



UNIVERSIDAD ESTATAL
PENÍNSULA DE
SANTA ELENA

FACSITEL

INGENIERÍA EN ELECTRÓNICA Y
AUTOMATIZACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR

Desarrollo de un rover con
sistemas Open Source y visión
artificial que identifique malezas
en parcelas agrícolas

CAÑARTE PITA ERICK BANNER

Dirigido por:
Ing. LUIS ENRIQUE CHUQUIMARCA, Ph.D.

La Libertad - 2026

DEDICATORIA

Dedico este trabajo, con profundo amor y gratitud, a mis padres y a mi familia, quienes han sido el pilar fundamental durante mi formación académica. Gracias por su esfuerzo, comprensión, consejos y apoyo incondicional en cada etapa de mi vida.

A todas las personas que confiaron en mí y me motivaron a continuar, incluso durante los momentos de dificultad. Sus palabras de aliento me permitieron mantener la constancia y culminar esta importante meta profesional.

Finalmente, dedico este logro a quienes forman parte de mi vida y han compartido conmigo este camino. Este trabajo representa no solo el resultado de mi esfuerzo, sino también el respaldo, la paciencia y el cariño de cada uno de ustedes.

AGRADECIMIENTO

Expreso mi más sincero agradecimiento a mi familia, especialmente a mis padres, por brindarme su apoyo incondicional, comprensión y motivación durante toda mi formación universitaria. Su esfuerzo y confianza fueron fundamentales para alcanzar esta meta.

A la Universidad Estatal Península de Santa Elena y a la carrera de Ingeniería en Electrónica y Automatización, por proporcionarme los conocimientos, recursos y espacios necesarios para mi preparación profesional.

De manera especial, agradezco al Ing. Luis Enrique Chuquimarca Jiménez, tutor de este trabajo, por su orientación, disposición, conocimientos y recomendaciones durante el desarrollo de la investigación. Su acompañamiento permitió fortalecer los aspectos técnicos y metodológicos del proyecto.

También agradezco a los docentes que contribuyeron a mi formación académica, así como a mis compañeros y amigos, quienes compartieron experiencias, conocimientos y palabras de ánimo a lo largo de este proceso.

Finalmente, agradezco a todas las personas que, de manera directa o indirecta, colaboraron para que el desarrollo y la culminación de este trabajo fueran posibles.

APROBACIÓN DEL TUTOR

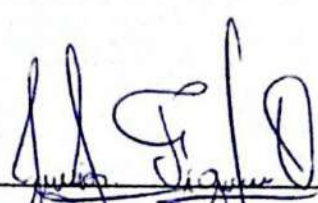
En mi calidad de tutor del trabajo de titulación denominado: **“Desarrollo de un rover con sistemas Open Source y visión artificial que identifique malezas en parcelas agrícolas”**, elaborado por el estudiante **Erick Banner Cañarte Pita**, de la carrera de **Ingeniería en Electrónica y Automatización** de la Universidad Estatal Península de Santa Elena, declaro que luego de haber orientado, estudiado y revisado, lo apruebo en todas sus partes y autorizo al estudiante que inicie los trámites legales correspondientes.

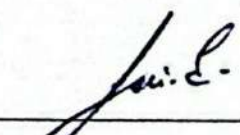


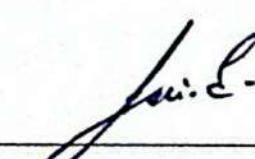
Ing. Luis Enrique Chuquimarca Jimenez, Ph.D.
Tutor

TRIBUNAL DE GRADO


Ing. Ronald Humberto Rovira Jurado, Ph.D.
DIRECTOR DE CARRERA


Ing. Junior Figueroa Olmedo, Mgtr.
DOCENTE ESPECIALISTA


Ing. Luis Enrique Chuquimarca Jimenez, Mgtr.
TUTOR


Ing. Luis Enrique Chuquimarca Jimenez, Mgtr.
DOCENTE GUÍA UIC


Ing. Corina Gonzabay, Mgtr.
SECRETARIA DEL TRIBUNAL

Resumen

El presente trabajo tuvo como objetivo desarrollar un rover basado en sistemas Open Source y visión artificial para identificar malezas en parcelas agrícolas. La solución integró el robot móvil Transbot-SE, una Raspberry Pi 5, una cámara USB con mecanismo PTZ, comunicación mediante una red Wi-Fi local y una interfaz web para el control remoto, la visualización de video y el almacenamiento de evidencias. Para el sistema de detección se conformó un conjunto de 952 imágenes capturadas en condiciones reales de campo, clasificadas en las categorías pimiento y maleza. Las imágenes fueron etiquetadas mediante cajas delimitadoras y distribuidas en 70 % para entrenamiento, 20 % para validación y 10 % para prueba. Se entrenó un modelo YOLO11n durante 100 épocas mediante aprendizaje por transferencia. El modelo alcanzó una precisión global de 0,7025, un recall de 0,6718 y un mAP@50 de 0,7132. En la Raspberry Pi 5, la inferencia se ejecutó a una velocidad media de 3,38 FPS. Durante las pruebas de campo se obtuvo una tasa global de acierto del 81,25 %, mientras que la validación funcional del prototipo alcanzó un cumplimiento del 90 %. Los resultados demuestran la viabilidad de integrar movilidad, procesamiento local y visión artificial en una plataforma agrícola basada en tecnologías accesibles y modificables; sin embargo, el desempeño disminuyó ante oclusiones, iluminación variable, movimiento de las hojas e inestabilidad de la conexión inalámbrica.

Palabras clave: rover agrícola, visión artificial, detección de malezas, YOLO11n, sistemas Open Source.

Abstract

This study aimed to develop a rover based on Open Source systems and artificial vision for weed identification in agricultural fields. The proposed solution integrated a Transbot-SE mobile robot, a Raspberry Pi 5, a USB camera with a PTZ mechanism, communication through a local Wi-Fi network, and a web interface for remote control, video monitoring, and evidence storage. For the detection system, a dataset containing 952 images captured under real field conditions was created and classified into two categories: pepper plants and weeds. The images were labeled using bounding boxes and divided into 70 % for training, 20 % for validation, and 10 % for testing. A YOLO11n model was trained for 100 epochs using transfer learning. The model achieved an overall precision of 0.7025, a recall of 0.6718, an mAP@50 of 0.7132, and an mAP@50–95 of 0.4235. When deployed on the Raspberry Pi 5, the model reached an average inference speed of 3.38 FPS. During field tests, the system obtained an overall accuracy rate of 81.25 %, while the functional validation of the prototype reached a compliance rate of 90 %. The results demonstrate the feasibility of integrating mobility, local processing, wireless communication, and artificial vision into an agricultural platform based on accessible and modifiable technologies. However, detection performance decreased under conditions involving occlusion, variable lighting, leaf movement, vegetation overlap, and wireless connection instability.

Keywords: agricultural rover, artificial vision, weed detection, YOLO11n, Open Source systems.

Índice

Índice de figuras	10
Índice de cuadros	12
1. Introducción	13
1.1. Justificación	14
1.2. Panorama actual	16
1.3. Objetivos	19
1.4. Objetivo general	19
1.5. Objetivos específicos	19
1.6. Fundamentos teóricos	20
1.6.1. Malezas en parcelas agrícolas	20
1.6.2. Competencia entre cultivos y malezas	21
1.6.3. Manejo de malezas	21
1.6.4. Agricultura de precisión	22
1.6.5. Robótica agrícola	23
1.6.6. Robótica móvil	25
1.6.7. Elementos de un robot móvil	25
1.6.8. Rover Transbot-SE	26
1.6.9. Sistemas Open Source	27
1.6.10. Sistemas embebidos y computación en el borde	28
1.6.11. Raspberry Pi 5	30
1.6.12. Visión artificial	30
1.6.13. Adquisición digital de imágenes	31
1.6.14. Inteligencia artificial, aprendizaje automático y aprendizaje profundo	32
1.6.15. Redes neuronales convolucionales	33
1.6.16. Clasificación, detección y segmentación	34
1.6.17. Arquitectura YOLO	35
1.6.18. Conjunto de datos	36
1.6.19. Etiquetado mediante cajas delimitadoras	37
1.6.20. División del conjunto de datos	38
1.6.21. Preprocesamiento y aumento de datos	38
1.6.22. Entrenamiento, épocas y sobreajuste	39
1.6.23. Métricas de evaluación	40
1.6.24. Intersección sobre unión	41
1.6.25. Precisión promedio y mAP	42
1.6.26. Matriz de confusión	42

1.6.27. Umbral de confianza	43
1.6.28. Comunicación inalámbrica	43
1.6.29. Interfaz web	43
1.6.30. Python y bibliotecas empleadas	44
1.7. Marco contextual	46
2. Métodos y diseño experimental	51
2.1. Métodos	51
2.2. Diseño experimental	54
2.2.1. Arquitectura general del sistema	55
2.2.2. Diseño de la plataforma robótica	56
2.2.3. Diseño del sistema de visión artificial	58
2.2.4. Diseño del entrenamiento y selección del modelo	64
2.2.5. Diseño del sistema de control	65
2.2.6. Diseño del sistema de comunicación	66
2.2.7. Diseño de la interfaz web	67
2.2.8. Integración del modelo en la Raspberry Pi	68
2.2.9. Gestión de evidencias	69
2.2.10. Protocolo de evaluación experimental	70
2.2.11. Criterios de seguridad y delimitación del experimento	73
2.3. Desarrollo experimental	74
2.3.1. Implementación de la planta	74
2.3.2. Implementación del sistema de alimentación	76
2.3.3. Preparación del entorno de ejecución	78
2.3.4. Implementación de la cámara USB	79
2.3.5. Implementación del sistema de control	80
2.3.6. Implementación del sistema de visión artificial	81
2.3.7. Implementación del sistema de comunicación	82
2.3.8. Implementación de la interfaz	83
2.3.9. Configuración del sistema de control	84
2.3.10. Configuración del sistema de comunicación	85
2.3.11. Configuración de la interfaz	85
2.3.12. Integración y verificación del sistema	86
2.3.13. Pruebas de funcionamiento	87
3. Resultados	89
3.1. Resultados del entrenamiento	89
3.2. Evaluación del modelo	91
3.3. Rendimiento de la inferencia en la Raspberry Pi 5	93
3.4. Resultados de las pruebas de campo	93
3.5. Validación funcional del prototipo	97

3.6. Discusión de los resultados	99
4. Conclusiones y recomendaciones	102
4.1. Conclusiones	102
4.2. Recomendaciones	104
Referencias	106
Anexos	112
Anexo A. Interfaz web final del sistema	112
Anexo B. Código de la interfaz web	113
Anexo C. Código de control del rover	133
Anexo D. Código de entrenamiento del modelo	139
Anexo E. Evidencias adicionales de detección	144
Anexo F. Certificado de Plagio	145

Índice de figuras

1.	Representación conceptual de la competencia entre el cultivo y las malezas. Fuente: Autor.	21
2.	Flujo general de información en una aplicación de agricultura de precisión. Fuente: Autor.	23
3.	Subsistemas funcionales considerados en el rover móvil. Fuente: Autor.	26
4.	Procesamiento en el borde aplicado al sistema de visión artificial del rover. Fuente: Autor.	28
5.	Etapas generales del sistema de visión artificial empleado en el rover. Fuente: Autor.	31
6.	Relación conceptual entre inteligencia artificial, aprendizaje automático y aprendizaje profundo. Fuente: Autor.	33
7.	Representación simplificada del procesamiento realizado por una red neuronal convolucional. Fuente: Autor.	34
8.	Comparación entre clasificación, detección de objetos y segmentación de imágenes. Fuente: Autor.	35
9.	Flujo simplificado de detección de objetos mediante YOLO. Fuente: Autor.	36
10.	Ejemplos conceptuales de las transformaciones empleadas para el aumento de datos. Fuente: Autor.	39
11.	Representación gráfica de la intersección sobre unión entre una caja predicha y una caja de referencia. Fuente: Autor.	42
12.	Ubicación general del escenario experimental en la provincia de Santa Elena. Fuente: Autor.	47
13.	Vista general de la parcela agrícola empleada como escenario experimental. Fuente: Autor.	49
14.	Flujo metodológico seguido para desarrollar y evaluar el sistema. Fuente: Autor.	52
15.	Arquitectura general del rover y del sistema de visión artificial. Fuente: Autor.	56
16.	Configuración física del rover utilizado en la investigación. Fuente: Autor.	58
17.	Adquisición y depuración de las imágenes utilizadas para conformar el conjunto de datos. Fuente: Autor.	59
18.	Etiquetado de plantas de pimiento y malezas mediante cajas delimitadoras. Fuente: Autor.	62
19.	Configuración de la versión del conjunto de datos utilizada durante el entrenamiento. Fuente: Autor.	64

20.	Arquitectura de comunicación entre el operador y el rover. Fuente: Autor.	67
21.	Interfaz web diseñada para el control y supervisión del rover. Fuente: Autor.	68
22.	Ejemplo de evidencia almacenada por el sistema durante las pruebas. Fuente: Autor.	70
23.	Rover Transbot-SE ensamblado para la adquisición y procesa- miento de imágenes agrícolas. Fuente: Autor.	76
24.	Representación esquemática de la batería utilizada en el rover y sus principales especificaciones técnicas. Fuente: Autor. . . .	77
25.	Cámara USB instalada sobre el mecanismo de orientación PTZ. Fuente: Autor.	80
26.	Acceso remoto y transferencia de archivos entre la computado- ra y la Raspberry Pi. Fuente: Autor.	83
27.	Interfaz web implementada para el control y la supervisión del rover. Fuente: Autor.	84
28.	Sistema integrado durante la preparación de las pruebas en la parcela agrícola. Fuente: Autor.	87
29.	Pruebas de funcionamiento del rover realizadas en la parcela agrícola. Fuente: Autor.	88
30.	Evolución de las pérdidas y métricas durante el entrenamiento del modelo. Fuente: Autor.	90
31.	Matriz de confusión obtenida durante la evaluación del mode- lo. Fuente: Autor.	91
32.	Curva precisión-recall obtenida para las clases evaluadas. Fuen- te: Autor.	92
33.	Ejemplos de detecciones correctas obtenidas durante las prue- bas de campo. Fuente: Autor.	94
34.	Errores representativos observados durante la identificación de plantas. Fuente: Autor.	95
35.	Interfaz web final utilizada para el control y la supervisión del rover. Fuente: Autor.	112
36.	Evidencias adicionales de detección obtenidas durante las prue- bas realizadas en la parcela agrícola. Fuente: Autor.	144

Índice de cuadros

1.	Comparación de métodos para la identificación de malezas . . .	18
2.	Aplicaciones principales de la robótica agrícola	24
3.	Herramientas de software utilizadas en el desarrollo del sistema	29
4.	Métricas utilizadas para evaluar el modelo de detección	40
5.	Caracterización general del escenario experimental	48
6.	Relación entre los métodos de investigación y su aplicación . .	51
7.	Subsistemas que conforman la arquitectura del rover	55
8.	Clases definidas para el conjunto de datos	60
9.	Distribución de anotaciones por clase	61
10.	Distribución del conjunto de datos	63
11.	Transformaciones aplicadas al conjunto de entrenamiento . . .	64
12.	Parámetros empleados durante el entrenamiento	65
13.	Acciones disponibles en el sistema de control	66
14.	Parámetros utilizados durante la inferencia	69
15.	Escenarios definidos para las pruebas de campo	71
16.	Factores e indicadores considerados durante las pruebas	72
17.	Configuración del entorno de ejecución del rover	78
18.	Resultados cuantitativos del modelo de detección	89
19.	Rendimiento de la inferencia ejecutada en la Raspberry Pi 5 .	93
20.	Resultados obtenidos en los escenarios de prueba de campo . .	96
21.	Validación funcional de los subsistemas implementados	98

1. Introducción

La producción agrícola constituye una actividad esencial para la seguridad alimentaria, el desarrollo rural y el abastecimiento de materias primas. Sin embargo, los sistemas de cultivo se encuentran expuestos a factores que limitan su productividad, entre los cuales se encuentra la presencia de malezas. Estas plantas se desarrollan en espacios destinados a los cultivos y compiten por recursos como agua, nutrientes, luz y superficie disponible. Un estudio de referencia sobre pérdidas agrícolas a escala mundial estimó que las malezas pueden producir una pérdida potencial cercana al 34 %, superior a las pérdidas potenciales atribuidas individualmente a patógenos y plagas animales [1].

La identificación de malezas es una etapa importante dentro de su manejo, debido a que permite reconocer su ubicación y diferenciarla del cultivo antes de aplicar medidas de control. En parcelas agrícolas, este procedimiento se realiza frecuentemente mediante inspección visual, por lo que su ejecución depende de la experiencia del operador, de las dimensiones del terreno y de las condiciones ambientales. Además, la similitud visual entre determinadas malezas y los cultivos, especialmente durante sus primeras fases de crecimiento, puede dificultar su diferenciación.

La automatización agrícola ha incorporado progresivamente sistemas de percepción, dispositivos digitales y plataformas robóticas para apoyar la ejecución de tareas de monitoreo y manejo de cultivos. La Organización de las Naciones Unidas para la Alimentación y la Agricultura señala que las tecnologías de automatización digital pueden contribuir a incrementar la productividad, fortalecer la resiliencia de los sistemas agrícolas y mejorar la eficiencia en el uso de los recursos; no obstante, su adopción también puede verse limitada por la conectividad, el acceso a energía, los costos y la disponibilidad de capacidades técnicas [2].

En este contexto, la visión artificial permite adquirir y procesar imágenes para reconocer características presentes en el entorno agrícola. Su integración con métodos de aprendizaje profundo ha impulsado el desarrollo de sistemas capaces de localizar cultivos y malezas mediante imágenes capturadas desde vehículos terrestres, maquinaria agrícola o vehículos aéreos. Las investigaciones recientes muestran que el aprendizaje profundo presenta ventajas para el reconocimiento vegetal, aunque su funcionamiento depende de la calidad y diversidad del conjunto de datos, las condiciones de iluminación, la presencia de oclusiones y la similitud morfológica entre las especies [3, 4].

Entre los métodos empleados para la detección de objetos se encuentra YOLO, acrónimo de *You Only Look Once*. Esta arquitectura plantea la de-

tección como un proceso unificado en el que una red neuronal analiza la imagen y predice directamente las clases y cajas delimitadoras de los objetos. Su diseño fue propuesto para equilibrar la velocidad de procesamiento y la capacidad de detección, por lo que resulta apropiado para aplicaciones que requieren analizar imágenes en tiempo real [7].

A partir de esta problemática, el presente trabajo desarrolla un rover con sistemas *Open Source* y visión artificial para identificar malezas en parcelas agrícolas. La propuesta utiliza el robot móvil Transbot-SE como plataforma de desplazamiento, una Raspberry Pi 5 como unidad de procesamiento, una cámara USB para la adquisición de imágenes y una interfaz web para el control remoto y la supervisión del sistema.

El rover es operado por un usuario mediante una conexión inalámbrica, mientras la cámara transmite imágenes del entorno. Sobre estas imágenes se ejecuta un modelo YOLO entrenado con dos clases: planta de pimiento, usada como cultivo de referencia durante la experimentación, y maleza. El sistema presenta las detecciones mediante cajas delimitadoras, etiquetas, niveles de confianza y un conteo por clase. También permite almacenar evidencias de las pruebas realizadas en campo.

El alcance de la propuesta se concentra en la identificación visual de malezas y plantas de cultivo. Por tanto, el proyecto no contempla la eliminación física o química de las malezas, la navegación autónoma, la generación de mapas ni la planificación automática de trayectorias. La investigación busca comprobar la viabilidad de integrar una plataforma robótica, herramientas de código abierto y un modelo de visión artificial en condiciones agrícolas reales.

1.1. Justificación

Las malezas representan una dificultad recurrente en la producción agrícola porque interfieren con el crecimiento de los cultivos y pueden ocasionar pérdidas de rendimiento. Su manejo requiere reconocer oportunamente las áreas afectadas y distinguir las plantas no deseadas del cultivo establecido. La magnitud de las pérdidas potenciales atribuidas a las malezas demuestra la importancia de desarrollar métodos que ayuden a mejorar su observación y localización dentro de las parcelas [1].

La inspección manual permite al agricultor evaluar directamente el estado de una parcela, pero puede demandar una cantidad considerable de tiempo cuando el área de trabajo es extensa o contiene vegetación abundante. A esto se suma que las plantas se encuentran sometidas a iluminación variable, sombras, superposición de hojas, movimiento producido por el viento y diferentes etapas de crecimiento. Estas condiciones dificultan la adquisición

de imágenes uniformes y la diferenciación automática entre cultivo y maleza. La literatura identifica la similitud visual, las oclusiones y los cambios de iluminación como algunos de los principales retos de los sistemas de detección agrícola [3].

La inteligencia artificial puede apoyar esta actividad mediante modelos entrenados para reconocer patrones visuales presentes en imágenes de campo. Su incorporación no sustituye el criterio agronómico ni ejecuta por sí misma el control de las malezas, pero puede funcionar como una herramienta de asistencia para localizar vegetación de interés y generar información que facilite posteriores decisiones de manejo. Las investigaciones sobre inteligencia artificial aplicada al manejo de malezas destacan la importancia conjunta de la tecnología de captura, los conjuntos de entrenamiento, los modelos empleados y las métricas utilizadas para evaluar su funcionamiento [4].

La utilización de un rover terrestre permite aproximar la cámara a la vegetación y obtener imágenes desde una perspectiva cercana al suelo. Las plataformas robóticas terrestres han adquirido relevancia en el manejo específico de malezas porque pueden integrar cámaras, sistemas de procesamiento y mecanismos de desplazamiento dentro de una misma arquitectura. No obstante, continúan existiendo desafíos relacionados con la identificación precisa, el costo de implementación y el funcionamiento robusto en tiempo real bajo condiciones de campo [5].

El empleo de sistemas *Open Source* se justifica por la posibilidad de estudiar, adaptar y reutilizar las herramientas de software que forman parte de la propuesta. La definición de código abierto no se limita al acceso al código fuente, sino que también considera las condiciones que permiten su redistribución, modificación y creación de trabajos derivados [17]. En el presente proyecto se utilizaron herramientas como Python, OpenCV, Flask y bibliotecas compatibles con YOLO, lo que permitió desarrollar una solución ajustada a los requerimientos del rover sin depender de una plataforma de software cerrada.

La investigación posee una utilidad práctica porque integra en una sola plataforma el desplazamiento controlado, la transmisión de video, la detección de objetos y el almacenamiento de evidencias. También presenta una utilidad académica, debido a que documenta el proceso completo de adquisición de imágenes, etiquetado, entrenamiento, validación, despliegue del modelo y evaluación en campo.

Finalmente, el trabajo puede servir como base para desarrollos posteriores orientados al monitoreo agrícola, ampliación del número de cultivos, clasificación de especies de malezas o implementación de mecanismos de control selectivo. La propuesta no pretende resolver de manera definitiva el manejo de malezas, sino demostrar el funcionamiento y las limitaciones de un prototipo

construido con tecnologías accesibles y modificables.

1.2. Panorama actual

La automatización agrícola ha evolucionado desde la mecanización convencional hacia el uso de tecnologías digitales capaces de recopilar, procesar y utilizar información del entorno. La FAO identifica dentro de esta evolución a los sistemas de guiado, sensores, robots, inteligencia artificial y tecnologías de automatización digital. Estas herramientas pueden mejorar la precisión de las operaciones agrícolas, aunque su implementación debe adaptarse a las condiciones económicas, técnicas y productivas de cada territorio [2].

La agricultura de precisión emplea información específica de la parcela para ejecutar actividades de forma localizada. Dentro de este enfoque, la identificación de malezas permite diferenciar zonas con vegetación no deseada y constituye una etapa previa para estrategias de manejo específico. En comparación con una intervención uniforme sobre toda la superficie, el conocimiento de la ubicación de las malezas puede facilitar acciones dirigidas y un uso más racional de los recursos.

Las plataformas robóticas terrestres representan una línea activa de investigación para el manejo específico de malezas. Una revisión publicada en *Computers and Electronics in Agriculture* analizó tecnologías de robots terrestres, sensores de imagen y estrategias de manejo utilizadas en agricultura de precisión. El estudio concluyó que el aprendizaje profundo presenta un potencial significativo para mejorar la identificación de malezas, pero también señaló que una parte de las investigaciones permanece limitada a simulaciones u optimización de modelos, sin una validación amplia en condiciones reales de campo [5].

En el campo de la visión artificial se han utilizado técnicas de procesamiento de color, extracción manual de características, aprendizaje automático y redes neuronales profundas. Los métodos tradicionales requieren definir previamente características como forma, textura o color. En cambio, los modelos de aprendizaje profundo pueden aprender representaciones directamente a partir de las imágenes. Las revisiones recientes señalan que estos modelos generalmente alcanzan un mejor desempeño, pero necesitan conjuntos de datos representativos y recursos de procesamiento superiores [3].

La detección de objetos es especialmente útil para este tipo de aplicación, debido a que no solo asigna una clase general a la imagen, sino que localiza cada objeto mediante una caja delimitadora. YOLO se convirtió en una de las familias de detectores más utilizadas en aplicaciones de tiempo real por realizar la predicción de clases y coordenadas dentro de una arquitectura

unificada [7]. Esta característica permite aplicar el modelo sobre secuencias de video y visualizar las detecciones durante la operación del rover.

A pesar de los avances alcanzados, la detección de malezas continúa siendo un problema complejo. Los cultivos y las malezas pueden compartir colores, formas y texturas; además, la vegetación suele aparecer parcialmente oculta y bajo condiciones de iluminación no controladas. Estas dificultades se incrementan cuando las plantas son pequeñas, se encuentran agrupadas o son afectadas por el movimiento del viento. Por esta razón, el desempeño de un modelo entrenado en un conjunto limitado de imágenes puede disminuir al enfrentarse a escenarios diferentes de los incluidos durante el entrenamiento [3, 4].

En Ecuador, la Encuesta de Superficie y Producción Agropecuaria Continua reportó para 2024 aproximadamente 4,8 millones de hectáreas bajo labor agropecuaria, considerando cultivos permanentes, transitorios, pastos cultivados y pastos naturales [28]. Esta extensión evidencia la importancia territorial del sector y la necesidad de continuar investigando herramientas tecnológicas adaptadas a las condiciones productivas del país.

El Ministerio de Agricultura y Ganadería del Ecuador también ha reconocido la importancia de la tecnología y la conectividad para impulsar la agricultura de precisión y reducir brechas tecnológicas en el medio rural [30]. Sin embargo, la adopción de sistemas avanzados puede encontrar restricciones relacionadas con conectividad, energía, capacitación y capacidad de inversión, factores que también han sido identificados por la FAO para productores de menor escala [2].

En respuesta a este panorama, la presente investigación implementa y evalúa un prototipo de rover controlado remotamente, equipado con una cámara y un modelo de detección de objetos. Su aporte se concentra en la integración y validación experimental de tecnologías disponibles en un entorno agrícola real, documentando tanto los resultados favorables como las limitaciones encontradas durante la identificación de malezas.

Las tecnologías utilizadas para la identificación de malezas presentan diferencias en cuanto a cobertura, costo, nivel de detalle y condiciones de operación. En la Tabla 1 se resumen las principales ventajas y limitaciones de cuatro métodos empleados para observar o detectar malezas en parcelas agrícolas.

Tabla 1: Comparación de métodos para la identificación de malezas

Método	Ventajas	Limitaciones
Inspección visual manual	Permite evaluar directamente las plantas mediante la experiencia y el criterio del operador. No requiere sistemas electrónicos especializados.	Demanda tiempo y personal, especialmente en parcelas extensas. El resultado puede depender de la experiencia del evaluador.
Sensores espectrales	Permiten analizar características de la vegetación en diferentes regiones del espectro electromagnético y detectar diferencias que pueden no ser evidentes a simple vista.	Pueden requerir equipos especializados, procedimientos de calibración y una mayor capacidad de procesamiento.
Vehículos aéreos no tripulados	Permiten adquirir imágenes de áreas extensas en periodos reducidos y observar la distribución espacial de la vegetación.	La altura de captura, la resolución espacial y la superposición de las plantas pueden limitar la identificación de malezas pequeñas.
Rover terrestre con visión artificial	Permite adquirir imágenes próximas al suelo, observar detalles de la vegetación y procesar las imágenes durante el recorrido por la parcela.	Su desempeño depende de la iluminación, el movimiento de las plantas, la calidad del conjunto de datos y la capacidad de procesamiento disponible.

Fuente: Elaboración propia a partir de [5, 4, 3].

Como se observa en la Tabla 1, los vehículos aéreos permiten cubrir superficies amplias, mientras que los robots terrestres pueden capturar imágenes desde una menor distancia respecto a la vegetación. Esta proximidad favorece la observación de detalles, aunque el funcionamiento de la detección continúa condicionado por la iluminación, la similitud visual entre cultivo y maleza, las oclusiones y la diversidad del conjunto de entrenamiento [5, 3].

1.3. Objetivos

1.4. Objetivo general

Desarrollar un rover mediante sistemas Open Source y visión artificial para la identificación de malezas en parcelas agrícolas.

1.5. Objetivos específicos

- Implementar un rover que pueda movilizarse en campo agrónomo para recorrer sus cultivos.
- Desarrollar un sistema de visión artificial para la captura de imágenes en parcelas agrícolas.
- Implementar un sistema de procesamiento que permita identificar malezas mediante el análisis de imágenes en tiempo real.
- Realizar pruebas experimentales para comprobar el funcionamiento del sistema desarrollado.

1.6. Fundamentos teóricos

En esta sección se presentan los conceptos necesarios para comprender el desarrollo del rover propuesto. La fundamentación se organiza desde el problema agrícola asociado a la presencia de malezas hasta las tecnologías empleadas para adquirir imágenes, entrenar un modelo de detección de objetos e integrarlo en una plataforma robótica móvil. En consecuencia, se abordan la agricultura de precisión, el manejo de malezas, la robótica agrícola, los sistemas *Open Source*, la visión artificial, el aprendizaje profundo, la arquitectura YOLO y las métricas utilizadas para evaluar el modelo.

1.6.1. Malezas en parcelas agrícolas

El término maleza se utiliza para describir una planta que crece en un lugar donde resulta indeseable para los objetivos productivos establecidos. En el ámbito agrícola, esta denominación no depende exclusivamente de la especie, sino de su presencia dentro de un espacio destinado a un cultivo. Una planta puede considerarse útil en determinado contexto y convertirse en maleza cuando interfiere con el desarrollo de una especie cultivada.

Las malezas afectan la producción porque compiten con los cultivos por recursos limitados como agua, nutrientes, radiación solar y espacio. También pueden dificultar las labores agrícolas, interferir con la cosecha, servir como hospedantes de organismos perjudiciales y aumentar los costos de manejo. Oerke estimó que, en ausencia de medidas de protección, las malezas podían ocasionar pérdidas potenciales cercanas al 34 % en los principales cultivos analizados, lo que demuestra su relevancia dentro de los factores que limitan la productividad agrícola [1].

La distribución de las malezas dentro de una parcela no suele ser uniforme. Pueden presentarse de forma aislada, agrupada o mezclada con las plantas cultivadas. Esta variabilidad hace necesario observar el terreno y reconocer las zonas donde existe vegetación no deseada antes de aplicar una estrategia de control. Por ello, la identificación constituye una etapa previa al manejo manual, mecánico, químico o tecnológico de las malezas.

En el presente proyecto no se realiza la eliminación de la maleza. El alcance del sistema se limita a su identificación visual mediante cajas delimitadoras y etiquetas generadas por un modelo de detección de objetos. La información producida puede utilizarse como apoyo para posteriores decisiones o para futuras investigaciones sobre control localizado.

1.6.2. Competencia entre cultivos y malezas

La competencia vegetal ocurre cuando dos o más plantas requieren simultáneamente un recurso cuya disponibilidad resulta insuficiente para cubrir completamente sus necesidades. En una parcela agrícola, las malezas pueden emerger junto al cultivo y competir desde etapas tempranas de crecimiento. La intensidad de esta interacción depende de factores como la densidad de las plantas, las especies presentes, la disponibilidad de agua, la fertilidad del suelo, el momento de emergencia y las condiciones ambientales.

El efecto de la competencia no se limita a la reducción del rendimiento. También puede modificar el tamaño, vigor y distribución espacial del cultivo. Esta situación justifica el desarrollo de métodos que ayuden a localizar tempranamente la vegetación no deseada. Sin embargo, la identificación visual automática resulta compleja cuando el cultivo y la maleza presentan colores, formas o tamaños similares, especialmente durante las primeras etapas de desarrollo [3].

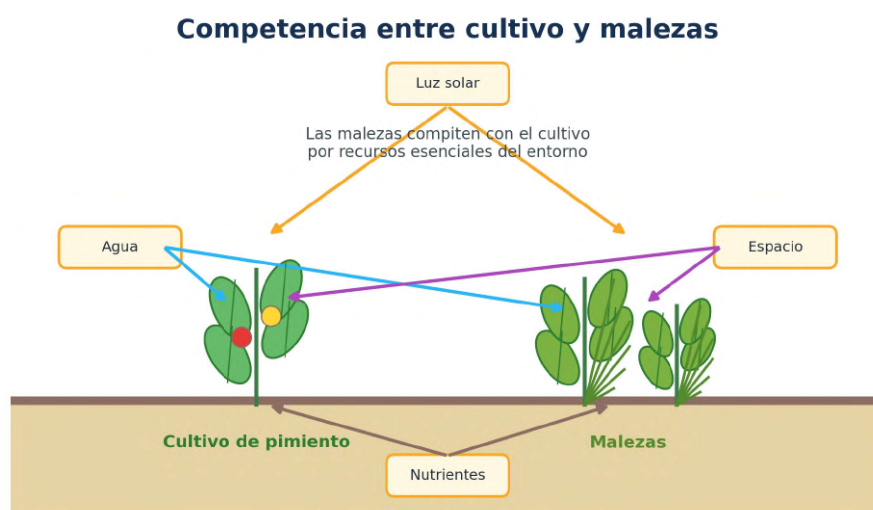


Figura 1: Representación conceptual de la competencia entre el cultivo y las malezas. Fuente: Autor.

1.6.3. Manejo de malezas

El manejo de malezas comprende el conjunto de acciones utilizadas para reducir su interferencia sobre la producción agrícola. Los métodos convencionales incluyen la remoción manual, el control mecánico, el empleo de coberturas, la rotación de cultivos y la aplicación de herbicidas. La selección de un método depende del cultivo, las especies de maleza, la extensión del terreno,

la disponibilidad de equipos y los recursos económicos.

La aplicación uniforme de una medida de control sobre toda la parcela puede resultar innecesaria cuando la maleza se concentra únicamente en sectores determinados. Por esta razón, el manejo específico por sitio busca reconocer la variabilidad espacial y actuar solamente sobre las áreas que lo requieren. Los sistemas de inteligencia artificial pueden contribuir a este proceso mediante el análisis de imágenes y la localización de plantas de interés, aunque su desempeño depende de la tecnología de captura, el conjunto de datos, el modelo seleccionado y las métricas utilizadas para validarlo [4].

La revisión desarrollada por Vasileiou et al. también señala que la inteligencia artificial aplicada al manejo de malezas debe analizarse junto con otras tecnologías, como robots de campo, sistemas de Internet de las cosas, vehículos aéreos y mecanismos de aplicación selectiva. En este proyecto se estudia específicamente la etapa de percepción visual instalada sobre un rover terrestre, sin implementar mecanismos de aplicación o eliminación [4].

1.6.4. Agricultura de precisión

La agricultura de precisión es una estrategia de gestión que observa, mide y responde a la variabilidad espacial y temporal presente dentro de una explotación agrícola. Su propósito es utilizar información específica del terreno para mejorar la toma de decisiones y ejecutar las actividades en el lugar y momento adecuados. Esta estrategia puede integrar sensores, cámaras, sistemas de posicionamiento, comunicaciones, maquinaria, plataformas robóticas y programas de análisis [6].

La automatización digital puede contribuir al aumento de la productividad, la eficiencia en el uso de recursos y la resiliencia de los sistemas agrícolas. No obstante, la FAO advierte que sus beneficios dependen del contexto y que persisten limitaciones asociadas con el costo, la conectividad, la disponibilidad de energía, la capacitación y la infraestructura rural [2].

Dentro de la agricultura de precisión, la detección de malezas permite obtener información localizada sobre la presencia de vegetación no deseada. A diferencia de una clasificación general de la imagen, la detección de objetos permite indicar dónde se encuentra cada planta reconocida. Esta característica resulta útil para sistemas de monitoreo, elaboración de mapas de infestación o futuras aplicaciones selectivas.

El rover desarrollado se relaciona con este enfoque porque captura imágenes desde una perspectiva próxima al suelo y procesa la información visual mientras se desplaza por la parcela bajo control del usuario. El sistema no constituye una solución integral de agricultura de precisión, pero demuestra la integración de percepción, procesamiento y movilidad en una aplicación

agrícola experimental.

Flujo general de una aplicación de agricultura de precisión

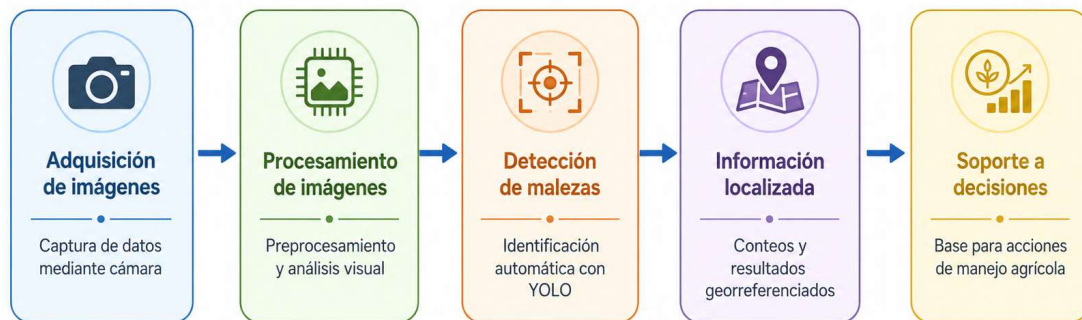


Figura 2: Flujo general de información en una aplicación de agricultura de precisión. Fuente: Autor.

1.6.5. Robótica agrícola

La robótica agrícola comprende el desarrollo y aplicación de sistemas robóticos destinados a ejecutar o asistir actividades relacionadas con la producción agropecuaria. Entre sus aplicaciones se encuentran el monitoreo de cultivos, la recolección de información, la cosecha, la pulverización selectiva, la detección de enfermedades y el manejo de malezas.

Las plataformas robóticas terrestres presentan la ventaja de desplazarse cerca de las plantas y transportar cámaras, sensores, unidades de procesamiento y mecanismos de actuación. Una revisión de 91 trabajos sobre manejo específico de malezas encontró que los robots terrestres poseen un potencial importante para integrar sensores de imagen y técnicas de aprendizaje profundo. Sin embargo, permanecen desafíos relacionados con la identificación precisa, el costo inicial, la integración entre componentes y la operación robusta en tiempo real [5].

Las investigaciones en este campo suelen concentrarse en navegación autónoma, percepción del cultivo o mecanismos de eliminación. En la presente propuesta, el rover se controla remotamente y se emplea como plataforma

móvil para transportar la cámara y el sistema de visión artificial. Esta delimitación reduce la complejidad de la navegación y permite concentrar la investigación en la adquisición de imágenes y la identificación de malezas.

La robótica agrícola comprende diferentes aplicaciones orientadas a la observación, análisis e intervención sobre los cultivos. Estas aplicaciones se distinguen por los sensores, sistemas de procesamiento y mecanismos de actuación que incorporan. En la Tabla 2 se presentan algunas de las principales aplicaciones de estas plataformas y su relación con el sistema desarrollado.

Tabla 2: Aplicaciones principales de la robótica agrícola

Aplicación	Tecnologías empleadas	Finalidad	Relación con esta investigación
Monitoreo de cultivos	Cámaras, sensores ambientales, sistemas de posicionamiento y plataformas móviles.	Observar las condiciones de las plantas y recopilar información del terreno.	El rover permite transmitir imágenes de la parcela y supervisar visualmente la vegetación durante su desplazamiento.
Identificación de malezas	Cámaras RGB o multispectrales, visión artificial y modelos de aprendizaje profundo.	Diferenciar la vegetación no deseada de las plantas pertenecientes al cultivo.	Constituye la función principal del proyecto mediante un modelo YOLO entrenado con las clases pimiento y maleza.
Control localizado de malezas	Robots terrestres, sistemas de detección y mecanismos mecánicos, térmicos o de aplicación selectiva.	Intervenir únicamente en las plantas o sectores donde se identifica vegetación no deseada.	No forma parte del alcance actual, pero representa una posible ampliación del sistema desarrollado.
Aplicación selectiva de productos	Sistemas de visión, boquillas controladas, actuadores y unidades de procesamiento.	Reducir la aplicación uniforme de productos y actuar en zonas específicas.	El rover no aplica productos; únicamente proporciona información visual que podría utilizarse en una etapa futura.
Cosecha robotizada	Cámaras, modelos de detección, manipuladores y efectores finales.	Localizar productos agrícolas, estimar su estado y realizar su recolección.	No pertenece al alcance del proyecto, pero constituye otra aplicación representativa de la visión artificial en agricultura.

Fuente: Elaboración propia a partir de [2, 5].

Como se observa en la Tabla 2, una plataforma robótica agrícola puede utilizarse únicamente para adquirir información o incorporar mecanismos destinados a intervenir sobre el cultivo. La presente investigación se concentra en la primera etapa del proceso: la adquisición de imágenes y la identificación visual de plantas. Por tanto, el rover no ejecuta la eliminación de las malezas ni la aplicación de productos agrícolas [2, 5].

1.6.6. Robótica móvil

La robótica móvil estudia sistemas capaces de desplazarse dentro de un entorno para realizar una tarea determinada. A diferencia de los robots industriales instalados en una posición fija, un robot móvil debe integrar una estructura de desplazamiento, actuadores, percepción, procesamiento, alimentación y comunicación [13].

El desplazamiento puede ser autónomo o depender de un operador. En este proyecto, el Transbot-SE funciona mediante control remoto a través de una interfaz web. Por tanto, no se considera un sistema de navegación autónoma: las decisiones de dirección y desplazamiento son emitidas por el usuario, mientras que el procesamiento de imágenes se ejecuta en la Raspberry Pi.

La tesis utilizada como referencia institucional también emplea el Transbot-SE, aunque su propósito se orienta a la planificación de trayectorias y generación de mapas. Dicho antecedente confirma la utilidad de la plataforma para proyectos de integración, pero el presente trabajo modifica su finalidad hacia la detección de malezas mediante visión artificial [14].

1.6.7. Elementos de un robot móvil

Un robot móvil está formado por subsistemas que trabajan de manera coordinada. Para esta investigación se consideran los siguientes elementos:

- **Sistema mecánico:** corresponde al chasis, estructura, soportes, orugas y elementos que sostienen y protegen los componentes.
- **Sistema de actuación:** incluye los motores encargados del desplazamiento y los servomotores que modifican la orientación de la cámara.
- **Sistema de percepción:** está constituido principalmente por la cámara USB, utilizada para capturar las imágenes de la parcela.
- **Sistema de procesamiento y control:** comprende la Raspberry Pi 5, la placa de expansión y los programas que administran el movimiento, el video y la detección.
- **Sistema de comunicación:** permite intercambiar comandos y video entre el rover y el computador del operador mediante una red inalámbrica.
- **Sistema de alimentación:** proporciona la energía necesaria para el controlador, los motores, servomotores, cámara y demás dispositivos.

La integración correcta de estos subsistemas permite que la plataforma se desplace, transmita imágenes, ejecute el modelo y presente los resultados al usuario. En este trabajo no se desarrolla un análisis cinemático o dinámico, debido a que el movimiento se realiza mediante las funciones proporcionadas por la biblioteca del fabricante.

Interacción de los subsistemas del rover móvil

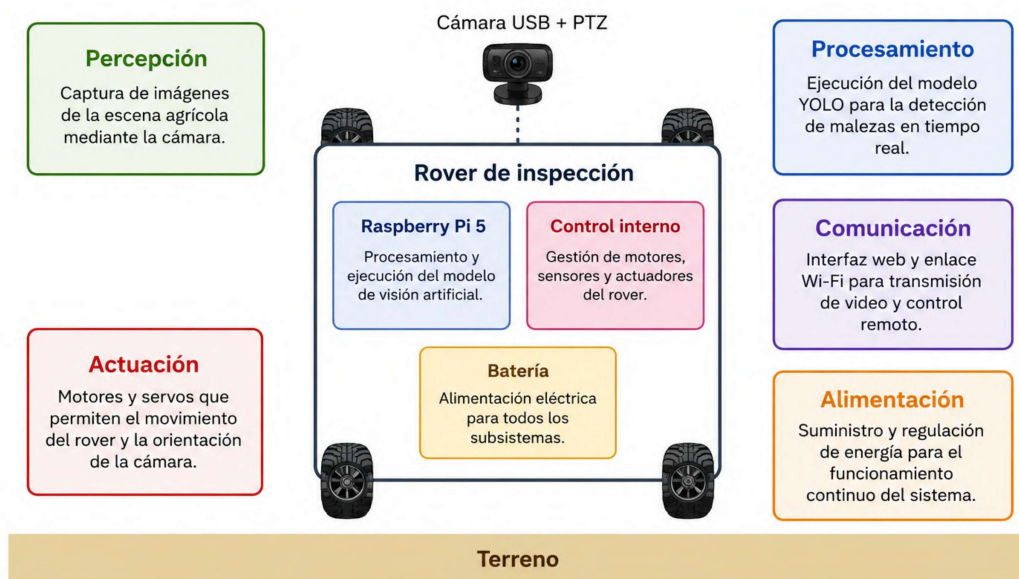


Figura 3: Subsistemas funcionales considerados en el rover móvil. Fuente: Autor.

1.6.8. Rover Transbot-SE

El Transbot-SE es una plataforma robótica terrestre con estructura de orugas, cuerpo de aleación de aluminio, motores con codificadores y soporte para una cámara orientable. El fabricante indica compatibilidad con Raspberry Pi 5 y programación mediante Python. Su arquitectura admite control de movimiento, transmisión de video y aplicaciones de visión artificial [15].

La estructura de orugas proporciona una superficie de contacto mayor que la de una rueda convencional y facilita el desplazamiento sobre terrenos con pequeñas irregularidades. No obstante, su operación en parcelas continúa dependiendo de factores como la humedad, la presencia de piedras, la altura de la vegetación y la capacidad de la batería.

Para esta investigación se utilizaron principalmente:

- la base móvil y sus motores;
- la placa de expansión;
- la Raspberry Pi 5;
- el soporte de cámara de dos grados de libertad;
- una cámara USB;
- el sistema de alimentación; y
- la biblioteca de control Transbot.

Aunque la plataforma dispone de capacidades adicionales, no se utilizaron LiDAR, brazo robótico, planificación de trayectorias, SLAM ni navegación automática. El operador controla la dirección del rover y la orientación de la cámara desde la página web.

1.6.9. Sistemas Open Source

La Open Source Initiative establece que el código abierto no consiste únicamente en permitir la visualización del código fuente. Una licencia *Open Source* también debe permitir, bajo las condiciones definidas por dicha licencia, su redistribución, modificación y utilización para crear trabajos derivados [17].

El empleo de herramientas de código abierto ofrece ventajas para proyectos académicos y prototipos tecnológicos, debido a que permite revisar el funcionamiento del programa, adaptar módulos, reutilizar bibliotecas, documentar cambios y reproducir el sistema. También reduce la dependencia de entornos completamente propietarios.

En esta investigación, el concepto *Open Source* se aplica principalmente al conjunto de herramientas de software empleadas para desarrollar el sistema. Entre ellas se encuentran Python, OpenCV, Flask y varias bibliotecas ejecutadas sobre Linux. Esto no significa que todos los elementos físicos del rover sean hardware abierto ni que todos los servicios utilizados posean la misma licencia.

Esta precisión es importante porque el proyecto combina herramientas de código abierto con dispositivos comerciales y servicios externos. La apertura del software facilita su estudio y modificación, mientras que las características físicas del robot dependen de las especificaciones y restricciones establecidas por sus fabricantes.

1.6.10. Sistemas embebidos y computación en el borde

Un sistema embebido integra hardware y software para cumplir funciones específicas dentro de un dispositivo. En aplicaciones de visión artificial, la unidad de procesamiento recibe las imágenes, ejecuta los algoritmos y genera una respuesta relacionada con la tarea programada.

Cuando el procesamiento se realiza cerca de la fuente que produce los datos se utiliza el concepto de computación en el borde o *edge computing*. En el rover, las imágenes no necesitan enviarse a un servidor externo para obtener cada detección, debido a que el modelo se ejecuta localmente en la Raspberry Pi. Esto favorece el funcionamiento en zonas donde el acceso a Internet es limitado, aunque sigue siendo necesaria una red local para controlar el rover desde la computadora.

La ejecución local también introduce restricciones. El modelo debe adaptarse a la memoria disponible, la capacidad de procesamiento, el consumo de energía y la temperatura del dispositivo. Por esta razón, se seleccionó un modelo de tamaño reducido y se ajustó la resolución utilizada durante la inferencia.

Computación en el borde aplicada al rover

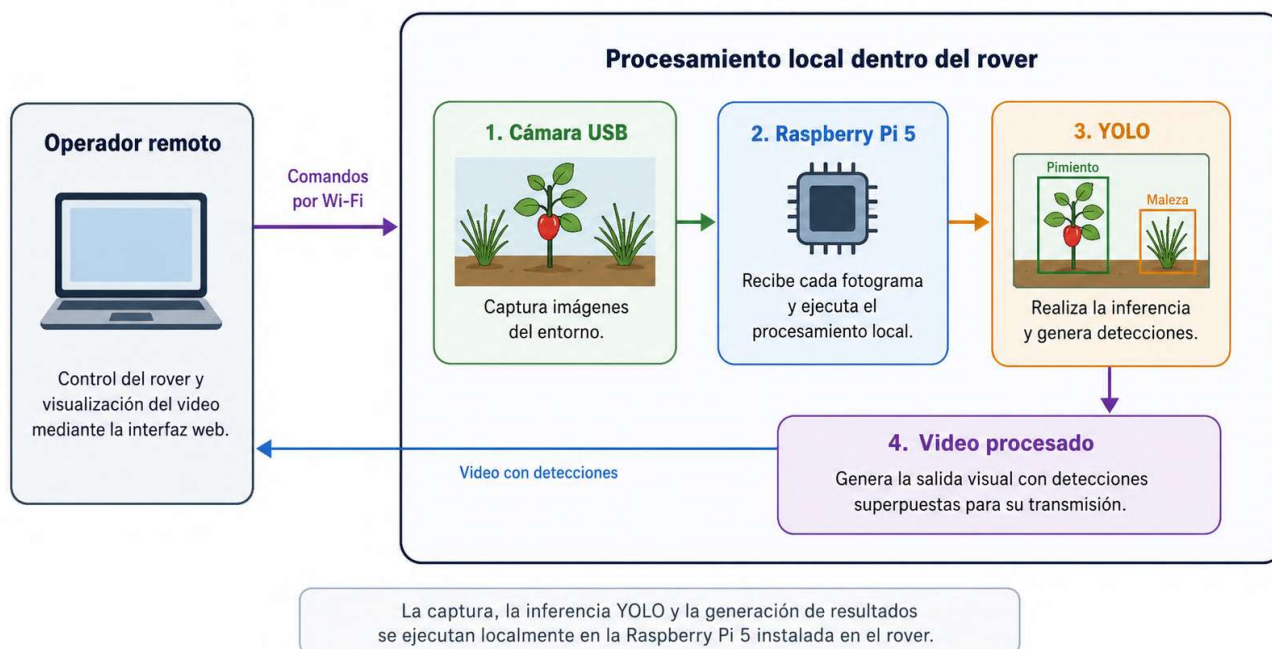


Figura 4: Procesamiento en el borde aplicado al sistema de visión artificial del rover. Fuente: Autor.

El sistema fue desarrollado mediante diferentes herramientas de software que permiten revisar, modificar y reutilizar el código bajo las condiciones de sus respectivas licencias. En la Tabla 3 se presentan las principales herramientas utilizadas, su función dentro del proyecto y la licencia declarada por cada organización responsable.

Tabla 3: Herramientas de software utilizadas en el desarrollo del sistema

Herramienta	Tipo	Función dentro del proyecto	Licencia o modalidad
Python	Lenguaje de programación	Integra el control del rover, el procesamiento de imágenes, la inferencia del modelo y el funcionamiento del servidor web.	Python Software Foundation License.
OpenCV	Biblioteca de visión artificial	Permite adquirir los fotogramas de la cámara USB, modificar su tamaño, dibujar cajas y etiquetas, y codificar el flujo de video.	Apache License 2.0 para OpenCV 4.5.0 y versiones posteriores.
Flask	Framework para aplicaciones web	Administra las rutas del servidor, la transmisión de video y las solicitudes de movimiento, orientación de cámara y almacenamiento de evidencias.	BSD de tres cláusulas.
Ultralytics	Biblioteca de aprendizaje profundo	Permite cargar el archivo de pesos entrenado y ejecutar la detección de objetos mediante la arquitectura YOLO.	AGPL-3.0 para su distribución abierta; también dispone de licencia empresarial.
NumPy	Biblioteca de cálculo numérico	Permite manipular matrices y estructuras numéricas utilizadas en el procesamiento de imágenes y resultados.	BSD de tres cláusulas.
PySerial	Biblioteca de comunicación serial	Proporciona funciones para establecer y administrar la comunicación serial entre Python y los dispositivos de control del rover.	BSD de tres cláusulas.

Fuente: Elaboración propia a partir de la documentación oficial de Python, OpenCV, Flask, Ultralytics, NumPy y PySerial [18, 19, 20, 21, 22, 23].

La Tabla 3 evidencia que la implementación combina herramientas de software que permiten estudiar y modificar el código bajo los términos de sus respectivas licencias. Sin embargo, la expresión *Open Source* no implica que todas las herramientas compartan la misma licencia ni que puedan utilizarse sin condiciones. En particular, la licencia AGPL-3.0 de Ultralytics establece obligaciones distintas de las licencias permisivas utilizadas por Python, OpenCV, Flask y NumPy [21, 19, 20].

1.6.11. Raspberry Pi 5

La Raspberry Pi 5 es una computadora de placa única utilizada en proyectos de automatización, sistemas embebidos, educación y procesamiento de datos. La documentación del fabricante indica que incorpora un procesador Broadcom BCM2712 de cuatro núcleos Arm Cortex-A76 de 64 bits a 2,4 GHz, además de interfaces USB, GPIO, Ethernet y conectividad inalámbrica [16].

En el sistema desarrollado, la Raspberry Pi 5 cumple varias funciones:

- ejecuta el programa principal escrito en Python;
- se comunica con la placa de control del Transbot;
- recibe el video de la cámara USB;
- procesa los fotogramas con OpenCV;
- ejecuta la inferencia del modelo YOLO;
- administra el servidor web Flask;
- recibe los comandos de movimiento; y
- almacena las evidencias de las pruebas.

La selección de esta plataforma responde a su compatibilidad con Linux, Python, cámaras USB y bibliotecas de visión artificial. Sin embargo, su capacidad no equivale a la de una estación de trabajo equipada con una GPU dedicada. Por ello, el entrenamiento se realizó en una plataforma externa y la Raspberry Pi se destinó principalmente a la inferencia.

1.6.12. Visión artificial

La visión artificial es el área que desarrolla métodos para adquirir, procesar e interpretar información contenida en imágenes o secuencias de video. Un sistema de visión puede incluir iluminación, cámara, formación de la imagen, procesamiento, extracción de características y una etapa de decisión.

En aplicaciones agrícolas, la información visual puede utilizarse para analizar color, textura, forma, tamaño y distribución de las plantas. Sin embargo, las imágenes de campo presentan condiciones variables que no siempre se encuentran bajo control. La iluminación solar, las sombras, el viento, la superposición de hojas, la humedad y el fondo del terreno pueden modificar la apariencia de los objetos.

Adhinata et al. identifican la similitud visual entre cultivo y maleza, las oclusiones y los efectos de iluminación como problemas importantes en los sistemas de reconocimiento agrícola. Los autores señalan que el aprendizaje profundo generalmente presenta mejores resultados que los métodos tradicionales, aunque requiere datos representativos y mayores recursos de procesamiento [3].

En el rover, la visión artificial se aplica sobre fotogramas capturados por una cámara USB. Cada fotograma es enviado al modelo, que intenta localizar las plantas correspondientes a las clases definidas durante el entrenamiento.

Flujo del sistema de visión artificial

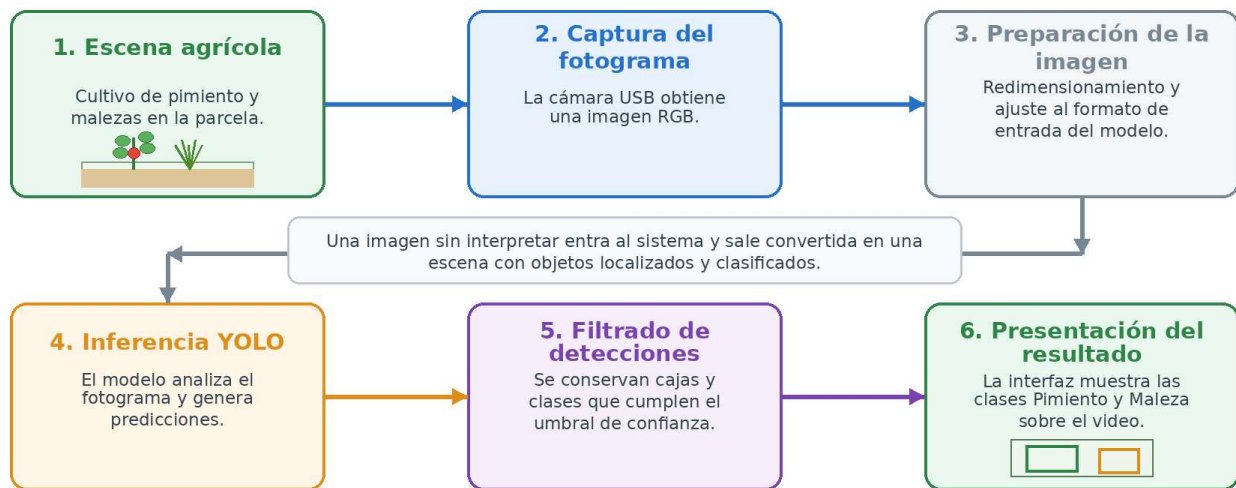


Figura 5: Etapas generales del sistema de visión artificial empleado en el rover. Fuente: Autor.

1.6.13. Adquisición digital de imágenes

Una imagen digital está formada por una matriz de elementos denominados píxeles. En una imagen en color, cada píxel contiene valores asociados con canales cromáticos. La resolución indica la cantidad de píxeles que forman la imagen, mientras que la tasa de fotogramas expresa cuántas imágenes se capturan por unidad de tiempo.

Una resolución elevada puede conservar más detalles de plantas pequeñas, pero también incrementa el volumen de información que debe procesarse. De manera similar, una tasa de fotogramas alta mejora la continuidad del video, aunque aumenta la carga computacional y el ancho de banda requerido. El sistema debe establecer un equilibrio entre detalle, velocidad de transmisión y capacidad de procesamiento.

La cámara USB se instaló sobre un soporte con movimiento horizontal y vertical. Este mecanismo permite orientar el campo visual sin desplazar completamente el rover. Durante las pruebas, la posición de la cámara, la distancia respecto a la vegetación y la presencia de cielo en la imagen influyeron sobre la calidad visual y sobre las detecciones obtenidas.

1.6.14. Inteligencia artificial, aprendizaje automático y aprendizaje profundo

La inteligencia artificial comprende métodos diseñados para que un sistema realice tareas relacionadas con percepción, predicción, razonamiento o toma de decisiones. El aprendizaje automático constituye una parte de la inteligencia artificial en la que los modelos ajustan sus parámetros a partir de datos, en lugar de depender exclusivamente de reglas programadas manualmente.

El aprendizaje profundo emplea redes neuronales formadas por múltiples capas de procesamiento. Estas redes pueden aprender representaciones jerárquicas: las primeras capas reconocen patrones simples, mientras las capas posteriores combinan esos patrones para representar estructuras más complejas [8].

En el reconocimiento tradicional de plantas, el desarrollador puede definir manualmente características relacionadas con color, bordes o textura. En un modelo profundo, una parte importante de estas características se aprende durante el entrenamiento. Esto facilita el tratamiento de escenas complejas, aunque no elimina la necesidad de disponer de imágenes correctamente etiquetadas y representativas.

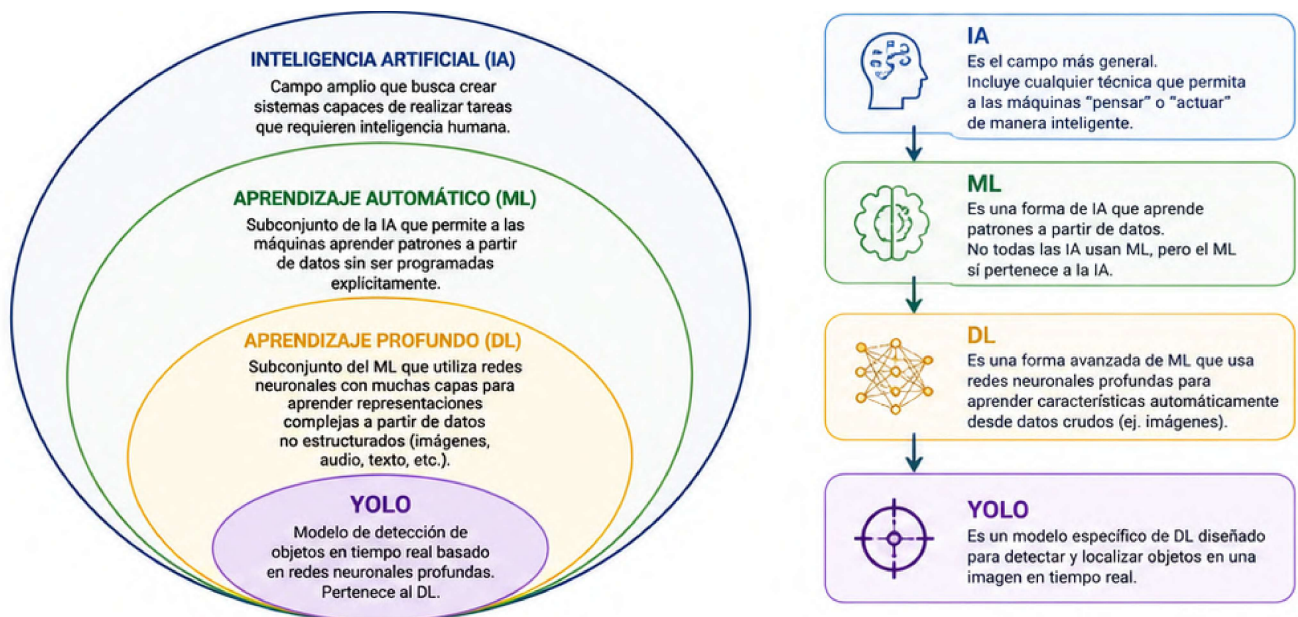


Figura 6: Relación conceptual entre inteligencia artificial, aprendizaje automático y aprendizaje profundo. Fuente: Autor.

1.6.15. Redes neuronales convolucionales

Las redes neuronales convolucionales, conocidas como CNN por *Convolutional Neural Networks*, fueron diseñadas para procesar datos con estructura espacial, como las imágenes. Su funcionamiento incluye operaciones de convolución que aplican filtros sobre regiones locales para extraer características visuales.

Las capas iniciales pueden responder a bordes, cambios de intensidad o texturas. A medida que la información avanza por la red, las capas posteriores combinan dichas respuestas para reconocer patrones de mayor complejidad. El entrenamiento modifica los parámetros de los filtros para reducir el error entre las predicciones y las etiquetas disponibles [9, 8].

Las CNN constituyen la base de numerosos modelos de clasificación, segmentación y detección de objetos. En la presente investigación se utilizan indirectamente dentro de la arquitectura YOLO para extraer características de las plantas observadas por la cámara.

Cómo una CNN transforma una imagen en una detección

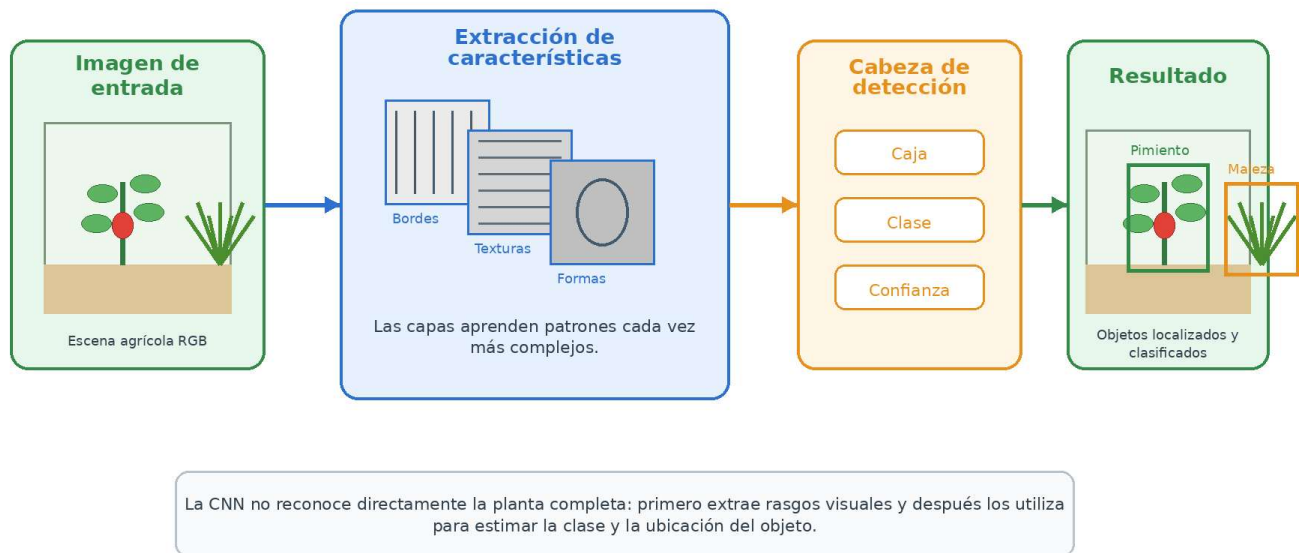


Figura 7: Representación simplificada del procesamiento realizado por una red neuronal convolucional. Fuente: Autor.

1.6.16. Clasificación, detección y segmentación

La clasificación de imágenes asigna una o varias clases al contenido general de una imagen, pero no indica necesariamente la ubicación de cada objeto. Por ejemplo, un clasificador podría indicar que una fotografía contiene maleza, aunque no mostrar dónde se encuentra.

La detección de objetos agrega información espacial. Su salida incluye una clase, una caja delimitadora y un nivel de confianza para cada objeto detectado. Este enfoque fue seleccionado porque permite visualizar simultáneamente plantas de cultivo y malezas dentro del mismo fotograma.

La segmentación trabaja con una localización más detallada, debido a que clasifica píxeles o delimita la forma precisa de cada instancia. Aunque puede ser útil cuando las plantas se superponen, requiere anotaciones más detalladas y una mayor carga de procesamiento. Considerando el alcance y los recursos del proyecto, se seleccionó detección de objetos mediante cajas delimitadoras.

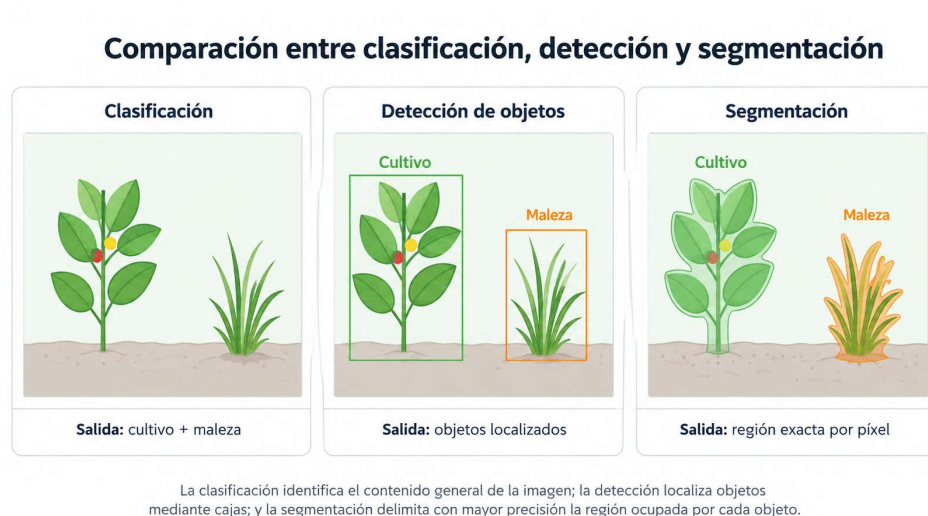


Figura 8: Comparación entre clasificación, detección de objetos y segmentación de imágenes. Fuente: Autor.

1.6.17. Arquitectura YOLO

YOLO, acrónimo de *You Only Look Once*, es una familia de modelos de detección de objetos. El trabajo original formuló la detección como un problema unificado en el que una sola red predice directamente las cajas delimitadoras y las probabilidades de clase a partir de la imagen completa [7].

A diferencia de métodos que generan regiones candidatas y luego las clasifican mediante procesos separados, YOLO integra la localización y la clasificación dentro de una misma arquitectura. Esta característica permite alcanzar velocidades adecuadas para aplicaciones sobre imágenes, videos y transmisiones en tiempo real.

Para cada detección, el modelo produce generalmente:

- las coordenadas de una caja delimitadora;
- la clase estimada;
- un valor de confianza; y
- la relación espacial entre la predicción y el objeto observado.

El nivel de confianza representa la seguridad estimada por el modelo, pero no garantiza que una detección sea correcta. Un modelo puede producir un

falso positivo con confianza elevada cuando observa un escenario diferente de los utilizados durante el entrenamiento. Este comportamiento fue considerado durante las pruebas y motivó el ajuste individual de los umbrales de las clases pimiento y maleza.

En esta investigación se entrenó un modelo YOLO con dos clases. La clase pimiento representa el cultivo de referencia y la clase maleza representa la vegetación no deseada identificada dentro de la parcela.

Flujo simplificado de detección de objetos mediante YOLO

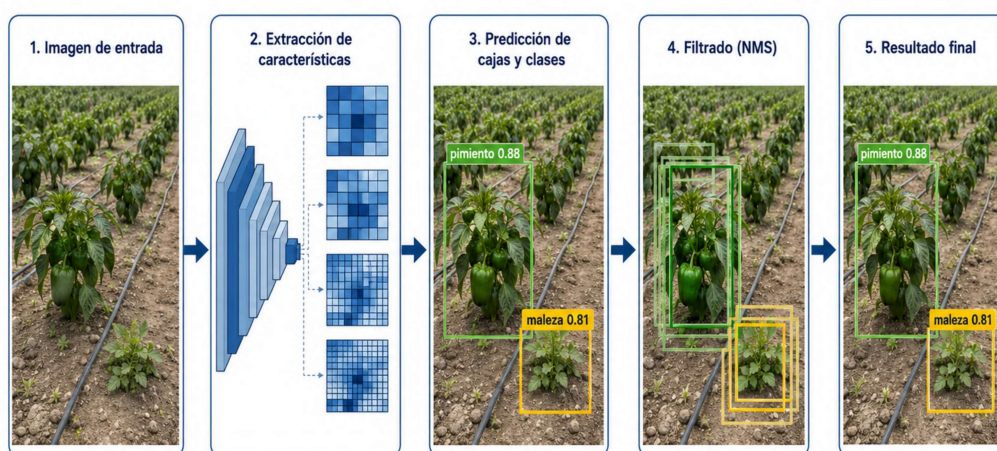


Figura 9: Flujo simplificado de detección de objetos mediante YOLO. Fuente: Autor.

1.6.18. Conjunto de datos

El conjunto de datos es uno de los elementos principales de un sistema de aprendizaje supervisado. Está compuesto por imágenes y anotaciones que indican al modelo qué objetos debe aprender. La calidad del conjunto no depende únicamente de la cantidad de imágenes, sino también de la diversidad de fondos, ángulos, distancias, tamaños, condiciones de iluminación y estados de crecimiento.

Un conjunto formado por imágenes demasiado similares puede limitar la capacidad de generalización. Aunque la métrica obtenida durante la validación sea favorable, el modelo puede presentar errores al observar escenarios nuevos. En detección agrícola, esta situación resulta especialmente importante por la variabilidad natural del terreno y la vegetación [3, 4].

Para reducir este problema, las imágenes deben incluir ejemplos de:

- plantas de cultivo observadas desde diferentes posiciones;
- malezas de distintos tamaños;
- cultivo y maleza en la misma escena;
- suelo, piedras, hojas secas y mangueras;
- sombras e iluminación intensa;
- vegetación parcialmente oculta; y
- imágenes sin objetos de interés.

Las imágenes sin anotaciones, conocidas como imágenes negativas o de fondo, ayudan a enseñar al modelo que una escena también puede contener ninguna de las clases. Su inclusión puede reducir falsos positivos cuando se seleccionan fondos representativos del entorno de operación.

1.6.19. Etiquetado mediante cajas delimitadoras

El etiquetado establece la referencia que utiliza el modelo durante el entrenamiento. En detección de objetos, cada instancia se encierra mediante una caja delimitadora y se asocia con una clase.

Roboflow permite realizar anotaciones mediante cajas, polígonos o máscaras. Las cajas delimitadoras requieren menos tiempo de anotación que los polígonos, aunque estos últimos describen con mayor precisión la forma del objeto [24].

En este proyecto se emplearon cajas delimitadoras debido a que la tarea seleccionada fue detección de objetos. Durante el etiquetado fue necesario mantener un criterio uniforme. Una misma clase no debería anotarse unas veces como una planta individual y otras como una agrupación extensa, salvo que esa diferencia forme parte de una decisión explícita del diseño.

La consistencia de las anotaciones resulta importante porque el modelo aprende tanto la apariencia del objeto como la manera en que fue delimitado. Cajas excesivamente amplias, incompletas o inconsistentes pueden producir predicciones superpuestas, cajas de tamaño inadecuado o errores de localización.

1.6.20. División del conjunto de datos

El conjunto de imágenes se divide normalmente en entrenamiento, validación y prueba. Cada subconjunto cumple una función diferente:

- **Entrenamiento:** se utiliza para ajustar los parámetros del modelo.
- **Validación:** se emplea durante el desarrollo para observar la evolución del modelo y ajustar configuraciones.
- **Prueba:** se reserva para evaluar el comportamiento final sobre imágenes que no participaron en el ajuste.

Roboflow permite establecer esta división al generar una versión del conjunto de datos, junto con las transformaciones de preprocesamiento y aumento seleccionadas [25].

Las imágenes consecutivas obtenidas por una cámara pueden ser casi idénticas. Si fotografías muy similares aparecen simultáneamente en entrenamiento y prueba, el resultado puede sobreestimar la capacidad de generalización. Por esta razón, la separación debe considerar no solo el nombre de los archivos, sino también la similitud entre escenas.

1.6.21. Preprocesamiento y aumento de datos

El preprocesamiento transforma las imágenes antes de utilizarlas en el entrenamiento. Puede incluir redimensionamiento, corrección de orientación y normalización. El redimensionamiento permite presentar al modelo entradas con dimensiones compatibles con su arquitectura.

El aumento de datos genera variaciones de las imágenes de entrenamiento mediante transformaciones controladas. Entre las operaciones comunes se encuentran el volteo, cambios de brillo, modificaciones de exposición, rotación, recorte y adición de ruido. Estas transformaciones pueden mejorar la diversidad del conjunto y reducir el sobreajuste, siempre que no alteren el significado de las clases [12].

La documentación de Roboflow indica que los aumentos se aplican a las imágenes asignadas al conjunto de entrenamiento y que pueden ayudar al modelo a responder mejor ante imágenes no observadas. Sin embargo, también recomienda revisar primero el equilibrio, la representación y el tamaño del conjunto original [26].

En el proyecto se utilizaron transformaciones moderadas de volteo horizontal, brillo y exposición. Estas operaciones se relacionan con las variaciones encontradas durante la adquisición de imágenes en campo.

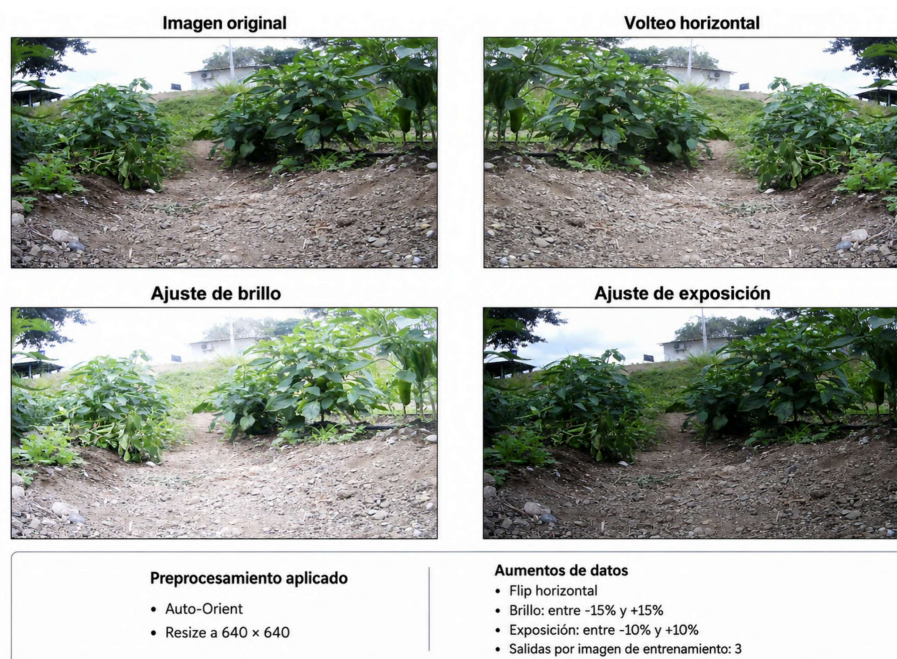


Figura 10: Ejemplos conceptuales de las transformaciones empleadas para el aumento de datos. Fuente: Autor.

1.6.22. Entrenamiento, épocas y sobreajuste

Durante el entrenamiento, el modelo procesa repetidamente las imágenes, calcula el error de sus predicciones y ajusta sus parámetros. Una época corresponde a una pasada completa por el conjunto de entrenamiento.

Aumentar el número de épocas no garantiza una mejora indefinida. En las primeras iteraciones, el modelo puede aprender características útiles; sin embargo, un entrenamiento excesivo puede provocar sobreajuste. Este fenómeno ocurre cuando el modelo se adapta demasiado a los ejemplos de entrenamiento y pierde capacidad para responder correctamente ante escenas nuevas.

La evolución de las pérdidas y las métricas de validación permite analizar si el entrenamiento continúa mejorando. El modelo seleccionado no debe ser necesariamente el correspondiente a la última época, sino aquel que obtiene el mejor comportamiento de validación de acuerdo con el criterio empleado.

Para disponer de capacidad de procesamiento gráfico, el entrenamiento se realizó mediante Google Colab. La documentación de este servicio lo define como un entorno alojado de cuadernos Jupyter que no requiere instalación y que puede proporcionar acceso a GPU y TPU, aunque sus recursos y límites no están garantizados [27].

1.6.23. Métricas de evaluación

Las métricas permiten analizar cuantitativamente el funcionamiento de un detector. Para interpretarlas se utilizan los siguientes conceptos:

- **Verdadero positivo (TP):** objeto detectado y asociado correctamente con una anotación real.
- **Falso positivo (FP):** detección generada donde no existe un objeto correcto o cuya clase no corresponde.
- **Falso negativo (FN):** objeto real que no fue detectado.

La evaluación de un modelo de detección no debe limitarse a una única medida, debido a que cada métrica representa un aspecto diferente del funcionamiento. En la Tabla 4 se resumen las métricas empleadas para analizar la identificación y localización de plantas en el presente proyecto.

Tabla 4: Métricas utilizadas para evaluar el modelo de detección

Métrica	Descripción	Interpretación en el proyecto	Rango o expresión
Precisión	Proporción de las detecciones generadas por el modelo que corresponden con objetos reales de la clase evaluada.	Permite analizar cuántas cajas identificadas como pimiento o maleza fueron correctas y cuántas correspondieron a falsos positivos.	$Precision = TP / (TP + FP)$
Exhaustividad o <i>recall</i>	Proporción de los objetos reales presentes en las imágenes que fueron detectados por el modelo.	Permite establecer qué cantidad de plantas de pimiento o malezas fueron localizadas y cuántas no fueron detectadas.	$Recall = TP / (TP + FN)$
Intersección sobre unión (IoU)	Relación entre el área de intersección y el área de unión de la caja predicha y la caja de referencia.	Permite evaluar la coincidencia espacial entre la ubicación predicha y la anotación utilizada como referencia.	Valores entre 0 y 1.
AP por clase	Área asociada con la relación entre precisión y <i>recall</i> para una clase y un criterio de coincidencia determinado.	Permite analizar separadamente el comportamiento del detector para la clase pimiento y para la clase maleza.	Valores entre 0 y 1.
mAP@50	Promedio de la precisión promedio de las clases utilizando un umbral de IoU igual a 0,50.	Proporciona una evaluación de la detección utilizando un criterio de coincidencia espacial moderado.	Promedio de AP con IoU = 0,50.
mAP@50-95	Promedio de la precisión promedio calculada con umbrales de IoU desde 0,50 hasta 0,95, con incrementos de 0,05.	Aplica criterios progresivamente más rigurosos y permite evaluar con mayor exigencia la localización de las cajas.	Promedio de AP para varios valores de IoU.
Matriz de confusión	Organiza las clases reales y las clases predichas para identificar aciertos, confusiones, detecciones de fondo y objetos omitidos.	Permite observar si el modelo confunde pimiento con maleza y analizar falsos positivos y falsos negativos relacionados con el fondo.	Matriz de frecuencias o valores normalizados.

Fuente: Elaboración propia a partir de [11, 10].

Como se resume en la Tabla 4, la precisión y el *recall* describen comportamientos complementarios. Una precisión elevada indica una menor proporción de falsos positivos, mientras que un *recall* elevado indica una menor cantidad de objetos omitidos. Las métricas mAP incorporan además la relación entre ambas medidas y el criterio de coincidencia espacial establecido mediante IoU [11, 10].

La precisión expresa la proporción de detecciones positivas que fueron correctas:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

Una precisión elevada indica que una gran parte de las cajas producidas por el modelo corresponde con objetos reales.

La exhaustividad o *recall* expresa la proporción de objetos reales que fueron detectados:

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

Un *recall* elevado indica que el modelo omite pocos objetos. Existe un compromiso entre ambas métricas, debido a que reducir el umbral de confianza puede aumentar las detecciones y el *recall*, pero también incrementar los falsos positivos.

1.6.24. Intersección sobre unión

La intersección sobre unión, denominada IoU por *Intersection over Union*, mide la coincidencia espacial entre una caja predicha y la caja real. Se calcula dividiendo el área de intersección entre el área de unión:

$$IoU = \frac{|B_p \cap B_r|}{|B_p \cup B_r|} \quad (3)$$

donde B_p representa la caja predicha y B_r la caja de referencia. Un valor próximo a uno indica una coincidencia espacial elevada. Para decidir si una predicción es correcta se establece un umbral mínimo de IoU [11].

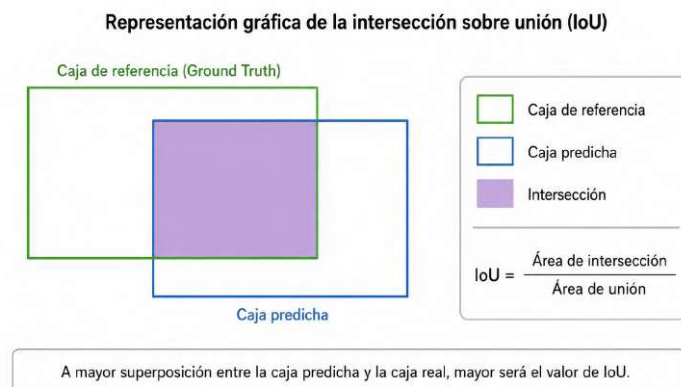


Figura 11: Representación gráfica de la intersección sobre unión entre una caja predicha y una caja de referencia. Fuente: Autor.

1.6.25. Precisión promedio y mAP

La precisión promedio o AP resume la relación entre precisión y *recall* para una clase. La media de la precisión promedio, conocida como mAP, obtiene un promedio entre las clases evaluadas.

La métrica mAP@50 utiliza un umbral IoU de 0,50. Por su parte, mAP@50–95 calcula el promedio para varios umbrales comprendidos entre 0,50 y 0,95, con incrementos de 0,05. Esta segunda métrica aplica criterios de localización progresivamente más estrictos [11, 10].

En una tesis no debe interpretarse el mAP como el único indicador de calidad. También deben analizarse los resultados por clase, la matriz de confusión, los falsos positivos, las omisiones y las evidencias obtenidas durante la prueba de campo.

1.6.26. Matriz de confusión

La matriz de confusión organiza las predicciones según la clase estimada y la clase real. En detección de objetos también puede incluir una categoría de fondo. Esta categoría no constituye una clase que el modelo deba detectar, sino que representa objetos reales no detectados o predicciones que no coinciden con una anotación válida.

La diagonal principal contiene las detecciones asignadas correctamente. Los valores ubicados fuera de la diagonal muestran confusiones entre clases, mientras que las celdas relacionadas con el fondo ayudan a analizar falsos positivos y falsos negativos.

En el presente trabajo, la matriz permite estudiar tres situaciones principales:

- malezas detectadas correctamente;
- plantas de pimienta detectadas correctamente; y
- predicciones u omisiones relacionadas con el fondo.

1.6.27. Umbral de confianza

El umbral de confianza determina el valor mínimo necesario para presentar una detección. Un umbral bajo permite conservar más cajas, pero puede incrementar los falsos positivos. Un umbral alto elimina predicciones débiles, aunque también puede ocultar objetos reales.

Los umbrales no deben seleccionarse únicamente para producir imágenes visualmente favorables. Deben relacionarse con el comportamiento del modelo en validación y con las condiciones de operación. En este proyecto se utilizaron valores diferentes para pimienta y maleza debido a que ambas clases presentaron distribuciones de confianza y dificultades distintas.

1.6.28. Comunicación inalámbrica

La comunicación inalámbrica permite controlar el rover sin una conexión física entre la computadora y la plataforma. El sistema utiliza Wi-Fi para formar una red local entre la Raspberry Pi y el equipo del operador.

Durante las pruebas de campo, la computadora funcionó como punto de acceso y la Raspberry Pi se conectó a dicha red. Este mecanismo permitió acceder mediante SSH para administrar el sistema y mediante un navegador web para controlar el movimiento, orientar la cámara y observar las detecciones.

La comunicación local no requiere necesariamente acceso permanente a Internet. No obstante, la estabilidad de la transmisión depende de la distancia, los obstáculos, la intensidad de la señal y la alimentación de los dispositivos.

1.6.29. Interfaz web

Una interfaz web permite utilizar un navegador como medio de interacción entre el operador y el rover. En este proyecto, la página presenta el video, las cajas de detección, el estado del movimiento, la orientación de la cámara y el conteo de las clases.

También contiene controles para:

- avanzar y retroceder;
- girar a la izquierda o derecha;
- detener el rover;
- orientar la cámara;
- activar o desactivar la detección; y
- guardar evidencias.

El servidor fue desarrollado con Flask, un framework ligero para aplicaciones web basado en Python [20]. La interfaz visible se construyó con HTML, CSS y JavaScript, mientras las solicitudes enviadas al servidor activan las funciones correspondientes en la Raspberry Pi.

1.6.30. Python y bibliotecas empleadas

Python es un lenguaje de programación de propósito general distribuido bajo una licencia de código abierto aprobada por la Open Source Initiative. Su sintaxis y disponibilidad de bibliotecas permiten integrarlo en aplicaciones de automatización, análisis de datos, desarrollo web e inteligencia artificial [18].

Las bibliotecas principales utilizadas fueron:

- **OpenCV:** biblioteca de visión artificial utilizada para abrir la cámara, adquirir fotogramas, redimensionar imágenes, dibujar cajas, escribir textos y codificar el flujo de video [19].
- **Flask:** framework encargado de crear el servidor web y gestionar las rutas para video, movimiento, orientación de cámara, estado y almacenamiento de evidencias [20].
- **Ultralytics:** biblioteca utilizada para cargar el archivo de pesos entrenado y ejecutar la detección sobre los fotogramas.
- **NumPy:** biblioteca utilizada para manipular matrices y estructuras numéricas asociadas con las imágenes.
- **PySerial:** biblioteca empleada para la comunicación serial con los dispositivos del rover.
- **Transbot-Lib:** biblioteca proporcionada para controlar motores, servomotores y funciones físicas de la plataforma.

La combinación de estas herramientas permitió integrar control, visión artificial, inferencia y presentación de resultados dentro de un único programa ejecutado en la Raspberry Pi.

1.7. Marco contextual

El presente trabajo se desarrolla dentro del campo de la ingeniería electrónica y automatización, específicamente en la integración de robótica móvil, sistemas embebidos, comunicación inalámbrica y visión artificial aplicadas al sector agrícola. El proyecto tiene como contexto institucional la Universidad Estatal Península de Santa Elena, donde se plantea el uso de tecnologías accesibles para responder a necesidades del entorno productivo y fortalecer la investigación aplicada.

La actividad agropecuaria posee relevancia territorial y económica en Ecuador. Según la Encuesta de Superficie y Producción Agropecuaria Continua de 2024, el país registró aproximadamente 4,8 millones de hectáreas con labor agropecuaria. La región Costa concentró el 55,4 % de dicha superficie, lo que evidencia la importancia de estudiar herramientas tecnológicas adaptadas a las condiciones agrícolas de esta región [28].

La provincia de Santa Elena presenta condiciones productivas en las que la disponibilidad y gestión del agua constituyen factores importantes. Los resultados del Módulo de Información Ambiental y Tecnificación Agropecuaria señalaron que, en 2023, Santa Elena presentó la mayor proporción provincial de superficie cultivada bajo riego en relación con su superficie sembrada o plantada [29]. Este contexto favorece la existencia de parcelas con sistemas de irrigación, mangueras, líneas de cultivo y vegetación espontánea, elementos que también forman parte de la complejidad visual enfrentada por el detector.

El escenario experimental corresponde a una parcela agrícola ubicada en Manglaralto, provincia de Santa Elena. En la parcela se encontraron plantas de pimiento utilizadas como cultivo de referencia y diferentes tipos de vegetación considerada maleza por crecer dentro o alrededor del área cultivada. La prueba se efectuó bajo condiciones reales de iluminación natural, suelo irregular, presencia de hojas secas, sombras, mangueras de riego y movimiento de las plantas producido por el viento.

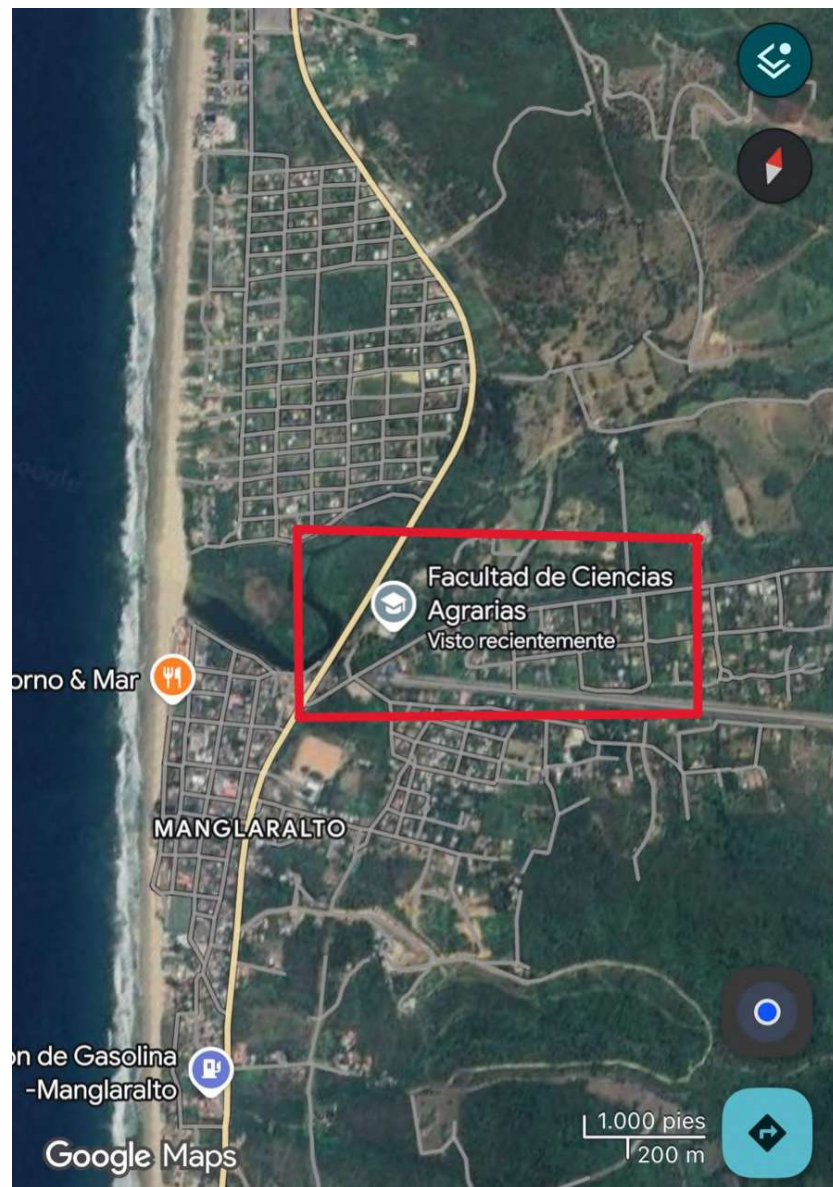


Figura 12: Ubicación general del escenario experimental en la provincia de Santa Elena. Fuente: Autor.

En la Tabla 5 se deberá registrar la información real que permita caracterizar el lugar donde se ejecutaron las pruebas.

Tabla 5: Caracterización general del escenario experimental

Aspecto	Descripción registrada
Provincia	Santa Elena
Cantón, parroquia o comuna	Manglaralto
Cultivo de referencia	Pimiento
Condiciones de iluminación	Luz natural, sombra y condición mixta
Características del terreno	Irregular, entorno no controlado
Medio de comunicación	Red Wi-Fi local
Condiciones ambientales relevantes	Mayormente Soleado

Fuente: Autor.

La elección de una parcela real, en lugar de un ambiente de laboratorio, permitió evaluar el funcionamiento del sistema frente a condiciones que no pueden controlarse completamente. Estas condiciones incluyeron cambios bruscos de iluminación, sectores sobreexpuestos por la radiación solar, variaciones en la distancia entre cámara y planta, superposición de hojas, vegetación de diferentes tamaños y fondos con tonalidades similares.



Figura 13: Vista general de la parcela agrícola empleada como escenario experimental. Fuente: Autor.

El cultivo de pimiento se seleccionó como referencia para diferenciar la planta deseada de la vegetación presente a su alrededor. No obstante, el título de la investigación mantiene un alcance general relacionado con la identificación de malezas en parcelas agrícolas. El modelo desarrollado no debe interpretarse como un detector universal de todas las especies de cultivo o maleza, debido a que sus predicciones dependen directamente de las clases, imágenes y condiciones incluidas durante el entrenamiento.

La plataforma física empleada fue el robot Transbot-SE compatible con Raspberry Pi 5. Aunque esta plataforma dispone de recursos para aplicaciones de ROS, seguimiento, manipulación y navegación, en el presente proyecto fue configurada como un rover controlado remotamente. Su función fue transportar la cámara y la unidad de procesamiento durante la observación de la parcela [15].

La Raspberry Pi 5 se instaló como unidad central. Sobre ella se ejecutaron el control del rover, el servidor web, la transmisión de video y el modelo de detección. La comunicación se estableció mediante una red Wi-Fi local generada por la computadora del operador. Esta configuración respondió a la limitada disponibilidad de infraestructura eléctrica y de conectividad en el lugar de prueba.

El proceso de desarrollo comprendió la configuración de la plataforma, programación del control, captura de imágenes, creación del conjunto de datos, etiquetado en Roboflow, entrenamiento en Google Colab, transferencia del archivo de pesos a la Raspberry Pi e integración de las detecciones en la página web.

Las clases definidas fueron “Pimiento” y “Maleza”. El suelo, las hojas secas, los objetos de riego y demás elementos del fondo no se etiquetaron como una tercera clase, debido a que en detección de objetos el fondo se representa mediante regiones sin anotación. El objetivo fue que el modelo aprendiera a generar cajas únicamente alrededor de las plantas asociadas con las dos clases de interés.

La interfaz desarrollada permitió al operador observar el video en tiempo real, controlar la dirección del rover, orientar la cámara y visualizar el nombre y la confianza de cada detección. Asimismo, se implementó una función para almacenar evidencias con la hora de captura, número de pimientos, número de malezas y porcentajes de confianza.

La evaluación combinó los resultados del conjunto de prueba con una experimentación en campo. Las métricas de precisión, *recall*, mAP@50, mAP@50–95 y la matriz de confusión se emplearon para analizar el modelo. Las evidencias capturadas en la parcela permitieron examinar su funcionamiento en el escenario para el cual fue desarrollado.

Durante las pruebas se observaron limitaciones asociadas con el tamaño del conjunto de datos, el parecido visual entre las plantas, el movimiento producido por el viento, la iluminación intensa y el tamaño reducido de algunas malezas. Estas condiciones forman parte del contexto real de aplicación y son consideradas posteriormente en el análisis de resultados, las conclusiones y las recomendaciones.

Los beneficiarios potenciales del trabajo son la comunidad académica, estudiantes e investigadores interesados en robótica agrícola, visión artificial y sistemas embebidos. El prototipo también puede funcionar como base para proyectos posteriores que amplíen el conjunto de datos, incorporen otras especies vegetales, mejoren la autonomía energética o integren mecanismos de manejo selectivo.

Por tanto, el aporte de la investigación no consiste en presentar un sistema comercial terminado, sino en desarrollar, integrar y evaluar un prototipo funcional basado en herramientas modificables. El trabajo documenta tanto las capacidades alcanzadas como las dificultades encontradas al trasladar un modelo de detección desde el entorno de entrenamiento hasta una parcela agrícola real.

2. Métodos y diseño experimental

2.1. Métodos

Para el desarrollo de la investigación se utilizaron métodos orientados al análisis del problema, la construcción del prototipo y la evaluación de su funcionamiento. Se consideraron la investigación aplicada, el método experimental, el método bibliográfico, el método tecnológico y el método de observación. La combinación de estos métodos permitió fundamentar la propuesta, integrar sus componentes y recopilar información sobre el comportamiento del rover y del sistema de visión artificial.

La relación entre los métodos seleccionados y las actividades realizadas se resume en la Tabla 6.

Tabla 6: Relación entre los métodos de investigación y su aplicación

Método	Aplicación en la investigación	Evidencia generada
Investigación aplicada	Desarrollo de una solución tecnológica para identificar pimienta y maleza.	Prototipo funcional e integración del sistema.
Experimental	Ejecución de pruebas planificadas sobre el modelo y el rover.	Registros, métricas e imágenes de las pruebas.
Bibliográfico	Revisión de publicaciones y documentación técnica.	Fundamentación teórica y selección tecnológica.
Tecnológico	Diseño, configuración e integración de hardware y software.	Rover, modelo, servidor e interfaz implementados.
Observación	Registro directo del comportamiento durante la operación.	Anotaciones sobre errores, condiciones ambientales y detecciones.

Fuente: Autor.

Flujo metodológico de la investigación



Figura 14: Flujo metodológico seguido para desarrollar y evaluar el sistema.
Fuente: Autor.

■ Investigación aplicada

La investigación es de tipo aplicada, debido a que utiliza conocimien-

tos existentes de robótica móvil, sistemas embebidos, visión artificial y aprendizaje profundo para atender una necesidad práctica relacionada con la identificación de malezas en parcelas agrícolas. La investigación aplicada se caracteriza por orientar la generación y utilización del conocimiento hacia un objetivo práctico específico. En este proyecto, dicha orientación se concreta mediante el desarrollo y evaluación de un rover equipado con una cámara y un modelo de detección de objetos [31].

■ **Método experimental**

Se empleó el método experimental para evaluar el funcionamiento del rover y del sistema de visión artificial mediante pruebas previamente planificadas. Durante estas pruebas se observaron variables relacionadas con la captura de imágenes, las condiciones de iluminación, la distancia entre la cámara y la vegetación, la presencia de plantas de pimiento y malezas, y las predicciones producidas por el modelo.

El diseño de experimentos permite organizar las pruebas, controlar o registrar los factores que pueden influir sobre una respuesta y analizar los resultados obtenidos. En la presente investigación, este método se aplicó tanto durante la evaluación del modelo como en las pruebas de funcionamiento realizadas en la parcela agrícola [32].

■ **Método bibliográfico**

El método bibliográfico se utilizó para recopilar, seleccionar y analizar información científica relacionada con agricultura de precisión, manejo de malezas, robótica agrícola, visión artificial, aprendizaje profundo, modelos YOLO y sistemas de código abierto.

La revisión de literatura constituye un método de investigación que debe realizarse de manera estructurada y transparente para identificar el conocimiento existente, comparar resultados y establecer los fundamentos que respaldan una investigación. Por esta razón, se priorizaron artículos científicos revisados por pares, libros académicos, documentos de organismos oficiales y documentación técnica de los fabricantes y desarrolladores de las herramientas utilizadas [33].

■ **Método tecnológico**

El método tecnológico se empleó para diseñar, construir, integrar y evaluar un artefacto funcional compuesto por elementos físicos y lógicos. El desarrollo comprendió la selección de componentes electrónicos, la configuración del rover Transbot-SE, la programación del control remo-

to, la implementación de la interfaz web, el entrenamiento del modelo y su integración en la Raspberry Pi 5.

La metodología de ciencia del diseño plantea que una investigación tecnológica puede desarrollarse mediante la identificación del problema, la definición de objetivos, el diseño y construcción de una solución, su demostración y su evaluación. Estas etapas guardan correspondencia con el proceso seguido para obtener el prototipo utilizado en esta investigación [34].

■ Método de observación

El método de observación se empleó durante las pruebas para registrar directamente el comportamiento del rover, la estabilidad de la transmisión de video, la respuesta de los controles, la orientación de la cámara y las detecciones generadas por el modelo.

La observación permite recopilar información de los acontecimientos mientras estos se producen en su contexto de estudio. En este proyecto se realizó una observación directa y sistemática del prototipo durante su funcionamiento, registrando las condiciones de la parcela, los cambios de iluminación, el movimiento de la vegetación, las detecciones correctas, los falsos positivos y los objetos no identificados [35].

2.2. Diseño experimental

El diseño experimental estableció la arquitectura, los procedimientos y los criterios empleados para desarrollar y evaluar el rover propuesto. Esta etapa comprendió la organización de la plataforma robótica, el sistema de adquisición de imágenes, la preparación del conjunto de datos, el entrenamiento del modelo de detección, la comunicación inalámbrica y la interacción del operador mediante una interfaz web.

La propuesta fue diseñada para identificar visualmente plantas de pimienta y malezas presentes en una parcela agrícola. El rover se utilizó como una plataforma móvil de inspección controlada por el usuario, mientras que la Raspberry Pi 5 ejecutó el procesamiento de imágenes, la inferencia del modelo y el servidor encargado de transmitir los resultados.

Los robots utilizados en operaciones agrícolas deben integrar sistemas de percepción, procesamiento, comunicación, alimentación y actuación dentro de una plataforma capaz de funcionar en entornos no estructurados. En estos escenarios pueden presentarse variaciones de iluminación, irregularidades del terreno, superposición de vegetación y cambios ambientales que afectan el

funcionamiento del sistema [36]. Por esta razón, el diseño se organizó mediante subsistemas que fueron evaluados individualmente antes de su integración.

2.2.1. Arquitectura general del sistema

La arquitectura fue organizada en cuatro subsistemas funcionales: adquisición, procesamiento, comunicación e interacción con el usuario. Esta división permitió establecer la función de cada elemento y definir el flujo de información desde la captura de la imagen hasta la presentación de las detecciones.

En la Tabla 7 se describen los subsistemas que componen la propuesta.

Tabla 7: Subsistemas que conforman la arquitectura del rover

Subsistema	Elementos principales	Función dentro de la propuesta
Adquisición	Cámara USB y mecanismo PTZ	Capturar los fotogramas de la vegetación y modificar la orientación del campo visual durante la inspección de la parcela.
Procesamiento	Raspberry Pi 5, Python, OpenCV y modelo YOLO	Procesar las imágenes, ejecutar la detección, filtrar las predicciones y generar las cajas, etiquetas y conteos.
Comunicación	Red Wi-Fi local, SSH, SCP y protocolo HTTP	Intercambiar comandos, video, estados y archivos entre la computadora del operador y la Raspberry Pi.
Interacción	Servidor Flask e interfaz web	Permitir el control del rover, la orientación de la cámara, la visualización de detecciones y el almacenamiento de evidencias.

Fuente: Autor.

El subsistema de adquisición genera los fotogramas que posteriormente son procesados en la Raspberry Pi. Cuando la detección se encuentra activa, la imagen es enviada al modelo entrenado, el cual devuelve las coordenadas de las cajas, las clases y los niveles de confianza. El programa representa estas predicciones sobre el fotograma y transmite el resultado hacia el navegador.

Paralelamente, la interfaz envía los comandos de movimiento y orientación al servidor. El servidor interpreta cada solicitud y ejecuta la función correspondiente mediante la biblioteca de control del Transbot-SE. De esta manera, el flujo de imágenes y el flujo de control operan dentro de la misma

aplicación, aunque la decisión de desplazamiento permanece bajo responsabilidad del operador.

La combinación de cámaras, unidades compactas de procesamiento y herramientas de software modificables ha permitido desarrollar dispositivos experimentales de bajo costo para la detección de malezas. Estas propuestas demuestran la viabilidad de realizar procesamiento visual próximo al lugar de adquisición de los datos [37].

La Figura 15 debe representar gráficamente las relaciones entre los componentes del sistema.

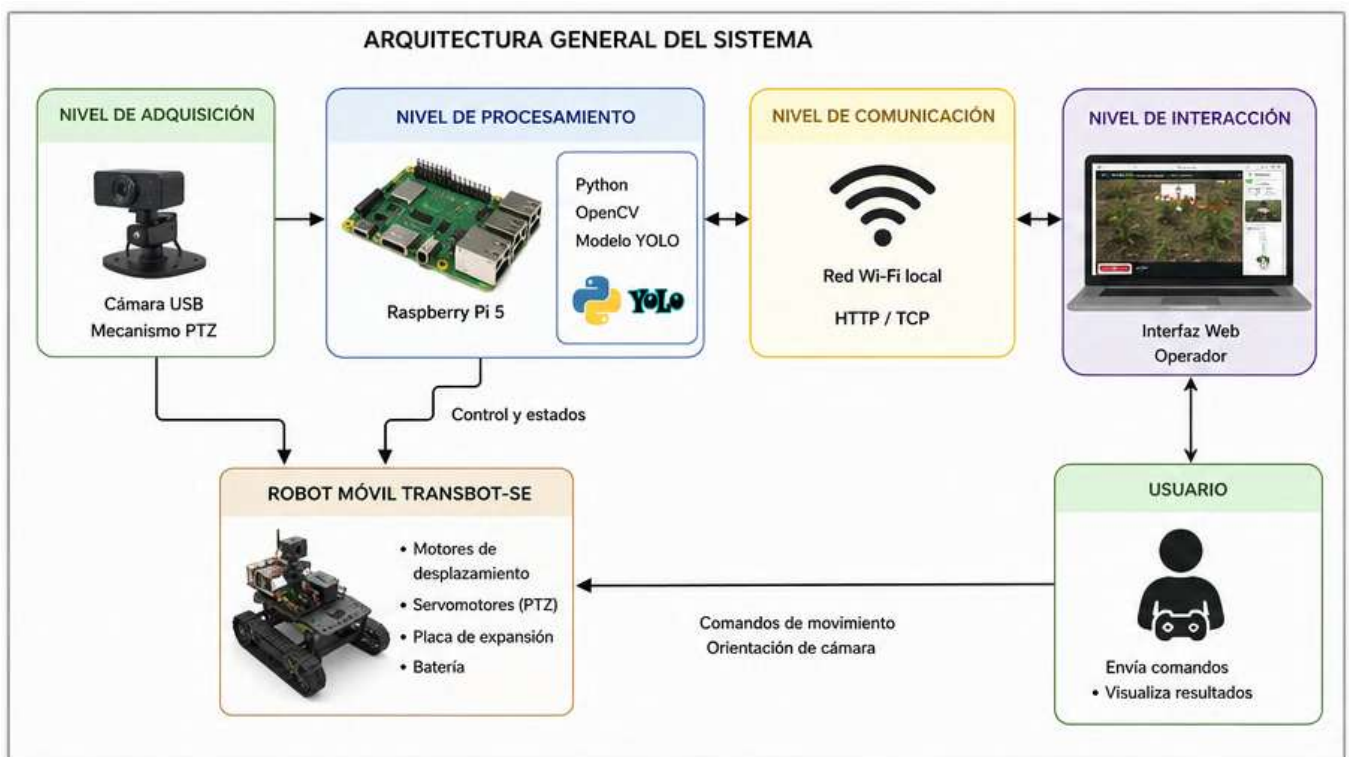


Figura 15: Arquitectura general del rover y del sistema de visión artificial.
Fuente: Autor.

2.2.2. Diseño de la plataforma robótica

La plataforma física seleccionada fue el robot móvil Transbot-SE, debido a que integra una estructura de orugas, motores, placa de expansión, sistema de alimentación y soporte para una cámara orientable. Esta configuración evitó la necesidad de construir un nuevo sistema de locomoción y permitió concentrar el desarrollo en la adquisición de imágenes, el procesamiento local

y la interacción con el usuario.

La Raspberry Pi 5 fue definida como unidad principal de procesamiento. Su función consistió en ejecutar el programa desarrollado en Python, comunicarse con la placa de expansión, recibir las imágenes de la cámara, ejecutar el modelo de detección y administrar la interfaz web.

La cámara USB fue instalada sobre el mecanismo PTZ de dos grados de libertad. El movimiento horizontal y vertical permitió modificar el campo visual y observar vegetación ubicada en diferentes posiciones respecto al rover. Esta capacidad resultó especialmente útil para mantener las plantas dentro del encuadre sin tener que reposicionar constantemente toda la plataforma.

El desplazamiento se diseñó como un sistema de control remoto. Las órdenes de avance, retroceso, giro y detención son emitidas por el usuario desde la interfaz. En consecuencia, el alcance de la investigación no incluye navegación autónoma, planificación de trayectorias, modelado cinemático, modelado dinámico, odometría o construcción de mapas.

La Figura 16 mostrará la disposición de los componentes utilizados en la plataforma.

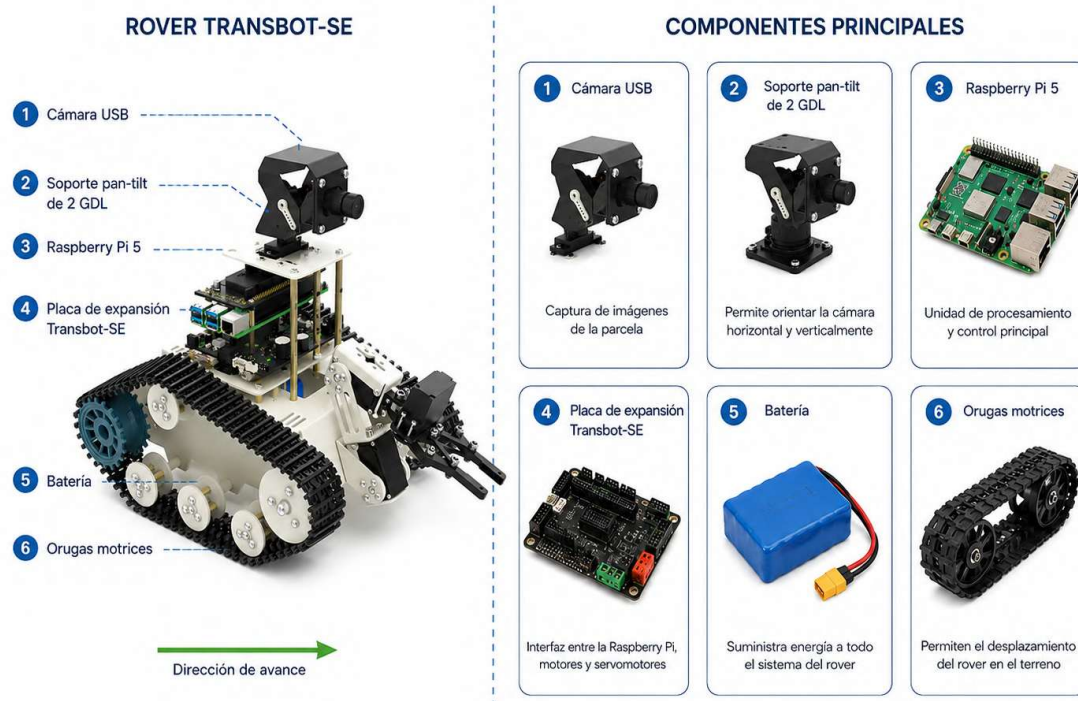


Figura 16: Configuración física del rover utilizado en la investigación. Fuente: Autor.

2.2.3. Diseño del sistema de visión artificial

El sistema de visión artificial se diseñó mediante una secuencia que comprendió la adquisición de imágenes, depuración de archivos, definición de clases, etiquetado, división del conjunto de datos, aumento de imágenes, entrenamiento, validación y selección del modelo.

Este proceso buscó obtener un detector capaz de localizar las plantas de interés mediante cajas delimitadoras. Las clases definidas fueron *Pimiento*, correspondiente al cultivo utilizado durante la experimentación, y *Maleza*, correspondiente a la vegetación no deseada presente dentro o alrededor de la parcela.

Adquisición y depuración de imágenes Las imágenes fueron capturadas en condiciones agrícolas reales con el propósito de incorporar variabilidad de iluminación, fondo, distancia, orientación y tamaño de las plantas. Se

registraron escenas con plantas de pimiento, malezas, ambas clases simultáneamente y zonas en las que no se encontraban objetos de interés.

Durante la captura se modificó la posición de la cámara con respecto a la vegetación y se incluyeron plantas con diferentes niveles de superposición, sombras y exposición solar. La adquisición directa en campo permite representar condiciones que no suelen presentarse en imágenes obtenidas dentro de un ambiente controlado. El conjunto DeepWeeds, por ejemplo, fue construido mediante imágenes adquiridas *in situ* en diferentes localizaciones, incluyendo vegetación y fondos naturales [38].

Después de la captura se efectuó una inspección visual para eliminar archivos dañados, imágenes sin suficiente enfoque, fotografías completamente sobreexpuestas o subexpuestas y duplicados exactos. También se revisaron las secuencias consecutivas para reducir la presencia de escenas prácticamente idénticas.

Esta depuración fue necesaria para disminuir el riesgo de distribuir fotografías muy similares entre entrenamiento y prueba, lo que podría producir métricas optimistas sin representar una capacidad real de generalización.

El conjunto inicial quedó conformado por 952 imágenes seleccionadas.

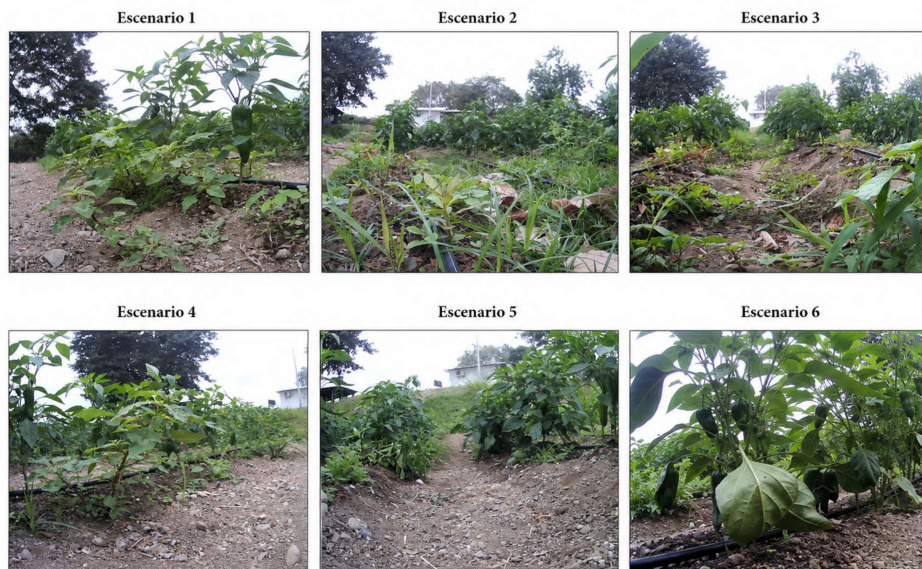


Figura 17: Adquisición y depuración de las imágenes utilizadas para conformar el conjunto de datos. Fuente: Autor.

Definición de las clases Las clases utilizadas se presentan en la Tabla 8.

Tabla 8: Clases definidas para el conjunto de datos

Clase	Criterio utilizado para la anotación
Pimiento	Planta correspondiente al cultivo de referencia utilizado durante la experimentación. La caja debe contener la planta visible y evitar incluir cantidades excesivas de suelo u objetos del fondo.
Maleza	Vegetación no deseada ubicada dentro o alrededor de la parcela. Cada planta debe anotarse individualmente cuando sus límites visuales puedan distinguirse.

Fuente: Autor.

Los elementos correspondientes al suelo, piedras, mangueras, hojas secas y estructuras no fueron definidos como una tercera clase. Estas zonas permanecieron sin anotación para que el modelo las interpretara como fondo.

Esta decisión permite que la salida del detector se concentre únicamente en las dos clases relacionadas con el objetivo de la investigación.

Documentación y trazabilidad del conjunto Para mantener la trazabilidad del conjunto de datos se registraron el origen de las imágenes, la ubicación general de captura, el dispositivo utilizado, las condiciones ambientales, las clases, la cantidad de anotaciones y las transformaciones aplicadas.

La documentación de un conjunto de datos debe describir su motivación, composición, proceso de recopilación, preprocesamiento, usos previstos y limitaciones. Esta información permite interpretar correctamente el alcance de los datos y evita considerarlos representativos de situaciones que no fueron observadas durante su construcción [39].

En el presente trabajo se conservaron como mínimo los siguientes registros: periodo de captura, resolución original, número de imágenes, cantidad de anotaciones por clase, distribución de los subconjuntos y configuración del aumento de datos.

Etiquetado de las imágenes El etiquetado fue realizado mediante cajas delimitadoras. Cada planta visible fue encerrada dentro de una caja y asociada con su clase correspondiente.

Las anotaciones se ajustaron procurando incluir la planta completa cuando esta fuera visible. En casos de oclusión parcial, la planta fue etiquetada cuando conservaba suficientes características para ser reconocida. También

se procuró mantener el mismo criterio al etiquetar agrupaciones de hojas y plantas próximas entre sí.

Al finalizar la primera etapa de etiquetado se realizó una revisión completa para detectar cajas incompletas, regiones excesivamente amplias, clases incorrectas y plantas sin anotar. La consistencia de estas anotaciones es importante porque el modelo aprende tanto la apariencia de la vegetación como la manera en la que los objetos fueron delimitados.

La cantidad total de anotaciones fue de 952, distribuidas de la siguiente manera:

Tabla 9: Distribución de anotaciones por clase

Clase	Número de anotaciones
Pimiento	520
Maleza	432
Total	952

Fuente: Autor.

La Figura 18 mostrará un ejemplo representativo de las cajas utilizadas para ambas clases.

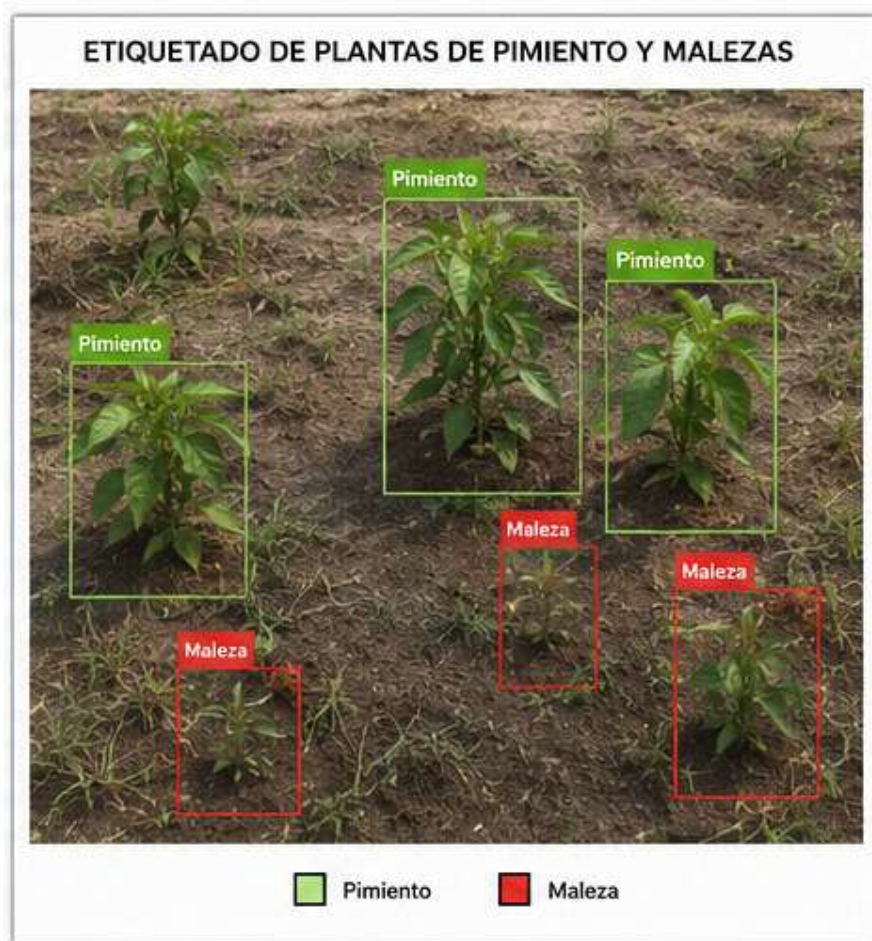


Figura 18: Etiquetado de plantas de pimienta y malezas mediante cajas delimitadoras. Fuente: Autor.

División del conjunto de datos El conjunto fue separado en entrenamiento, validación y prueba. El subconjunto de entrenamiento se utilizó para ajustar los parámetros del modelo. El de validación permitió observar su evolución durante el entrenamiento y seleccionar el mejor archivo de pesos. Finalmente, el subconjunto de prueba se reservó para la evaluación con imágenes que no participaron directamente en el ajuste.

La distribución aplicada se presenta en la Tabla 10.

Tabla 10: Distribución del conjunto de datos

Subconjunto	Porcentaje	Número de imágenes
Entrenamiento	70 %	667
Validación	20 %	190
Prueba	10 %	95

Fuente: Autor.

Antes de generar la división definitiva se revisaron las imágenes para evitar que fotografías consecutivas o casi idénticas quedaran en subconjuntos diferentes.

Preprocesamiento y aumento de datos Las imágenes fueron redimensionadas al tamaño de entrada establecido para el entrenamiento. Además, se aplicaron transformaciones únicamente sobre el subconjunto de entrenamiento para aumentar la variabilidad visual sin modificar el significado de las clases.

Las transformaciones seleccionadas fueron volteo horizontal, variación moderada de brillo y variación de exposición. Estas operaciones guardan relación con las condiciones que pueden presentarse en la parcela, donde la dirección de observación y la iluminación cambian durante la captura.

El uso de preprocesamiento y aumento de datos es habitual en aplicaciones agrícolas basadas en aprendizaje profundo, debido a que ayuda a representar variaciones presentes durante la operación y puede disminuir el sobreajuste [40].

La configuración aplicada se presenta en la Tabla 11.

Tabla 11: Transformaciones aplicadas al conjunto de entrenamiento

Transformación	Configuración utilizada
Redimensionamiento	640x640 píxeles.
Volteo horizontal	Aplicado con la probabilidad definida en la versión del conjunto de datos.
Variación de brillo	-15 and +15.
Variación de exposición	-10 and +10.

Fuente: Autor.

Después de aplicar las transformaciones, el conjunto quedó conformado por 952 imágenes.

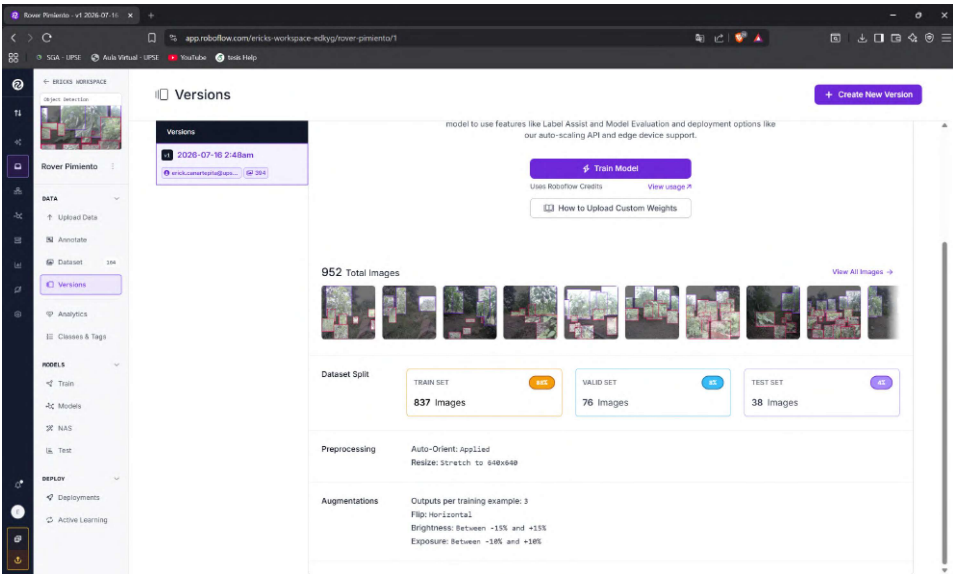


Figura 19: Configuración de la versión del conjunto de datos utilizada durante el entrenamiento. Fuente: Autor.

2.2.4. Diseño del entrenamiento y selección del modelo

El entrenamiento fue realizado mediante aprendizaje por transferencia. En lugar de iniciar el ajuste con parámetros aleatorios, se utilizaron pesos previamente entrenados y posteriormente adaptados a las clases pimienta y maleza.

El aprendizaje por transferencia utiliza conocimiento obtenido en una tarea o dominio de origen para mejorar el aprendizaje en una tarea relacionada. Esta estrategia resulta útil cuando el conjunto específico posee una cantidad de imágenes menor que la requerida para entrenar una red profunda completamente desde cero [41].

El entrenamiento se ejecutó en un entorno con aceleración mediante GPU. Los principales parámetros deben registrarse en la Tabla 12.

Tabla 12: Parámetros empleados durante el entrenamiento

Parámetro	Configuración
Modelo inicial	YOLO11n preentrenado (yolo11n.pt)
Número de épocas	100
Tamaño de imagen	640 x 640 píxeles
Tamaño de lote	16
Paciencia	30 épocas
Optimizador	Automático
Dispositivo	GPU de Google Colab
Archivo del conjunto	data.yaml

Fuente: Autor.

Durante cada época el modelo procesó las imágenes de entrenamiento, calculó las pérdidas y actualizó sus parámetros. Posteriormente, su comportamiento fue evaluado mediante el subconjunto de validación.

Al finalizar se obtuvieron archivos correspondientes a la última época y al mejor resultado de validación. Para la integración en la Raspberry Pi 5 se seleccionó el archivo:

`best.pt`

La selección se basó en el comportamiento obtenido durante la validación y no únicamente en la época final ejecutada.

2.2.5. Diseño del sistema de control

El control se diseñó mediante funciones de alto nivel proporcionadas por la biblioteca del Transbot-SE. Cada función recibe una instrucción procedente de la interfaz y activa los motores o servomotores correspondientes.

El sistema permite avanzar, retroceder, girar hacia ambos lados y detener el rover. También permite modificar la orientación horizontal y vertical de la cámara. La relación entre los controles de la interfaz y las acciones físicas se presenta en la Tabla 13.

Tabla 13: Acciones disponibles en el sistema de control

Comando	Respuesta del sistema
Avanzar	Activa los motores para desplazar el rover hacia adelante.
Retroceder	Invierte el sentido de movimiento para desplazar la plataforma hacia atrás.
Girar a la izquierda o derecha	Modifica el sentido relativo de los motores para cambiar la orientación del rover.
Detener	Interrumpe la activación de los motores.
Orientar cámara	Modifica la posición de los servomotores horizontal y vertical del soporte PTZ.

Fuente: Autor.

El comando de detención fue considerado prioritario para evitar desplazamientos no deseados. La velocidad de operación se mantuvo reducida con el propósito de disminuir vibraciones y facilitar la observación de las plantas.

El operador permanece dentro del lazo de control y selecciona la dirección mientras observa el video transmitido. El modelo identifica plantas, pero no genera automáticamente instrucciones para los motores.

2.2.6. Diseño del sistema de comunicación

La comunicación entre la computadora y el rover fue diseñada mediante una red Wi-Fi local. La computadora del operador funciona como punto de acceso y la Raspberry Pi se conecta a esta red.

La administración remota utiliza SSH, mientras que la transferencia del código, los archivos de configuración y el modelo se realiza mediante SCP. La interfaz web se consulta mediante HTTP.

Esta configuración permite operar el rover sin una conexión permanente a Internet, debido a que el modelo y las dependencias se ejecutan localmente. Sin embargo, la continuidad de la interacción depende del alcance de la red, la intensidad de la señal y la presencia de obstáculos entre los equipos.

La Figura 20 debe representar la conexión entre la computadora, la Raspberry Pi, el servidor Flask y la placa del rover.

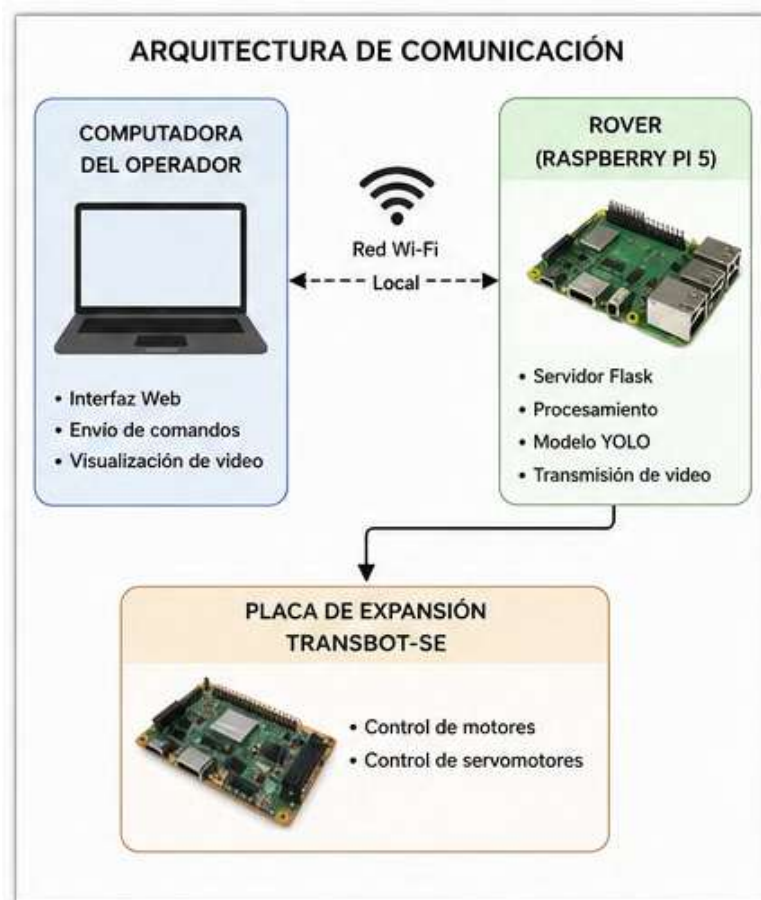


Figura 20: Arquitectura de comunicación entre el operador y el rover. Fuente: Autor.

2.2.7. Diseño de la interfaz web

La interfaz fue diseñada para centralizar la supervisión y el control del prototipo. La pantalla principal contiene el flujo de video, los controles de desplazamiento, los controles de la cámara, el estado de la detección y el conteo de objetos.

Cuando el modelo se encuentra activo, cada caja aceptada se representa sobre la imagen junto con la clase estimada y el porcentaje de confianza. El conteo de pimientos y malezas se actualiza a partir de las detecciones del fotograma procesado.

Además, se incorporó una función de almacenamiento de evidencias. Esta función permite guardar una imagen que contiene las predicciones, los conteos y la información temporal de la captura.

La Figura 21 presentará la distribución final de los elementos de la página.



Figura 21: Interfaz web diseñada para el control y supervisión del rover.
Fuente: Autor.

2.2.8. Integración del modelo en la Raspberry Pi

El archivo seleccionado fue transferido a la Raspberry Pi 5 y cargado desde el programa principal. La aplicación obtiene una secuencia de fotogramas desde la cámara y ejecuta la inferencia cuando el usuario activa la detección.

El procesamiento comprende la adquisición del fotograma, adecuación al tamaño de entrada, ejecución del modelo, lectura de cajas, filtrado por confianza, representación de etiquetas y transmisión del resultado.

La inferencia en sistemas embebidos está condicionada por la memoria, la capacidad de procesamiento, el ancho de banda y el consumo energético. Por este motivo, la selección del tamaño del modelo y la resolución de entrada debe mantener un equilibrio entre precisión y velocidad [42].

Se han desarrollado sistemas capaces de ejecutar reconocimiento de cultivos y malezas sobre Raspberry Pi mediante arquitecturas optimizadas. No

obstante, el rendimiento obtenido depende de la complejidad del modelo, del tamaño de la entrada y de los recursos disponibles [43].

La configuración de inferencia se registra en la Tabla 14.

Tabla 14: Parámetros utilizados durante la inferencia

Parámetro	Configuración
Tamaño de entrada	640 × 640 píxeles
Confianza mínima para pimiento	0,50 (50 %)
Confianza mínima para maleza	0,40 (40 %)
Umbral de IoU	0,45
Resolución del video	640 × 480 píxeles a 30 FPS
Dispositivo de inferencia	CPU de la Raspberry Pi 5

Fuente: Autor.

2.2.9. Gestión de evidencias

El sistema fue diseñado para almacenar imágenes representativas de las pruebas de campo. Cada evidencia conserva el fotograma procesado, las cajas delimitadoras, las clases, los niveles de confianza, el número de pimientos, el número de malezas y la hora de captura.

Los archivos se almacenan en una carpeta específica dentro de la Raspberry Pi. El nombre incorpora la fecha y la hora con el propósito de evitar que una nueva captura sobrescriba una evidencia anterior.

La estructura utilizada puede expresarse de la siguiente manera:

evidencia_AAAA-MM-DD_HH-MM-SS.jpg

Las imágenes almacenadas constituyen una fuente de información para el análisis del Capítulo 3, debido a que permiten revisar detecciones correctas, falsos positivos, falsos negativos y confusiones entre clases.

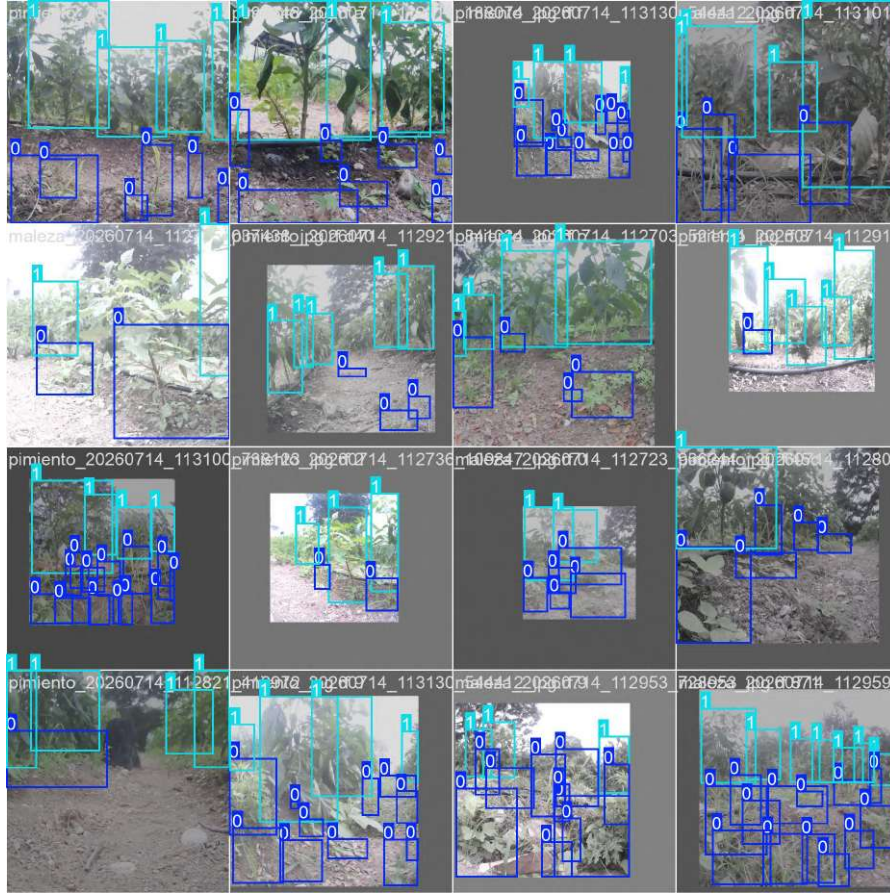


Figura 22: Ejemplo de evidencia almacenada por el sistema durante las pruebas. Fuente: Autor.

2.2.10. Protocolo de evaluación experimental

El protocolo fue diseñado para evaluar el funcionamiento integrado del rover en condiciones agrícolas. Antes de iniciar cada sesión se comprobó la alimentación, la conexión de la cámara, la comunicación inalámbrica, la respuesta de los motores y la carga correcta del modelo. El texto original ya contemplaba estas verificaciones y la evaluación por escenarios reales. :contentReference[oaicite:2]index=2

La evaluación incluyó escenas con diferentes combinaciones de vegetación y condiciones visuales. Los escenarios se resumen en la Tabla 15.

Tabla 15: Escenarios definidos para las pruebas de campo

Código	Escenario	Finalidad de la prueba
E1	Plantas de pimiento sin malezas visibles.	Evaluar detecciones correctas y falsos positivos de la clase maleza.
E2	Malezas sin plantas de pimiento visibles.	Evaluar detecciones de maleza y falsos positivos de pimiento.
E3	Pimiento y maleza dentro de la misma escena.	Analizar la diferenciación entre las dos clases.
E4	Suelo y objetos de fondo sin plantas de interés.	Analizar detecciones incorrectas sobre el fondo.
E5	Plantas parcialmente ocluidas o agrupadas.	Evaluar el comportamiento ante superposición de hojas.
E6	Iluminación solar directa.	Observar el efecto de exposición y contraste elevados.
E7	Sombras sobre la vegetación.	Evaluar cambios en color, brillo y textura.
E8	Movimiento de hojas producido por el viento.	Analizar desenfoque, variación de posición y estabilidad de las cajas.

Fuente: Autor.

La evaluación de sistemas de visión instalados en robots agrícolas debe realizarse bajo condiciones reales, debido a que el comportamiento observado en imágenes estáticas o controladas puede diferir del obtenido durante la operación en campo [44].

Factores e indicadores Los factores observados durante las pruebas se presentan en la Tabla 16.

Tabla 16: Factores e indicadores considerados durante las pruebas

Factor o respuesta	Descripción	Indicador	Forma de registro
Clase presente	Tipo de planta visible en la escena.	Pimiento, maleza, ambas o ninguna.	Revisión visual de la evidencia.
Condición de iluminación	Distribución de luz sobre la escena.	Luz directa, sombra o condición mixta.	Descripción y fotografía.
Distancia de captura	Separación entre la cámara y la vegetación.	Distancia aproximada en centímetros.	Medición registrada durante la prueba.
Estado del rover	Condición de la plataforma durante la captura.	Detenido o en desplazamiento.	Registro realizado por el operador.
Detección correcta	Coincidencia entre objeto real y predicción.	Verdadero positivo.	Revisión manual.
Detección incorrecta	Predicción producida sobre una clase u objeto incorrecto.	Falso positivo.	Revisión manual.
Objeto omitido	Planta real no localizada por el modelo.	Falso negativo.	Revisión manual.
Confianza	Valor asignado por el modelo a la predicción.	Número entre 0 y 1 o porcentaje.	Información mostrada en la interfaz.
Tiempo de procesamiento	Duración de la inferencia sobre un fotograma.	Milisegundos o fotogramas por segundo.	Registro generado por el programa.

Fuente: Autor.

Procedimiento de ejecución Cada prueba comenzó con la ubicación del rover en un punto seguro y la verificación del estado de la batería, la cámara y la conexión inalámbrica. Posteriormente, se ejecutó la aplicación, se accedió a la interfaz y se comprobó la respuesta de los controles.

Una vez activada la detección, el rover fue dirigido hacia el escenario seleccionado. La cámara se orientó hacia la vegetación y la plataforma se mantuvo detenida durante la captura de las evidencias más representativas. Después de registrar las condiciones observadas, el rover fue desplazado hacia el siguiente escenario.

Cada escenario fue repetido 10 veces. La repetición permitió observar si las detecciones se mantenían estables o cambiaban ante pequeñas variaciones en la posición, iluminación o movimiento de las plantas.

Evaluación de las detecciones Las evidencias fueron revisadas para determinar las detecciones correctas de pimiento y maleza, las confusiones entre clases, las predicciones generadas sobre el fondo y las plantas no identificadas.

Estos registros fueron relacionados posteriormente con las métricas del conjunto de prueba. La evaluación de campo no sustituye las métricas cuantitativas, sino que permite observar cómo se manifiestan los errores durante la utilización real del prototipo.

Fuentes de error Las principales fuentes de error fueron agrupadas en factores ambientales, visuales, tecnológicos y asociados con el conjunto de datos.

Los factores ambientales comprendieron la iluminación solar, las sombras y el movimiento de hojas producido por el viento. Los factores visuales incluyeron la similitud entre plantas, las oclusiones, el tamaño reducido de algunas malezas y el desenfoque producido por el movimiento.

Entre los factores tecnológicos se consideraron las vibraciones del rover, los retrasos en la comunicación y las limitaciones de procesamiento de la Raspberry Pi. Finalmente, los factores asociados con el conjunto de datos incluyeron la cantidad limitada de imágenes, la variabilidad insuficiente y las posibles inconsistencias de etiquetado.

La variabilidad entre terrenos, etapas de crecimiento, apariencia de las plantas y condiciones del suelo constituye una dificultad importante para generalizar los sistemas de reconocimiento hacia parcelas diferentes de las utilizadas durante el entrenamiento [45].

2.2.11. Criterios de seguridad y delimitación del experimento

Durante las pruebas se mantuvo una distancia adecuada entre el rover, el operador y las plantas. La velocidad fue limitada y el comando de detención permaneció disponible durante toda la operación.

La alimentación fue desconectada antes de manipular cables o componentes. También se evitó utilizar el prototipo bajo lluvia, sobre suelo excesivamente húmedo o en condiciones que pudieran afectar los elementos electrónicos.

El rover fue empleado únicamente para inspección e identificación visual. No se instalaron mecanismos de corte, pulverización, aplicación de herbicidas o actuadores destinados a eliminar malezas. Esta delimitación garantiza que la evaluación se concentre en el control de la plataforma, la adquisición de imágenes y el funcionamiento del modelo de detección.

2.3. Desarrollo experimental

El desarrollo experimental correspondió a la etapa en la que los elementos definidos durante el diseño fueron instalados, configurados e integrados para obtener un prototipo funcional. El trabajo comprendió la preparación del robot móvil Transbot-SE, la configuración de la Raspberry Pi 5, la instalación de la cámara USB, la implementación del sistema de control, el desarrollo de la comunicación inalámbrica y la integración del modelo de visión artificial con la interfaz web.

La implementación se realizó de forma progresiva. Inicialmente se comprobó el funcionamiento individual de los componentes físicos y, posteriormente, se integraron los módulos de control, adquisición de imágenes, inferencia y transmisión de video. Este procedimiento permitió identificar problemas en cada etapa antes de ejecutar las pruebas completas en la parcela agrícola.

El resultado de esta fase fue una plataforma robótica controlada por un operador desde una computadora, capaz de adquirir imágenes del entorno, ejecutar un modelo de detección de objetos en la Raspberry Pi 5, diferenciar las clases pimiento y maleza, presentar las predicciones en una página web y almacenar evidencias de las pruebas.

2.3.1. Implementación de la planta

Para esta investigación, la planta estuvo constituida por el rover Transbot-SE y los elementos físicos y lógicos que intervienen en su funcionamiento. La plataforma incluyó la estructura de orugas, los motores de desplazamiento, la placa de expansión, la Raspberry Pi 5, el mecanismo PTZ de dos grados de libertad, la cámara USB, el sistema de alimentación y la red de comunicación inalámbrica.

La implementación comenzó con una revisión física del robot. Se comprobó que los motores se encontraran conectados a la placa de expansión, que las orugas estuvieran correctamente ubicadas y que los componentes no presentaran cables sueltos o elementos que limitaran el desplazamiento. Esta verificación fue necesaria debido a que la operación posterior se desarrolló sobre suelo agrícola con pequeñas irregularidades, piedras, hojas secas y vegetación próxima a la estructura.

La Raspberry Pi 5 se instaló como unidad central de procesamiento. Su función fue ejecutar el programa principal, comunicarse con la placa del Transbot-SE, recibir los fotogramas de la cámara, realizar la inferencia del modelo, administrar el servidor web y almacenar las evidencias. La placa de expansión permaneció como interfaz entre la computadora de placa única y los motores y servomotores del robot.

La cámara USB se colocó sobre el soporte PTZ del Transbot-SE. Esta disposición permitió modificar su orientación en los ejes horizontal y vertical desde la interfaz web. La posibilidad de orientar la cámara resultó importante para observar plantas ubicadas a diferentes distancias y evitar que una parte considerable del encuadre correspondiera al cielo, al chasis o a zonas sin vegetación.

La integración de cámaras, unidades de procesamiento y sistemas móviles es una característica recurrente en robots orientados al reconocimiento de malezas. No obstante, el funcionamiento de estos sistemas puede verse afectado por las variaciones de iluminación, el fondo, las etapas de crecimiento, la superposición de hojas y las diferencias morfológicas entre cultivo y maleza [46].

En el presente proyecto, el rover se utilizó como una plataforma de inspección visual. No se implementaron mecanismos de pulverización, corte, extracción o eliminación de malezas. De igual manera, el desplazamiento no fue autónomo, debido a que las decisiones de avance, retroceso y giro fueron emitidas directamente por el operador.



Figura 23: Rover Transbot-SE ensamblado para la adquisición y procesamiento de imágenes agrícolas. Fuente: Autor.

2.3.2. Implementación del sistema de alimentación

El sistema de alimentación utilizado correspondió al incorporado en la plataforma Transbot-SE. La batería suministró energía a la placa de expansión, la Raspberry Pi 5, los motores de desplazamiento, los servomotores y la cámara USB.

Antes de encender el rover se verificaron el estado físico de la batería, la polaridad de las conexiones y la sujeción de los conductores. Las manipulaciones físicas se realizaron con el sistema apagado para evitar cortocircuitos

o movimientos inesperados de los motores.

Una vez conectada la alimentación, se comprobó el encendido de la placa de expansión y el inicio correcto de la Raspberry Pi. Posteriormente, se verificó que la cámara fuera reconocida y que los motores y servomotores respondieran a comandos simples. Esta revisión permitió confirmar que la fuente disponible era suficiente para iniciar el sistema integrado.

Durante las pruebas se observó el estado de carga, debido a que una disminución de la energía disponible puede afectar la estabilidad de la Raspberry Pi, la captura de video y la respuesta de los actuadores. Por esta razón, las pruebas de campo se iniciaron con la batería cargada y se detuvieron cuando el sistema presentó señales de alimentación insuficiente o respuesta irregular.

Las características reales de la batería instalada deben registrarse directamente a partir de su etiqueta o ficha técnica:

- voltaje nominal: 11,1 v;
- capacidad nominal: 4400 mAh;
- tecnología de la batería: Ion-litio (Li-ion) recargable;
- autonomía aproximada observada: 45 min.

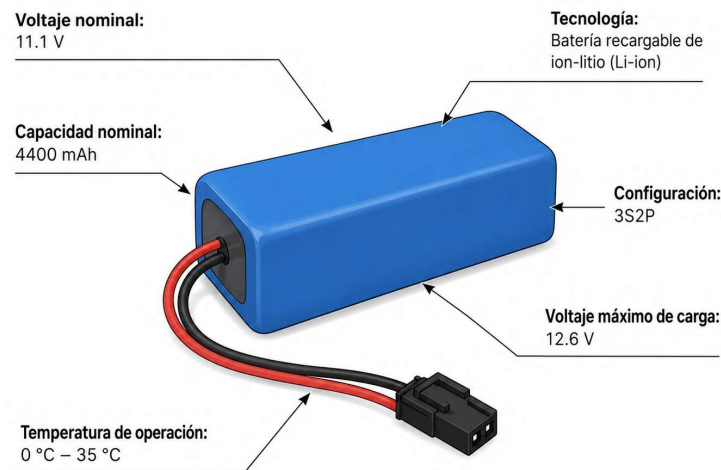


Figura 24: Representación esquemática de la batería utilizada en el rover y sus principales especificaciones técnicas. Fuente: Autor.

2.3.3. Preparación del entorno de ejecución

La Raspberry Pi 5 operó con un sistema GNU/Linux basado en Debian. Después de instalar el sistema operativo se configuraron el usuario, la contraseña, la zona horaria, la conexión de red y el acceso remoto mediante SSH.

Antes de instalar las herramientas del proyecto se actualizaron los índices de paquetes y los componentes disponibles en el sistema:

```
sudo apt update
sudo apt upgrade
```

Para evitar que las dependencias del proyecto modificaran los paquetes globales del sistema, se creó un entorno virtual de Python dentro del directorio de trabajo:

```
python3 -m venv /home/rover/tesis/venv
source /home/rover/tesis/venv/bin/activate
```

El entorno virtual concentró las bibliotecas utilizadas por la aplicación. Entre ellas se incluyeron Ultralytics, PyTorch, OpenCV, NumPy, Flask, PySerial, Pillow, SciPy, Psutil y las dependencias requeridas por el modelo de detección.

Las versiones registradas durante la configuración se presentan en la Tabla 17. Esta información es importante para facilitar la reproducibilidad del sistema y detectar posibles diferencias entre futuras instalaciones.

Tabla 17: Configuración del entorno de ejecución del rover

Componente	Versión o configuración
Sistema operativo	Debian GNU/Linux 13
Arquitectura	aarch64
Python	3.13.5
Ultralytics	8.4.83
PyTorch	2.12.1+cpu
Torchvision	0.27.1
OpenCV-Python	4.13.0.92
NumPy	2.4.6
Matplotlib	3.11.0
Psutil	7.2.2

Fuente: Autor, a partir de las versiones instaladas en el entorno virtual.

La aplicación principal se almacenó en el directorio `/home/rover/tesis/`. Para simplificar su ejecución se utilizó un archivo de inicio encargado de activar el entorno virtual y ejecutar el programa principal. Esta organización permitió iniciar el sistema sin introducir manualmente todos los comandos en cada prueba.

2.3.4. Implementación de la cámara USB

La cámara USB se conectó directamente a la Raspberry Pi 5 y se instaló sobre el mecanismo PTZ. Después de la conexión se comprobó que el sistema operativo reconociera el dispositivo de video.

En Linux, las cámaras y demás dispositivos de captura se administran mediante la interfaz Video4Linux2. Esta interfaz proporciona mecanismos para abrir dispositivos, consultar sus capacidades, seleccionar formatos y capturar secuencias de imágenes [47].

La existencia del dispositivo se comprobó mediante:

```
ls /dev/video*
```

Posteriormente, la cámara fue abierta desde el programa desarrollado en Python utilizando OpenCV:

```
cap = cv2.VideoCapture(0)
```

El índice empleado dependió de la posición asignada por el sistema operativo. Por esta razón, antes de iniciar cada prueba se verificó que la fuente seleccionada correspondiera a la cámara instalada sobre el rover.

Después de abrir el dispositivo se comprobó la captura de fotogramas, el enfoque, la orientación, la resolución y la estabilidad de la imagen. La posición inicial del soporte PTZ se ajustó para observar la vegetación delante del rover, procurando reducir la presencia de elementos que no formaban parte del área de interés.

La resolución utilizada durante la ejecución fue de 640 píxeles de ancho por 480 píxeles de alto. La tasa aproximada de adquisición fue de 30 fotogramas por segundo.

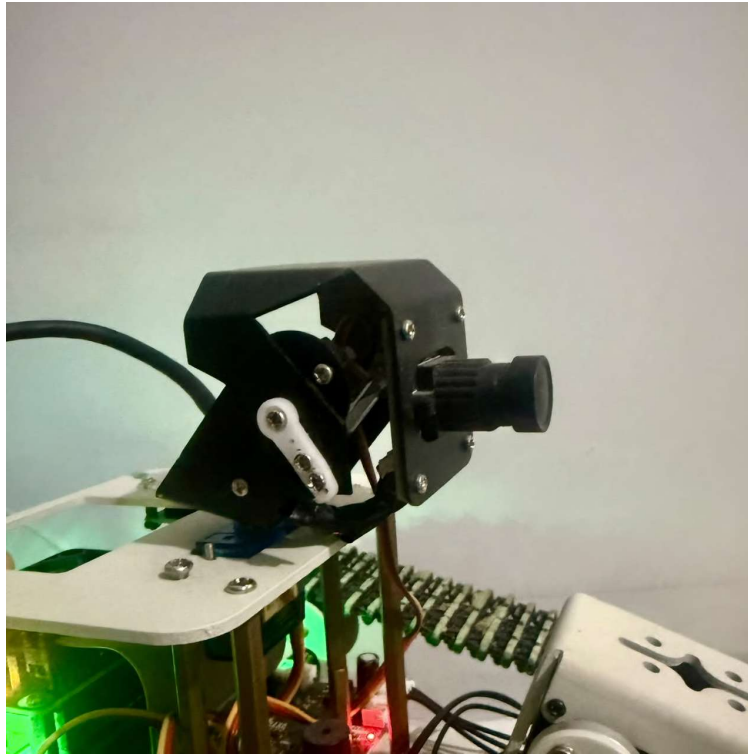


Figura 25: Cámara USB instalada sobre el mecanismo de orientación PTZ.
Fuente: Autor.

2.3.5. Implementación del sistema de control

El sistema de control fue implementado en Python mediante la biblioteca asociada con el Transbot-SE. Las funciones proporcionadas por esta biblioteca permitieron enviar instrucciones a los motores de desplazamiento y a los servomotores del soporte de cámara.

El programa principal incorporó funciones independientes para avanzar, retroceder, girar hacia la izquierda, girar hacia la derecha y detener el rover. También se implementaron funciones para modificar la orientación horizontal y vertical de la cámara.

La separación de cada acción en funciones facilitó su integración con el servidor web. Cuando el usuario presionaba un botón en la interfaz, el servidor recibía la solicitud, identificaba la acción requerida y ejecutaba la función correspondiente.

El movimiento se configuró en modo manual. El modelo de visión artificial no tomó decisiones sobre la dirección del rover ni generó comandos para los motores. Esta separación permitió que el operador mantuviera el control de la plataforma mientras observaba las detecciones producidas por el sistema.

El comando de detención se trató como una acción prioritaria. Su propósito fue detener los motores cuando el operador dejaba de accionar un control, cuando el rover se aproximaba demasiado a una planta o cuando se presentaba una interrupción en la comunicación.

La velocidad se mantuvo en un nivel moderado para reducir las vibraciones de la cámara y el desenfoque de las imágenes. Los valores empleados deben coincidir con los establecidos en el código definitivo:

- velocidad de avance: 0,15 m/s;
- velocidad de retroceso: 0,15 m/s en sentido inverso;
- velocidad de giro: 0,30 rad/s;
- posición horizontal inicial de la cámara: 90°;
- posición vertical inicial de la cámara: 90°.

2.3.6. Implementación del sistema de visión artificial

El modelo seleccionado durante el entrenamiento fue transferido a la Raspberry Pi 5 mediante el archivo `best.pt`. Este archivo contenía los pesos correspondientes a la mejor evaluación obtenida durante el proceso de validación.

El modelo fue almacenado en el directorio del proyecto y cargado al iniciar la aplicación. Antes de abrir la interfaz se verificó que la ruta del archivo fuera correcta y que la biblioteca Ultralytics pudiera interpretar la arquitectura y los pesos entrenados.

La salida de la cámara fue procesada fotograma a fotograma. Cuando la detección se encontraba desactivada, la aplicación transmitía la imagen sin ejecutar el modelo. Cuando el usuario habilitaba la detección, cada fotograma era enviado al modelo y se obtenían las coordenadas de las cajas, la clase estimada y el nivel de confianza.

Las investigaciones sobre detección agrícola muestran que los detectores de una etapa pueden ofrecer un equilibrio adecuado entre velocidad y exactitud. También señalan que las versiones compactas presentan ventajas para dispositivos con recursos limitados, aunque el comportamiento final depende de la diversidad de las imágenes y de la calidad de las anotaciones [48].

Las clases utilizadas fueron *Pimiento* y *Maleza*. Las regiones correspondientes al suelo, piedras, hojas secas, mangueras y demás objetos del entorno permanecieron como fondo.

Después de obtener las predicciones, el programa aplicó los umbrales definidos para cada clase. Las detecciones aceptadas fueron representadas mediante cajas delimitadoras y etiquetas que contenían el nombre de la clase y el porcentaje de confianza. Paralelamente, se actualizó el conteo visible en la interfaz.

Los estudios comparativos de la familia YOLO para malezas evidencian que la detección se beneficia de conjuntos de datos obtenidos bajo iluminación natural, diferentes etapas de crecimiento y variabilidad en el fondo. Estos estudios también muestran que la exactitud y la velocidad cambian según el tamaño y la versión del detector [49].

Los parámetros empleados durante la inferencia fueron:

- tamaño de entrada: 640×640 píxeles;
- confianza mínima para pimiento: 0,50;
- confianza mínima para maleza: 0,40;
- umbral de IoU: 0,45; y
- dispositivo de inferencia: CPU.

Los umbrales de confianza se ajustaron de forma independiente porque ambas clases presentaron comportamientos diferentes durante las pruebas. El objetivo del ajuste fue disminuir las detecciones incorrectas sin eliminar una cantidad excesiva de plantas reales.

2.3.7. Implementación del sistema de comunicación

La comunicación entre el rover y la computadora del operador se estableció mediante una red Wi-Fi local. La computadora funcionó como punto de acceso y la Raspberry Pi se conectó a la red generada.

Esta configuración permitió operar el rover en la parcela sin depender de un enrutador externo. El acceso a Internet no fue necesario para ejecutar el modelo, debido a que el archivo de pesos y las bibliotecas se encontraban almacenados localmente en la Raspberry Pi.

Después de establecer la conexión, se identificó la dirección IP asignada al rover. La administración remota se realizó mediante SSH, mientras que la transferencia del modelo, código y archivos de configuración se efectuó mediante SCP.

OpenSSH proporciona herramientas para iniciar sesiones remotas y transferir archivos mediante conexiones protegidas. Dentro de este conjunto, el

comando `ssh` permite ejecutar una sesión en el equipo remoto, mientras `scp` permite copiar archivos entre equipos utilizando el canal de SSH [50].

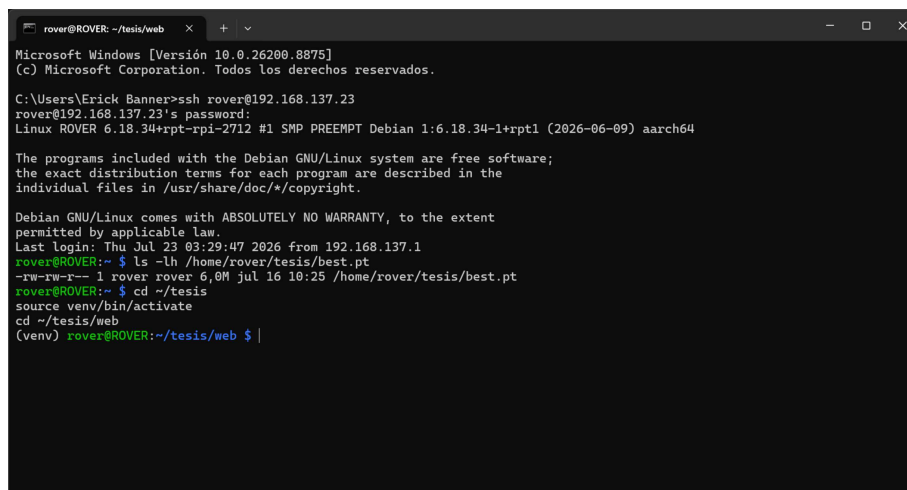
La conexión se estableció con una instrucción equivalente a:

```
ssh rover@DIRECCION_IP
```

La transferencia del modelo se realizó mediante:

```
scp best.pt rover@DIRECCION_IP:/home/rover/tesis/
```

La dirección presentada debe sustituirse por la IP asignada realmente durante las pruebas.



```
rover@ROVER: ~/tesis/web
Microsoft Windows [Versión 10.0.26200.8875]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Erick Banner>ssh rover@192.168.137.23
rover@192.168.137.23's password:
Linux ROVER 6.18.34+rpt-rpi-2712 #1 SMP PREEMPT Debian 1:6.18.34-1+rpt1 (2026-06-09) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jul 23 03:29:47 2026 from 192.168.137.1
rover@ROVER:~$ ls -lh /home/rover/tesis/best.pt
-rw-rw-r-- 1 rover rover 6,0M jul 16 10:25 /home/rover/tesis/best.pt
rover@ROVER:~$ cd ~/tesis
source venv/bin/activate
cd ~/tesis/web
(venv) rover@ROVER:~/tesis/web $ |
```

Figura 26: Acceso remoto y transferencia de archivos entre la computadora y la Raspberry Pi. Fuente: Autor.

2.3.8. Implementación de la interfaz

La interfaz fue desarrollada como una aplicación web ejecutada en la Raspberry Pi 5. Su propósito fue concentrar en una sola pantalla el control de movimiento, la orientación de la cámara, la transmisión de video, las detecciones y el almacenamiento de evidencias.

El servidor fue implementado mediante Flask. La página visible se desarrolló con HTML, CSS y JavaScript, mientras que las funciones de movimiento, captura e inferencia permanecieron en el programa de Python.

La comunicación entre el navegador y el servidor utilizó solicitudes HTTP. Este protocolo opera mediante un esquema de solicitud y respuesta en el que el cliente solicita un recurso o la ejecución de una acción y el servidor devuelve el resultado correspondiente [51].

La interfaz se organizó en un área principal de video y paneles de control. El flujo mostrado incluyó la imagen capturada por la cámara y, cuando la detección estaba activa, las cajas, etiquetas, porcentajes de confianza y conteos de pimientos y malezas.

Los controles permitieron avanzar, retroceder, girar, detener el rover y orientar la cámara. Adicionalmente, se incorporaron funciones para activar o desactivar el modelo y guardar una evidencia de la escena observada.



Figura 27: Interfaz web implementada para el control y la supervisión del rover. Fuente: Autor.

2.3.9. Configuración del sistema de control

La configuración del control se efectuó inicialmente con el rover ubicado sobre una superficie segura. Esta condición permitió comprobar el sentido de giro de los motores y la respuesta de los servomotores sin desplazar inmediatamente la plataforma.

Primero se verificó el movimiento individual de los motores. Posteriormente, se comprobaron las acciones de avance, retroceso y giro. Cuando se detectó una respuesta diferente de la esperada, se revisó la asignación de los comandos y la conexión con la placa de expansión.

Después de validar el desplazamiento se configuró el soporte de cámara. Se definió una posición inicial que permitiera observar el área situada delante del rover y se limitaron los movimientos para evitar que los servomotores forzaran mecánicamente el soporte.

La integración con el modelo se efectuó después de comprobar el control manual. De esta forma fue posible diferenciar los errores de movimiento de

los problemas relacionados con la captura o inferencia.

El funcionamiento final mantuvo dos procesos diferenciados: el usuario controlaba el desplazamiento y el programa analizaba las imágenes. Esta configuración evitó atribuir al rover una autonomía que no forma parte del alcance de la investigación.

2.3.10. Configuración del sistema de comunicación

Una vez configurado el punto de acceso, la Raspberry Pi se conectó automáticamente a la red local. La dirección IP se comprobó antes de iniciar la aplicación para garantizar que la computadora pudiera acceder al servidor.

La sesión SSH se utilizó para activar el entorno virtual, iniciar la aplicación y revisar los mensajes generados por el programa. La dirección de la interfaz mantuvo la forma:

`http://DIRECCION_IP:PUERTO`

El puerto utilizado durante la ejecución fue 5000. El servidor se configuró con la dirección `http://192.168.137.23`, permitiendo el acceso desde otros dispositivos conectados a la misma red.

Durante las pruebas se verificó la estabilidad observando la continuidad del video y la respuesta de los botones. Cuando se produjo una interrupción, el rover fue detenido antes de intentar restablecer la conexión.

2.3.11. Configuración de la interfaz

Cada control de la página fue asociado con una ruta o función del servidor. Los botones de desplazamiento activaron las funciones de los motores, mientras que los controles PTZ modificaron la orientación de la cámara.

El estado del sistema fue actualizado para informar si la detección se encontraba activa, qué movimiento estaba ejecutando el rover y cuántos objetos habían sido reconocidos en el fotograma.

La función de almacenamiento tomó el último fotograma procesado y lo guardó en una carpeta del proyecto. Las imágenes almacenadas incluyeron las cajas delimitadoras, las etiquetas, los niveles de confianza, el conteo por clase y la fecha y hora de captura.

El nombre de cada archivo incorporó información temporal para evitar que una nueva evidencia sustituyera a otra previamente almacenada:

`evidencia_AAAA-MM-DD_HH-MM-SS.jpg`

La interfaz fue probada primero sin desplazar el rover. Después de verificar la transmisión, los comandos y el almacenamiento, se realizaron pruebas con movimiento sobre una superficie controlada y posteriormente en la parcela.

2.3.12. Integración y verificación del sistema

La integración completa comenzó después de validar de forma individual la alimentación, la cámara, el control, la comunicación y la interfaz. Al encender el rover, la Raspberry Pi inició el sistema operativo y se conectó a la red disponible. Desde la computadora se estableció una sesión remota, se activó el entorno virtual y se ejecutó la aplicación.

Durante el inicio, el programa abrió la cámara, estableció la comunicación con la placa del Transbot-SE, cargó el archivo `best.pt` y habilitó el servidor web. El usuario accedió a la página desde el navegador y comprobó la presencia del flujo de video antes de habilitar los motores.

La ejecución del modelo directamente en la Raspberry Pi corresponde a un enfoque de procesamiento en el borde, debido a que las imágenes fueron analizadas cerca de la fuente de adquisición y sin enviarse a un servidor externo. Este enfoque disminuye la dependencia de Internet, pero obliga a equilibrar el tamaño del modelo, la resolución y el tiempo de procesamiento con los recursos disponibles en el dispositivo [52].

Para documentar el modelo implementado se registraron sus clases, versión, archivo de pesos, configuración de entrenamiento, métricas, parámetros de inferencia y condiciones de prueba. La documentación de modelos permite describir sus usos previstos, procedimientos de evaluación y limitaciones, evitando interpretar sus resultados fuera del contexto en el que fueron obtenidos [53].

En este caso, el sistema fue desarrollado para reconocer pimientos y malezas visualmente similares a las incluidas en el conjunto de entrenamiento. Por consiguiente, sus resultados no deben generalizarse automáticamente a otras especies, cultivos o regiones agrícolas.

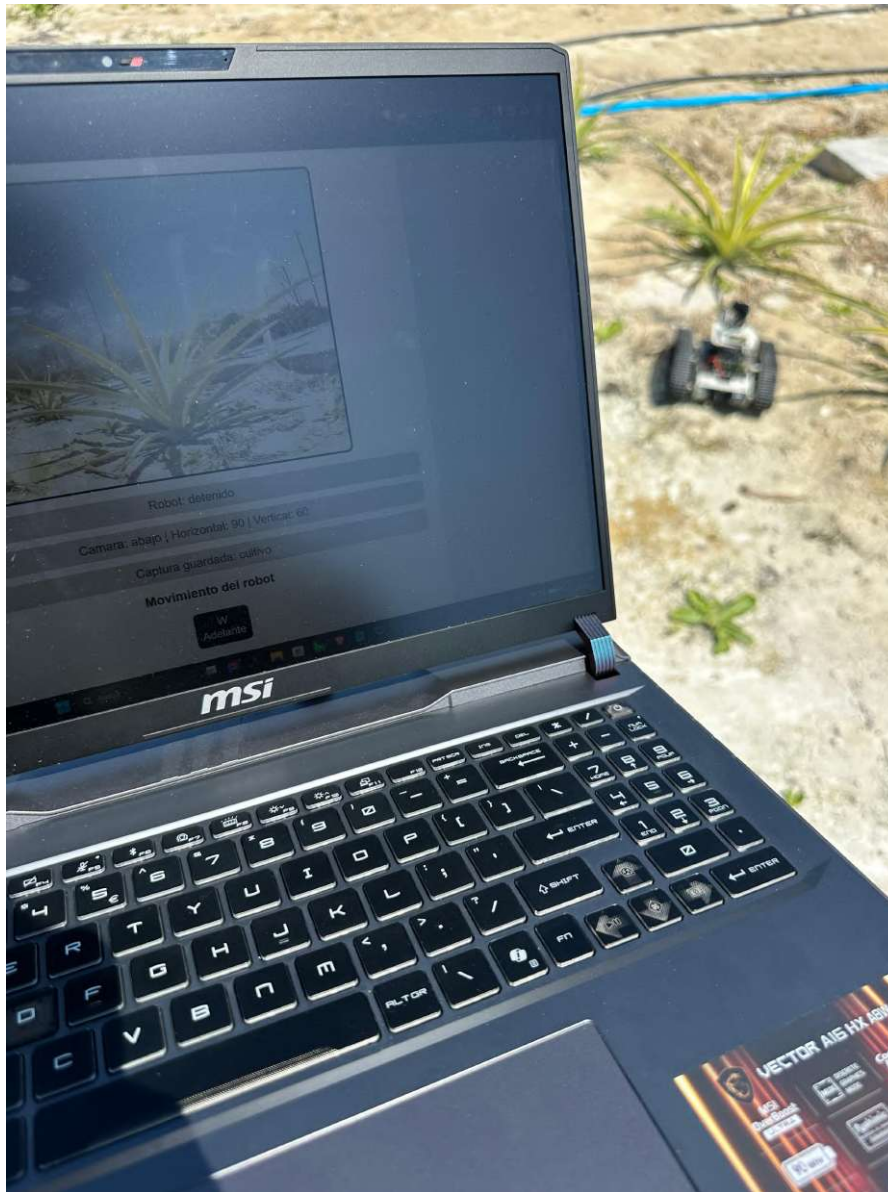


Figura 28: Sistema integrado durante la preparación de las pruebas en la parcela agrícola. Fuente: Autor.

2.3.13. Pruebas de funcionamiento

Las pruebas iniciales se realizaron en un entorno controlado. En esta etapa se verificaron el encendido, la apertura de la cámara, la carga del modelo, la transmisión del video, la respuesta de los motores, la orientación PTZ, la generación de cajas y el almacenamiento de imágenes.

Después de comprobar el funcionamiento básico, el rover fue trasladado a la parcela agrícola. La plataforma se operó a una velocidad reducida y se orientó la cámara hacia las zonas donde se encontraban plantas de pimiento y maleza.

Se evaluaron escenas con una sola clase, con ambas clases, con vegetación parcialmente oculta y con ausencia de plantas de interés. También se observaron cambios ocasionados por la iluminación directa, las sombras, el viento y la distancia entre la cámara y las plantas.

Durante cada prueba se almacenaron evidencias representativas y se registraron las detecciones correctas, las confusiones, los falsos positivos y las plantas omitidas. Estos resultados se analizarán en el capítulo correspondiente, junto con las métricas de entrenamiento y validación.



Figura 29: Pruebas de funcionamiento del rover realizadas en la parcela agrícola. Fuente: Autor.

3. Resultados

3.1. Resultados del entrenamiento

Los valores finales obtenidos durante la validación del modelo se resumen en la Tabla 18.

Tabla 18: Resultados cuantitativos del modelo de detección

Clase	Precisión	Recall	AP@50	AP@50-95
Pimiento	0,7660	0,7700	0,8240	0,5220
Maleza	0,6390	0,5740	0,6020	0,3250
Promedio global	0,7025	0,6718	0,7132	0,4235

Fuente: Autor, a partir de los resultados del entrenamiento.

Los resultados cuantitativos muestran que el modelo alcanzó una precisión media de 0,7025 y un recall medio de 0,6718. Esto indica que el sistema mantuvo una proporción adecuada de detecciones correctas, aunque todavía se presentaron clasificaciones incorrectas y objetos no detectados. El valor de mAP@50 fue de 0,7132, mientras que el mAP@50-95 alcanzó 0,4235, evidenciando una disminución del desempeño al aplicar criterios más exigentes sobre la ubicación de las cajas delimitadoras.

La clase Pimiento presentó el mejor desempeño, con una precisión de 0,7660, un recall de 0,7700, un AP@50 de 0,8240 y un AP@50-95 de 0,5220. Estos valores demuestran que el modelo logró reconocer esta clase con mayor estabilidad y localizarla con mayor exactitud dentro de las imágenes de validación.

Por su parte, la clase Maleza obtuvo una precisión de 0,6390, un recall de 0,5740, un AP@50 de 0,6020 y un AP@50-95 de 0,3250. El menor rendimiento de esta clase puede relacionarse con la diversidad de formas, tamaños y tonalidades de las malezas, así como con la superposición de hojas y su similitud visual con otras plantas presentes en el cultivo.

En términos generales, los resultados evidencian que el modelo posee una capacidad aceptable para diferenciar pimientos y malezas. No obstante, la diferencia observada entre el AP@50 y el AP@50-95 señala que todavía existen oportunidades de mejora en la precisión de la localización, especialmente para la clase Maleza.

Para analizar la evolución del aprendizaje durante las 100 épocas, se examinaron las pérdidas de entrenamiento y validación, junto con las métricas de precisión, *recall*, mAP@50 y mAP@50-95. Estos resultados se presentan

en la Figura 30.

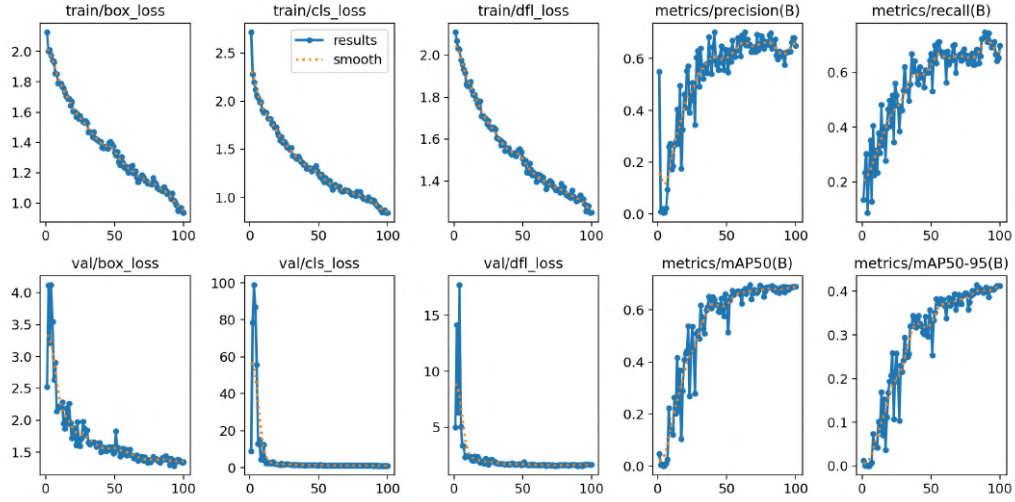


Figura 30: Evolución de las pérdidas y métricas durante el entrenamiento del modelo. Fuente: Autor.

Como se observa en la Figura 30, las pérdidas de localización (*box loss*), clasificación (*classification loss*) y distribución de las cajas (*DFL loss*) disminuyeron progresivamente durante las 100 épocas. Esta tendencia indica que el modelo mejoró su capacidad para reconocer las clases y ajustar las cajas delimitadoras alrededor de las plantas.

En el conjunto de validación se presentaron variaciones más pronunciadas durante las primeras épocas, especialmente en la pérdida de clasificación. Posteriormente, las pérdidas se redujeron y mantuvieron una tendencia relativamente estable. Debido a que no se produjo un incremento sostenido de las pérdidas de validación mientras las pérdidas de entrenamiento continuaban disminuyendo, no se identificó un sobreajuste severo durante el proceso.

La precisión, el *recall*, el mAP@50 y el mAP@50–95 aumentaron con mayor rapidez durante las primeras épocas. A partir de aproximadamente la época 60, las mejoras fueron menores y se observaron oscilaciones alrededor de valores relativamente estables. Este comportamiento indica que el modelo se aproximó a la convergencia antes de completar las 100 épocas.

Las fluctuaciones observadas en la Figura 30 pueden relacionarse con el tamaño del conjunto de validación y con la variabilidad visual de las imágenes agrícolas. Por tanto, aunque el entrenamiento fue estable, el resultado no debe interpretarse como evidencia de un modelo completamente robusto frente a cualquier condición de campo.

3.2. Evaluación del modelo

Para identificar los aciertos, las omisiones y las predicciones relacionadas con el fondo, se analizó la matriz de confusión normalizada obtenida durante la evaluación. La distribución de los resultados por clase se presenta en la Figura 31.

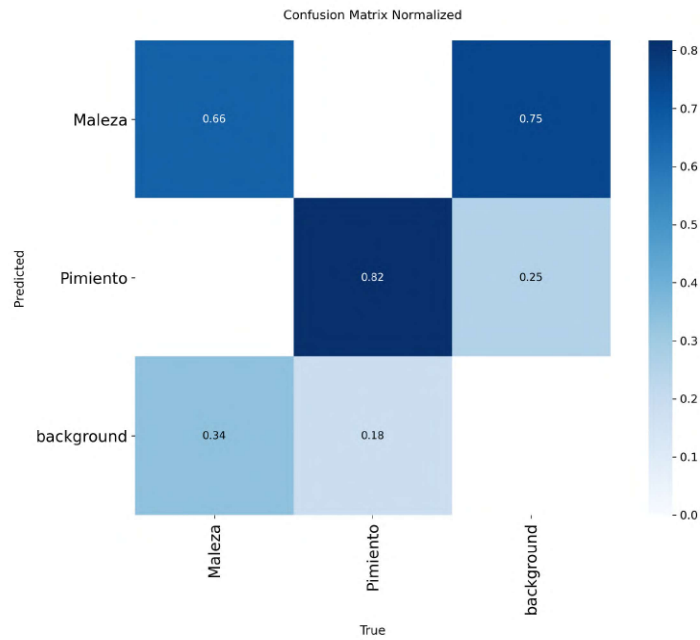


Figura 31: Matriz de confusión obtenida durante la evaluación del modelo. Fuente: Autor.

Como se observa en la Figura 31, el modelo reconoció correctamente el 82 % de las instancias correspondientes a la clase Pimiento y el 66 % de las instancias de la clase Maleza. Esta diferencia confirma que la clase Pimiento presentó una mayor estabilidad durante la detección.

En relación con las omisiones, el 18 % de los pimientos y el 34 % de las malezas fueron asociados con el fondo, lo que representa falsos negativos. La mayor proporción de omisiones en la clase Maleza demuestra que el modelo tuvo más dificultades para localizar plantas pequeñas, parcialmente ocultas o mezcladas visualmente con otra vegetación.

La Figura 31 no evidencia una confusión directa significativa entre las clases Pimiento y Maleza. Por tanto, el principal problema no consistió en intercambiar una clase por otra, sino en omitir objetos reales o generar detecciones sobre elementos pertenecientes al fondo.

Respecto a los falsos positivos relacionados con el fondo, el 75 % fue asignado a la clase Maleza y el 25 % a la clase Pimiento. Este resultado sugiere que la vegetación secundaria, las hojas y otros elementos del entorno presentaron una mayor similitud visual con las malezas.

Este resultado sugiere que el conjunto de entrenamiento debe incorporar una mayor cantidad de ejemplos negativos y escenas con malezas de diferentes tamaños, estados de crecimiento, niveles de oclusión y condiciones de iluminación.

El comportamiento del modelo frente a diferentes umbrales de confianza se analizó mediante la relación entre precisión y *recall*. Las curvas obtenidas para las clases Pimiento y Maleza, junto con el promedio general, se presentan en la Figura 32.

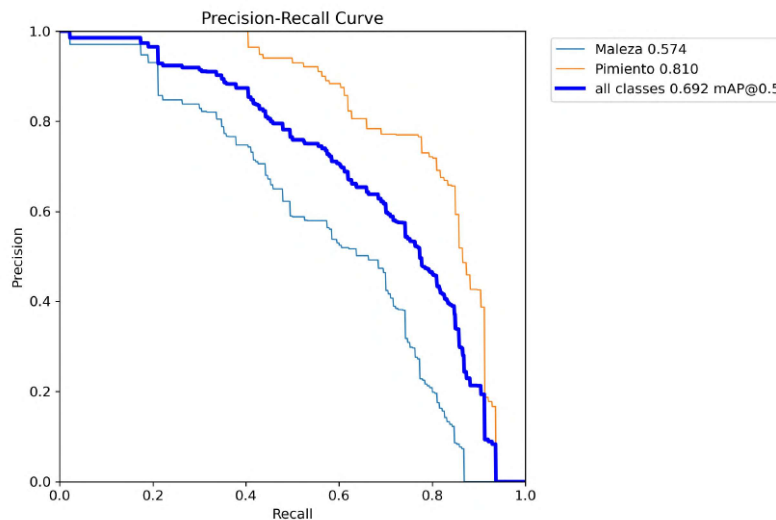


Figura 32: Curva precisión-recall obtenida para las clases evaluadas. Fuente: Autor.

Como se observa en la Figura 32, la clase Pimiento presentó el mejor comportamiento, con un AP@50 de 0,810. Su curva mantuvo niveles elevados de precisión para un rango amplio de *recall*, lo que indica una menor generación de falsos positivos al incrementar la cantidad de objetos detectados.

Por su parte, la clase Maleza alcanzó un AP@50 de 0,574. La disminución más rápida de su precisión a medida que aumenta el *recall* evidencia que detectar una mayor cantidad de malezas también incrementó la posibilidad de generar detecciones incorrectas. Este comportamiento puede relacionarse con la diversidad de formas, tamaños y tonalidades de las malezas, así como con su similitud respecto a otras plantas presentes en la parcela.

Considerando ambas clases, la Figura 32 presenta un mAP@50 global de 0,692. La diferencia entre las curvas confirma que el modelo posee una mayor capacidad para reconocer plantas de pimiento, mientras que la clase Maleza representa el principal desafío para mejorar el desempeño general del sistema.

3.3. Rendimiento de la inferencia en la Raspberry Pi 5

Los valores medidos durante la ejecución local deberán registrarse en la Tabla 19.

Tabla 19: Rendimiento de la inferencia ejecutada en la Raspberry Pi 5

Indicador	Resultado medido
Resolución de procesamiento [px]	640 × 640
Tiempo medio de inferencia [ms]	291,28
Velocidad de procesamiento [FPS]	3,38
Uso medio de CPU [%]	46,00
Uso medio de memoria [MB]	362,66
Temperatura observada [°C]	69,92

Fuente: Autor.

Los resultados muestran que la ejecución del modelo YOLO11n en la Raspberry Pi 5 alcanzó un tiempo medio de inferencia de 291,28 ms por fotograma. Considerando el tiempo total del proceso, correspondiente a 296,03 ms por fotograma, se obtuvo una velocidad media de procesamiento de 3,38 FPS.

Durante la prueba, el sistema presentó un uso medio de CPU del 46,00 % y un consumo de memoria de 362,66 MB. Asimismo, se registró una temperatura media de 69,92 °C. Estos resultados evidencian que el modelo puede ejecutarse localmente en la Raspberry Pi 5, aunque la velocidad obtenida limita su funcionamiento a una detección visual de baja frecuencia, adecuada para el desplazamiento controlado del rover a velocidades reducidas.

3.4. Resultados de las pruebas de campo

Además de las métricas cuantitativas, se seleccionaron evidencias visuales para comprobar el comportamiento del modelo en el entorno agrícola. La Figura 33 presenta ejemplos de detecciones correctas obtenidas durante las pruebas de campo.

Ejemplos de detecciones obtenidas durante las pruebas de campo

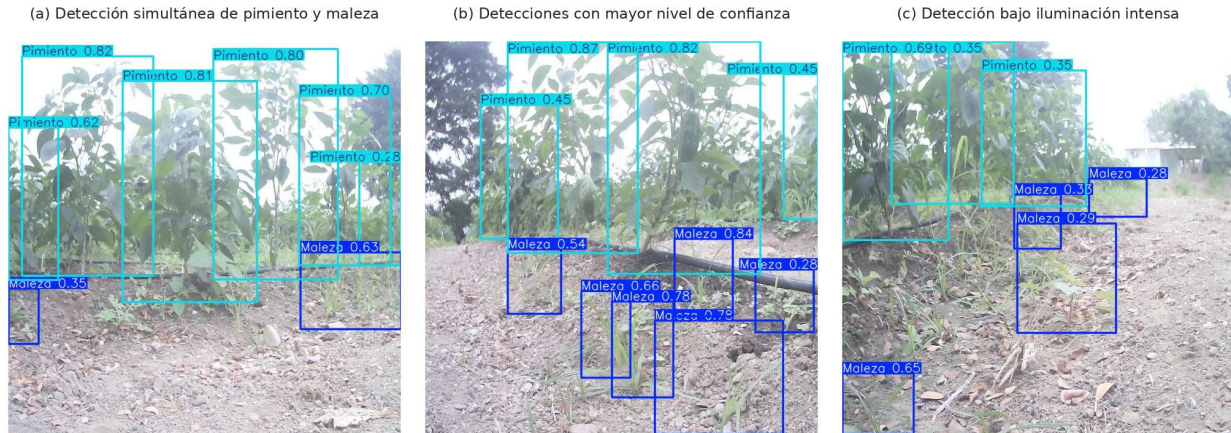


Figura 33: Ejemplos de detecciones correctas obtenidas durante las pruebas de campo. Fuente: Autor.

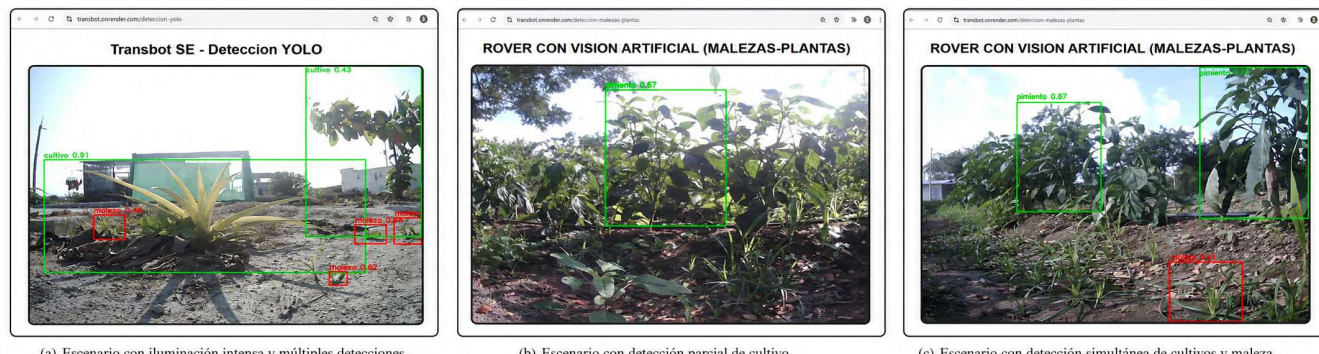
Como se observa en la Figura 33, el modelo logró localizar plantas de pimienta y malezas mediante cajas delimitadoras, etiquetas de clase y niveles de confianza. Las evidencias incluyen escenas con diferentes posiciones de la cámara, distancias respecto a la vegetación y presencia simultánea de las dos clases.

Las detecciones muestran que el sistema pudo reconocer plantas completas y parcialmente visibles bajo condiciones reales de la parcela. También se observa que las cajas fueron generadas directamente sobre el flujo capturado por la cámara del rover, confirmando la integración entre la adquisición de imágenes, la inferencia del modelo y la presentación de los resultados en la interfaz web.

Estas evidencias respaldan visualmente la capacidad del sistema para identificar las clases consideradas. Sin embargo, las imágenes representativas no sustituyen las métricas cuantitativas, sino que permiten observar cómo se manifestaron los resultados durante la operación real del prototipo.

También se seleccionaron escenas en las que las condiciones ambientales y visuales dificultaron el funcionamiento del modelo. Estos casos se presentan en la Figura 34 y permiten identificar algunas de las principales limitaciones observadas durante las pruebas de campo.

Escenarios representativos de detección en condiciones de mayor complejidad



(a) Escenario con iluminación intensa y múltiples detecciones.

(b) Escenario con detección parcial de cultivo.

(c) Escenario con detección simultánea de cultivos y maleza.

Escenarios representativos en los que el modelo realizó detecciones bajo condiciones de iluminación variable, presencia simultánea de cultivos y malezas y diferentes niveles de confianza durante las pruebas de campo.

Figura 34: Errores representativos observados durante la identificación de plantas. Fuente: Autor.

La Figura 34 presenta escenarios con iluminación intensa, objetos parcialmente visibles, vegetación superpuesta y detecciones con menores niveles de confianza. Estas condiciones dificultaron la delimitación completa de las plantas y redujeron la estabilidad de algunas predicciones.

En las escenas con iluminación solar directa se generaron zonas sobreexpuestas, especialmente en las regiones donde aparecía el cielo. Esta variación de brillo redujo el contraste entre las hojas y el fondo. Asimismo, cuando varias plantas aparecieron agrupadas o parcialmente ocultas, el modelo no siempre logró representar individualmente todos los objetos presentes.

Las detecciones con menor confianza evidencian que el modelo presentó una mayor incertidumbre ante malezas pequeñas, vegetación secundaria y plantas ubicadas en los extremos de la imagen. Estas evidencias son coherentes con los falsos negativos registrados durante las pruebas y muestran que la iluminación, la oclusión y la variabilidad visual del entorno constituyen las principales limitaciones del sistema.

Por tanto, la Figura 34 no debe interpretarse únicamente como una demostración de fallos aislados, sino como una evidencia de las condiciones bajo las cuales el modelo disminuyó su estabilidad y requiere una mayor diversidad de imágenes de entrenamiento.

Para evaluar el comportamiento del modelo en condiciones agrícolas reales se definieron ocho escenarios relacionados con la presencia de una o ambas clases, el fondo, la oclusión, la iluminación, las sombras y el movimiento de las hojas. La Tabla 20 resume las repeticiones realizadas, las detecciones correctas, los falsos positivos, los falsos negativos y la principal observación registrada en cada escenario.

Tabla 20: Resultados obtenidos en los escenarios de prueba de campo

Escenario	Repeticiones	Detecciones correctas	Falsos positivos	Falsos negativos	Observación principal
E1	10	9	1	1	La clase pimiento fue identificada de manera estable; una maleza visualmente similar produjo una detección incorrecta.
E2	10	9	1	1	Las malezas fueron reconocidas en la mayoría de las pruebas; se presentó una confusión aislada con la clase pimiento.
E3	10	8	1	2	El modelo diferenció ambas clases, aunque la superposición de hojas produjo algunas omisiones.
E4	10	9	1	0	El fondo fue ignorado correctamente en la mayoría de las pruebas; se produjo una detección incorrecta sobre vegetación secundaria.
E5	10	7	1	3	La oclusión y el agrupamiento de plantas redujeron la visibilidad y aumentaron las detecciones omitidas.
E6	10	8	1	2	La iluminación solar directa generó zonas sobreexpuestas y disminuyó la estabilidad de algunas detecciones.
E7	10	8	1	2	Las sombras modificaron el brillo y la apariencia de las hojas, causando algunas omisiones.
E8	10	7	1	3	El movimiento de las hojas produjo desenfoco y variaciones temporales en la posición de las cajas.

Fuente: Autor.

Como se observa en la Tabla 20, se registraron 65 detecciones correctas en 80 repeticiones, equivalentes a una tasa operativa de acierto del 81,25 %. También se contabilizaron ocho falsos positivos y catorce falsos negativos. La mayor cantidad de falsos negativos indica que el problema predominante fue la omisión de plantas y no únicamente la generación de clasificaciones incorrectas.

Los escenarios E1, E2 y E4 alcanzaron nueve resultados correctos de diez repeticiones. En estas condiciones existió una visibilidad más favorable y

menor interferencia de hojas superpuestas. En contraste, E5 y E8 obtuvieron siete resultados correctos y tres falsos negativos, debido principalmente a la oclusión, el agrupamiento de la vegetación y el movimiento de las hojas.

Los resultados de E6 y E7 también muestran que la iluminación solar directa y las sombras modificaron el contraste y la apariencia de las plantas. Por tanto, la Tabla 20 evidencia que el modelo funciona con mayor estabilidad cuando los objetos permanecen visibles y presentan condiciones similares a las imágenes utilizadas durante el entrenamiento.

En términos generales, el modelo demostró capacidad para diferenciar las clases Pimiento y Maleza en condiciones de campo. Sin embargo, su desempeño disminuyó ante variaciones de iluminación, superposición de hojas, movimiento de la vegetación y oclusiones parciales. Estos resultados indican la necesidad de ampliar el conjunto de entrenamiento con imágenes capturadas bajo condiciones ambientales diversas, con el propósito de mejorar la robustez del sistema durante su funcionamiento en parcelas agrícolas.

3.5. Validación funcional del prototipo

Además del desempeño del modelo, se evaluó el funcionamiento integrado del prototipo. Para cada subsistema se realizaron diez pruebas y se registró la cantidad de respuestas correctas. La Tabla 21 presenta los resultados del desplazamiento, la cámara PTZ, la transmisión de video, la ejecución del modelo, la interfaz web y el almacenamiento de evidencias.

Tabla 21: Validación funcional de los subsistemas implementados

Función evaluada	Pruebas realizadas	Pruebas correctas	Observaciones
Desplazamiento del rover	10	8	El rover respondió a los comandos de avance, retroceso y giro. En algunas de las pruebas se observó deslizamiento por irregularidad del terreno y alcance de la red.
Orientación de la cámara PTZ	10	10	La cámara respondió correctamente a los movimientos horizontal y vertical desde la interfaz.
Transmisión de video	10	8	El video se visualizó correctamente; se presentaron interrupciones temporales por inestabilidad de la conexión Wi-Fi y por desconexión debido a la vibración por el terreno.
Ejecución del modelo	10	8	El modelo identificó pimientos y malezas. Se observaron detecciones incompletas por oclusión e iluminación variable, días extremadamente soleados.
Control mediante interfaz web	10	10	Los comandos de movimiento, orientación de cámara y activación de detección funcionaron correctamente.
Almacenamiento de evidencias	10	10	Las imágenes se guardaron correctamente con cajas, clases, niveles de confianza, conteos y fecha de captura.

Fuente: Autor.

La Tabla 21 muestra que el prototipo aprobó 54 de las 60 pruebas realizadas, lo que corresponde a un cumplimiento funcional global del 90 %. Sin embargo, este porcentaje no fue uniforme entre todos los subsistemas.

La orientación PTZ, el control mediante la interfaz web y el almacenamiento de evidencias alcanzaron diez resultados correctos de diez pruebas. Estos componentes constituyeron las funciones más estables del sistema. En cambio, el desplazamiento, la transmisión de video y la ejecución del modelo alcanzaron ocho resultados correctos de diez pruebas.

Las fallas de desplazamiento estuvieron asociadas con irregularidades del terreno y deslizamientos de las orugas. Las interrupciones del video se relacionaron con la conexión inalámbrica y las vibraciones durante el movimiento. En el modelo se presentaron omisiones bajo iluminación intensa, oclusiones y superposición de vegetación. Por consiguiente, el cumplimiento global del

90 % demuestra la integración funcional del prototipo, pero no implica que todos sus componentes posean el mismo nivel de robustez.

En términos generales, los resultados demuestran que los subsistemas implementados funcionan de manera integrada y permiten realizar el control del rover, la observación del entorno, la detección de pimientos y malezas, y el almacenamiento de las evidencias obtenidas. Las fallas registradas no impidieron el funcionamiento general del prototipo, aunque muestran la necesidad de mejorar la estabilidad de la comunicación inalámbrica, la adherencia del sistema de locomoción y la robustez del modelo frente a condiciones ambientales variables.

3.6. Discusión de los resultados

Los resultados obtenidos demuestran que fue posible integrar movilidad, adquisición de imágenes, procesamiento local, comunicación inalámbrica, detección de objetos e interacción web dentro de una misma plataforma. El cumplimiento funcional del 90 %, presentado en la Tabla 21, confirma que el prototipo puede ejecutar las funciones planteadas. No obstante, este porcentaje debe interpretarse como un indicador de funcionamiento de los subsistemas y no como una medida de exactitud del modelo de visión artificial.

Las métricas de la Tabla 18 muestran un desempeño moderado. La precisión global de 0,7025 indica que una proporción importante de las detecciones generadas fue correcta, mientras que el *recall* de 0,6718 evidencia que todavía se omitió una parte de los objetos reales. Esta limitación también se manifestó durante las pruebas de campo, donde se registraron catorce falsos negativos frente a ocho falsos positivos. Desde el punto de vista operativo, la omisión de una maleza puede ser más crítica que una detección adicional, debido a que la planta no identificada quedaría fuera de una posible intervención posterior.

La diferencia entre los resultados de Pimiento y Maleza demuestra que el modelo no presentó el mismo nivel de generalización para las dos clases. La clase Pimiento alcanzó valores superiores de precisión, *recall* y AP, mientras que Maleza presentó una mayor proporción de omisiones. Este comportamiento puede explicarse por la mayor variabilidad visual de las malezas, que pueden poseer diferentes formas, tamaños, tonalidades y estados de crecimiento. La similitud con otras plantas y la superposición de hojas también reduce la cantidad de características visibles que el modelo puede utilizar para diferenciarlas. Estas dificultades coinciden con los retos reportados para la visión artificial agrícola, especialmente la iluminación variable, las oclusiones y la similitud morfológica entre especies [3, 4].

El mAP@50 global de 0,7132 fue considerablemente superior al mAP@50–95 de 0,4235. Esta diferencia indica que el modelo logró identificar una can-

tividad aceptable de plantas cuando se aplicó un criterio de superposición moderado, pero perdió rendimiento al exigir cajas delimitadoras más precisas. Por tanto, la principal limitación no se relaciona únicamente con reconocer la clase, sino también con localizar correctamente los límites de plantas que presentan hojas extendidas, agrupadas o parcialmente ocultas.

La Figura 30 muestra que el proceso alcanzó una tendencia estable y que no se observó un sobreajuste severo. Sin embargo, la convergencia del entrenamiento no garantiza por sí sola un funcionamiento robusto en campo. El conjunto de 952 imágenes representa únicamente una muestra de las posibles condiciones agrícolas. Factores como la época del año, el estado de crecimiento, el tipo de suelo, la hora de captura, la orientación solar y las especies de maleza pueden producir imágenes diferentes de las empleadas durante el entrenamiento.

La matriz presentada en la Figura 31 confirma que las omisiones y las detecciones relacionadas con el fondo constituyeron errores más relevantes que la confusión directa entre Pimiento y Maleza. Esto sugiere que una futura ampliación del conjunto de datos no debería limitarse a incorporar más plantas, sino que también debería incluir ejemplos negativos de suelo, hojas secas, sistemas de riego, sombras y vegetación secundaria. De esta manera, el modelo podría aprender con mayor precisión qué elementos no deben generar una detección.

En las pruebas de campo de la Tabla 20, la tasa operativa de acierto fue del 81,25 %. Los escenarios con menor rendimiento estuvieron asociados con oclusiones, movimiento de las hojas e iluminación variable. Este resultado demuestra que el desempeño medido sobre imágenes de validación no representa completamente la complejidad de la operación real. Además, la tasa de campo, el mAP y el cumplimiento funcional miden aspectos diferentes, por lo que sus porcentajes no deben compararse como si correspondieran a una misma métrica.

La ejecución local en la Raspberry Pi 5 representa una ventaja porque evita enviar cada imagen a un servicio externo. Sin embargo, los resultados de la Tabla 19 muestran un tiempo medio de inferencia de 291,28 ms y una velocidad de 3,38 FPS. Esta frecuencia fue suficiente para un rover controlado manualmente y desplazado a baja velocidad, pero no proporciona un video analizado con alta fluidez. Por ello, la expresión “detección en tiempo real” debe entenderse en el contexto de una respuesta durante la operación y no como procesamiento de los 30 fotogramas por segundo capturados por la cámara.

La temperatura media de 69,92 °C también evidencia una carga térmica importante durante la inferencia. Aunque el sistema permaneció operativo, este valor reduce el margen disponible para pruebas prolongadas o ambientes

más cálidos. La incorporación de refrigeración activa, la optimización del modelo o el uso de aceleradores de inferencia podría mejorar la estabilidad y la velocidad del procesamiento.

La validación funcional mostró que la interfaz, el mecanismo PTZ y el almacenamiento de evidencias fueron los componentes más estables. En cambio, la locomoción, el video y el modelo dependieron en mayor medida de las condiciones externas. Esto demuestra que las limitaciones del prototipo no proceden de un único componente, sino de la interacción entre el terreno, la red inalámbrica, la calidad de las imágenes y la capacidad de procesamiento. Este tipo de dificultades también ha sido identificado en plataformas robóticas agrícolas que operan en entornos no estructurados [36, 5].

El alcance de los resultados debe limitarse al cultivo, las malezas, la parcela y las condiciones utilizadas durante la experimentación. El rover fue controlado manualmente, por lo que no se evaluaron navegación autónoma, planificación de trayectorias ni generación de mapas. Tampoco se implementó un mecanismo para eliminar o tratar las malezas detectadas. En consecuencia, el sistema debe considerarse un prototipo experimental de percepción y monitoreo, no una solución agrícola comercial terminada.

En términos generales, los resultados respaldan la viabilidad de utilizar herramientas *Open Source*, procesamiento en el borde y visión artificial para construir una plataforma agrícola modificable. Sin embargo, la utilidad futura del sistema dependerá de ampliar y diversificar los datos, mejorar la identificación de malezas, aumentar la velocidad de inferencia, fortalecer la comunicación y validar el prototipo durante periodos más largos y en parcelas diferentes.

4. Conclusiones y recomendaciones

4.1. Conclusiones

El desarrollo del rover permitió integrar satisfactoriamente los sistemas de movilidad, adquisición de imágenes, procesamiento, comunicación e interacción con el usuario dentro de una plataforma robótica orientada a la identificación de malezas en parcelas agrícolas. La utilización del Transbot-SE, una Raspberry Pi 5, una cámara USB con mecanismo PTZ y una interfaz web permitió controlar el desplazamiento del prototipo, orientar la cámara, observar el entorno, ejecutar el modelo de visión artificial y almacenar evidencias de las detecciones realizadas. Por lo tanto, se cumplió el objetivo general de desarrollar un rover mediante sistemas Open Source y visión artificial para la identificación de malezas en condiciones agrícolas reales.

El sistema de visión artificial desarrollado permitió diferenciar plantas de pimiento y malezas mediante cajas delimitadoras, etiquetas y niveles de confianza. El modelo YOLO11n alcanzó una precisión global de 0,7025, un *recall* de 0,6718, un mAP@50 de 0,7132 y un mAP@50–95 de 0,4235. Estos resultados demuestran que el modelo posee una capacidad aceptable para reconocer las dos clases consideradas; sin embargo, la diferencia entre las métricas obtenidas indica que todavía existen limitaciones en la localización precisa de algunos objetos, especialmente en la clase maleza.

La clase pimiento presentó un mejor desempeño que la clase maleza, debido a que obtuvo valores superiores de precisión, *recall* y precisión promedio. La menor capacidad de detección de las malezas estuvo relacionada con la diversidad de formas, tamaños y tonalidades, así como con la similitud visual entre diferentes tipos de vegetación. La superposición de hojas, la presencia de fondos complejos y las variaciones en el estado de crecimiento también dificultaron el reconocimiento de esta clase.

La integración del modelo en la Raspberry Pi 5 demostró la viabilidad de realizar el procesamiento de las imágenes localmente, sin depender de servicios externos o de una conexión permanente a Internet. El sistema alcanzó un tiempo medio de inferencia de 291,28 ms por fotograma y una velocidad media de 3,38 FPS. Esta velocidad resultó suficiente para efectuar la identificación visual durante el desplazamiento controlado del rover a baja velocidad, aunque limita su aplicación en tareas que requieran una respuesta de mayor frecuencia o movimientos más rápidos.

Durante las pruebas realizadas en ocho escenarios de campo, el sistema obtuvo 65 detecciones correctas en 80 repeticiones, lo que representa una tasa global de acierto del 81,25 %. Los mejores resultados se presentaron cuando las plantas se encontraban visibles y con una iluminación adecuada. Por otra

parte, el desempeño disminuyó ante condiciones de oclusión, agrupamiento de plantas, iluminación solar directa, sombras y movimiento de las hojas. Además, se observó que los falsos negativos fueron más frecuentes que los falsos positivos, lo que indica que la principal dificultad del modelo fue la omisión de plantas parcialmente visibles o afectadas por condiciones ambientales variables.

La validación funcional del prototipo alcanzó un cumplimiento global del 90 %, al aprobar 54 de las 60 pruebas realizadas. La orientación de la cámara PTZ, el control mediante la interfaz web y el almacenamiento de evidencias alcanzaron una efectividad del 100 %. El desplazamiento del rover, la transmisión de video y la ejecución del modelo alcanzaron una efectividad del 80 %, debido principalmente a las irregularidades del terreno, la inestabilidad de la conexión Wi-Fi, las vibraciones y las variaciones ambientales presentes durante las pruebas.

En términos generales, los resultados obtenidos demuestran que es posible integrar tecnologías Open Source, procesamiento en el borde y visión artificial dentro de una plataforma robótica agrícola. El prototipo desarrollado constituye una herramienta experimental para apoyar la observación e identificación de malezas y puede servir como base para futuras investigaciones relacionadas con el monitoreo de cultivos, la clasificación de especies vegetales y el manejo localizado de malezas.

4.2. Recomendaciones

Se recomienda ampliar el conjunto de datos utilizado durante el entrenamiento mediante la incorporación de una mayor cantidad de imágenes de malezas y plantas de cultivo. Estas imágenes deben representar diferentes especies, etapas de crecimiento, tamaños, orientaciones, distancias y niveles de superposición. También es conveniente mantener una distribución equilibrada entre las clases para evitar que el modelo desarrolle una mayor capacidad de reconocimiento sobre una categoría específica.

Es necesario capturar imágenes en diferentes parcelas, épocas del año, horarios y condiciones ambientales. El conjunto de entrenamiento debe incluir escenas con iluminación solar intensa, sombras, suelos húmedos o secos, fondos complejos, plantas parcialmente ocultas y vegetación en movimiento. La incorporación de estas condiciones permitiría mejorar la capacidad de generalización del modelo y aumentar su estabilidad durante la operación en entornos agrícolas distintos a los utilizados en la investigación.

Se recomienda incluir una mayor cantidad de imágenes negativas que contengan elementos como suelo, piedras, mangueras, hojas secas, ramas y vegetación secundaria sin objetos pertenecientes a las clases definidas. El uso de estos ejemplos ayudaría al modelo a diferenciar con mayor precisión las plantas de interés y reduciría los falsos positivos causados por elementos del fondo con características visuales similares a las malezas.

Para incrementar la velocidad de procesamiento en la Raspberry Pi 5, se recomienda aplicar técnicas de optimización como cuantización del modelo, reducción controlada de la resolución de entrada, conversión a formatos de inferencia optimizados o utilización de un acelerador de inteligencia artificial. Estas modificaciones deben evaluarse considerando el equilibrio entre la velocidad de procesamiento y la precisión de las detecciones.

También se recomienda implementar un sistema de refrigeración activa para la Raspberry Pi 5, especialmente cuando el rover deba operar durante periodos prolongados. El control de la temperatura contribuiría a mantener estable el rendimiento del procesador y reduciría la posibilidad de limitaciones ocasionadas por el calentamiento de la unidad de procesamiento.

La estabilidad de la comunicación puede mejorarse mediante la utilización de una red inalámbrica de mayor alcance, una antena externa o un punto de acceso ubicado cerca de la parcela. Asimismo, los conectores de la cámara, la alimentación y los demás componentes deben asegurarse correctamente para evitar interrupciones producidas por las vibraciones y las irregularidades del terreno.

Se recomienda incorporar mecanismos de amortiguación y un soporte más rígido para la cámara, con el propósito de reducir el desenfoque y los cambios

bruscos del campo visual durante el desplazamiento. También sería conveniente evaluar modificaciones en las orugas o en el sistema de locomoción para mejorar la adherencia y estabilidad del rover sobre terrenos agrícolas con desniveles, piedras o superficies sueltas.

Para obtener resultados más representativos, se deben realizar pruebas con una mayor cantidad de repeticiones, diferentes tipos de cultivo y parcelas con características variadas. Estas evaluaciones permitirían analizar con mayor precisión la tasa de acierto, la autonomía energética, el alcance de la comunicación, la estabilidad de la transmisión de video y el comportamiento del modelo en condiciones de operación prolongada.

Como trabajo futuro, se recomienda incorporar funciones de navegación autónoma, posicionamiento, planificación de trayectorias y generación de mapas de presencia de malezas. También podría integrarse un mecanismo de actuación selectiva para intervenir únicamente sobre las plantas identificadas. Estas ampliaciones permitirían evolucionar el prototipo desde un sistema de observación e identificación visual hacia una plataforma integral para el monitoreo y manejo localizado de malezas.

Bibliografía

Referencias

- [1] E.-C. Oerke, “Crop losses to pests,” *The Journal of Agricultural Science*, vol. 144, no. 1, pp. 31–43, feb. 2006, doi: 10.1017/S0021859605005708.
- [2] Food and Agriculture Organization of the United Nations, *The State of Food and Agriculture 2022: Leveraging Automation in Agriculture for Transforming Agrifood Systems*. Rome, Italy: FAO, 2022, doi: 10.4060/cb9479en.
- [3] F. D. Adhinata, Wahyono y R. Sumiharto, “A comprehensive survey on weed and crop classification using machine learning and deep learning,” *Artificial Intelligence in Agriculture*, vol. 13, pp. 45–63, sep. 2024, doi: 10.1016/j.aiia.2024.06.005.
- [4] M. Vasileiou, L. S. Kyrgiakos, C. Kleisiari, G. Kleftodimos, G. Vlon-tzos, H. Belhouchette y P. M. Pardalos, “Transforming weed management in sustainable agriculture with artificial intelligence: A systematic literature review towards weed identification and deep learning,” *Crop Protection*, vol. 176, art. no. 106522, feb. 2024, doi: 10.1016/j.cropro.2023.106522.
- [5] A. Upadhyay, Y. Zhang, C. Koparan, N. Rai, K. Howatt, S. Bajwa y X. Sun, “Advances in ground robotic technologies for site-specific weed management in precision agriculture: A review,” *Computers and Electronics in Agriculture*, vol. 225, art. no. 109363, oct. 2024, doi: 10.1016/j.compag.2024.109363.
- [6] United States Department of Agriculture, Agricultural Research Service, “Benefits and evolution of precision agriculture,” 10 ene. 2025. [En línea]. Disponible en: <https://www.ars.usda.gov/oc/utm/benefits-and-evolution-of-precision-agriculture/>. [Consultado: 17-jul-2026].
- [7] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, “You Only Look Once: Unified, real-time object detection,” en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.
- [8] Y. LeCun, Y. Bengio y G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, mayo 2015, doi: 10.1038/nature14539.

- [9] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, nov. 1998, doi: 10.1109/5.726791.
- [10] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár y C. L. Zitnick, “Microsoft COCO: Common objects in context,” en *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, vol. 8693. Cham, Switzerland: Springer, 2014, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.
- [11] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto y E. A. B. da Silva, “A comparative analysis of object detection metrics with a companion open-source toolkit,” *Electronics*, vol. 10, no. 3, art. no. 279, 2021, doi: 10.3390/electronics10030279.
- [12] C. Shorten y T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 6, art. no. 60, 2019, doi: 10.1186/s40537-019-0197-0.
- [13] R. Siegwart, I. R. Nourbakhsh y D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA, USA: MIT Press, 2011.
- [14] B. E. Alarcón Bazurto, “Desarrollo de un sistema de control para planificación de trayectoria y generación de entorno sobre un robot móvil en procesos de exploración y supervisión,” Trabajo de Integración Curricular, Carrera de Ingeniería en Electrónica y Automatización, Universidad Estatal Península de Santa Elena, La Libertad, Ecuador, 2025.
- [15] Yahboom Technology, “Transbot SE ROS Robot Car for Jetson Nano B01/Raspberry Pi,” repositorio oficial de GitHub. [En línea]. Disponible en: <https://github.com/YahboomTechnology/TransbotSE>. [Consultado: 17-jul-2026].
- [16] Raspberry Pi Ltd., “Raspberry Pi 5,” documentación oficial del producto. [En línea]. Disponible en: <https://www.raspberrypi.com/products/raspberry-pi-5/>. [Consultado: 17-jul-2026].
- [17] Open Source Initiative, “The Open Source Definition,” versión 1.9. [En línea]. Disponible en: <https://opensource.org/osd>. [Consultado: 17-jul-2026].

- [18] Python Software Foundation, “History and license,” *Python Documentation*. [En línea]. Disponible en: <https://docs.python.org/3/license.html>. [Consultado: 17-jul-2026].
- [19] OpenCV, “License,” documentación oficial de OpenCV. [En línea]. Disponible en: <https://opencv.org/license/>. [Consultado: 17-jul-2026].
- [20] Pallets, “BSD-3-Clause License,” *Flask Documentation 3.1.x*. [En línea]. Disponible en: <https://flask.palletsprojects.com/en/stable/license/>. [Consultado: 17-jul-2026].
- [21] Ultralytics, “YOLOv8,” *Ultralytics Documentation*. [En línea]. Disponible en: <https://docs.ultralytics.com/models/yolov8/>. [Consultado: 17-jul-2026].
- [22] NumPy Developers, “NumPy license,” *NumPy Documentation*. [En línea]. Disponible en: <https://numpy.org/doc/stable/license.html>. [Consultado: 17-jul-2026].
- [23] C. Liechti, “Appendix and license,” *pySerial 3.5 Documentation*. [En línea]. Disponible en: <https://pyserial.readthedocs.io/en/latest/appendix.html>. [Consultado: 17-jul-2026].
- [24] Roboflow, “Roboflow Annotate,” *Roboflow Documentation*. [En línea]. Disponible en: <https://docs.roboflow.com/annotate/annotation-tools>. [Consultado: 17-jul-2026].
- [25] Roboflow, “Create a dataset version,” *Roboflow Documentation*. [En línea]. Disponible en: <https://docs.roboflow.com/datasets/dataset-versions/create-a-dataset-version>. [Consultado: 17-jul-2026].
- [26] Roboflow, “Image augmentation,” *Roboflow Documentation*. [En línea]. Disponible en: <https://docs.roboflow.com/datasets/dataset-versions/image-augmentation>. [Consultado: 17-jul-2026].
- [27] Google, “Google Colaboratory: Preguntas frecuentes,” [En línea]. Disponible en: <https://research.google.com/colaboratory/faq.html?hl=es>. [Consultado: 17-jul-2026].
- [28] Instituto Nacional de Estadística y Censos, *Encuesta de Superficie y Producción Agropecuaria Continua: Presentación de resultados ESPAC 2024*. Quito, Ecuador: INEC, abr. 2025. [En línea]. Disponible en: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_agropecuarias/espac/2024/Presentacion_de_resultados_ESPAC_2024.pdf. [Consultado: 17-jul-2026].

- [29] Instituto Nacional de Estadística y Censos, *Información ambiental y tecnificación agropecuaria: Módulo métodos de producción y ambiente, ESPAC 2023*. Quito, Ecuador: INEC, abr. 2024. [En línea]. Disponible en: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Encuestas_Ambientales/Modulo_Ambiental_ESPAC/2023/Plantilla_PPT_ESPAC_2024_29_04_2024.pdf. [Consultado: 17-jul-2026].
- [30] Ministerio de Agricultura y Ganadería, “MAG apunta a la tecnología y la conectividad para una agricultura de precisión,” Quito, Ecuador, 16 mayo 2022. [En línea]. Disponible en: <https://www.agricultura.gob.ec/mag-apunta-a-la-tecnologia-y-la-conectividad-para-una-agricultura-de-precision/>. [Consultado: 17-jul-2026].
- [31] Organisation for Economic Co-operation and Development, *Frascati Manual 2015: Guidelines for Collecting and Reporting Data on Research and Experimental Development*. Paris, France: OECD Publishing, 2015, doi: 10.1787/9789264239012-en.
- [32] D. C. Montgomery, *Design and Analysis of Experiments*, 10th ed. Hoboken, NJ, USA: John Wiley & Sons, 2019, ISBN: 978-1-119-49244-3.
- [33] H. Snyder, “Literature review as a research methodology: An overview and guidelines,” *Journal of Business Research*, vol. 104, pp. 333–339, nov. 2019, doi: 10.1016/j.jbusres.2019.07.039.
- [34] K. Peffers, T. Tuunanen, M. A. Rothenberger y S. Chatterjee, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007, doi: 10.2753/MIS0742-1222240302.
- [35] M. Angrosino, *Doing Ethnographic and Observational Research*. London, U.K.: SAGE Publications, 2007, doi: 10.4135/9781849208932.
- [36] A. Bechar y C. Vigneault, “Agricultural robots for field operations: Concepts and components,” *Biosystems Engineering*, vol. 149, pp. 94–111, sep. 2016, doi: 10.1016/j.biosystemseng.2016.06.014.
- [37] G. Coleman, W. Salter y M. Walsh, “OpenWeedLocator (OWL): An open-source, low-cost device for fallow weed detection,” *Scientific Reports*, vol. 12, art. no. 170, ene. 2022, doi: 10.1038/s41598-021-03858-9.
- [38] A. Olsen et al., “DeepWeeds: A multiclass weed species image dataset for deep learning,” *Scientific Reports*, vol. 9, art. no. 2058, feb. 2019, doi: 10.1038/s41598-018-38343-3.

- [39] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III y K. Crawford, “Datasheets for datasets,” *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, dic. 2021, doi: 10.1145/3458723.
- [40] A. Kamilaris y F. X. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, abr. 2018, doi: 10.1016/j.compag.2018.02.016.
- [41] S. J. Pan y Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, oct. 2010, doi: 10.1109/TKDE.2009.191.
- [42] V. Sze, Y.-H. Chen, T.-J. Yang y J. S. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, dic. 2017, doi: 10.1109/JPROC.2017.2761740.
- [43] L. Chechliński, B. Siemiątkowska y M. Majewski, “A system for weeds and crops identification—Reaching over 10 FPS on Raspberry Pi with the usage of MobileNets, DenseNet and custom modifications,” *Sensors*, vol. 19, no. 17, art. no. 3787, ago. 2019, doi: 10.3390/s19173787.
- [44] A. Milioto, P. Lottes y C. Stachniss, “Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in CNNs,” en *Proc. IEEE Int. Conf. Robotics and Automation*, Brisbane, Australia, 2018, pp. 2229–2235, doi: 10.1109/I-CRA.2018.8460962.
- [45] P. Lottes, J. Behley, A. Milioto y C. Stachniss, “Fully convolutional networks with sequential information for robust crop and weed detection in precision farming,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2870–2877, oct. 2018, doi: 10.1109/LRA.2018.2846289.
- [46] Y. Li, Z. Guo, F. Shuang, M. Zhang y X. Li, “Key technologies of machine vision for weeding robots: A review and benchmark,” *Computers and Electronics in Agriculture*, vol. 196, art. no. 106880, mayo 2022, doi: 10.1016/j.compag.2022.106880.
- [47] The Linux Kernel Organization, “Video for Linux API, Version 2,” *Linux Kernel Documentation*. [En línea]. Disponible en: <https://docs.kernel.org/userspace-api/media/v4l/v4l2.html>. [Consultado: 22-jul-2026].

- [48] A. Rahman, Y. Lu y H. Wang, “Performance evaluation of deep learning object detectors for weed detection for cotton,” *Smart Agricultural Technology*, vol. 3, art. no. 100126, feb. 2023, doi: 10.1016/j.atech.2022.100126.
- [49] F. Dang, D. Chen, Y. Lu y Z. Li, “YOLOWeeds: A novel benchmark of YOLO object detectors for multi-class weed detection in cotton production systems,” *Computers and Electronics in Agriculture*, vol. 205, art. no. 107655, feb. 2023, doi: 10.1016/j.compag.2023.107655.
- [50] OpenBSD Project, “OpenSSH manual pages.” [En línea]. Disponible en: <https://www.openbsd.org/openssh/manual.html>. [Consultado: 22-jul-2026].
- [51] R. Fielding, M. Nottingham y J. Reschke, *HTTP Semantics*, Internet Engineering Task Force, RFC 9110, STD 97, jun. 2022, doi: 10.17487/RFC9110.
- [52] A. Upadhyay, S. G. C., Y. Zhang, C. Koparan y X. Sun, “Development and evaluation of a machine vision and deep learning-based smart sprayer system for site-specific weed management in row crops: An edge computing approach,” *Journal of Agriculture and Food Research*, vol. 18, art. no. 101331, dic. 2024, doi: 10.1016/j.jafr.2024.101331.
- [53] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji y T. Gebru, “Model cards for model reporting,” en *Proc. Conference on Fairness, Accountability, and Transparency*, Atlanta, GA, USA, 2019, pp. 220–229, doi: 10.1145/3287560.3287596.

Anexos

Anexo A. Interfaz web final del sistema

La Figura 35 presenta la versión final de la interfaz web implementada para controlar y supervisar el rover. La aplicación permite visualizar el video capturado por la cámara, observar las detecciones generadas por el modelo YOLO, consultar los conteos por clase, controlar el movimiento del rover, orientar la cámara y almacenar evidencias.



Figura 35: Interfaz web final utilizada para el control y la supervisión del rover. Fuente: Autor.

Anexo B. Código de la interfaz web

En este anexo se presenta el código empleado para implementar la aplicación web. El programa desarrollado con Flask administra la cámara, la transmisión de video, la inferencia del modelo, las solicitudes enviadas desde el navegador y el almacenamiento de evidencias.

Servidor Flask y procesamiento de video

```
1  #!/usr/bin/env python3
2  """
3  Aplicacin web para el control y la supervisin del rover.
4
5  La aplicacin integra:
6  - Cmara USB.
7  - Inferencia YOLO11n.
8  - Transmisin MJPEG.
9  - Control de movimiento.
10 - Control de cámara PTZ.
11 - Conteo de pimientos y malezas.
12 - Almacenamiento de evidencias.
13 """
14
15 import atexit
16 import logging
17 import os
18 import threading
19 import time
20 from datetime import datetime
21 from pathlib import Path
22
23 import cv2
24 from flask import Flask, Response, jsonify, render_template
25 from ultralytics import YOLO
26
27 from control_movimiento import ControlRover
28
29
30 # =====
31 # CONFIGURACIN GENERAL
32 # =====
33
34 CAMERA_INDEX = 0
35 CAMERA_WIDTH = 640
36 CAMERA_HEIGHT = 480
37 CAMERA_FPS = 30
38
39 MODEL_PATH = "best.pt"
```

```

40 INFERENCE_SIZE = 640
41 IOU_THRESHOLD = 0.45
42
43 CONFIDENCE_THRESHOLDS = {
44     "pimiento": 0.50,
45     "maleza": 0.40,
46 }
47
48 EVIDENCE_DIRECTORY = Path("evidencias")
49
50 app = Flask(__name__)
51
52 logging.basicConfig(
53     level=logging.INFO,
54     format="%(asctime)s - %(levelname)s - %(message)s"
55 )
56
57
58 # =====
59 # INICIALIZACION
60 # =====
61
62 EVIDENCE_DIRECTORY.mkdir(parents=True, exist_ok=True)
63
64 model = YOLO(MODEL_PATH)
65
66 camera = cv2.VideoCapture(CAMERA_INDEX)
67 camera.set(cv2.CAP_PROP_FRAME_WIDTH, CAMERA_WIDTH)
68 camera.set(cv2.CAP_PROP_FRAME_HEIGHT, CAMERA_HEIGHT)
69 camera.set(cv2.CAP_PROP_FPS, CAMERA_FPS)
70
71 if not camera.isOpened():
72     raise RuntimeError("No se pudo abrir la cámara USB.")
73
74 try:
75     rover = ControlRover()
76 except Exception as error:
77     rover = None
78     logging.error(
79         "El servidor se inició sin control físico del rover: %s",
80         error
81     )
82
83
84 # =====
85 # VARIABLES COMPARTIDAS
86 # =====
87
88 state_lock = threading.Lock()

```

```

89 detection_enabled = True
90 last_processed_frame = None
91
92
93 current_counts = {
94     "Pimiento": 0,
95     "Maleza": 0,
96 }
97
98
99 # =====
100 # PROCESAMIENTO DE IMGENES
101 # =====
102
103 def process_frame(frame):
104     """
105     Ejecuta la inferencia y dibuja las detecciones aceptadas.
106     """
107     global current_counts
108
109     counts = {
110         "Pimiento": 0,
111         "Maleza": 0,
112     }
113
114     results = model.predict(
115         source=frame,
116         imgsz=INFERENCE_SIZE,
117         conf=0.25,
118         iou=IOU_THRESHOLD,
119         device="cpu",
120         verbose=False,
121     )
122
123     result = results[0]
124
125     for box in result.bboxes:
126         class_id = int(box.cls[0].item())
127         confidence = float(box.conf[0].item())
128         class_name = str(model.names[class_id])
129
130         normalized_name = class_name.strip().lower()
131
132         class_threshold = CONFIDENCE_THRESHOLDS.get(
133             normalized_name,
134             0.50
135         )
136
137         if confidence < class_threshold:

```



```

138         continue
139
140     x1, y1, x2, y2 = map(
141         int,
142         box.xyxy[0].tolist()
143     )
144
145     if normalized_name == "pimiento":
146         display_name = "Pimiento"
147         color = (0, 180, 0)
148
149     elif normalized_name == "maleza":
150         display_name = "Maleza"
151         color = (0, 120, 255)
152
153     else:
154         continue
155
156     counts[display_name] += 1
157
158     cv2.rectangle(
159         frame,
160         (x1, y1),
161         (x2, y2),
162         color,
163         2
164     )
165
166     label = f"{display_name} {confidence:.2f}"
167
168     text_size, _ = cv2.getTextSize(
169         label,
170         cv2.FONT_HERSHEY_SIMPLEX,
171         0.55,
172         2
173     )
174
175     text_width, text_height = text_size
176
177     label_y1 = max(0, y1 - text_height - 10)
178
179     cv2.rectangle(
180         frame,
181         (x1, label_y1),
182         (x1 + text_width + 8, y1),
183         color,
184         -1
185     )
186

```

```

187         cv2.putText(
188             frame,
189             label,
190             (x1 + 4, y1 - 5),
191             cv2.FONT_HERSHEY_SIMPLEX,
192             0.55,
193             (255, 255, 255),
194             2,
195             cv2.LINE_AA
196         )
197
198     current_counts = counts
199
200     summary = (
201         f"Pimientos: {counts['Pimiento']} | "
202         f"Malezas: {counts['Maleza']}"
203     )
204
205     cv2.rectangle(
206         frame,
207         (10, 10),
208         (350, 45),
209         (0, 0, 0),
210         -1
211     )
212
213     cv2.putText(
214         frame,
215         summary,
216         (20, 35),
217         cv2.FONT_HERSHEY_SIMPLEX,
218         0.65,
219         (255, 255, 255),
220         2,
221         cv2.LINE_AA
222     )
223
224     return frame
225
226
227 def generate_frames():
228     """
229     Genera el flujo MJPEG enviado al navegador.
230     """
231     global last_processed_frame
232
233     while True:
234         success, frame = camera.read()
235

```

```

236     if not success:
237         logging.warning("No se pudo adquirir un fotograma.")
238         time.sleep(0.05)
239         continue
240
241     frame = cv2.resize(
242         frame,
243         (CAMERA_WIDTH, CAMERA_HEIGHT)
244     )
245
246     with state_lock:
247         active = detection_enabled
248
249     if active:
250         try:
251             processed_frame = process_frame(frame.copy())
252
253             except Exception as error:
254                 logging.exception(
255                     "Error durante la inferencia: %s",
256                     error
257                 )
258             processed_frame = frame
259
260     else:
261         processed_frame = frame
262
263         cv2.putText(
264             processed_frame,
265             "Deteccion desactivada",
266             (20, 35),
267             cv2.FONT_HERSHEY_SIMPLEX,
268             0.7,
269             (0, 0, 255),
270             2,
271             cv2.LINE_AA
272         )
273
274     with state_lock:
275         last_processed_frame = processed_frame.copy()
276
277     encoded, buffer = cv2.imencode(
278         ".jpg",
279         processed_frame,
280         [cv2.IMWRITE_JPEG_QUALITY, 80]
281     )
282
283     if not encoded:
284         continue

```

```

285
286     yield (
287         b"--frame\r\n"
288         b"Content-Type: image/jpeg\r\n\r\n"
289         + buffer.tobytes()
290         + b"\r\n"
291     )
292
293
294 # =====
295 # RUTAS DE LA INTERFAZ
296 # =====
297
298 @app.route("/")
299 def index():
300     """Muestra la pagina principal."""
301     return render_template("index.html")
302
303
304 @app.route("/video_feed")
305 def video_feed():
306     """Transmite el flujo de video procesado."""
307     return Response(
308         generate_frames(),
309         mimetype="multipart/x-mixed-replace; boundary=frame"
310     )
311
312
313 @app.route("/control/<action>", methods=["POST"])
314 def control(action):
315     """Recibe una orden de movimiento."""
316     if rover is None:
317         return jsonify({
318             "ok": False,
319             "message": "Control del rover no disponible."
320         }), 503
321
322     try:
323         movement = rover.mover(action)
324
325         return jsonify({
326             "ok": True,
327             "movement": movement
328         })
329
330     except ValueError as error:
331         return jsonify({
332             "ok": False,
333             "message": str(error)

```

```

334         }), 400
335
336     except Exception as error:
337         logging.exception("Error de movimiento: %s", error)
338
339         return jsonify({
340             "ok": False,
341             "message": "No se pudo ejecutar el movimiento."
342         }), 500
343
344
345 @app.route("/ptz/<action>", methods=["POST"])
346 def ptz(action):
347     """Modifica la orientacin de la cmara."""
348     if rover is None:
349         return jsonify({
350             "ok": False,
351             "message": "Control PTZ no disponible."
352         }), 503
353
354     try:
355         angles = rover.mover_camara(action)
356
357         return jsonify({
358             "ok": True,
359             "angles": angles
360         })
361
362     except ValueError as error:
363         return jsonify({
364             "ok": False,
365             "message": str(error)
366         }), 400
367
368     except Exception as error:
369         logging.exception("Error PTZ: %s", error)
370
371         return jsonify({
372             "ok": False,
373             "message": "No se pudo orientar la cmara."
374         }), 500
375
376
377 @app.route("/toggle_detection", methods=["POST"])
378 def toggle_detection():
379     """Activa o desactiva la inferencia."""
380     global detection_enabled
381
382     with state_lock:

```

```

383     detection_enabled = not detection_enabled
384     active = detection_enabled
385
386     return jsonify({
387         "ok": True,
388         "detection_enabled": active
389     })
390
391
392 @app.route("/save_evidence", methods=["POST"])
393 def save_evidence():
394     """Guarda el ltimo fotograma procesado."""
395     with state_lock:
396         if last_processed_frame is None:
397             return jsonify({
398                 "ok": False,
399                 "message": "Todava no existe un fotograma disponible."
400             }), 409
401
402         evidence = last_processed_frame.copy()
403
404         timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
405
406         filename = (
407             f"evidencia_{timestamp}_"
408             f"p{current_counts['Pimiento']}_"
409             f"m{current_counts['Maleza']}.jpg"
410         )
411
412         destination = EVIDENCE_DIRECTORY / filename
413
414         saved = cv2.imwrite(
415             str(destination),
416             evidence
417         )
418
419         if not saved:
420             return jsonify({
421                 "ok": False,
422                 "message": "No se pudo guardar la evidencia."
423             }), 500
424
425         return jsonify({
426             "ok": True,
427             "filename": filename
428         })
429
430
431 @app.route("/status")

```

```

432 def status():
433     """Devuelve el estado actual del sistema."""
434     rover_state = {
435         "movimiento": "No disponible",
436         "angulo_horizontal": 0,
437         "angulo_vertical": 0,
438     }
439
440     if rover is not None:
441         rover_state = rover.obtener_estado()
442
443     with state_lock:
444         active = detection_enabled
445         counts = current_counts.copy()
446
447     return jsonify({
448         "detection_enabled": active,
449         "counts": counts,
450         "rover": rover_state,
451     })
452
453
454 # =====
455 # CIERRE DEL SISTEMA
456 # =====
457
458 def cleanup():
459     """Libera la cámara y detiene el rover."""
460     try:
461         camera.release()
462     except Exception:
463         pass
464
465     if rover is not None:
466         rover.cerrar()
467
468
469 atexit.register(cleanup)
470
471
472 if __name__ == "__main__":
473     app.run(
474         host="0.0.0.0",
475         port=5000,
476         debug=False,
477         threaded=True
478     )

```

Listing 1: Código del servidor Flask y del procesamiento de video.

Página HTML de la interfaz

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <meta charset="UTF-8">
5
6     <meta
7         name="viewport"
8         content="width=device-width, initial-scale=1.0"
9     >
10
11 <title>Control del rover agrcola</title>
12
13 <style>
14     * {
15         box-sizing: border-box;
16     }
17
18     body {
19         margin: 0;
20         padding: 20px;
21         min-height: 100vh;
22         font-family: Arial, Helvetica, sans-serif;
23         background: #eef2f4;
24         color: #1f2933;
25     }
26
27     .container {
28         width: min(1200px, 100%);
29         margin: auto;
30     }
31
32     h1 {
33         margin: 0 0 20px;
34         text-align: center;
35         color: #173f35;
36     }
37
38     .layout {
39         display: grid;
40         grid-template-columns: minmax(0, 2fr) minmax(300px, 1fr);
41         gap: 20px;
42     }
43
44     .card {
45         padding: 18px;
46         border-radius: 12px;
47         background: white;
```



```

48         box-shadow: 0 3px 12px rgba(0, 0, 0, 0.12);
49     }
50
51     .video-container {
52         width: 100%;
53         overflow: hidden;
54         border-radius: 10px;
55         background: black;
56     }
57
58     .video-container img {
59         display: block;
60         width: 100%;
61         height: auto;
62     }
63
64     .section-title {
65         margin: 0 0 14px;
66         text-align: center;
67         font-size: 18px;
68     }
69
70     .controls-grid {
71         display: grid;
72         grid-template-columns: repeat(3, 1fr);
73         gap: 8px;
74         margin-bottom: 20px;
75     }
76
77     button {
78         min-height: 48px;
79         border: none;
80         border-radius: 8px;
81         padding: 10px;
82         font-size: 15px;
83         font-weight: bold;
84         cursor: pointer;
85         background: #dfe7e5;
86         color: #173f35;
87     }
88
89     button:hover {
90         filter: brightness(0.95);
91     }
92
93     button:active {
94         transform: scale(0.98);
95     }
96

```

```

97     .movement {
98         background: #2d7d64;
99         color: white;
100     }
101
102     .stop {
103         background: #b83232;
104         color: white;
105     }
106
107     .ptz {
108         background: #376a9a;
109         color: white;
110     }
111
112     .action {
113         width: 100%;
114         margin-top: 8px;
115         background: #d99020;
116         color: white;
117     }
118
119     .save {
120         background: #654aa5;
121     }
122
123     .status-box {
124         margin-top: 15px;
125         padding: 12px;
126         border-radius: 8px;
127         background: #f4f7f6;
128         line-height: 1.7;
129     }
130
131     .message {
132         min-height: 24px;
133         margin-top: 12px;
134         text-align: center;
135         font-weight: bold;
136     }
137
138     .empty {
139         visibility: hidden;
140     }
141
142     @media (max-width: 850px) {
143         .layout {
144             grid-template-columns: 1fr;
145         }

```

```

146     }
147     </style>
148 </head>
149
150 <body>
151     <main class="container">
152         <h1>Rover agrcola con visin artificial</h1>
153
154         <div class="layout">
155             <section class="card">
156                 <h2 class="section-title">
157                     Video y detecciones
158                 </h2>
159
160                 <div class="video-container">
161                     
165                 </div>
166
167                 <div class="status-box">
168                     <div>
169                         <strong>Pimientos:</strong>
170                         <span id="pimiento-count">0</span>
171                     </div>
172
173                     <div>
174                         <strong>Malezas:</strong>
175                         <span id="maleza-count">0</span>
176                     </div>
177
178                     <div>
179                         <strong>Deteccin:</strong>
180                         <span id="detection-status">Activa</span>
181                     </div>
182
183                     <div>
184                         <strong>Movimiento:</strong>
185                         <span id="movement-status">Detenido</span>
186                     </div>
187                 </div>
188             </section>
189
190             <aside class="card">
191                 <h2 class="section-title">
192                     Control de movimiento
193                 </h2>
194

```

```

195 <div class="controls-grid">
196   <span class="empty">.</span>
197
198   <button
199     class="movement"
200     onpointerdown="sendMovement('forward')"
201     onpointerup="sendMovement('stop')"
202     onpointerleave="sendMovement('stop')"
203   >
204     Avanzar
205   </button>
206
207   <span class="empty">.</span>
208
209   <button
210     class="movement"
211     onpointerdown="sendMovement('left')"
212     onpointerup="sendMovement('stop')"
213     onpointerleave="sendMovement('stop')"
214   >
215     Izquierda
216   </button>
217
218   <button
219     class="stop"
220     onclick="sendMovement('stop')"
221   >
222     Detener
223   </button>
224
225   <button
226     class="movement"
227     onpointerdown="sendMovement('right')"
228     onpointerup="sendMovement('stop')"
229     onpointerleave="sendMovement('stop')"
230   >
231     Derecha
232   </button>
233
234   <span class="empty">.</span>
235
236   <button
237     class="movement"
238     onpointerdown="sendMovement('backward')"
239     onpointerup="sendMovement('stop')"
240     onpointerleave="sendMovement('stop')"
241   >
242     Retroceder
243   </button>

```

```

244         <span class="empty">.</span>
245     </div>
246
247     <h2 class="section-title">
248         Orientacin de la cmara
249     </h2>
250
251     <div class="controls-grid">
252         <span class="empty">.</span>
253
254         <button
255             class="ptz"
256             onclick="sendPtz('up')"
257         >
258             Arriba
259         </button>
260
261         <span class="empty">.</span>
262
263         <button
264             class="ptz"
265             onclick="sendPtz('left')"
266         >
267             Izquierda
268         </button>
269
270         <button
271             class="ptz"
272             onclick="sendPtz('center')"
273         >
274             Centrar
275         </button>
276
277         <button
278             class="ptz"
279             onclick="sendPtz('right')"
280         >
281             Derecha
282         </button>
283
284         <span class="empty">.</span>
285
286         <button
287             class="ptz"
288             onclick="sendPtz('down')"
289         >
290             Abajo
291         </button>
292

```

```

293         <span class="empty">.</span>
294     </div>
295
296     <button
297         class="action"
298         onclick="toggleDetection()"
299     >
300         Activar o desactivar detección
301     </button>
302
303     <button
304         class="action save"
305         onclick="saveEvidence()"
306     >
307         Guardar evidencia
308     </button>
309
310     <div
311         id="message"
312         class="message"
313     ></div>
314 </aside>
315 </div>
316 </main>
317
318 <script>
319     async function request(url, options = {}) {
320         const response = await fetch(url, options);
321         const data = await response.json();
322
323         if (!response.ok) {
324             throw new Error(
325                 data.message || "Error en la solicitud"
326             );
327         }
328
329         return data;
330     }
331
332     function showMessage(text, isError = false) {
333         const element = document.getElementById("message");
334
335         element.textContent = text;
336         element.style.color = isError ? "#b83232" : "#26734d";
337     }
338
339     async function sendMovement(action) {
340         try {
341

```

```

342         const data = await request(
343             '/control/${action}',
344             { method: "POST" }
345         );
346
347         document.getElementById(
348             "movement-status"
349         ).textContent = data.movement;
350
351     } catch (error) {
352         showMessage(error.message, true);
353     }
354 }
355
356 async function sendPtz(action) {
357     try {
358         await request(
359             '/ptz/${action}',
360             { method: "POST" }
361         );
362
363         showMessage("Posicin de cmara actualizada.");
364
365     } catch (error) {
366         showMessage(error.message, true);
367     }
368 }
369
370 async function toggleDetection() {
371     try {
372         const data = await request(
373             "/toggle_detection",
374             { method: "POST" }
375         );
376
377         const status = data.detection_enabled
378             ? "Activa"
379             : "Inactiva";
380
381         document.getElementById(
382             "detection-status"
383         ).textContent = status;
384
385         showMessage(`Deteccin: ${status}.`);
386
387     } catch (error) {
388         showMessage(error.message, true);
389     }
390 }

```

```

391
392     async function saveEvidence() {
393         try {
394             const data = await request(
395                 "/save_evidence",
396                 { method: "POST" }
397             );
398
399             showMessage(
400                 'Evidencia guardada: ${data.filename}'
401             );
402
403         } catch (error) {
404             showMessage(error.message, true);
405         }
406     }
407
408     async function updateStatus() {
409         try {
410             const data = await request("/status");
411
412             document.getElementById(
413                 "pimiento-count"
414             ).textContent = data.counts.Pimiento;
415
416             document.getElementById(
417                 "maleza-count"
418             ).textContent = data.counts.Maleza;
419
420             document.getElementById(
421                 "detection-status"
422             ).textContent = data.detection_enabled
423                 ? "Activa"
424                 : "Inactiva";
425
426             document.getElementById(
427                 "movement-status"
428             ).textContent = data.rover.movimiento;
429
430         } catch (error) {
431             console.error(error);
432         }
433     }
434
435     updateStatus();
436     setInterval(updateStatus, 1000);
437 </script>
438 </body>
439 </html>

```

Listing 2: Código HTML, CSS y JavaScript de la interfaz web.

Anexo C. Código de control del rover

El código presentado contiene las funciones utilizadas para controlar el desplazamiento manual del rover y la orientación de la cámara PTZ. Se emplearon una velocidad lineal de 0,15 y una velocidad angular de 0,30. Las órdenes son recibidas desde el servidor Flask y ejecutadas mediante la biblioteca del Transbot-SE.

```
1  #!/usr/bin/env python3
2  """
3  Control manual del rover Transbot-SE.
4
5  Funciones:
6  - Avanzar.
7  - Retroceder.
8  - Girar a izquierda y derecha.
9  - Detener el rover.
10 - Orientar la cmara mediante el mecanismo PTZ.
11 """
12
13 import logging
14 import threading
15 import time
16
17 from Transbot_Lib import Transbot
18
19
20 class ControlRover:
21     """Administra el movimiento del rover y de la cmara PTZ."""
22
23     VELOCIDAD_LINEAL = 0.15
24     VELOCIDAD_ANGULAR = 0.30
25
26     PASO_SERVO = 10
27     ANGULO_MINIMO = 0
28     ANGULO_MAXIMO = 180
29     ANGULO_CENTRAL = 90
30
31     SERVO_HORIZONTAL = 1
32     SERVO_VERTICAL = 2
33
34     def __init__(self) -> None:
35         """Inicializa la comunicacin con el Transbot-SE."""
36         self._lock = threading.Lock()
37         self.movimiento_actual = "Detenido"
38
39         self.angulo_horizontal = self.ANGULO_CENTRAL
40         self.angulo_vertical = self.ANGULO_CENTRAL
41
```

```

42     try:
43         self.bot = Transbot(debug=False)
44         self.bot.create_receive_threading()
45         time.sleep(1)
46
47         self.centrar_camara()
48         self.detener()
49
50         logging.info("Transbot-SE inicializado correctamente.")
51
52     except Exception as error:
53         logging.exception(
54             "No se pudo inicializar el Transbot-SE: %s",
55             error
56         )
57         raise RuntimeError(
58             "Error durante la inicializacin del rover."
59         ) from error
60
61     def mover(self, accion: str) -> str:
62         """
63         Ejecuta una accin de desplazamiento.
64
65         Parmetros
66         -----
67         accion:
68             forward, backward, left, right o stop.
69         """
70         accion = accion.lower().strip()
71
72         acciones_validas = {
73             "forward",
74             "backward",
75             "left",
76             "right",
77             "stop",
78         }
79
80         if accion not in acciones_validas:
81             raise ValueError(f"Accin de movimiento no vlida: {accion}")
82
83         with self._lock:
84             try:
85                 if accion == "forward":
86                     self.bot.set_car_motion(
87                         self.VELOCIDAD_LINEAL,
88                         0
89                     )
90                     self.movimiento_actual = "Avanzando"

```

```

91
92         elif accion == "backward":
93             self.bot.set_car_motion(
94                 -self.VELOCIDAD_LINEAL,
95                 0
96             )
97             self.movimiento_actual = "Retrocediendo"
98
99         elif accion == "left":
100             self.bot.set_car_motion(
101                 0,
102                 self.VELOCIDAD_ANGULAR
103             )
104             self.movimiento_actual = "Girando a la izquierda"
105
106         elif accion == "right":
107             self.bot.set_car_motion(
108                 0,
109                 -self.VELOCIDAD_ANGULAR
110             )
111             self.movimiento_actual = "Girando a la derecha"
112
113         else:
114             self.bot.set_car_motion(0, 0)
115             self.movimiento_actual = "Detenido"
116
117         logging.info(
118             "Movimiento ejecutado: %s",
119             self.movimiento_actual
120         )
121
122         return self.movimiento_actual
123
124     except Exception as error:
125         self.bot.set_car_motion(0, 0)
126         self.movimiento_actual = "Detenido"
127         logging.exception(
128             "Error durante el movimiento: %s",
129             error
130         )
131         raise
132
133     def detener(self) -> str:
134         """Detiene inmediatamente los motores."""
135         return self.mover("stop")
136
137     def mover_camara(self, accion: str) -> dict:
138         """
139         Modifica la orientacin de la cmara.

```

```

140
141 Parmetros
142 -----
143 accion:
144     up, down, left, right o center.
145 """
146 accion = accion.lower().strip()
147
148 acciones_validas = {
149     "up",
150     "down",
151     "left",
152     "right",
153     "center",
154 }
155
156 if accion not in acciones_validas:
157     raise ValueError(f"Accin PTZ no vlida:{accion}")
158
159 with self._lock:
160     if accion == "left":
161         self.angulo_horizontal = min(
162             self.ANGULO_MAXIMO,
163             self.angulo_horizontal + self.PASO_SERVO
164         )
165
166     elif accion == "right":
167         self.angulo_horizontal = max(
168             self.ANGULO_MINIMO,
169             self.angulo_horizontal - self.PASO_SERVO
170         )
171
172     elif accion == "up":
173         self.angulo_vertical = min(
174             self.ANGULO_MAXIMO,
175             self.angulo_vertical + self.PASO_SERVO
176         )
177
178     elif accion == "down":
179         self.angulo_vertical = max(
180             self.ANGULO_MINIMO,
181             self.angulo_vertical - self.PASO_SERVO
182         )
183
184     else:
185         self.angulo_horizontal = self.ANGULO_CENTRAL
186         self.angulo_vertical = self.ANGULO_CENTRAL
187
188 self.bot.set_pwm_servo(

```

```

189         self.SERVO_HORIZONTAL,
190         self.angulo_horizontal
191     )
192
193     self.bot.set_pwm_servo(
194         self.SERVO_VERTICAL,
195         self.angulo_vertical
196     )
197
198     return {
199         "horizontal": self.angulo_horizontal,
200         "vertical": self.angulo_vertical,
201     }
202
203 def centrar_camara(self) -> dict:
204     """Coloca los dos servomotores en la posicin central."""
205     return self.mover_camara("center")
206
207 def obtener_estado(self) -> dict:
208     """Devuelve el estado actual del sistema de control."""
209     return {
210         "movimiento": self.movimiento_actual,
211         "angulo_horizontal": self.angulo_horizontal,
212         "angulo_vertical": self.angulo_vertical,
213         "velocidad_lineal": self.VELOCIDAD_LINEAL,
214         "velocidad_angular": self.VELOCIDAD_ANGULAR,
215     }
216
217 def cerrar(self) -> None:
218     """Detiene el rover antes de finalizar el programa."""
219     try:
220         self.detener()
221         logging.info("Sistema de movimiento detenido.")
222     except Exception as error:
223         logging.error(
224             "Error al cerrar el control del rover: %s",
225             error
226         )
227
228
229 if __name__ == "__main__":
230     logging.basicConfig(level=logging.INFO)
231
232     rover = ControlRover()
233
234     try:
235         rover.mover("forward")
236         time.sleep(1)
237         rover.detener()

```

```
238         rover.mover_camara("left")
239         time.sleep(1)
240         rover.centrar_camara()
241
242     finally:
243         rover.cerrar()
244
```

Listing 3: Código utilizado para controlar el movimiento del rover y la orientación de la cámara PTZ.

Anexo D. Código de entrenamiento del modelo

En este anexo se presenta el código utilizado para entrenar y validar el modelo YOLO11n en Google Colab. El entrenamiento se efectuó durante 100 épocas, con imágenes de 640×640 píxeles, un tamaño de lote de 16 y una paciencia de 30 épocas. El archivo `best.pt` fue seleccionado para su implementación en la Raspberry Pi 5.

```
1  #!/usr/bin/env python3
2  """
3  Entrenamiento del modelo YOLO11n en Google Colab.
4  """
5
6  import shutil
7  import subprocess
8  import sys
9  from pathlib import Path
10
11
12  # =====
13  # INSTALACION DE ULTRALYTICS
14  # =====
15
16  subprocess.check_call([
17      sys.executable,
18      "-m",
19      "pip",
20      "install",
21      "--quiet",
22      "ultralytics",
23  ])
24
25
26  # La importacion se realiza despues de instalar la biblioteca.
27  from ultralytics import YOLO  # noqa: E402
28
29
30  # =====
31  # CONEXION CON GOOGLE DRIVE
32  # =====
33
34  try:
35      from google.colab import drive
36
37      drive.mount("/content/drive")
38
39  except ImportError:
40      print(
41          "El programa no se est ejecutando en Google Colab. "
```



```

42         "Se utilizarn las rutas disponibles en el sistema."
43     )
44
45
46 # =====
47 # RUTAS DEL PROYECTO
48 # =====
49
50 DATA_YAML = Path(
51     "/content/drive/MyDrive/tesis/dataset/data.yaml"
52 )
53
54 OUTPUT_DIRECTORY = Path(
55     "/content/drive/MyDrive/tesis/resultados_yolo"
56 )
57
58 FINAL_MODEL_DIRECTORY = Path(
59     "/content/drive/MyDrive/tesis/modelo_final"
60 )
61
62 FINAL_MODEL_DIRECTORY.mkdir(
63     parents=True,
64     exist_ok=True
65 )
66
67
68 if not DATA_YAML.exists():
69     raise FileNotFoundError(
70         f"No se encontr el archivo data.yaml en: {DATA_YAML}"
71     )
72
73
74 # =====
75 # CARGA DEL MODELO PREENTRENADO
76 # =====
77
78 model = YOLO("yolo11n.pt")
79
80
81 # =====
82 # ENTRENAMIENTO
83 # =====
84
85 model.train(
86     data=str(DATA_YAML),
87     epochs=100,
88     imgsz=640,
89     batch=16,
90     patience=30,

```

```

91     optimizer="auto",
92     device=0,
93     workers=2,
94     project=str(OUTPUT_DIRECTORY),
95     name="entrenamiento_rover",
96     exist_ok=True,
97     pretrained=True,
98     verbose=True,
99     plots=True,
100     seed=42,
101 )
102
103
104 # =====
105 # LOCALIZACIN DE LOS MEJORES PESOS
106 # =====
107
108 training_directory = Path(model.trainer.save_dir)
109
110 best_weights = (
111     training_directory
112     / "weights"
113     / "best.pt"
114 )
115
116 last_weights = (
117     training_directory
118     / "weights"
119     / "last.pt"
120 )
121
122
123 if not best_weights.exists():
124     raise FileNotFoundError(
125         f"No se encontr el archivo best.pt en: {best_weights}"
126     )
127
128
129 print(f"Mejores pesos encontrados en: {best_weights}")
130 print(f"ltimos pesos encontrados en: {last_weights}")
131
132
133 # =====
134 # VALIDACIN DEL MEJOR MODELO
135 # =====
136
137 best_model = YOLO(str(best_weights))
138
139 validation_metrics = best_model.val(

```

```

140     data=str(DATA_YAML),
141     imgsz=640,
142     batch=16,
143     conf=0.25,
144     iou=0.45,
145     device=0,
146     plots=True,
147     project=str(OUTPUT_DIRECTORY),
148     name="validacion_modelo_final",
149     exist_ok=True,
150 )
151
152
153 print("\nResultados generales de validacin")
154 print(
155     "Precisin media:",
156     validation_metrics.box.mp
157 )
158 print(
159     "Recall medio:",
160     validation_metrics.box.mr
161 )
162 print(
163     "mAP@50:",
164     validation_metrics.box.map50
165 )
166 print(
167     "mAP@50-95:",
168     validation_metrics.box.map
169 )
170
171
172 # =====
173 # COPIA DEL ARCHIVO FINAL
174 # =====
175
176 final_model_path = (
177     FINAL_MODEL_DIRECTORY
178     / "best.pt"
179 )
180
181 shutil.copy2(
182     best_weights,
183     final_model_path
184 )
185
186 print(
187     "\nModelo final copiado correctamente en:",
188     final_model_path

```

```

189 )
190
191
192 # =====
193 # PRUEBA OPCIONAL SOBRE EL CONJUNTO TEST
194 # =====
195
196 test_results = best_model.predict(
197     source=str(DATA_YAML.parent / "test" / "images"),
198     imgsz=640,
199     conf=0.25,
200     iou=0.45,
201     device=0,
202     save=True,
203     project=str(OUTPUT_DIRECTORY),
204     name="predicciones_test",
205     exist_ok=True,
206 )
207
208 print(
209     f"Se procesaron {len(test_results)} imagenes "
210     "del conjunto de prueba."
211 )

```

Listing 4: Código utilizado para el entrenamiento y la validación del modelo YOLO11n.

Anexo E. Evidencias adicionales de detección

La Figura ?? presenta evidencias complementarias obtenidas durante la operación del sistema en la parcela agrícola. Las capturas muestran la identificación de plantas de pimienta y malezas mediante cajas delimitadoras, etiquetas y niveles de confianza. Se incluyen escenas con diferentes condiciones de iluminación, distancia, perspectiva y superposición de vegetación.



Figura 36: Evidencias adicionales de detección obtenidas durante las pruebas realizadas en la parcela agrícola. Fuente: Autor.

Anexo F. Certificado de Plagio



trabajo_UICII_final_final(9)_compressed

ID : b7c652595be8f213a2850ea38d2844c027413cc4



Nombre del fichero : trabajo_UICII_final_final_(9)_compressed.txt
Tamaño del archivo original : 2,12 MB
Número de palabras : 33.006

Depositante : LUIS ENRIQUE CHUQUIMARCA JIMENEZ
Fecha de depósito : 3 de agosto de 2026
Tipo de carga : interface
fecha de fin de análisis : 3 de agosto de 2026

 Resumen (sección 1/2)

Localización de los textos sospechosos en el documento :



Incluido en el porcentaje de textos sospechosos :

 Similitudes

Pasajes con similitudes a fuentes encontradas en diferentes colecciones.



 Detección de IA

Este índice es un indicador y no una prueba. Comprueba con el autor si domina los conocimientos mencionados en el documento.



 Idiomas no reconocidos

Pasajes en los que parte del vocabulario utilizado no forma parte del diccionario de la lengua. Puede tratarse de un intento del autor de modificar el texto para evitar ser detectado.



No incluido en el porcentaje de textos sospechosos :

” Textos entre comillas

Pasajes entre comillas, a menudo indicativos de una cita.

