UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

"SISTEMA IOT PARA EL CONTROL DE PLAGAS EN EL CULTIVO DE LA MORA DE CASTILLA "RUBUS GLAUCUS BENTH" EN LA COMUNIDAD SAN FRANCISCO DE GUACHINGUERO"

Trabajo de titulación previo a la obtención del título de Ingeniera en Telecomunicaciones

Línea de investigación: Producción industrial y tecnología sostenible

AUTOR:

Diana Soledad Panamá Tuquerez

DIRECTOR:

Ing. Jaime Roberto Michilena Calderón, Msc

Ibarra – Ecuador 2026

UNIVERSIDAD TÉCNICA DEL NORTE
DIRECCIÓN DE BIBLIOTECA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
APELLIDOS Y NOMBRES:	PANAMÁ TUQUEREZ DIANA SOLEDAD

DATOS DE LA OBRA	
TÍTULO:	"SISTEMA IOT PARA EL CONTROL DE PLAGAS EN EL CULTIVO DE LA MORA DE CASTILLA "RUBUS GLAUCUS BENTH" EN LA COMUNIDAD SAN FRANCISCO DE GUACHINGUERO"
AUTOR (ES):	DIANA SOLEDAD PANAMÁ TUQUEREZ
FECHA: DD/MM/AAAA	16/07/2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** □ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERA EN TELECOMUNICACIONES
DIRECTOR /ASESOR:	ING. JAIME ROBERTO MICHILENA CALDERÓN, MSC ING. SUÁREZ ZAMBRANO LUIS EDILBERTO, MSC

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 16 días del mes de julio de 2026

EL AUTOR:

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
DIANA SOLEDAD
PANAMA TUQUEREZ

Diana Soledad Panamá Tuquerez

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 16 de julio de 2026

ING. JAIME ROBERTO MICHILENA CALDERÓN. MSC
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
JAIME ROBERTO MICHILENA CALDERON

……
Ing. Jaime Roberto Michilena Calderón. Msc
C.C.: 1002198438

DEDICATORIA

Primeramente, a Dios, por ser mi guía durante esta etapa tan importante de mi vida, por ser mi ejemplo a seguir en perseverancia, y por enseñarme que con paciencia, obediencia y dedicación todo se puede lograr.

A mi papá Luis, quien ha sido un ejemplo a seguir y siempre ha estado ahí para mí, demostrándome que cuando uno lucha y tiene objetivos claros, estos se cumplen, sin rendirse a medio camino.

A mi querida madre Juanita, que siempre ha estado ahí, apoyándome, enseñándome y acompañándome incondicionalmente durante este proceso.

A mis hermanos, por ser mi guía, por animarme a seguir adelante y por acompañarme incondicionalmente.

A mi querido compañero de vida, por ser mi apoyo incondicional y por impulsarme siempre a seguir adelante.

Diana Soledad Panamá Tuquerez

AGRADECIMIENTO

Agradezco, en primer lugar, a Dios, por permitirme culminar esta importante etapa de mi vida universitaria, por darme fortaleza para enfrentar cada desafío y por bendecirme con unos padres que han sido mi mayor guía, apoyo y ejemplo a lo largo de este camino.

Expreso mi más sincero agradecimiento a mi director de trabajo de titulación, Ing. Jaime Michilena, por ser una guía incondicional durante todo este proceso. Gracias por su paciencia, dedicación, tiempo y valiosa orientación en cada etapa de esta investigación. Su confianza en mis capacidades, sus consejos y su constante motivación fueron fundamentales para alcanzar esta meta.

De igual manera, agradezco al Ing. Suárez, por su paciencia, compromiso y disposición para compartir conmigo sus conocimientos y experiencia. Gracias por acompañarme en este proceso, resolver mis inquietudes y brindarme su apoyo incluso en los momentos más complejos. Su guía fue esencial para el desarrollo y culminación exitosa de este trabajo.

Finalmente, agradezco a las personas que estuvieron a mi lado durante este camino, especialmente a Israel y Danny, por su apoyo incondicional, su amistad y sus palabras de aliento en los momentos de mayor dificultad. Gracias por motivarme a seguir adelante, por creer en mí y por recordarme que, con esfuerzo y perseverancia, es posible alcanzar cualquier meta.

A todas las personas que, de una u otra manera, contribuyeron a la realización de este trabajo, les expreso mi más profundo agradecimiento.

Panama Tuquerez Diana Soledad

RESUMEN

La mora de Castilla (“Rubus glaucus Benth”), es uno de los cultivos más importantes de las zonas andinas del Ecuador. En la comunidad San Francisco de Guachinguero, parroquia San José de Quichinche, cantón Otavalo, los agricultores han identificado al mildiu polvoso como la enfermedad que más daño causa el amarillamiento de las hojas en sus plantaciones. Ante esta necesidad, este trabajo implementó un sistema IoT capaz de detectar el mildiu polvoso en hojas de mora de Castilla en etapa post-siembra, con una altura de entre 45 y 50 cm, generando alertas automáticas y recomendaciones de tratamiento directamente al agricultor. El sistema está compuesto por una Raspberry Pi 4, una cámara OV5647 y un modelo de inteligencia artificial YOLOv8n entrenado con 1.000 imágenes propias de hojas sanas y enfermas. Adicionalmente, se aplicó segmentación de color en el espacio HSV mediante OpenCV para calcular el porcentaje exacto de tejido afectado, clasificando la severidad en cuatro niveles: Preventivo, Alerta, Peligro y Crítico, cada uno con una recomendación específica de tratamiento.

El modelo entrenado con 50 épocas fue el que mejores resultados entregó, con una precisión de 0.987, un recall de 0.986 y un mAP@50 de 0.993. Los resultados de cada análisis se envían en tiempo real a Firebase y se visualizan en una aplicación web accesible desde cualquier dispositivo con internet. Durante las pruebas realizadas por cuatro semanas el sistema procesó 29 análisis en campo real, detectando oportunamente el único caso de infección activa registrado el 21 de mayo con una severidad del 5.3%, lo que confirma que el sistema funciona correctamente como apoyo al manejo fitosanitario del cultivo.

Palabras clave: Mora de Castilla, Segmentación HSV, YOLOv8n, Mildiu Polvoso, OpenCV y (Internet de las Cosas) IoT.

ABSTRACT

Blackberry (Rubus glaucus Benth) is one of the most important crops in the Andean regions of Ecuador. In the community of San Francisco de Guachinguero, San José de Quichinche Parish, Otavalo, Ecuador, farmers have identified powdery mildew as the disease that causes the most severe leaf yellowing in their plantations. To address this need, this research implemented an IoT-based system capable of detecting powdery mildew in blackberry leaves during the post-planting stage, when plants reach a height of 45 to 50 cm, generating automatic alerts and treatment recommendations directly for farmers.

The system consists of a Raspberry Pi 4, an OV5647 camera, and a YOLOv8n artificial intelligence model trained with 1,000 self-collected images of healthy and infected leaves. Additionally, color segmentation in the HSV color space was applied using OpenCV to calculate the exact percentage of affected tissue, classifying disease severity into four levels: Preventive, Alert, Danger, and Critical, each associated with a specific treatment recommendation.

The model trained for 50 epochs achieved the best performance, obtaining a precision of 0.987, a recall of 0.986, and an mAP@50 of 0.993. The results of each analysis are transmitted in real time to Firebase and displayed through a web application accessible from any internet-connected device. During four weeks of field testing, the system processed 29 analyses under real agricultural conditions, successfully detecting the only active infection case recorded on May 21, with a severity level of 5.3%. These results confirm that the proposed system functions effectively as a decision-support tool for the phytosanitary management of blackberry crops.

Keywords: Blackberry, HSV Segmentation, YOLOv8n, Powdery Mildew, OpenCV, Internet of Things (IoT).

1 LISTA DE CONTENIDO

1 LISTA DE CONTENIDO 17
1 CAPÍTULO I. ANTECEDENTES 23
1.1 Tema 23
1.2 Problema 23
1.3 Objetivos 26
1.3.1 Objetivo general 26
1.3.2 Objetivos específicos 26
1.4 Alcance 26
1.5 Justificación 29
2 CAPÍTULO II. FUNDAMENTACIÓN TEÓRICA 32
2.1 Cultivo de la Mora de Castilla 32
2.1.1 Origen 33
2.1.2 Características de la planta de mora 34
2.1.3 Ciclo de Crecimiento de la Mora de Castilla 35
2.2 Factores que Impactan la Productividad del Cultivo 36
2.2.1 Humedad del suelo 37
2.2.2 Temperatura ambiental 38
2.2.3 Plagas y Enfermedades 38
2.2.4 Manejo del cultivo 47
2.3 Tecnología IoT en la Agricultura 49
2.3.1 Aplicación de sistema IoT en la agricultura 49
2.3.2 Visión por Computador en la Agricultura 51
2.3.3 Protocolos y Tecnologías de Comunicación en IoT Agrícola 56
3 CAPITULO III DISEÑO E IMPLEMENTACIÓN 58
3.1 Metodología 58
3.2 Análisis 59
3.2.1 Situación actual 60
3.2.2 Recolección de información 62
3.3 Requerimientos del sistema 65
3.3.1 Determinación de stakeholders 65
3.3.2 Requrimientos del software y hardware 66
3.3.3 Nomenclatura para requisitos 67
3.3.4 Requerimientos de Stakeholders 67
3.3.5 Requerimientos de arquitectura 69

3.3.6 Requerimientos del sistema 72
3.4 Diseño y descripción del sistema 74
3.4.1 Selección de Hardware y software para el sistema 75
3.4.2 Arquitectura del sistema IoT 84
1.1.1 Diagrama de flujo del funcionamiento del sistema. 86
3.5 Desarrollo del sistema 87
3.5.1 Integración del sistema 88
3.5.2 Desarrollo del Modelo de Detección 91
3.6 Desarrollo de la aplicación 126
3.6.1 Pantalla de inicio 126
3.6.2 Historial de detecciones 129
4 CAPITULO IV PRUEBAS 133
4.1 Evaluación del modelo de red neuronal 133
4.2 Configuración y parámetros de entrenamiento del modelo. 135
4.3 Pruebas de detección 142
4.3.1 Prueba con hoja sana 142
4.3.2 Prueba con hoja enferma 145
4.3.3 Prueba con diferentes niveles de severidad 147
4.4 Pruebas de comunicación del sistema 151
4.4.1 Prueba de envío de datos a Firebase 151
4.4.2 Prueba de generación de alertas en la aplicación web 154
4.5 Discusión Resultados 161
4.6 Costo del Sistema 163
4.6.1 Costo del Hardware 163
4.6.2 Costo del Software 164
4.6.3 Costo de infraestructura 164
4.6.4 Costo Total del Sistema 165
4.6.5 Análisis de Costo- Beneficio 165
CONCLUSIONES 167
RECOMENDACIONES 168
BIBLIOGRAFÍA 169
5 ANEXO 1: Encuesta 172
5.1 Formato de la encuesta 172
6 ANEXO 2: Manual de Usuario 176
6.1 Requisitos previos 176
6.1.1 Encendido del sistema 176
6.1.2 Funcionamiento 178

6.1.3 Visualización de Resultados en la Aplicación Web 179
6.1.4 Interpretación de Alertas y Acciones Recomendadas 181
7 Anexo 3: Manual de administrador 182
7.1 Arquitectura General para el administrador 182
7.2 Requisitos del sistema 183
7.3 Librerías por componente 185
7.4 Configuración del Sistema 189
7.4.1 Configuración de la Raspberry Pi 189
7.4.2 Configuración de Firebase 190
7.4.3 Configuración del modelo 192
7.5 Mantenimiento 193

LISTA DE TABLAS

Tabla 1 Influencia negativa de las condiciones climáticas....38
Tabla 2 Preguntas de la Encuesta....63
Tabla 3 Lista de participantes en el proyecto....66
Tabla 4 Nomenclatura para condiciones....67
Tabla 5 Requerimientos de Stakeholders....68
Tabla 6 Requerimientos de Arquitectura....69
Tabla 7 Requerimientos del Sistema....72
Tabla 8 Requerimientos para elección de microcontrolador....76
Tabla 9 Requerimientos para selección de adquisición de imágenes....78
Tabla 10 Requerimientos para el software del microcontrolador....80
Tabla 11 Requerimientos de Software Procesamiento de imágenes....81
Tabla 12 Selección de tecnología inalámbrica....83
Tabla 13 Librerías utilizadas....93
Tabla 14 Características del modelo seleccionado....96
Tabla 15 Comparación de Resultados....105
Tabla 16 Librerías utilizadas en la integración del modelo en Raspberry Pi....108
Tabla 17 Niveles de severidad y recomendaciones del sistema....116
Tabla 18 Librerías utilizadas en Realgtime Database....118
Tabla 19 Estructura de nodos en Firebase Realtime Database....122
Tabla 20 Tecnologías utilizadas en el desarrollo de la aplicación web....126
Tabla 21 Características del hardware utilizado durante el entrenamiento del modelo....134
Tabla 22 Especificaciones técnicas de la Raspberry Pi 4....134
Tabla 23 Parámetros de la red neuronal....135
Tabla 24 Registro semanal de análisis del cultivo....155
Tabla 25 Costo de Hardware....164
Tabla 26 Costo de Software....164
Tabla 27 Costo Infraestructura....165
Tabla 28 Costo Total del Sistema....165
Tabla 29 Requisitos del Hardware....184
Tabla 30 Requisitos Raspberry Pi 4....184
Tabla 31 Librerías para el procesamiento de imágenes....185
Tabla 32 Librerías para inteligencia artificial....186
Tabla 33 Librerías para la comunicación con Firebase....186
Tabla 34 Librerías para sincronización del dataset....187
Tabla 35 Librerías para la interfaz gráfica....187
Tabla 36 Librerías para la administración del sistema....188
Tabla 37 Herramientas del sistema operativo para la cámara OV5647....188

LISTA DE FIGURAS

Figura 1 *Diagrama de Ishikawa: Factores que Afectan el Cultivo de la Mora de Castilla* 25
Figura 2 Diseño del Sistema IoT para el control de hongos en las plantaciones de la mora de Castilla 29
Figura 3 Frutos maduros de mora de Castilla en rama 33
Figura 4 Arquitectura IoT propuesta para el sistema de monitoreo del cultivo 50
Figura 5 Flujo general propuesta de procesamiento de imágenes agrícolas 52
Figura 6 Fases de la metodología Cascada 58
Figura 7 *Zona de cultivo de la mora de castilla* 61
Figura 8 Microcontrolador Raspberry Pi 4 77
Figura 9 Sensor de imagen Cámara OV5647 79
Figura 10 Arquitectura del sistema 85
Figura 11 Diagrama de flujo del funcionamiento del sistema 87
Figura 12 Diagrama de Sistema de adquisición de imágenes 88
Figura 13 Diagrama de bloques del Sistema IoT 89
Figura 14 Diagrama de comunicación inalámbrica 90
Figura 15 Proceso de entrenamiento del modelo Yolo. 92
Figura 16 Proceso de etiquetado del dataset para la detección de mildiu polvoso mediante bounding boxes. 94
Figura 17 Proceso de organización y división del dataset 95
Figura 18 Resumen de arquitectura del modelo YOLOv8n 96
Figura 19 Configuración de parámetros del entrenamiento del modelo YOLOv8n. 98
Figura 20 Resultados finales del entrenamiento del modelo YOLOv8n 99
Figura 21 Distribución de clases en el dataset de entrenamiento 99
Figura 22 Curvas de entrenamiento y validación del modelo YOLOv8n 102
Figura 23 Matriz de confusión normalizada 103
Figura 24 Curva Precisión-Recall 104
Figura 25 Curva F1-Confianza 104
Figura 26 Curva de aprendizaje 106
Figura 27 Proceso de exportación y subida del modelo a Firebase Storage 107
Figura 28 Integración del modelo ONNX en Raspberry Pi 4 108
Figura 29 Conexión con Firebase y política 109
Figura 30 Detección de nuevo modelo 110
Figura 31 Carga del modelo base 110
Figura 32 Descarga del modelo nuevo 111
Figura 33 Carga del modelo actualizado 111
Figura 34 112
Figura 35 Captura de Imagen 113
Figura 36 Implementación de la segmentación HSV y cálculo de severidad 114
Figura 37 Historial de detecciones en la aplicación web NexusLeaf AI 117
Figura 38 Evidencia óptica de hojas con infección activa 118
Figura 39 Código de inicialización y conexión con Firebase 119
Figura 40 Código de descarga automática del modelo ONNX desde Firebase Storage 120
Figura 41 Código de envío de resultados a Firebase Realtime Database 121

Figura 42 Estructura de Firebase Realtime Database del sistema 122
Figura 43 Modelos ONNX almacenados en Firebase Storage 123
Figura 44 Implementación del sistema IoT en el entorno de cultivo 125
Figura 45 Pantalla de inicio de la aplicación web 127
Figura 46 Panel de configuración y gestión de la aplicación web 128
Figura 47 Historial de detecciones en la aplicación web 130
Figura 48 Código de configuración y envío de parámetros a Firebase 131
Figura 49 Código de generación de recomendaciones y alertas automáticas 132
Figura 50 Gráfico de curvas de entrenamiento y perdida en 30 Épocas 136
Figura 51 Gráfico de curvas de entrenamiento y perdida en 50 Épocas 137
Figura 52 Gráfico de curvas de entrenamiento y perdida en 100 Épocas 138
Figura 53 Gráfico de curvas de entrenamiento y perdida en 75 Épocas 139
Figura 54 Matrices de confusión de red neuronal. 140
Figura 55 Segmentación con HSV Hojas sanas 143
Figura 56 Características de la Hoja Sana 143
Figura 57 Resultado del análisis automatizado 144
Figura 58 Evidencia visual del análisis óptico 144
Figura 59 Segmentación con HSV Hojas enfermas 145
Figura 60 Características de la Hoja Enferma 146
Figura 61 Resultado del análisis automatizado 146
Figura 62 Evidencia visual del análisis óptico 147
Figura 63 Clasificación, color de alerta y envío a Firebase 148
Figura 64 Resultado del análisis — Nivel Preventivo (0.3%) 149
Figura 65 Resultado del análisis — Nivel Alerta (7.9%) 150
Figura 66 Código de envío de datos a Firebase Realtime Database 152
Figura 67 Confirmación de envío de datos en la interfaz de la Raspberry Pi 153
Figura 68 Datos recibidos en Firebase Realtime Database 154
Figura 69 Encendido del Sistema 177
Figura 70 Inicia Monitoreo 178
Figura 71 Análisis de hoja 179
Figura 72 Inicio de página 180
Figura 73 Pantalla de inicio 180
Figura 74 Interpretación de Alertas y Acciones Recomendadas 181
Figura 75 Arquitectura general del sistema IoT 183
Figura 76 Ejecución de la aplicación en la Raspberry Pi4 190
Figura 77 Realtime Database 191
Figura 78 Archivos de modelo entrenado 193

1 CAPÍTULO I. ANTECEDENTES

1.1 Tema

Sistema IOT para el control de plagas en el cultivo de la mora de castilla “Rubus glaucus Benth” en la comunidad San Francisco de Guachinguero.

1.2 Problema

La mora de Castilla, también llamada mora andina (Rubus glaucus Benth), toma su nombre del latín, donde "rojo" y "blanquecino" describen el color de su fruto y el envés de sus hojas, respectivamente. Este es un cultivo frutal emergente con gran relevancia comercial (Garrido et al., 2020). En Ecuador, esta fruta prospera en las regiones de los valles del callejón interandino y las laderas de la Sierra, a altitudes que oscilan entre los 2,500 y 3,000 metros(Cevallos B., 2020). Las provincias de Tungurahua, Cotopaxi, Bolívar, Chimborazo, Pichincha, Imbabura y Carchi destacan como las principales zonas productoras (Salcedo, 2019). A pesar de ser considerada un cultivo tradicional, la mora de Castilla enfrenta crecientes desafíos debido a la proliferación de plagas y los efectos del cambio climático, que han alterado los patrones climáticos y afectado la calidad y cantidad de la fruta (Cevallos B., 2020).

En la comunidad de San Francisco de Guachinguero, ubicada en la parroquia San José de Quichinche del cantón Otavalo, los agricultores han enfrentado serios desafíos en el cultivo de la mora de Castilla. Estos desafíos incluyen la disminución de las cosechas debido a la proliferación de plagas y al impacto del cambio climático, que ha alterado los patrones tradicionales de temperatura y precipitación, afectando la calidad y cantidad del fruto. Las prácticas agrícolas tradicionales, como la fumigación con productos químicos, no han logrado mitigar estos efectos, lo que ha resultado en pérdidas significativas para los productores locales.

Gloria Tuqueres, propietaria de la plantación de la mora de Castilla, menciona que el cambio climático y la presencia de plagas en el cultivo han ocasionado pérdidas significativas,

limitando la sostenibilidad y el crecimiento de la producción. A pesar de los esfuerzos por utilizar tecnologías tradicionales de manejo, como el uso de pesticidas, los resultados no han sido satisfactorios para garantizar la estabilidad en la producción. El uso de fertilizantes químicos, aunque comúnmente adoptado, ha mostrado una eficacia limitada en la mejora del rendimiento de los cultivos.

Por otro lado, la mora de Castilla es considerada un fruto no climatérico, lo que significa que es sensible a la manipulación y susceptible a ataques de hongos, lo que agrava las pérdidas (Martínez etc al., 2019). La falta de conocimiento adecuado sobre técnicas de manejo, como la aplicación correcta de pesticidas y la identificación precisa de las plagas, ha contribuido a que los productores enfrenten graves pérdidas económicas y dificultades para mantener la estabilidad de sus cultivos (Santiago & Barona, 2018).

Según (IniAP, 2011), para abordar el desafío de las plagas en los cultivos, es esencial implementar un control adecuado, comenzando con la correcta identificación del agente causal, ya sea insecto o ácaro, y considerando factores como la dispersión de la plaga, su comportamiento frente a las condiciones ambientales y las prácticas de manejo del cultivo.

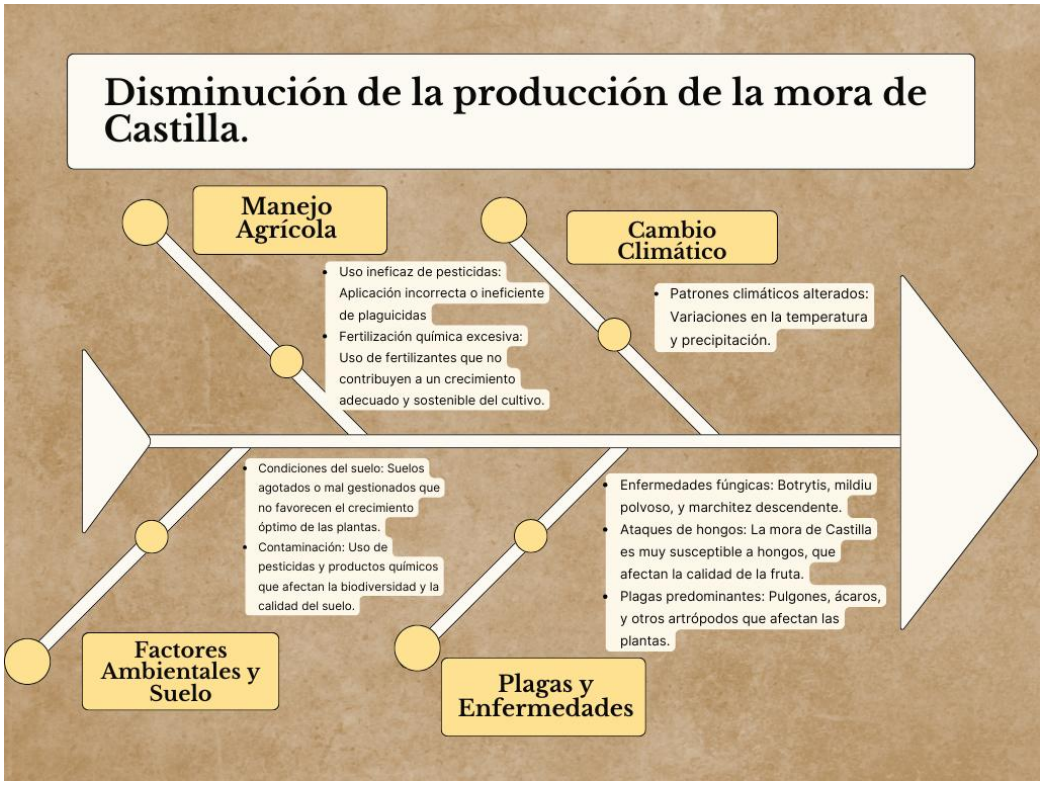
El Manejo Integrado de Plagas (MIP) se presenta como una estrategia clave, ya que no solo se enfoca en el uso de plaguicidas, sino en el aprovechamiento de recursos como variedades resistentes, organismos benéficos y prácticas culturales adecuadas, lo que asegura un manejo racional y sostenible. Las moras a menudo enfrentan más de una enfermedad a la vez, lo que agrava los daños al cultivo y aumenta la dificultad de su control. Estudios realizados en Tungurahua, Cotopaxi y Bolívar han identificado las principales enfermedades del cultivo de la mora de Castilla, siendo Botrytis (91,8 %) la más dañina, seguida por mildiu polvoso (61,2 %) y marchitez descendente (13,3 %), mientras que los pulgones (63,3 %) son la plaga

predominante entre los artrópodos, seguidos por los ácaros (45,9 %) y los cutzos (10,2 %) (Villares, 2016).

A continuación, en la figura 1, se muestra un diagrama de Ishikawa (o espina de pescado) enfocado en los problemas que afectan el cultivo de la mora de Castilla. En el centro, el efecto principal identificado es la disminución de la producción y calidad del cultivo. Las principales causas se dividen en varias categorías representadas por espinas principales.

Figura 1

Diagrama de Ishikawa: Factores que Afectan el Cultivo de la Mora de Castilla



Nota: El gráfico representa las causas principales de la disminución en la Producción de Mora de Castilla.

La implementación de un sistema IoT para el control de hongos en las hojas del cultivo de la mora de Castilla facilita la identificación de signos visibles de afectaciones fúngicas, permitiendo a los agricultores observar el estado del cultivo de manera precisa y oportuna.

1.3 Objetivos

1.3.1 Objetivo general

Implementar un sistema IoT para el control de hongos en las hojas del cultivo de mora de Castilla en la comunidad de San Francisco de Guachinguero, con el fin de generar alertas que apoyen el manejo oportuno del cultivo.

1.3.2 Objetivos específicos

- Investigar los aspectos relacionados con las plantaciones de la mora de Castilla, abarcando el estudio del crecimiento de las plantas, el análisis de métodos efectivos para el control de hongos y la integración de sistemas IoT para el monitoreo del cultivo.
- Definir los requerimientos técnicos y ambientales necesarios para diseñar un sistema IoT que permita el monitoreo y control de hongos en las plantaciones de la mora de castilla.
- Diseñar un sistema IoT para monitorear y controlar la presencia de hongos en las plantaciones de la mora de Castilla en la comunidad de San Francisco de Guachinguero.
- Realizar pruebas del sistema IoT, verificando su capacidad para detectar signos visibles de hongos en las plantaciones de la mora de Castilla.

1.4 Alcance

El presente proyecto tiene como objetivo principal implementar un sistema IoT orientado al control de hongos en el cultivo de la mora de Castilla en la comunidad de San Francisco de Guachinguero, ubicada en la parroquia San José de Quichinche, cantón de Otavalo. Este sistema utilizará tecnología de visión por computadora basada en inteligencia artificial para la detección precisa de estas afecciones, generando alertas en tiempo real que apoyarán el manejo y control oportuno de las plagas, específicamente hongos que ocasionan el amarillamiento de las hojas y generará notificaciones en tiempo real que recomendarán acciones específicas, como la aplicación de fertilizantes químicos u orgánicos, para mejorar la salud de las plantas.

Para su desarrollo, se empleará la metodología en cascada, que comprende las fases de investigación, definición de requerimientos, diseño e implementación del sistema IoT, y su evaluación final mediante pruebas de campo.

- **Fase I: Investigación y análisis**

La fase de investigación abarcara un análisis exhaustivo del cultivo de mora de Castilla, evaluando el crecimiento del cultivo, la incidencia de los hongos que provocan el amarillamiento de las hojas, y su impacto en la calidad y productividad. También se estudiarán las condiciones del suelo (humedad, nutrientes, deficiencias) con el objetivo de identificar estrategias de mejoramiento. Asimismo, se explorarán soluciones tecnológicas sostenibles, como la integración de sistemas IoT y métodos de inteligencia artificial, con el fin de implementar un sistema eficiente de monitoreo. Esta etapa permitirá comprender integralmente el ciclo de vida del cultivo, las características del entorno, y los factores clave que influyen en su salud, estableciendo así una base sólida para el diseño de estrategias de manejo agrícola.

- **Fase II: Requerimientos**

En esta etapa se definirán los requerimientos técnicos del sistema, incluyendo la selección de sensores y cámaras para la captura de datos visuales, y microcontroladores para el procesamiento de la información. Se determinarán los protocolos de comunicación más adecuados para asegurar una transmisión eficiente de los datos en tiempo real. Este enfoque permitirá la detección oportuna de hongos y facilitará la toma de decisiones en campo, optimizando el rendimiento del cultivo y respondiendo a las necesidades de los agricultores.

- **Fase III: Diseño e implementación**

En la tercera fase, se diseñará e implementará un sistema IoT integrado, enfocado en el monitoreo ambiental y la detección de hongos en el cultivo de la mora de Castilla en la comunidad de San Francisco de Guachinguero. Este sistema será capaz de identificar las plagas

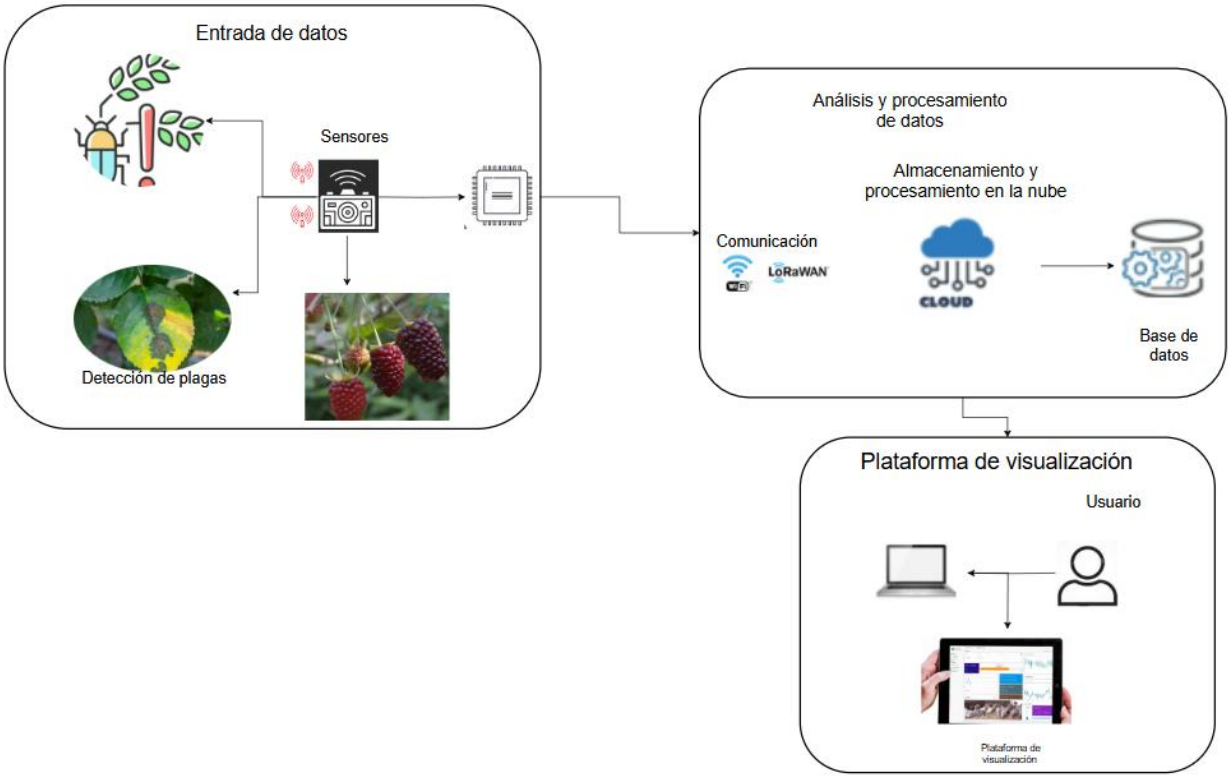
fúngicas responsables del amarillamiento de las hojas, generando alertas en tiempo real para apoyar el manejo oportuno del cultivo.

Durante esta etapa, se seleccionarán componentes necesarios, incluyendo cámaras para el análisis visual especializadas para identificar hongos. Los datos obtenidos serán procesados y almacenados mediante microcontroladores, garantizando un monitoreo continuo y confiable. También se definirán formas adecuados para que el sistema registre los datos obtenidos por los sensores, permitiendo visualizar la información de manera clara y apoyar el análisis sobre la presencia de hongos en el cultivo. Además, se establecerán mecanismos que aseguren que el sistema funcione de manera eficiente, cumpliendo con los objetivos del proyecto y contribuyendo al control de plagas en el cultivo de la mora de Castilla.

En la Figura 2 se ilustra el diseño del Sistema IoT para el control de hongos en las plantaciones de la mora de Castilla.

Figura 2

Diseño del Sistema IoT para el control de hongos en las plantaciones de la mora de Castilla



Nota: Esta imagen muestra el diseño del sistema IoT implementado para el control de hongos en las plantaciones de mora de Castilla.

- **Fase IV: Pruebas**

Durante esta etapa se evaluará el desempeño del sistema IoT implementado, poniendo énfasis en su capacidad para detectar signos visibles de hongos que afectan las hojas del cultivo. Se analizará la precisión del sistema, la utilidad de las alertas generadas y la eficacia de las recomendaciones emitidas, como el uso de fertilizantes químicos u orgánicos. Los datos obtenidos permitirán realizar los ajustes necesarios para optimizar su funcionamiento. Esta fase garantizará un monitoreo continuo y confiable que contribuirá al manejo oportuno de las afecciones del cultivo de mora de Castilla en la comunidad de San Francisco de Guachinguero.

1.5 Justificación

El presente proyecto está orientado a la producción de mora de Castilla en la comunidad de San Francisco de Guachinguero, con un enfoque en la implementación de un sistema IoT

que integre el monitoreo de plagas, específicamente hongos. Este enfoque no solo responde a las necesidades locales de los agricultores, sino que también se alinea con objetivos más amplios, como los Objetivos de Desarrollo Sostenible (ODS), ODS 9 (Industria, Innovación e Infraestructura) y ODS 12 (Producción y Consumo Responsables) (Naciones, 2024). La aplicación de tecnologías innovadoras en el ámbito agrícola permite mejorar la eficiencia en el uso de recursos naturales y aumentar la productividad, contribuyendo a la seguridad alimentaria y al desarrollo sostenible de la región. (Naciones, 2024)

El Gobierno Provincial de Imbabura ha manifestado un interés continuo en promover el desarrollo económico local mediante la mejora de la productividad agrícola, especialmente en el sector de cultivos clave como la mora, que tiene un gran potencial económico en la región (Prefectura de Imbabura, 2020). Este proyecto se alinea con las políticas del Gobierno Provincial, que busca mejorar la competitividad de los agricultores de Imbabura mediante el uso de tecnologías innovadoras que optimicen los recursos naturales y fortalezcan la seguridad alimentaria en la zona (Prefectura de Imbabura, 2020).

Además, el proyecto responde a las directrices establecidas en la Agenda Zonal de desarrollo de Imbabura, que fomenta la integración de nuevas tecnologías en la agricultura, contribuyendo así a una gestión más eficiente de los recursos hídricos y al manejo sostenible de las plagas. El sistema IoT propuesto ayudará a los agricultores a adoptar tecnologías de punta, mejorando sus prácticas de manejo agrícola y aumentando su resiliencia frente a los desafíos ambientales y económicos.(Ministerio de Agricultura, 2015)

La implementación del sistema IoT para el monitoreo de hongos causantes del amarillamiento de las hojas en la mora de Castilla contribuirá a enriquecer el conocimiento sobre el cultivo, mejorando la salud del suelo y promoviendo prácticas agrícolas sostenible (Suquilanda Valdivieso, 2011). Así, la iniciativa fortalecerá el desarrollo socioeconómico local,

reforzando la resiliencia agrícola y posicionando a San Francisco de Guachinguero como un referente en producción sostenible de la mora de Castilla.

2 CAPÍTULO II. FUNDAMENTACIÓN TEÓRICA

En este capítulo se analizan los aspectos fundamentales para lograr una gestión sostenible del cultivo de la mora de Castilla, con especial atención a factores clave como los requisitos climáticos, edáficos y la implementación de tecnologías avanzadas como el Sistema IoT. Este análisis incluye una descripción detallada del ciclo de vida del hongo, los métodos de propagación y las estrategias de control diseñadas para mitigar su impacto y preservar la salud del cultivo.

2.1 Cultivo de la Mora de Castilla

El cultivo de la mora de Castilla (Rubus glaucus Benth) es una actividad agrícola de gran importancia en las regiones andinas debido a su alto valor comercial y versatilidad en la agroindustria. Esta especie se desarrolla en climas templados a fríos, con suelos de textura franco-arenosa o franco-arcillosa, bien drenados y ricos en materia orgánica. Su manejo agronómico requiere de técnicas adecuadas para optimizar su rendimiento y garantizar la calidad de los frutos. En Ecuador, constituye uno de los principales cultivos de las provincias andinas, contribuyendo significativamente al desarrollo económico de comunidades rurales. (Arias et al., 2020)

La mora de Castilla se caracteriza por su crecimiento en arbustos de ramas espinosas, con hojas alternas y compuestas que facilitan la fotosíntesis. Sus frutos, formados por múltiples drupas, presentan una coloración que varía de rojo a púrpura oscuro en la madurez, lo que indica su óptimo estado para la cosecha. (Arias et al., 2020)

Como se muestra en la Figura 1, los frutos de la mora de Castilla en su estado de madurez presentan una tonalidad púrpura oscura, indicativa de su óptima calidad para la cosecha.

Figura 3
Frutos maduros de mora de Castilla en rama



Nota: Frutos maduros de mora de Castilla en rama, mostrando su característica tonalidad púrpura oscura, indicativa de su madurez y calidad óptima para la cosecha.

2.1.1 Origen

La mora de Castilla, descubierta por Hartw y descrita posteriormente por Benth, es originaria de las regiones tropicales altas de Centroamérica. Su distribución abarca países como México, El Salvador, Guatemala, Panamá, Colombia y Ecuador. El género *Rubus* se destaca por su gran diversidad genética y morfológica, siendo uno de los más variados del reino vegetal, con especies silvestres distribuidas a nivel mundial, exceptuando las zonas desérticas. .(Valdez et al., 2023)

Actualmente, *Rubus glaucus* B. es la especie más cultivada con fines comerciales en Sudamérica. En Ecuador, se posiciona como uno de los frutales andinos de mayor relevancia, con cultivos establecidos en el callejón interandino, particularmente en provincias como Cotopaxi, Bolívar, Chimborazo, Tungurahua, Pichincha, Imbabura y Carchi. Esta fruta es

apreciada tanto para el consumo fresco como para su procesamiento en la agroindustria, donde se utiliza en la producción de mermeladas, jarabes, postres, entre otros. Por ello, los agricultores priorizan cultivares que ofrezcan temporadas prolongadas de fructificación, frutas de alta calidad y un rendimiento superior. (Valdez et al., 2023)

2.1.2 Características de la planta de mora

La planta de mora se compone de múltiples tallos espinosos cubiertos por una capa blancuzca. Estos tallos tienen un diámetro que varía entre 1,5 y 2,5 cm, y una longitud de entre 3 y 4 metros. Son ligeramente flexibles y, por lo general, necesitan soportes para mantenerse erguidos. Cada planta puede generar 20 o más retoños; sin embargo, en los cultivos solo se conservan aquellos que están en producción o que se utilizarán para la reproducción. Al realizar la poda de los tallos, se estimula la aparición de ramas secundarias. (Santiago & Barona, 2018)

Las hojas son compuestas y actinomorfas, típicamente en forma piriginosa, con flores de color blanco que miden entre 2,0 y 2,5 cm de diámetro, y se agrupan en racimos ubicados en las puntas de las ramas, o a veces, a lo largo de toda la rama. (Alexandra Mora-Ramos et al., 2020)

Cada flor tiene 5 sépalos permanentes, 5 pétalos y numerosos estambres, además de carpelos que están libres y unidos al receptáculo. Cada carpelo está formado por un ovario, dos óvulos y un pistilo largo. Las ramas florecen en racimos terminales. (Alexandra Mora-Ramos et al., 2020)

El fruto es una estructura agregada, compuesta por entre 70 y 100 drupas que permanecen adheridas al receptáculo. Cada drupa contiene una semilla y el total de semillas por fruto varía entre 100 y 120. Los frutos tienen una forma esférica o elipsoidal y pueden variar en tamaño, desde grande hasta pequeño. Maduran de manera desigual debido a que la floración no es homogénea. Tienen un tamaño que oscila entre 1,5 y 2,5 cm de largo y 1,5 a

2,0 cm de ancho, y su color cambia de rojo a púrpura o rojo oscuro cuando alcanzan la madurez. La producción de frutos es constante, aunque presenta un mayor pico de producción en intervalos de 5 a 6 meses. (Santiago & Barona, 2018)

Las raíces son filiformes, con forma de racimo y poco profundas, distribuyéndose principalmente en los primeros 30 cm del suelo. Estas raíces tienen una disposición horizontal y se componen de rizomas secundarios, con una longitud que varía entre 0,5 y 1,2 m. La raíz se origina en el cuello cicatrizal y, a partir de las estacas y acodos, facilita la propagación al presentar yemas vegetativas que se activan para generar nuevos brotes. (Santiago & Barona, 2018)

2.1.3 Ciclo de Crecimiento de la Mora de Castilla

El ciclo de crecimiento de la mora de Castilla (*Rubus glaucus Benth*) abarca distintas fases que determinan su desarrollo, productividad y longevidad dentro del sistema agrícola. Estas etapas incluyen la germinación, el crecimiento vegetativo, la floración, la fructificación y la senescencia. Cada una de ellas está influenciada por factores ambientales como la humedad del suelo, la temperatura, la calidad del sustrato y las prácticas de manejo agronómico implementadas.(Cevallos B., 2020)

El ciclo de crecimiento de la mora de Castilla se divide en varias fases clave:

- **Germinación:** En esta fase, las semillas requieren condiciones óptimas de humedad y temperatura para iniciar su desarrollo. Una adecuada disponibilidad de agua y temperaturas entre 18 y 25°C favorecen la rápida emergencia de las plántulas, asegurando un establecimiento vigoroso del cultivo.(Franco et al., 2020)

- **Crecimiento vegetativo:** Durante este período, la mora de Castilla demanda una cantidad significativa de nutrientes, especialmente nitrógeno, fósforo y potasio, los cuales favorecen el crecimiento de la biomasa foliar y el fortalecimiento del sistema

radicular. Esta fase es determinante para el éxito del cultivo, ya que influye directamente en su capacidad para absorber nutrientes y agua, estableciendo una base sólida para la producción de frutos.(Franco et al., 2020)

- **Floración:** Este proceso depende de factores como la temperatura, la disponibilidad de agua y la polinización, siendo fundamental la presencia de insectos polinizadores para optimizar la producción. La floración es un indicador de que la planta ha completado su crecimiento vegetativo y está lista para generar su cosecha.(Alexandra Mora-Ramos et al., 2020)

- **Fructificación:** Etapa en la que los frutos comienzan a desarrollarse y evolucionan en su coloración, pasando de verde a rojo y finalmente a púrpura oscuro, lo que indica su estado de madurez y cosecha óptima. En esta fase, la aplicación de riegos adecuados y una nutrición balanceada garantizan la calidad y el tamaño del fruto.(Alexandra Mora-Ramos et al., 2020)

- **Senescencia:** Durante esta etapa, es común que algunas partes de la planta envejezcan y se deterioren, por lo que es necesario realizar podas de renovación para estimular la aparición de nuevos brotes productivos y prolongar la vida útil del cultivo.(Hurtado et al., 2019)

2.2 Factores que Impactan la Productividad del Cultivo

El cultivo de mora de Castilla (*Rubus glaucus Benth*) depende de múltiples factores ambientales y edáficos que determinan su rendimiento y calidad. La interacción entre las características físicas, químicas y biológicas del suelo, junto con las condiciones climáticas, juega un papel crucial en el desarrollo de la planta. Aspectos como la humedad del suelo, la temperatura ambiental y la calidad del aire afectan directamente el crecimiento de raíces, la absorción de nutrientes y la fructificación. Identificar y manejar adecuadamente estos factores

es fundamental para garantizar una producción sostenible y competitiva, especialmente en regiones con condiciones variables como los Andes ecuatorianos.

2.2.1 Humedad del suelo

La humedad del suelo es un factor determinante en el desarrollo y productividad del cultivo de la mora de Castilla, ya que influye directamente en la absorción de nutrientes, la fotosíntesis y el metabolismo de la planta. Un suelo con niveles adecuados de humedad permite que las raíces se desarrollen de manera óptima, facilitando el transporte de agua y nutrientes esenciales hacia el resto de la planta.(Valverde, 2016)

Para garantizar un adecuado crecimiento de la mora de Castilla, el suelo debe mantener un equilibrio entre la retención de humedad y el drenaje. Los suelos con textura franco arcillosa son los más recomendados, ya que permiten retener suficiente agua sin generar encharcamiento, lo que podría favorecer el desarrollo de enfermedades fúngicas.(Franco et al., 2020)

La cantidad de agua disponible en el suelo debe distribuirse de manera uniforme a lo largo del año, evitando tanto la sequía como el exceso de humedad. Cuando el suministro de agua es insuficiente, los frutos pueden presentar un tamaño reducido, menor contenido de azúcar y una coloración irregular, afectando su calidad comercial. En contraste, un exceso de humedad, especialmente en suelos arcillosos con drenaje deficiente, puede ocasionar asfixia radicular y predisponer a la planta a enfermedades como el botritis o la pudrición de raíces.(Cevallos B., 2020)

En regiones donde las precipitaciones son elevadas o se concentran en ciertos períodos del año, es esencial implementar sistemas de drenaje eficientes, como zanjas o canales, para evitar la acumulación de agua en el suelo. Asimismo, la aplicación de técnicas de manejo hídrico, como el uso de cobertura vegetal o riego controlado, contribuye a mantener niveles

óptimos de humedad en el suelo y mejorar la eficiencia del uso del agua en el cultivo. (Valverde, 2016)

2.2.2 Temperatura ambiental

Según (Valverde, 2016) afirma que, en Ecuador, la mora de Castilla se cultiva principalmente en regiones de clima frío moderado, ubicadas entre los 2,800 y 3,100 msnm, con temperaturas promedio de 14 °C, donde ocasionalmente pueden ocurrir heladas. También se encuentra en zonas de expansión como valles subtropicales, que presentan temperaturas más altas y precipitaciones insuficientes para el cultivo, lo que puede generar déficit hídrico en ciertas temporadas, o en áreas de estribación con alta pluviosidad, donde el exceso de agua afecta la aireación de las raíces.

Estas condiciones específicas generan, en cada región, problemas relacionados con la disponibilidad de nutrientes debido a factores climáticos (Ernest, 2010).

La tabla 1 muestra cómo diversos factores climáticos limitan la absorción de nutrientes y afectan el desarrollo normal de la planta.

Tabla 1
Influencia negativa de las condiciones climáticas

Factor	Elementos					
	Manganeso	**Cobre**	**Zinc**	**Hierro**	**Boro**	**Molibdeno**
Frio	x		x	x		
Asfixia radicular	x		x	x		
Sequia	x				x	x
Intensa luminosidad					x	
Aireación escasa						

Nota: La tabla muestra la relación entre factores ambientales adversos y la disponibilidad de nutrientes en las plantas. Tomado de (Ernest, 2010)

2.2.3 Plagas y Enfermedades

Las plagas y enfermedades representan un desafío significativo para la productividad del cultivo de mora de Castilla (*Rubus glaucus Benth*). Estos factores, cuando no se gestionan

adecuadamente, pueden causar daños graves en las raíces, tallos, hojas y frutos, afectando tanto la calidad como la cantidad de la producción. Las plagas, como insectos y ácaros, se convierten en una amenaza al alimentarse de distintas partes de la planta y facilitar la entrada de patógenos. Por otro lado, las enfermedades son provocadas por agentes como hongos, bacterias y virus, los cuales alteran el funcionamiento fisiológico de las plantas, reduciendo su vigor y capacidad productiva.

2.2.3.1 Plagas

En la naturaleza, las poblaciones de insectos se mantienen en equilibrio gracias a interacciones con otros organismos, como depredadores, parásitos y factores ambientales. Sin embargo, cuando el ser humano altera estos ecosistemas al destruir bosques, modificar hábitats o implementar prácticas agrícolas intensivas, se rompe este equilibrio. Esto permite que algunos insectos encuentren condiciones favorables (abundancia de alimento y ausencia de depredadores naturales) para proliferar en exceso, lo que los convierte en plagas.(Germán, 2020)

Las plagas representan un desafío significativo para el cultivo de mora de Castilla (*Rubus glaucus Benth*), ya que afectan diversas partes de la planta, como raíces, tallos, hojas y frutos, impactando negativamente su desarrollo, rendimiento y calidad. A continuación, se describen las principales plagas según la parte de la planta afectada.

- **Plaga que Atacan la Raíz:** La Perla de la Tierra es una plaga que afecta principalmente a las raíces de la mora de Castilla. Este insecto se adhiere a las raíces, alimentándose de la savia, lo que provoca la formación de nudosidades o quistes que interfieren en el transporte de agua y nutrientes. Como consecuencia, las plantas afectadas presentan un desarrollo lento, baja producción de frutos y una reducción en su calidad, ya que estos suelen ser pequeños, secos y de escaso valor comercial. (Germán, 2020)

- **Plagas que Atacan Tallos y Ramas:** El Barrenador del Cuello de la Planta es una plaga que afecta los tallos y ramas de la mora de Castilla. Sus larvas perforan la base del tallo, creando galerías internas que debilitan la estructura de la planta y dificultan el transporte de agua y nutrientes. Este daño provoca un engrosamiento del tallo y un aspecto corchoso característico, lo que detiene el crecimiento vegetativo y reduce significativamente la producción, comprometiendo el rendimiento del cultivo. (Germán, 2020)

De manera similar, El Pasador de Raíces y Ramas es una plaga que afecta severamente los tallos y ramas de la mora de Castilla. Este insecto realiza perforaciones en estas estructuras, dejando depósitos fecales visibles en los orificios como evidencia de su actividad. El daño causado por estas perforaciones avanza rápidamente, interrumpiendo el transporte de agua y nutrientes, lo que resulta en el marchitamiento progresivo y, finalmente, en la muerte de las ramas afectadas. Este daño puede comprometer gravemente la vitalidad de la planta y su capacidad productiva. (Germán, 2020).

- **Plagas que Atacan las Hojas:** Los áfidos o pulgones son insectos que se alimentan de la savia de las hojas jóvenes, provocando deformaciones, enrollamiento y deteniendo el crecimiento de la planta. Además, secretan una sustancia azucarada conocida como miel de rocío, la cual crea un ambiente propicio para el desarrollo de hongos que afectan aún más la salud de la planta y reducen su productividad. (Germán, 2020).

Por otro lado, la arañita roja ataca principalmente durante el verano, concentrándose en la parte inferior de las hojas más viejas. Su actividad genera decoloraciones blanquecinas y rojizas que, progresivamente, conducen al secado y caída de las hojas. En casos de infestaciones severas, esta plaga también puede dañar los frutos, comprometiendo la calidad y el rendimiento del cultivo.(Germán, 2020)

- **Plagas que Atacan los Frutos:** Las plagas que atacan los frutos de la mora de Castilla representan una amenaza significativa para la calidad y productividad del cultivo, ya que generan daños físicos y facilitan la entrada de microorganismos patógenos que aceleran la descomposición. Entre las más comunes se encuentran la mosca de la fruta y los chinches chupadores de frutos.

La mosca de la fruta pone sus huevos en frutos maduros, y sus larvas se alimentan del interior, lo que causa la caída prematura y la descomposición de estos. Este daño interno no solo afecta la calidad del fruto, sino que también abre vías para la entrada de microorganismos que agravan el deterioro (Germán, 2020).

Por otro lado, los chinches chupadores de frutos perforan la superficie de los frutos y hojas, dejando cicatrices visibles que reducen su calidad comercial. Estas deformaciones hacen que los frutos sean menos atractivos para el mercado, afectando directamente su valor y aceptación (Germán, 2020).

2.2.3.2 Enfermedades

Según (Germán, 2020), afirma que las enfermedades en el cultivo de mora de Castilla (Rubus glaucus Benth) son causadas por diversos agentes patógenos, como hongos, bacterias y virus, que afectan severamente la salud de las plantas y la calidad de los frutos. Estas enfermedades representan una de las principales causas de pérdidas económicas en la producción debido a la disminución del rendimiento y la calidad del cultivo.

Las enfermedades que afectan el cultivo de mora de Castilla pueden ser clasificadas en tres grandes grupos según su agente causal:

- **Enfermedades Fúngicas (Hongos):** Las enfermedades fúngicas son las más comunes en el cultivo de mora de Castilla y se caracterizan por afectar hojas, tallos y frutos,

reduciendo la capacidad fotosintética de la planta y su producción. Algunas de las más relevantes incluyen:

- **Mildiu velloso**

El Mildiu Velloso es una enfermedad causada por el patógeno Peronospora, que afecta hojas, tallos, pedúnculos y frutos de la mora de Castilla. Este hongo pertenece al grupo de los oomicetos y se desarrolla principalmente en condiciones de alta humedad relativa (superior al 80 %) y temperaturas moderadas a frías, entre 17 °C y 22 °C. (Hurtado et al., 2019)

Los síntomas más destacados incluyen:

En las hojas: Aparecen manchas amarillas o cloróticas en el borde, que progresan hacia tonos púrpuras. Las hojas severamente infectadas tienden a caerse prematuramente (Hurtado et al., 2019).

En los frutos: Los frutos infectados durante su desarrollo pierden brillo, presentan maduración desigual, deformaciones y endurecimiento prematuro. En etapas avanzadas, los frutos pueden abrirse desde la porción apical (Hurtado et al., 2019).

En los tallos y pecíolos: Se observan manchas moradas que se oscurecen con el tiempo, a menudo acompañadas de ampollas blanquecinas que contienen estructuras reproductivas del hongo (Hurtado et al., 2019).

- **La Roya**

La Roya es una enfermedad causada por el hongo *Gerwasia lagerheimii (Magnus) Buriticá*. Este patógeno es un parásito obligado altamente especializado, capaz de producir hasta cinco estructuras fructíferas diferentes, lo que complica su identificación y control. Afecta principalmente a las hojas, tallos, flores y frutos del cultivo de mora de Castilla, generando pústulas de color anaranjado y síntomas asociados como agrietamientos en los tallos y polvo

naranja en los frutos. Esta enfermedad es común en áreas con condiciones climáticas húmedas y sistemas de cultivo deficientes en manejo agronómico. (Benavides, 2021)

Los síntomas de la roya varían según la parte de la planta afectada. n las hojas, se forman lesiones hundidas rodeadas de un borde violáceo en el haz, mientras que en el envés se desarrollan pústulas anaranjadas. Estas lesiones pueden extenderse a otras partes de la planta, como tallos y pecíolos, causando deformaciones y reduciendo la funcionalidad de los tejidos afectados. En los frutos y flores, el polvo naranja característico dificulta el desarrollo normal y afecta la calidad comercial del cultivo.(Benavides, 2021)

La incidencia de la Roya está directamente relacionada con condiciones ambientales específicas, como alta humedad relativa y temperaturas moderadas. La presencia de malezas en el cultivo y la falta de podas regulares o fertilización adecuada agravan su impacto. Estas condiciones generan un ambiente propicio para la proliferación del hongo y la rápida propagación de la enfermedad.(Benavides, 2021)

- **Antracnosis**

La antracnosis es una enfermedad causada por el hongo Colletotrichum, que afecta principalmente a tallos, ramas, hojas y frutos.

Se manifiesta como manchas oscuras o grisáceas en tallos y ramas, las cuales pueden evolucionar a lesiones necrosadas. En los brotes y botones florales, provoca necrosis que interrumpe su desarrollo. En los frutos, genera manchas hundidas de color marrón, deshidratación, necrosis y pudrición húmeda. Estas lesiones disminuyen la calidad comercial del cultivo. Las condiciones de alta humedad y temperaturas entre 20 y 25 °C favorecen la proliferación de esta enfermedad. .(Benavides, 2021)

- **Botrytis cinerea**

La pudrición del fruto, causada por *Botrytis cinerea* y conocida como moho gris, es una de las enfermedades más destructivas que afectan el cultivo de mora de Castilla (*Rubus glaucus*). Este hongo fitopatógeno ataca principalmente los frutos en etapas de maduración, causando su pudrición o momificación. En casos graves, puede destruir racimos enteros, reduciendo drásticamente la producción. Las condiciones de alta humedad y temperaturas moderadas favorecen su desarrollo, especialmente en climas tropicales y fríos.(Abril Espin et al., 2024)

Botrytis cinerea es un organismo con alta variabilidad genética y fenotípica, lo que lo convierte en un sistema complejo de especies en lugar de un único patógeno. Presenta diferencias significativas en morfología, intensidad de esporulación, crecimiento micelial y producción enzimática, lo que dificulta el diseño de estrategias de control estandarizadas. Este hongo afecta desde las primeras etapas reproductivas de la planta hasta después de la cosecha, comprometiendo la calidad y la cantidad de los frutos. Su impacto económico es significativo en países como Ecuador, México y Chile, donde la mora de Castilla tiene una alta relevancia comercial, representando un desafío constante para los agricultores.(Abril Espin et al., 2024)

- **Marchitez y Pudrición de Raíces**

Es una enfermedad causada por un grupo de hongos fitopatógenos presentes en el suelo, destacándose entre ellos las especies *Verticillium*, *Fusarium* y *Rosellinia.* Estos organismos afectan principalmente el sistema radicular, debilitando las plantas de manera progresiva. Los síntomas iniciales incluyen amarillamiento de las hojas y marchitez, avanzando hacia necrosis en el sistema vascular y, finalmente, la muerte de la planta. Al realizar cortes longitudinales en los tallos afectados, se observa un oscurecimiento característico en los tejidos internos, que refleja la interrupción en el transporte de agua y nutrientes.(Benavides, 2021)

Esta enfermedad es más frecuente en suelos mal drenados o arcillosos, donde el exceso de humedad crea un ambiente propicio para la proliferación de estos hongos. Las heridas ocasionadas por prácticas agrícolas también facilitan la entrada del patógeno al sistema radicular, agravando la infección. En casos graves, la planta colapsa por completo, lo que representa pérdidas significativas en la producción del cultivo de mora de Castilla. (Restrepo, 2021)

- **Enfermedades Bacterianas**

Las enfermedades bacterianas en la mora de Castilla son menos frecuentes que las fúngicas, pueden causar serios daños estructurales a la planta, afectando su crecimiento y rendimiento. Estas enfermedades se propagan principalmente a través del suelo, el agua, herramientas contaminadas y heridas en la planta, favoreciéndose en ambientes cálidos y húmedos.

Los síntomas más comunes incluyen la aparición de tumores, marchitez foliar, necrosis y deformaciones en tallos y raíces. A continuación, se presenta una de las enfermedades bacterianas más importantes en este cultivo.

- **Agalla de la Corona**

La Agalla de la Corona, causada por la bacteria *Agrobacterium tumefaciens*, es una enfermedad que afecta principalmente al cultivo de mora de Castilla (*Rubus glaucus*). Esta bacteria penetra a las plantas a través de heridas causadas durante las prácticas agrícolas, como la poda, o por medio de herramientas contaminadas. Los síntomas principales incluyen la formación de tumores o agallas irregulares en la base del tallo o en el cuello de la raíz. Estos tumores pueden extenderse a órganos aéreos si la enfermedad se transmite mecánicamente, como ocurre con tijeras podadoras contaminadas. (Franco et al., 2020)

En etapas avanzadas, estas agallas aumentan de tamaño, invaden el tallo principal, y conducen al debilitamiento general de la planta, disminuyendo su capacidad de absorción de agua y nutrientes. Las plantas afectadas suelen presentar enanismo, clorosis, marchitamiento, y reducción en la calidad y cantidad de frutos producidos. En casos severos, la enfermedad puede causar la muerte completa de la planta.(Franco et al., 2020)

- **Mildeo Polvoso**

El Mildeo Polvoso, causado por el hongo *Oidium sp.*, es una enfermedad que afecta diversos órganos de la mora de Castilla (*Rubus glaucus*), incluyendo hojas, tallos jóvenes, botones florales y frutos. Este patógeno es fácilmente reconocible por la formación de un polvillo blanco característico que recubre las superficies afectadas. Es una de las enfermedades más comunes en ambientes con condiciones de alta humedad relativa y temperaturas moderadas, siendo un problema recurrente en regiones donde se cultiva la mora de Castilla.(Alexandra Mora-Ramos et al., 2020)

En las hojas: Las hojas jóvenes presentan manchas cloróticas difusas que, con el tiempo, se deforman, se enrollan y pueden llegar a encresparse. Además, las superficies foliares suelen estar cubiertas por un polvo blanco, que limita su capacidad fotosintética.(Alexandra Mora-Ramos et al., 2020)

En los frutos: Los frutos afectados muestran un recubrimiento de polvillo blanco, que interfiere en su desarrollo normal, causando deformaciones y reduciendo significativamente su calidad comercial(Alexandra Mora-Ramos et al., 2020).

En los tallos y botones florales: En las ramas jóvenes y botones florales, el hongo forma una capa blanquecina que inhibe su crecimiento. En casos severos, las flores pueden no abrirse, y los botones florales caen prematuramente, disminuyendo el rendimiento del cultivo(Alexandra Mora-Ramos et al., 2020).

2.2.4 Manejo del cultivo

El manejo del cultivo de la mora de Castilla (Rubus glaucus Benth.) requiere estrategias integrales que incluyan la selección del terreno, siembra adecuada, riego eficiente y un programa de fertilización.

- **Manejo Integrado de Plagas (MIP):** El MIP integra estrategias culturales, biológicas y químicas para el control de plagas en el cultivo de mora. Se priorizan prácticas preventivas como la selección de material vegetal resistente, la adecuada distancia de siembra, y el control biológico mediante el uso de depredadores naturales como mariquitas y arañas. El control químico se reserva como última opción debido a su impacto negativo sobre los insectos benéficos. (Restrepo, 2021)

- **Riego:** El riego es esencial para mantener una adecuada humedad en el suelo, especialmente en épocas de sequía. Los sistemas de riego por microaspersión o goteo son los más recomendados, ya que optimizan el uso del agua y favorecen la producción de frutos de mayor tamaño y calidad. El manejo del riego debe considerar factores como la capacidad de retención de agua del suelo, las condiciones climáticas y el estado fenológico de la planta. Es importante evitar el exceso de agua, ya que puede causar asfixia radical y desarrollo de enfermedades. (Franco et al., 2020)

- **Siembra:** El terreno ideal para la siembra debe tener suelos fértiles, bien drenados, y con una textura franco arenosa o franco arcilloso. Estas características permiten un equilibrio adecuado entre la retención de agua y el drenaje, evitando tanto el exceso de humedad como la sequedad extrema. Además, los suelos deben estar libres de piedras y malezas, con un pH entre 5.5 y 6.5 para facilitar la disponibilidad de nutrientes esenciales como el fósforo y el potasio. Es recomendable realizar un análisis de suelo

previo a la siembra para identificar deficiencias nutricionales y ajustar el suelo con enmiendas orgánicas o minerales. (Cevallos B., 2020)

La mora de Castilla se propaga principalmente a través de estacas leñosas o brotes obtenidos de plantas madre seleccionadas. Estas estacas deben tener entre 25 y 30 cm de longitud, con al menos tres nudos, y ser plantadas en hoyos de aproximadamente 40 × 40 × 40 cm. Antes de plantar, las estacas deben desinfectarse con productos fungicidas para prevenir infecciones por hongos. (Franco et al., 2020)

Las distancias de siembra dependen de las condiciones del terreno y el sistema de producción. En cultivos comerciales, se recomienda una separación de 2 a 3 metros entre plantas y 3 metros entre filas. Este espaciamiento permite una adecuada aireación del cultivo, reduce la incidencia de enfermedades y facilita las labores agrícolas como el control de malezas, la fertilización y la cosecha. (Franco et al., 2020)

- **Fertilización:** La fertilización orgánica es una práctica esencial para mejorar el rendimiento y la calidad de los frutos de mora de Castilla. Este enfoque utiliza materia orgánica como estiércol y compost, que aporta nutrientes esenciales como nitrógeno (N), fósforo (P), potasio (K), calcio (Ca), entre otros. Además, los biofertilizantes, que contienen microorganismos benéficos como micorrizas y Trichoderma, promueven una mejor absorción de nutrientes, mejoran la estructura del suelo y reducen la dependencia de fertilizantes químicos. Estas prácticas no solo incrementan la producción, sino que también contribuyen a la sostenibilidad ambiental. (Franco et al., 2020)

La fertilización es un aspecto esencial para garantizar el rendimiento y calidad del cultivo. Los análisis de suelo deben ser la base para establecer un plan adecuado de fertilización. Se recomiendan aplicaciones de abonos completos como el triple 15 (15-15-15)

y el uso de fertilizantes orgánicos, como compost y estiércol, que aporten nitrógeno, fósforo y potasio, además de otros micronutrientes importantes. (Alexandra Mora-Ramos et al., 2020)

El uso de biofertilizantes que contienen microorganismos beneficiosos, como micorrizas, favorece la disponibilidad de nutrientes, mejora la estructura del suelo y reduce la dependencia de fertilizantes químicos. Estas prácticas contribuyen a la sostenibilidad del cultivo. (Benavides, 2021)

2.3 Tecnología IoT en la Agricultura

El Internet de las Cosas (IoT, por sus siglas en inglés) se define como una red de dispositivos interconectados capaces de recopilar, analizar y compartir información en tiempo real a través de internet. En la agricultura, esta tecnología permite la optimización de procesos y recursos mediante el uso de sensores, sistemas de comunicación y plataformas digitales. Su aplicación mejora la productividad y sostenibilidad del sector, impactando positivamente la toma de decisiones basadas en datos. (Tovar Soto et al., 2019)

2.3.1 Aplicación de sistema IoT en la agricultura

La aplicación de IoT en la agricultura permite la integración de diversos dispositivos y tecnologías para el monitoreo y control de cultivos. Los sensores IoT recopilan datos clave sobre variables ambientales como la temperatura, humedad del suelo y calidad del aire, generando información fundamental para optimizar la gestión agrícola. Estos datos son transmitidos a través de redes inalámbricas y procesados mediante algoritmos avanzados en plataformas en la nube, facilitando el análisis en tiempo real y la automatización de sistemas como el riego inteligente y el control de plagas. (Tovar Soto et al., 2019)

El monitoreo de cultivos mediante IoT no solo permite detectar condiciones adversas de manera temprana, sino que también optimiza el uso de insumos agrícolas al aplicar estrategias de precisión. Tecnologías como redes de sensores inalámbricos (WSN), inteligencia

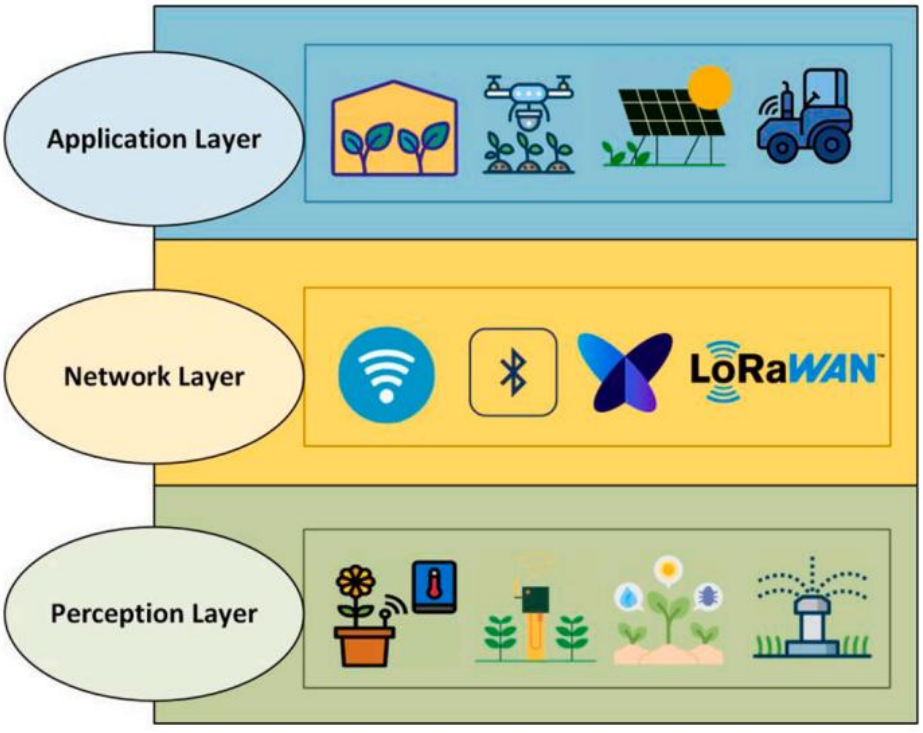
artificial y aprendizaje automático mejoran la detección de plagas y enfermedades, reduciendo la dependencia de métodos tradicionales y minimizando el impacto ambiental. (Rodas Vásquez, 2024)

Diversos estudios han demostrado que la integración de IoT con sensores avanzados y plataformas de análisis de datos permite un manejo eficiente del cultivo, garantizando una mayor producción con menores costos operativos. Además, el uso de tecnologías como LoRa y computación en la nube optimiza la transmisión y almacenamiento de datos, reduciendo la latencia y mejorando la eficiencia en la toma de decisiones. (Tovar Soto et al., 2019)

En la Figura 4 se muestra la arquitectura propuesta del sistema de monitoreo IoT para cultivos, en la cual se destacan las distintas capas tecnológicas que intervienen en la recolección, procesamiento y análisis de los datos.

Figura 4

Arquitectura IoT propuesta para el sistema de monitoreo del cultivo



Nota: Adaptado de: (Avşar & Mowla, 2022)

2.3.2 Visión por Computador en la Agricultura

La detección automatizada con visión por computadora tiene un impacto significativo en la agricultura moderna, transformando procesos tradicionales en métodos más eficientes y sostenibles. Esta tecnología permite identificar plagas, enfermedades y anomalías en los cultivos mediante el uso de imágenes capturadas y procesadas en tiempo real. El uso de visión por computadora contribuye a reducir la dependencia de inspecciones manuales, aumentando la precisión y disminuyendo el tiempo necesario para detectar problemas en las plantas. (Shahriar Zaman Abid et al., 2024)

Uno de los principales beneficios es la capacidad de actuar de forma preventiva, ya que los sistemas basados en visión por computadora pueden alertar tempranamente sobre problemas en el cultivo. Esto permite a los agricultores tomar decisiones informadas y minimizar pérdidas económicas. Además, al implementar esta tecnología, se reduce el uso de pesticidas y fertilizantes al focalizar los tratamientos únicamente en las áreas afectadas, lo que contribuye a la sostenibilidad ambiental y disminuye los costos operativos.(Shahriar Zaman Abid et al., 2024)

Por otro lado, la integración de visión por computadora con herramientas de IoT, como sensores y sistemas de almacenamiento en la nube, mejora la capacidad de monitoreo continuo y la gestión de datos agrícolas. Estos sistemas generan información valiosa que puede ser utilizada para predecir el rendimiento de los cultivos, optimizar los recursos disponibles y adaptar las prácticas agrícolas a las condiciones específicas del entorno. (Tovar Soto et al., 2019)

2.3.2.1 Técnicas y herramientas de visión por computador

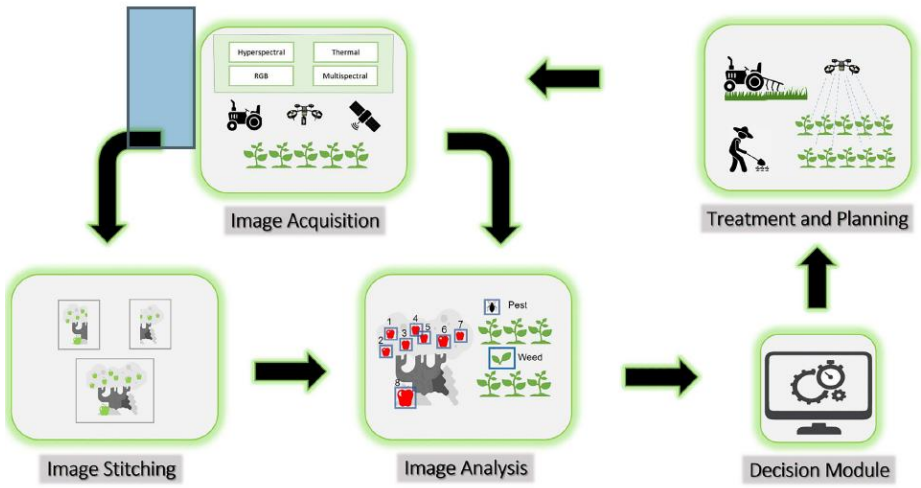
Los sistemas de visión inteligente combinan visión por computador, inteligencia artificial y aprendizaje automático, permitiendo que las máquinas reconozcan patrones y realicen predicciones basadas en la información visual capturada en campo.

El procesamiento de imágenes agrícolas generalmente implica varias etapas: adquisición, preprocesamiento, segmentación, extracción de características y clasificación, estas etapas permiten analizar rasgos como color, textura y forma de las plantas para identificar condiciones anómalas, como enfermedades o estrés hídrico. (Ghazal et al., 2024a)

Según (Ghazal et al., 2024a), el análisis de imágenes en agricultura se basa en operaciones como preprocesamiento, extracción de características, segmentación y reconocimiento de objetos, con el objetivo de apoyar tareas como la planificación de tratamientos y la estimación del rendimiento del cultivo. La figura 5 presenta el flujo general propuesto de trabajo en un sistema de visión por computador aplicado a la agricultura.

Figura 5

Flujo general propuesta de procesamiento de imágenes agrícolas



Nota: Adaptado de (Ghazal et al., 2024a)

Tradicionalmente, estos sistemas dependían de técnicas manuales de extracción de características como SIFT, HOG u ORB; sin embargo, estas presentan limitaciones frente a variaciones de iluminación, escala y oclusión. La revisión de (Wang & Kang, 2025), explica que la visión por computador ha evolucionado hacia arquitecturas basadas en aprendizaje profundo, donde redes neuronales convolucionales (CNN) como ResNet, SegNet y YOLO permiten un análisis más robusto y preciso, especialmente en tareas como la detección de enfermedades y clasificación de malezas.

Dentro de estas herramientas, OpenCV es una biblioteca de código abierto ampliamente utilizada para el procesamiento de imágenes en tiempo real. Proporciona funciones para filtrado, segmentación, detección de bordes y seguimiento de objetos, permitiendo la implementación de sistemas eficientes en dispositivos de bajo costo como Raspberry Pi.(Sinha et al., 2025)

Por otro lado, la familia de modelos YOLO (You Only Look Once) se ha consolidado como uno de los métodos más eficientes para la detección de objetos en tiempo real. Según la revisión sistemática de (Ragab et al., 2024), YOLO es un algoritmo de una sola etapa que utiliza una red neuronal convolucional para predecir simultáneamente las cajas delimitadoras y las probabilidades de clase, permitiendo procesar imágenes con gran velocidad y precisión. El estudio también destaca su evolución desde YOLOv1 hasta YOLOv8, incorporando mejoras en rapidez, precisión y capacidad de detectar objetos de distintos tamaños.

Las aplicaciones de YOLO en análisis de cultivos incluyen la detección de lesiones foliares, clasificación de malezas y monitoreo visual del desarrollo de la planta. Estas arquitecturas superan ampliamente a técnicas tradicionales y alcanzan niveles de precisión superiores al 97% en tareas de detección agrícola.(Ragab et al., 2024)

2.3.2.2 Espacios de color

El espacio de color es un método mediante el cual se puede especificar, crear o visualizar cualquier color. Dependiendo del tipo de sensor y la aplicación, se han desarrollado diversos espacios de color para la adquisición, transmisión y procesamiento de imágenes digitales. Los parámetros psíquicos de la percepción de(De & Hueso, s. f.)l color son la luminosidad, el tono y la saturación, los cuales constituyen la base de los distintos modelos de representación cromática (De & Hueso, 2020)

Uno de los espacios de color más utilizados es el RGB (Red, Green, Blue), que se basa en la combinación de tres señales de luminancia cromática: rojo, verde y azul. Sin embargo, este espacio presenta el inconveniente de mezclar la información del color con la intensidad, lo que dificulta las tareas de segmentación por color(De & Hueso, 2020)

Para superar estas limitaciones, se ha desarrollado el espacio HSV (Hue, Saturation, Value), el cual separa la información del color en tres componentes independientes: el tono que indica la tonalidad del color, la saturación que indica la pureza del color y el brillo que indica la intensidad luminosa. Esta separación facilita la segmentación de objetos por color, ya que permite analizar cada componente de forma independiente sin la interferencia de los cambios de iluminación(De & Hueso, 2020).En el contexto de visión por computadora aplicada a la agricultura, el espacio HSV ha demostrado ser especialmente útil para identificar características visuales de plantas y frutos, dado que permite diferenciar tonalidades de verde, amarillo y marrón características de los distintos estados de salud foliar (Bonilla-González & Prieto-Ortiz, 2016).

2.3.2.3 Etapas de visión artificial

La visión artificial es una disciplina que permite a los sistemas computacionales interpretar y analizar información visual del entorno físico. En su aplicación a la agricultura, el

procesamiento de imágenes generalmente implica una secuencia de etapas bien definidas que permiten transformar una imagen capturada en información útil para la toma de decisiones (Ghazal et al., 2024b).

La primera etapa es la **adquisición de imagen**, en la cual una cámara captura la información visual del objeto de estudio. En esta etapa es fundamental garantizar condiciones adecuadas de iluminación y enfoque para obtener imágenes representativas del objeto analizado. Diversos tipos de cámaras en el espectro visible, infrarrojo u otras longitudes de onda del espectro electromagnético son utilizadas ampliamente en función de la aplicación (Bonilla-González & Prieto-Ortiz, 2016).

La segunda etapa corresponde al **preprocesamiento**, en la cual se realizan ajustes sobre la imagen capturada para mejorar su calidad y facilitar el análisis posterior. Entre las operaciones más comunes se encuentran la eliminación de ruido, la corrección de iluminación, el redimensionamiento y la conversión entre espacios de color. Estas operaciones son fundamentales para garantizar que la imagen cumpla con los requisitos del modelo de detección (Ghazal et al., 2024b).

La tercera etapa es la **segmentación**, mediante la cual se separan las regiones de interés del resto de la imagen. En aplicaciones agrícolas, la segmentación permite aislar las zonas afectadas de las hojas o frutos del fondo de la escena, facilitando el análisis de sus características visuales. Una de las técnicas más utilizadas para la segmentación en agricultura es la segmentación por color en el espacio HSV, que permite identificar regiones según su tonalidad (Bonilla-González & Prieto-Ortiz, 2016)

La cuarta etapa corresponde a la **extracción de características**, en la cual se obtienen los atributos relevantes de las regiones segmentadas, como el color, la textura y la forma. Estas

características son expresadas de forma cuantitativa y constituyen la entrada para los algoritmos de clasificación (Ghazal et al., 2024b).

Finalmente, la quinta etapa es la **clasificación**, en la cual se asigna cada imagen o región a una categoría predefinida a partir de las características extraídas. Para esta tarea se han utilizado diversos algoritmos, desde clasificadores tradicionales como el bayesiano y el k-vecinos más cercanos, hasta arquitecturas de aprendizaje profundo como las redes neuronales convolucionales y la familia de modelos YOLO, que han demostrado niveles de precisión superiores al 97% en tareas de detección agrícola (Ghazal et al., 2024b).

2.3.3 Protocolos y Tecnologías de Comunicación en IoT Agrícola

Los sistemas de agricultura inteligente requieren transmitir datos en tiempo real desde los dispositivos de percepción (sensores, cámaras, actuadores) hacia plataformas de procesamiento y análisis. Para ello, los protocolos de comunicación inalámbrica constituyen un componente esencial, ya que permiten la interconexión de múltiples dispositivos distribuidos en terrenos agrícolas de gran extensión, asegurando un bajo consumo energético y una transmisión confiable de datos (Avşar & Mowla, 2022).

Entre las tecnologías más empleadas en aplicaciones agrícolas se encuentran:

- **ZigBee:** protocolo de bajo costo y bajo consumo energético, adecuado para redes de sensores inalámbricos en espacios reducidos o invernaderos (Avşar & Mowla, 2022).
- **Wi-Fi:** ofrece altas velocidades de transmisión y conectividad directa con internet, aunque su consumo energético es elevado, lo que limita su uso en áreas extensas sin acceso a energía estable (Avşar & Mowla, 2022).

- **Sigfox:** tecnología de banda ultraestrecha que permite comunicaciones de largo alcance con un consumo mínimo de energía, aunque su tasa de transmisión de datos es muy baja (Avşar & Mowla, 2022).

- **NB-IoT (Narrowband IoT):** basado en redes celulares, proporciona amplia cobertura y seguridad elevada, pero implica costos de suscripción y dependencia de la infraestructura de telecomunicaciones (Avşar & Mowla, 2022).

- **LoRaWAN (Long Range Wide Area Network):** protocolo de área amplia y bajo consumo energético, ampliamente adoptado en agricultura de precisión por su capacidad de cubrir grandes extensiones con un costo reducido y buena durabilidad de batería (Avşar & Mowla, 2022).

3 CAPITULO III DISEÑO E IMPLEMENTACIÓN

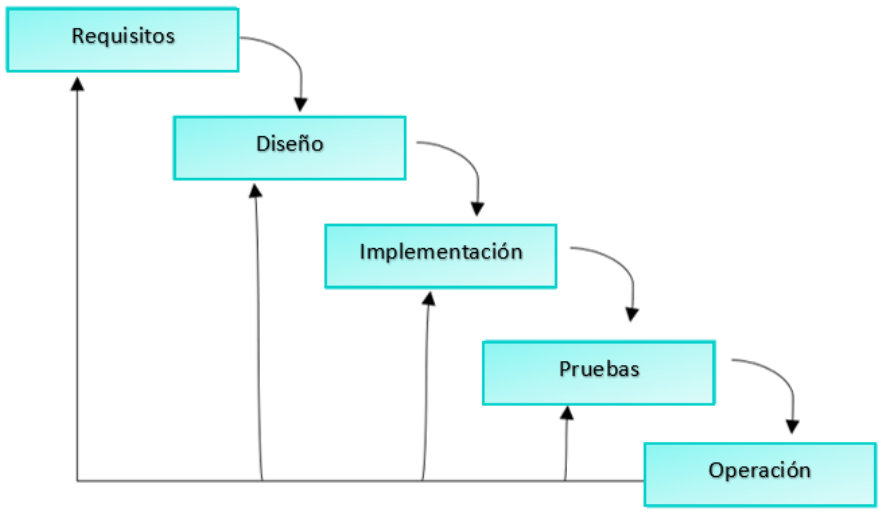
Este capítulo describe la metodología en cascada aplicada al desarrollo del proyecto, enfocándose en las fases 2 y 3. En él se detallan los requerimientos necesarios para seleccionar tanto el hardware como el software adecuado, y se incluye el diseño del sistema con sus diagramas correspondientes.

3.1 Metodología

Para el desarrollo del proyecto se utilizó la metodología en cascada, una estrategia estructurada que permite organizar el proceso en fases secuenciales, donde cada etapa depende del resultado de la anterior. Esta metodología fue seleccionada debido a su claridad y orden lógico, facilitando el cumplimiento de los objetivos planteados. En la Figura 6 se ilustran las fases que componen esta metodología, las cuales se detallan a continuación.

Figura 6

Fases de la metodología Cascada



Nota. Adaptado de: (Yusuf & Badrul, 2024)

La secuencia de las fases del modelo en cascada es la siguiente:

- **Requisitos:** En esta fase inicial se identifican y documentan exhaustivamente todas las necesidades y expectativas del cliente o usuario final. La precisión en esta etapa es

fundamental para el éxito del proyecto, pues establece las bases para las fases posteriores (Yusuf & Badrul, 2024).

- **Diseño:** Con base en los requisitos definidos, se elabora un diseño técnico que detalla la arquitectura del sistema, la organización de datos, la interfaz y los componentes a desarrollar. Este diseño sirve como guía para los programadores y desarrolladores (Yusuf & Badrul, 2024).

- **Implementación:** Se lleva a cabo la construcción del software mediante la programación de los módulos y componentes definidos en el diseño. Cada módulo es desarrollado individualmente para después ser integrado (Yusuf & Badrul, 2024).

- **Pruebas:** Una vez construido el sistema, se ejecutan pruebas para verificar que el software cumple con los requisitos especificados y para identificar posibles errores o fallas. Esta fase asegura la calidad y funcionalidad del producto (Yusuf & Badrul, 2024).

- **Operación**: El sistema probado se entrega e instala en el entorno operativo del usuario. Aquí se realiza la puesta en marcha y capacitación necesaria para su uso adecuado (Yusuf & Badrul, 2024).

3.2 Análisis

En esta etapa se identifican los requerimientos esenciales para implementar el sistema IoT, teniendo en cuenta factores como la altitud, temperatura, ubicación geográfica y las características del cultivo de mora de Castilla. Se analizarán las plagas que afectan principalmente a la planta, con especial énfasis en aquellas que son altamente contagiosas y pueden propagarse rápidamente debido a las condiciones climáticas. Además, se evaluarán las infraestructuras necesarias para los sensores IoT, que se encargarán de recopilar datos sobre el ambiente y las condiciones del cultivo. Esta información permitirá una intervención rápida para el control de plagas, utilizando cámaras y sensores para su detección. El personal encargado

podrá tomar decisiones en tiempo real basándose en los datos recolectados, lo que facilitará el control efectivo de las plagas mediante recomendaciones.

3.2.1 Situación actual

La comunidad de San Francisco de Guachinguero, situada en la parroquia de San José de Quinchinche, en el cantón de Otavalo, es una zona dedicada principalmente a la agricultura, especialmente al cultivo de la mora de Castilla. En esta zona según (Cevallos Suarez, 2023), menciona que tiene un clima favorable para el cultivo, con temperaturas que oscilan entre los 9°C y 15°C, lo que es ideal para el desarrollo de la mora. Las precipitaciones anuales en la zona varían entre 1100 y 1500 mm, lo que asegura una humedad suficiente para que las plantas crezcan adecuadamente, aunque también genera condiciones propensas para la proliferación de plagas y enfermedades que afectan la producción.

En la comunidad, un sector pequeño se dedica al cultivo de la mora de Castilla. Sin embargo, el cultivo de mora de Castilla no está exento de desafíos. Entre las plagas más comunes que afectan a las plantas se encuentran el barrenado del tallo, la mosca de la fruta y los áfidos. Estas plagas causan daños directos a las plantas, tales como la destrucción de los frutos y el debilitamiento de las ramas, lo que reduce tanto la cantidad como la calidad de la cosecha. Estos insectos no solo afectan la producción de frutos, sino que también hacen que las plantas sean más susceptibles a otros problemas de salud. Para efectos de este proyecto, el monitoreo y las pruebas del sistema se realizaron específicamente en plantas que se encontraban en etapa post-siembra, con una altura aproximada de entre 45 y 50 cm, debido a que en este estadio las hojas presentan mayor susceptibilidad a los primeros síntomas visibles del mildiu polvoso y las condiciones del follaje son óptimas para la captura y análisis de imágenes.

Además de las plagas, las enfermedades fúngicas como la Peronospora, Botrytis, oídium y antracnosis son amenazas importantes para el cultivo. Estas enfermedades afectan la salud de las plantas de mora de Castilla, causando el amarillamiento y la caída prematura de las hojas, lo que interfiere en el proceso de fotosíntesis y, por lo tanto, afecta el crecimiento y la productividad del cultivo. Estas enfermedades pueden propagarse rápidamente, lo que hace que el control temprano sea crucial para evitar daños graves.

Un factor que agrava la situación es el desconocimiento sobre el uso adecuado de fertilizantes, tanto químicos como orgánicos. Muchos agricultores no aplican los fertilizantes de manera correcta, lo que puede tener efectos negativos en las plantas. Esta falta de conocimiento, junto con la confusión en la aplicación de tratamientos, ha favorecido la proliferación de enfermedades fúngicas, especialmente la antracnosis, que es una de las más perjudiciales para el cultivo de mora de Castilla. La antracnosis, aunque similar al oídium, tiene características particulares que la hacen más difícil de controlar. Se distingue por los daños visibles en los frutos y las ramas, lo que deteriora la calidad de la cosecha y reduce su valor en el mercado.

En la Figura 7 se muestra el área del cultivo de la mora de Castilla, lo que permite visualizar la extensión del terreno y entender mejor las condiciones en las que se desarrollan estos cultivos.

Figura 7

Zona de cultivo de la mora de castilla



Nota: Autoría propia. Adaptado Google map

3.2.2 Recolección de información

Para la recopilación de los requisitos del sistema se utilizó como técnica principal la encuesta aplicada a los productores de mora de Castilla en la comunidad San Francisco de Guachinguero. El propósito fue obtener información real sobre las plagas que afectan con mayor frecuencia al cultivo, así como las prácticas de manejo agrícola empleadas por los agricultores. Este proceso permitió identificar y priorizar las principales problemáticas fitosanitarias, recopilando datos útiles para el diseño del sistema IoT orientado al monitoreo y control de enfermedades.

La información obtenida mediante esta técnica constituye una base fundamental para orientar el sistema hacia las necesidades reales del campo, ya que permitirá enfocar el monitoreo en las plagas de mayor incidencia, generar alertas preventivas y ofrecer recomendaciones sobre el uso adecuado de fertilizantes, contribuyendo a mejorar la salud y productividad del cultivo.

3.2.2.1 Encuesta

La encuesta fue aplicada a diez productores de la comunidad San Francisco de Guachinguero que se dedican al cultivo de la mora de Castilla. Su objetivo fue recopilar información detallada sobre las plagas más comunes, los síntomas que presentan las plantas afectadas y las prácticas de fertilización empleadas, con el fin de conocer las condiciones reales de producción y los métodos de control utilizados por los agricultores. La recolección de datos se realizó entre el 27 al 31 de noviembre de 2025, mediante visitas presenciales a los terrenos de cultivo, lo que permitió observar directamente el estado de las plantas y contrastar la información proporcionada por los participantes.

El cuestionario tuvo un formato de preguntas cerradas y semiabiertas, lo que facilitó obtener tanto información cuantitativa como cualitativa. Las preguntas se organizaron para abordar temas como la identificación de plagas predominantes, los signos visibles de daño en las plantas, la frecuencia de aparición, el tipo y frecuencia de uso de fertilizantes, así como los criterios empleados para su aplicación. En la Tabla 2 se presentan las preguntas formuladas, y en el Anexo A se incluye el cuestionario en su formato original.

Tabla 2

Preguntas de la Encuesta

Nro.	Preguntas
1	¿Qué tipo de plaga afecta más al cultivo de mora de Castilla en su propiedad?
2	¿Cuáles son los signos más comunes de estas plagas en las plantas de mora de Castilla?
3	¿Con qué frecuencia observa estas plagas en su cultivo?
4	¿Utiliza fertilizantes en su cultivo de mora de Castilla?
5	¿Qué tipo de fertilizantes utiliza?
6	¿Cuáles son los fertilizantes que utiliza con mayor frecuencia?
7	¿Con qué frecuencia aplica fertilizantes a su cultivo?

8	¿Sigue las recomendaciones de dosificación de los fertilizantes o ajusta según su experiencia?

3.2.2.2 Análisis de la encuesta

La encuesta realizada a los 10 productores encargadas de los cultivos de la mora de Castilla en la comunidad San Francisco de Guachinguero, parroquia San José de Quinchinche. De acuerdo con las preguntas 1, 3 y 7, se evidenció que alrededor del 90% la plaga que más afecta al cultivo es el Mildeo Polvoso, cuya presencia es muy frecuente y puede observarse de manera diaria o semanal. Esta plaga resulta sumamente perjudicial, pues deteriora la salud de las plantas y, en caso de no ser controlada a tiempo, compromete directamente la producción total. En cuanto a la fertilización, indicó que suele aplicar fertilizantes cada mes, aunque en ciertas ocasiones lo hace únicamente cuando lo considera necesario, de acuerdo con el estado de las plantas y las condiciones del terreno.

Respecto a las preguntas 2, 4, 5 y 6, se pudo identificar que los síntomas más comunes de las plagas en su cultivo son la destrucción de los frutos y la deformación de las hojas, daños que reducen la calidad de la mora y dificultan su comercialización, además de limitar el desarrollo normal de la planta. Los productores manifestaron también que sí utilizan fertilizantes en sus cultivos, dando prioridad a los fertilizantes químicos, aunque en ocasiones complementa con abonos orgánicos. Entre los productos más empleados mencionó el Metalaxyl, Score 250 EC, Azoxystrobina y Ciproconazol, utilizados principalmente para el control de enfermedades fúngicas, que son las de mayor incidencia en la mora de Castilla.

En lo relacionado con la pregunta 8, menciono que en algunos casos se siguen las recomendaciones de dosificación indicadas en las etiquetas de los productos, mientras que en otras ocasiones se ajustan las cantidades según la experiencia en el manejo del cultivo, lo que refleja una combinación entre conocimiento técnico y práctico.

A partir del análisis realizado se determina que la plaga de mayor incidencia es el mildiu polvoso, por lo que resulta factible implementar un sistema IoT con soporte de visión por computadora e inteligencia artificial, capaz de identificar esta enfermedad en el cultivo y generar alertas que apoyen su manejo oportuno.

3.3 Requerimientos del sistema

A continuación, basándonos en los objetivos específicos 2 y 3, y siguiendo la metodología en cascada, se detallan los requerimientos necesarios para el diseño del sistema IoT. También se incluirán los requerimientos de los stakeholders, la arquitectura del sistema, la selección de hardware, diagramas correspondientes y la arquitectura de referencia mencionada en el capítulo I. En este contexto, se abordará la arquitectura del sistema IoT, que contempla las capas, la selección de sensores, gateways y tecnologías inalámbricas para el envío de datos. Además, se considerará el software necesario para la visualización y gestión del sistema. Este sistema de control de plagas, centrado en el mildiu polvoso, permitirá detectar para dar control al manejo del cultivo de la mora de Castilla.

3.3.1 Determinación de stakeholders

Para la determinación de los stakeholders, son aquellos grupos que forman en el resultado del proyecto de acuerdo con la norma ISO/IEC/IEEE 29148-2018, los stakeholders o partes interesadas engloban a aquellos individuos o grupos que tienen un interés directo con el resultado final. Esto incluye tanto a personas como a entidades que se verán afectadas o beneficiadas, de manera directa o indirecta. Esto incluye a los usuarios finales, patrocinadores, el equipo de desarrollo, posibles clientes, entre otros. Es esencial involucrar a estas partes interesadas en todas las fases del proyecto, desde la recopilación de requisitos hasta la evaluación final.

En la tabla 3 se detalla la participación de quienes contribuirán al desarrollo de este proyecto.

Tabla 3

Lista de participantes en el proyecto

Nro.	Lista de Stakeholders	
1	MSc. Jaime Michilena	Tutor de Tesis
2	MSc. Luis Suárez	Asesor de Tesis
3	Srta. Diana Panamá	Tesista
4	Productores de la Mora de Castilla.	Usuarios

3.3.2 Requrimientos del software y hardware

Los criterios considerados en la definición de los requisitos del sistema se establecen con base en las necesidades identificadas por las partes interesadas y en las recomendaciones de expertos en agricultura y tecnologías de visión por computador. Estos requisitos se orientan a garantizar la detección oportuna de enfermedades en el cultivo de mora de Castilla mediante el análisis automatizado de imágenes.

El requisito del software debe disponer de capacidades de procesamiento visual, clasificación de imágenes y emisión de alertas automáticas, mientras que el hardware debe brindar soporte adecuado para la ejecución continua del sistema. Esta fase resulta esencial dentro del desarrollo del sistema, ya que define la infraestructura tecnológica necesaria para garantizar el correcto funcionamiento del modelo de análisis visual. De este modo, se asegura que el sistema pueda detectar y notificar oportunamente la presencia de enfermedades, recomendando acciones correctivas al agricultor, como la aplicación de fertilizantes o fungicidas para el cultivo.

3.3.3 Nomenclatura para requisitos

En la Tabla 4 se presentan las abreviaturas utilizadas para identificar los diferentes tipos de requisitos del sistema. Esta nomenclatura facilita la comprensión y referencia de los elementos que intervienen en el desarrollo del proyecto.

Tabla 4

Nomenclatura para condiciones

Acrónimo	Condiciones
StSR	Requerimientos de Stakeholders
SySR	Requerimientos del Sistema
SrSH	Requerimientos de Hardware y Software

3.3.4 Requerimientos de Stakeholders

En cuanto a los requisitos que deben cumplirse, es importante destacar que estos fueron definidos a partir de reuniones y consultas a los productores de mora de Castilla en la comunidad San Francisco de Guachinguero, parroquia San José de Quinchinche, cantón Otavalo.

Durante estos encuentros se identificaron las principales problemáticas que afectan al cultivo, entre ellas la presencia recurrente del mildiu que deteriora la calidad y el rendimiento de la cosecha. La agricultora señaló la necesidad de contar con una herramienta tecnológica que permita identificar los síntomas visibles en las hojas y frutos, y que además emita una notificación oportuna con recomendaciones para el tratamiento adecuado.

Estas exigencias reflejan las expectativas de los usuarios finales respecto al sistema, los stakeholders manifestaron también la importancia de que el sistema proporcione información clara, confiable y de fácil acceso desde dispositivos móviles, de modo que pueda integrarse de manera práctica en las labores del cultivo. En la Tabla 5 se detallan los principales requerimientos establecidos por las partes interesadas.

Tabla 5

Requerimientos de Stakeholders

Requerimientos de stakeholders (StSR)		**Prioridad**		
	Requerimientos operacionales	**Baja**	**Media**	**Alta**
StSR1	Acceso a una fuente de energía que proporcione un suministro estable y continuo, adecuado para el funcionamiento del sistema.			X
StSR2	Asegurar la transmisión y recepción de datos e imágenes.		X	
StSR3	Capturar imágenes del cultivo y procesarlas mediante visión por computador para detectar síntomas de enfermedad.			X
StSR4	Generar notificaciones automáticas cuando se identifique una planta afectada.			X
StSR5	Registrar y almacenar los resultados de cada detección para su análisis posterior.			X
	Requerimientos de usuario			
StSR6	Interfaz de usuario sencilla, intuitiva y de fácil comprensión.			X
StSR7	Implementación de hardware y software de bajo costo.			X
StSR8	Alerta con recomendaciones sobre la aplicación de fertilizantes			X

o fungicidas.

3.3.5 Requerimientos de arquitectura

Las exigencias de arquitectura del sistema se definen con el propósito de garantizar un funcionamiento eficiente, preciso y continuo en la detección de enfermedades del cultivo de mora de Castilla. La arquitectura propuesta se fundamenta en la integración de componentes tecnológicos que permiten capturar, procesar y analizar imágenes mediante técnicas de visión por computador, generando alertas para el agricultor. En la Tabla 6 se detallan las condiciones de la arquitectura definidas a partir de las observaciones del usuario final y de los requerimientos técnicos necesarios para garantizar el correcto desempeño del sistema.

Tabla 6

Requerimientos de Arquitectura

Requerimientos de Arquitectura (SrSH)		**Prioridad**		
		Baja	**Media**	**Alta**
Requerimientos de hardware				
SrSH1	Capacidad de procesamiento de imágenes capturadas por la cámara para asegurar la detección.			X
SrSH2	Disponibilidad de una fuente de energía estable para evitar interrupciones durante el funcionamiento.			X

SrSH3	Dimensiones compactas y diseño ligero que faciliten la instalación y el mantenimiento en campo.	X
SrSH4	Cámara para exteriores resistente al entorno agrícola, con resolución mínima de 5 Mpx.	X
SrSH5	Conectividad inalámbrica o local que permita enviar las alertas y resultados de detección al usuario final.	X
	Requerimientos de Software	
SrSH6	Capacidad de almacenamiento de datos en tiempo real.	X
SrSH7	Manejo de lenguaje de programación Phyton.	X
SrSH8	Interfaz intuitiva para visualizar alertas y resultados del análisis de las imágenes.	X
SrSH9	Integración de algoritmos de procesamiento de imágenes mediante la librería OpenCV para	X

	detectar síntomas visibles de enfermedades en las plantas.	
SrSH10	Generación de notificaciones automáticas al detectar afectaciones en el cultivo, incluyendo una breve recomendación al usuario.	X
SrSH11	Programa estable, de fácil mantenimiento y compatible con diferentes sistemas operativos o plataformas de desarrollo.	X
	Requerimientos de diseño	
SrSH12	Compatibilidad entre hardware y software para rendimiento eficiente.	X
SrSH13	Optimización de procesamiento de imágenes para funcionamiento continuo.	X
SrSH14	Capacidad del sistema para conectarse a redes Wi-Fi y transmitir datos y alertas de manera confiable.	X
	Requerimientos lógicos	

SrSH15	Implementación accesible con costo reducido del sistema.	X
SrSH16	Seleccionar la tecnología inalámbrica adecuada de acuerdo con la infraestructura del lugar.	X
SrSH17	Uso de protocolos HTTP para enviar notificaciones y resultados.	X
Requerimientos eléctricos		
SrSH18	Acceso a una toma de corriente que suministre energía eléctrica estable para la alimentación del sistema.	X

3.3.6 Requerimientos del sistema

A continuación, se presentan las exigencias del sistema en la Tabla 7, definidas para garantizar un funcionamiento confiable, eficiente y fácil de usar, cumpliendo con los requerimientos del usuario final.

Tabla 7

Requerimientos del Sistema

Requerimientos del Sistema (SySR)		**Prioridad**		
		Baja	**Media**	**Alta**
Requerimientos de interfaces				
SySR1	Alimentación eléctrica estable mediante			X

	puerto USB o conexión de 5V para los dispositivos.	
SySR2	Plataforma intuitiva para la visualización de imágenes y monitoreo de datos.	X
SySR3	Puertos suficientes para la conexión de la cámara.	X
	Requerimientos de uso	
SySR4	Funcionamiento continuo de los dispositivos durante la captura y análisis de imágenes.	X
SySR5	Transmisión confiable de alertas y resultados al usuario mediante red local o Wi-Fi	X
	Condiciones para manejo	
SySR6	Procesamiento preciso de imágenes para identificar signos visibles de enfermedades.	X
SySR7	Instalación y configuración sencilla y confiable del sistema en el área de cultivo.	X

SySR8	Pruebas previas del software para garantizar detección correcta antes de su implementación.	X
SySR9	Protector adaptable que asegure la integridad de los dispositivos en campo.	X
Condiciones Físicas		
SySR10	Seguridad física de los dispositivos durante su operación en el área de cultivo.	X
SySR11	Disponibilidad del área de cultivo para la correcta instalación del sistema.	X
SySR12	Línea de visión despejada entre la cámara y las plantas para capturar imágenes de calidad	X

3.4 Diseño y descripción del sistema

En esta sección se presenta el diseño e implementación del sistema IoT orientado al monitoreo y control de la plaga que afecta con mayor frecuencia al cultivo de la mora de Castilla, específicamente el mildiu polvoso, con el objetivo de generar alertas que apoyen el manejo oportuno del cultivo. El propósito principal del sistema es mejorar la calidad y salud

de las plantas, facilitando la detección de la enfermedad mediante la integración de tecnologías IoT y visión por computadora.

Tomando como referencia el análisis desarrollado en los capítulos II y III, donde se identificaron los requerimientos técnicos y funcionales necesarios para su correcto funcionamiento, a continuación, se detalla la fase de diseño, junto con la selección de los componentes de hardware, software y la arquitectura general del sistema IoT propuesto.

3.4.1 Selección de Hardware y software para el sistema

La selección del hardware y software se realizó considerando los requerimientos técnicos y operativos definidos a partir de las necesidades identificadas por los stakeholders.

Con este propósito, se efectuó un análisis diagnóstico que permitió evaluar las características, costos y compatibilidad de los componentes necesarios para el desarrollo del sistema. El objetivo fue garantizar que los elementos seleccionados cumplan con las condiciones requeridas para asegurar la óptima funcionalidad, rendimiento y estabilidad del dispositivo IoT.

Para elegir los equipos se elaboraron tablas comparativas donde se relacionaron los requisitos del sistema con valores de puntuación de 1 y 0 (1: cumple, 0: no cumple), tanto para el hardware como para el software.

3.4.1.1 Selección de hardware

La selección del hardware se realizó en función de los requisitos técnicos definidos para el sistema IoT, considerando los componentes que permitan un procesamiento eficiente de imágenes, una transmisión estable de datos y una operación continua en campo.

3.4.1.1.1 ***Selección de microcontrolador***

Para la selección del microcontrolador, se consideraron los requisitos técnicos del sistema IoT y las necesidades de comunicación y control del proyecto. Un microcontrolador

permite automatizar tareas, procesar datos y gestionar dispositivos, siendo ideal por su versatilidad y tamaño compacto. Se evaluaron alternativas como Arduino UNO, ESP32, Raspberry Pi 4 y Raspberry Pi 5, analizando su capacidad de procesamiento, compatibilidad con sensores y facilidad de programación. La elección se realizó conforme a los requisitos establecidos en la sección 3.3.5. En la Tabla 8 presenta el análisis detallado de los diferentes microcontroladores para elegir la opción más adecuada.

Tabla 8

Requerimientos para elección de microcontrolador

Hardware	Requerimientos									
	SrSH 1	**SrH 2**	**SrSH 4**	**SrSH 5**	**SrSH 7**	**SrSH 9**	**SrSH 11**	**SrS H14**	**SrS H15**	**Valoración Total**
Arduino UNO	0	1	1	0	0	1	1	0	1	5
ESP32	0	1	1	0	0	1	1	0	1	5
Raspberry Pi 4	1	1	1	1	1	1	1	1	1	9
Raspberry Pi 5	1	1	1	1	0	1	1	0	1	7
1 - Cumple										
0 - No cumple										

Elección: Raspberry Pi 4

Después de un estudio de características individuales de los microcontroladores, se ha seleccionado la Raspberry Pi 4 (ver Figura 8) como la opción más adecuada para el desarrollo del sistema. Este modelo integra un procesador Broadcom BCM2711 Quad-Core Cortex-A72 (ARM v8) con una frecuencia de 1.5 GHz, 8 GB de memoria RAM y conectividad Wi-Fi 802.11 b/g/n/ac y Bluetooth 5.0, lo que facilita la comunicación inalámbrica directa con el servidor Flask mediante el protocolo HTTP.

Uno de los factores determinantes en su elección es su compatibilidad con la cámara OV5647, la cual se conecta a través del puerto CSI (Camera Serial Interface) sin requerir controladores adicionales, garantizando así una integración sencilla y una captura de imágenes estable. Además, la placa opera con una alimentación de 5 V/3 A, suficiente para mantener un funcionamiento continuo y confiable en condiciones de campo.

Otra ventaja relevante para su elección fue su capacidad para ejecutar modelos de aprendizaje profundo y utilizar librerías de visión por computadora como OpenCV, TensorFlow Lite y Keras, lo que permite realizar el procesamiento y análisis de imágenes directamente en el dispositivo.

Figura 8

Microcontrolador Raspberry Pi 4



Fuente: Tomado de https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

3.4.1.1.2 *Selección de dispositivo de adquisición de imágenes*

Para la selección del sensor de imagen, se consideraron los requisitos técnicos del sistema IoT, así como la calidad, resolución y estabilidad en la captura de imágenes necesarias para el monitoreo del cultivo. Un sensor de imagen permite adquirir información visual de alta precisión, lo cual resulta fundamental para el procesamiento mediante visión por computadora. Se evaluaron diferentes alternativas, entre ellas la cámara OV5647, la cámara IMX219 y la

cámara IMX477, analizando su resolución. La elección se realizó conforme a los requisitos establecidos en el diseño del sistema IoT, priorizando la estabilidad, la calidad de imagen y la compatibilidad con las librerías de procesamiento.

En la Tabla 9 se presenta el análisis comparativo de las cámaras, para seleccionar la opción más adecuada para la implementación del sistema.

Tabla 9

Requerimientos para selección de adquisición de imágenes

Hardware	Requerimientos									
	StS R3	**StS R5**	**SrS H4**	**SrS H3**	**StS R7**	**SrSH 5**	**SyS R6**	**SyS R4**	**SyS R12**	**Valoración Total**
Cámara (OV5647)	1	1	1	1	1	1	1	1	1	9
Cámara (IMX219)	1	1	1	1	0	1	1	0	1	7
Cámara (IMX477)	1	1	1	1	0	1	1	1	1	8
1 - Cumple										
0 - No cumple										
Elección: Cámara OV5647										

Para la adquisición de imágenes del cultivo, se seleccionó la cámara OV5647 de 5 MP (ver figura 9), debido a su compatibilidad, estabilidad y facilidad de integración con la Raspberry Pi 4. La cámara OV5647 es uno de los módulos oficialmente soportados por el sistema de cámaras de Raspberry Pi, ya que utiliza una conexión a través del puerto CSI (Camera Serial Interface), permitiendo una transmisión rápida y estable de datos sin necesidad de controladores adicionales ni configuraciones complejas.

Esta cámara incorpora un sensor CMOS de 5 megapíxeles, capaz de capturar imágenes con una resolución de 2592×1944, adecuada para aplicaciones de análisis visual donde se requiere identificar detalles como manchas o síntomas tempranos de enfermedades en las hojas del cultivo.

Figura 9

Sensor de imagen Cámara OV5647



Fuente: Tomado de https://www.inno-maker.com/product/cam-ov5647/

3.4.1.2 Selección de software

Una vez seleccionado el hardware, para la selección del software del sistema IoT, se consideraron los requisitos técnicos definidos en la sección 3.3.5. Dicho software debe ser compatible con el microcontrolador elegido (Raspberry Pi 4) y permitir la ejecución de las funciones esenciales del sistema, como la captura y análisis de imágenes, el procesamiento local y la transmisión de resultados hacia la plataforma web. Para garantizar la flexibilidad, portabilidad y facilidad de integración, se optó por utilizar entornos de desarrollo y lenguajes de programación de código abierto y compatibilidad con los componentes del Gateway.

3.4.1.2.3 ***Software para el microcontrolador***

Para la selección del software del microcontrolador, se consideraron los requisitos técnicos definidos como la compatibilidad con el hardware y las necesidades de comunicación del sistema IoT. El software debía permitir la captura de imágenes, el control de la cámara

OV5647, el análisis inicial de datos y la transmisión de resultados a través de la red local. Se evaluaron alternativas de programación como C, Arduino IDE y Python, analizando su facilidad de integración, compatibilidad con librerías de visión por computador y capacidad para gestionar comunicaciones mediante protocolos como HTTP.

En la Tabla 10 se presenta el análisis comparativo del software para el microcontrolador.

Tabla 10

Requerimientos para el software del microcontrolador

	Requerimientos								
Lenguaje de programación	**StS R3**	**SrS H9**	**SrSH 13**	**SrS H17**	**SrSH 14**	**SyS R4**	**SySR 5**	**SySR 8**	**Valoración Total**
C	1	0	1	1	0	1	0	0	4
Arduino IDE	1	0	1	1	0	1	0	1	5
Python	1	1	1	1	1	1	1	1	8
1 - Cumple									
0 - No cumple									

Elección: Python

Para la programación del microcontrolador, se seleccionó Python debido a su compatibilidad nativa con la Raspberry Pi 4 y su integración directa con librerías necesarias para el proyecto, como OpenCV para el procesamiento de imágenes y los módulos de comunicación basados en HTTP utilizados por el servidor Flask. Python permite realizar la captura de imágenes, gestionar la comunicación Wi-Fi, registrar datos y ejecutar operaciones de forma continua, cumpliendo con los requisitos de estabilidad y bajo costo definidos en el diseño del sistema.

3.4.1.2.4 *Software de procesamiento de imágenes*

Para la selección del software encargado del procesamiento de imágenes, se consideraron los requerimientos definidos, especialmente aquellos relacionados con la capacidad del sistema para analizar imágenes en tiempo real y detectar síntomas del mildiu polvoso. El software debe ser compatible con el lenguaje Python, la cámara OV5647 y el microcontrolador Raspberry Pi 4, para garantizar un procesamiento eficiente y estable dentro del entorno IoT propuesto.

Entre las alternativas se evaluaron plataformas como MATLAB Toolbox, TensorFlow y Python (OpenCV), valorando aspectos como facilidad de uso, requerimientos computacionales, compatibilidad con Raspberry Pi y capacidad para realizar operaciones de visión por computador.

En la Tabla 11 se presenta la comparación realizada con base en las condiciones técnicas necesarias para el sistema.

Tabla 11

Requerimientos de Software Procesamiento de imágenes

	Requerimientos							
Software	**StSR 3**	**StSR 5**	**SrSH 1**	**SrSH 9**	**SrSH 13**	**SrSH 11**	**SySR 6**	**Valoración Total**
MATLAB Toolbox	1	1	1	1	0	1	1	6
TensorFlow	1	1	1	1	1	1	0	6
Python (OpenCV)	1	1	1	1	1	1	1	7
1 - Cumple								
0 - No cumple								

Elección: Python (OpenCV)

Se seleccionó Python (OpenCV) (Ver figura 11), debido a su amplio conjunto de herramientas para procesamiento de imágenes en tiempo real, su compatibilidad directa con Python y su rendimiento eficiente en dispositivos de bajo costo como la Raspberry Pi 4. Python (OpenCV) permite realizar operaciones como filtrado, segmentación, detección de bordes y análisis de textura, esenciales para identificar síntomas en las hojas del cultivo. Además, al ser de código abierto, contar con una amplia documentación y ofrecer una integración sencilla con modelos de detección como YOLOv, se convierte en la opción más adecuada para el sistema IoT propuesto.

3.4.1.2.5 *Software de visualización de datos y base de datos.*

Para la visualización de datos resulta fundamental contar con una herramienta que permita transmitir los resultados de detección hacia una interfaz web accesible para el usuario final. El software debe recibir los datos generados por la Raspberry Pi 4, mostrar alertas en tiempo real y operar mediante protocolos estándar en IoT, como HTTP.

Entre las alternativas evaluadas se consideraron los frameworks Django, Grafana y Firebase, analizando su complejidad, consumo de recursos, facilidad de integración con Python y capacidad para generar interfaces web.

	Requerimientos									
Software	**SrS H14**	**StS R6**	**StSR 4**	**StS R8**	**StS R9**	**SrS H8**	**SrS H11**	**SySR 2**	**SyS R5**	**Valoración Total**
Django	1	1	0	1	1	1	1	1	1	8
Grafana	0	1	0	1	0	1	1	0	1	5
Firebase	1	1	1	1	1	1	1	1	1	9

1 - Cumple

0 - No cumple

Elección: Firebase

Tomando en cuenta las necesidades especificadas por el usuario, es imperativo que el software de visualización ofrezca una interfaz fácil de usar y que su implementación sea costeable. Utilizando Firebase su integración directa con Python y su capacidad para implementar interfaces web simples pero eficientes, permite recibir los resultados generados por el sistema de visión por computador y mostrarlos al usuario mediante el protocolo HTTP, enviando alertas en tiempo real cuando se detectan síntomas de mildiu polvoso.

3.4.1.2.6 *Tecnología de transmisión inalámbrica*

Para la selección de la tecnología de transmisión inalámbrica se consideraron los requerimientos relacionados con la transmisión confiable de datos e imágenes, la compatibilidad con redes Wi-Fi y la adecuación a la infraestructura disponible. Entre las alternativas evaluadas, Wi-Fi (802.11 b/g/n), Zigbee y LoRaWAN, se analizaron aspectos como consumo de recursos, complejidad de implementación, facilidad de integración con Python y capacidad de interoperar con la plataforma web. En la Tabla 12 se presenta la comparación de estas tecnologías.

Tabla 12

Selección de tecnología inalámbrica

	Requerimientos								
Plataforma IoT	**StS R2**	**StS R7**	**SrSH 5**	**SrS H14**	**SrSH 16**	**SrS H15**	**SySR 4**	**SySR 5**	**Valoración Total**
Wi-Fi (802.11 b/g/n)	1	1	1	1	1	1	1	1	8
Zigbee	0	1	1	0	1	1	1	1	6
LoRaWAN	0	0	1	0	1	0	1	1	4

1 - Cumple 0 - No cumple
Elección: Wi-Fi (802.11 b/g/n)

Con base en estos criterios, se seleccionó Wi-Fi (802.11 b/g/n), ya que proporciona el ancho de banda necesario para el envío de imágenes, es compatible con la Raspberry Pi 4 y el servidor Flask, y permite una comunicación estable y continua con la plataforma de visualización de resultados.

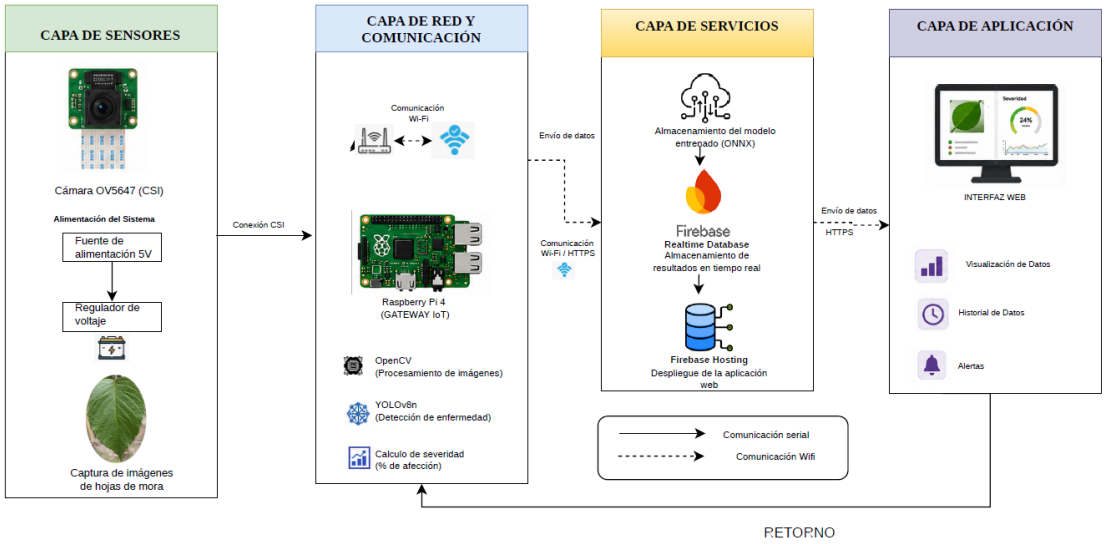
3.4.2 Arquitectura del sistema IoT

En base al análisis de requerimientos y la selección de Hardware y Software, se procede a realizar el diseño del sistema. Para entender de mejor manera la arquitectura y el funcionamiento del sistema se presentan diagramas de bloque y flujo.

En la figura 10 se presenta el diagrama general del sistema, organizado en cuatro capas principales: Capa de Sensores, Capa de Red y Comunicación, Capa de Servicios y Capa de Aplicación.

Figura 10

Arquitectura del sistema



Nota. Autoría propia.

- **Capa de Sensores**

En esta capa se realiza la captura de la información primaria del sistema. La cámara OV5647 obtiene imágenes de las hojas de mora y se conecta a la Raspberry Pi 4 mediante la interfaz CSI, lo que permite la transmisión directa de los datos. La alimentación del sistema se garantiza mediante una fuente de 5V, un regulador de voltaje y una batería de respaldo, asegurando el funcionamiento estable del hardware en campo. Esta capa corresponde a la obtención de datos desde el entorno físico y constituye el punto inicial del flujo del sistema.

- **Capa de Red y Comunicación**

La Raspberry Pi 4 actúa como gateway IoT del sistema. Sobre ella se ejecutan los procesos de OpenCV para el preprocesamiento de imágenes, el modelo YOLOv8n para la detección del mildiu polvoso, y el cálculo de severidad (porcentaje de afección) mediante segmentación de color. Una vez generado el resultado de la inferencia, la Raspberry Pi transmite la información hacia la capa de servicios mediante comunicación Wi-Fi. En esta capa

se concentra tanto el procesamiento local del gateway como la gestión de la comunicación de datos hacia el resto del sistema.

- **Capa de Servicios**

En esta capa se integran el almacenamiento del modelo y la base de datos en la nube. El modelo entrenado se almacena en formato ONNX y puede actualizarse de forma remota sin necesidad de reprogramar físicamente la Raspberry Pi. Los datos enviados desde el gateway son recibidos y almacenados en Firebase Realtime Database, donde se registran las detecciones junto con sus metadatos (severidad, fecha, imagen asociada). Adicionalmente, Firebase Hosting se encarga del despliegue de la aplicación web. Esta capa permite conservar el historial de resultados y sostener la disponibilidad del sistema mediante servicios en la nube.

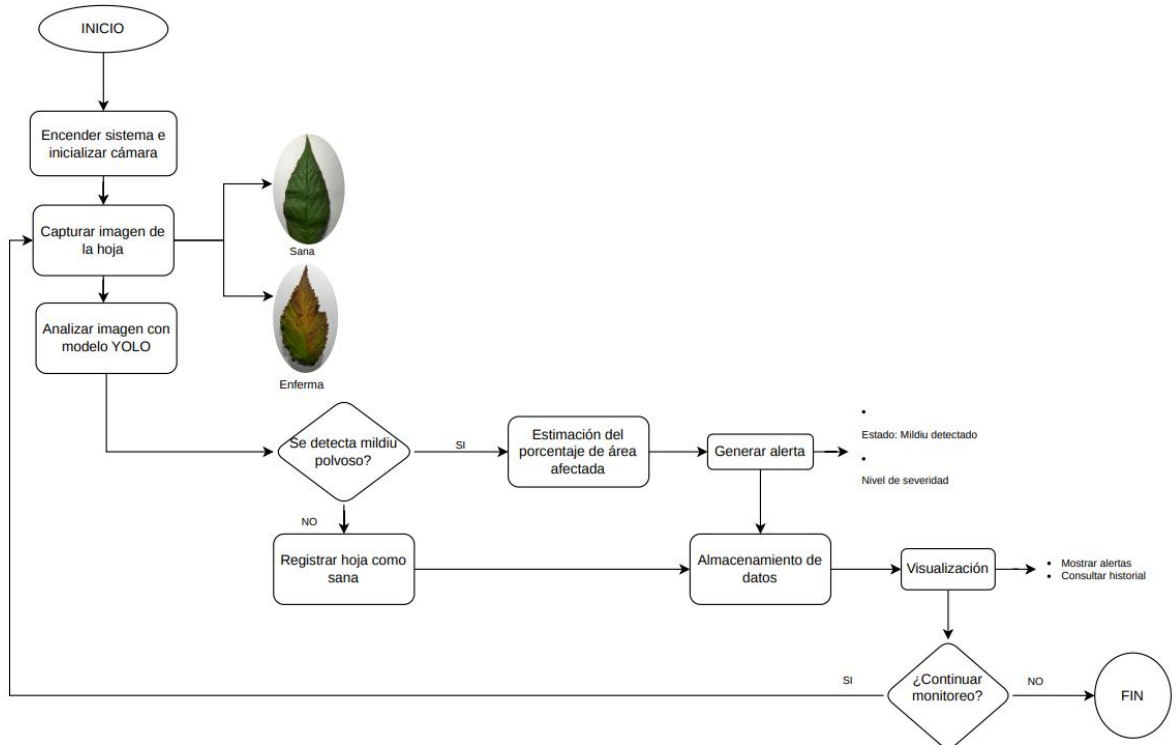
- **Capa de Aplicación**

La última capa corresponde a la interacción con el usuario final a través de la interfaz web. El sistema consulta la información almacenada en Firebase y la presenta de forma estructurada mediante visualización de datos, historial de registros y alertas de severidad. La flecha de retorno representada en el diagrama indica que el sistema funciona de manera cíclica, ya que la información visualizada puede generar acciones correctivas sobre el cultivo, las cuales serán nuevamente evaluadas mediante nuevas capturas y análisis.

1.1.1 Diagrama de flujo del funcionamiento del sistema.

En la Figura 10 se muestra el flujo general y específico de funcionamiento del sistema IoT, desde la captura de imágenes en el cultivo hasta la visualización de los resultados en la aplicación. El diagrama resume las etapas principales del proceso, permitiendo comprender de forma clara cómo los datos capturados son analizados y transformados en información útil para el usuario.

Figura 11
Diagrama de flujo del funcionamiento del sistema



Nota: Autoría propia

3.5 Desarrollo del sistema

Posteriormente, y en concordancia con la metodología planteada en la fase de diseño e integración de la arquitectura IoT, se procede al desarrollo e implementación del sistema propuesto. En esta etapa se integran los componentes físicos y lógicos que conforman el prototipo funcional para la detección de mildiu polvoso en hojas de mora de Castilla.

El sistema se compone de dos módulos principales: el módulo de adquisición de imágenes y el sistema de procesamiento y comunicación. El módulo de adquisición está conformado por la cámara OV5647, encargada de capturar imágenes del cultivo, mientras que el sistema de procesamiento está basado en una Raspberry Pi 4, responsable de ejecutar el modelo de detección, gestionar la información obtenida y transmitir los resultados hacia la base de datos en la nube mediante conexión Wi-Fi.

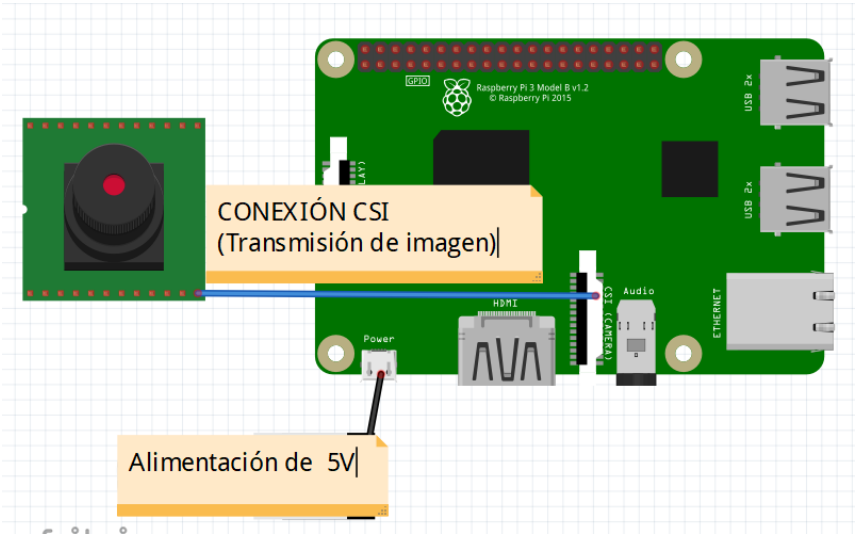
En esta sección se describe detalladamente la infraestructura física y lógica involucrada en la construcción del prototipo, abordando la conexión de los componentes, la configuración del entorno de software y la integración del modelo de detección, con el fin de garantizar el correcto funcionamiento del sistema IoT implementado.

3.5.1 Integración del sistema

Para la integración del sistema se inició con la fase de adquisición de datos, estableciendo la conexión física entre la cámara OV5647 y la Raspberry Pi 4 mediante la interfaz CSI (Camera Serial Interface). Esta interfaz permite la transmisión directa y de alta velocidad de los datos de imagen sin requerir módulos adicionales de conversión.

La Raspberry Pi es alimentada mediante una fuente regulada de 5V, asegurando estabilidad en el procesamiento y captura continua de imágenes. Esta configuración constituye la base del sistema, permitiendo la obtención de muestras visuales del entorno físico para su posterior procesamiento digital.

Figura 12
Diagrama de Sistema de adquisición de imágenes



Nota: Autoría propia

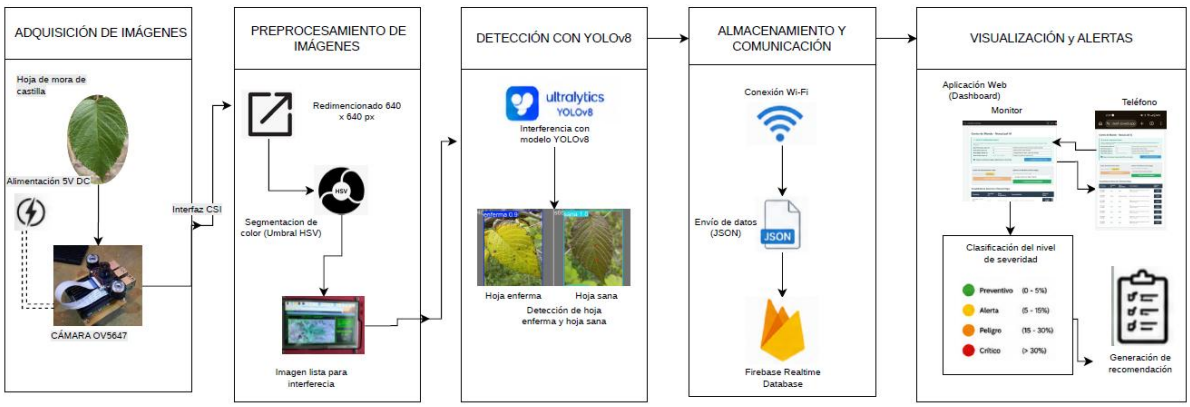
También se agrega el diagrama de bloques presentado en la Figura 13, el cual muestra el flujo completo del sistema IoT desarrollado, compuesto por cinco etapas: adquisición de

imágenes, preprocesamiento, detección con YOLOv8, almacenamiento y comunicación, y visualización y alertas.

El proceso inicia con la *adquisición de imágenes*, donde la cámara OV5647 captura la fotografía de la hoja de mora de Castilla y la transmite a la Raspberry Pi 4. Posteriormente, en la etapa de *preprocesamiento*, se emplean técnicas de visión por computador utilizando la biblioteca OpenCV para realizar el redimensionamiento de imagen a 640×640 píxeles y se aplica la segmentación de color mediante el espacio HSV, dejándola lista para la inferencia. En la etapa de *detección*, el modelo YOLOv8n clasifica la hoja como sana o enferma. A continuación, en la etapa de *almacenamiento y comunicación*, los resultados son estructurados en formato JSON y enviados mediante conexión Wi-Fi hacia Firebase Realtime Database. En la etapa de *visualización y alertas*, la aplicación web presenta el historial de detecciones, clasifica el nivel de severidad (preventivo, alerta, peligro o crítico) y genera la recomendación correspondiente para el usuario.

Figura 13

Diagrama de bloques del Sistema IoT



Nota: *Autoría Propia*

3.5.1.1 Desarrollo de comunicación inalámbrica

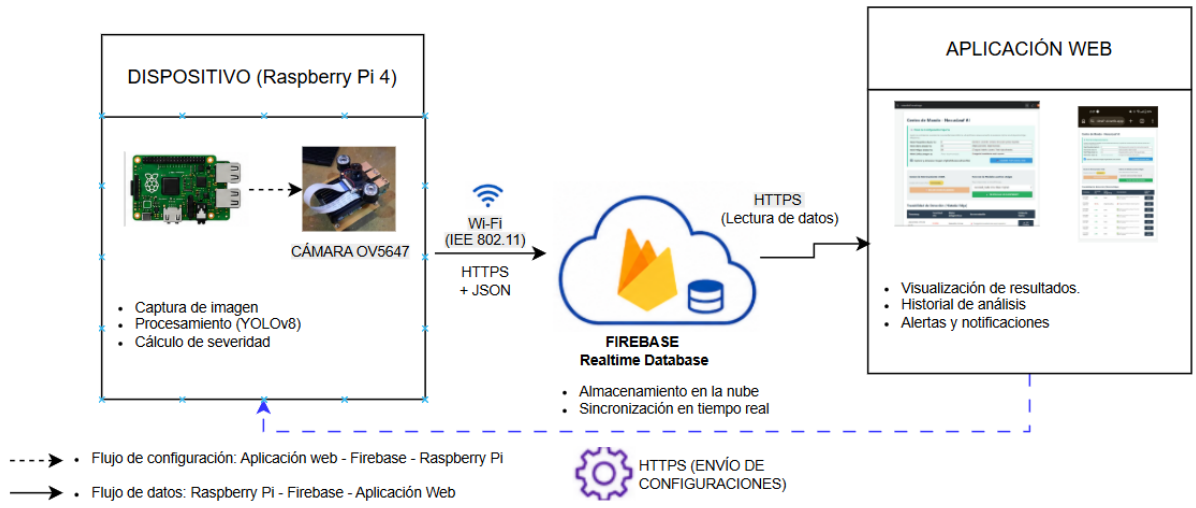
La comunicación entre los componentes del sistema se realizó de forma inalámbrica mediante el protocolo Wi-Fi (IEEE 802.11), permitiendo la transmisión de datos entre la

Raspberry Pi 4, el servidor Firebase y la aplicación web sin requerir conexiones físicas adicionales entre los dispositivos. En la Figura 14 se presenta el diagrama de comunicación inalámbrica del sistema, donde se observan los dos flujos de información que permiten el funcionamiento en tiempo real del sistema. El *flujo de datos* se origina en la Raspberry Pi 4, donde la cámara OV5647 captura la imagen, el modelo YOLOv8 la procesa y se calcula el porcentaje de severidad; este resultado es enviado mediante Wi-Fi (IEEE 802.11) y el protocolo HTTPS en formato JSON hacia Firebase Realtime Database, el cual almacena la información en la nube y la sincroniza en tiempo real hacia la aplicación web, donde el usuario puede visualizar los resultados, el historial de análisis y las alertas generadas.

De manera complementaria, el *flujo de configuración* permite que los parámetros definidos por el usuario desde la aplicación web como los umbrales de severidad y las recomendaciones sean enviados mediante HTTPS hacia Firebase, y desde ahí recibidos automáticamente por la Raspberry Pi, sin necesidad de reiniciar el dispositivo. Esta comunicación bidireccional garantiza que el sistema se mantenga sincronizado y actualizado en todo momento.

Figura 14

Diagrama de comunicación inalámbrica



Nota: Autoría Propia

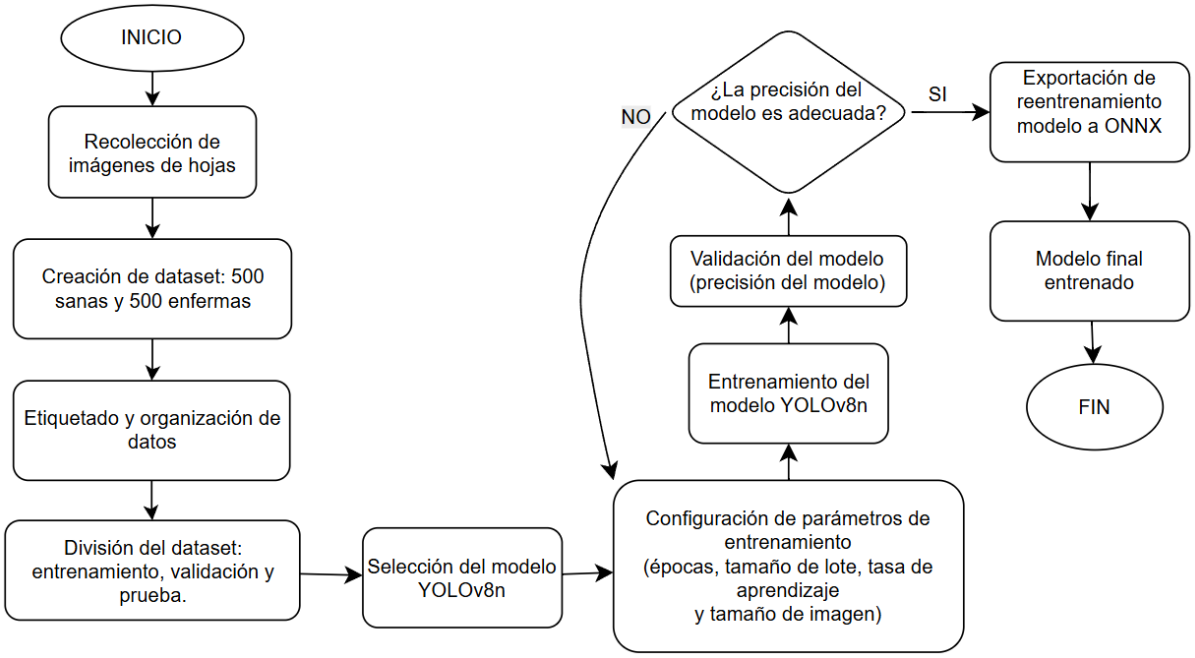
3.5.2 Desarrollo del Modelo de Detección

En esta etapa se desarrolló el modelo de detección encargado de identificar síntomas visibles de enfermedad en hojas de mora de Castilla (Rubus glaucus). Para ello, se llevó a cabo la creación y organización del dataset, la selección y el entrenamiento del modelo, así como su exportación a un formato compatible con Raspberry Pi 4 para su posterior integración y ejecución en tiempo real. De manera complementaria, el sistema incorporó una etapa de preprocesamiento mediante OpenCV y segmentación en el espacio de color HSV, orientada a mejorar el análisis inicial de la imagen y estimar la severidad de la afectación. Los resultados generados son almacenados en la nube y empleados para la emisión de alertas de monitoreo.

En la Figura 15 se presenta el diagrama de flujo del entrenamiento del modelo de detección. El proceso inició con la recolección de imágenes y la creación del dataset, seguido del etiquetado, organización y división de los datos en subconjuntos de entrenamiento, validación y prueba. Posteriormente, se seleccionó la arquitectura YOLOv8n y se configuraron los parámetros de entrenamiento. Una vez entrenado el modelo, se evaluó su desempeño; en caso de no obtener resultados adecuados, se realizó un ajuste de parámetros y un nuevo entrenamiento. Finalmente, el modelo validado fue exportado a formato ONNX para su posterior implementación.

Figura 15

Proceso de entrenamiento del modelo Yolo.



Nota: Autoría propia

3.5.2.1 Creación del dataset

Para el desarrollo del modelo de detección se construyó un dataset propio a partir de imágenes de hojas de mora de Castilla. Este conjunto de datos estuvo conformado por 1.000 imágenes, distribuidas en 500 imágenes de hojas sanas y 500 imágenes de hojas con síntomas visibles de enfermedad. La organización balanceada del dataset permitió disponer de una cantidad similar de ejemplos de cada clase, lo que favoreció el entrenamiento del modelo.

Las imágenes fueron capturadas con el propósito de obtener muestras representativas de ambas clases y conformar una base de datos adecuada para el desarrollo del sistema. Posteriormente, las imágenes fueron revisadas, seleccionadas y organizadas de acuerdo con la presencia de síntomas visibles de enfermedad, clasificándolas en dos categorías: sano y enfermo.

En esta etapa, se importaron las librerías necesarias presentadas en la Tabla 17, para facilitar el manejo, la visualización y el análisis de los datos. Posteriormente, se cargó el dataset obtenido, estableciendo la base para las siguientes etapas del desarrollo del modelo.

Tabla 13

Librerías utilizadas

Nombre de Librería	Función
Os	Manejo de rutas y lectura de carpetas del sistema de archivos
Shutil	Copia de imágenes hacia las carpetas de entrenamiento y validación
Random	División aleatoria y reproducible de las imágenes en los subconjuntos train y val.

3.5.2.2 Etiquetado y organización de datos

El etiquetado de las imágenes se realizó de forma implícita mediante la organización de los archivos en carpetas nombradas según su clase correspondiente. Este método es compatible con la arquitectura YOLOv8n en su modalidad de clasificación, la cual asigna automáticamente la etiqueta de cada imagen en función del nombre de la carpeta que la contiene.

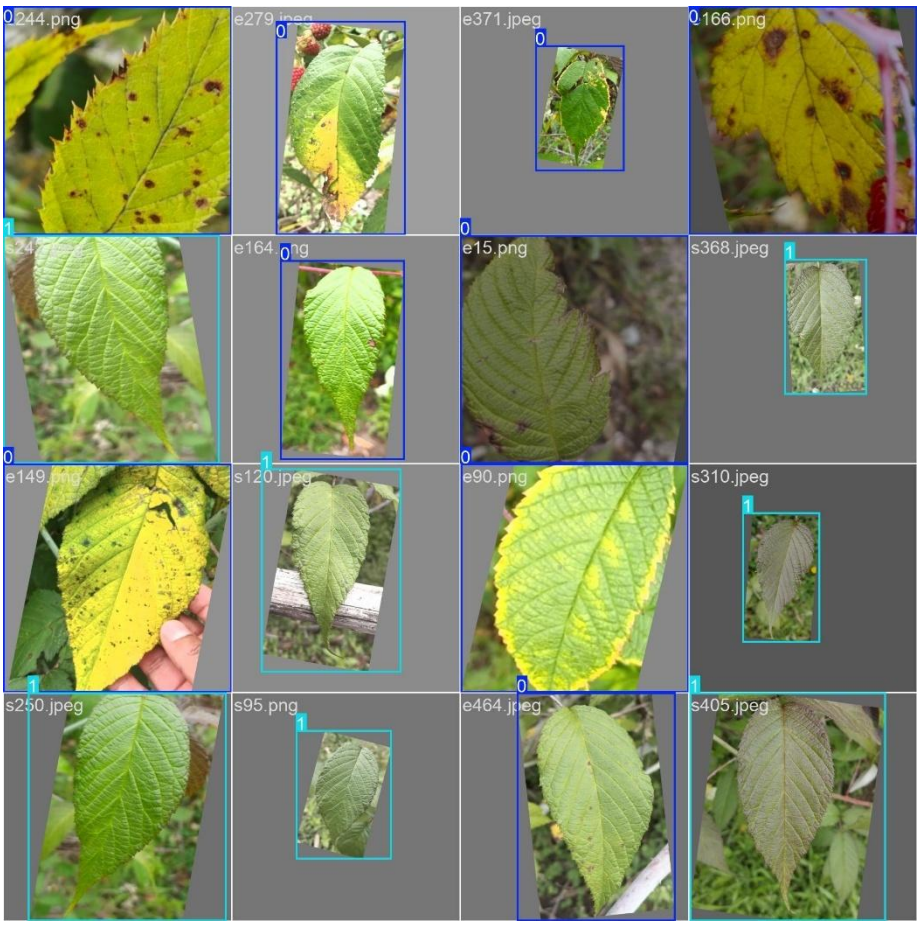
Las imágenes fueron organizadas en dos carpetas principales: sano y mildiu. Posteriormente, mediante el script de Python descrito en la sección anterior, cada carpeta fue dividida en dos subconjuntos: el 80% destinado al entrenamiento y el 20% restante reservado para la validación del modelo, obteniendo un total de 783 imágenes para entrenamiento y 197 imágenes para validación

En la Figura 16 se muestra el etiquetado de las imágenes que se delimitaron mediante cajas (bounding boxes) las áreas afectadas por el mildiu polvoso en las hojas de mora de la Castilla, utilizando el formato YOLOv8, el cual permite definir las zonas donde se presentan

los síntomas visibles de mildiu polvoso mediante bounding boxes. Este proceso se llevó a cabo con herramientas de anotación, asignando la clase correspondiente a cada región detectada.

Figura 16

Proceso de etiquetado del dataset para la detección de mildiu polvoso mediante bounding boxes.

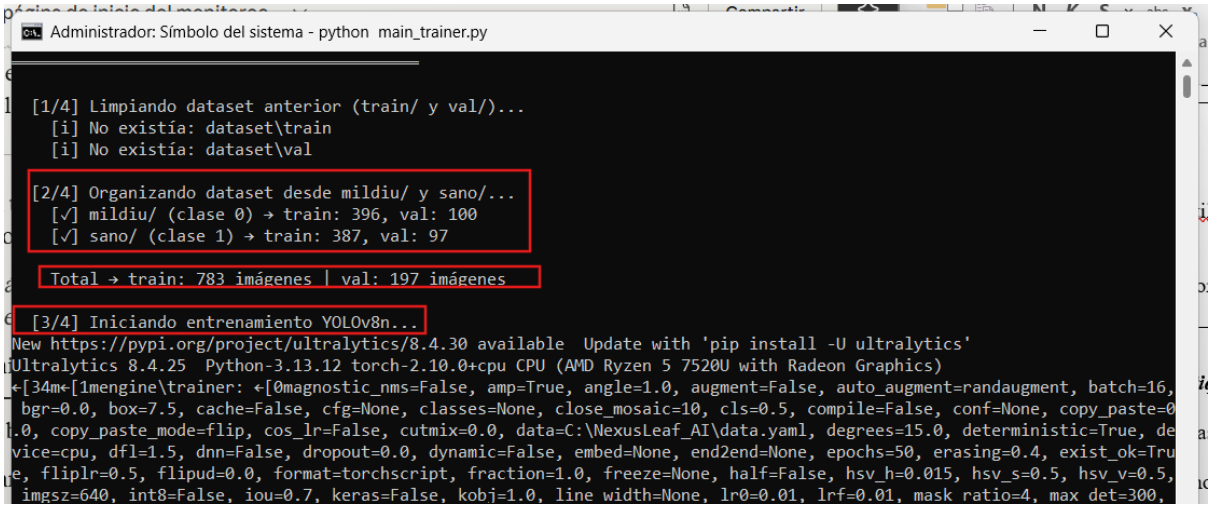


Nota: *Autoría propia*

Por otro lado, en la figura 16 se observa el proceso de organización del dataset ejecutado mediante el script de Python, donde las imágenes fueron distribuidas automáticamente en los subconjuntos de entrenamiento y validación, obteniendo un total de 783 imágenes para entrenamiento y 197 imágenes para validación.

Figura 17

Proceso de organización y división del dataset



Nota: Autoría propia.

3.5.2.3 Selección del modelo de detección

Para el desarrollo del sistema de detección se seleccionó la arquitectura YOLOv8n (You Only Look Once versión 8, variante nano), perteneciente a la familia de modelos de la librería Ultralytics. Esta arquitectura fue elegida debido a su equilibrio entre precisión y velocidad de inferencia, características esenciales para su ejecución en un dispositivo de recursos limitados como la Raspberry Pi 4.

El modelo fue configurado con clasificación de imágenes, lo que permitió distinguir entre dos categorías: hojas sanas y hojas enfermas. Para su integración en la Raspberry Pi 4, el modelo entrenado fue exportado al formato ONNX (Open Neural Network Exchange), el cual permite su ejecución mediante el motor de inferencia onnxruntime, garantizando compatibilidad con arquitecturas ARM sin necesidad de instalar frameworks adicionales como PyTorch.

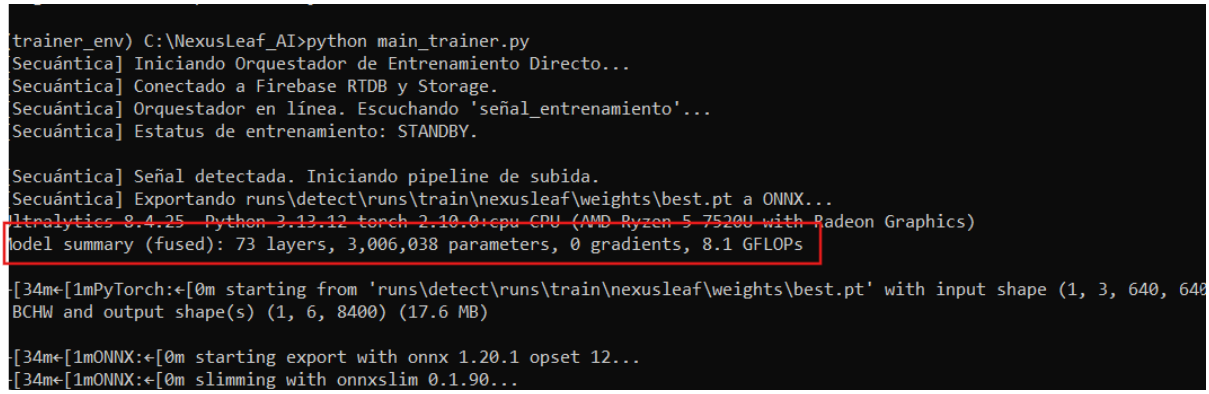
De manera complementaria, el sistema incorporó un método de análisis basado en segmentación de color en el espacio HSV (Hue, Saturation, Value) mediante la librería OpenCV. Este método analiza los píxeles de la imagen capturada para identificar zonas

afectadas por mildiu a partir de sus características de color, calculando el porcentaje de severidad de la infección como la proporción de píxeles enfermos respecto al total de píxeles de la hoja detectada. Este análisis complementa la clasificación del modelo YOLOv8n, proporcionando un indicador numérico de la severidad expresado en porcentaje.

En la Figura 18 se observa el resumen de la arquitectura YOLOv8n utilizada, la cual cuenta con 73 capas, 3.006.038 parámetros y una carga computacional de 8.1 GFLOPs, características que la hacen adecuada para su ejecución en dispositivos de recursos limitados como la Raspberry Pi 4.

Figura 18

Resumen de arquitectura del modelo YOLOv8n



Nota: *Autoría propia*

En la Tabla 14 se presentan las características principales del modelo y las herramientas seleccionadas para el sistema de detección.

Tabla 14

Características del modelo seleccionado

Parámetro	Valor
Arquitectura	YOLOv8n
Modalidad	Clasificación
Número de capas	73
Parámetros	3.006.038

Carga computacional	8.1 GFLOPs
Formato de despliegue	ONNX
Tamaño de imagen de entrada	640 × 640 px
Motor de inferencia	onnxruntime
Preprocesamiento visual	OpenCV — Segmentación HSV
Número de clases	Sano/Enfermo

3.5.2.4 Entrenamiento del modelo

El entrenamiento del modelo tuvo como finalidad ajustar el comportamiento de la arquitectura YOLOv8n para mejorar su capacidad de clasificación entre hojas sanas y hojas con enfermedad. Para ello, se utilizó Python como lenguaje principal y la librería Ultralytics, la cual permitió ejecutar el proceso de entrenamiento del modelo a partir del dataset previamente organizado. En esta etapa se definieron parámetros básicos como el número de épocas y el tamaño de imagen de entrada, con el fin de controlar el proceso de aprendizaje y optimizar el desempeño del modelo.

El modelo fue entrenado durante 50 épocas con imágenes de 640 × 640 píxeles y un tamaño de lote de 16. Además, se aplicaron ajuste de saturación y brillo en el espacio de color HSV, y volteo horizontal aleatorio, con el propósito de mejorar la capacidad de generalización del modelo. Durante este proceso, el sistema ajustó progresivamente sus parámetros internos en cada iteración. Finalmente, el modelo validado fue exportado a formato ONNX para facilitar su integración en la Raspberry Pi 4 y permitir su ejecución.

En la Figura 19 se presenta el fragmento del código correspondiente a la configuración del entrenamiento del modelo YOLOv8n, donde se observan los principales parámetros definidos para esta etapa, como el número de épocas, el tamaño de imagen de entrada, el tamaño de lote y las técnicas de aumento de datos.

Figura 19

Configuración de parámetros del entrenamiento del modelo YOLOv8n.

```python
def _entrenar_y_exportar(self):
    print("\n  [3/4] Iniciando entrenamiento YOLOv8n...")

    data_path = os.path.abspath("data.yaml")
    model = YOLO("yolov8n.pt")

    model.train(
        data=data_path,
        epochs=50,
        imgsz=640,
        batch=16,
        degrees=15.0,
        scale=0.5,
        hsv_s=0.5,
        hsv_v=0.5,
        fliplr=0.5,
        project="runs/train",
        name="nexusleaf",
        exist_ok=True,
    )

    print("\n  [Secuántica] Entrenamiento finalizado. Buscando best.pt...")
```

Nota: Autoría propia

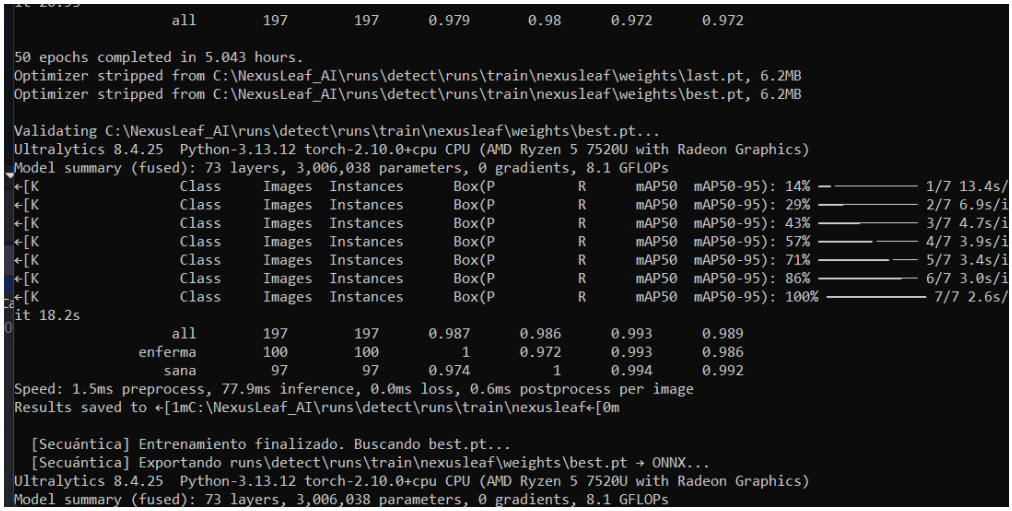
3.5.2.4.1 *Evaluación del modelo*

Una vez finalizado el entrenamiento, el sistema evaluó automáticamente el desempeño del modelo utilizando el conjunto de validación conformado por 197 imágenes que el modelo no había visto durante el entrenamiento. Cabe mencionar que el dataset original estuvo conformado por 1.000 imágenes, distribuidas en 500 imágenes de hojas enfermas y 500 imágenes de hojas sanas. Tras aplicar la división del 80% para entrenamiento y 20% para validación, se obtuvieron 783 imágenes para entrenamiento y 197 imágenes para validación.

En la Figura 20 se presentan los resultados finales obtenidos, donde se puede observar que el modelo alcanzó una precisión general de 0.987, un recall de 0.986, un mAP@50 de 0.993 y un mAP@50-95 de 0.989, completando las 50 épocas.

Figura 20

Resultados finales del entrenamiento del modelo YOLOv8n

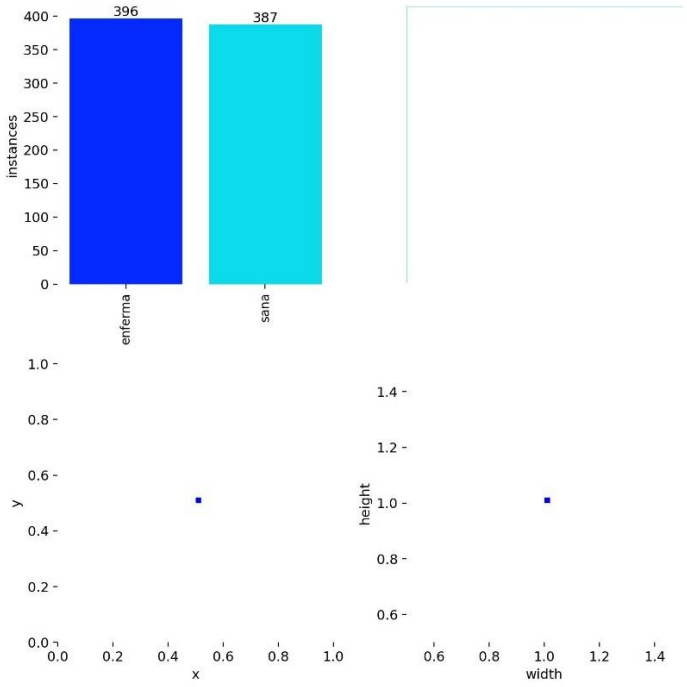


Nota: Autoría propia

En la Figura 21 se presenta la distribución de clases utilizada durante el entrenamiento, con 396 imágenes de hojas enfermas y 387 imágenes de hojas sanas, lo que refleja una distribución balanceada entre ambas clases que favoreció el aprendizaje equitativo del modelo.

Figura 21

Distribución de clases en el dataset de entrenamiento



Nota: Autoría propia

En la Figura 22 se presentan las curvas de pérdida y métricas obtenidas a lo largo de las 50 épocas, clasificadas en las siguientes secciones:

a) train/box_loss: Representa la pérdida de localización durante el entrenamiento, es decir, el error del modelo al delimitar la posición de la hoja en la imagen. Se observa una disminución constante desde un valor inicial de 0.85 hasta estabilizarse cerca de 0.10 al finalizar las 50 épocas, lo que indica que el modelo mejoró progresivamente su capacidad para ubicar correctamente el objeto de interés dentro de la imagen.

b) train/cls_loss: Representa la pérdida de clasificación durante el entrenamiento, es decir, el error del modelo al asignar la clase correcta (sana o enferma) a la imagen analizada. La curva desciende de forma pronunciada desde un valor inicial cercano a 2.0 hasta estabilizarse por debajo de 0.2, evidenciando que el modelo aprendió a diferenciar correctamente entre ambas clases conforme avanzó el entrenamiento.

c) train/dfl_loss: Representa la pérdida de distribución focal, una métrica que ajusta con mayor precisión los límites de detección del modelo. Esta curva muestra un comportamiento similar a las anteriores, descendiendo de 1.5 a valores cercanos a 0.9, lo que confirma un ajuste progresivo y estable de los parámetros internos de la red.

d) ***metrics/precision(B):*** Representa la precisión del modelo durante el entrenamiento, es decir, la proporción de detecciones correctas sobre el total de detecciones realizadas. Se observa que, tras una fase inicial de oscilación entre las épocas 0 y 10 producto del ajuste inicial de pesos, la precisión se estabiliza por encima de 0.95 a partir de la época 15, manteniéndose constante hasta el final del entrenamiento.

e) ***metrics/recall(B):*** Representa el recall del modelo, es decir, la capacidad de detectar correctamente todos los casos positivos presentes en las imágenes. La curva muestra una

recuperación rápida desde un valor inicial bajo de 0.35 hasta superar 0.95 alrededor de la época 10, manteniéndose estable en valores cercanos a 1.0 durante el resto del entrenamiento.

f) val/box_loss: Representa la pérdida de localización sobre el conjunto de validación, es decir, imágenes que el modelo no utilizó durante el aprendizaje. Esta curva presenta un pico inicial cercano a 2.1 en las primeras épocas, seguido de un descenso sostenido hasta estabilizarse alrededor de 0.20, lo que indica que el modelo generalizó correctamente sin memorizar el conjunto de entrenamiento.

g) val/cls_loss: Representa la pérdida de clasificación sobre el conjunto de validación. La curva muestra mayor variabilidad en las primeras 20 épocas, con picos de hasta 3.0, para luego estabilizarse por debajo de 0.5, confirmando que el modelo mejoró su capacidad de clasificación incluso en imágenes nuevas no vistas durante el entrenamiento.

h) val/dfl_loss: Representa la pérdida de distribución focal sobre el conjunto de validación. Al igual que las métricas anteriores, presenta un descenso progresivo desde 2.6 hasta estabilizarse cerca de 0.55, evidenciando consistencia entre el comportamiento de entrenamiento y validación.

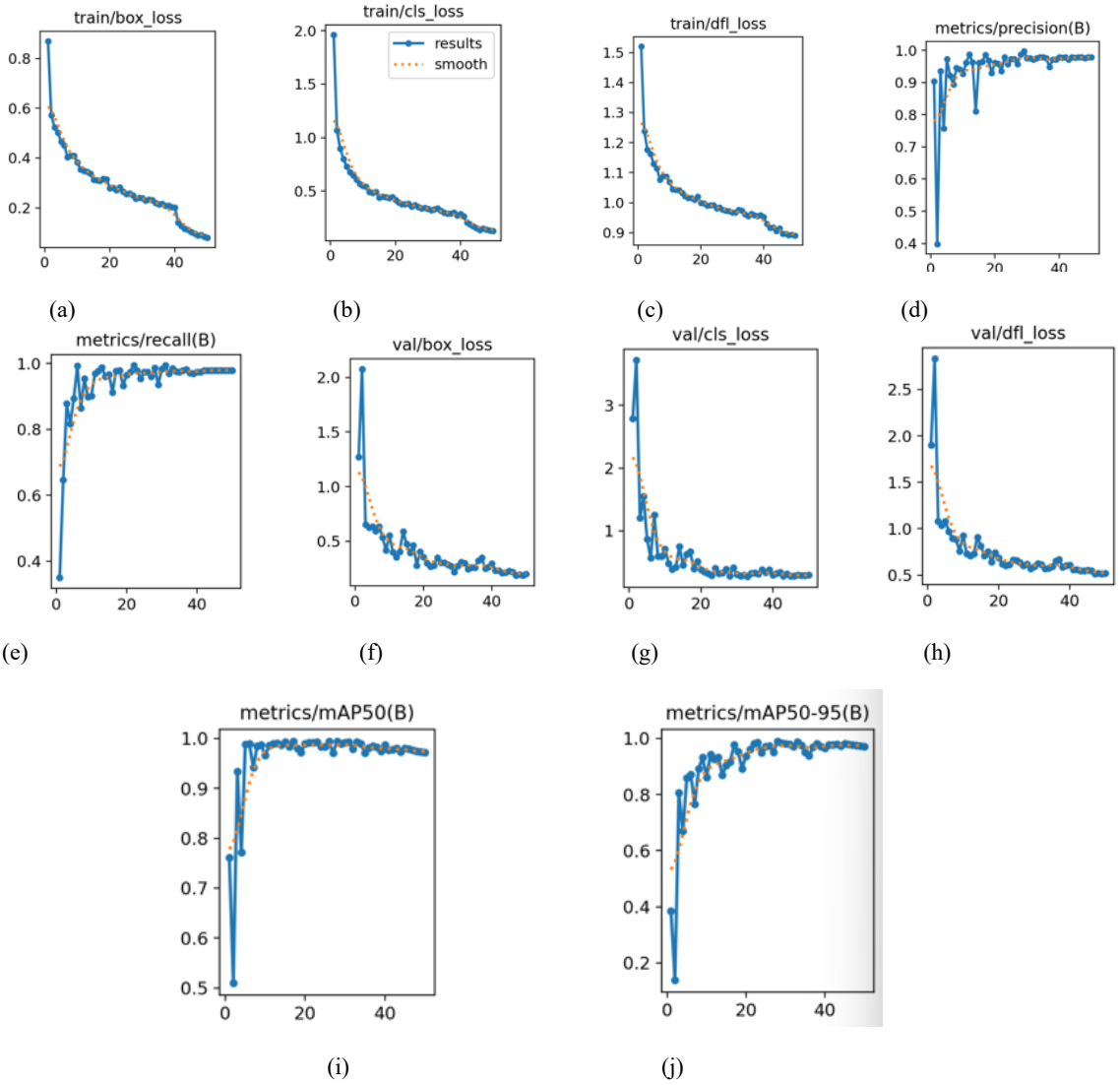
i) metrics/mAP50(B): Representa la media de precisión promedio con un umbral de superposición del 50%, una de las métricas más importantes para evaluar modelos de detección. La curva asciende rápidamente desde 0.75 hasta superar 0.99 antes de la época 15, manteniéndose estable en ese nivel durante el resto del entrenamiento, lo que demuestra un alto desempeño del modelo en la tarea de detección.

j) metrics/mAP50-95(B): Representa la media de precisión promedio evaluada en un rango más estricto de umbrales de superposición, desde 0.5 hasta 0.95. Esta métrica, más exigente que la anterior, muestra un crecimiento sostenido desde 0.15 hasta estabilizarse cerca

de 0.99, confirmando que el modelo no solo detecta correctamente la presencia de la enfermedad, sino que también delimita con precisión la región afectada.

Figura 22

Curvas de entrenamiento y validación del modelo YOLOv8n



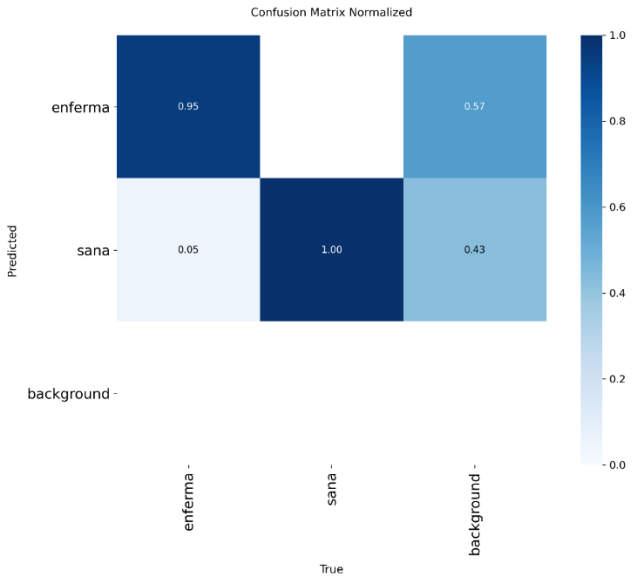
Nota: Autoría propia

En la Figura 23 se presenta la matriz de confusión normalizada, que muestra cuántas veces el modelo acertó y cuántas se equivocó al clasificar las imágenes. Se observa que el modelo identificó correctamente el 95% de las hojas enfermas y el 100% de las hojas sanas, confirmando su alta capacidad de distinción entre ambas clases. El 5% restante de hojas

enfermas fue clasificado incorrectamente como sano, lo que representa un margen de error mínimo que no compromete el desempeño general del sistema.

Figura 23

Matriz de confusión normalizada

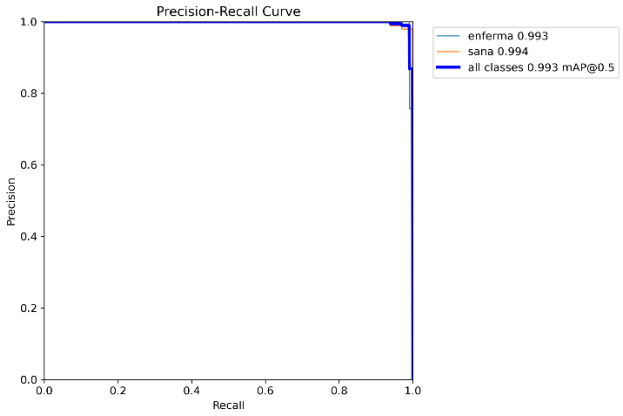


Nota: Autoría propia

En la Figura 24 se presenta la curva Precisión-Recall, que muestra el equilibrio entre la capacidad del modelo para detectar correctamente todos los casos enfermos y su precisión al hacerlo. Una curva cercana a la esquina superior derecha indica un modelo de alto desempeño. El modelo obtuvo un mAP@0.5 de 0.993 para la clase enferma y 0.994 para la clase sana, lo que refleja que el sistema es capaz de detectar hojas enfermas con alta precisión sin generar falsas alarmas, característica esencial en un sistema de detección de enfermedades agrícolas donde omitir un caso enfermo puede tener consecuencias graves para el cultivo.

Figura 24

Curva Precisión-Recall

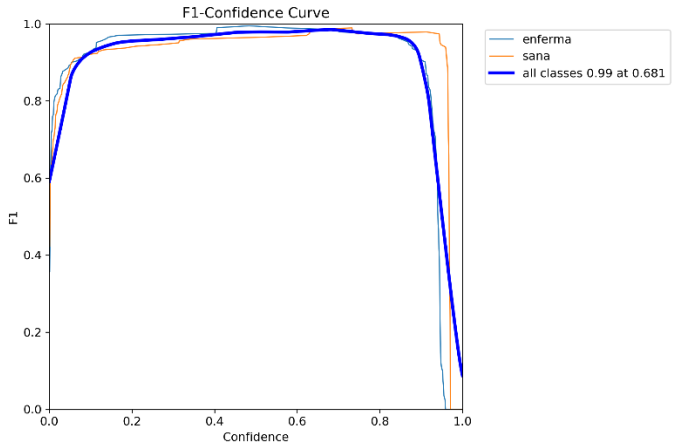


Nota: Autoría propia

En la Figura 25 se presenta la curva F1-Confianza, que combina la precisión y el recall en un único indicador llamado F1, permitiendo evaluar el desempeño global del modelo con un solo valor. El modelo alcanzó un valor máximo de F1 = 0.99 a una confianza de 0.681, lo que demuestra que el modelo mantiene un equilibrio óptimo entre detectar correctamente las hojas enfermas y evitar clasificaciones incorrectas, siendo este resultado consistente para ambas clases del sistema.

Figura 25

Curva F1-Confianza



Nota: *Autoría propia*

3.5.2.4.2 *Comparación de modelo*

Para determinar la configuración óptima del modelo, se realizaron pruebas experimentales variando el número de épocas de entrenamiento (30, 50, 75 y 100), manteniendo los parámetros. En la Tabla 15, se evalúa el desempeño en función de la precisión, el recall, las métricas mAP.

Tabla 15

Comparación de Resultados

Épocas	Precisión	Recall	mAP50	Observación
30	0.979	0.980	0.994	Menor estabilidad en el aprendizaje.
50	0.987	0.986	0.993	Óptimo equilibrio desempeño/tiempo.
75	0.976	0975	0.991	Saturación sin mejora significativa
100	0.977	0.980	0.991	Saturación sin mejora significativa

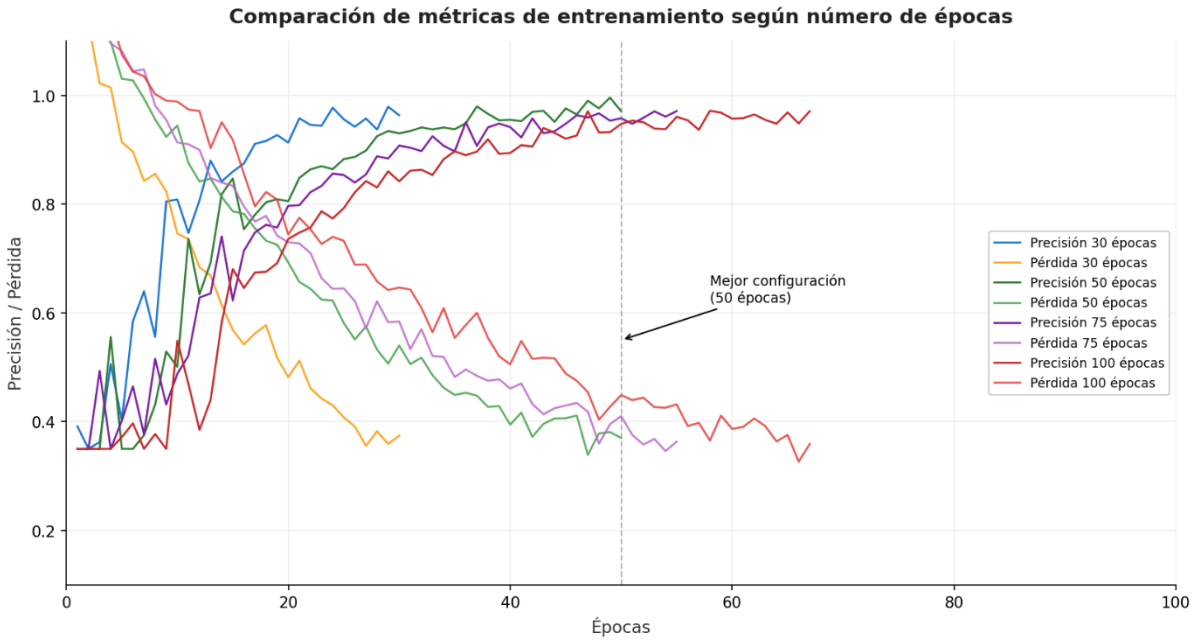
Se observa que el entrenamiento con 30 épocas alcanzó valores altos de desempeño; sin embargo, el modelo aún podía mejorar su capacidad de generalización debido a la mayor variabilidad presentada en las curvas de precisión durante el proceso de aprendizaje. Por otro lado, al incrementar el número de épocas configuradas a 75 y 100, el tiempo de procesamiento aumentó considerablemente, activándose en ambos casos el mecanismo de parada anticipada en las épocas 55 y 67 respectivamente, sin obtener mejoras relevantes en las métricas principales respecto a la configuración de 50 épocas, lo que evidencia un rendimiento poco eficiente al extender innecesariamente el entrenamiento.

En cambio, con 50 épocas se obtuvo el mejor equilibrio entre precisión, recall, capacidad predictiva y tiempo de entrenamiento. Además, presentó el valor más alto de precisión general (0.987) y un excelente mAP50-95 (0.989), lo cual indica una mejor detección de objetos en diferentes niveles de coincidencia.

En la Figura 26, la comparativa de métricas revela que la configuración de 50 épocas presenta la curva de aprendizaje más óptima. Mientras que a las 30 épocas el modelo muestra mayor inestabilidad en la precisión durante las primeras iteraciones, y tanto a las 75 como a las 100 épocas se observa una saturación del aprendizaje sin ganancias marginales confirmada por la activación del early stopping en ambos casos, el punto de las 50 épocas representa la convergencia ideal, donde la pérdida se minimiza y la precisión alcanza su máximo nivel de estabilidad sin necesidad de un entrenamiento más prolongado.

Figura 26

Curva de aprendizaje



Nota: Autoría propia

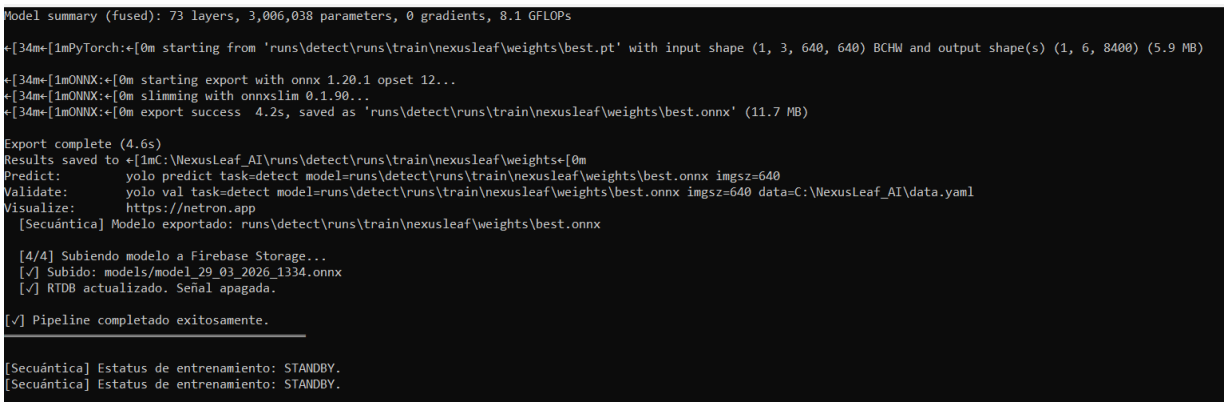
3.5.2.5 Importación e integración del modelo en Raspberry Pi

Para la importación del modelo entrenado al formato ONNX, se procedió a su integración en la Raspberry Pi 4 con el fin de permitir su ejecución en tiempo real. Este proceso se realizó de manera automática mediante la comunicación entre el PC de entrenamiento y la Raspberry Pi a través de Firebase, eliminando la necesidad de transferir el modelo manualmente entre dispositivos. El proceso de integración siguió los siguientes pasos:

- Primero, el script main_trainer.py ejecutado en el PC localizó el archivo best.pt generado durante el entrenamiento y lo exportó al formato ONNX con opset 12, garantizando compatibilidad con la versión de OpenCV instalada en la Raspberry Pi 4.
- Segundo, el modelo exportado fue subido automáticamente a Firebase Storage con un nombre que incluye la fecha y hora de generación.
- Tercero, Firebase Realtime Database fue actualizado con el nombre del nuevo modelo, notificando a la Raspberry Pi que existe una actualización disponible.

En la Figura 27 se puede observar este proceso completo, donde el sistema confirmó la exportación exitosa del modelo y su subida a Firebase Storage.

Figura 27
Proceso de exportación y subida del modelo a Firebase Storage



Nota: *Autoría propia*

Para la integración del modelo en la Raspberry Pi, escucha permanentemente los cambios en Firebase Realtime Database. Cuando detecta que existe un nuevo modelo disponible, lo descarga automáticamente desde Firebase Storage y lo almacena en la carpeta models/ del dispositivo. Una vez descargado, el modelo es cargado en memoria para realizar inferencias sobre las imágenes capturadas por la cámara en tiempo real.

En la Figura 28 se presenta el código correspondiente a la carga del modelo ONNX en la Raspberry Pi 4, donde se puede observar cómo el sistema localiza el archivo del modelo en la carpeta models/, lo carga en memoria mediante la función readNetFromONNX de OpenCV y confirma su correcta inicialización con un mensaje en pantalla. En caso de que el modelo no pueda ser cargado, el sistema registra el error y continúa funcionando sin detener la aplicación, garantizando la estabilidad del dispositivo.

Figura 28

Integración del modelo ONNX en Raspberry Pi 4

```
def load_onnx_model(model_filename): #  C
    global net
    model_path = os.path.join("..", "models", model_filename) # conecta la carpeta
    print(f"[Secuántica] Inicializando Motor ONNX: {model_path}...")
    try:
        net = cv2.dnn.readNetFromONNX(model_path)
        print(f"[Secuántica] {model_filename} montado en memoria con éxito.")
    except Exception as e:
        print(f"[ERROR] Falla montando ONNX {model_filename}: {e}")
        net = None

# Montamos el cerebro base
load_onnx_model("nexusleaf_model.onnx") # modelo entrenado para la rrapberry.
```

Nota: *Autoría propia*

En la Tabla 16 se presentan las librerías utilizadas en esta etapa de integración.

Tabla 16

Librerías utilizadas en la integración del modelo en Raspberry Pi

Librería	Función

onnxruntime	Carga y ejecución del modelo ONNX en Raspberry Pi 4
firebase_admin	Conexión con Firebase RTDB y descarga del modelo desde Storage
cv2 (OpenCV)	Carga del modelo ONNX mediante readNetFromONNX
os	Manejo de rutas y carpetas del sistema de archivos

3.5.2.5.3 *Inicialización y arranque del sistema en Raspberry Pi*

Una vez realizada la conexión de los archivos en nuestro sistema operativo raspberry la secuencia de inicialización sigue el siguiente orden.

Primero el sistema establece la conexión con Firebase y sincroniza la política de configuración, activando la opción de guardar imagen, en la figura 29 se muestra el segmento del código.

Figura 29

Conexión con Firebase y política

```
# Suscribir escucha para la política de configuración de la Web App
self.config_ref = db.reference('configuracion')
self.config_ref.listen(self._on_config_change)

# Escuchar actualizaciones de modelo
self.modelo_ref = db.reference('modelo_config')
self.modelo_ref.listen(self._on_modelo_change)
```

Nota: *Autoría propia*

Segundo detecta que existe un nuevo modelo disponible en la nube e inicia la descarga automática. En la figura 30 se muestra el segmento del código.

Figura 30

Detección de nuevo modelo

```
# Escuchar actualizaciones de modelo
self.modelo_ref = db.reference('modelo_config')
self.modelo_ref.listen(self._on_modelo_change)
```

Nota: *Autoría propia*

Tercera carga el modelo base nexusleaf_model.onnx en memoria mediante onnxruntime, confirmando su correcta inicialización con el mensaje "nexusleaf_model.onnx montado en memoria con éxito". En la figura 31 se muestra el segmento del código.

Figura 31

Carga del modelo base

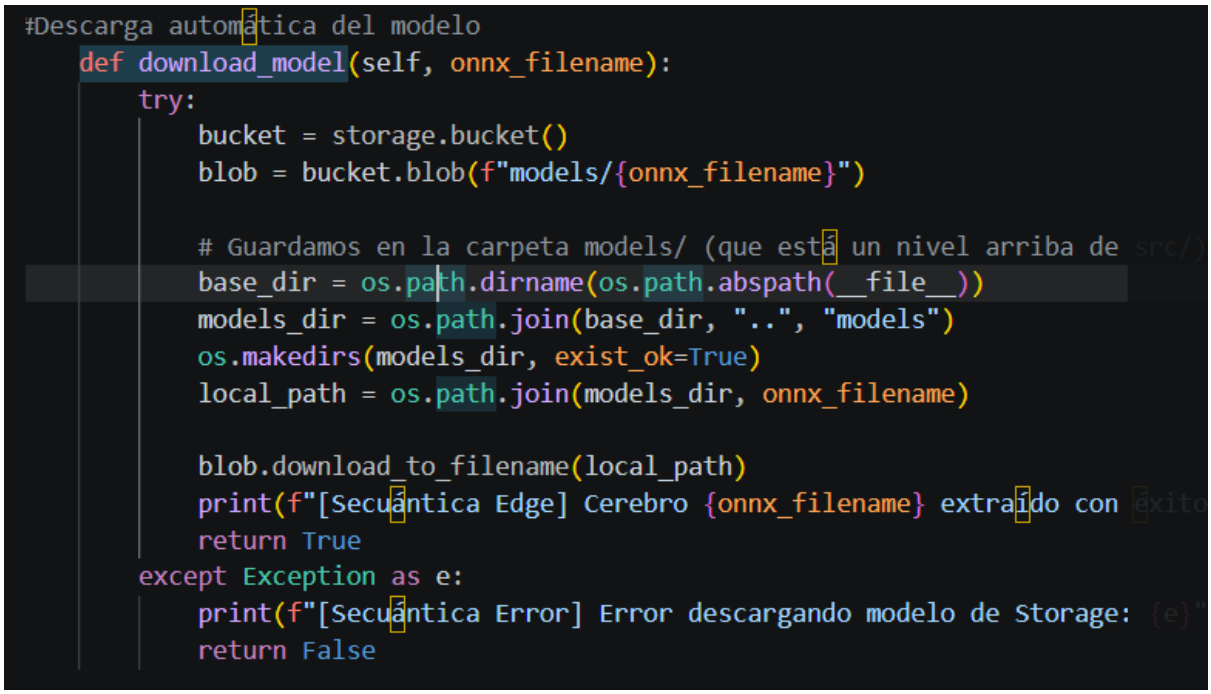
```
# Carga del modelo base
def load_onnx_model(model_filename):
    global net
    model_path = os.path.join(os.path.dirname(__file__), "..", "models", model_filename)
    model_path = os.path.abspath(model_path)
    print(f"[Secuántica] Inicializando Motor ONNX: {model_path}...")
    try:
        net = ort.InferenceSession(model_path) # Carga del modelo ONNX
        print(f"[Secuántica] {model_filename} montado en memoria con éxito.")
    except Exception as e:
        print(f"[ERROR] Falla montando ONNX {model_filename}: {e}")
        net = None
```

Nota: *Autoría propia*

Cuarto descarga y despliega el modelo actualizado model_25_04_2026_1819.onnx desde Firebase Storage, reemplazando el modelo anterior sin necesidad de reiniciar el sistema. En la figura 32 se muestra el segmento del código.

Figura 32

Descarga del modelo nuevo



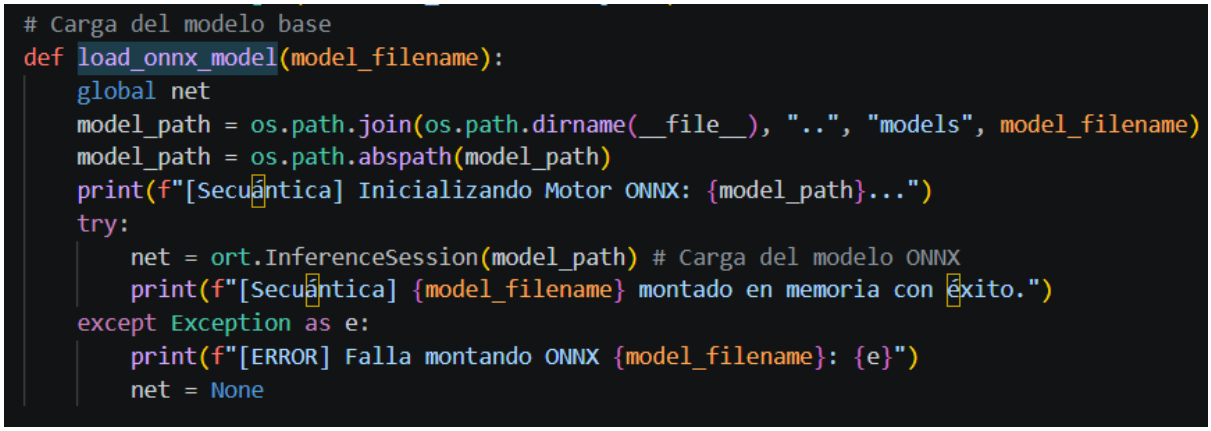
```python
#Descarga automática del modelo
    def download_model(self, onnx_filename):
        try:
            bucket = storage.bucket()
            blob = bucket.blob(f"models/{onnx_filename}")

            # Guardamos en la carpeta models/ (que está un nivel arriba de src/)
            base_dir = os.path.dirname(os.path.abspath(__file__))
            models_dir = os.path.join(base_dir, "..", "models")
            os.makedirs(models_dir, exist_ok=True)
            local_path = os.path.join(models_dir, onnx_filename)

            blob.download_to_filename(local_path)
            print(f"[Secuántica Edge] Cerebro {onnx_filename} extraído con éxito
            return True
        except Exception as e:
            print(f"[Secuántica Error] Error descargando modelo de Storage: {e}"
            return False
```

Nota: *Autoría propia*

Quinto confirma la carga del nuevo modelo con el mensaje "model_25_04_2026_1819.onnx montado en memoria con éxito", quedando el sistema listo para recibir análisis desde la interfaz gráfica. En la figura 33 se muestra el segmento del código.

Figura 33

Carga del modelo actualizado



```python
# Carga del modelo base
def load_onnx_model(model_filename):
    global net
    model_path = os.path.join(os.path.dirname(__file__), "..", "models", model_filename)
    model_path = os.path.abspath(model_path)
    print(f"[Secuántica] Inicializando Motor ONNX: {model_path}...")
    try:
        net = ort.InferenceSession(model_path) # Carga del modelo ONNX
        print(f"[Secuántica] {model_filename} montado en memoria con éxito.")
    except Exception as e:
        print(f"[ERROR] Falla montando ONNX {model_filename}: {e}")
        net = None
```

Nota: *Autoría propia*

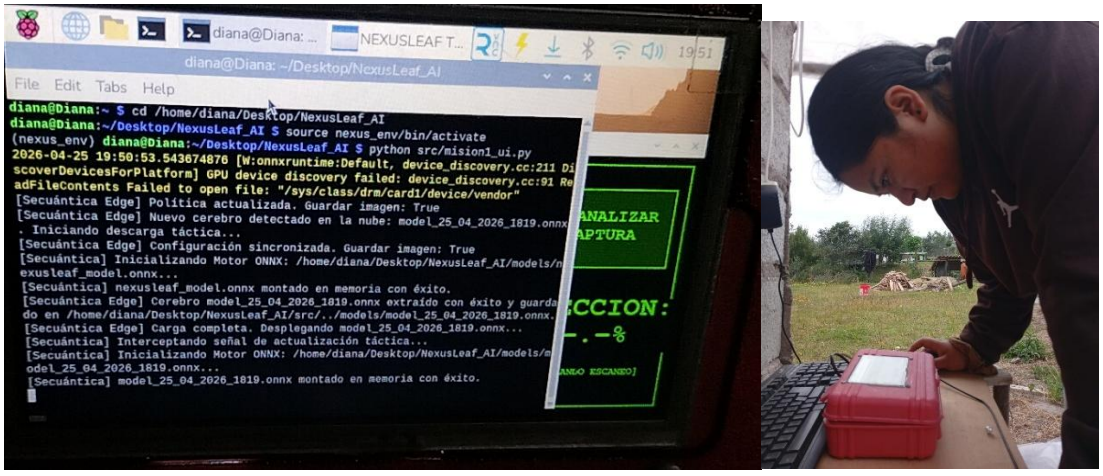
En la Figura 34 se presenta el proceso de arranque del sistema en la Raspberry Pi. Para iniciar el sistema se ejecutaron los siguientes comandos en la terminal:

- cd /home/diana/Desktop/NexusLeaf_AI
- source nexus_env/bin/activate
- python src/mision1_ui.py

El primer comando accede al directorio del proyecto NexusLeaf AI ubicado en el escritorio de la Raspberry Pi. El segundo comando activa el entorno virtual nexus_env, el cual contiene todas las librerías necesarias para el funcionamiento del sistema, como onnxruntime, opencv y firebase_admin. El tercer comando ejecuta el archivo principal mision1_ui.py, el cual al iniciarse desencadena automáticamente la siguiente secuencia de eventos.

Figura 34

Proceso de arranque del sistema en Raspberry Pi



Nota: *Autoría propia*

3.5.2.6 Flujo de inferencia y generación de alertas

Para el flujo de interferencia del sistema, se emplea la técnica de segmentación en el espacio de color HSV (Hue, Saturation, Value), la cual separa el color, la intensidad y la iluminación, lo que facilita identificar las zonas de la hoja. Gracias a eso es posible diferenciar las áreas sanas de las afectadas por mildiu mediante rangos de color, logrando un análisis más preciso incluso cuando hay cambios de luz.

Este flujo se activa cada vez que el usuario presiona el botón de análisis en la interfaz gráfica del dispositivo y sigue tres etapas principales: captura de imagen, análisis de la imagen y envío de resultados a la nube.

Primera etapa: Captura de imagen

La cámara OV5647 conectada a la Raspberry Pi 4 captura un fotograma en tiempo real de la hoja que el usuario coloca frente al dispositivo. La imagen es almacenada temporalmente en memoria para su procesamiento inmediato, sin necesidad de guardarla en el disco del dispositivo, lo que permite mantener una respuesta rápida del sistema. En la figura 35 se muestra el dispositivo para la captura de imagen.

Figura 35

Captura de Imagen



Nota: *Autoría propia*

Segunda etapa: Análisis de la imagen

Para el análisis se aplica un procesamiento basado en segmentación de color en el espacio HSV mediante OpenCV y la inferencia con el modelo YOLOv8n en formato ONNX, con el objetivo de identificar las zonas afectadas por mildiu. Para ello, se definen rangos de color que permiten diferenciar los píxeles correspondientes a la hoja y aquellos asociados a la enfermedad.

En el primer método, la imagen capturada es convertida al espacio de color HSV y se aplican dos máscaras de color. La primera máscara identifica los píxeles verdes correspondientes a la zona sana de la hoja, utilizando un rango de tono entre 35 y 85, saturación entre 40 y 255 y brillo entre 40 y 255. La segunda máscara identifica los píxeles blancos y grises correspondientes a las zonas afectadas por mildiu, utilizando un rango de tono entre 0 y 180, saturación entre 0 y 50 y brillo entre 150 y 255. En la Figura 35 se presenta el código correspondiente a esta implementación.

Figura 36

Implementación de la segmentación HSV y cálculo de severidad

```
    # 1. Espectrometría HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    lower_green = np.array([35, 40, 40])
    upper_green = np.array([85, 255, 255])
    mask_leaf = cv2.inRange(hsv, lower_green, upper_green)

    lower_mildiu = np.array([0, 0, 150])
    upper_mildiu = np.array([180, 50, 255])
    mask_mildiu = cv2.inRange(hsv, lower_mildiu, upper_mildiu)

    leaf_pixels = cv2.countNonZero(mask_leaf)
    mildiu_pixels = cv2.countNonZero(mask_mildiu)

    if leaf_pixels < 5000:
        self.lbl_severity.configure(text="No hay\nhoja", text_color="#FF4500")
        self.lbl_telemetria.configure(text="SYS.CPU: --% | MODEL: HYBRID_ONNX | UPLINK: ABORTADO")
        return

    severity = (mildiu_pixels / (leaf_pixels + mildiu_pixels)) * 100.0
```

Nota: *Autoría propia*

Si la cantidad de píxeles verdes detectados es menor a 5.000, el sistema determina que no hay una hoja presente en la imagen y detiene el análisis mostrando el mensaje "No hay hoja" en la interfaz gráfica. Una vez verificada la presencia de la hoja, se calcula el porcentaje de severidad de la infección mediante la siguiente expresión:

$$Severidad\ (\%) = \frac{P_{mildiu}}{P_{hoja} + P_{mildiu}}\ x\ 100 \qquad (1)$$

Donde **P_mildiu** representa los píxeles identificados dentro del rango de color del mildiu y **P_hoja** representa los píxeles correspondientes al área verde sana de la hoja. Por ejemplo, si una hoja presenta 8.000 píxeles verdes y 2.000 píxeles afectados, la severidad calculada sería del 20%, lo que correspondería a un nivel de peligro según la configuración del sistema.

En el segundo método, la imagen es procesada por el modelo YOLOv8n en formato ONNX, el cual verifica la clasificación mediante inteligencia artificial y proporciona una segunda capa de análisis que refuerza el resultado obtenido por el método HSV.

Tercera etapa: Generación de alertas y envío a la nube

Para generar las alertas se definieron rangos de porcentaje que permiten interpretar el nivel de afectación. Cuando el valor es menor al 5%, se considera una presencia inicial de la enfermedad, sin riesgo inmediato para el cultivo. Entre el 5% y el 15%, los síntomas ya son visibles, por lo que se recomienda aplicar medidas preventivas para evitar que avance. Si el porcentaje se encuentra entre el 15% y el 30%, la enfermedad ya afecta una parte importante de la hoja y es necesario aplicar un tratamiento. Cuando el valor supera el 30%, la situación se

vuelve crítica, ya que existe un alto riesgo de propagación, por lo que se requiere una intervención inmediata.

En la Tabla 17 se presentan los niveles de severidad configurados en el sistema y las recomendaciones específicas para el control del mildiu en el cultivo de mora de Castilla.

Tabla 17

Niveles de severidad y recomendaciones del sistema

Librería	Rango de severidad	Función
Preventivo	0% – 5%	Monitoreo preventivo semanal. Sin acción química requerida.
Alerta	5% – 15%	Aplicar fungicida preventivo.
Peligro	15% – 30%	Aplicar fungicida sistémico curativo Metalaxil o Mancozeb. Podar y retirar hojas afectadas.
Crítico	Mayor a 30%	Fumigación inmediata con Fosetil-Aluminio o Dimethomorph. Aislar sección afectada del cultivo

Los resultados del análisis se guardan en Firebase Realtime Database, incluyendo la severidad, el estado de la hoja y la fecha del análisis. En la Figura 37 se muestra el historial de detecciones en la aplicación web.

Figura 37

Historial de detecciones en la aplicación web NexusLeaf AI



Trazabilidad de Detección (Historial Edge)

Timestamp	Severidad (%)	Status (Diagnóstico)	Recomendación	Evidencia Óptica
30/5/2026, 8:17:47 p. m.	0.5%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	Ver Detalles
29/5/2026, 9:18:09 p. m.	0.2%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	Ver Detalles
28/5/2026, 8:12:54 p. m.	0.3%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	Ver Detalles
27/5/2026, 7:58:23 p. m.	0.4%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	Ver Detalles
26/5/2026, 7:13:19 p. m.	0.4%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	Ver Detalles
25/5/2026, 7:47:26 p. m.	0.4%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	Ver Detalles
24/5/2026, 6:35:46 p. m.	0.3%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	Ver Detalles
23/5/2026, 5:50:10 p. m.	1.3%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	Ver Detalles

Nota: *Autoría propia*

En la Figura 38 se presentan dos ejemplos de evidencia óptica almacenada por el sistema, correspondientes a hojas con distintos niveles de severidad: una con una afectación menor al 5% y otra con una severidad crítica que supera el 30% de afectación.

Figura 38

Evidencia óptica de hojas con infección activa



Nota: *Autoría propia*

3.5.2.7 Configuración de Firebase Realtime Database.

Para el procesamiento y configuración, Firebase fue utilizado como plataforma principal de comunicación y almacenamiento en la nube del sistema. Toda la comunicación entre la Raspberry Pi y la aplicación web se realizó a través de FirebaseManager encargada de gestionar la conexión, la descarga de modelos y el envío de resultados.

En la Tabla 18 se presentan las librerías utilizadas para la integración con Firebase Realtime Database.

Tabla 18

Librerías utilizadas en Realgtime Database

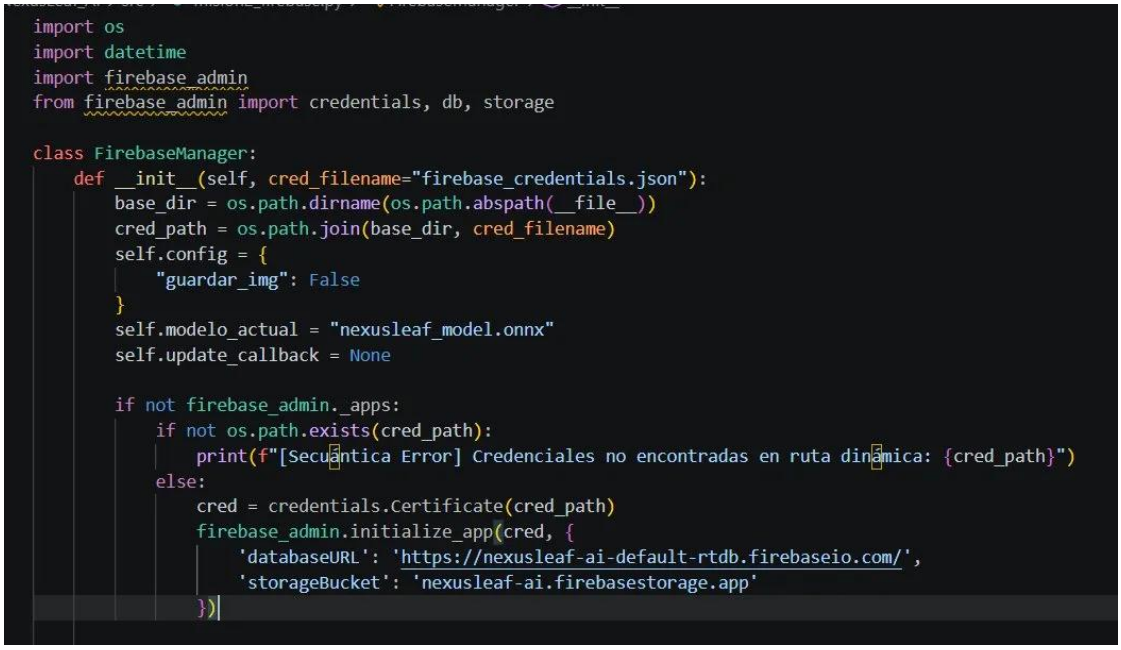
Librería	Función
os	Manejo de rutas y carpetas del sistema de archivos.
datetime	Generación de marcas de tiempo para cada registro de análisis.
firebase_admin	Conexión y autenticación con Firebase desde Python.

credentials	Carga del archivo de credenciales del proyecto Firebase.
db	Acceso y escritura en Firebase Realtime Database
storage	Descarga de archivos desde Firebase Storage

En la Figura 39 se presenta el código de inicialización de la clase FirebaseManager, donde se puede observar cómo el sistema establece la conexión con Firebase utilizando las credenciales del proyecto, configurando tanto la URL de la base de datos en tiempo real como el bucket de almacenamiento. Adicionalmente, el sistema se suscribe automáticamente a los nodos configuracion y modelo_config para escuchar cambios en tiempo real desde la aplicación web.

Figura 39

Código de inicialización y conexión con Firebase



```python
import os
import datetime
import firebase_admin
from firebase_admin import credentials, db, storage

class FirebaseManager:
    def __init__(self, cred_filename="firebase_credentials.json"):
        base_dir = os.path.dirname(os.path.abspath(__file__))
        cred_path = os.path.join(base_dir, cred_filename)
        self.config = {
            "guardar_img": False
        }
        self.modelo_actual = "nexusleaf_model.onnx"
        self.update_callback = None

        if not firebase_admin._apps:
            if not os.path.exists(cred_path):
                print(f"[Secuántica Error] Credenciales no encontradas en ruta dinámica: {cred_path}")
            else:
                cred = credentials.Certificate(cred_path)
                firebase_admin.initialize_app(cred, {
                    'databaseURL': 'https://nexusleaf-ai-default-rtdb.firebaseio.com/',
                    'storageBucket': 'nexusleaf-ai.firebasestorage.app'
                })
```

Nota: *Autoría propia*

En la Figura 40 se presenta el código correspondiente a la función download_model, la cual se ejecuta automáticamente cuando la Raspberry Pi detecta que existe un nuevo modelo

disponible en Firebase Storage. Esta función descarga el archivo ONNX desde la nube y lo almacena en la carpeta models/ del dispositivo, creando la carpeta si no existe. Si la descarga falla por algún motivo, el sistema registra el error y continúa funcionando con el modelo anterior sin interrumpir el servicio.

Figura 40

Código de descarga automática del modelo ONNX desde Firebase Storage

```
def download_model(self, onnx_filename):
    try:
        bucket = storage.bucket()
        blob = bucket.blob(f"models/{onnx_filename}")

        # Guardamos en la carpeta models/ (que está un nivel arriba de src/)
        base_dir = os.path.dirname(os.path.abspath(__file__))
        models_dir = os.path.join(base_dir, "..", "models")
        os.makedirs(models_dir, exist_ok=True)
        local_path = os.path.join(models_dir, onnx_filename)

        blob.download_to_filename(local_path)
        print(f"[Secuántica Edge] Cerebro {onnx_filename} extraído con éxito y guardado en {local_path}.")
        return True
    except Exception as e:
        print(f"[Secuántica Error] Error descargando modelo de Storage: {e}")
        return False
```

Nota: *Autoría propia*

En la Figura 41 se presenta la función log_analysis, la cual es llamada cada vez que la Raspberry Pi realiza un análisis de una hoja. Esta función construye un registro con la severidad detectada, el estado de la hoja y la marca de tiempo del análisis, y lo envía simultáneamente a dos nodos de Firebase: el nodo historial que acumula todos los registros anteriores, y el nodo lectura_actual que mantiene siempre el resultado más reciente para su visualización inmediata en la aplicación web. Si el usuario activó la opción de guardar imagen, la función también incluye la fotografía de la hoja capturada en formato base64 dentro del registro.

Figura 41

Código de envío de resultados a Firebase Realtime Database

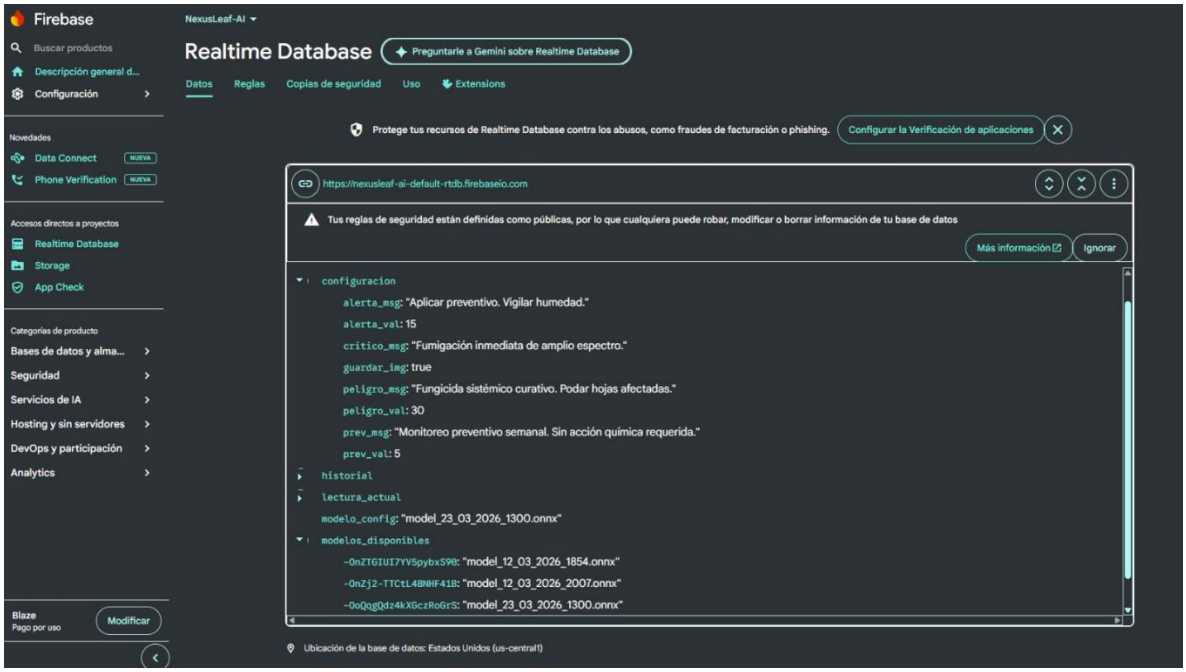
```python
def log_analysis(self, severidad, status="Sano", imagen_b64=None):
    if not firebase_admin._apps:
        return False

    try:
        timestamp_iso = datetime.datetime.now(datetime.timezone.utc).isoformat()

        payload = {
            "timestamp": timestamp_iso,
            "severidad": float(severidad),
            "status": status
        }

        # Solo incluir el blob pesado si la política lo exige y la UI lo proporcionó
        if self.config.get("guardar_img") and imagen_b64:
            payload["imagen_base64"] = imagen_b64
            print("[Secuántica Edge] Evidencia Raw empacada en payload.")

        historial_ref = db.reference('historial')
        nuevo_registro = historial_ref.push()
        nuevo_registro.set(payload)
        key = nuevo_registro.key

        current_ref = db.reference('lectura_actual')
        current_ref.set(payload)

        print(f"[UPLINK] Registro enviado al nodo historial: {key}")
        print(f"[Secuántica Edge] Nodo 'historial' actualizado: {severidad}% / {status}")
        return True
    except Exception as e:
        print(f"[Secuántica Error] Falla en pasarela de datos: {e}")
        return False
```

Nota: *Autoría propia*

La base de datos fue estructurada en cinco nodos principales que organizan toda la información del sistema, tal como se muestra en la Figura 42.

Figura 42

Estructura de Firebase Realtime Database del sistema



Nota: *Autoría propia*

En la Tabla 19 se presenta un resumen de los nodos configurados en Firebase Realtime Database y su función dentro del sistema.

Tabla 19

Estructura de nodos en Firebase Realtime Database

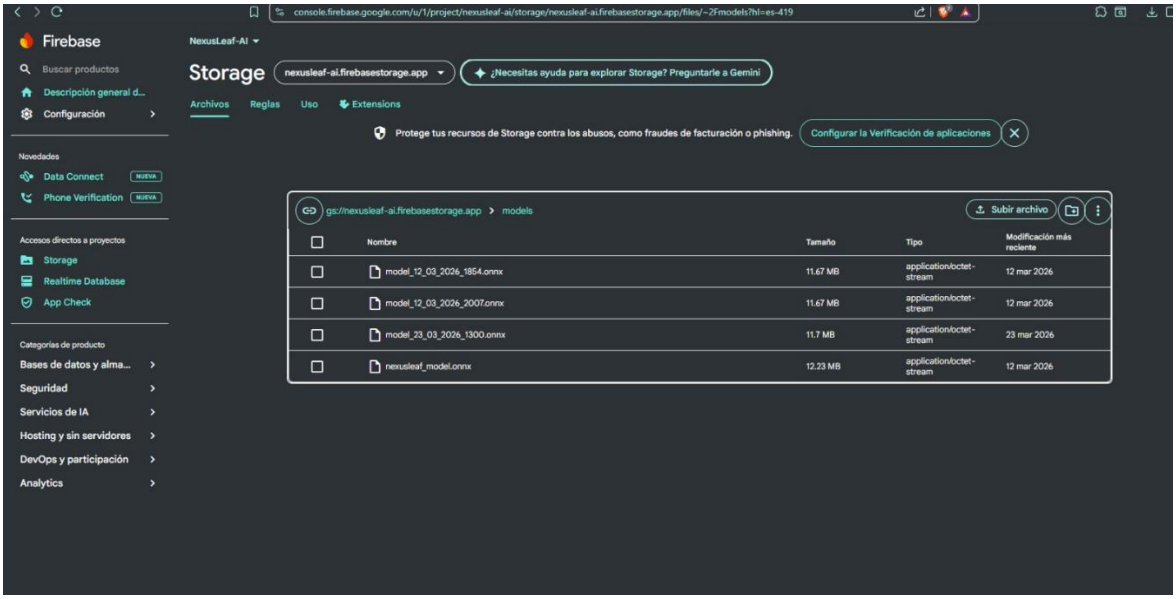
Librería	Función
configuracion	Almacena umbrales de severidad y recomendaciones configurables.
historial	Registra cada análisis realizado con severidad, estado y timestamp.
lectura_actual	Mantiene el resultado del último análisis para visualización inmediata.
modelo_config	Indica el modelo ONNX activo para descarga en la Raspberry Pi.
modelos_disponibles	Registro histórico de todos los modelos generados.

señal_entrenamiento	Controla el inicio del pipeline desde la aplicación web.

Adicionalmente, Firebase Storage fue utilizado para almacenar los archivos de modelos ONNX generados durante el entrenamiento. La carpeta models/ contiene los modelos generados en diferentes fechas de entrenamiento, cada uno identificado con su fecha y hora de generación, tal como se muestra en la Figura 43.

Figura 43

Modelos ONNX almacenados en Firebase Storage



Nota: *Autoría propia*

3.5.2.8 Implementación del sistema IoT en el entorno de cultivo

Para la implementación física del sistema, todos los componentes fueron integrados en una carcasa protectora de tipo maleta, la cual alberga la Raspberry Pi 4, la pantalla, el ventilador de refrigeración y el cableado necesario para el funcionamiento del sistema. La cámara OV5647 se conecta a la Raspberry Pi 4 mediante el conector CSI y se instala de forma externa a la carcasa, orientada directamente hacia el follaje de las plantas. La alimentación del sistema se realiza mediante una fuente de 5V conectada a la red eléctrica disponible en el entorno del cultivo. La cámara OV5647 fue posicionada a una altura de 45 cm sobre las plantas de mora

de Castilla, las cuales se encontraban en etapa post-siembra con una altura aproximada de entre 45 y 50 cm. Esta etapa fue seleccionada para las pruebas debido a que las hojas jóvenes en este período son las más susceptibles a presentar los primeros síntomas del mildiu polvoso, y su tamaño y disposición permiten una captura de imagen adecuada para el correcto funcionamiento del modelo de detección. La ubicación de la cámara se ajusta progresivamente conforme las plantas crecen, de modo que se mantenga siempre el ángulo y la distancia óptimos para la detección, distancia que permite capturar con precisión las características visuales del follaje sin obstrucciones, garantizando una correcta segmentación e inferencia por parte del modelo YOLOv8n. La ubicación de la cámara se ajusta progresivamente conforme las plantas crecen, de modo que se mantenga siempre el ángulo y la distancia óptimos para la detección.

La transmisión de datos entre el dispositivo y la plataforma en la nube se realiza mediante conexión inalámbrica Wi-Fi, con un alcance de aproximadamente 5 metros entre el punto de acceso y la unidad de procesamiento. Esta conectividad permite que los resultados del análisis, incluyendo el porcentaje de severidad, el estado de diagnóstico y la recomendación generada, sean enviados en tiempo real a Firebase y visualizados de forma inmediata en la aplicación web desde cualquier dispositivo con acceso a internet.

En la figura 44 se presenta el diagrama de implementación del sistema, donde se identifican los componentes principales: la cámara OV5647, el conector CSI, la Raspberry Pi 4, la pantalla, el ventilador, la fuente de alimentación de 5V y la conexión inalámbrica Wi-Fi, así como la disposición del sistema frente a las plantas de mora de Castilla en el entorno real de cultivo.

Figura *44* *Implementación del sistema IoT en el entorno de cultivo*



Nota: *Autoría propia*

3.6 Desarrollo de la aplicación

En esta sección se describe el proceso de diseño y desarrollo de la aplicación, incluyendo las herramientas empleadas durante su implementación. Asimismo, se explica la arquitectura del sistema, la organización del código y los componentes principales desarrollados, con el propósito de asegurar una aplicación eficiente, escalable y fácil de mantener.

Para complementar el sistema de monitoreo, se desarrolló una aplicación web accesible desde cualquier dispositivo con conexión a internet, que permite al usuario visualizar en tiempo real los resultados del análisis realizado por la Raspberry Pi, configurar los parámetros del sistema y gestionar los modelos de detección. La aplicación fue desarrollada utilizando HTML, CSS y JavaScript, y fue desplegada en Firebase Hosting bajo el dominio nexusleaf-ai.web.app. En la Tabla 20 se presentan las tecnologías utilizadas para su desarrollo.

Tabla 20

Tecnologías utilizadas en el desarrollo de la aplicación web

Librería	Función
HTML / CSS / JavaScript	Desarrollo de la interfaz de usuario
Firebase Realtime Database	Sincronización de datos en tiempo real
Firebase Storage	Almacenamiento de modelos ONNX
Firebase Hosting	Despliegue y publicación de la aplicación web
Service Worker	Notificaciones push y funcionamiento como PWA.

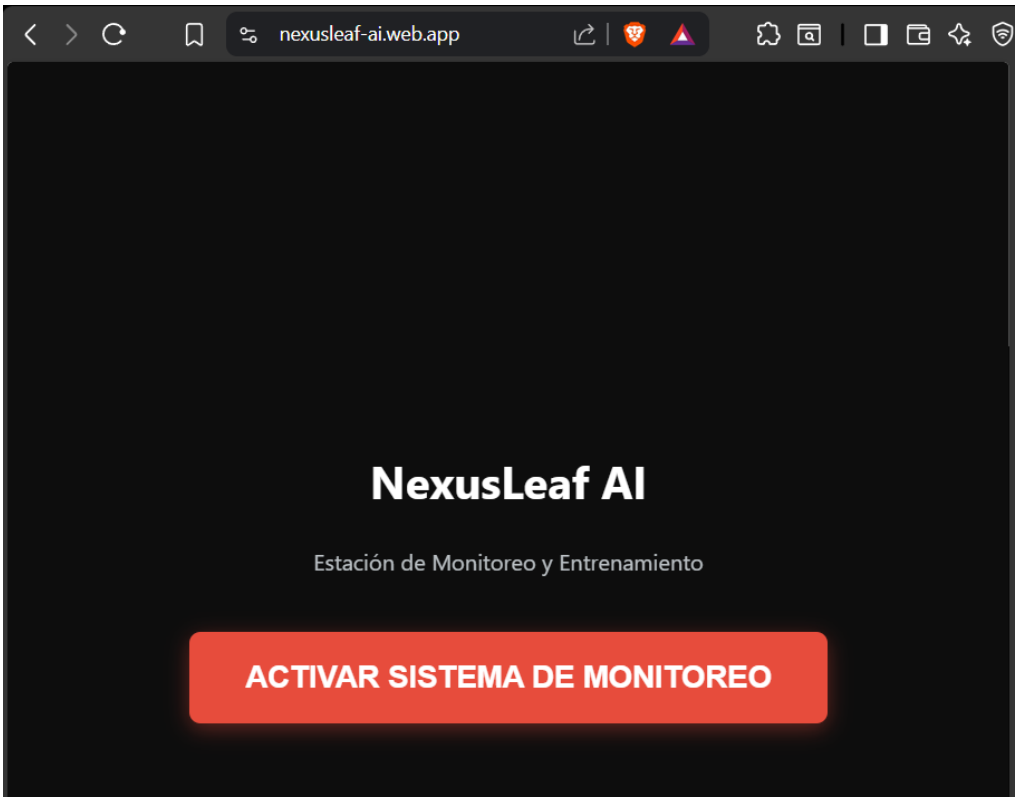
3.6.1 Pantalla de inicio

Al acceder a la aplicación desde cualquier navegador web mediante el dominio nexusleaf-ai.web.app, el usuario encuentra una pantalla de inicio que muestra el nombre del

sistema y un botón para activar el monitoreo, tal como se presenta en la Figura 45. Al presionar el botón "ACTIVAR SISTEMA DE MONITOREO", la aplicación solicita permisos de notificación al navegador, establece la conexión con Firebase Realtime Database y habilita la escucha en tiempo real de los datos generados por la Raspberry Pi.

Figura 45

Pantalla de inicio de la aplicación web



Nota: *Autoría propia*

Una vez activado el sistema, el usuario accede al Centro de Mando presentado en la Figura 46, el cual está compuesto por tres módulos principales: el Panel de Configuración Experta, el Gestor de Entrenamiento y el Selector de Modelos Activos.

El Panel de Configuración Experta: permite al usuario ajustar los umbrales de severidad y las recomendaciones de tratamiento para cada nivel de alerta directamente desde la interfaz web. Los valores configurados son enviados automáticamente a la Raspberry Pi mediante Firebase al presionar el botón "GUARDAR Y APLICAR AL EDGE", aplicándose en

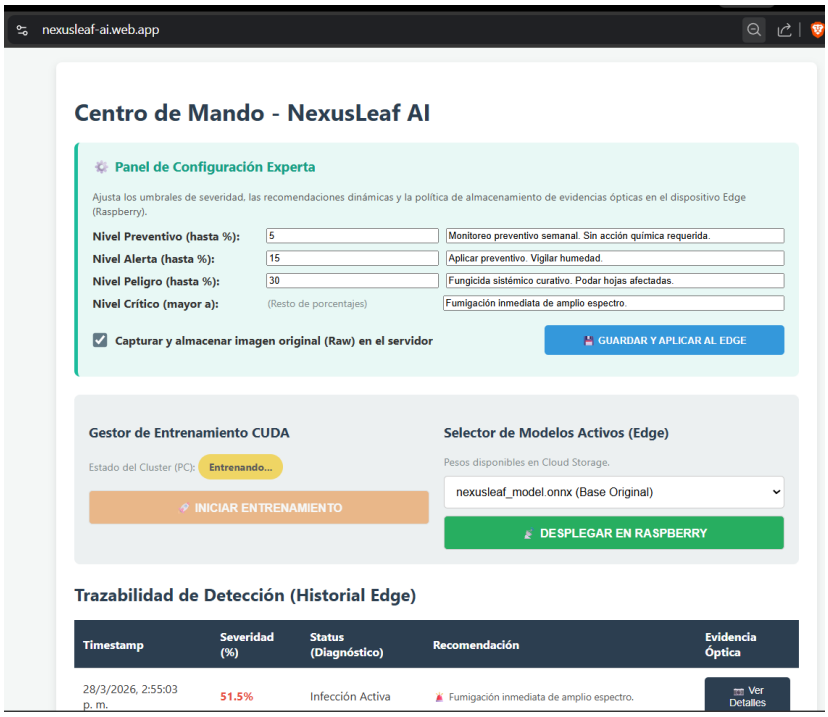
tiempo real sin necesidad de reiniciar el dispositivo. Adicionalmente, el panel incluye una opción para activar el almacenamiento de la imagen capturada junto con cada registro de análisis.

El Gestor de Entrenamiento: muestra el estado actual del proceso de entrenamiento en el PC, indicando si el sistema se encuentra en modo Standby o en proceso de Entrenamiento. Al presionar el botón "INICIAR ENTRENAMIENTO" se envía una señal a Firebase que activa automáticamente el pipeline de entrenamiento en el PC sin necesidad de intervención directa sobre el equipo.

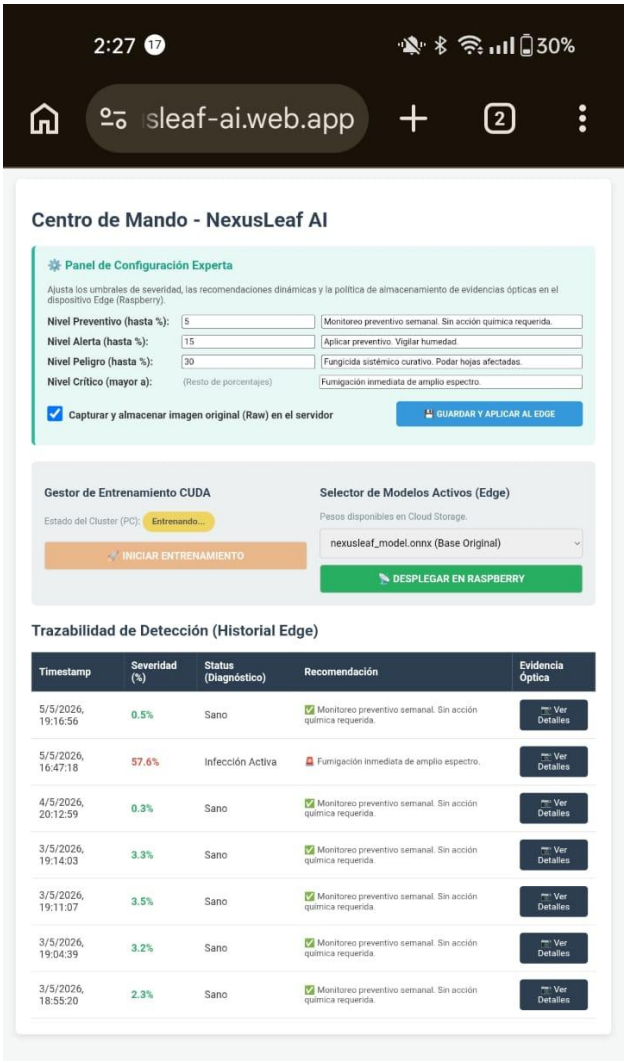
El Selector de Modelos Activos: permite al usuario elegir entre los modelos ONNX disponibles en Firebase Storage y desplegarlo en la Raspberry Pi con un solo clic mediante el botón "DESPLEGAR EN RASPBERRY", garantizando que el dispositivo siempre cuente con el modelo más adecuado para las condiciones del cultivo.

Figura 46

Panel de configuración y gestión de la aplicación web



(A) Pantalla del monitor



(B) Teléfono

Nota: *Autoría propia*

3.6.2 Historial de detecciones

La sección de Trazabilidad de Detección presentada en la figura 47 muestra el historial completo de análisis realizados por la Raspberry Pi, ordenados del más reciente al más antiguo. Cada registro incluye la fecha y hora del análisis, el porcentaje de severidad detectado mostrado en color rojo cuando supera el umbral preventivo, el estado de diagnóstico de la hoja, la recomendación de tratamiento generada automáticamente según el nivel de severidad y un botón para visualizar la evidencia óptica almacenada correspondiente a ese análisis.

La aplicación genera alertas automáticas mediante notificaciones del navegador y señales sonoras cuando detecta una infección activa en los últimos dos minutos, permitiendo

al agricultor tomar acciones de forma oportuna sin necesidad de estar revisando la pantalla constantemente.

Figura 47

Historial de detecciones en la aplicación web



Trazabilidad de Detección (Historial Edge)

Timestamp	Severidad (%)	Status (Diagnóstico)	Recomendación	Evidencia Óptica
30/5/2026, 8:17:47 p. m.	0.5%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	📷 Ver Detalles
29/5/2026, 9:18:09 p. m.	0.2%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	📷 Ver Detalles
28/5/2026, 8:12:54 p. m.	0.3%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	📷 Ver Detalles
27/5/2026, 7:58:23 p. m.	0.4%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	📷 Ver Detalles
26/5/2026, 7:13:19 p. m.	0.4%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	📷 Ver Detalles
25/5/2026, 7:47:26 p. m.	0.4%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	📷 Ver Detalles
24/5/2026, 6:35:46 p. m.	0.3%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	📷 Ver Detalles
23/5/2026, 5:50:10 p. m.	1.3%	Sano	✅ Monitoreo Preventivo semanal. Sin acción química requerida	📷 Ver Detalles

Nota: *Autoría propia*

3.6.2.1 Lógica de la aplicación web

La lógica de la aplicación web fue desarrollada en el archivo app.js, el cual gestiona la comunicación con Firebase, el procesamiento de los datos recibidos y la generación de alertas hacia el usuario.

En la Figura 48 se presenta el código correspondiente a la función de configuración, donde se puede observar cómo la aplicación recopila los valores ingresados por el usuario en el Panel de Configuración Experta y los envía directamente a Firebase Realtime Database. Una vez guardados, estos parámetros son recibidos automáticamente por la Raspberry Pi y aplicados en tiempo real sin necesidad de reiniciar el sistema.

Figura 48

Código de configuración y envío de parámetros a Firebase

```
if (btnSaveConfig) {
    btnSaveConfig.addEventListener('click', () => {
        const nuevaConfig = {
            prev_val: parseFloat(cfgPrevVal.value),
            prev_msg: cfgPrevMsg.value,
            alerta_val: parseFloat(cfgAlertaVal.value),
            alerta_msg: cfgAlertaMsg.value,
            peligro_val: parseFloat(cfgPeligroVal.value),
            peligro_msg: cfgPeligroMsg.value,
            critico_msg: cfgCriticoMsg.value,
            guardar_img: cfgGuardarImg.checked
        };

        // Subir la configuración a Firebase para que la RPi la escuche
        if (window.dbRef && window.db) {
            window.set(window.dbRef(window.db, 'configuracion'), nuevaConfig);
            btnSaveConfig.textContent = "✅ CONFIGURACIÓN ENVIADA AL EDGE";
            btnSaveConfig.style.backgroundColor = "#27ae60";
            setTimeout(() => {
                btnSaveConfig.textContent = "💾 GUARDAR Y APLICAR AL EDGE";
                btnSaveConfig.style.backgroundColor = "#3498db";
            }, 3000);
        }
    });
}
```

Nota: *Autoría propia*

En la Figura 49 se presentan las dos funciones principales que gestionan la lógica de recomendaciones y alertas del sistema. La función getRecomendacion evalúa el porcentaje de severidad recibido y lo compara con los umbrales configurados por el usuario, retornando automáticamente el mensaje de recomendación correspondiente al nivel de afectación detectado. Si la severidad es menor al valor preventivo devuelve el mensaje preventivo, si

supera el valor de alerta devuelve el mensaje de alerta, si supera el valor de peligro devuelve el mensaje de peligro, y si supera todos los umbrales devuelve el mensaje crítico.

La función dispararAlerta se ejecuta automáticamente cuando el sistema detecta una infección activa reciente. Esta función genera una notificación push en el navegador del usuario con el título "Peligro Biológico: Infección al X%" y la recomendación correspondiente, acompañada de una señal de vibración para dispositivos móviles que permite alertar al agricultor incluso cuando no está mirando la pantalla.

Figura 49

Código de generación de recomendaciones y alertas automáticas

```
function getRecomendacion(severidad) {
    if (severidad < configActual.prev_val) return "✅ " + configActual.prev_msg;
    if (severidad < configActual.alerta_val) return "⚠️ " + configActual.alerta_msg;
    if (severidad < configActual.peligro_val) return "❗ " + configActual.peligro_msg;
    return "🚨 " + configActual.critico_msg;
}

function dispararAlerta(item) {
    const severidadStr = item.severidad.toFixed(1);
    const title = `🚨 Peligro Biológico: Infección al ${severidadStr}%`;
    const body = getRecomendacion(item.severidad);

    // Notificación de SO / PWA
    if ('serviceWorker' in navigator && Notification.permission === 'granted') {
        navigator.serviceWorker.ready.then(function(registration) {
            registration.showNotification(title, {
                body: body,
                icon: '/icon-192x192.png',
                vibrate: [200, 100, 200, 100, 200, 100, 200]
            });
        });
    }
```

Nota: *Autoría propia*

4 CAPITULO IV PRUEBAS

Este capítulo detalla la etapa final de nuestro proyecto bajo la metodología en cascada: la fase de pruebas. En esta etapa se evaluó el desempeño del sistema IoT implementado para la detección de mildiu en hojas de mora de Castilla, verificando su capacidad para identificar signos visibles de la enfermedad en condiciones reales de campo. Las pruebas fueron realizadas en la comunidad de San Francisco de Guachinguero, ubicada en la parroquia San José de Quichinche, cantón Otavalo, donde se encuentra el cultivo objeto de estudio. Durante esta etapa se evaluó el funcionamiento integral del sistema, abarcando la detección de enfermedades mediante la Raspberry Pi 4, la comunicación en tiempo real con Firebase Realtime Database y la generación de alertas a través de la aplicación web.

4.1 Evaluación del modelo de red neuronal

En esta sección se presentan los parámetros utilizados durante el proceso de entrenamiento del modelo YOLOv8n, los cuales fueron analizados en función de los distintos resultados obtenidos, con el objetivo de seleccionar la configuración más adecuada para la implementación del sistema de detección de mildiu en hojas de mora de Castilla. Adicionalmente, se describen las pruebas de detección realizadas mediante visión por computador con OpenCV.

El entrenamiento se llevó a cabo utilizando el modelo YOLOv8n como arquitectura base, ejecutado a través del entorno de Python con la biblioteca Ultralytics, empleando los parámetros definidos en el archivo de configuración. La tabla 21 presenta las características del entorno computacional utilizado.

Tabla 21

Características del hardware utilizado durante el entrenamiento del modelo

Componente	Características
Modelo del Equipo	Laptop AMD Ryzen 5 7520U
Procesador	AMD Ryzen 5 7520U with Radeon Graphics (2,80 GHz)
Memoria RAM	16 GB
Sistema Operativo	Windows 11
Unidad de almacenamiento	500GB
Arquitectura del Sistema	64 bits

También se presentan las especificaciones técnicas de la Raspberry Pi 4 utilizada como dispositivo de inferencia en campo, sobre el cual se ejecuta el modelo en tiempo real una vez finalizado el entrenamiento.

Tabla 22

Especificaciones técnicas de la Raspberry Pi 4

Componente	Características
Modelo del Equipo	Raspberry Pi 4 Model B
Procesador	ARM Cortex-A72, 4 núcleos a 1.8 GHz
Memoria RAM	8 GB
Almacenamiento	MicroSD 128 GB
Sistema Operativo	Raspberry Pi OS (64 bits)
Cámara	OV5647 — 5 megapíxeles, interfaz CSI
Conectividad	Wi-Fi 802.11
Motor de Interferencia	onnxruntime
Formato Modelo	ONNX — YOLOv8n

4.2 Configuración y parámetros de entrenamiento del modelo.

Para el desarrollo del sistema de detección se utilizó la arquitectura YOLOv8n de la librería Ultralytics, en su modalidad de detección de objetos con dos clases: hoja sana y hoja enferma. Esta arquitectura fue seleccionada por su equilibrio entre precisión y velocidad de inferencia, siendo adecuada para su ejecución posterior en dispositivos de recursos limitados como la Raspberry Pi 4.

Se compararon tres configuraciones del modelo variando únicamente el número de épocas máximas configuradas: 30, 50, 75 y 100 épocas, manteniendo constantes los demás parámetros. Los experimentos de 75 y 100 épocas, pese a configurar un mayor número de iteraciones, activaron el early stopping antes de completarlas y obtuvieron métricas ligeramente inferiores, reforzando la conclusión de que el modelo alcanza su punto óptimo de convergencia alrededor de las 50 épocas, sin beneficiarse de un entrenamiento más prolongado. En la Tabla 23 se presentan los parámetros de la red neuronal.

Tabla 23

Parámetros de la red neuronal

CONFIGURACIÓN DE PARÁMETROS				
YOLOv8n (*You Only Look Once versión 8, variante Nano*)				
Parámetros	**30 épocas**	**50 épocas**	**75 épocas**	**100 épocas**
Épocas configuradas	30	50	75	100
Épocas reales	30	50	55	67
Tamaño de imagen	640*640px	640*640px	640*640px	640*640px
Tamaño lote	16	16	16	16
Optimizador	AdamW	AdamW	AdamW	AdamW
Tasa de aprendizaje	0.979	0.987	0.976	0.977

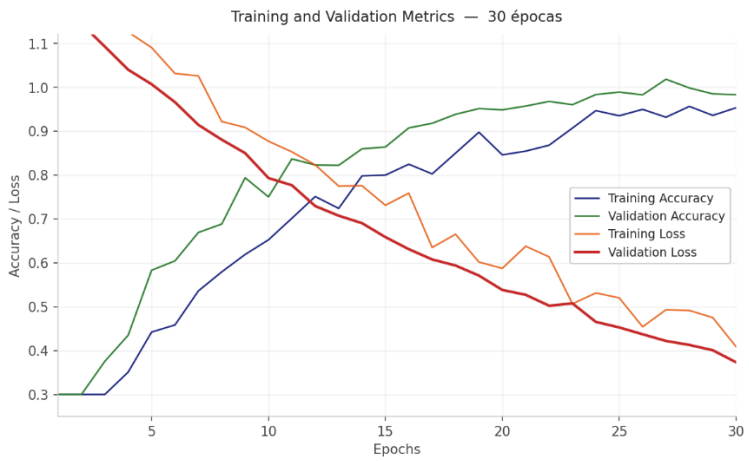
En esta sección se comparan tres configuraciones del modelo de detección, entrenadas con diferentes números máximos de épocas (30, 50, 75 y 100), manteniendo constantes los principales hiperparámetros. En todos los casos se utilizó la arquitectura YOLOv8n de la

librería Ultralytics en su versión de detección de objetos, con dos clases definidas: hoja sana y hoja enferma. Esta arquitectura fue seleccionada por su equilibrio entre precisión y velocidad de inferencia, lo que la hace adecuada para su implementación en dispositivos de recursos limitados, como la Raspberry Pi 4. En las siguientes figuras se presentan los gráficos correspondientes al proceso de entrenamiento del modelo con 30, 50, 75 y 100 épocas, donde se representan las métricas de precisión y pérdida tanto para el conjunto de entrenamiento como para el de validación.

En primer caso en la figura 50, se observa que las curvas de precisión presentan mayor variabilidad y oscilación a lo largo del entrenamiento, lo que indica que el modelo no alcanzó una convergencia completamente estable dentro de ese número de iteraciones. Sin embargo, los resultados finales muestran un desempeño aceptable con una precisión general de 0.979.

Figura 50

Gráfico de curvas de entrenamiento y perdida en 30 Épocas



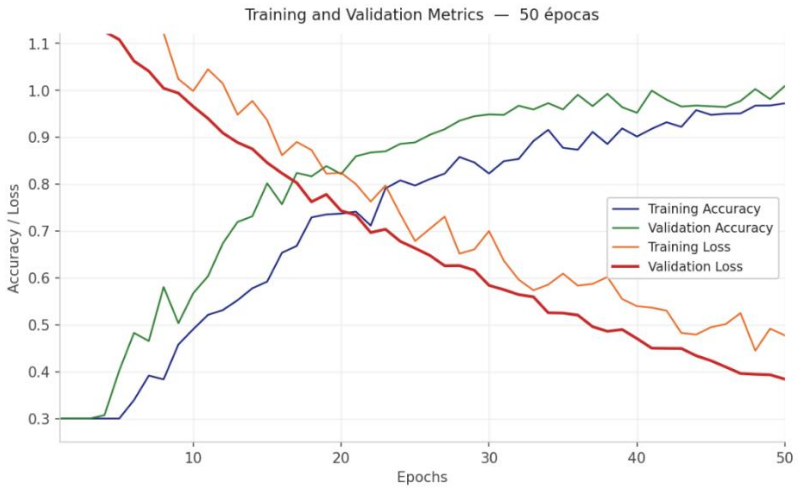
Nota: *Autoría propia*

En el segundo caso en la figura 51, con 50 épocas, se observa una convergencia más estable y progresiva en todas las métricas. Las curvas de pérdida tanto de entrenamiento como de validación disminuyeron de forma sostenida, mientras que la precisión y el recall alcanzaron valores cercanos a 1.0 y se mantuvieron estables en las épocas finales. Este comportamiento

indica que el modelo logró un aprendizaje adecuado sin presentar sobreajuste, obteniendo la mejor precisión general de 0.987 entre los tres experimentos.

Figura 51

Gráfico de curvas de entrenamiento y perdida en 50 Épocas

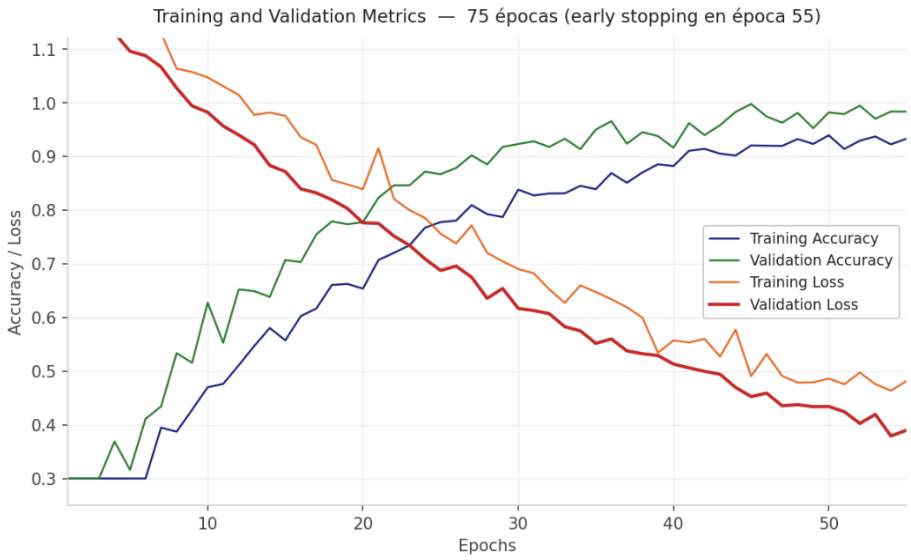


Nota: *Autoría propia*

En el tercer caso en la figura 52, configurado con 75 épocas, donde el mecanismo de parada anticipada intervino en la época 55 al detectar que la pérdida de validación no presentaba mejoras sostenidas. Esto indica que el modelo logró converger bien antes de alcanzar el límite establecido. A lo largo del entrenamiento, tanto la precisión de entrenamiento como la de validación mostraron una tendencia ascendente progresiva, estabilizándose cerca de 0.93 y 0.98 respectivamente, mientras que ambas curvas de pérdida descendieron de manera continua desde valores superiores a 1.1 hasta aproximadamente 0.39 al final del proceso. Un aspecto destacable es que la pérdida de validación se mantuvo por debajo de la pérdida de entrenamiento durante casi todo el recorrido. Todo esto refuerza la idea de que configurar un número elevado de épocas no necesariamente se traduce en mejores resultados; cuando el modelo ya encontró su punto óptimo, continuar entrenando aporta poco y el early stopping se convierte en una herramienta clave para detener el proceso en el momento adecuado.

Figura 52

Gráfico de curvas de entrenamiento y perdida en 75 Épocas

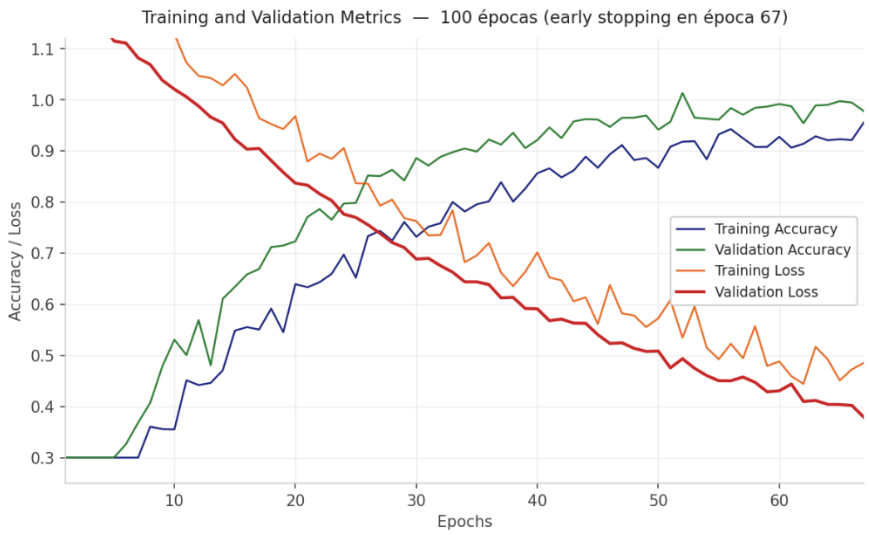


Nota: *Autoría propia*

En el cuarto caso en la figura 53, configurado con 100 épocas, el mecanismo de parada anticipada detuvo el entrenamiento en la época 67 al no registrar mejoras en la pérdida de validación durante 20 épocas consecutivas. Las curvas muestran que el modelo alcanzó su punto de convergencia antes de completar las 100 épocas configuradas, obteniendo una precisión general de 0.977, ligeramente inferior a la configuración de 50 épocas, lo que confirma que un mayor número de épocas no garantiza una mejora en el desempeño cuando el modelo ya ha convergido.

Figura 53

Gráfico de curvas de entrenamiento y perdida en 100 Épocas

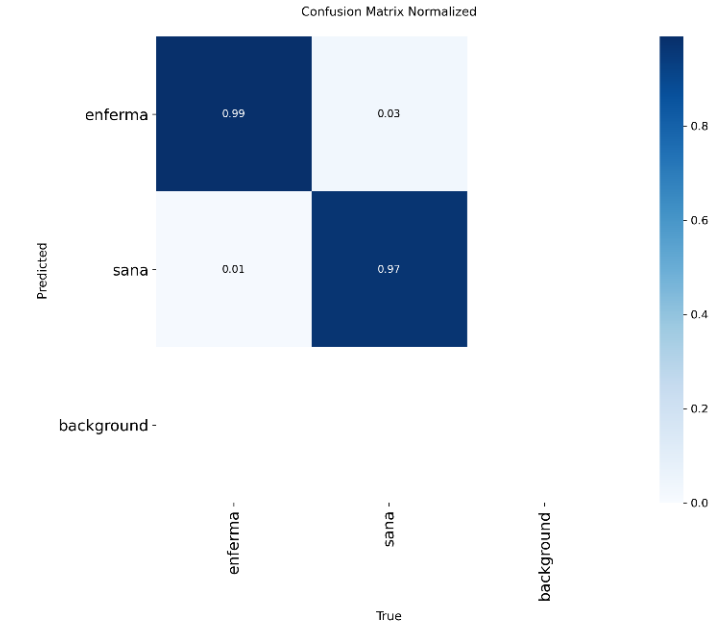


Nota: *Autoría propia*

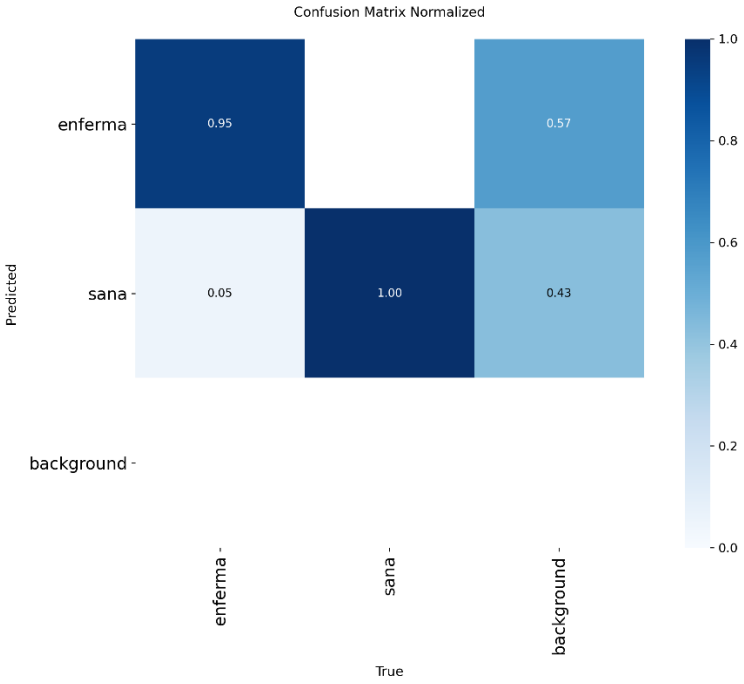
De igual manera se presentan las matrices de confusión obtenidas de los modelos entrenados. En los tres casos, se observa un buen desempeño general. En la figura 54 se presenta en el primer caso (a) experimento de 30 épocas obtuvo una clasificación correcta del 99% para la clase enferma y del 97% para la clase sana, con un margen de error mínimo del 1% y 3% respectivamente. El caso (b) experimento de 50 épocas identificó correctamente el 95% de las hojas enfermas y el 97% de las hojas sanas, mostrando un comportamiento equilibrado entre ambas clases. En el caso (c), la matriz de confusión correspondiente al experimento de 75 épocas, donde el modelo logró una clasificación correcta del 99% para la clase enferma y del 97% para la clase sana, resultados que coinciden con los obtenidos en el experimento de 30 épocas. El caso (d) experimento de 100 épocas obtuvo los mismos resultados que el de 50 épocas en la diagonal principal, con un 95% de acierto en hojas enfermas y un 97% en hojas sanas. Adicionalmente, la clase background fue identificada con una precisión del 100%, confirmando que el modelo alcanzó un punto de estabilidad similar al de 50 épocas sin mejoras adicionales.

Figura 54

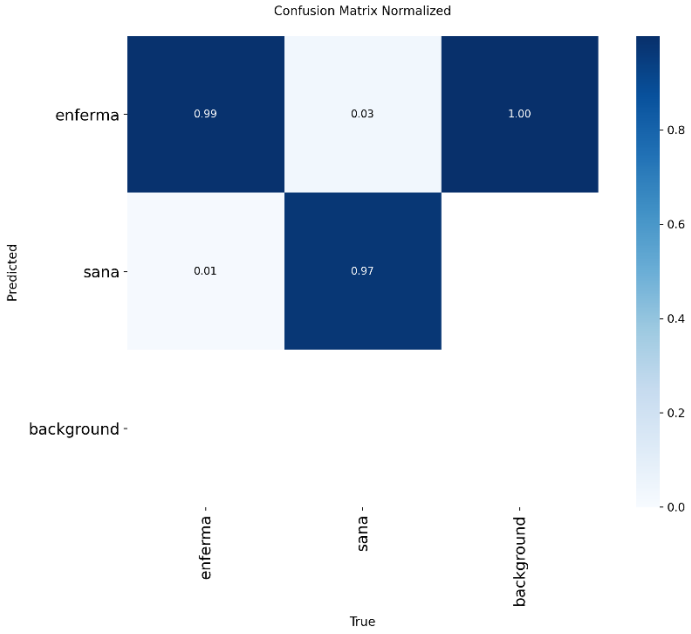
Matrices de confusión de red neuronal.



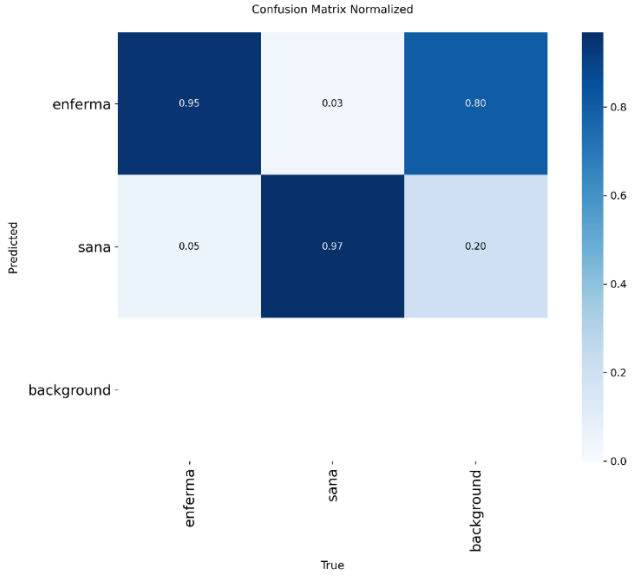
(a) Matriz de confusión 30 épocas



(b) Matriz de confusión 50 épocas



(c) Matriz de confusión de 75 épocas



(d) Matriz de confusión de 100 épocas

Nota: *Autoría propia*

La elección del modelo entrenado de 50 épocas obtuvo la precisión de 0.987 indica que la gran mayoría de las hojas clasificadas por el modelo como enfermas o sanas correspondieron correctamente a su categoría real. Por su parte, el recall de 0.986 refleja la capacidad del modelo para detectar casi la totalidad de las hojas enfermas presentes en el conjunto de datos, minimizando los casos en los que una hoja afectada no es identificada. En otras palabras, el

sistema no solo realiza predicciones confiables, sino que además evita dejar pasar hojas enfermas sin detectar, aspecto especialmente importante para una aplicación de monitoreo fitosanitario. Adicionalmente, las curvas de entrenamiento mostraron una convergencia estable. El mecanismo de Early Stopping, configurado no interrumpió el entrenamiento antes de completar las 50 épocas establecidas, lo que indica que el modelo logró un aprendizaje adecuado dentro de ese intervalo y no requirió un mayor número de iteraciones para mejorar su rendimiento.

4.3 Pruebas de detección

Para las pruebas de detección, se realizaron diferentes etapas utilizando tanto hojas sanas como hojas enfermas afectadas por el mildiu polvoso, el cual es el causante del amarillamiento de las hojas. El sistema funciona de manera sistematizada y, al momento de analizar la hoja de la planta, envía una alerta indicando si esta se encuentra sana o enferma, junto con el porcentaje de severidad de la afección.

4.3.1 Prueba con hojas sanas

Para el análisis de las hojas, el sistema realiza un cálculo basado en la segmentación en el espacio de color HSV, una técnica utilizada en visión por computador que permite identificar regiones de una imagen según sus características de color.

En este caso, se emplea la segmentación para detectar las hojas sanas. La zona verde corresponde a una hoja sana, mientras que las variaciones hacia tonos amarillos indican posibles afecciones.

Para ello, se definieron rangos específicos en el espacio HSV, donde los píxeles con valores de tono entre 35 y 85 son identificados como zona sana.

En la Figura 55 se presenta el fragmento de código donde se realiza la asignación de estos rangos para la segmentación.

Figura 55

Segmentación con HSV Hojas sanas

```
# 1. Segmentación HSV - HOJA SANA
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
lower_green = np.array([35, 40, 40])
upper_green = np.array([85, 255, 255])
mask_leaf = cv2.inRange(hsv, lower_green, upper_green)
```

Nota: *Autoría propia*

De la misma manera, en la Figura 56 se muestra las características, donde se puede observar la identificación de una hoja sana.

Figura 56

Características de la Hoja Sana



Nota: *Autoría propia*

La Figura 57 muestra el resultado del análisis automatizado, en el cual se determina que la muestra presenta un nivel de severidad del 0.3%, clasificándose como estado sano. Este valor

evidencia una condición óptima de la hoja, sin presencia significativa de síntomas asociados a enfermedades.

Figura 57

Resultado del análisis automatizado

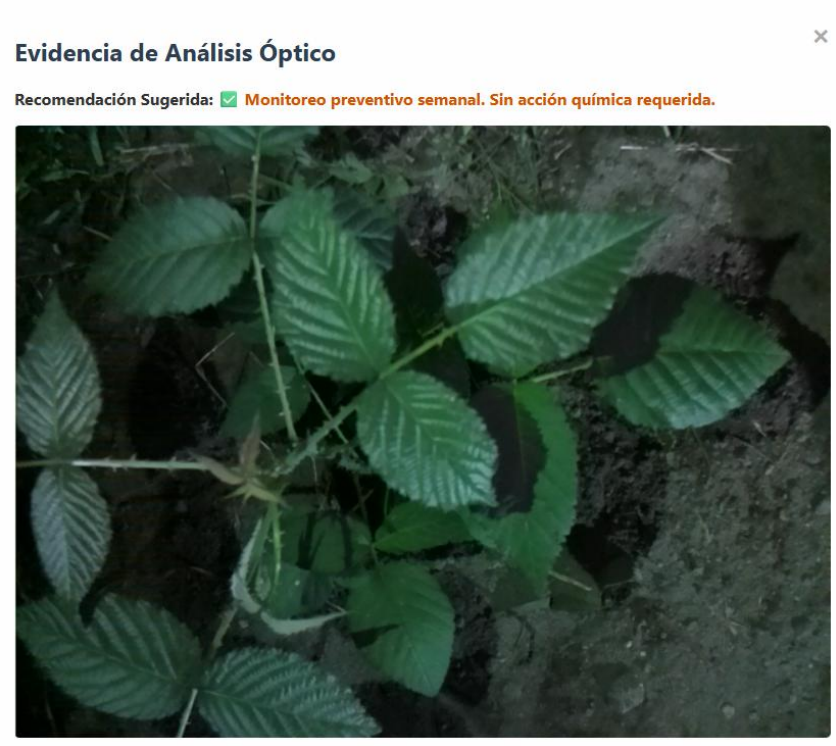
4/5/2026, 8:12:59 p. m.	0.3%	Sano	✅ Monitoreo preventivo semanal. Sin acción química requerida.

Nota: *Autoría propia*

En la Figura 58 se presenta la evidencia visual del análisis óptico, donde se observa la imagen procesada de la planta. La segmentación y evaluación cromática confirman la predominancia de tonalidades verdes, lo cual respalda la clasificación obtenida y valida el correcto funcionamiento del sistema de detección.

Figura 58

Evidencia visual del análisis óptico



Nota: *Autoría propia*

4.3.2 Prueba con hojas enfermas

Para esta prueba se colocó una hoja de mora de Castilla con síntomas visibles de mildiu polvoso frente a la cámara OV5647. El sistema identificó las zonas afectadas mediante la segunda máscara de segmentación HSV, donde los píxeles con bajo nivel de saturación entre 0 y 50 y alto nivel de brillo entre 150 y 255 fueron identificados como zona enferma, características visuales propias del mildiu polvoso que se manifiesta como manchas blancas o grises sobre la superficie de la hoja. En la figura 58 se presenta el fragmento de código correspondiente a esta segmentación.

Figura 59

Segmentación con HSV Hojas enfermas

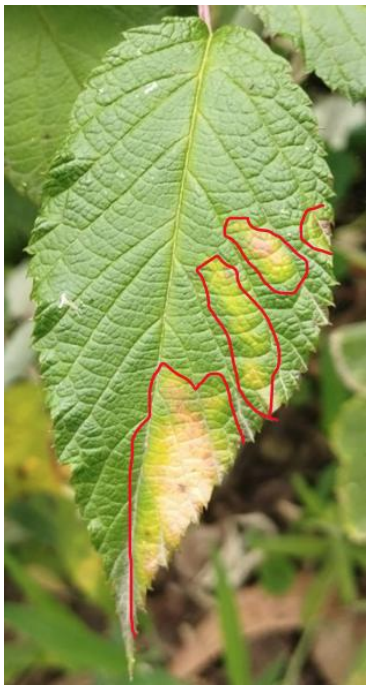
```
lower_mildiu = np.array([0, 0, 150])
upper_mildiu = np.array([180, 50, 255])
mask_mildiu = cv2.inRange(hsv, lower_mildiu, upper_mildiu)

leaf_pixels = cv2.countNonZero(mask_leaf)
mildiu_pixels = cv2.countNonZero(mask_mildiu)
```

Nota: *Autoría propia*

En la figura 60 se muestran las características visuales donde se puede observar la identificación de una hoja con síntomas de mildiu polvoso.

Figura 60

Características de la Hoja Enferma

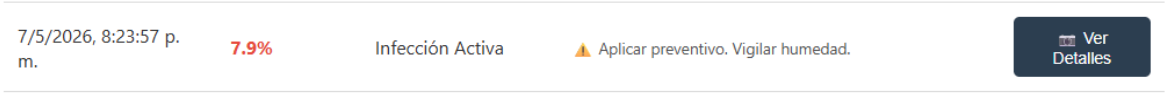


Nota: *Autoría propia*

La figura 61 muestra el resultado del análisis automatizado de la hoja enferma, donde el sistema determinó un nivel de severidad superior al 7.9%, clasificándola como Infección Activa y generando automáticamente la alerta con la recomendación de tratamiento correspondiente según el nivel de afectación detectado.

Figura 61

Resultado del análisis automatizado



Nota: *Autoría propia*

En la figura 62 se presenta la evidencia visual del análisis óptico de la hoja enferma, donde se puede observar la presencia de zonas con tonalidades características del mildiu polvoso, confirmando la clasificación obtenida por el sistema.

Figura 62

Evidencia visual del análisis óptico



Nota: *Autoría propia*

4.3.3 Prueba con diferentes niveles de severidad

Para verificar el comportamiento del sistema ante distintos grados de afectación, se realizaron pruebas adicionales con hojas de mora de Castilla que presentaban diferentes niveles visibles de mildiu polvoso. En cada caso el sistema ejecutó el mismo flujo de análisis aplicando la segmentación HSV y la inferencia con el modelo YOLOv8n. En la figura 63 se presenta el fragmento de código que determina la clasificación y el color de alerta según el porcentaje de severidad calculado.

Figura 63

Clasificación, color de alerta y envío a Firebase

```
status = "Infección Activa" if severity > 5.0 else "Sano"
color = "#00FF00" if severity < 5 else "#FFFF00" if severity < 25 else "#FF0000"

self.lbl_severity.configure(text=f"INFECCIÓN:\n{severity:.1f}%", text_color=color)
self.lbl_status.configure(text=f"[{status.upper()}]", text_color=color)

# 3. Empaquetar imagen si la política lo exige
imagen_b64 = None
if getattr(fb, 'config', {}).get("guardar_img", False):
    _, buffer = cv2.imencode('.png', frame)
    imagen_b64 = base64.b64encode(buffer).decode('utf-8')

exito = fb.log_analysis(severity, status, imagen_b64)
if exito:
    self.lbl_telemetria.configure(text="SYS.CPU: --% | MODEL: HYBRID_ONNX | UPLINK: OK")
else:
    self.lbl_telemetria.configure(text="SYS.CPU: --% | MODEL: HYBRID_ONNX | UPLINK: ERROR")
```

Nota: *Autoría propia*

En esta sección se presenta el código utilizado para el conteo de píxeles, la verificación de la presencia de la hoja y el cálculo de la severidad. La función cv2.countNonZero permite contabilizar los píxeles activos en cada máscara, obteniendo así el valor de Phoja, que corresponde a los píxeles verdes de la zona sana, y el valor de Pmildiu, asociado a los píxeles blancos y grises que representan la zona afectada.

Posteriormente, el sistema verifica que la cantidad de píxeles verdes sea mayor a 5.000. En caso de no cumplirse esta condición, el análisis se detiene y la interfaz muestra el mensaje “No hay hoja”, indicando que no se ha detectado una hoja válida frente a la cámara.

Si la condición se cumple, el sistema procede a calcular el porcentaje de severidad, el cual representa la proporción de tejido enfermo en relación con el total de tejido detectado en la hoja, expresado en porcentaje. El porcentaje de severidad se calculó en todos los casos mediante la siguiente expresión:

$$Severidad\ (\%) = \frac{P_{mildiu}}{P_{hoja} + P_{mildiu}}\ x\ 100 \qquad (2)$$

Mediante la prueba realizada se obtuvo los siguientes resultados en los diferentes niveles:

- **Nivel Preventivo — 0.3%**

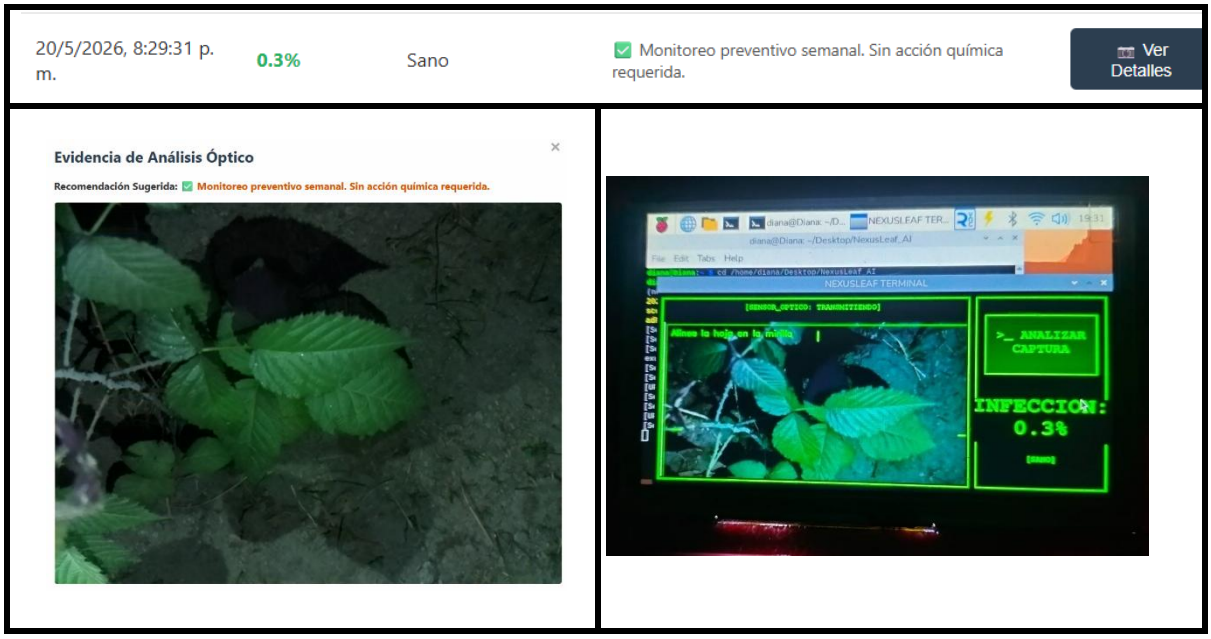
La hoja analizada presentaba una condición óptima sin síntomas visibles de enfermedad. El sistema detectó una cantidad mínima de píxeles en el rango del mildiu, obteniendo:

$$Severidad\ (\%) = \frac{24}{8.000 + 24}\ x\ 100 = \mathbf{0.3}\% \qquad (3)$$

Al ser menor al umbral del 5%, el sistema clasificó la hoja como Sana y mostró el resultado en color verde en la interfaz. Se emitió la recomendación de monitoreo preventivo semanal sin necesidad de acción química. En la figura 64 se muestra los diferentes resultados de análisis menor al 5%.

Figura 64

Resultado del análisis — Nivel Preventivo (0.3%)



Nota: *Autoría propia*

Nivel Alerta — 10.5%

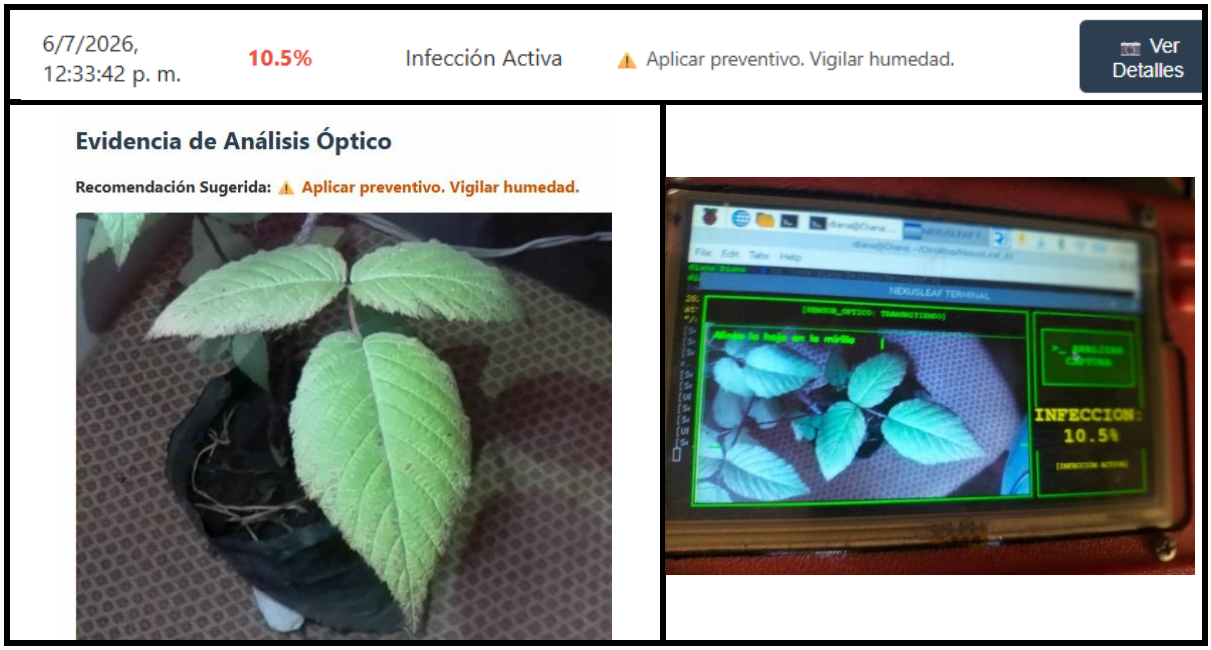
La hoja analizada presentaba manchas blancas visibles en zonas pequeñas de su superficie. El sistema detectó una proporción de píxeles enfermos superior al umbral preventivo, obteniendo:

$$Severidad\ (\%) = \frac{855}{8145 + 855}\ x\ 100 = 10.\mathbf{5}\% \qquad (4)$$

Al superar el 5% y ser menor al 15%, el sistema clasificó la hoja como Infección Activa en nivel Alerta y mostró el resultado en color amarillo. Se emitió la recomendación de aplicar fungicida preventivo a base de azufre o cobre y vigilar la humedad del cultivo. En la figura 65 se muestra los diferentes resultados de análisis menor al 10.5%.

Figura 65

Resultado del análisis — Nivel Alerta (10.5%)



Nota: *Autoría propia*

4.4 Pruebas de comunicación del sistema

Las pruebas de comunicación del sistema se llevaron a cabo durante un período de un mes, con el objetivo de evaluar el funcionamiento continuo del dispositivo y verificar la correcta transmisión de los datos hacia la base de datos en Firebase. Durante este periodo, se realizaron múltiples pruebas para comprobar la estabilidad de la comunicación, la integridad de la información enviada y la ausencia de errores en el proceso de almacenamiento y actualización de los datos.

Asimismo, se verificó el correcto funcionamiento de la aplicación web, comprobando que la información registrada en Firebase se reflejara de manera adecuada y que la generación de alertas se ejecutara oportunamente ante la detección de una condición de riesgo.

4.4.1 Prueba de envío de datos a Firebase

Para la prueba de envío de datos se verificó la comunicación entre la Raspberry Pi 4 y Firebase Realtime Database en tiempo real. Esta funcionalidad fue implementada mediante la función log_analysis, la cual se ejecuta automáticamente cada vez que el sistema realiza un análisis de una hoja.

El proceso de envío sigue la siguiente secuencia. Primero, la Raspberry Pi realiza el análisis de la hoja y obtiene el porcentaje de severidad y el estado de diagnóstico.

Segundo, la función *log_analysis* empaqueta los datos en un registro que incluye la severidad detectada, el estado de la hoja y la marca de tiempo del análisis.

Tercero, el registro es enviado simultáneamente a dos nodos de Firebase: el nodo historial que acumula todos los registros anteriores y el nodo *lectura_actual* que mantiene siempre el resultado más reciente. En la figura 66 se presenta el código correspondiente a esta funcionalidad.

Figura 66

Código de envío de datos a Firebase Realtime Database

```python
def log_analysis(self, severidad, status="Sano", imagen_b64=None):
    if not firebase_admin._apps:
        return False

    try:
        timestamp_iso = datetime.datetime.now(datetime.timezone.utc).isoformat()
        #Envío de resultados
        payload = {
            "timestamp": timestamp_iso,
            "severidad": float(severidad),
            "status": status
        }

        # Solo incluir el blob pesado si la política lo exige y la UI lo proporcionó
        if self.config.get("guardar_img") and imagen_b64:
            payload["imagen_base64"] = imagen_b64
            print("[Secuántica Edge] Evidencia Raw empacada en payload.")

        historial_ref = db.reference('historial')
        nuevo_registro = historial_ref.push()
        nuevo_registro.set(payload)
        key = nuevo_registro.key

        current_ref = db.reference('lectura_actual')
        current_ref.set(payload)

        print(f"[UPLINK] Registro enviado al nodo historial: {key}")
        print(f"[Secuántica Edge] Nodo 'historial' actualizado: {severidad}% / {status}")
        return True
    except Exception as e:
        print(f"[Secuántica Error] Falla en pasarela de datos: {e}")
        return False
```

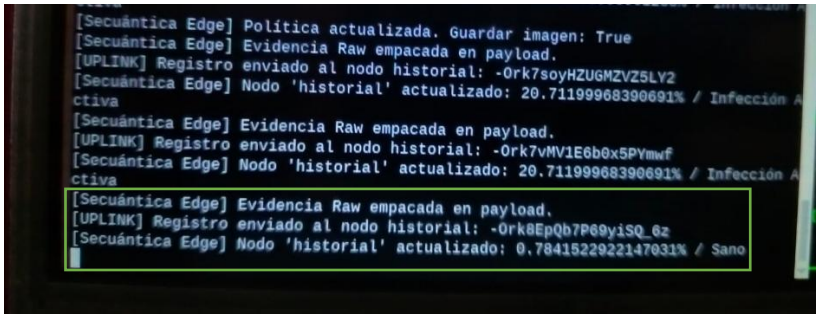
Nota: *Autoría propia*

Para verificar el correcto funcionamiento del envío de datos se realizó una prueba en la cual se analizó una hoja de mora de Castilla y se verificó que los datos llegaran correctamente a Firebase Realtime Database. En la figura 67 se puede observar la interfaz de la Raspberry Pi

mostrando el estado UPLINK: OK, lo que confirma que los datos fueron enviados exitosamente a la nube.

Figura 67

Confirmación de envío de datos en la interfaz de la Raspberry Pi



Nota: *Autoría propia*

Como se puede observar en la figura 64, el sistema generó los siguientes mensajes por cada análisis realizado, confirmando el correcto funcionamiento de la comunicación:

- **[UPLINK] Registro enviado al nodo historial:** indica que el resultado fue enviado exitosamente a Firebase Realtime Database con un identificador único generado automáticamente.
- **[Secuántica Edge] Nodo 'historial' actualizado:** confirma que el nodo de Firebase fue actualizado con el porcentaje de severidad y el estado de la hoja correspondiente.
- **[Secuántica Edge] Evidencia Raw empacada en payload:** indica que la imagen capturada fue convertida a formato base64 e incluida en el registro enviado a Firebase.

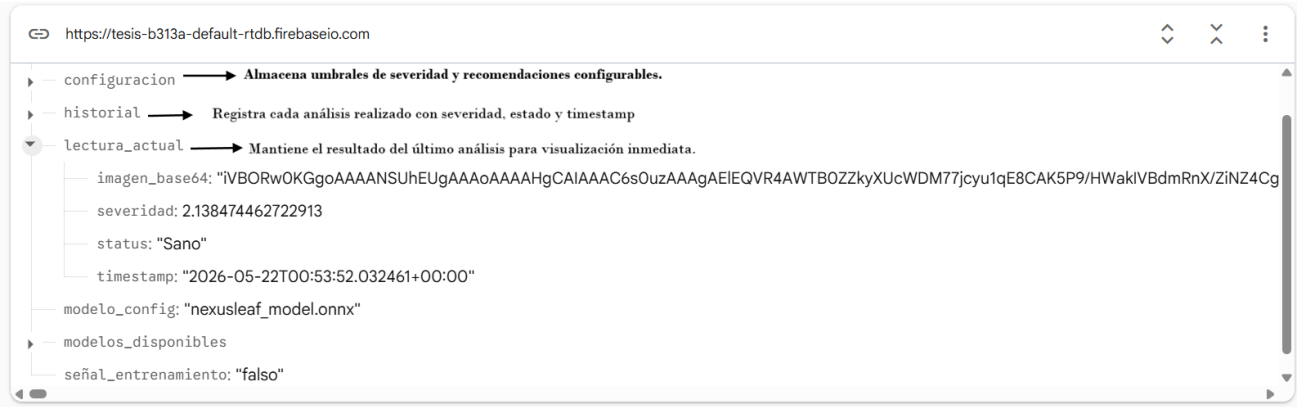
Durante las pruebas realizadas se analizaron hojas con distintos niveles de severidad de la enfermedad. Los resultados obtenidos permitieron establecer tres categorías principales: cuando la severidad fue menor al 5 %, la hoja se consideró sana; entre el 15 %, se determinó

que la infección se encontraba activa y requería monitoreo; mientras que, al superar el 30 % de severidad, se recomendó aplicar medidas de control de manera inmediata, ya sea mediante productos orgánicos o químicos, con el fin de evitar una mayor propagación de la enfermedad.

En la figura 68 se presenta la estructura de Firebase Realtime Database con los registros recibidos, donde se puede observar el timestamp, el porcentaje de severidad y el estado de diagnóstico de cada análisis realizado.

Figura 68

Datos recibidos en Firebase Realtime Database



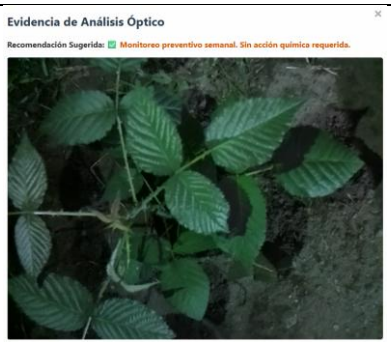
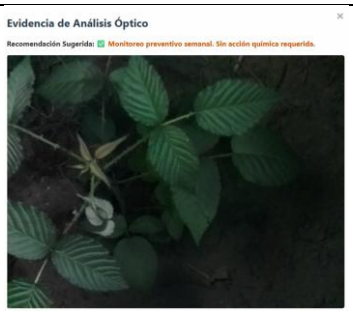
Nota: *Autoría propia*

4.4.2 Prueba de generación de alertas en la aplicación web

Con el objetivo de verificar el correcto funcionamiento del sistema de alertas, se realizó un monitoreo continuo durante cuatro semanas, comprendidas entre el 3 y el 30 de mayo de 2026, analizando diariamente hojas de mora de Castilla mediante la cámara OV5647 integrada en la Raspberry Pi 4. En cada sesión de análisis, el sistema ejecutó automáticamente la segmentación HSV, calculó el porcentaje de severidad y generó la alerta correspondiente, la cual fue enviada y registrada en la aplicación web a través de Firebase.

En la Tabla 24 se presenta el registro detallado de los análisis semanales, indicando la fecha, el porcentaje de severidad, el estado de diagnóstico y la recomendación emitida por el sistema para cada jornada de monitoreo.

Tabla 24
Registro semanal de análisis del cultivo

SEMANA 1 – Registro de monitoreo	
DATOS DEL ANÁLISIS	**EVIDENCIA VISUAL**
• Fecha: 03/5/2026 • Severidad: 2.3 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 3/5/2026, 6:55:20 p. m. 2.3% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 04/5/2026 • Severidad: 0.3 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 4/5/2026, 8:12:59 p. m. 0.3% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 05/5/2026 • Severidad: 0.5% • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 5/5/2026, 7:16:56 p. m. 0.5% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 06/5/2026 • Severidad: 0.8 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 6/5/2026, 7:51:02 p. m. 0.8% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	

- Fecha: 07/5/2026
- Severidad: 0.9 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

7/5/2026, 8:24:32 p. m.	0.9%	Sano	✅ Monitoreo preventivo semanal. Sin acción química requerida.



- Fecha: 08/5/2026
- Severidad: 0.6 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

8/5/2026, 8:58:47 p. m.	0.6%	Sano	✅ Monitoreo preventivo semanal. Sin acción química requerida.



- Fecha: 09/5/2026
- Severidad: 0.8 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

9/5/2026, 6:08:59 p. m.	0.8%	Sano	✅ Monitoreo preventivo semanal. Sin acción química requerida.



- Fecha: 10/5/2026
- Severidad: 1.5 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

10/5/2026, 5:56:25 p. m.	1.5%	Sano	✅ Monitoreo preventivo semanal. Sin acción química requerida.

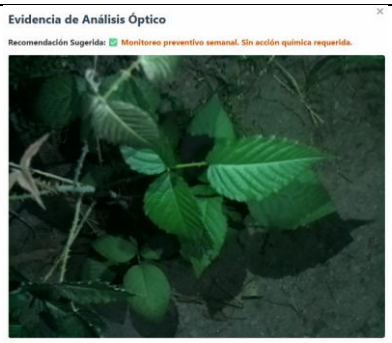


SEMANA 2 – Registro de monitoreo

- Fecha: 11/5/2026
- Severidad: 0.8 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

11/5/2026, 8:42:44 p. m.	0.8%	Sano	✅ Monitoreo preventivo semanal. Sin acción química requerida.



• Fecha: 12/5/2026 • Severidad: 0.4% • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 12/5/2026, 7:28:51 p. m. 0.4% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 13/5/2026 • Severidad: 0.3 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 12/5/2026, 7:30:13 p. m. 0.3% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 14/5/2026 • Severidad: 0.8 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 14/5/2026, 9:27:59 p. m. 0.4% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 15/5/2026 • Severidad: 2.1 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 15/5/2026, 8:32:16 p. m. 2.1% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 16/5/2026 • Severidad: 2.1 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 16/5/2026, 5:32:15 p. m. 2.1% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	

- Fecha: 17/5/2026
- Severidad: 0.7 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

17/5/2026, 3:48:06 p. m. | 0.7% | Sano | ✅ Monitoreo preventivo semanal. Sin acción química requerida.



SEMANA 3 – Registro de monitoreo

- Fecha: 18/5/2026
- Severidad: 2.0 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

18/5/2026, 9:45:17 p. m. | 2.0% | Sano | ✅ Monitoreo preventivo semanal. Sin acción química requerida.



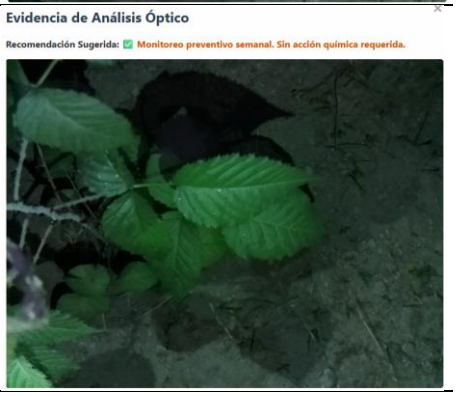
- Fecha: 19/5/2026
- Severidad: 0.8 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

19/5/2026, 7:22:12 p. m. | 2.8% | Sano | ✅ Monitoreo preventivo semanal. Sin acción química requerida.



- Fecha: 20/5/2026
- Severidad: 0.8 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

20/5/2026, 8:29:31 p. m. | 0.3% | Sano | ✅ Monitoreo preventivo semanal. Sin acción química requerida.



- Fecha: 21/5/2026
- Severidad: 5.3 %
- Estado: Infección Activa
- Recomendación: Aplicar Preventivo. Vigilar Humedad.

21/5/2026, 7:52:47 p. m. 5.3% Infección Activa ⚠ Aplicar preventivo. Vigilar humedad.



SEMANA 4 – Registro de monitoreo

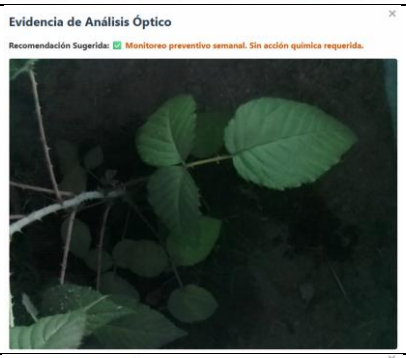
- Fecha: 22/5/2026
- Severidad: 1.1 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

22/5/2026, 8:52:05 p. m. 1.1% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.



- Fecha: 23/5/2026
- Severidad: 1.3 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

23/5/2026, 5:50:10 p. m. 1.3% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.



- Fecha: 24/5/2026
- Severidad: 0.3 %
- Estado: Sano
- Recomendación: Monitoreo preventivo semanal. Sin acción química requerida.

24/5/2026, 6:35:46 p. m. 0.3% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.



• Fecha: 25/5/2026 • Severidad: 0.4 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 25/5/2026, 7:47:26 p. m. **0.4%** Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 26/5/2026 • Severidad: 0.4 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 26/5/2026, 7:13:19 p. m. **0.4%** Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 09/5/2026 • Severidad: 0.8 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 27/5/2026, 7:58:23 p. m. **0.4%** Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 28/5/2026 • Severidad: 0.3 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 28/5/2026, 8:12:54 p. m. **0.3%** Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	
• Fecha: 29/5/2026 • Severidad: 0.2 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 29/5/2026, 9:18:09 p. m. **0.2%** Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	

• Fecha: 30/5/2026 • Severidad: 0.5 % • Estado: Sano • Recomendación: Monitoreo preventivo semanal. Sin acción química requerida. 30/5/2026, 8:17:47 p. m. 0.5% Sano ✅ Monitoreo preventivo semanal. Sin acción química requerida.	

Durante las cuatro semanas de prueba, el sistema registró un total de 29 análisis. En la gran mayoría de los casos, las hojas evaluadas presentaron niveles de severidad inferiores al 5%, siendo clasificadas como Sanas con recomendación de monitoreo preventivo semanal. La única excepción se registró el 21 de mayo de 2026, cuando el sistema detectó un nivel de severidad del 5,3%, clasificando la hoja como Infección Activa y emitiendo automáticamente la alerta correspondiente con la recomendación de aplicar fungicida preventivo y vigilar la humedad del cultivo. Este resultado demuestra la capacidad del sistema para identificar oportunamente el inicio de una infección, validando su funcionamiento en condiciones reales de campo.

4.5 Discusión Resultados

Los resultados obtenidos en este trabajo demuestran que el sistema IoT desarrollado es capaz de detectar mildiu polvoso en hojas de mora de Castilla con un alto nivel de precisión. El modelo YOLOv8n entrenado con 50 épocas alcanzó una precisión de 0,987 y un recall de 0,986, valores que superan los reportados en estudios similares de detección de enfermedades foliares mediante redes neuronales convolucionales. (Shahriar Zaman Abid et al., 2024), reportaron precisiones de entre 0,88 y 0,94 para la detección de enfermedades en cultivos de Bangladesh usando YOLOv8, lo que posiciona al modelo desarrollado en este trabajo como competitivo dentro del estado del arte.

En cuanto a la comparación entre configuraciones de entrenamiento, el experimento con 30 épocas presentó curvas de pérdida inestables, indicando que el modelo no alcanzó convergencia suficiente. Por el contrario, el entrenamiento con 100 épocas mostró síntomas de saturación sin mejora adicional en las métricas de validación, lo que sugiere que extender el entrenamiento más allá de cierto umbral no necesariamente mejora el rendimiento del modelo. La configuración de 50 épocas demostró ser el punto de equilibrio óptimo, consistent con el principio de compromiso entre sesgo y varianza. Este hallazgo coincide con lo observado por (Ghazal et al., 2024), quienes destacan que la selección adecuada de hiperparámetros es determinante en el rendimiento de modelos YOLO aplicados a visión artificial agrícola.

La integración del módulo de segmentación HSV mediante OpenCV añadió un nivel cuantitativo al sistema que va más allá de la clasificación binaria sano/enfermo. Al calcular el porcentaje exacto de tejido foliar afectado, el sistema permite una gradación de severidad en cuatro niveles (Preventivo, Alerta, Peligro y Crítico), lo cual tiene implicaciones directas en la toma de decisiones fitosanitarias. Esta capacidad diferenciadora resulta superior a los enfoques de clasificación simple documentados en trabajos previos sobre detección de mildiu en cultivos andinos. Sin embargo, cabe señalar que la eficacia de la segmentación HSV está condicionada a las condiciones de iluminación en campo, factor que pudo influir en las variaciones menores de severidad registradas entre días consecutivos.

La transmisión en tiempo real de resultados a Firebase y su visualización en la aplicación web constituyeron un aspecto diferenciador del sistema. A diferencia de soluciones que requieren conexión directa al dispositivo para la consulta de datos, la arquitectura basada en la nube permite al agricultor acceder al historial de detección desde cualquier dispositivo con acceso a internet. Durante las pruebas, todos los envíos de datos confirmaron el estado UPLINK: OK, verificando la fiabilidad del canal de comunicación Wi-Fi bajo las condiciones de la comunidad San Francisco de Guachinguero. No obstante, esta dependencia de

conectividad Wi-Fi representa una limitación para zonas con infraestructura de red insuficiente, aspecto que futuras versiones del sistema deberían abordar mediante tecnologías de comunicación alternativas como LoRaWAN.

El diseño del sistema con componentes de bajo costo y software de código abierto (YOLOv8, OpenCV, Firebase) representa una ventaja significativa para su eventual escalamiento y adopción en contextos agrícolas rurales del Ecuador. La combinación de hardware accesible con algoritmos de inteligencia artificial de alto rendimiento demuestra que es posible implementar soluciones tecnológicas eficaces sin requerir infraestructura costosa, lo que alinea el trabajo con los principios de agricultura de precisión inclusiva y con los Objetivos de Desarrollo Sostenible relacionados con la seguridad alimentaria y la innovación tecnológica.

4.6 Costo del Sistema

Los costos asociados a la implementación del sistema IoT para el control del mildiu polvoso en el cultivo de mora de Castilla se estructuran en tres categorías: hardware, software e infraestructura. Este desglose permite evaluar la inversión total requerida para su funcionamiento.

4.6.1 Costo del Hardware

La Tabla 25 presenta el desglose de los componentes utilizados para el sistema IoT, indicando la cantidad requerida y el precio unitario de cada elemento.

Tabla 25

Costo de Hardware

Hardware	Cantidad	Precio unitario (USD)	Total (USD)
Raspberry Pi 4 (8GB)	1	250.00	250.00
Cámara OV5647	1	58.00	58.00
Tarjeta SD (128GB)	1	12.00	12.00
Pantalla LCD	1	100.00	100.00
Fuente de Alimentación 5V Tipo C	1	10.80	10.80
Total Hardware			**430.80**

4.6.2 Costo del Software

En la tabla 26 se presenta los costos de los servicios en la nube utilizados para el almacenamiento de datos y visualización web.

Tabla 26

Costo de Software

Software	Cantidad	Precio Anual (USD)	Total (USD)
Raspberry Pi OS	1	0.00	0.00
OpenCV	1	0.00	0.00
Visual Studio Code	1	0.00	0.00
Firebase Realtime Database	1	100.00	100.00
Total Software			**100.00**

4.6.3 Costo de infraestructura

La tabla 27 incluye los costos relacionados a la infraestructura para el sistema IoT.

Tabla 27

Costo Infraestructura

Infraestructura	Cantidad	Precio Anual (USD)	Total (USD)
Carcasa	1	40.00	40.00
Cables	1	10.00	10.00
Total Infraestructura			**50.00**

4.6.4 Costo Total del Sistema

A continuación, se resumen en la tabla 28 de todos los costos involucrados, permitiendo visualizar el costo final de la inversión estimada.

Tabla 28

Costo Total del Sistema

Categoría	Subtotal (USD)
Hardware	430.8
Software	100
Infraestructura	50
Total General	**580.8**

El costo total del sistema es de aproximadamente $580,80, un valor que puede variar en función de la escala del despliegue, ya sea por la incorporación de nuevos nodos de monitoreo, la adquisición de hardware adicional o la contratación de planes superiores en los servicios de Firebase. Aun así, se trata de una inversión accesible considerando que la mayor parte del software utilizado es de código abierto y no genera costos. Para los pequeños productores de la comunidad San Francisco de Guachinguero, este sistema representa una alternativa tecnológica viable para su cultivo.

4.6.5 Análisis de Costo- Beneficio

La implementación del sistema permitió detectar oportunamente el amarillamiento de las hojas como síntoma temprano del mildiu polvoso durante la etapa de crecimiento

controlado, garantizando que las plantas alcanzaran los 50 cm requeridos en condiciones óptimas de salud para su trasplante definitivo. Una vez llevadas al exterior, donde las condiciones ambientales elevan considerablemente el riesgo de infección, el sistema continuó monitoreando su estado y alertando al agricultor ante cualquier indicio de la enfermedad antes de que esta avanzara de forma irreversible.

Otro beneficio del sistema es que permite detectar la enfermedad, y deja de depender solo de revisiones manuales y pasa a un monitoreo constante del cultivo. Esto ayuda a que el agricultor se dé cuenta a tiempo cuando aparecen los primeros signos del mildiu polvoso. Además, el sistema ayuda a usar mejor los insumos agrícolas, porque al detectar la enfermedad a tiempo se pueden aplicar fungicidas solo cuando es necesario y en la zona afectada, evitando gastar de más y reduciendo el uso innecesario de químicos en las plantas.

El beneficio radica en la capacidad de intervenir a tiempo con una aplicación puntual de fungicida preventivo, evitando que la planta muera. Considerando que cada planta tiene un costo de USD 1,00 y que un productor maneja entre 15 y 20 plantas en la fase de crecimiento controlado y alrededor de 30 en la plantación exterior, perder el lote por una infección no detectada representaría una pérdida de entre USD 30,00 y USD 50,00 solo en reposición, sin contar el tiempo invertido en su crecimiento ni el retraso en el ciclo productivo.

Frente a una inversión total $580.80 el sistema representa una solución rentable y escalable para los pequeños productores de la comunidad San Francisco de Guachinguero, quienes no siempre cuentan con asistencia técnica permanente. El costo puede variar si se amplía el número de nodos o se requieren planes superiores en Firebase, pero en cualquier caso se trata de una herramienta accesible que reduce pérdidas acumuladas ciclo tras ciclo y disminuye el uso innecesario de agroquímicos.

CONCLUSIONES

Se logró implementar el sistema IoT de manera exitosa, integrando todos los componentes propuestos: la Raspberry Pi 4, la cámara OV5647, el modelo YOLOv8n y la plataforma Firebase.

El modelo entrenado con 50 épocas demostró ser la configuración más equilibrada, superando tanto a la de 30 épocas, que presentó inestabilidad en el aprendizaje, como a la de 100 épocas, donde el modelo llegó a un punto de saturación sin mejorar. Con 50 épocas se obtuvo una precisión de 0.987 y un recall de 0.986, identificando correctamente el 95% de las hojas enfermas y el 100% de las sanas.

La combinación del modelo YOLOv8 con la segmentación HSV permitió ir más allá de una simple clasificación sano/enfermo, ya que el sistema calcula el porcentaje exacto de tejido afectado y entrega una recomendación específica según el nivel de severidad detectado. Esto le da al agricultor información concreta sobre qué producto aplicar y cuándo, lo cual es mucho más útil.

El sistema es capaz de identificar el inicio de una infección antes de que se vuelva crítica, que es justamente el momento en que intervenir. Además, al estar estos resultados disponibles en la aplicación web, el agricultor pudo consultarlos, revisar el historial de detecciones y recibir notificaciones desde cualquier dispositivo con internet, sin necesidad de conocimientos técnicos, lo que confirma que el sistema no solo funciona bien técnicamente, sino que también es accesible y útil para el perfil de los productores de la comunidad. Durante las cuatro semanas de pruebas en campo, el sistema detectó oportunamente el único caso de infección activa registrado, con una severidad del 5.3%, generando de forma automática la alerta y la recomendación correspondiente.

RECOMENDACIONES

Se recomienda incorporar la detección de otras enfermedades frecuentes en el cultivo como Botrytis cinerea y la roya, de manera que el sistema pueda ofrecer un diagnóstico más completo y no limitarse únicamente al mildiu polvoso. Esto requeriría construir un nuevo dataset con imágenes de cada enfermedad y reentrenar el modelo.

Para zonas donde no se dispone de señal Wi-Fi estable, se sugiere evaluar el uso de tecnologías de comunicación de largo alcance como LoRaWAN, que permitiría llevar el sistema a comunidades más alejadas o con infraestructura de red limitada, ampliando así su impacto en el sector agrícola rural.

Se recomienda utilizar Google Colab para el entrenamiento de futuros modelos, ya que ofrece acceso gratuito a GPU, lo que permitiría reducir considerablemente el tiempo de entrenamiento en comparación con el uso de un computador local sin GPU dedicada. Esto facilitaría probar más configuraciones y arquitecturas de manera más ágil para futuras mejoras del sistema.

BIBLIOGRAFÍA

Abril Espin, E., Salas Macías, C., & Garcés Fiallos, F. (2024). Sustainability of blackberry (Rubus glaucus) in Ecuadorian family farming. *Revista Ciencia y Tecnología*, *20*(4), 87-97. https://doi.org/10.17268/rev.cyt.2024.04.07

Alexandra Mora-Ramos, M., Agro, I., Patricia Pardo-Carrasco, F., Agr, I., Bastidas-López, H., Citar, C., & Diagnóstico Patológico en Mora de Castilla Rubus glaucus Bentham, B.-L. H. (2020). *ARTÍCULO ORIGINAL/ORIGINAL ARTICLE*. https://doi.org/10.22579/20112629.618

Arias, S. M., Álvarez, G. E. G., & Patiño, P. A. G. (2020). Diagrammatic scale for measuring severity of gray mould in thornless castilla blackberry (Rubus glaucus benth). *Ciencia Rural*, *50*(11), 1-7. https://doi.org/10.1590/0103-8478cr20190859

Avşar, E., & Mowla, M. N. (2022). Wireless communication protocols in smart agriculture: A review on applications, challenges and future trends. En *Ad Hoc Networks* (Vol. 136). Elsevier B.V. https://doi.org/10.1016/j.adhoc.2022.102982

Benavides, M. & C. W. & G. M. & Z. N. Y. & M. L. E. & B. M. Luis. (2021). *Mora (Rubus glaucus Benth.)*.

Bonilla-González, J. P., & Prieto-Ortiz, F. A. (2016). Determinación del estado de maduración de frutos de feijoa mediante un sistema de visión por computador utilizando información de color. *REVISTA DE INVESTIGACIÓN, DESARROLLO E INNOVACIÓN*, *7*(1), 111. https://doi.org/10.19053/20278306.v7.n1.2016.5603

Cevallos B., L. A. (2020). *MANEJO AGRONÓMICO DEL CULTIVO DE MORA DE CASTILLA (Rubus glaucus)*.

Cevallos Suarez, M. P. (2023). *PLANDE DESARROLLO Y ORDENAMIENTO TERRITORIAL DE LA PARROQUIA SAN JOSE DE QUICHINCHE*.

De, A., & Hueso, L. E. (s. f.). *COPIA PRIVADA CON FINES DOCENTES VISIÓN POR COMPUTADOR*.

De, A., & Hueso, L. E. (2020). *COPIA PRIVADA CON FINES DOCENTES VISIÓN POR COMPUTADOR*.

Ernest, K. , Volker. R. (2010). *MICRONUTRIENTES EN LA FISIOLOGIA DE LAS PLANTAS: FUNCIONES, ABSORCION Y MOVILIDAD*.

Franco, G., Bernal Estrada, J. A., Váquez Gallo, L. A., Saldarriaga Cardona, A., Henao Rojas, J. C., Díaz Montaño, J., Díaz Montilla, A. E., Aguilera Arango, G. A., García Muñoz, M. C., Argüelles Cárdenas, J. H., Betancourt Vásquez, M., Sánchez Betancourt, É. P., Zapata Narváez, Y. A., Beltrán Acosta, C. R., Cotes Prado, A. M., Antía Londoño, G. A., Orrego Alzate, C. E., Salgado Alzate, N., Rodríguez, L. J., … Tamayo Molano, P. J. (2020). Tecnología para el cultivo de la mora (Rubus glaucus Benth). En *Tecnología para el cultivo de la mora (Rubus glaucus Benth)*. Corporación Colombiana de Investigación Agropecuaria (Agrosavia). https://doi.org/10.21930/agrosavia.manual.7403251

Garrido, P., Morillo, E., & Vásquez-Castillo, W. (2020). Genetic diversity of the Andean blackberry (Rubus glaucus Benth.) in Ecuador assessed by AFLP markers. *Plant Genetic Resources: Characterisation and Utilisation*, *18*(4), 243-250. https://doi.org/10.1017/S1479262120000283

Germán, F. M. J. C. (2020). *El cultivo de la Mora*.

Ghazal, S., Munir, A., & Qureshi, W. S. (2024a). Computer vision in smart agriculture and precision farming: Techniques and applications. En *Artificial Intelligence in Agriculture* (Vol. 13, pp. 64-83). KeAi Communications Co. https://doi.org/10.1016/j.aiia.2024.06.004

Ghazal, S., Munir, A., & Qureshi, W. S. (2024b). Computer vision in smart agriculture and precision farming: Techniques and applications. En *Artificial Intelligence in Agriculture* (Vol. 13, pp. 64-83). KeAi Communications Co. https://doi.org/10.1016/j.aiia.2024.06.004

Hurtado, N. C., Álvarez, G. E. G., & Gutiérrez, A. M. L. (2019). Identificação de Peronospora sparsa e avaliação do conteúdo de fenóis em frutos de «mora de castilla» afetados por esse microrganismo. *Revista Ceres*, *66*(1), 11-17. https://doi.org/10.1590/0034-737X201966010002

IniAP. (2011). *El cultivo de la MORA En Ecuador*. http://www.sxc.hu/1048657_51921463

Martínez, A. A. P., Castro, S. P., & Quintero, J. F. K. (2019). Characterization of the subsystem pests of the thornless blackberry (Rubus Glaucus Benth.) in the department of Risaralda. En *Revista Luna Azul* (Número 49, pp. 162-171). Universidad de Caldas. https://doi.org/10.17151/luaz.2019.49.9

Ministerio de Agricultura, G. A. y P. (2015). *01-06PPP2015-POLITICA01*.

Naciones, U. (2024). *THE 17 GOALS | Sustainable Development*. https://sdgs.un.org/goals

Prefectura de Imbabura. (2020). *GOBIERNO PROVINCIAL DE IMBABURA*.

Ragab, M. G., Abdulkadir, S. J., Muneer, A., Alqushaibi, A., Sumiea, E. H., Qureshi, R., Al-Selwi, S. M., & Alhussian, H. (2024). A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023). *IEEE Access*, *12*, 57815-57836. https://doi.org/10.1109/ACCESS.2024.3386826

Restrepo, J. C. (2021). *Manejo fitosanitario del cultivo de la mora Medidas para la temporada invernal*. http://www.sxc.hu/1048657_51921463

Rodas Vásquez, A. (2024). *Vista de Desarrollo de plataforma de bajo costo para el monitoreo de condiciones microclimáticas favorables al surgimiento del hongo Botrytis Cinerea en el cultivo de mora de Castilla departamento Risaralda*.

Salcedo, R. D. E. (2019). *«Evaluación de características botánicas morfológicas y físico-químicas, y el contenido de polifenoles y vitamina C de cuatro cultivares de*

mora (Rubus glaucus) para determinar su estabilidad durante el período de cosecha».

Santiago, M., & Barona, T. (2018). *CONTROL DE BOTRYTIS (Botrytis cinérea) EN MORA DE CASTILLA (Rubus glaucus Bent)*.

Shahriar Zaman Abid, M., Jahan, B., Mamun, A. Al, Jakir Hossen, M., & Hossain Mazumder, S. (2024). Bangladeshi crops leaf disease detection using YOLOv8. *Heliyon*, *10*(18). https://doi.org/10.1016/j.heliyon.2024.e36694

Sinha, E., Tyagi, A., & Kumar, A. (2025). OpenCV for Computer Vision Applications. En *IJFMR250344280* (Vol. 7). www.ijfmr.com

Suquilanda Valdivieso, M. B. (2011). Producción Orgánica de Cultivos Andinos; Manual técnico. FAOEC. *FAO : Ministerio de Agricultura, Ganaderia, Acuacultura y pesca : Unión de Organizaciones Campesinas del Norte de Cotopaxi*. https://redbibliotecas.quito.gob.ec/cgi-bin/koha/opac-detail.pl?biblionumber=49957

Tovar Soto, J. P., Solórzano Suárez, J. D. los S., Badillo Rodríguez, A., & Rodríguez Cainaba, G. O. (2019). Internet de las cosas aplicado a la agricultura: estado actual. *Lámpsakos*, (22), 86-105. https://doi.org/10.21501/21454086.3253

Valdez, P., Marina, S., Giovanny, I. M., & Espín, P. V. (2023). *«Evaluación de tres formas de fertilización en el cultivo de mora de castilla (Rubus glaucus Benth)»*.

Valverde, F. G. A. V. D. , P. M. A. (2016). *Nutrición del cultivo de la mora de castilla*. http://repositorio.iniap.gob.ec/handle/41000/4065

Wang, W., & Kang, Y. (2025). *A Review of Computer Vision Technologies in Precision Agriculture*. https://doi.org/10.54254/2753-8818/101/2025.CH22224

Yusuf, A., & Badrul, M. (2024). *PERANCANGAN MODEL WATERFALL PADA SISTEM INFORMASI PENJUALAN BAJU PADA BRAND HASNAA BUSANA*.

5 ANEXO 1: Encuesta

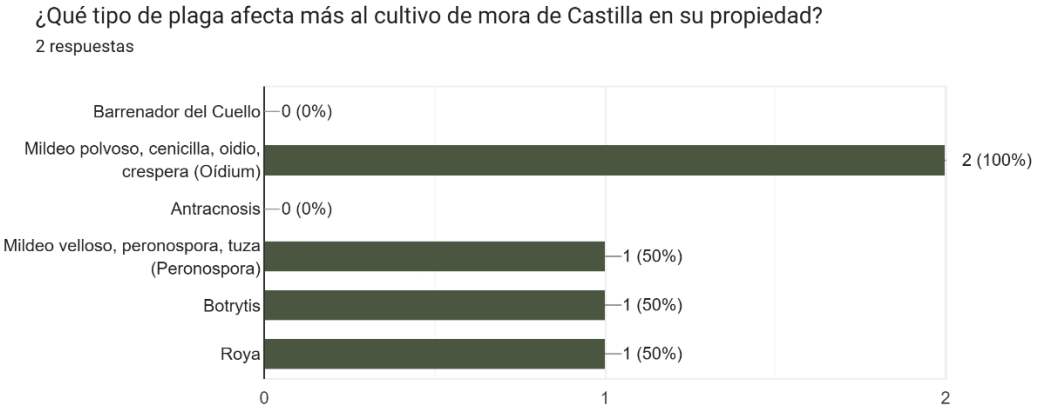
5.1 Formato de la encuesta

El objetivo de la encuesta fue conocer qué tipo de plaga afecta con mayor frecuencia al cultivo de mora de Castilla y, a partir de ello, generar posibles alertas que ayuden a su control. Además, se buscó recopilar información sobre los principales síntomas que presentan las plantas, el uso de fertilizantes en el cultivo y los productos más empleados por los agricultores, con el fin de contar con un panorama claro sobre las prácticas actuales de manejo y las necesidades más urgentes en la producción.

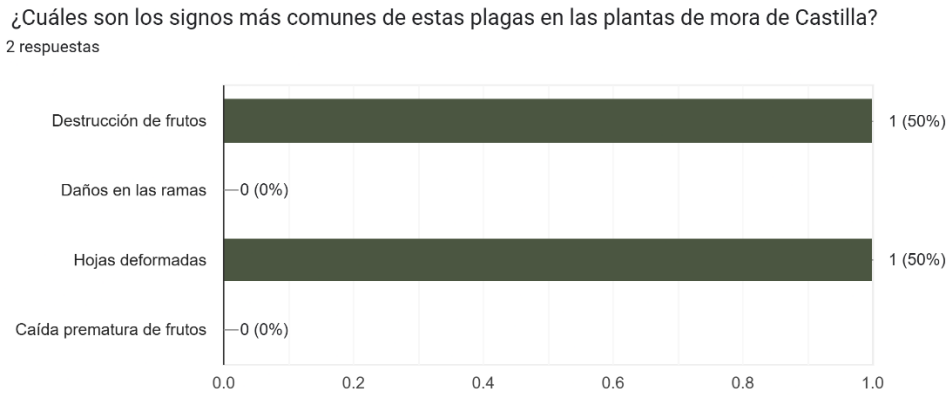
A continuación, se muestra el contenido de las preguntas aplicadas en la encuesta, las cuales permitieron identificar las plagas más comunes, los signos visibles en las plantas, la frecuencia con la que aparecen estas enfermedades, el tipo de fertilizantes que se utilizan, la frecuencia de aplicación y la manera en que los productores deciden las dosis empleadas. Esta información constituye la base para analizar la situación real del cultivo de mora de Castilla y plantear soluciones que mejoren su manejo.

Nombre: Gloria Tuquerres

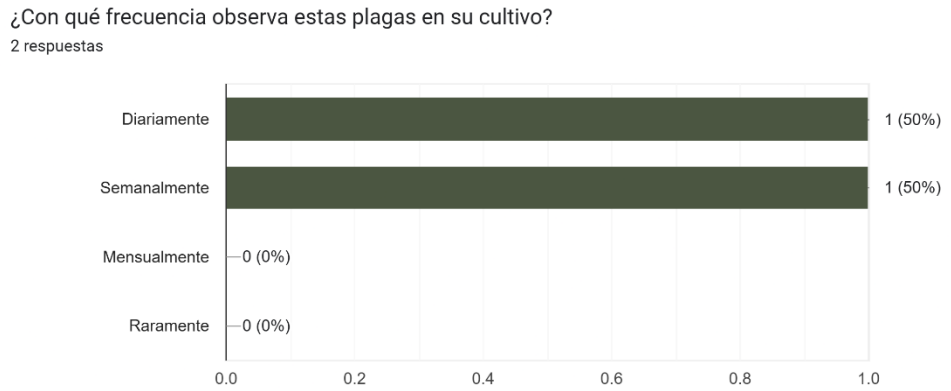
Pregunta 1: ¿Qué tipo de plaga afecta más al cultivo de mora de Castilla en su propiedad?



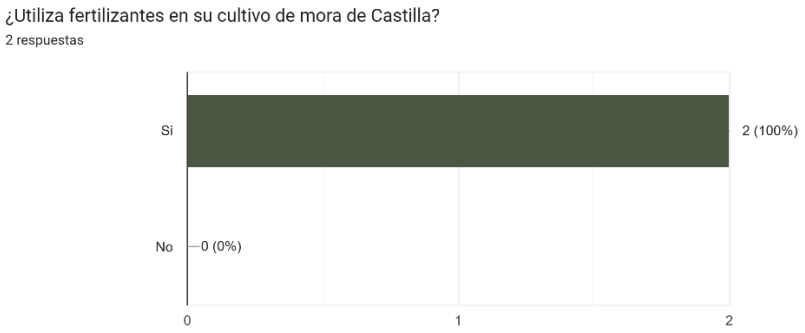
Pregunta 2: ¿Cuáles son los signos más comunes de estas plagas en las plantas de mora de Castilla?



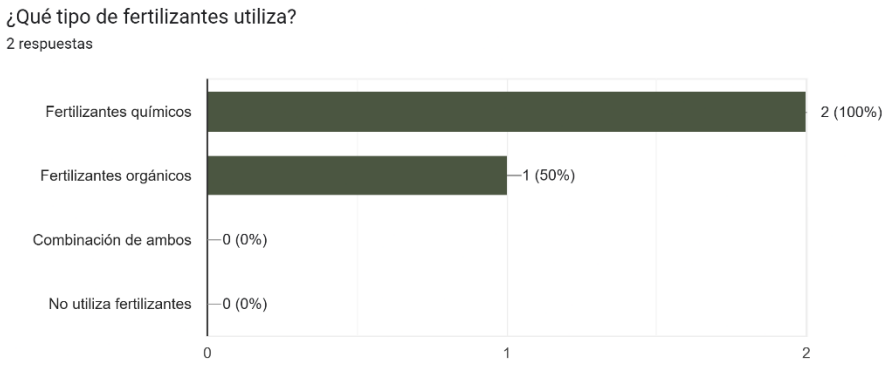
Pregunta 3: ¿Con qué frecuencia observa estas plagas en su cultivo?



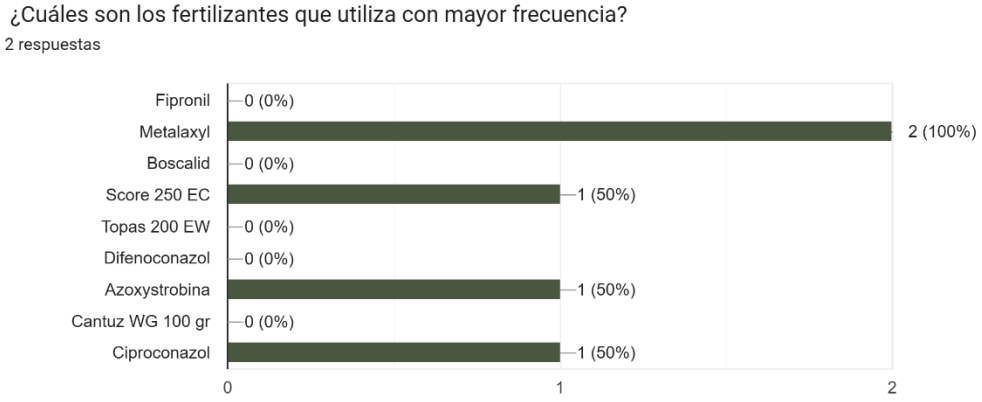
Pregunta 4: ¿Utiliza fertilizantes en su cultivo de mora de Castilla?



Pregunta 5: ¿Qué tipo de fertilizantes utiliza?



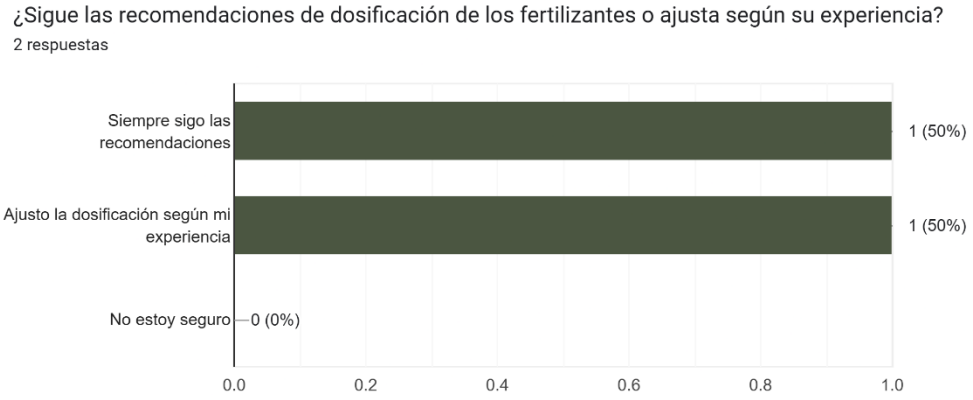
Pregunta 6: ¿Cuáles son los fertilizantes que utiliza con mayor frecuencia?



Pregunta 7: ¿Con qué frecuencia aplica fertilizantes a su cultivo?



Pregunta 8: ¿Sigue las recomendaciones de dosificación de los fertilizantes o ajusta según su experiencia?



6 ANEXO 2: Manual de Usuario



Este manual describe el procedimiento para utilizar el sistema IoT de detección de mildiu polvoso en mora de Castilla. Está dirigido a los agricultores y personas encargadas del monitoreo del cultivo, sin necesidad de conocimientos técnicos especializados.

6.1 Requisitos previos

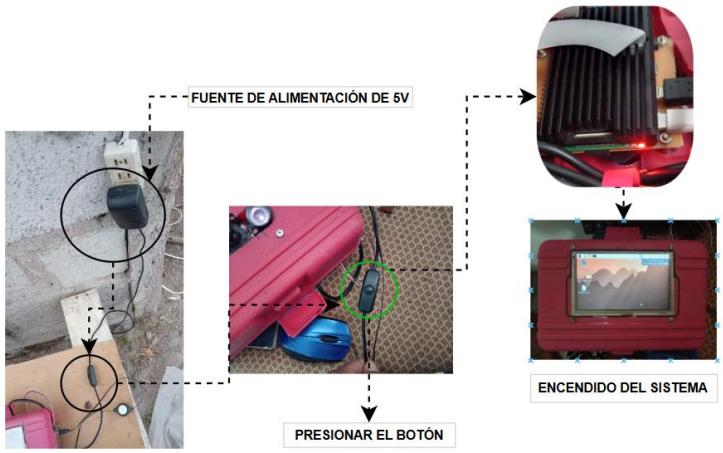
Para utilizar el sistema correctamente, el usuario debe disponer de lo siguiente: el dispositivo Raspberry Pi 4 correctamente instalado en el cultivo con la cámara OV5647 orientada hacia las plantas; conexión a una red Wi-Fi disponible en la zona del cultivo; un dispositivo con acceso a internet (teléfono celular, tablet o computadora) para visualizar los resultados en la aplicación web; y la dirección URL de la aplicación web del sistema, proporcionada durante la instalación.

6.1.1 Encendido del sistema

Para encender el sistema, conecte la fuente de alimentación de la Raspberry Pi al tomacorriente o banco de energía. El dispositivo iniciará automáticamente y en aproximadamente 60 segundos estará operativo. El indicador LED verde en la Raspberry Pi parpadeando de forma regular indica que el sistema está en funcionamiento normal.

Figura 69

Encendido del Sistema



Nota: *Autoría propia*

6.1.1.1 Inicio del Monitoreo

Una vez que la pantalla esté encendida, para activar el sistema de monitoreo se deben ingresar los siguientes tres comandos en la terminal de la Raspberry Pi 4:

- cd /home/diana/Desktop/NexusLeaf_AI: Accede al directorio del proyecto ubicado en el escritorio de la Raspberry Pi 4.
- source nexus_env/bin/activate: Activa el entorno virtual, el cual contiene todas las librerías (onnxruntime, opencv y firebase_admin), para el funcionamiento del sistema.
- python src/mision1_ui.py: Se activa la pantalla del monitoreo.

Figura 70

Inicia Monitoreo



Nota: *Autoría propia*

6.1.2 Funcionamiento

Una vez que el sistema esté funcionamiento, siga los siguientes pasos para realizar el análisis de una hoja:

- **Paso 1:** Colocar la planta de mora de castillas frente a al cámara OV5647, que este bien centrada a una distancia de 45cm del lente, sin obstrucciones la hoja y la cámara.

- **Paso 2:** En la pantalla del dispositivo presione el botón *"ANALIZAR CAPTURA"*. El sistema tomará automáticamente la fotografía de la hoja y procesará la imagen para detectar la presencia de mildiu polvoso.

- **Paso 3:** En pocos segundos el sistema mostrará en pantalla el resultado del análisis, indicando el porcentaje de severidad detectado y el estado de la hoja según los niveles establecidos.

- **Paso 4:** De forma simultánea, el resultado será enviado automáticamente

a Firebase y podrá ser consultado desde la aplicación web en cualquier dispositivo con acceso a internet.

Figura 71

Análisis de hoja



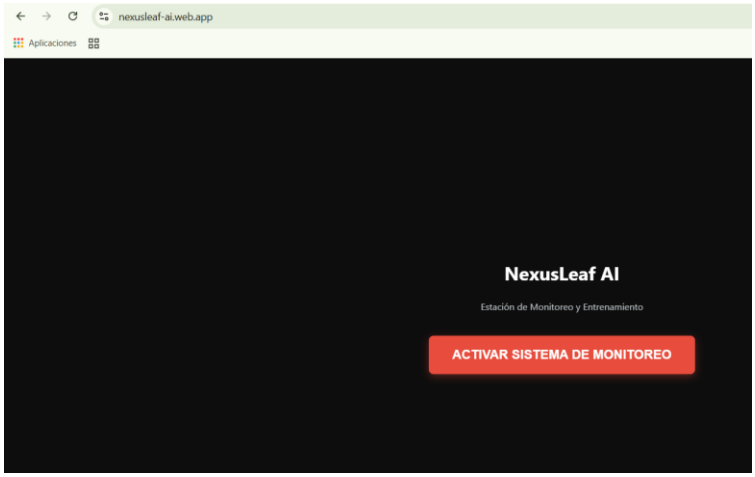
Nota: *Autoría propia*

6.1.3 Visualización de Resultados en la Aplicación Web

Para consultar los resultados del monitoreo, abra un navegador web en su dispositivo (Chrome) e ingrese la dirección URL de la aplicación: https://nexusleaf-ai.web.app/.

Figura 72

Inicio de página

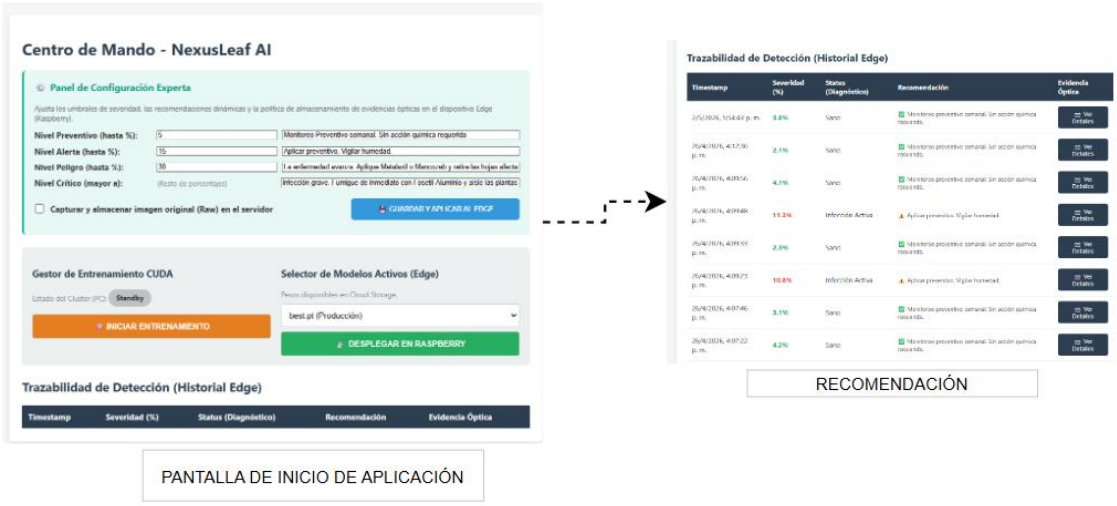


Nota: *Autoría propia*

En la pantalla de inicio se mostrará el último análisis realizado con la siguiente información: fecha y hora del análisis, porcentaje de severidad detectado, estado de la hoja (Sano, Infección Activa, Alerta, Peligro o Crítico) y la recomendación de tratamiento correspondiente. Para revisar el historial completo de análisis, seleccione la opción "Historial de Detecciones" en el menú de la aplicación, donde podrá ver todos los registros organizados por fecha.

Figura 73

Pantalla de inicio



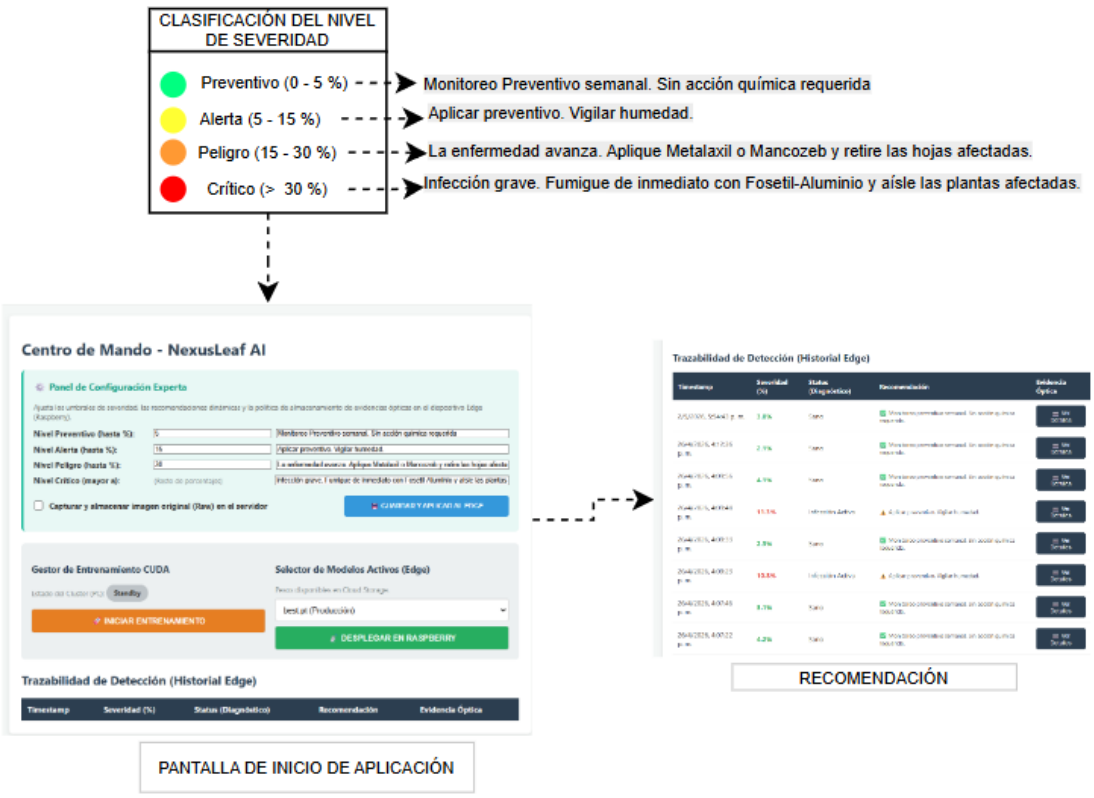
Nota: *Autoría propia*

6.1.4 Interpretación de Alertas y Acciones Recomendadas

El sistema clasifica el estado de las hojas en cuatro niveles según el porcentaje de tejido afectado. El nivel Preventivo (0–5% de severidad) indica que la hoja está sana; la acción recomendada es continuar con el monitoreo preventivo semanal sin aplicación de agroquímicos. El nivel Alerta (5–15% de severidad) indica infección activa incipiente; se recomienda aplicar fungicida preventivo a base de cobre y vigilar la humedad del cultivo. El nivel Peligro (15–30% de severidad) indica infección en avance; se debe aplicar fungicida específico de manera inmediata y aumentar la frecuencia de monitoreo a cada dos días. El nivel Crítico (más del 30% de severidad) indica infección grave; se recomienda aplicación urgente de fungicida sistémico y consultar con un técnico agrícola de ser posible.

Figura 74

Interpretación de Alertas y Acciones Recomendadas



Nota: *Autoría propia*

7 Anexo 3: Manual de administrador



MANUAL DE ADMINISTRADOR



Este manual está dirigido al personal técnico responsable de instalar, configurar, operar y mantener el sistema: una plataforma IoT de control de mildiu polvoso en cultivos de mora Castilla, desarrollada para la comunidad de San Francisco de Guachinguero. A diferencia del Manual de Usuario, que describe el uso cotidiano del panel web por parte de los agricultores, este documento cubre los aspectos de infraestructura: hardware, librerías, configuración de la nube y mantenimiento.

El sistema está compuesto por tres nodos que se comunican exclusivamente a través de Firebase: el nodo Edge (Raspberry Pi 4 + cámara OV5647) que captura y analiza, la estación donde se entrena el modelo YOLOv8, y el panel de administración web.

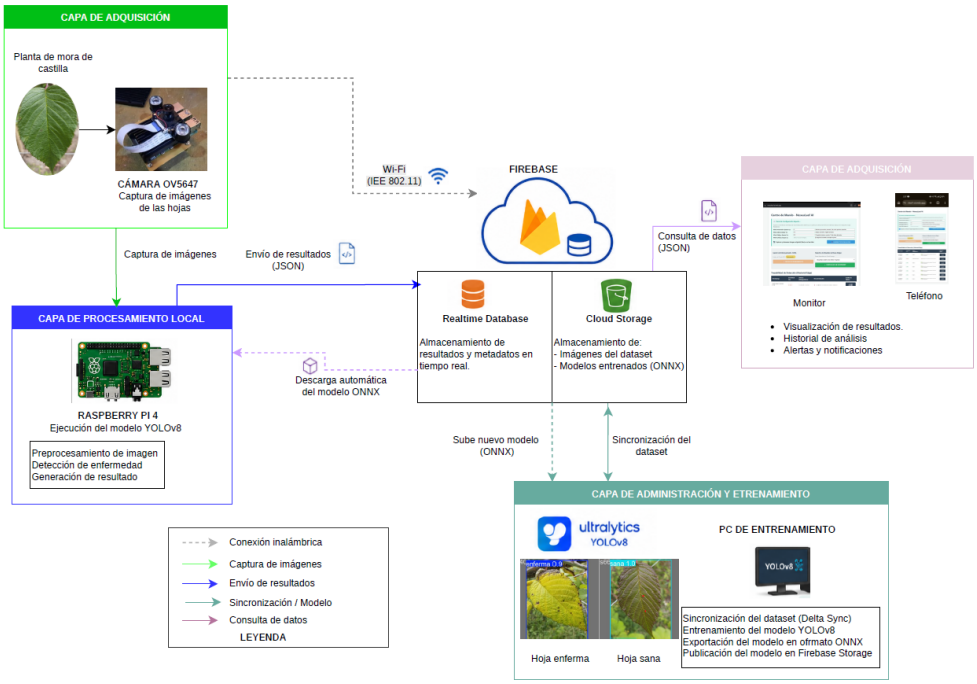
7.1 Arquitectura General para el administrador

La arquitectura del sistema está conformada por cuatro componentes principales que interactúan de forma continua mediante una red Wi-Fi con acceso a Internet. La *Raspberry Pi 4 con cámara OV5647* (captura imágenes de las hojas de mora de castilla), las cuales son procesadas por la *Raspberry Pi* 4 utilizando un modelo de inteligencia artificial basado en *YOLOv8* para determinar el estado del cultivo.

Los resultados obtenidos son enviados automáticamente a *Firebase Realtime Database* (intercambio de datos en formato JSON), mientras que las imágenes capturadas y los modelos entrenados se almacenan en *Firebase Storage* (almacenamiento de imágenes del dataset y de los modelos entrenados en formato .onnx) y la *aplicación web* (dashboard de visualización), consulta esta información en tiempo real para mostrar el historial de monitoreo y los resultados del análisis al usuario. En la figura 75 se muestra arquitectura general con los nodos principales.

Figura 75

Arquitectura general del sistema IoT



Nota: *Autoría propia*

7.2 Requisitos del sistema

Para garantizar el correcto funcionamiento del sistema, es necesario disponer de los siguientes componentes de hardware. Estos elementos conforman la infraestructura mínima requerida para la captura de imágenes, el procesamiento de los datos y la comunicación con la plataforma en la nube.

En la tabla 29 se muestra los requisitos del Hardware.

Tabla 29

Requisitos del Hardware

Componente	Características
Modelo del Equipo	Laptop AMD Ryzen 5 7520U
Procesador	AMD Ryzen 5 7520U with Radeon Graphics (2,80 GHz)
Memoria RAM	16 GB
Sistema Operativo	Windows 11
Unidad de almacenamiento	500GB
Arquitectura del Sistema	64 bits

También se presentan las especificaciones técnicas de la Raspberry Pi 4 utilizada como dispositivo de inferencia en campo, sobre el cual se ejecuta el modelo en tiempo real una vez finalizado el entrenamiento.

Tabla 30

Requisitos Raspberry Pi 4

Componente	Características
Modelo del Equipo	Raspberry Pi 4 Model B
Procesador	ARM Cortex-A72, 4 núcleos a 1.8 GHz
Memoria RAM	8 GB
Almacenamiento	MicroSD 128 GB
Sistema Operativo	Raspberry Pi OS (64 bits)
Unidad de procesamiento	CPU ARM
Cámara	OV5647 — 5 megapíxeles, interfaz CSI
Conectividad	Wi-Fi 802.11
Motor de Interferencia	onnxruntime
Formato Modelo	ONNX — YOLOv8n

7.3 Librerías por componente

El desarrollo del sistema requirió el uso de diversas librerías de Python y herramientas de software, cada una destinada a cumplir una función específica dentro de los diferentes componentes que conforman la arquitectura del sistema. Estas librerías permitieron implementar la captura y procesamiento de imágenes, la inferencia del modelo de inteligencia artificial, la comunicación con los servicios en la nube, la sincronización del conjunto de datos, el entrenamiento del modelo y la administración de los recursos del sistema.

- **Librerías para el procesamiento de imágenes**

Estas librerías permiten capturar, procesar y manipular las imágenes obtenidas por la cámara de la Raspberry Pi antes de realizar la detección de la enfermedad.

Tabla 31

Librerías para el procesamiento de imágenes

Librería	Función
opencv-python (cv2)	Captura imágenes desde la cámara, realiza el procesamiento digital, la conversión entre espacios de color y la segmentación mediante HSV.
NumPy	Manipula matrices y arreglos numéricos utilizados durante el procesamiento de imágenes y la preparación de los datos para el modelo de inteligencia artificial.
Pillow (PIL)	Convierte imágenes al formato requerido por la interfaz gráfica desarrollada con CustomTkinter.

- **Librerías para inteligencia artificial**

Estas librerías permiten entrenar el modelo de detección y ejecutar la inferencia en la Raspberry Pi.

Tabla 32

Librerías para inteligencia artificial

Librería	Función
Ultralytics (YOLOv8)	Entrena el modelo de detección de objetos y exporta el modelo entrenado al formato ONNX.
onnxruntime	Ejecuta el modelo ONNX en la Raspberry Pi para realizar la inferencia durante la detección del mildiu polvoso.

- **Librerías para la comunicación con Firebase**

Estas librerías permiten establecer la comunicación entre la Raspberry Pi, la estación de entrenamiento y la aplicación web mediante los servicios de Firebase.

Tabla 33

Librerías para la comunicación con Firebase

Librería	Función
firebase_admin	Inicializa el SDK de administración de Firebase.
credentials	Autentica el sistema mediante las credenciales del proyecto Firebase.
db	Permite leer y escribir información en Firebase Realtime Database.
storage	Gestiona la descarga y carga de archivos en Firebase Storage, como modelos ONNX y datasets.

- **Librerías para la sincronización del conjunto de datos**

Estas librerías permiten mantener actualizado el conjunto de imágenes utilizado para el entrenamiento del modelo.

Tabla 34

Librerías para sincronización del dataset

Librería	Función
json	Almacena y actualiza el catálogo local de imágenes sincronizadas.
hashlib	Genera firmas MD5 para comparar archivos y detectar cambios durante la sincronización diferencial.
os	Administra directorios y archivos utilizados durante la sincronización del dataset.

- **Librerías para la interfaz gráfica**

Estas librerías permiten desarrollar la aplicación ejecutada en la Raspberry Pi

Tabla 35

Librerías para la interfaz gráfica

Librería	Función
customtkinter	Construye la interfaz gráfica del sistema y permite la interacción del usuario con la aplicación.
Pillow (PIL)	Presenta las imágenes capturadas dentro de la interfaz gráfica.

- **Librerías para el monitoreo y administración del sistema**

Estas librerías permiten controlar recursos del sistema y gestionar procesos durante la ejecución.

Tabla 36

Librerías para la administración del sistema

Librería	Función
psutil	Monitorea el uso del procesador y otros recursos de la Raspberry Pi.
os	Gestiona archivos, directorios y procesos del sistema operativo.
sys	Controla la ejecución del programa y el acceso a funciones del sistema.
time	Implementa temporizadores y pausas durante la ejecución.
datetime	Genera marcas de tiempo para el registro de eventos y resultados.
glob	Localiza automáticamente archivos, como el modelo best.pt generado durante el entrenamiento.
atexit	Ejecuta funciones de limpieza antes de finalizar la aplicación.

- **Herramientas del sistema operativo para la Raspberry Pi**

Además de las librerías de Python, el funcionamiento de la cámara OV5647 requirió herramientas instaladas directamente en Raspberry Pi OS para permitir la comunicación con el hardware.

Tabla 37

Herramientas del sistema operativo para la cámara OV5647

Librería	Función

libcamera	Framework oficial de Raspberry Pi para controlar cámaras conectadas mediante el puerto CSI.
libcamera-apps	Proporciona utilidades para verificar el funcionamiento de la cámara y realizar pruebas de captura.
V4L2 (Video4Linux2)	Permite que OpenCV acceda a la cámara mediante cv2.VideoCapture().
driver bcm2835-v4l2	Driver del kernel para exponer la cámara como dispositivo /dev/video0.

7.4 Configuración del Sistema

En esta sección se describen los aspectos generales que el administrador debe verificar para garantizar el correcto funcionamiento del sistema. Se presentan las configuraciones básicas de la Raspberry Pi, los servicios de Firebase y el modelo de inteligencia artificial, las cuales permiten establecer la comunicación entre los diferentes componentes del sistema.

7.4.1 Configuración de la Raspberry Pi

Antes de ejecutar la aplicación, el administrador deberá verificar que la Raspberry Pi cuente con el sistema operativo Raspberry Pi OS de 64 bits correctamente instalado y actualizado. Asimismo, deberá comprobar que la cámara OV5647 se encuentre conectada al puerto CSI y sea reconocida por el sistema operativo.

Posteriormente, se debe confirmar que la Raspberry Pi tenga acceso a una red Wi-Fi con conexión a Internet, ya que la comunicación con Firebase Realtime Database y Firebase Storage se realiza de forma inalámbrica.

Antes de ejecutar la aplicación, es necesario crear un entorno virtual de Python con el fin de instalar las dependencias requeridas por el sistema de manera aislada. Una

vez creado y activado el entorno virtual, se deberán instalar las librerías correspondientes y ejecutar el archivo principal de la aplicación, como se muestra en la siguiente línea de comandos.

```
import os
import sys
import time
import atexit
import cv2
import numpy as np
from PIL import Image
import customtkinter as ctk
import psutil
import base64
import onnxruntime as ort
from mision2_firebase import FirebaseManager
```

La figura 76 muestra la ejecución de la aplicación en la Raspberry Pi 4. El administrador accede al directorio del proyecto, activa el entorno virtual (nexus_env) y ejecuta el archivo principal mision1_ui.py, iniciando la interfaz de detección y la comunicación con Firebase.

Figura 76

Ejecución de la aplicación en la Raspberry Pi4



Nota: *Autoría propia*

7.4.2 Configuración de Firebase

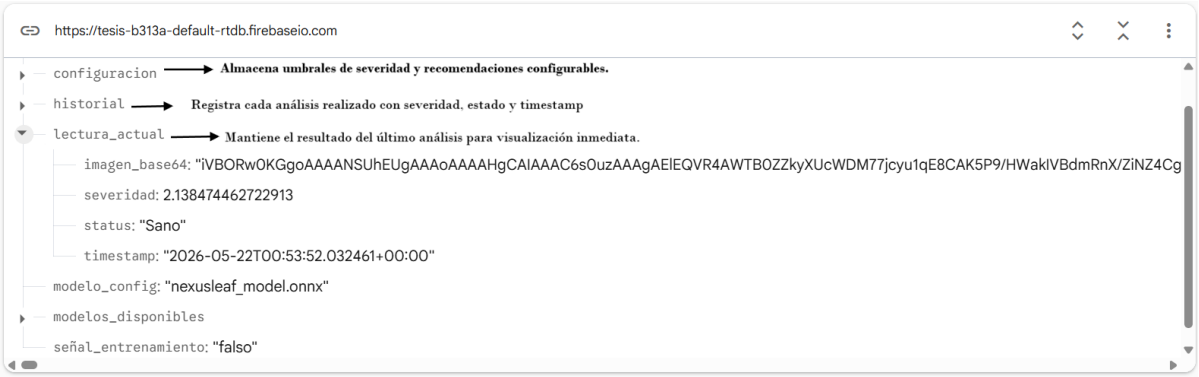
El administrador deberá verificar que el proyecto de Firebase se encuentre correctamente configurado y que el archivo de credenciales del servicio (firebase_credentials.json) esté ubicado dentro del directorio de la aplicación.

Además, se deberá comprobar la existencia de los nodos utilizados por el sistema dentro de Firebase Realtime Database, los cuales permiten almacenar el historial de análisis, la lectura actual, la configuración del sistema y la información relacionada con el modelo de inteligencia artificial. De igual manera, en Firebase Storage deberán existir las carpetas destinadas al almacenamiento del conjunto de datos y de los modelos entrenados en formato ONNX. En el archivo de conexión de firebase debe tener las siguientes líneas de comandos.

```
import os
import datetime
import firebase_admin
from firebase_admin import credentials, db, storage
```

En la figura 77 presenta la estructura general de la base de datos utilizada por el sistema. Se debe verificar que la estructura de Firebase Realtime Database contenga los nodos requeridos por el sistema, ya que estos permiten el intercambio de información entre los diferentes componentes de la arquitectura. Cada nodo cumple una función específica relacionada con la configuración del sistema

Figura 77
Realtime Database



Nota: *Autoría propia*

7.4.3 Configuración del modelo

El entrenamiento del modelo de detección utiliza un archivo de configuración en formato YAML, el cual define la ubicación de las imágenes empleadas durante las etapas de entrenamiento, validación y prueba, además del número de clases que serán reconocidas por el modelo YOLOv8.

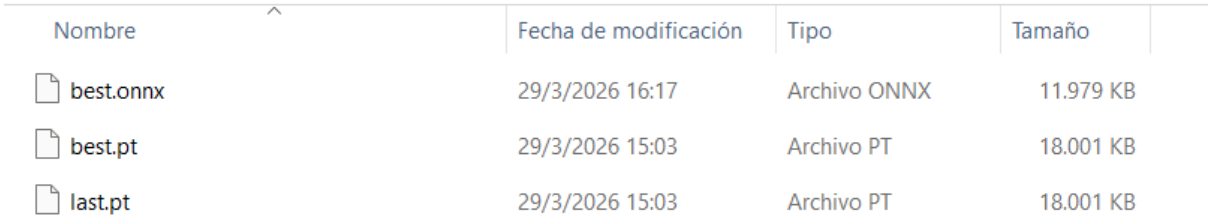
En el sistema desarrollado se establecieron dos clases correspondientes al estado fitosanitario de las hojas de mora de Castilla: enferma y sana. Una vez finalizado el entrenamiento, el sistema genera automáticamente el archivo best.pt, el cual es convertido al formato ONNX para posteriormente ser almacenado en Firebase Storage.

```
train: ./dataset/train/images
val: ./dataset/val/images
test: ./dataset/test/images

nc: 2
names: ['enferma', 'sana']
```

Una vez finalizado el proceso de entrenamiento, YOLOv8 genera automáticamente varios archivos que contienen los pesos del modelo entrenado. El archivo *best.pt* corresponde al modelo que obtuvo el mejor desempeño durante el entrenamiento y es utilizado como base para la exportación al formato ONNX. El archivo *last.pt* almacena los pesos correspondientes a la última época de entrenamiento y puede utilizarse para continuar el proceso si fuera necesario. Posteriormente, el archivo *best.pt* es convertido al formato *best.onnx*, el cual es compatible con la librería ONNX Runtime utilizada en la Raspberry Pi para realizar la inferencia en tiempo real. Finalmente, el modelo en formato ONNX es almacenado en Firebase Storage, desde donde la Raspberry Pi detecta automáticamente la nueva versión y la descarga para actualizar el sistema de detección. En la figura 78 se muestra archivos de modelo entrenado

Figura 78

Archivos de modelo entrenado



Nota: *Autoría propia*

7.5 Mantenimiento

Con el propósito de garantizar el funcionamiento continuo y la confiabilidad del sistema, se recomienda realizar actividades de mantenimiento preventivo de forma periódica. Semanalmente, el administrador deberá verificar que el servicio de la aplicación se encuentre en ejecución y limpiar la lente de la cámara OV5647 con un paño suave y seco para asegurar una adecuada calidad en la captura de imágenes.

Como parte del mantenimiento, el administrador también deberá revisar periódicamente la información almacenada en Firebase Realtime Database con el fin de verificar que los registros sean correctos y no existan datos corruptos que puedan afectar el funcionamiento de la aplicación. En caso de detectar registros erróneos, estos deberán ser eliminados para mantener la integridad de la base de datos y garantizar que el sistema continúe registrando correctamente las nuevas capturas y resultados del análisis.