  en el numerador garantiza que el movimiento sea siempre exploratorio aunque  $\Pi(y) = 0$  en toda la vecindad, evitando distribuciones degeneradas.

### 8.3.5. Dinámica de apareamiento y generación de juveniles

El ciclo reproductivo conecta la dinámica adulta y la juvenil a través de tres eventos que ocurren secuencialmente en cada paso  $\Delta t$ :

**I. Encuentro.** La hembra  $F_U$  cuenta los machos  $M$  presentes en su vecindario de radio  $r_m$ . La tasa de apareamiento efectiva es  $\lambda_m = \beta \cdot M_{r_m}(x_i)$  (véase la ecuación 8.31).

**II. Transición de estado.** Con probabilidad  $P = 1 - e^{-\lambda_m \Delta t}$  la hembra transiciona  $F_U \rightarrow F_G$ , condicionado a que haya sobrevivido en el mismo paso. El evento se muestrea con una distribución de Bernoulli en cada paso.

**III. Oviposición y generación de juveniles.** Las hembras grávidas  $F_G$  que sobreviven al paso temporal pueden ovipositar con probabilidad  $P(\text{oviposita} \mid \text{sobrevive}) = 1 - e^{-f \Delta t}$ . Cuando el evento ocurre, la hembra genera un nuevo agente juvenil en el sitio de cría más cercano dentro del radio de oviposición  $r_o$ .

A nivel microscópico, la oviposición se implementa mediante ensayos de Bernoulli independientes aplicados sobre cada hembra grávida. Dado que  $f$  es homogéneo para todas las hembras grávidas del vecindario de  $s_k$ , el número agregado de nuevos juveniles en cada sitio de cría queda representado por la distribución Binomial definida en la ecuación 8.25.

La hembra permanece en el estado  $F_G$  tras ovipositar, pudiendo repetir el evento en pasos subsiguientes hasta su muerte. Este diseño colapsa el ciclo gonotrófico completo —ingesta de sangre, maduración de huevos y oviposición— en una única tasa efectiva  $f$ , constituyendo una simplificación biológica deliberada que podría extenderse en trabajos futuros mediante la incorporación de nuevos estados.

### 8.3.6. Orquestador temporal de la simulación

La definición del esquema de actualización constituye un componente fundamental del segundo modelo, ya que determina cómo se sincronizan las interacciones entre agentes y cómo se integran las dinámicas espaciales y demográ-

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

ficas dentro de cada paso temporal. En modelos basados en agentes, distintas estrategias de actualización pueden producir dinámicas emergentes diferentes incluso bajo los mismos parámetros biológicos.

El Algoritmo resume el flujo general implementado en la simulación.

**Algoritmo . Paso temporal de la simulación**

- 1: Mezclar aleatoriamente la lista de agentes
- 2: Reconstruir índice espacial del entorno
- 3: **for** cada agente  $a_i$  **do**
- 4:   **if**  $a_i \neq D$  **then**
- 5:     Ejecutar transición demográfica:
- 6:       mortalidad, maduración,
- 7:       apareamiento u oviposición
- 8:   **end if**
- 9: **end for**
- 10: Eliminar agentes muertos
- 11: **for** cada posición almacenada en el buffer de nacimientos **do**
- 12:   Crear nuevo agente juvenil  $J$
- 13: **end for**
- 14: Vaciar buffer de nacimientos
- 15: **for** cada agente adulto vivo **do**
- 16:   Ejecutar movimiento estocástico
- 17: **end for**
- 18: Actualizar tiempo:  $t \leftarrow t + \Delta t$
- 19: Registrar estadísticas poblacionales

Este esquema evita dos problemas frecuentes en simulaciones basadas en agentes. Por un lado, actualizar siempre los individuos en el mismo orden introduce sesgos computacionales artificiales, otorgando ventajas sistemáticas a los

primeros agentes procesados. Por otro, combinar transiciones demográficas y movimiento espacial dentro de un único ciclo individual genera inconsistencias temporales, ya que algunos agentes interactúan utilizando estados ya actualizados y otros estados correspondientes al instante previo.

Para evitar estos efectos, el simulador implementó un esquema de actualización híbrido sincronizado. En cada paso temporal, la lista de agentes se mezcla aleatoriamente; luego, todas las transiciones demográficas se resuelven utilizando el mismo estado espacial global del sistema y, únicamente después, se ejecuta el movimiento espacial de los agentes sobrevivientes. Este diseño preserva la consistencia entre la integración temporal y espacial del modelo y elimina dependencias artificiales asociadas al orden de actualización.

## Síntesis del segundo modelo

El modelo combina **heterogeneidad espacial urbana** derivada de datos geográficos reales, **interacciones estocásticas estrictamente locales** (mortalidad densodependiente, apareamiento por radio y oviposición condicionada a proximidad) y **movimiento probabilístico tipo Markov** guiado por el campo de potencial II (ecuación 8.23).

A diferencia del primer modelo, cuya dinámica demográfica depende del sistema ODE de [1], este segundo enfoque define las transiciones mediante reglas estocásticas locales independientes, lo que lo vuelve arquitectónicamente más escalable y extensible. La estructura de estados  $\{J, M, F_U, F_G\}$  permite incorporar nuevos comportamientos biológicos (e.g., búsqueda de sangre, diapausa o infección por *Wolbachia*) sin modificar el núcleo del simulador. Además, el paradigma de **localidad total** —toda interacción ocurre dentro de radios finitos— resulta computacionalmente favorable para dominios extensos y poblaciones densas.

No obstante, el modelo presenta aún varias limitaciones que constituyen líneas directas de extensión futura:

- **Dependencia climática de parámetros.** Las tasas  $\mu_J$ ,  $\mu_M$ ,  $\mu_F$ ,  $\gamma$ ,  $\beta$  y  $f$  permanecen constantes durante toda la simulación. Incorporar dependencias explícitas con la temperatura permitiría reproducir variaciones diurnas y estacionales de la dinámica poblacional.
- **Integración climática georreferenciada.** Variables como precipitación y humedad relativa podrían modificar dinámicamente la disponibilidad de sitios de cría y la mortalidad larval, permitiendo un entorno ambiental evolutivo más realista.
- **Ampliación del análisis de sensibilidad.** Como se verá en el Capítulo 9, el análisis global mediante LHS+PRCC identificó la mortalidad femenina y la dinámica de apareamiento como los parámetros más influyentes sobre la población final, mientras que la regulación densodependiente larval mostró baja influencia en el corto plazo. Como línea de trabajo futuro, resulta necesario ampliar el tamaño de muestra para obtener estimaciones estadísticamente más robustas.

Estas extensiones constituyen la hoja de ruta hacia un simulador ABM calibrado y validado, capaz de generar datos *in silico* comparables con observaciones epidemiológicas reales, línea de trabajo cuyos resultados preliminares fueron aceptados en el **XLV Congreso Nacional de Matemática Aplicada e Computacional** (CNMAC 2026).

## **CAPÍTULO 9**

### **Resultados**

Este Capítulo consolida los resultados experimentales y computacionales obtenidos en las fases finales del proyecto. Por parte de la trampa inteligente, se documenta la fase FDT4: la presentación del prototipo final ensamblado, los resultados de las pruebas de semi-campo en condiciones de laboratorio controladas y los resultados de la prueba de campo realizada en el Barrio Ycuá Satí, Asunción. Por parte del modelado, se presentan los resultados de las simulaciones exploratorias de los modelos ABM correspondientes a las fases FDM3 y FDM4, descritos en el Capítulo 8, incluyendo la dinámica poblacional temporal, la evolución espacial de los agentes y el análisis estadístico sobre múltiples corridas independientes. Las implicaciones de estos resultados se discuten en el Capítulo 10.

#### **9.1. Montaje Final de la Trampa**

En esta sección se presenta el prototipo final funcional de la trampa inteligente desarrollada. Asimismo, se documentan los resultados obtenidos durante las pruebas experimentales de semi-campo y campo.

##### **9.1.1. Prototipo Final de la trampa**

Las Figuras 9.1–9.5 presentan el prototipo ensamblado en sus distintas configuraciones: la disposición interna de los componentes, el sistema cerrado en

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

operación normal, el mecanismo de clasificación rotativo, los módulos electrónicos integrados y el cebo biológico empleado para la atracción de mosquitos.



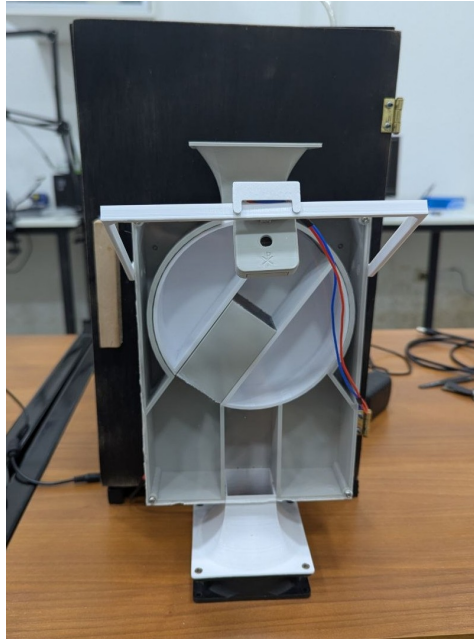
**Figura 9.1:** Trampa en la caja protectora con la puerta abierta, mostrando la disposición interna de los componentes. **Fuente:** Elaboración propia.



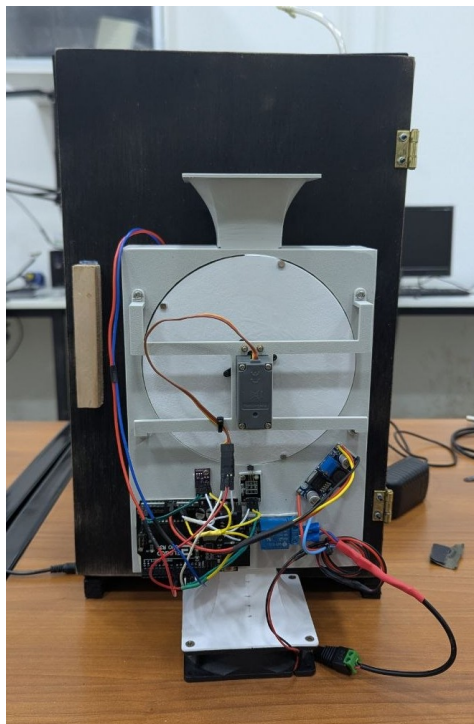
**Figura 9.2:** Trampa en la caja protectora con la puerta cerrada durante su operación normal. **Fuente:** Elaboración propia.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

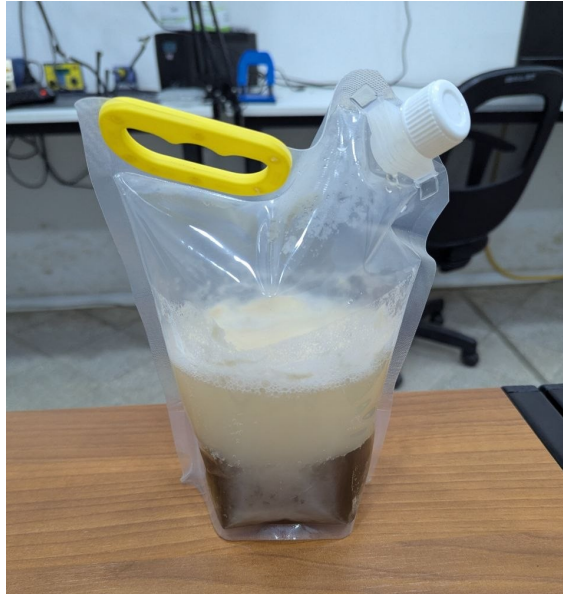


**Figura 9.3:** Mecanismo de clasificación rotativo encargado de la separación de los mosquitos detectados. **Fuente:** Elaboración propia.



**Figura 9.4:** Módulos electrónicos de control, procesamiento y comunicación integrados en la trampa, incluyendo los circuitos de alimentación, control de actuadores y el computador embebido para la ejecución del modelo de visión artificial. **Fuente:** Elaboración propia.





**Figura 9.5:** Cebo utilizado para la atracción de mosquitos, dispuesto en su contenedor dentro de la trampa. **Fuente:** Elaboración propia.

La Figura 9.6 muestra el funcionamiento del sistema durante una prueba experimental con un espécimen de prueba. En esta etapa se verifica la operación conjunta de los módulos de captura, detección y clasificación. El monitor conectado al Raspberry Pi permitió visualizar en tiempo real la salida del algoritmo de visión artificial, incluyendo la detección y clasificación del mosquito procesado. Esta prueba permitió validar la sincronización entre el procesamiento de imágenes y el accionamiento mecánico del sistema de clasificación.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS



**Figura 9.6:** Trampa funcionando con un mosquito de prueba. El monitor conectado al Raspberry Pi permitió visualizar la detección en tiempo real. **Fuente:** Elaboración propia.

### 9.1.2. Pruebas de Semi-Campo

A continuación se describe la metodología empleada en las pruebas de semi-campo, incluyendo las condiciones del entorno, los sujetos de prueba y el protocolo seguido. Ambas pruebas se llevaron a cabo bajo condiciones idénticas de entorno, sujetos y protocolo experimental, por lo que la descripción metodológica se aplica en su totalidad a cada una de ellas.

#### 9.1.2.1. Preparación del Entorno y Período de Prueba

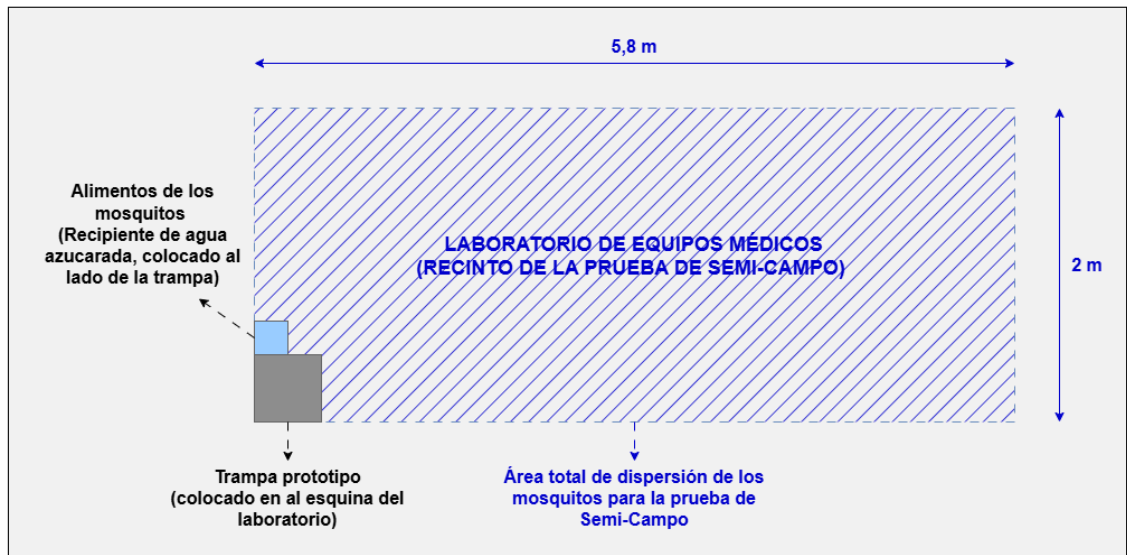
- **Recinto:** Laboratorio de Equipos Médicos de la FIUNA, San Lorenzo, Paraguay, con una extensión de  $5,8\text{ m} \times 2\text{ m}$ .

- **Condiciones ambientales:** El recinto permaneció sellado y climatizado durante toda la prueba, con temperatura de 21 a 25 °C y humedad relativa de 40–60 %.
- **Ubicación de la trampa:** La trampa se colocó en una esquina del recinto; junto a ella se dispuso agua azucarada como fuente de alimentación para los mosquitos, con el fin de mantenerlos en cercanía al dispositivo.
- **Períodos de prueba:** Cada ensayo tuvo una duración de 36 horas. Los períodos específicos de cada prueba se detallan en la Tabla 9.1.

**Tabla 9.1:** Períodos de realización de las pruebas de semi-campo.

| Prueba  | Inicio                                 | Fin                                  |
|---------|----------------------------------------|--------------------------------------|
| Primera | 22:00 h — viernes 8 de mayo de 2026    | 10:00 h — domingo 10 de mayo de 2026 |
| Segunda | 22:00 h — miércoles 13 de mayo de 2026 | 10:00 h — viernes 15 de mayo de 2026 |

**Fuente:** Elaboración propia.



**Figura 9.7:** Esquema del recinto utilizado para las pruebas de semi-campo, con la disposición de la trampa y el área de liberación de los mosquitos. **Fuente:** Elaboración propia.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---



**Figura 9.8:** Vista del recinto de experimentación con la trampa colocada en una esquina del laboratorio durante la prueba de semi-campo. **Fuente:** Elaboración propia.



**Figura 9.9:** Trampa posicionada en la esquina del recinto de experimentación junto al agua azucarada dispuesta como fuente de alimentación. **Fuente:** Elaboración propia.



### 9.1.2.2. Sujetos de Prueba

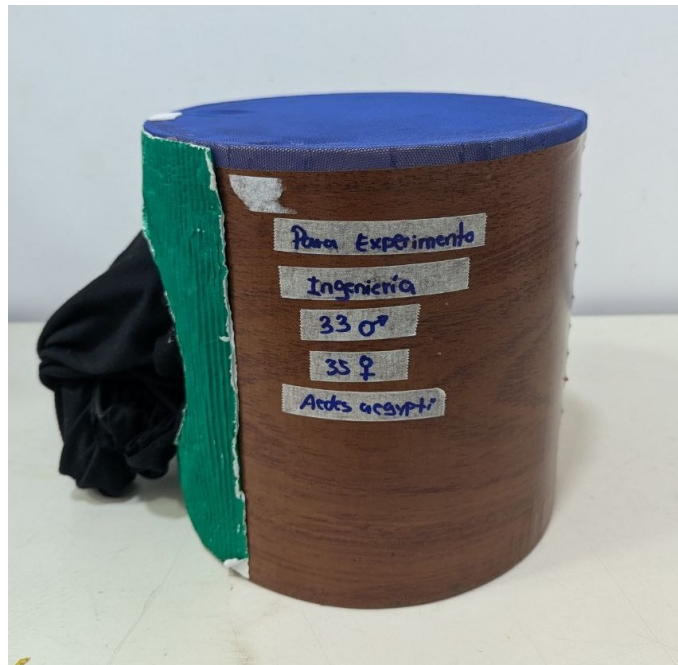
Los sujetos de prueba fueron mosquitos donados por el IICS, todos de la especie *Aedes aegypti*, con edades comprendidas entre 3 y 7 días. La distribución por sexo empleada en cada prueba se detalla en la Tabla 9.2.

**Tabla 9.2:** Distribución de los sujetos de prueba por sexo.

| Sexo         | Cantidad (individuos) |
|--------------|-----------------------|
| Macho        | 33                    |
| Hembra       | 33                    |
| <b>Total</b> | <b>66</b>             |

**Fuente:** Elaboración propia.

Los mosquitos fueron entregados mezclados en un contenedor con agua azucarada para mantener su alimentación previa a la realización de cada experimento.



**Figura 9.10:** Mosquitos de la especie *Aedes aegypti* proporcionados por el IICS para las pruebas de semi-campo, en el contenedor de transporte con agua azucarada. **Fuente:** Elaboración propia.

### 9.1.2.3. Protocolo del Experimento

Para la realización de cada prueba de semi-campo se adoptó el siguiente procedimiento:

1. Retirar el alimento (agua azucarada) del contenedor de mosquitos cuatro horas antes de su liberación, con el fin de garantizar que los individuos se encontrasen en estado de hambre al momento del experimento.
2. Liberar los mosquitos en el recinto y aguardar 30 minutos antes de activar la trampa, permitiendo así su dispersión espacial por el laboratorio.
3. Activar la trampa y mantenerla en operación durante el período de prueba establecido, registrando las detecciones de forma automática en la base de datos.
4. Al finalizar el período de prueba, abrir los compartimientos de la trampa, realizar el conteo manual de los mosquitos capturados y comparar los resultados con los registros del sistema automatizado para cuantificar el error de clasificación.

### 9.1.2.4. Definición de Ratios de Precisión y Eficiencia

Para evaluar el desempeño del sistema en cada prueba de forma homogénea y comparable, se definen a continuación los indicadores de rendimiento empleados en el análisis de resultados.

**Población efectiva ( $N_{eff}$ ).** No todos los especímenes liberados participan activamente en el ensayo: algunos individuos permanecen inactivos dentro del contenedor de liberación sin ingresar al recinto. Para garantizar que los ratios de eficiencia reflejen únicamente la fracción de la muestra efectivamente expuesta a la trampa, se define la **población efectiva** como:

$$N_{eff} = N_{total} - N_{inactivos} \quad (9.1)$$

donde  $N_{total}$  es el número de especímenes liberados y  $N_{inactivos}$  los que no salieron del contenedor al inicio del experimento.

**Precisión del algoritmo de clasificación ( $P_{class}$ ).** Mide la proporción de individuos correctamente clasificados por el modelo de visión artificial ONNX sobre el total de mosquitos procesados por el sistema:

$$P_{class} = \frac{\text{Clasificaciones correctas}}{\text{Total de individuos procesados}} \times 100 \% \quad (9.2)$$

Un acierto implica que el sexo asignado por el modelo coincide con el sexo real del espécimen verificado en el conteo manual post-ensayo.

**Ratio de retención física ( $R_{ret}$ ).** Evalúa la eficacia mecánica del sistema para retener físicamente en los compartimentos a los individuos detectados. Un valor inferior al 100 % indica escapes o fallas en el mecanismo de clasificación:

$$R_{ret} = \frac{\text{Mosquitos retenidos en trampa}}{\text{Mosquitos detectados por el sistema}} \times 100 \% \quad (9.3)$$

**Ratio de captura ( $R_c$ ).** Expresa qué fracción de la población efectiva fue capturada físicamente, combinando el desempeño del sistema de detección y el del mecanismo físico:

$$R_c = \frac{\text{Mosquitos capturados en trampa}}{N_{eff}} \times 100 \% \quad (9.4)$$

### 9.1.2.5. Resultados de la Primera Prueba de Semi-Campo

**Balance de la muestra entomológica.** Se inició el experimento con una población total de  $N = 66$  especímenes. La Tabla 9.3 presenta el balance de la muestra, descontando los individuos que permanecieron inactivos en el contenedor de liberación.

**Tabla 9.3:** Balance de especímenes durante la primera prueba de semi-campo.

| Estado                                           | Hembras ( $\varphi$ ) | Machos ( $\sigma$ ) | Total     |
|--------------------------------------------------|-----------------------|---------------------|-----------|
| Liberados inicialmente                           | 33                    | 33                  | 66        |
| Inactivos (permanecieron en contenedor)          | 2                     | 4                   | 6         |
| <b>Población Efectiva (<math>N_{eff}</math>)</b> | <b>31</b>             | <b>29</b>           | <b>60</b> |

**Fuente:** Elaboración propia.

**Detección y clasificación.** La Tabla 9.4 presenta el contraste entre las detecciones realizadas por la IA (modelo ONNX) y la realidad biológica de los especímenes procesados. Se procesaron un total de 16 mosquitos durante el tiempo operativo del sistema.

**Tabla 9.4:** Comparativa entre Detección IA y Realidad Biológica ( $N = 16$ ) — Primera prueba.

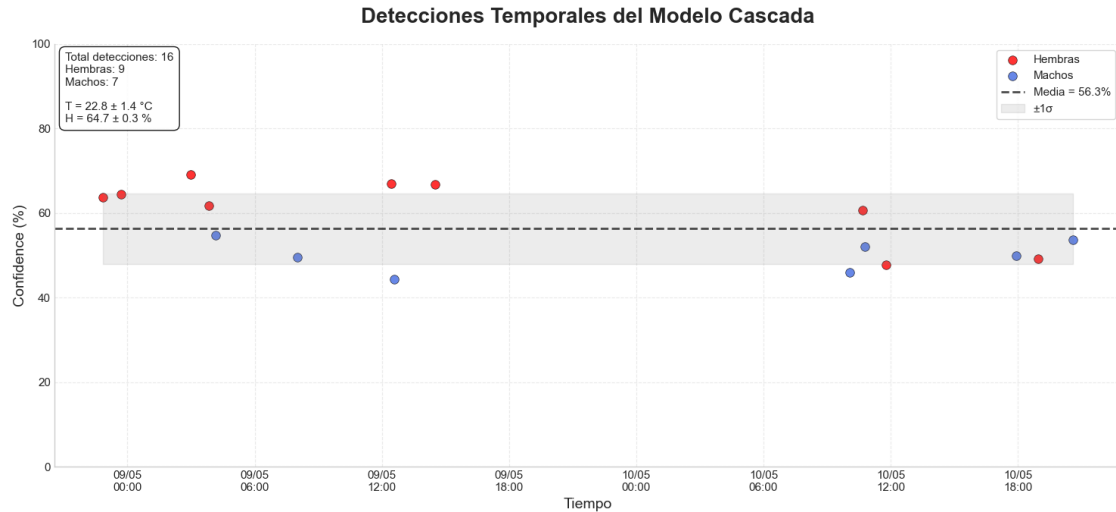
| Sexo                  | Detección IA | Realidad Total | En Trampa (Físico) | Escapes  |
|-----------------------|--------------|----------------|--------------------|----------|
| Hembras ( $\varphi$ ) | 9            | 10             | 8                  | 2        |
| Machos ( $\sigma$ )   | 7            | 6              | 6                  | 0        |
| <b>Total</b>          | <b>16</b>    | <b>16</b>      | <b>14</b>          | <b>2</b> |

**Fuente:** Elaboración propia.

**Dinámica temporal de detecciones.** La Figura 9.11 muestra la distribución temporal de las detecciones registradas automáticamente por el sistema IoT a lo largo del período de la primera prueba.



**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**



**Figura 9.11:** Distribución temporal de las detecciones registradas durante la primera prueba de semi-campo (36 horas). **Fuente:** Elaboración propia.

**Ratios de desempeño.** Aplicando las definiciones de la Sección 9.1.2.4 con los datos anteriores, se obtienen los siguientes indicadores para la primera prueba:

$$P_{class} = \frac{15}{16} \times 100 = 93,75 \% \quad (9.5)$$

$$R_{ret} = \frac{14}{16} \times 100 = 87,5 \% \quad (9.6)$$

$$R_c = \frac{16}{60} \times 100 \approx 26,66 \% \quad (9.7)$$

#### 9.1.2.6. Resultados de la Segunda Prueba de Semi-Campo

**Balance de la muestra entomológica.** La Tabla 9.5 presenta el balance de la muestra entomológica para la segunda prueba de semi-campo, realizada con la misma población inicial declarada en la Tabla 9.2 ( $N = 66$ ).

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

**Tabla 9.5:** Balance de especímenes durante la segunda prueba de semi-campo.

| Estado                                           | Hembras (♀) | Machos (♂) | Total     |
|--------------------------------------------------|-------------|------------|-----------|
| Liberados inicialmente                           | 33          | 33         | 66        |
| Inactivos (permanecieron en contenedor)          | 5           | 6          | 11        |
| <b>Población Efectiva (<math>N_{eff}</math>)</b> | <b>28</b>   | <b>27</b>  | <b>55</b> |

**Fuente:** Elaboración propia.

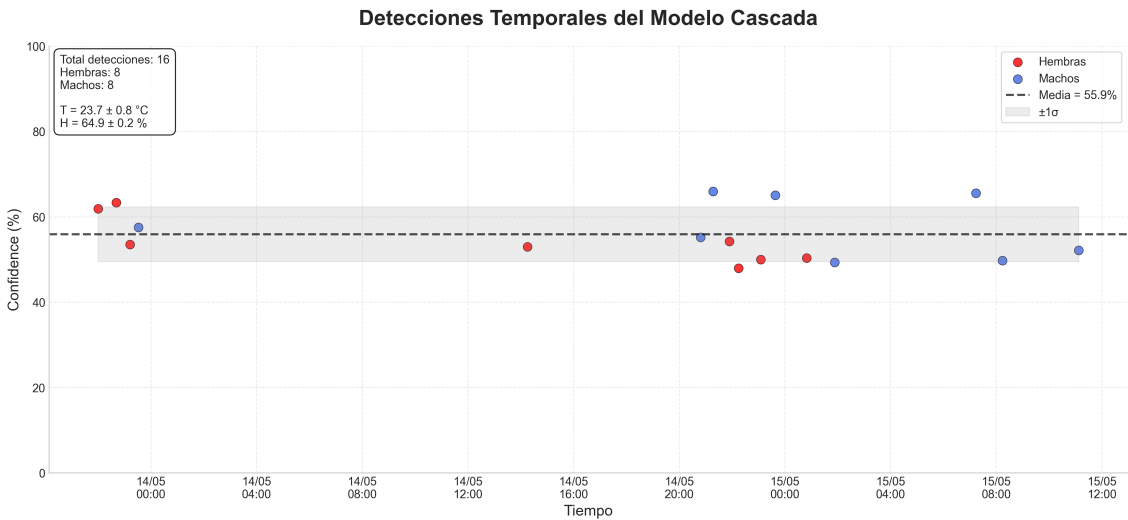
**Detección y clasificación.** La Tabla 9.6 presenta el contraste entre las detecciones realizadas por la IA y la realidad biológica para la segunda prueba.

**Tabla 9.6:** Comparativa entre Detección IA y Realidad Biológica — Segunda prueba.

| Sexo         | Detección IA | Realidad Total | En Trampa (Físico) | Escapes  |
|--------------|--------------|----------------|--------------------|----------|
| Hembras (♀)  | 8            | 7              | 5                  | 2        |
| Machos (♂)   | 8            | 9              | 9                  | 0        |
| <b>Total</b> | <b>16</b>    | <b>16</b>      | <b>14</b>          | <b>2</b> |

**Fuente:** Elaboración propia.

**Dinámica temporal de detecciones.** La Figura 9.12 muestra la distribución temporal de las detecciones registradas durante la segunda prueba.



**Figura 9.12:** Distribución temporal de las detecciones registradas durante la segunda prueba de semi-campo (36 horas). **Fuente:** Elaboración propia.

**Ratios de desempeño.** Aplicando las definiciones de la Sección 9.1.2.4:

$$P_{class} = \frac{15}{16} \times 100 = 93,75 \% \quad (9.8)$$

$$R_{ret} = \frac{14}{16} \times 100 = 87,5 \% \quad (9.9)$$

$$R_c = \frac{16}{55} \times 100 \approx 29,09 \% \quad (9.10)$$

### 9.1.2.7. Síntesis comparativa de ratios

**Tabla 9.7:** Comparación de ratios de desempeño entre ambas pruebas de semi-campo.

| Indicador   | Primera prueba | Segunda prueba | Variación |
|-------------|----------------|----------------|-----------|
| $P_{class}$ | 93,75 %        | 93,75 %        | 0,0 %     |
| $R_{ret}$   | 87,5 %         | 87,5 %         | 0,0 %     |
| $R_c$       | 26,66 %        | 29,09 %        | +2,43 %   |

**Fuente:** Elaboración propia.

### 9.1.2.8. Discusión de Resultados y Fallas Identificadas

Los resultados obtenidos en ambas pruebas de semi-campo evidencian un desempeño consistente del sistema, tanto en el módulo de visión artificial como en el mecanismo físico de clasificación. En ambas pruebas se obtuvo una precisión de clasificación ( $P_{class}$ ) de 93,75 %, lo que demuestra la capacidad del modelo ONNX para diferenciar correctamente entre machos y hembras bajo condiciones reales de operación.

Asimismo, el ratio de retención física ( $R_{ret}$ ) alcanzó un valor de 87,5 % en ambas pruebas, indicando que la mayoría de los individuos detectados fueron retenidos correctamente dentro de la trampa. Por otra parte, el ratio de captura ( $R_c$ )

se mantuvo entre 26,66 % y 29,09 %, mostrando que una fracción moderada de la población efectiva interactuó con el sistema durante el período experimental.

Las detecciones temporales registradas demostraron además la capacidad del sistema IoT para operar de forma continua y almacenar eventos en tiempo real durante las 36 horas de ensayo.

### **Fallas identificadas.**

1. **Confusión en la clasificación por sexo:** Se identificó una pequeña confusión entre hembras y machos en ambas pruebas de semi-campo. En particular, se observó un caso en el que una hembra fue clasificada como macho por el modelo ONNX. Este error puede atribuirse a oclusiones parciales o a una velocidad de desplazamiento superior a la tasa de refresco de la cámara durante el tránsito por el canal de succión.
2. **Falla de estanqueidad mecánica:** Los escapes registrados correspondieron exclusivamente a hembras. La inspección post-ensayo permitió identificar una holgura entre el cilindro giratorio y la placa de acrílico, utilizada como vía de escape debido al reducido tamaño morfológico de los especímenes.
3. **Limitaciones de captura:** Una parte importante de la población efectiva no ingresó a la trampa durante el ensayo, posiblemente debido a factores biológicos y a la ausencia de atrayentes específicos. Esto sugiere la necesidad de incorporar futuros mecanismos de atracción para incrementar la interacción mosquito-trampa.

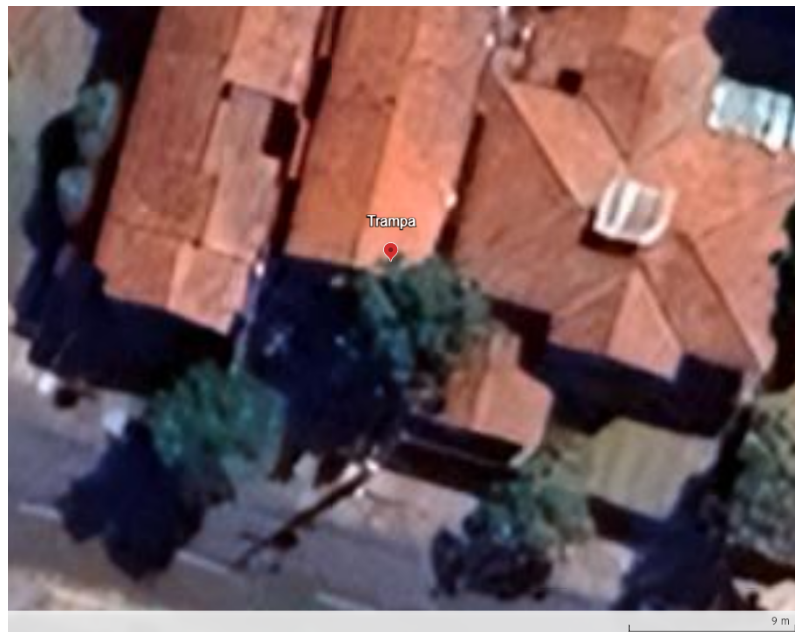
#### **9.1.3. Prueba de Campo**

Complementariamente a las pruebas de semi-campo, se realizó una validación en condiciones reales de campo con el prototipo desplegado en un entorno

urbano. A continuación se describen la localización, las condiciones experimentales y los resultados obtenidos durante este ensayo.

#### 9.1.3.1. Localización y Condiciones Experimentales

La fase de validación en campo se llevó a cabo durante un periodo de 6 días consecutivos, desde el viernes 1 de mayo (07:00 h) hasta el jueves 7 de mayo (07:00 h). El dispositivo se ubicó en una residencia situada en el Barrio Ycuá Satí, Asunción, posicionándose en el patio delantero a 8 metros de la calzada asfáltica. Para proteger la integridad del hardware, el prototipo se instaló adosado a la pared frontal bajo un alero de protección contra la precipitación.



**Figura 9.13:** Localización satelital del punto de muestreo en el Barrio Ycuá Satí.  
**Fuente:** Google Maps (2026).

#### 9.1.3.2. Resultados y Variabilidad Climática

Durante el ensayo se observaron dos periodos climáticos diferenciados que incidieron directamente en la tasa de captura. En la Tabla 9.8 se detallan los resultados diarios.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

**Tabla 9.8:** Resultados de captura diaria y condiciones ambientales durante la prueba de campo (1–7 de mayo de 2026).

| Día          | Hembras   | Machos   | Otros                    | Temp. (°C)        | Hum. (%)    |
|--------------|-----------|----------|--------------------------|-------------------|-------------|
| 1            | 4         | 1        | -                        | 29 – 30           | 84 %        |
| 2            | 5         | 0        | 1 ( <i>Geometridae</i> ) | 29 – 30           | 82 %        |
| 3            | 3         | 1*       | -                        | 29 – 30           | 85 %        |
| 4            | 2 (1*)    | 0        | -                        | 18 – 20           | 63 %        |
| 5            | 0         | 1        | -                        | 18 – 20           | 59 %        |
| 6            | 2         | 2        | -                        | 18 – 20           | 60 %        |
| <b>Total</b> | <b>16</b> | <b>5</b> | <b>1</b>                 | <b>Prom: 24,5</b> | <b>72 %</b> |

(\*) Indica detección identificada como falso positivo.

**Fuente:** Elaboración propia.

La Figura 9.14 muestra el estado del dashboard de monitoreo durante el período de la prueba de campo, consultado en tiempo casi real desde la aplicación web desplegada en la nube.

|          |                     |              |       |      |                                                                                       |
|----------|---------------------|--------------|-------|------|---------------------------------------------------------------------------------------|
| rpi_cam1 | 2026-05-06 20:33:02 | Female 45.5% | 31.1  | 74.8 |  |
| rpi_cam1 | 2026-05-06 19:58:59 | Female 53.5% | 30.16 | 74.7 |  |
| rpi_cam1 | 2026-05-06 17:54:17 | Male 51.0%   | 29.8  | 74.6 |  |
| rpi_cam1 | 2026-05-06 05:37:50 | Male 57.0%   | 27    | 74.1 |  |

**Figura 9.14:** Dashboard de monitoreo IoT durante la prueba de campo (1–7 de mayo), mostrando detecciones automáticas con clasificación por sexo, variables ambientales (temperatura y humedad, sensor BME280) e imagen asociada a cada evento de detección. **Fuente:** Elaboración propia.

### 9.1.3.3. Discusión de la Prueba de Campo

- **Efecto Climático:** Se observó una correlación directa entre el descenso de temperatura (de 30 °C a 18 °C) y la reducción en la frecuencia de capturas de hembras. Esto es consistente con la biología del vector, cuya actividad de vuelo disminuye en condiciones de frío y baja humedad.
- **Selectividad y Captura Accidental:** El día 2 se registró la entrada de un espécimen de la familia *Geometridae* (oruga medidora). El sistema procesó al individuo sin generar una clasificación de mosquito. Este espécimen fue excluido del cómputo de precisión por no constituir una detección de mosquito.
- **Falsos Positivos:** Se documentaron 2 falsos positivos sobre un total de 21 detecciones de mosquito (16 hembras + 5 machos), verificadas en la inspección post-ensayo. La precisión de campo se calcula como:

$$P_{\text{campo}} = \frac{21 - 2}{21} \times 100 = \frac{19}{21} \times 100 \approx 90,5 \% \quad (9.11)$$

un margen aceptable considerando las condiciones variables de iluminación y turbulencia de aire en exteriores.

En conjunto, la prueba de campo demostró la viabilidad del despliegue autónomo del sistema en condiciones reales: el prototipo operó de forma continua durante 6 días, transmitió datos de forma automática a la infraestructura en la nube y respondió a las variaciones climáticas del entorno de manera observable y consistente con la biología del vector. La precisión de campo obtenida (90,5 %) —lograda sin ajuste de hiperparámetros ni reentrenamiento para el entorno exterior— ratifica la robustez del pipeline en cascada y sienta una base empírica sólida para las recomendaciones de mejora discutidas en el Capítulo 10.

## 9.2. Modelado Basado en Agentes

Los resultados presentados en esta sección corresponden a simulaciones de carácter exploratorio. Los parámetros biológicos empleados no han sido calibrados con datos experimentales reales; en consecuencia, los valores numéricos obtenidos —tamaños poblacionales, tiempos de equilibrio, distribuciones espaciales— no deben interpretarse como predicciones cuantitativas del sistema real. El propósito de estas corridas fue verificar la coherencia interna de los modelos, ilustrar la dinámica emergente bajo los parámetros propuestos y establecer la base computacional para una calibración rigurosa en trabajos futuros.

### 9.2.1. Simulación y Resultados con el Primer Modelo

Los resultados presentados en esta sección corresponden a la primera versión del simulador ABM, cuyos fundamentos matemáticos se describen en la Sección 8.2.

#### 9.2.1.1. Entorno de simulación

El entorno utilizado para esta corrida consiste en una grilla urbana discreta generada mediante el pipeline descrito en el Capítulo 7, aplicado a una zona geográfica diferente a la del caso de estudio principal. Los parámetros de entrada empleados se detallan en la Tabla 9.9.

**Tabla 9.9:** Parámetros de entrada del Grid Map — Primer modelo

| Símbolo     | Descripción            | Valor empleado | Unidad      |
|-------------|------------------------|----------------|-------------|
| $\varphi_c$ | Latitud del centroide  | −25,2968       | °           |
| $\lambda_c$ | Longitud del centroide | −57,6534       | °           |
| $L$         | Extensión del mapa     | 2600           | m           |
| $\Delta s$  | Tamaño de celda        | 4              | m/celda     |
| $N$         | Dimensión de la grilla | 650            | celdas/lado |

**Fuente:** Elaboración propia.



Con estos parámetros se obtuvo una grilla de  $650 \times 650$  celdas que cubre aproximadamente  $2,6 \times 2,6 \text{ km}^2$ . El tamaño de celda  $\Delta s = 4 \text{ m}$  fue seleccionado como un compromiso entre la representación realista del desplazamiento por paso de los agentes —que, al tratarse de mosquitos, debería ser reducido para reflejar la movilidad limitada del vector— y el costo computacional, que crece cuadráticamente con la reducción de  $\Delta s$  sobre un área fija. A  $4 \text{ m}$  por celda, el desplazamiento máximo por paso es del orden de pocos metros, valor aceptable para una primera exploración computacional del modelo. Sobre este entorno se distribuyeron **15 sitios de cría sintéticos** de forma aleatoria, ubicados exclusivamente sobre celdas de vegetación ( $c(x) = 2$ ).

### 9.2.1.2. Condiciones iniciales

Los agentes y sitios de cría fueron posicionados aleatoriamente en la grilla al inicio de cada corrida. La Tabla 9.10 resume los parámetros de inicialización de la simulación.

**Tabla 9.10:** Condiciones iniciales y parámetros de simulación — Primer modelo

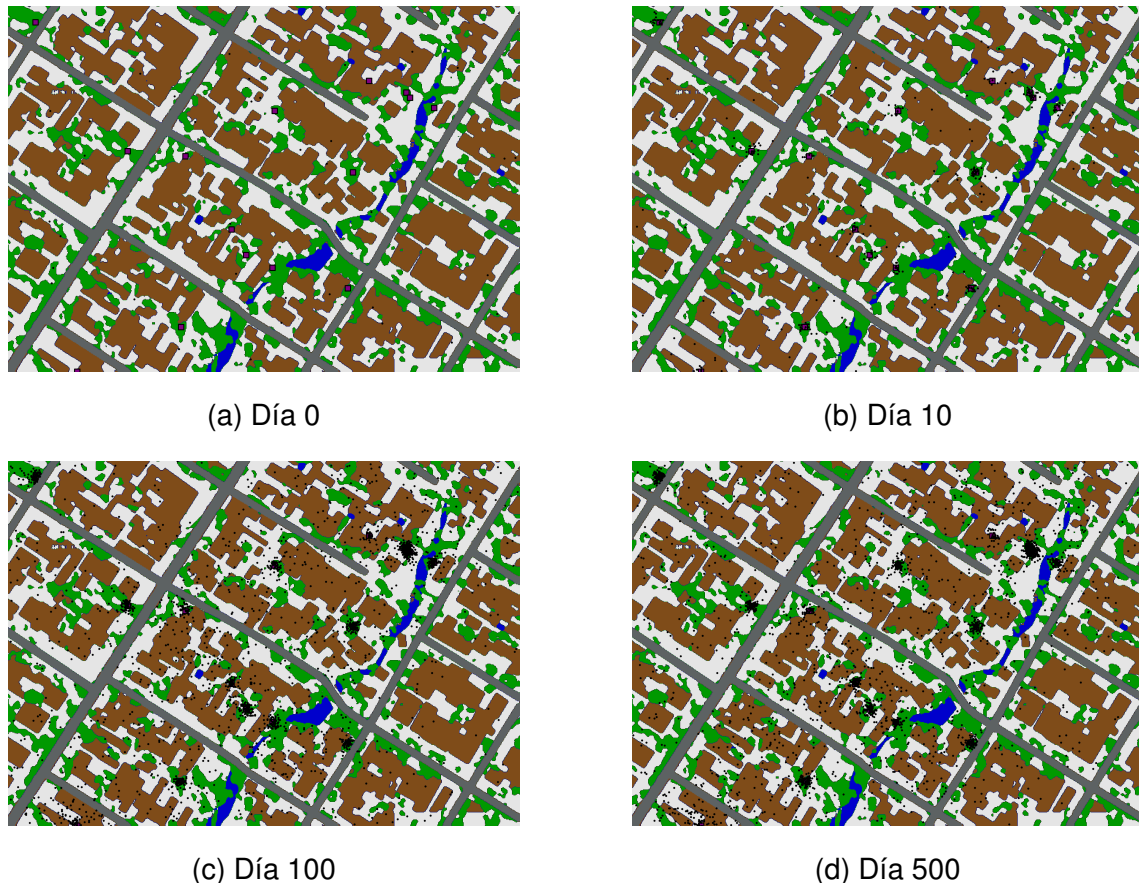
| Símbolo    | Descripción                  | Valor | Unidad     |
|------------|------------------------------|-------|------------|
| $J_0$      | Juveniles iniciales          | 60    | individuos |
| $A_0$      | Adultos iniciales            | 40    | individuos |
| $\Delta t$ | Paso de tiempo               | 0,1   | días       |
| $K_s$      | Capacidad de carga por sitio | 100   | ind/sitio  |
| $m$        | Número de sitios de cría     | 15    | sitios     |

**Fuente:** Elaboración propia.

### 9.2.1.3. Evolución espacial de los agentes

La Figura 9.15 presenta capturas del estado del sistema en cuatro instantes representativos de la simulación. En el día 0, los agentes se encuentran distribuidos aleatoriamente sobre la grilla. A partir del día 10 comienza a evidenciarse la atracción de los adultos hacia los sitios de cría, guiada por el campo de potencial

$\Pi(x)$  (ecuación 8.7). En el día 100 la distribución espacial se ha estabilizado, con concentraciones notorias en las proximidades de los sitios de mayor potencial. En el día 500 el patrón persiste, reflejando el equilibrio dinámico entre reproducción, maduración y mortalidad que caracteriza al sistema en régimen estacionario.

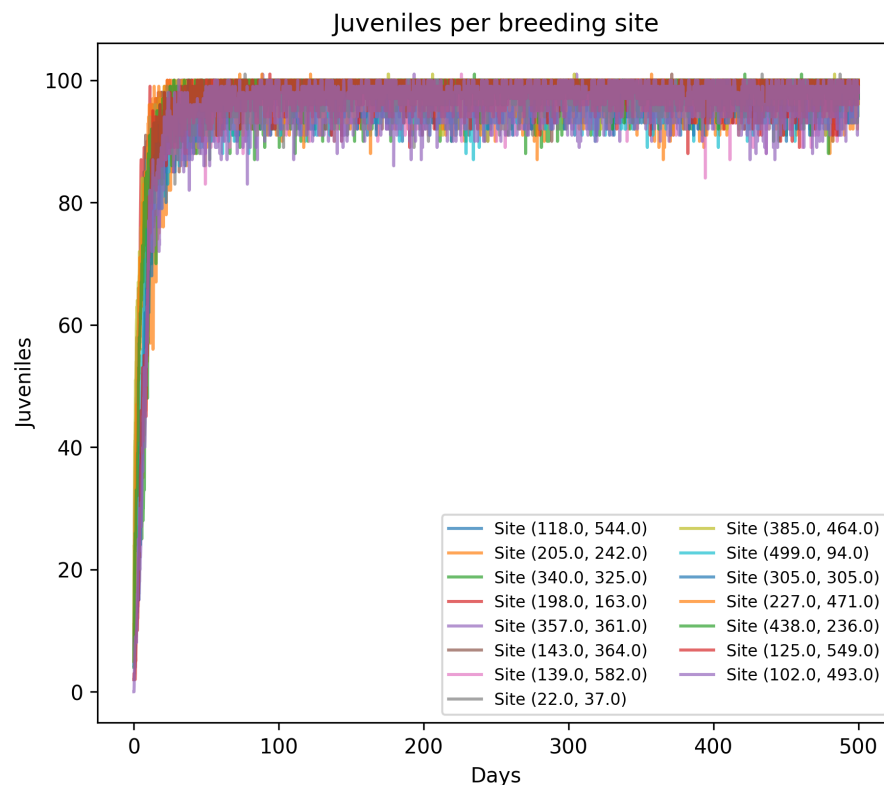


**Figura 9.15:** Capturas de la simulación en cuatro instantes temporales representativos. Los cuadrados magenta representan los 15 sitios de cría sintéticos; los puntos negros indican la posición de los adultos en cada instante. **Fuente:** Elaboración propia.

#### 9.2.1.4. Dinámica de juveniles por sitio de cría

La Figura 9.16 muestra la evolución de la población de juveniles en cada uno de los quince sitios de cría a lo largo de una corrida completa (5 000 pasos de tiempo, donde un día equivale a 10 pasos de simulación). Se observa que la distribución de juveniles entre sitios es heterogénea: algunos acumulan consistente-

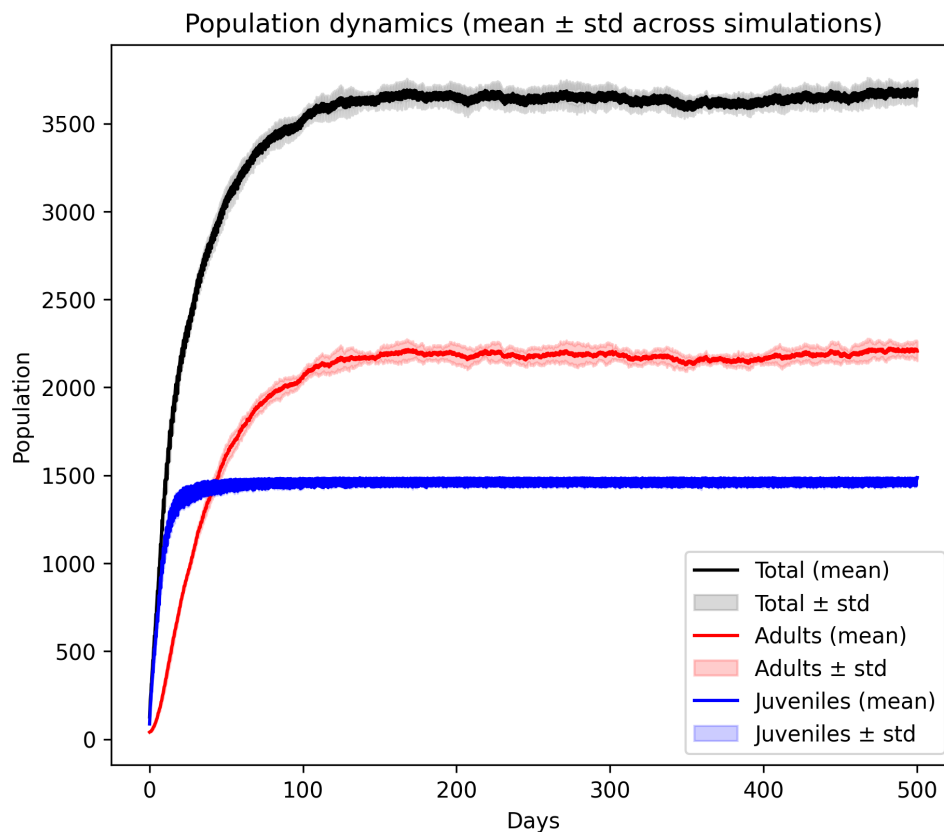
mente mayor cantidad de individuos, lo que refleja diferencias en su accesibilidad espacial desde las posiciones habituales de los adultos, moduladas por el campo de potencial  $\Pi(x)$  y la función de asignación  $\phi_s(X_i(t))$  (ecuación 8.17). Esta heterogeneidad surge de forma emergente, sin estar codificada explícitamente en las reglas del modelo.



**Figura 9.16:** Evolución de la población de juveniles en los 15 sitios de cría a lo largo de 5 000 pasos de tiempo. Las coordenadas (en celdas) de cada sitio se indican en la leyenda. **Fuente:** Elaboración propia.

#### 9.2.1.5. Estadísticas poblacionales sobre múltiples corridas

La Figura 9.17 presenta la media y la desviación estándar de la dinámica poblacional total (juveniles + adultos) a lo largo del tiempo, calculadas sobre seis corridas independientes con agentes y sitios de cría distribuidos aleatoriamente en cada ejecución. La banda de incertidumbre (media  $\pm$  DE) permite evaluar la sensibilidad del modelo frente a las condiciones iniciales estocásticas.



**Figura 9.17:** Media  $\pm$  desviación estándar de la dinámica poblacional global a lo largo del tiempo, calculadas sobre seis corridas independientes con agentes y sitios de cría distribuidos aleatoriamente en cada ejecución. **Fuente:** Elaboración propia.

En conjunto, los resultados del primer modelo evidencian que la incorporación de un campo de potencial georreferenciado genera, de forma emergente, distribuciones espaciales no uniformes coherentes con la biología del mosquito: los adultos tienden a concentrarse en las proximidades de los sitios de cría sin que este comportamiento esté codificado explícitamente en las reglas del modelo. La convergencia de las trayectorias poblacionales a partir de distintas condiciones iniciales estocásticas indica que el comportamiento macroscópico del sistema es robusto, propiedad necesaria para que el modelo pueda ser calibrado y comparado con datos de campo en trabajos futuros.

## 9.2.2. Simulación y Resultados con el Segundo Modelo

Los resultados presentados en esta sección corresponden al segundo modelo ABM, cuyos fundamentos matemáticos se describen en la Sección 8.3.

### 9.2.2.1. Entorno de simulación

El entorno empleado es el Grid Map georreferenciado generado a partir de datos de OpenStreetMap para el barrio de estudio en Asunción, Paraguay, cuya construcción se detalla en el Capítulo 7. Los parámetros de entrada utilizados corresponden a los de la Tabla 7.1, reproducidos en la Tabla 9.11 para facilitar la lectura.

**Tabla 9.11:** Parámetros de entrada del Grid Map — Segundo modelo

| Símbolo     | Descripción            | Valor empleado | Unidad      |
|-------------|------------------------|----------------|-------------|
| $\varphi_c$ | Latitud del centroide  | -25,2637       | °           |
| $\lambda_c$ | Longitud del centroide | -57,5759       | °           |
| $L$         | Extensión del mapa     | 1000           | m           |
| $\Delta_s$  | Tamaño de celda        | 3              | m/celda     |
| $N$         | Dimensión de la grilla | 333            | celdas/lado |

**Fuente:** Elaboración propia.

Con estos parámetros se obtuvo una grilla de  $333 \times 333$  celdas cubriendo  $1 \text{ km}^2$  del entorno urbano real. El tamaño de celda  $\Delta_s = 3 \text{ m}$  constituye el mismo tipo de compromiso entre resolución de desplazamiento y costo computacional; al operar sobre un área más reducida que en el primer modelo, fue posible disminuir  $\Delta_s$  sin incurrir en un costo computacional prohibitivo, mejorando así la resolución espacial del entorno urbano real representado. Se integraron las **7 trampas físicas** reales como sitios de referencia georreferenciados (clase 5 en la grilla); los cuales, para este modelo, se toman como sitios de cría de referencia para su futuro estudio y calibración de parámetros. Además, se agregaron aleatoriamente otros 20 sitios de cría sintéticos, alcanzando un total de 27 sitios en la simulación.

### 9.2.2.2. Condiciones iniciales y parámetros del modelo

La Tabla 9.12 resume las condiciones iniciales de la simulación. Los parámetros biológicos y espaciales del modelo corresponden a los valores por defecto de la Tabla 8.3 del Capítulo 8.

**Tabla 9.12:** Condiciones iniciales y parámetros de simulación — Segundo modelo

| Símbolo    | Descripción                    | Valor | Unidad     |
|------------|--------------------------------|-------|------------|
| $J_0$      | Juveniles iniciales            | 300   | individuos |
| $M_0$      | Machos adultos iniciales       | 1000  | individuos |
| $F_{U0}$   | Hembras no apareadas iniciales | 500   | individuos |
| $F_{G0}$   | Hembras grávidas iniciales     | 500   | individuos |
| $\Delta t$ | Paso de tiempo                 | 1,0   | días       |
| $m$        | Número de sitios de cría       | 27    | sitios     |

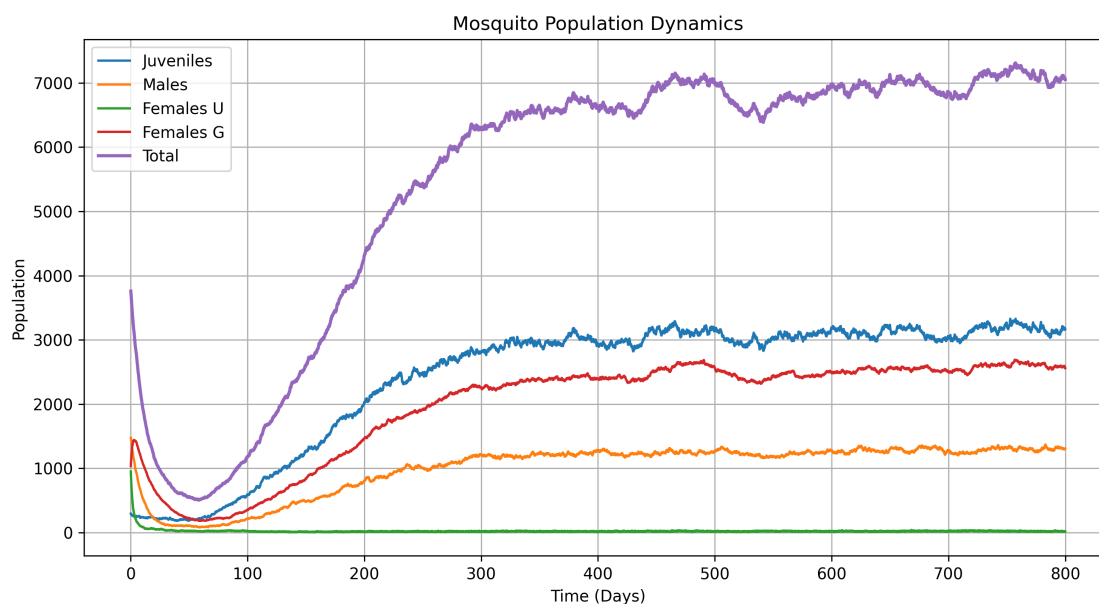
**Fuente:** Elaboración propia.

Al finalizar la corrida, el estado final del sistema registró un total de 7 055 agentes activos distribuidos en los cuatro estados no absorbentes ( $J$ ,  $M$ ,  $F_U$ ,  $F_G$ ).

### 9.2.2.3. Dinámica poblacional temporal

La Figura 9.18 muestra la evolución del número de agentes por estado a lo largo del tiempo simulado. Se observaron las fases características del ciclo de vida del mosquito *Aedes aegypti*: un crecimiento inicial dominado por la eclosión de juveniles, el establecimiento progresivo de una población adulta diferenciada por sexo y estado reproductivo, y una estabilización determinada por el equilibrio entre las tasas de mortalidad, la dinámica de apareamiento y la disponibilidad de sitios de cría en el entorno.

APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS

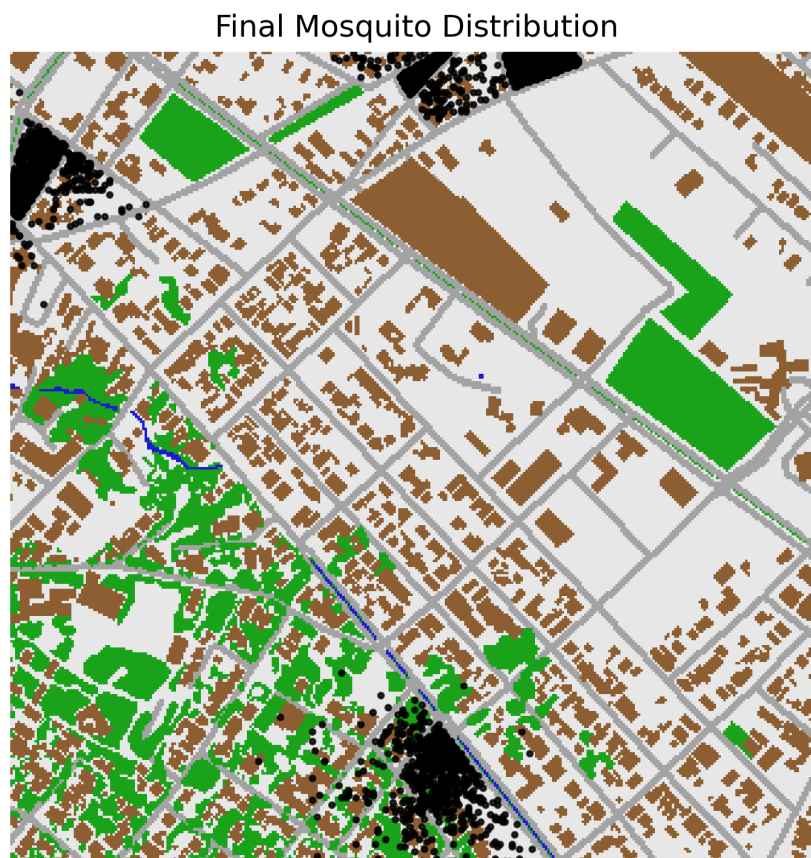


**Figura 9.18:** Dinámica poblacional de *Aedes aegypti* a lo largo de la simulación. Cada curva representa la evolución temporal del número de agentes en cada estado: Juvenil ( $J$ ), Macho adulto ( $M$ ), Hembra no apareada ( $F_U$ ) y Hembra grávida ( $F_G$ ). Los parámetros del modelo corresponden a la Tabla 8.3 (Capítulo 8); los parámetros del entorno corresponden a la Tabla 9.11. **Fuente:** Elaboración propia.

La estabilización observada en las poblaciones diferenciadas por sexo y estado reproductivo es consistente con el equilibrio dinámico esperado bajo condiciones de temperatura constante y disponibilidad estable de sitios de cría. La presencia de hembras grávidas ( $F_G$ ) como estado dominante en la fase estacionaria es coherente con la biología del vector: en ausencia de factores limitantes adicionales, la fracción de hembras que ha completado el ciclo de apareamiento tiende a ser mayoritaria. Este comportamiento, obtenido bajo parámetros no calibrados, resulta cualitativamente consistente con la dinámica esperada del *Aedes aegypti* y ofrece un punto de partida interpretable para la comparación futura con datos de campo.

#### 9.2.2.4. Distribución espacial final de agentes

La Figura 9.19 presenta la distribución espacial de los agentes en el estado final de la simulación, superpuesta sobre la grilla del entorno urbano. La concentración de individuos en determinadas zonas reflejó la influencia del campo de potencial  $\Pi(x)$  (ecuación 8.23): los agentes tendieron a acumularse en las proximidades de los sitios de cría, mientras que las celdas de tipo calle ( $c(x) = 1$ ), agua ( $c(x) = 3$ ) y edificio ( $c(x) = 4$ ) actuaron como barreras que moldearon los patrones de dispersión espacial.



**Figura 9.19:** Distribución espacial final de los agentes sobre la grilla del entorno urbano. Cada punto representa un agente individual codificado por color según su estado ( $J, M, F_U, F_G$ ). El fondo corresponde al Grid Map clasificado a partir de datos de OpenStreetMap ( $333 \times 333$  celdas,  $\Delta s = 3$  m). Los puntos magenta indican las posiciones de las 7 trampas físicas integradas. **Fuente:** Elaboración propia.



La distribución espacial final refleja la acción combinada del campo de potencial y las restricciones impuestas por el tipo de terreno: los agentes evitan las celdas de calle y agua, y se agrupan en torno a los sitios de cría disponibles, particularmente en zonas con presencia de vegetación. Esta concentración es cualitativamente consistente con la biología del *Aedes aegypti*, cuya actividad está fuertemente determinada por la disponibilidad de recipientes con agua en el entorno urbano. La superposición de las 7 trampas físicas georreferenciadas sobre la distribución simulada ilustra el potencial de la herramienta para representar escenarios de monitoreo real: en una etapa futura de calibración, los conteos registrados en cada trampa podrían utilizarse directamente para ajustar los parámetros del modelo y evaluar la validez de los patrones espaciales simulados.

### 9.2.3. Análisis de Sensibilidad del Segundo Modelo

Con el fin de identificar los parámetros más influyentes sobre el comportamiento del segundo modelo ABM y orientar futuras campañas de calibración, se realizó un análisis de sensibilidad global mediante **LHS** combinado con el **PRCC**.

El carácter estocástico y no lineal de los ABM hace inadecuados los métodos locales de sensibilidad (perturbación de un parámetro a la vez), que no capturan interacciones entre parámetros ni comportamientos de umbral. La combinación LHS+PRCC es el estándar para este tipo de modelos en la literatura de sensibilidad en biología computacional [44]: el diseño estratificado LHS [45] garantiza una cobertura uniforme del espacio de parámetros con un número reducido de evaluaciones, mientras que el estimador PRCC cuantifica la influencia *net*a de cada parámetro sobre una cantidad de interés, eliminando el efecto de los demás mediante residuos de regresión sobre rangos.

### 9.2.3.1. Configuración del experimento

El diseño LHS se generó sobre el espacio de los  $k = 13$  parámetros del modelo (Tabla 8.3, Capítulo 8), empleando  $N = 100$  muestras estratificadas con semilla determinista. El rango de variación de cada parámetro se fijó en  $\pm 50\%$  alrededor de su valor por defecto. Para cada muestra se ejecutó una réplica de la simulación durante  $T_{\text{máx}} = 80$  días ( $\Delta t = 1,0$  d), bajo condiciones iniciales  $J_0 = 200$ ,  $M_0 = 80$ ,  $F_{U0} = 40$ ,  $F_{G0} = 40$  y  $m = 20$  sitios de cría aleatorios (sin incluir las trampas físicas) sobre el Grid Map georreferenciado de la Sección 8.3, utilizando un entorno simplificado respecto a la simulación principal para reducir el costo computacional del análisis. Las simulaciones se ejecutaron en paralelo mediante `multiprocessing.Pool`; los casos de extinción temprana se interrumpieron anticipadamente para reducir el costo computacional total.

Con  $N = 100$  muestras se supera ampliamente la regla práctica  $N > 4k$  ( $100 \gg 52$ ), lo que otorga una estabilidad estadística adecuada a los estimadores PRCC y reduce la variabilidad de la jerarquía de sensibilidad respecto a diseños más pequeños.

El estimador PRCC del parámetro  $p_i$  sobre la Quantity of Interest (QoI)  $y$  se obtiene como la correlación de Pearson entre los residuales de la regresión de  $\text{rk}(p_i)$  sobre  $\{\text{rk}(p_j)\}_{j \neq i}$  y los residuales de la regresión de  $\text{rk}(y)$  sobre los mismos predictores, donde  $\text{rk}(\cdot)$  denota el rango. Los valores pertenecen al intervalo  $[-1, 1]$ : positivo si el parámetro incrementa la QoI; negativo si la reduce. Convención adoptada:  $|\text{PRCC}| > 0,5$ : influencia alta;  $0,3-0,5$ : influencia moderada;  $< 0,1$ : influencia negligible.

Las cinco **cantidades de interés** evaluadas fueron:

- $N_{\text{final}}$ : número total de agentes al final de la simulación.
- $\overline{F_G}$ : media temporal del número de hembras grávidas.

- $N_{\text{máx}}$ : pico máximo de población total a lo largo de la simulación.
- $\bar{J}$ : media temporal del número de juveniles.
- $E$ : indicador binario de extinción ( $E = 1$  si  $N_{\text{final}} = 0$ ,  $E = 0$  en caso contrario).

### 9.2.3.2. Resultados e interpretación

La Tabla 9.13 presenta la jerarquía de sensibilidad sobre  $N_{\text{final}}$ ; el heatmap completo (Figura 9.20) y el diagrama de tornado (Figura 9.21) complementan el análisis para las cinco QoI.

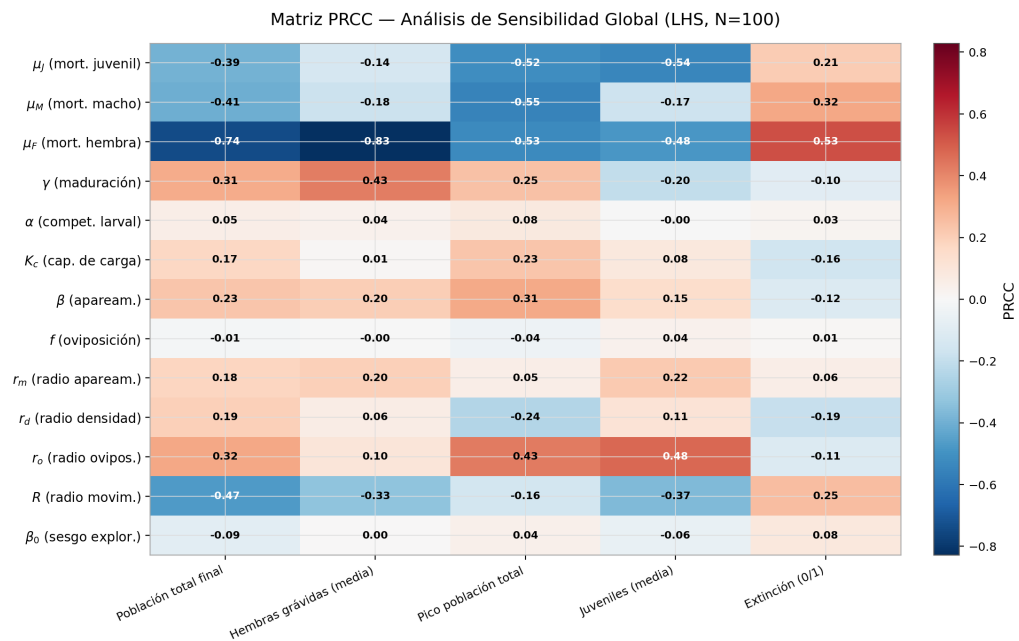
**Tabla 9.13:** Coeficientes PRCC sobre  $N_{\text{final}}$  ( $N = 100$  muestras LHS,  $T_{\text{máx}} = 80$  d). El símbolo  $\star$  indica  $|\text{PRCC}| > 0,3$ .

| Rango | Parámetro                      | PRCC  | Signo   |
|-------|--------------------------------|-------|---------|
| 1     | $\mu_F$ (mortalidad hembra)    | -0,79 | $\star$ |
| 2     | $\mu_M$ (mortalidad macho)     | -0,52 | $\star$ |
| 3     | $r_m$ (radio apareamiento)     | +0,48 | $\star$ |
| 4     | $r_o$ (radio oviposición)      | +0,38 | $\star$ |
| 5     | $\beta$ (tasa apareamiento)    | +0,35 | $\star$ |
| 6     | $R$ (radio de movimiento)      | -0,34 | $\star$ |
| 7     | $\mu_J$ (mortalidad juvenil)   | -0,32 | $\star$ |
| 8     | $\gamma$ (maduración larval)   | +0,31 | $\star$ |
| 9     | $f$ (oviposición)              | -0,15 |         |
| 10    | $r_d$ (radio densidad)         | -0,13 |         |
| 11    | $K_c$ (capacidad de carga)     | -0,05 |         |
| 12    | $\beta_0$ (sesgo exploratorio) | +0,04 |         |
| 13    | $\alpha$ (competencia larval)  | +0,01 |         |

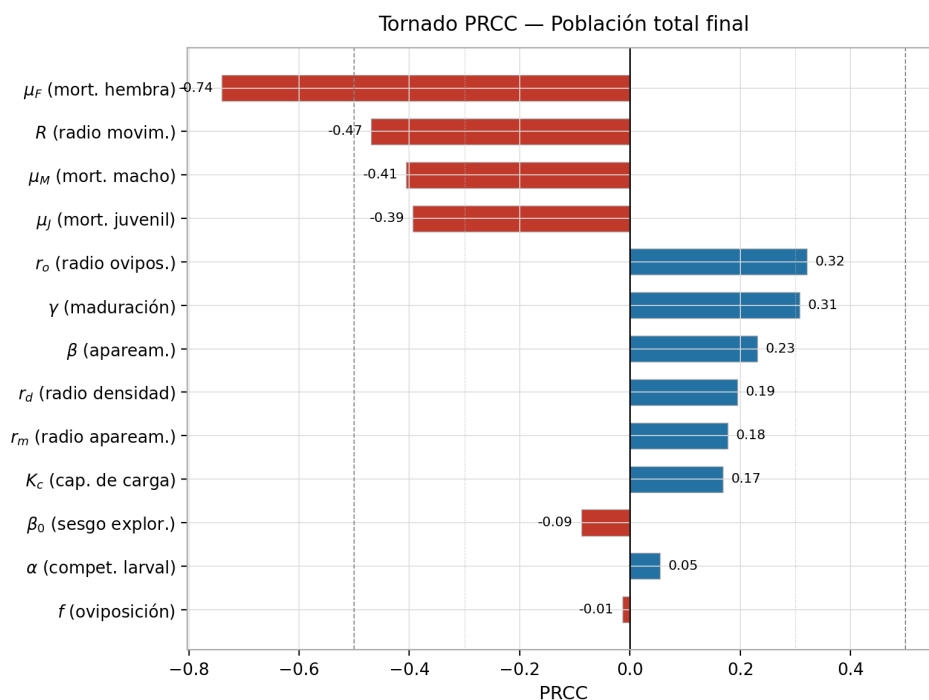
**Fuente:** Elaboración propia.

La Figura 9.20 presenta la matriz completa de PRCC para las cinco QoI analizadas, donde cada fila corresponde a un parámetro y cada columna a una variable de respuesta.

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**



**Figura 9.20:** Heatmap de la matriz PRCC completa (13 parámetros  $\times$  5 QoL). Los valores en rojo indican efectos negativos (el parámetro reduce la QoL); en azul, efectos positivos. La intensidad del color es proporcional a  $|\text{PRCC}|$ . **Fuente:** Elaboración propia.



**Figura 9.21:** Diagrama de tornado de los índices PRCC sobre  $N_{\text{final}}$ . Las barras rojas indican efectos negativos; las barras azules, efectos positivos. Las líneas verticales punteadas marcan los umbrales  $|\text{PRCC}| = 0,5$  (influencia alta). **Fuente:** Elaboración propia.

La Figura 9.21 muestra el diagrama de tornado correspondiente a  $N_{\text{final}}$ , que ordena visualmente los parámetros por magnitud de influencia. Los resultados revelan tres patrones que estructuran la jerarquía de sensibilidad.

**Dominancia de la mortalidad adulta femenina.** La tasa  $\mu_F$  es con diferencia el parámetro más influyente ( $|\text{PRCC}| = 0,79$ ). Este resultado es biológicamente coherente: la hembra es el nexo entre el apareamiento, la oviposición y la supervivencia del ciclo reproductivo. Un incremento de  $\mu_F$  reduce simultáneamente la fracción de hembras grávidas disponibles y el tiempo en que cada individuo puede ovipositar, generando un efecto compuesto que supera la influencia individual de cualquier otro parámetro.

**El apareamiento como cuello de botella espacial.** La mortalidad de machos ( $\mu_M$ ,  $|\text{PRCC}| = 0,52$ ) y el radio de apareamiento ( $r_m$ ,  $|\text{PRCC}| = 0,48$ ) ocupan el segundo y tercer lugar. Este resultado pone de relieve que, en el modelo, la disponibilidad local de machos es un factor limitante de la reproducción tan relevante como la demografía femenina: sin machos accesibles en la vecindad de radio  $r_m$ , las hembras permanecen en el estado no reproductivo  $F_U$  durante intervalos prolongados, reduciendo la tasa efectiva de oviposición.

**Radios espaciales como determinantes de la dinámica.** Los parámetros  $r_o$  ( $|\text{PRCC}| = 0,38$ ),  $R$  ( $|\text{PRCC}| = 0,34$ ) y  $r_m$  ( $|\text{PRCC}| = 0,48$ ) tienen influencia moderada o alta, lo que confirma que la dimensión espacial del modelo —la geometría de los radios de interacción— no es un detalle de implementación sino un determinante sustancial del comportamiento macroscópico. En particular, el radio de movimiento  $R$  presenta un efecto *negativo* sobre  $N_{\text{final}}$ : un mayor radio dispersa a los agentes adultos, reduciéndolos en zonas con alta densidad de sitios de cría y disminuyendo la frecuencia efectiva de encuentros de apareamiento y oviposición. Este comportamiento emergente —no codificado explícitamente en las reglas del modelo— ilustra cómo la interacción entre el movimiento estocástico y la distribución espacial de los sitios de cría puede generar consecuencias

contraintuitivas sobre la dinámica global.

**Parámetros de regulación larval con influencia negligible.** Los parámetros de densodependencia larval ( $\alpha$ ,  $K_c$ ) y el sesgo exploratorio  $\beta_0$  presentan  $|\text{PRCC}| < 0,1$ , indicando que su variación en el rango  $\pm 50\%$  no tiene efecto detectable sobre las QoI a 80 días de simulación. Esto sugiere que, en el horizonte temporal analizado y bajo los valores por defecto de los parámetros demográficos y espaciales, la regulación densodependiente larval no actúa como cuello de botella efectivo: la dinámica poblacional está dominada por la mortalidad adulta y las interacciones de apareamiento antes de que los mecanismos de competencia larval puedan limitar el crecimiento. Esta observación tiene implicaciones directas para la calibración: esforzarse por estimar con precisión  $\alpha$  o  $K_c$  a partir de datos de campo tiene escaso impacto sobre el poder predictivo del modelo en el corto plazo.

## Síntesis de los Resultados del Modelado

Los resultados de ambos modelos ABM constituyen una validación de coherencia computacional, no empírica. El análisis de sensibilidad identificó la mortalidad femenina y la dinámica de apareamiento como los determinantes dominantes de la población final, orientando la priorización de parámetros para la futura calibración con datos de campo generados por el sistema de trampas desarrollado en la primera parte de este trabajo.

## **CAPÍTULO 10**

### **Conclusiones y Recomendaciones**

Este Capítulo cierra el recorrido trazado a lo largo del trabajo. En los capítulos anteriores se completaron las ocho fases de desarrollo planteadas en la metodología: las cuatro fases de la trampa inteligente (FDT1–4), que abarcaron la construcción del hardware, el diseño y entrenamiento de los modelos de visión artificial, la integración de la arquitectura IoT y la validación experimental en campo; y las cuatro fases del modelado basado en agentes (FDM1–4), que comprendieron el modelo preliminar con base en ODE, la construcción del entorno georreferenciado a partir de datos GIS, y el desarrollo de dos versiones del simulador ABM con progresiva incorporación de realismo biológico y localidad espacial. Se presentan a continuación las conclusiones derivadas de este proceso, las reflexiones técnicas surgidas durante el desarrollo, las recomendaciones para la continuación del trabajo y la proyección de integración futura entre ambas líneas.

#### **10.1. Conclusiones**

El presente trabajo desarrolló dos herramientas complementarias para la adquisición de datos de alta resolución y el modelado de poblaciones de mosquitos. La trampa inteligente articula electrónica, diseño mecánico, visión por computador, aprendizaje profundo y conectividad IoT; el simulador basado en agentes integra representación geoespacial, procesos estocásticos locales y entornos de-

rivados de datos GIS. Ambas herramientas tienen valor independiente, pero su potencial mayor reside en su integración: los datos generados por la trampa constituyen el insumo natural para calibrar el ABM, cerrando el ciclo entre adquisición de datos y modelado poblacional. Esa integración queda planteada como la principal línea de desarrollo futuro; lo que este trabajo establece es la base técnica de cada uno de sus componentes.

### 10.1.1. Síntesis de Aportes

**Trampa inteligente.** Se desarrolló un pipeline en cascada YOLOv8 + MobileNetV3 exportado a formato ONNX y ejecutado íntegramente en CPU sobre la Raspberry Pi 5. Las mediciones realizadas en el Capítulo 9 confirman la viabilidad del despliegue en el dispositivo de borde: el sistema completo opera a **133,53 ms por imagen (7,49 FPS)** (Tabla 5.20), con el detector YOLOv8 concentrando 101,11 ms y el clasificador MobileNetV3 aportando únicamente 14,77 ms adicionales.

La validación experimental arrojó resultados consistentes en ambas pruebas de semi-campo realizadas con especímenes de *Aedes aegypti* proporcionados por el IICS: una **precisión de clasificación**  $P_{\text{class}} = 93,75\%$  y un **ratio de retención física**  $R_{\text{ret}} = 87,5\%$  (Tabla 9.7, Capítulo 9). El ratio de captura sobre la población efectiva fue de 26,66 % y 29,09 % en la primera y segunda prueba, respectivamente. En la prueba de campo de seis días realizada en el Barrio Ycuá Satí, el sistema capturó 21 mosquitos en condiciones reales de operación con una **precisión de campo**  $P_{\text{campo}} = 90,5\%$  (ecuación 9.11, Capítulo 9), operando de forma continua y autónoma sin ajuste de hiperparámetros para el entorno exterior. Se implementó además un diseño mecánico novedoso basado en un mecanismo de rotación para la clasificación física de los individuos detectados. El conjunto de componentes —trampa, inferencia local en el dispositivo de borde y arquitectura IoT desplegada en la nube— constituye un sistema de



adquisición de datos **completamente automatizado**: desde la captura física y clasificación por sexo hasta la transmisión, almacenamiento y visualización en tiempo real, sin intervención manual continua (Capítulo 6, Capítulo 9).

**Modelado basado en agentes.** Se desarrollaron dos versiones del simulador ABM georreferenciado (FDM3 y FDM4) sobre una retícula discreta derivada de datos de OpenStreetMap que cubre  $1 \text{ km}^2$  de un barrio real de Asunción. El segundo modelo, que incorpora distinción de sexo, mortalidad densodependiente y apareamiento por vecindario, reprodujo cualitativamente las fases características del ciclo vital del mosquito —crecimiento inicial, establecimiento de población adulta diferenciada y estabilización— bajo condiciones de temperatura constante. El análisis de sensibilidad global mediante LHS+PRCC ( $N = 100$  muestras,  $k = 13$  parámetros, Sección 9.2.3) identificó la mortalidad femenina ( $\mu_F$ ,  $|\text{PRCC}| = 0,79$ ) como el parámetro dominante sobre la población final, seguida por la mortalidad masculina ( $\mu_M$ ,  $|\text{PRCC}| = 0,52$ ) y el radio de apareamiento ( $r_m$ ,  $|\text{PRCC}| = 0,48$ ), orientando la priorización de parámetros para la calibración futura con datos de las trampas (Tabla 9.13, Capítulo 9). El diseño escalable del simulador permite incorporar nuevos estados —como mosquitos con *Wolbachia*— sin modificar el núcleo del modelo.

### 10.1.2. Reflexiones sobre el proceso de desarrollo

El desarrollo de ambas herramientas dio lugar a observaciones de carácter reflexivo que complementan los aportes técnicos enumerados anteriormente. Estas reflexiones no constituyen recomendaciones de trabajo futuro —las cuales se detallan en la Sección 10.2—, sino consideraciones surgidas directamente de la experiencia de diseño e implementación.

**Mecanismo de clasificación física.** El mecanismo rotatorio (FDT1) demostró que la simplicidad funcional no está reñida con la eficiencia operativa: al reducir la clasificación física a una única rotación determinada por la señal del

modelo de visión, se obtuvo un sistema robusto y de bajo costo mecánico. Su limitación inherente —la incapacidad de clasificar correctamente cuando dos individuos ingresan de forma simultánea— no invalida el diseño, sino que delimita con precisión el rango de condiciones bajo el cual el sistema opera de forma fiable, constituyendo un criterio de diseño bien definido para iteraciones futuras.

**Pipeline en cascada.** La arquitectura en cascada YOLOv8 + MobileNetV3 (FDT2) demostró ser técnicamente sólida para la detección y clasificación simultánea de mosquitos. La caída de eficiencia observada en la evaluación integrada fue atribuible, en gran medida, a una diferencia de dominio entre el dataset del detector y el conjunto de prueba empleado —y no a una limitación estructural del modelo—, lo que sugiere que la arquitectura dispone de margen de mejora sin cambios de fondo. Igualmente relevante fue la facilidad de extensión del pipeline: incorporar un tercer clasificador de especie tiene un costo de entrenamiento e inferencia relativamente bajo dado que opera únicamente sobre las detecciones ya filtradas por las dos etapas previas. Un aspecto crítico emergido durante el desarrollo fue la necesidad de verificar exhaustivamente los grafos exportados en formato ONNX: en varias instancias se observó el colapso silencioso del nodo de clases de salida, generando un modelo que difería funcionalmente del original en PyTorch sin emitir ningún error explícito, lo que confirmó que la validación dimensional de la capa de salida es indispensable antes de cualquier despliegue.

**Evolución del simulador: de la demografía derivada del ODE a la localidad estricta.** El trayecto entre el primer y el segundo modelo no fue un simple refinamiento incremental, sino un cambio de paradigma motivado por una limitación estructural concreta. Derivar las probabilidades de transición demográfica del sistema ODE de referencia [1] (FDM3) permitió anclar la coherencia poblacional del simulador en parámetros validados empíricamente en la literatura; sin embargo, el mecanismo de oviposición resultante distribuía los nacimientos de cada adulto entre *todos* los sitios de cría del dominio mediante una función de decaimiento

global, violando el principio de interacción estrictamente local que define al paradigma ABM. Esta tensión entre base bibliográfica y coherencia paradigmática fue la que impulsó el desarrollo del segundo modelo (FDM4): al reemplazar las tasas continuas del ODE por reglas estocásticas de Bernoulli aplicadas individualmente, y al restringir el apareamiento y la oviposición a vecindarios finitos, la dinámica reproductiva pasó a emerger de encuentros entre agentes concretos en posiciones concretas. El resultado fue un simulador en el que los patrones espaciales —concentración de adultos en zonas con alta densidad de sitios de cría, formación de gradientes de distribución por sexo— surgieron sin estar codificados explícitamente, cumpliendo así la propiedad de emergencia que justifica el uso del ABM frente a enfoques de campo medio.

**Diseño de interacciones locales en el ABM.** El desarrollo del segundo modelo puso de relieve un desafío conceptual fundamental en el diseño de modelos basados en agentes: cómo lograr comportamientos complejos a nivel poblacional a partir de reglas estrictamente locales y, más difícil aún, cómo establecer si una regla dada es *completamente* local o solo *parcialmente* local. Esta distinción, que puede parecer sutil, tiene consecuencias profundas sobre el realismo biológico y la coherencia del paradigma: una regla aparentemente local que en su implementación realiza cálculos sobre toda la población pierde la propiedad de localidad que define al ABM. La experiencia reforzó la necesidad de un estudio más riguroso de los marcos formales de sistemas complejos para fundamentar estas decisiones de diseño, especialmente en lo que respecta a la generación de propiedades globales —como la capacidad de carga efectiva— únicamente como fenómeno emergente de interacciones puramente locales entre agentes y entorno.

## 10.2. Recomendaciones para Trabajo Futuro

A continuación se presentan recomendaciones divididas para cada línea de trabajo correspondiente.

### 10.2.1. Recomendaciones para la Trampa

**Mejoras en el hardware de la trampa:** El hardware puede optimizarse significativamente mediante la selección de dispositivos de borde más económicos, considerando que la inferencia en ONNX ha demostrado ser altamente eficiente. Asimismo, se recomienda el diseño de una placa unificada con componentes Surface-Mount Device (SMD) que integre sensado y accionamiento de potencia. Adicionalmente, la incorporación de nuevos sensores permitiría enriquecer la adquisición de datos, junto con la inclusión de almacenamiento local de imágenes para escenarios de desconexión del servidor.

**Mejoras en el diseño mecánico de la trampa:** El diseño puede extenderse fácilmente para incluir un tercer compartimento, lo que permitiría clasificar otros insectos no deseados, mejorando la selectividad del sistema.

**Mecanismo de clasificación física:** Se recomienda incorporar un sensor de conteo en el canal de ingreso o rediseñarlo para forzar el paso secuencial de un individuo a la vez, eliminando la limitación de ingreso simultáneo señalada en la Sección 10.1.2.

**Mejoras en el modelo de IA:** El desempeño de los modelos puede mejorarse mediante la ampliación del dataset de entrenamiento, especialmente incorporando datos reales capturados por la trampa. La caída de eficiencia observada en el pipeline en cascada sugiere que parte de la reducción de precisión puede estar asociada a la distribución del dataset del modelo de detección; incorporar imágenes del dataset de clasificación en el entrenamiento del detector constituye una línea de mejora directa. Además, la arquitectura en cascada es fácilmente ex-

tensible: añadir un modelo de discriminación de especies como etapa adicional tiene un costo de entrenamiento e inferencia relativamente bajo.

**Validación de grafos exportados al formato ONNX:** Complementando la observación de la Sección 10.1.2, se recomienda comparar las predicciones del grafo exportado contra el modelo fuente en PyTorch sobre un conjunto de muestras conocidas antes de cualquier despliegue, como paso obligatorio para detectar colapsos silenciosos en la capa de salida.

**Autonomía de la trampa:** Actualmente, el sistema no cuenta con autonomía energética ni de conectividad. Para un despliegue en campo más robusto, se recomienda el uso de baterías recargables, sistemas de energía solar y conectividad mediante redes móviles, garantizando operación continua y acceso remoto.

#### 10.2.2. Recomendaciones para el Modelo Basado en Agentes

**Ajuste de parámetros de la simulación:** Se propone como prioridad la calibración de los parámetros del modelo utilizando datos reales, como los obtenidos de ovitrampas, con el fin de mejorar la validez del modelo.

**Mejora del clima y microclima:** Se recomienda la incorporación de modelos climáticos más realistas, incluyendo variaciones espacio-temporales (microclima), que influyen directamente en la dinámica poblacional.

**Implementación de nuevos estados de agentes:** El modelo está diseñado para ser extensible, por lo que se sugiere la inclusión de nuevos estados, como mosquitos infectados con *Wolbachia*, junto con la adaptación de las reglas de interacción.

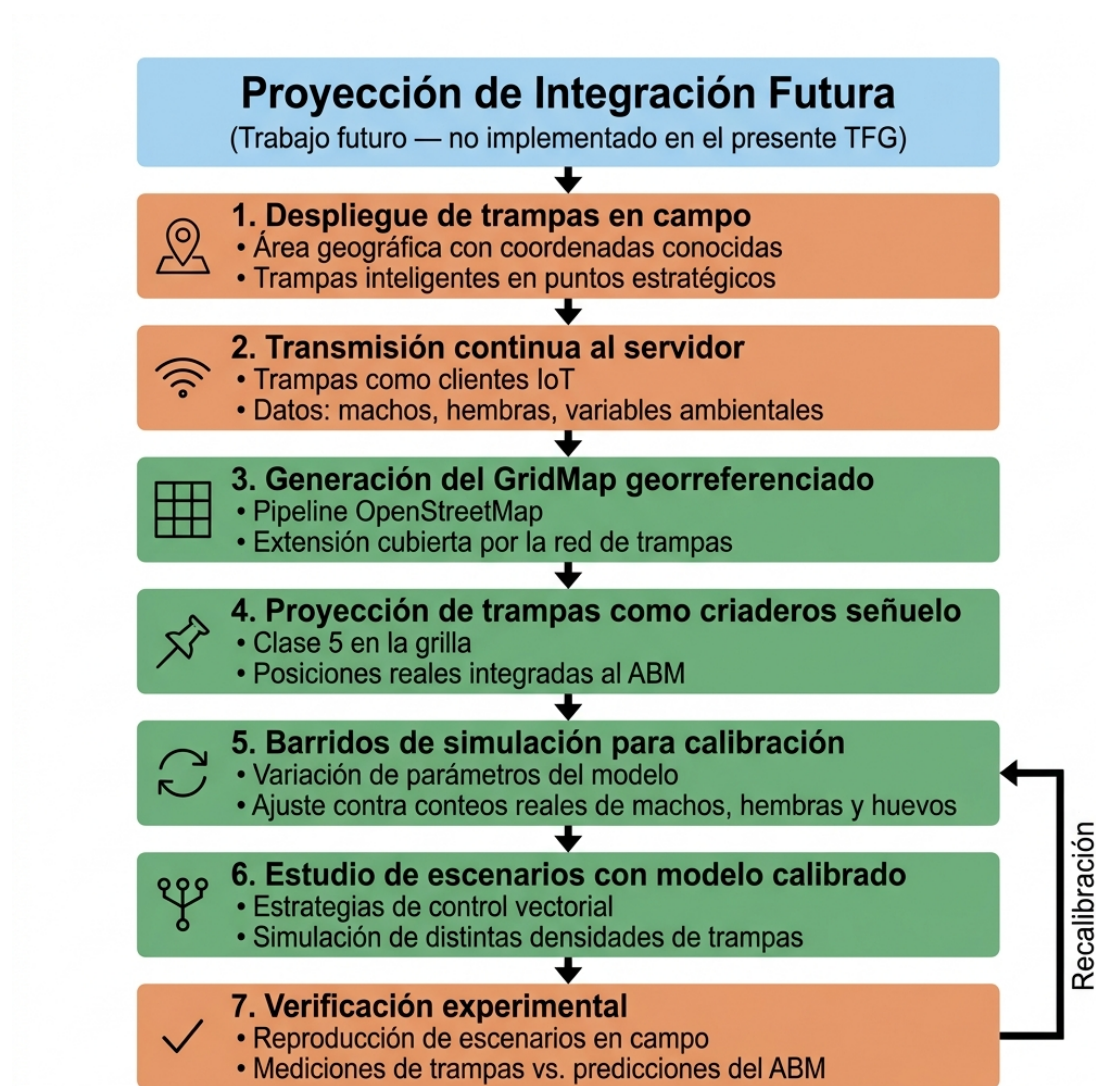
**Análisis de convergencia y complejidad:** Se recomienda realizar un análisis más riguroso de la convergencia del modelo, así como estudiar la influencia de la distribución de sitios de cría y la posibilidad de inferir comportamientos globales a partir de interacciones locales. También se sugiere la generación de

múltiples escenarios para análisis estadístico y clustering.

**Extensión del análisis de sensibilidad:** Un análisis de sensibilidad global mediante LHS+PRCC sobre los  $k = 13$  parámetros del modelo fue realizado como parte de este trabajo (Sección 9.2.3), identificando la mortalidad femenina ( $\mu_F$ ) y los radios de apareamiento y oviposición como los determinantes más influyentes de la dinámica poblacional a corto plazo. Como extensión, se recomienda ampliar el tamaño de muestra a  $N \geq 500$  para obtener intervalos de confianza estadísticamente robustos sobre los índices PRCC, y complementar el análisis con índices de Sobol de segundo orden, que permiten cuantificar interacciones entre pares de parámetros y son especialmente adecuados para horizontes temporales más largos donde la regulación densodependiente larval puede volverse determinante.

**Fundamentación formal del diseño de reglas locales en el ABM:** Se recomienda profundizar en el estudio de marcos formales de sistemas complejos — como los formalismos Discrete Event System Specification (DEVS) o las teorías de campo medio discretas— para fundamentar con mayor rigor las decisiones de diseño del modelo (véase la reflexión correspondiente en la Sección 10.1.2). En particular, se identifica como línea prioritaria la generación de la capacidad de carga efectiva como propiedad puramente emergente, en lugar de imponerla mediante el parámetro explícito  $K_c$ .

### 10.2.3. Integración de Ambas Partes



**Figura 10.1:** Proyección de integración futura entre el sistema de trampas inteligentes y el modelo basado en agentes. Los pasos naranjas corresponden al componente IoT/trampa; los pasos verdes, al componente ABM. **Fuente:** Elaboración propia.

La Figura 10.1 ilustra el pipeline propuesto, que operaría en siete estadios. En primer lugar, se desplegarían trampas inteligentes en un área geográfica de interés, cuyas coordenadas son conocidas con precisión. Cada trampa actúa como un cliente IoT que transmite de forma continua al servidor los datos de detección y variables ambientales. A partir de la extensión geográfica cubierta por

la red de trampas, se genera un GridMap georreferenciado mediante el proceso descrito en el Capítulo 7; las posiciones conocidas de las trampas se proyectan sobre la grilla y se registran como sitios de cría señuelo (clase 5), de modo que el ABM puede simular la dinámica poblacional en el entorno real de despliegue.

Con este entorno configurado, se ejecutan barridos sistemáticos de simulación variando los parámetros del modelo —tasas de mortalidad, maduración, apareamiento y oviposición— con el objetivo de reproducir, mediante calibración, los conteos de mosquitos machos, hembras y huevos registrados por las trampas. Una vez que el modelo calibrado reproduce satisfactoriamente las mediciones, se generan escenarios alternativos —distintas densidades de trampas, estrategias de control, modificaciones ambientales— que son evaluados en simulación. Algunos de estos escenarios se reproducen experimentalmente en campo y los conteos reales registrados por las trampas se comparan con las predicciones del modelo, cerrando el ciclo de verificación y retroalimentando el proceso de calibración.

Este enfoque integrado tiene el potencial de evolucionar hacia un sistema capaz de no solo monitorear, sino también predecir y optimizar estrategias de control vectorial, contribuyendo de manera significativa al manejo de enfermedades transmitidas por mosquitos. En particular, la disponibilidad conjunta de datos de campo de alta resolución —generados por las trampas inteligentes— y de un modelo espacio-temporal validado constituiría la base técnica necesaria para abordar, en trabajos futuros, la pregunta de largo alcance que motivó el presente trabajo: la posibilidad de inferir la dinámica poblacional del vector a partir de observaciones parciales. Esta capacidad predictiva, a su vez, es precisamente la que permitiría apoyar de forma cuantitativa estrategias de control como *Wolbachia*, cuya implementación efectiva en el país requiere tanto datos precisos de campo como modelos capaces de simular escenarios antes de comprometer recursos en el territorio.



**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

Las páginas de este trabajo se abren con el reconocimiento de que todo esfuerzo por imponer orden sobre lo complejo conlleva una cierta ilusión de dominio. Lo desarrollado aquí no ha sido ajeno a esa tensión: cada decisión —qué capturar, qué modelar, qué simplificar, qué dejar fuera— es, en última instancia, un acto de reducción deliberada frente a la complejidad real del sistema estudiado. Lo que aquí se presenta no pretende ser una solución, sino un conjunto de herramientas y una orientación de camino. Y si ese camino conduce, en trabajos futuros, a modelos más rigurosos, a trampas más precisas y a estrategias de control más efectivas, el intento habrá cumplido su propósito —tan necesario como inevitablemente insuficiente.

## BIBLIOGRAFÍA

- [1] J. Chen, X. Huo, A. B. Wilke, J. C. Beier, C. Vasquez, W. Petrie, R. S. Cantrell, C. Cosner, y S. Ruan, "Linking mathematical models and trap data to infer the proliferation, abundance, and control of *aedes aegypti*," *Acta Tropica*, vol. 239, p. 106837, 2023. [En línea]. Disponible: <https://www.sciencedirect.com/science/article/pii/S0001706X23000244>
- [2] Ministerio da Saúde do Brasil / MERCOSUR - COVIGSAL, "Boletín epidemiológico mercosur – covigsal, volumen 3 (2025)," 2025, documento oficial regional con datos epidemiológicos de Argentina, Brasil, Paraguay y Uruguay. Incluye estadísticas de dengue en Paraguay para 2024. Accedido: 31-03-2026. [En línea]. Disponible: <https://www.gov.br/saude/pt-br/centrais-de-conteudo/publicacoes/boletins/epidemiologicos/especiais/2025/boletin-epidemiologico-mercosur-covigsal.pdf>
- [3] Programa Wolbachia Paraguay, "Estrategia wolbachia en paraguay," n.d., [Accessed: 31-03-2026]. [En línea]. Disponible: <https://wolbachiaparaguay.org/#estrategia>
- [4] Congreso Nacional de Paraguay, "Expediente n° 123352 - implementación de la estrategia wolbachia," n.d., [Accessed: 31-03-2026]. [En línea]. Disponible: <https://silpy.congreso.gov.py/web/expediente/123352>
- [5] World Mosquito Program, "Wolbachia reduce dramáticamente los casos de dengue," 2021, comunicado de prensa oficial. Resultados del ensayo controlado en Yogyakarta, Indonesia: 77% reducción de casos y 86% reducción de hospitalizaciones. Accedido: 31-03-2026. [En línea]. Disponible: <https://www.worldmosquitoprogram.org/es/noticias-historias/comunicados-de-prensa/wolbachia-reduce-dramaticamente-los-casos-de-dengue>

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

- [6] I. R. Zimmermann, R. R. Alves Fernandes, M. G. Santos da Costa, M. Pinto, y H. M. Peixoto, "Simulation-based economic evaluation of the wolbachia method in brazil: a cost-effective strategy for dengue control," *The Lancet Regional Health – Americas*, vol. 35, p. 100783, 2024.
- [7] Ministerio de Salud del Perú (MINSA), "Minsa lanzará novedosa estrategia wolbachia para reducir el dengue," 2025, noticia oficial del Gobierno del Perú sobre la implementación de la estrategia Wolbachia como política pública de control del dengue. Accedido: 31-03-2026. [En línea]. Disponible: <https://www.gob.pe/institucion/minsa/noticias/1231127-minsa-lanzara-novedosa-estrategia-wolbachia>
- [8] Servicio Nacional de Erradicación del Paludismo (SENEPA), "Paraguay presente en taller internacional sobre control de aedes aegypti," 2025, artículo oficial del SENEPA. Participación de Paraguay en taller internacional sobre nuevas tecnologías para el control del mosquito transmisor del dengue. Accedido: 31-03-2026. [En línea]. Disponible: <https://senepa.gov.py/2025/06/02/paraguay-presente-en-taller-internacional-sobre-control-de-aedes-aegypti/>
- [9] Biogents AG, "Bg-sentinel mosquito trap," 2024, accessed: 2026-04-29. [En línea]. Disponible: <https://eu.biogents.com/bg-sentinel/>
- [10] —, "Bg-counter: Innovative and autonomous mosquito trap station," 2024, accessed: 2026-04-29. [En línea]. Disponible: <https://research-shop.biogents.com/products/bg-counter-innovative-and-autonomous-mosquito-trap-station>
- [11] Anti-Moustiques Intelligent, "Gel co2 for mosquito traps," 2024, accessed: 2026-04-29. [En línea]. Disponible: <https://www.anti-moustiques-intelligent.com/product-page/gel-co2>

- [12] C. A. Day, S. L. Richards, M. H. Reiskind, M. S. Doyle, y B. D. Byrd, "Context-dependent accuracy of the bg-counter remote mosquito surveillance device in north carolina," *Journal of the American Mosquito Control Association*, vol. 36, no. 2, pp. 74–80, 2020.
- [13] J. Redmon, S. Divvala, R. Girshick, y A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [14] C.-Y. Wang, A. Bochkovskiy, y H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2023, arXiv preprint arXiv:2207.02696. [Accessed: 31-03-2026]. [En línea]. Disponible: <https://arxiv.org/abs/2207.02696>
- [15] Ultralytics, "Yolov8 documentation," 2023, [Accessed: 31-03-2026]. [En línea]. Disponible: <https://docs.ultralytics.com>
- [16] L. T. Ramos y A. D. Sappa, "A comprehensive analysis of yolo architectures for tomato leaf disease identification," *Scientific Reports*, vol. 15, p. 26890, 2025.
- [17] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, y H. Adam, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- [18] Microsoft, "Onnx runtime documentation," 2024, [Accessed: 23-04-2026]. [En línea]. Disponible: <https://learn.microsoft.com/en-us/azure/machine-learning/concept-onnx>
- [19] T. Jin, G.-T. Bercea, T. D. Le *et al.*, "Compiling onnx neural network models using mlir," *arXiv preprint arXiv:2008.08272*, 2020. [En línea]. Disponible: <https://arxiv.org/abs/2008.08272>

**APLICACIÓN DE DEEP LEARNING Y MODELADO BASADO EN AGENTES PARA EL DESARROLLO DE UNA  
TRAMPA INTELIGENTE DE MOSQUITOS**

---

- [20] F. Xu y B. Lambert, “Performant on-device inferencing with onnx runtime,” 2023, [Accessed: 23-04-2026]. [En línea]. Disponible: <https://opensource.microsoft.com/blog/2023/02/08/performant-on-device-inferencing-with-onnx-runtime>
- [21] R. M. Hampau, M. Kaptein, R. Van Emden, T. Rost, y I. Malavolta, “An empirical study on the performance and energy consumption of ai containerization strategies for computer-vision tasks on the edge,” in *Proceedings of the ACM International Conference on Evaluation and Assessment in Software Engineering (EASE 2022)*. Association for Computing Machinery, June 2022, pp. 50–59. [En línea]. Disponible: <https://research.vu.nl/en/publications/an-empirical-study-on-the-performance-and-energy-consumption-of-a>
- [22] N. Banfic, D. Fan, K. Vaishnavi *et al.*, “Pushing the limits of on-device streaming asr: A compact, high-accuracy english model for low-latency inference,” *arXiv preprint arXiv:2604.14493*, 2026. [En línea]. Disponible: <https://arxiv.org/abs/2604.14493>
- [23] I. Suchý y M. Turčaník, “Review of large yolov8 and rt-detr energy efficiency on edge devices for real-time detection,” *Scientific Reports*, vol. 16, no. 1, p. 10908, March 2026. [En línea]. Disponible: <https://www.nature.com/articles/s41598-026-46453-6>
- [24] W. M. S. Yamashita, S. S. Das, y G. Chapiro, “Numerical modeling of mosquito population dynamics of aedes aegypti,” *Parasites & Vectors*, vol. 11, p. 245, 2018.
- [25] S. Ishida, J. Lankeit, y G. Viglialoro, “A keller–segel type taxis model with ecological interpretation and boundedness due to gradient nonlinearities,” *arXiv preprint arXiv:2306.12137*, 2023.

- [26] H. Sayama, *Introduction to the Modeling and Analysis of Complex Systems*. Geneseo, NY: Open SUNY Textbooks, 2015.
- [27] D. Oliveira y S. Mafra, "Implementation of an intelligent trap for effective monitoring and control of the aedes aegypti mosquito," *Sensors*, vol. 24, no. 21, p. 6932, 2024.
- [28] W.-L. Liu, Y. Wang, Y.-X. Chen, B.-Y. Chen, A. Y.-C. Lin, S.-T. Dai, C.-H. Chen, y L.-D. Liao, "An iot-based smart mosquito trap system embedded with real-time mosquito image processing by neural networks for mosquito surveillance," *Frontiers in Bioengineering and Biotechnology*, vol. 11, p. 1100968, 2023.
- [29] J. Park, D. I. Kim, B. Choi, W. Kang, y H. W. Kwon, "Classification and morphological analysis of vector mosquitoes using deep convolutional neural networks," *Scientific Reports*, vol. 10, p. 1012, 2020.
- [30] S. Lee, H. Kim, y B.-K. Cho, "Deep learning-based image classification for major mosquito species inhabiting korea," *Insects*, vol. 14, no. 6, p. 526, 2023.
- [31] T. C. Wilkes, A. J. S. McGonigle, T. D. Pering, A. J. Taggart, B. S. White, R. G. Bryant, y J. R. Willmott, "Ultraviolet imaging with low cost smartphone sensors: Development and application of a raspberry pi-based uv camera," *Sensors*, vol. 16, no. 10, p. 1649, 2016.
- [32] R. L. Paul, J. R. Hagler, E. G. Janasov, N. S. McDonald, S. Voyvot, y J. C. Lee, "An effective fluorescent marker for tracking the dispersal of small insects with field evidence of mark–release–recapture of trissolcus japonicus," *Insects*, vol. 15, no. 7, p. 487, 2024.
- [33] S. Maneerat y É. Daudé, "Agent-based simulation study of the intra-urban discontinuity effects in delhi on dispersal of aedes aegypti mosquitoes,"

- Cybergeog: European Journal of Geography*, 2017. [En línea]. Disponible: <http://journals.openedition.org/cybergeog/28499>
- [34] C. P. Ferreira, H. M. Yang, y L. Esteva, "Assessing the suitability of sterile insect technique applied to *aedes aegypti*," *Journal of Biological Systems*, vol. 16, no. 4, pp. 565–577, 2008.
- [35] A. Garcia, F. L. Cônsoli, W. A. C. Godoy, y C. P. Ferreira, "A mathematical approach to simulate spatio-temporal patterns of an insect-pest, the corn rootworm *diabrotica speciosa* in intercropping systems," *Landscape Ecology*, vol. 29, pp. 1531–1540, 2014.
- [36] H. T. Honge Kudo, "Diseño e implementación de un sistema mecatrónico de clasificación de mosquitos," 2025, [Accessed: 27-04-2026]. [En línea]. Disponible: <https://www.ing.una.py/FIUNA3/?p=80882&>
- [37] N. A. Zichner von Schmeling, "Implementación de un detector de frecuencia de aleteo basado en luz infrarroja y mejoras en un sistema automatizado para la clasificación de mosquitos," 2026, [Accessed: 27-04-2026]. [En línea]. Disponible: <https://www.ing.una.py/FIUNA3/?p=83704>
- [38] D. C. Balbuena Duarte y M. R. Barrios Pereira, "Trampa inteligente para detección de mosquitos basada en ia," 2026, [Accessed: 27-04-2026]. [En línea]. Disponible: <https://www.ing.una.py/FIUNA3/?p=84014&>
- [39] F. P. Bleier, *Fan Handbook: Selection, Application, and Design*. McGraw-Hill Education, 1998.
- [40] Y. A. Çengel y J. M. Cimbala, *Mecánica de fluidos: fundamentos y aplicaciones*, 4th ed. McGraw-Hill Interamericana de España, 2018.
- [41] OpenStreetMap contributors, "OpenStreetMap," <https://www.openstreetmap.org>, 2004, accessed: 2025.

- [42] G. Boeing, "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017.
- [43] L. Pin, D. H. Stalder, A. D. Reckziegel, N. Gonzalez, y M. Ferreira, "Spatio-temporal analysis of mosquito populations through agent-based simulation," in *Caderno de resumos do XLIV Congresso Nacional de Matemática Aplicada e Computacional*. Campinas: Galoá, 2025.
- [44] S. Marino, I. B. Hogue, C. J. Ray, y D. E. Kirschner, "A methodology for performing global uncertainty and sensitivity analysis in systems biology," *Journal of Theoretical Biology*, vol. 254, no. 1, pp. 178–196, 2008.
- [45] M. D. McKay, R. J. Beckman, y W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.



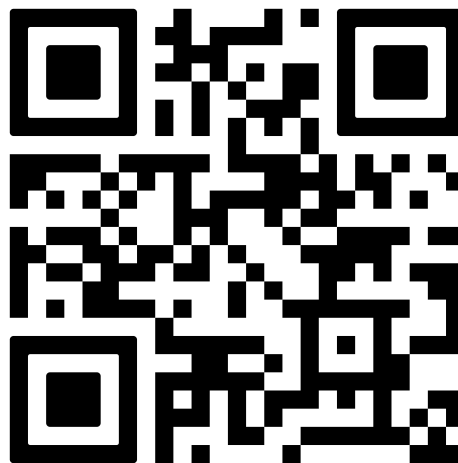
## APÉNDICE A. Repositorio de GitHub

El repositorio desarrollado en el marco de este trabajo reúne los principales recursos empleados durante el diseño, implementación y validación del sistema propuesto. Entre los materiales disponibles se incluyen los códigos de entrenamiento de los modelos de visión artificial, los pesos de los modelos entrenados, el conjunto de datos utilizado, la arquitectura y código fuente de la interfaz web, el simulador basado en agentes, los diseños 3D de las piezas del prototipo y las imágenes utilizadas a lo largo del desarrollo.

El acceso al repositorio puede realizarse mediante el siguiente enlace:

[https://github.com/Spin7/TFG\\_PinMartinez](https://github.com/Spin7/TFG_PinMartinez)

Alternativamente, el repositorio puede consultarse escaneando el código QR presentado en la Figura A.1.



**Figura A.1:** Código QR de acceso al repositorio del proyecto. **Fuente:** Elaboración propia.