

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

SISTEMA DE CLASIFICACIÓN AUTOMÁTICO DE FRESAS SEGÚN SU CALIDAD EMPLEANDO INTELIGENCIA ARTIFICIAL

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Programación y redes

LÍNEA DE INVESTIGACIÓN: Tecnología de la Información y Sistemas de control

AUTOR: Kevin Andres Manobanda Jimenez

TUTOR: Ing. Ana Pamela Castro Martin, MSc.

Ambato - Ecuador

julio – 2025

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE CLASIFICACIÓN AUTOMÁTICO DE FRESAS SEGÚN SU CALIDAD EMPLEANDO INTELIGENCIA ARTIFICIAL, desarrollado bajo la modalidad Proyecto de Investigación por el señor Kevin Andres Manobanda Jimenez, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, julio 2025.

Firmado electrónicamente por:
ANA PAMELA CASTRO MARTIN
Validar únicamente con FirmaEC

Ing. Ana Pamela Castro Martin, MSc.
TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA DE CLASIFICACIÓN AUTOMÁTICO DE FRESAS SEGÚN SU CALIDAD EMPLEANDO INTELIGENCIA ARTIFICIAL es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, julio 2025.



Kevin Andres Manobanda Jimenez

C.C. 1850285006

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, julio 2025.



Kevin Andres Manobanda Jimenez

C.C. 1850285006

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Kevin Andres Manobanda Jimenez, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE CLASIFICACIÓN AUTOMÁTICO DE FRESAS SEGÚN SU CALIDAD EMPLEANDO INTELIGENCIA ARTIFICIAL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, julio 2025.

Firmado electrónicamente por:
FRANKLIN OSWALDO MAYORGA MAYORGA
Validar únicamente con FirmaEC

Ing. Franklin Oswaldo Mayorga Mayorga, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
MARLON ANTONIO SANTAMARIA VILLACIS
Validar únicamente con FirmaEC

Ing. Marlon Santamaría Villacís, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
JUAN CAMILO ESCOBAR NARANJO
Validar únicamente con FirmaEC

Ing. Camilo Escobar Naranjo, Mg.
PROFESOR CALIFICADOR

DEDICATORIA

El presente proyecto de investigación se lo dedico a Dios, quien ha sido mi guía y fortaleza en cada momento de este proceso. Su luz ha iluminado mi camino, dándome el coraje y la inspiración necesarios para superar cada desafío que he encontrado.

A mis queridos padres, Lourdes y Néstor, cuyo amor incondicional y apoyo constante han sido la base de todos mis logros. Su fe en mis capacidades y su aliento en los momentos más difíciles han sido fundamentales para alcanzar esta meta. A mi madre Lourdes, una mujer de inagotable amor y sabiduría, gracias por ser mi ejemplo y mi refugio en cada paso de mi vida. A mi padre Néstor, cuyo esfuerzo y dedicación me han enseñado el verdadero valor del trabajo duro y la perseverancia.

A mis amados hermanos, Alexander y Pamela, quienes han sido una fuente constante de motivación y alegría. Su apoyo y cariño han sido esenciales en este camino, y este logro es tanto mío como de ustedes. Su amor ha sido un pilar de fuerza que me ha impulsado a seguir adelante y a nunca rendirme.

AGRADECIMIENTO

Quiero expresar mis más sinceros agradecimientos a mis padres, por su amor y apoyo incondicional durante estos años de estudio, por sus sacrificios y palabras de aliento que me ha motivado a seguir adelante.

También extiendo mi gratitud a quienes me acompañaron durante toda la carrera, en especial a Cindy, Niky, Lenin, Yajira, Evelyn y a todos mis queridos amigos de la carrera, conocidos con cariño como los "Chonta panas", quienes han hecho de esta experiencia universitaria algo verdaderamente inolvidable. Un agradecimiento muy especial para Nayeli, quien estuvo a mi lado desde el día cero.

Quiero expresar también mi profunda gratitud a mi tutora de tesis, la Ing. Pamela Castro, por su invaluable orientación y apoyo a lo largo del desarrollo de este proyecto. Su dedicación y guía han sido fundamentales para la culminación de este trabajo.

Finalmente, agradezco a la Universidad Técnica de Ambato por proporcionarme los recursos académicos necesarios para llevar a cabo mis estudios, permitiéndome crecer tanto personal como profesionalmente.

ÍNDICE GENERAL DE CONTENIDOS

PORTADA ..... **i**

APROBACIÓN DEL TUTOR ..... **ii**

AUTORÍA ..... **iii**

DERECHOS DE AUTOR ..... **iv**

APROBACIÓN DEL TRIBUNAL DE GRADO ..... **v**

DEDICATORIA ..... **vi**

AGRADECIMIENTO ..... **vii**

ÍNDICE GENERAL DE CONTENIDOS ..... **viii**

ÍNDICE DE TABLAS ..... **xii**

ÍNDICE DE FIGURAS ..... **xiv**

ÍNDICE DE ANEXOS ..... **xviii**

RESUMEN EJECUTIVO ..... **xix**

ABSTRACT ..... **xx**

CAPÍTULO I. MARCO TEÓRICO **1**

1.1 Tema de investigación 1

1.1.1 Planteamiento del problema 1

1.2 Antecedentes investigativos 2

1.3 Fundamentación teórica 5

1.3.1 Agricultura de precisión 5

1.3.2 Sistemas de clasificación automática 7

1.3.3 Características generales de las fresas en Ecuador 9

1.3.4 Clasificación de fresas en grados de calidad 11

1.3.5 Inteligencia Artificial 17

1.3.6 Visión artificial 20

1.3.7 Procesamiento de imágenes 23

1.3.8 Adquisición y digitalización de imágenes 26

1.3.9 Análisis comparativo sobre las técnicas de visión artificial empleados en la clasificación de frutas 29

1.3.10 Redes Convolucionales 31

1.3.11 YOLO (You Only Look Once) 32

1.3.12 Metodología para Machine Learning: CRISP-DM 35

1.4 Objetivos 36

1.4.1 Objetivo general 36

1.4.2 Objetivos específicos 36

CAPÍTULO II. METODOLOGÍA 37

2.1 Materiales 37

2.2 Métodos 39

2.2.1 Modalidad de la investigación 39

2.2.2 Recolección de información..... 39

2.2.3 Procesamiento y análisis de datos 40

CAPÍTULO III. RESULTADOS Y DISCUSIÓN..... 41

3.1 Comprensión del negocio: Análisis de la producción de fresa en el Ecuador 42

3.1.1 Zonas y épocas donde la producción de la fresa es mayor 43

3.1.2 Producción de la fresa en el Ecuador 43

3.2 Comprensión de los datos: Análisis de las características de calidad de la fresa 44

3.3 Preparación de los datos: Etiquetado de las fresas según tipos..... 45

3.3.1 Selección de cámara para la adquisición de imágenes..... 46

3.3.2 Adquisición de imágenes para el Dataset..... 46

3.3.3 Selección de software de etiquetado de imágenes 47

3.3.4 Etiquetado utilizando la herramienta online Roboflow 48

3.4 Modelado: Entrenamiento del modelo de Inteligencia Artificial..... 50

3.4.1 Selección de algoritmo de Inteligencia Artificial..... 51

3.4.2 Selección de herramienta para ejecutar código en nube 52

3.4.3 Configuración del entrenamiento en Google Colab..... 52

3.4.4 Visualización de resultados del entrenamiento con Google Colab..... 56

3.5 Evaluación: Resultados del entrenamiento del modelo de inteligencia artificial 58

3.5.1 Entrenamiento 1 con 50 etapas 59

3.5.2 Entrenamiento 2 con 75 etapas 63

3.5.3 Entrenamiento 3 con 100 etapas 67

3.5.4 Comparación de modelos 71

3.6 Despliegue: Diseño del sistema clasificador de fresas según su calidad 72

3.6.1 Selección de lenguaje de programación 72

3.6.2 Algoritmo del sistema de clasificación 73

3.6.3 Selección de microcontrolador 81

3.6.4 Algoritmo de clasificación para el microcontrolador 81

3.6.5 Prototipo de la máquina clasificadora 85

3.6.6 Visualización de los resultados 90

3.7 Pruebas de funcionamiento del sistema 91

3.7.1 Resultados de las pruebas de funcionamiento 94

3.7.2 Análisis de los resultados del sistema 96

3.8 Presupuesto 98

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES 100

4.1 Conclusiones 100

4.2 Recomendaciones 102

REFERENCIAS BIBLIOGRÁFICAS 104

ANEXOS 110

ÍNDICE DE TABLAS

Tabla 1. Tecnologías utilizadas en agricultura de precisión 5

Tabla 2. Máquinas y sistemas utilizados en la agricultura de precisión 6

Tabla 3. Comparativa de actuadores en sistemas de clasificación agrícola 7

Tabla 4. Tipos de servo motores y sus características 8

Tabla 5. Tipos de microcontrolador 9

Tabla 6. Grados de calidad de la fresa según la norma NMX-FF-062-SCFI-2002 ... 12

Tabla 7, Especificaciones de tamaño 13

Tabla 8. Tolerancias de daños permitidos 13

Tabla 9. Tipos de aprendizaje automático 18

Tabla 10. Tipos de Redes Neuronales 19

Tabla 11. Herramientas para ejecutar código en nube 20

Tabla 12. Etapas de un proceso de visión artificial 21

Tabla 13. Componentes de la visión artificial 22

Tabla 14. Importancia de las microcomputadoras en la visión artificial 23

Tabla 15. Tipos de microcomputadoras y sus características 23

Tabla 16. Lenguajes de programación utilizados en inteligencia artificial 26

Tabla 17. Etapas fundamentales de la adquisición y el procesamiento de imágenes 26

Tabla 18. Tipos de Cámaras 27

Tabla 19. Tipos de software para etiquetado de imágenes 29

Tabla 20. Técnicas de visión artificial empleados en la clasificación de frutas 30

Tabla 21. Tipos de Redes Convolucionales 31

Tabla 22. Tabla comparativa de las versiones de Yolo..... 33

Tabla 23. Comparación de las versiones de Yolov11 35

Tabla 24. Hardware utilizado en el proyecto de investigación. 38

Tabla 25. Software utilizado en el proyecto de investigación. 38

Tabla 26. Criterios de recolección de información. 40

Tabla 27. Diferentes calidades de la fresa..... 45

Tabla 28. Característica de la máquina usada para entrenamiento 52

Tabla 29. Datos de la matriz de confusión del modelo entrenado 1 62

Tabla 30. Datos de la matriz de confusión del modelo entrenado 2 66

Tabla 31. Datos de la matriz de confusión del modelo entrenado 3 70

Tabla 32. Comparación de resultados obtenidos del entrenamiento. 72

Tabla 33. Pruebas de funcionamiento 95

Tabla 34. Eficiencia del sistema..... 96

Tabla 35. Costos de la elaboración del prototipo 99

ÍNDICE DE FIGURAS

Figura 1. Fresas y los fresones más comercializados..... 10

Figura 2. Grados de madurez de las fresas..... 14

Figura 3. Grados de madures de la fresa según el CENICAFE 15

Figura 4. Ramas de la inteligencia artificial..... 17

Figura 5. Diagrama básico de la visión artificial 21

Figura 6. Segmentación semántica..... 24

Figura 7. Oclusión en fruta. a) Imagen Original b) Etiquetas rectangulares c) Etiquetas de segmentos 25

Figura 8. Esquema general de del sistema de clasificación automática de fresas..... 41

Figura 9. Estructura de carpetas del dataset de fresas..... 47

Figura 10. Etiquetado de imágenes en Roboflow 48

Figura 11. Proceso para incrementar los datos del Dataset..... 49

Figura 12. Procedimiento para utilizar el Dataset creado en Roboflow 49

Figura 13. Contenido del archivo data.yaml 50

Figura 14. Líneas código para importar las librerias en Google Colab 53

Figura 15. Líneas de código para importar el Dataset de Roboflow..... 53

Figura 16. Líneas de código para verificar la exportación del Dataset..... 54

Figura 17. Configuración del entrenamiento con YOPLOv11l (50 épocas)..... 55

Figura 18. Configuración del entrenamiento con YOPLOv11l (75 y 100 épocas)... 55

Figura 19. Inicio de la ejecución del entrenamiento con 50 epocas..... 56

Figura 20. Finalización de la ejecución del entrenamiento con 50 épocas 56

Figura 21. Inicio de la ejecución del entrenamiento con 75 épocas..... 57

Figura 22. Finalización de la ejecución del entrenamiento con 75 épocas 57

Figura 23. Inicio de la ejecución del entrenamiento con 100 épocas..... 58

Figura 24. Finalización de la ejecución del entrenamiento con 100 épocas 58

Figura 25. Métricas de rendimiento del modelo entrenado 1..... 59

Figura 26. Matriz de confusión del modelo entrenado 1 61

Figura 27. Precisión del modelo 1..... 63

Figura 28. Métricas de rendimiento del modelo entrenado 2..... 64

Figura 29. Matriz de confusión del modelo entrenado 2 65

Figura 30. Precisión del modelo 2..... 67

Figura 31. Métricas de rendimiento del modelo entrenado 3..... 68

Figura 32. Matriz de confusión del modelo entrenado 3 69

Figura 33. Precisión del modelo 3..... 71

Figura 34. Diagrama de flujo del código de Python 74

Figura 35. Código de Python: Importación de librerias..... 75

Figura 36. Código de Python: Función __init__..... 76

Figura 37. Código de Python: Función para cargar el modelo YOLOv11l 77

Figura 38. Código de Python: Función para establecer conexión con serial 77

Figura 39. Código de Python: Función para iniciar la clasificación de fresas..... 77

Figura 40. Código Python: Función para marcar y enumerar los frames 78

Figura 41. Código Python: Función para agregar los resultados detectados a una tabla 78

Figura 42. Código Python: Función para enviar los datos al Arduino 79

Figura 43. Código Python: Función para detener la clasificación de fresas 79

Figura 44. Código Python: Función para agregar el video para analizar 79

Figura 45. Código Python: Función para reproducir el video que se va a analizar ... 80

Figura 46. Resultados del sistema de clasificación 81

Figura 47. Diagrama de flujo del condigo de Arduino 82

Figura 48. Código de Arduino: Definición de constantes, inclusión de bibliotecas y comunicación serial. 83

Figura 49. Código Arduino: Función loop(). 84

Figura 50. Código: Función processData(). 84

Figura 51. Código de Arduino: Función activateGroup() y activateFifthServo() 85

Figura 52. Modelado en 3D del prototipo 87

Figura 53. Colocación de los actuadores en el prototipo. 87

Figura 54. Prototipo del sistema de clasificación automática de fresas en base a su calidad. 88

Figura 55. Circuito de la máquina clasificadora automática de fresas. 89

Figura 56. Interfaz gráfica de usuario 91

Figura 57. Resultado 1 92

Figura 58. Resultado 2 92

Figura 59. Mensaje serial que se envía al microcontrolador del resultado 1 93

Figura 60. Mensaje serial que se envía al microcontrolador del resultado 2 93

Figura 61. Acciones que realiza el microcontrolador con los datos del resultado 1 .. 94

Figura 62. Acciones que realiza el microcontrolador con los datos del resultado 2 .. 94

ÍNDICE DE ANEXOS

Anexo A. Diseño del prototipo clasificador de fresas según su calidad. 110

Anexo B. Laptop OMEN by HP 15-dh1xxx 110

Anexo C. NVIDIA GeForce GTX 1660 Ti 111

Anexo D. Arduino Mega 2560 112

Anexo E. Algoritmo de clasificación en Python 113

Anexo F. Algoritmo de clasificación para el microcontrolador. 120

Anexo G. Pruebas del sistema 122

RESUMEN EJECUTIVO

El cultivo de fresas en Ecuador tiene una relevancia económica y social notable. Gracias a su calidad, valor nutricional y versatilidad culinaria, el país se ha posicionado como exportador hacia Estados Unidos, Europa y América Latina, mercados que exigen altos estándares de calidad y tecnologías avanzadas.

Este proyecto propone un sistema automático de clasificación de fresas mediante inteligencia artificial, estructurado en cuatro etapas. La primera etapa consistió en la adquisición de imágenes con la cámara trasera de un smartphone Samsung A10. Estas imágenes fueron etiquetadas usando Roboflow y organizadas en una base de datos. En la segunda etapa, se entrenó un modelo YOLOv11l en Google Colab y se utilizó Python y librerías como tkinter (interfaz gráfica), PIL (visualización de imágenes), OpenCV (procesamiento de video), ultralytics.YOLO (modelo IA), serial (comunicación con Arduino) y time para ejecutar el modelo de IA logrando clasificar las fresas en “Extra”, “Primera”, “Segunda” y “Desecho”. Para la ejecución eficiente del sistema, se empleó una laptop HP OMEN 15, aprovechando su tarjeta gráfica NVIDIA GeForce GTX 1660 Ti, que mejora significativamente la velocidad de procesamiento del modelo. La tercera etapa incluyó la construcción de una máquina clasificadora automatizada con Arduino Mega, actuadores y botones para la separación física de las fresas.

Por último, se desarrolló una interfaz amigable para visualizar resultados en tiempo real. El sistema logró una eficacia del 83,05% en la clasificación, reduciendo el error humano, optimizando procesos y asegurando un control de calidad uniforme. Esto mejora la competitividad de los productores ecuatorianos en los mercados locales e internacionales.

Palabras clave: Visión artificial, Yolov11l, calidad de la fresa, máquina clasificadora.

ABSTRACT

Strawberry cultivation in Ecuador has significant economic and social relevance. Thanks to their quality, nutritional value, and culinary versatility, the country has positioned itself as an exporter to the United States, Europe, and Latin America, markets that demand high quality standards and advanced technologies.

This project proposes an automatic strawberry classification system using artificial intelligence, structured in four stages. The first stage consisted of acquiring images with the rear camera of a Samsung A10 smartphone. These images were labeled using Roboflow and organized into a database. In the second stage, a YOLOv11l model was trained in Google Colab, and Python and libraries such as tkinter (graphical interface), PIL (image visualization), OpenCV (video processing), ultralytics. YOLO (AI model), serial (communication with Arduino), and time to run the AI model, classifying the strawberries into “Extra,” “First,” “Second,” and “Discard.” For efficient system execution, an HP OMEN 15 laptop was used, taking advantage of its NVIDIA GeForce GTX 1660 Ti graphics card, which significantly improves the model's processing speed. The third stage included the construction of an automated sorting machine with Arduino Mega, actuators, and buttons for the physical separation of strawberries.

Finally, a user-friendly interface was developed to display results in real time. The system achieved 83.05% sorting efficiency, reducing human error, optimizing processes, and ensuring uniform quality control. This improves the competitiveness of Ecuadorian producers in local and international markets.

Keywords: Computer vision, Yolov11l, strawberry quality, sorting machine.

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA DE CLASIFICACIÓN AUTOMÁTICO DE FRESAS SEGÚN SU CALIDAD EMPLEANDO INTELIGENCIA ARTIFICIAL

1.1.1 Planteamiento del problema

En la industria agrícola, especialmente en la producción y comercialización de fresas, la calidad del producto es fundamental. Esto exige el cumplimiento de altos estándares de calidad para que los productos sean competitivos en el mercado. Por lo tanto, el uso de tecnología que respalde los procesos de control de calidad es indispensable para los productores que desean escalar en el mercado. Según el Observatorio de Complejidad Económica (OEC), a nivel mundial en 2022 los principales exportadores de fresas, frescas fueron: México, España, Estados Unidos, Países Bajos y Bélgica [1].

Según la Asociación Ecuatoriana de Fruticultores la distribución de cultivos de fresas dentro del país se da, con Pichincha liderando la producción en 400 hectáreas, seguida de cerca por Tungurahua con 240 hectáreas. Otras provincias, como Chimborazo, Cotopaxi, Imbabura y Azuay, también muestran cifras considerables, superando las 40 hectáreas en producción [2].

Dentro de la provincia de Tungurahua, el cantón Tisaleo es un referente para la producción de moras y fresas, que son los cultivos insignia de la región ya que el 60% de la producción nacional de estas frutas proviene de la provincia siendo en este caso la mora la protagonista [3].

La falta de precisión en la selección de fresas de distinta calidad dentro de un mismo lote es una problemática recurrente en la industria agrícola ecuatoriana. Esta situación suele deberse a la deficiente capacitación del personal y a la carencia de herramientas adecuadas para evaluar la calidad, especialmente en la etapa de postcosecha. Un ejemplo representativo se observa en los cultivos de la comunidad de Paquiestancia, cantón Cayambe, donde se cosechan aproximadamente 1000 kg de fresas por semana,

lo que equivale al 100% de la producción. Durante el proceso de selección, los productores clasifican los frutos en "buenos" y "defectuosos", generando pérdidas que oscilan entre el 14% y el 15% [4]. Esta situación refleja una realidad similar en la mayoría de las provincias productoras de fresas del país, donde la falta de precisión en la clasificación impacta directamente en la eficiencia y rentabilidad del proceso productivo.

También la inconsistencia en los criterios de clasificación con productos de menor calidad también plantea desafíos significativos en la cadena de suministro. Cuando los criterios de clasificación no son claros o no se aplican de manera consistente, las fresas de calidad inferior pueden pasar desapercibidas durante el proceso de selección y empaquetado. Esto conduce directamente a la comercialización de productos de menor calidad, lo que puede dañar la reputación del productor y reducir la confianza del consumidor en la marca.

Además, la limitada velocidad del proceso manual de clasificación puede afectar significativamente la eficiencia operativa y la puntualidad en la entrega de productos. Cuando la clasificación se realiza manualmente y a un ritmo lento, se crea un cuello de botella en la cadena de producción, lo que provoca demoras en la entrega del producto final. Estos retrasos no solo perjudican la satisfacción del cliente al no recibir su pedido en el tiempo esperado, sino que también pueden generar costos adicionales para el productor, como multas por entregas tardías.

1.2 Antecedentes investigativos

Como resultado de la investigación bibliográfica realizada en los repositorios de diferentes universidades dentro y fuera del país, así como en bases de datos en línea de artículos científicos, se exponen los siguientes antecedentes investigativos que pueden aportar al tema de investigación:

En 2018, Sitti Wetenriajeng Sidehabi, Ansar Suyuti, Intan Sari Areni e Ingrid Nurtanio llevaron a cabo la investigación sobre "El desarrollo de un sistema de visión artificial para clasificar maracuyá utilizando Multi-Class Support Vector Machine" publicado en la Journal of Engineering Science and Technology Review 11 (5) (2018) 178 – 184.

En esta investigación se utilizaron K-Means Clustering con Neural Network feed forward para clasificar el nivel de madurez del maracuyá en estado maduro, casi maduro y no maduro adoptando características del color RGBy a* con un resultado de 90% de precisión. Sin embargo, su desarrollo y validación se realizaron únicamente en laboratorio, lo que limita su aplicabilidad en campo. Condiciones variables como la iluminación, impurezas o cambios en el ángulo de visión podrían afectar su rendimiento. Además, el uso exclusivo de atributos cromáticos, sin integrar información sobre textura, forma o propiedades espectrales, restringe su capacidad para adaptarse a la variabilidad física del fruto [5].

En 2019, Pablo Esteban Villota Neira llevo a cabo la investigación sobre la "Implementación de una estación prototipo con visión artificial, aplicado a la agricultura de precisión" en la Universidad de Cuenca. Empleo la técnica de procesamiento de imágenes llamada Thresholding, un algoritmo sencillo y de bajo costo computacional que segmenta imágenes mediante umbrales. Aunque los resultados fueron satisfactorios, el algoritmo fue comparado con GMM con EM, el cual ofreció una segmentación más precisa, aunque con mayor tiempo de procesamiento. Thresholding resultó más rápido y eficiente energéticamente, lo que lo hace atractivo para entornos con recursos limitados. Sin embargo, la investigación presentó limitaciones como la falta de información sobre el tamaño y diversidad del conjunto de datos, lo que restringe la generalización del modelo [6].

En 2022, el Ing. Dany Orlando Orbes Padilla llevo a cabo la investigación sobre la "Implementación de un Sistema de detección por visión artificial en la etapa de recolección del cultivo de fresas" en la Universidad Técnica del Norte. En esta investigación para el análisis y evaluación del rendimiento del modelo de aprendizaje aplicado en el algoritmo de visión artificial, se utilizó la matriz de confusión que es una representación de verdaderos positivo, falsos positivos, negativos verdaderos falsos negativos, de la cual se obtuvo que este sistema tiene un 82,99% de exactitud, 80,33% de precisión, 86,6% de recuperación y 79% de especificidad. La investigación presentó varias limitaciones que afectan la escalabilidad y aplicabilidad del sistema propuesto en entornos reales. Entre ellas se destaca el proceso de entrenamiento requirió un tiempo prolongado, cercano a las dos semanas, lo que implica un alto costo computacional. El sistema también mostró una fuerte dependencia de condiciones

ambientales controladas, ya que solo funcionaba adecuadamente con iluminación natural y en climas secos; la lluvia o la oscuridad impedían su operatividad [7].

En 2024, Lenin Andrés Gutiérrez Sánchez llevo a cabo la de investigación sobre un "Sistema inteligente de selección de cosecha en cultivo de pitahaya mediante visión artificial" en la Universidad Técnica de Ambato. En su investigación implemento un sistema, basado en YOLOv5, este realiza una evaluación de la maduración de las pitahayas en tres etapas distintas. El modelo de entrenamiento ha logrado clasificar con éxito estas etapas, alcanzando una precisión del 99%. Para diferenciar entre los grados de maduración, se examinan los porcentajes de color verde y amarillo presentes en el conjunto de imágenes procesadas. Este enfoque proporciona una evaluación detallada y precisa de la calidad de la cosecha, generando resultados cuantitativos sobre la distribución de los diferentes grados de maduración. Sin embargo, la investigación presenta limitaciones como que la evaluación del modelo se llevó a cabo en un entorno controlado, lo que limita la extrapolación directa de sus resultados ante condiciones ambientales variables y dinámicas, como la calidad de la iluminación natural, la presencia de sombras o reflejos. Por otro lado, la caracterización de la madurez del fruto se basó exclusivamente en el análisis de los componentes cromáticos verde y amarillo, sin incorporar variables adicionales relacionadas con la textura superficial, forma o características espectrales más complejas [8].

En 2024, Juan Andres Chisaguano Tapia llevo a cabo la investigación sobre un "Sistema de visión artificial para el control de thrips en rosas de exportación." en la Universidad Técnica de Ambato. En este proyecto se realizó una instalación en el dispositivo móvil de una aplicación llamada "IP WEBCAM PRO" la cual por DHCP nos refleja una IP aleatoria del servidor la cual se conecta por el puerto 8080. Mediante el protocolo HTTP, la cual realiza peticiones, en este caso realiza capturas y adicional nos muestra un video en tiempo real de las rosas de exportación que se va a procesar garantizando una calidad de imagen del 100% en términos de resolución. Pese a sus aportes, el estudio presenta limitaciones que restringen su aplicabilidad en condiciones agronómicas diversas. La dependencia de direcciones IP asignadas dinámicamente mediante DHCP puede afectar la estabilidad y continuidad del monitoreo en tiempo real [9].

1.3 Fundamentación teórica

1.3.1 Agricultura de precisión

La Agricultura de Precisión (AP) es un enfoque que utiliza un conjunto de tecnologías, incluyendo las Tecnologías de la Información y la Comunicación (TIC), para optimizar la toma de decisiones y la gestión de cultivos. Surgió en los años 90 y ha evolucionado constantemente con la incorporación de nuevas tecnologías. La llegada de los Sistemas de Navegación Global por Satélite (GNSS) marcó un avance significativo, proporcionando datos geoespaciales que permiten un manejo más eficiente de los cultivos. El objetivo principal de la AP es maximizar el rendimiento y el uso eficiente de insumos agrícolas, mediante la evaluación de la variabilidad espacial de los terrenos y la precisión temporal, para mejorar la producción agrícola de forma sostenible [10].

a. Tecnologías utilizadas en la agricultura de precisión

La agricultura de precisión es un enfoque innovador que utiliza tecnologías avanzadas para optimizar la producción agrícola. Algunas de estas tecnologías se presentan en la Tabla 1.

Tabla 1. Tecnologías utilizadas en agricultura de precisión [11] [12]

Tecnologías	**Funcionalidad**
Sensores	Sensores de humedad del suelo, temperatura y nutrientes que permiten monitorear las condiciones del cultivo.
Drones	Utilizados para la vigilancia aérea, mapeo y análisis de cultivos.
GPS y Teledetección	Permite la localización precisa de los equipos agrícolas y teledetección para el análisis de la cobertura del suelo y las condiciones de los cultivos.
Software de Gestión Agrícola	Plataformas que integran datos de diferentes fuentes para facilitar la toma de decisiones. Ayudan en la planificación y seguimiento de actividades agrícolas.
Robótica y Automatización	Uso de maquinaria automatizada para la siembra, cosecha y cuidado de los cultivos. Mejora la eficiencia y reduce la necesidad de mano de obra.
Big Data y Análisis Predictivo	Análisis de grandes volúmenes de datos para prever tendencias y optimizar la producción.

b. Tipos de máquinas utilizadas en la agricultura de precisión

La agricultura de precisión representa una revolución tecnológica en el sector agrícola, incorporando una amplia gama de máquinas y sistemas automatizados que optimizan cada aspecto del proceso productivo. En la Tabla 2, se presenta una clasificación detallada de las principales tecnologías empleadas en la agricultura moderna. Cada categoría incluye las características específicas y aplicaciones prácticas de las diferentes máquinas, permitiendo comprender cómo estas innovaciones tecnológicas contribuyen a una agricultura más eficiente, sostenible y productiva [13].

Tabla 2. Máquinas y sistemas utilizados en la agricultura de precisión [13] [14] [15]

Categoría	**Tecnología**	**Características Principales**	**Aplicaciones**
Sistemas de Navegación y Guiado	GPS RTK	• Precisión centimetrada • Guiado automático • Optimización de rutas	• Control de trayectorias • Reducción de solapamientos • Navegación precisa
Sistemas de Monitoreo	Drones Agrícolas	• Mapeo multiespectral • Cámaras de alta resolución • Vuelo automatizado	• Detección de enfermedades • Evaluación de cultivos • Monitoreo de crecimiento
	Sensores de Campo	• Medición de humedad • Sensores de nutrientes • Estaciones meteorológicas	• Control de riego • Monitoreo de suelo • Variables climáticas
Máquinas de Aplicación	Sembradoras Inteligentes	• Dosificación precisa • Ajuste automático • Control de densidad	• Siembra variable • Optimización de semillas • Control de profundidad
	Sistemas de Riego	• Riego por goteo • Control de caudal • Monitoreo en tiempo real	• Ahorro de agua • Riego selectivo • Gestión de recursos
Sistemas de Clasificación	Cosechadoras con Sensores	• Mapeo de rendimiento • Control de calidad • Separación automática	• Cosecha selectiva • Medición de producción • Optimización de recursos
	Clasificadores Automatizados	• Visión artificial • Sistemas de pesaje • Bandas transportadoras	• Control de calidad • Selección por tamaño • Clasificación automática

1.3.2 Sistemas de clasificación automática

Los sistemas de clasificación automática representan una parte fundamental de la agricultura de precisión, permitiendo la categorización y selección eficiente de productos agrícolas basándose en diferentes parámetros como tamaño, color, forma, peso y calidad. Estos sistemas han evolucionado significativamente con la incorporación de tecnologías avanzadas como la visión artificial, sensores especializados y actuadores de alta precisión. Los siguientes son los componentes principales por los cuales esta conformados [16]:

- Sistemas de visión artificial: Cámaras multiespectrales, sensores ópticos, procesamiento de imágenes en tiempo real.
- Sistemas de control: PLCs (Controladores Lógicos Programables), sistemas embebidos, software de control y monitoreo.
- Sistemas mecánicos: Bandas transportadoras, mecanismos de separación, actuadores.

a. Actuadores en sistemas de clasificación

Los actuadores son componentes críticos que ejecutan las acciones físicas necesarias para la clasificación y separación de productos. A continuación, se presenta la Tabla 3 que es una comparativa de los principales tipos de actuadores utilizados [17].

Tabla 3. Comparativa de actuadores en sistemas de clasificación agrícola [17]

Tipo de actuador	**Principio de funcionamiento**	**Ventajas**	**Desventajas**	**Aplicaciones típicas**
Pistones neumáticos	Aire comprimido	• Alta velocidad de respuesta • Bajo costo • Mantenimiento simple	• Consumo de aire • Ruido • Precisión limitada	• Expulsión de productos • Separación binaria
Actuadores eléctricos	Motores eléctricos	• Alta precisión • Control preciso • Operación silenciosa	• Mayor costo • Mantenimiento complejo • Velocidad limitada	• Posicionamiento preciso • Clasificación múltiple

Brazos robóticos	Servomotores múltiples	• Alta flexibilidad • Movimientos complejos • Precisión extrema	• Costo muy elevado • Programación compleja • Mantenimiento especializado	• Clasificación delicada • Manipulación compleja

b. Tipos de servo motores y microcontroladores

Los servomotores son componentes fundamentales en los sistemas de automatización y control de movimiento, especialmente en aplicaciones que requieren precisión y control exacto de posición. Estos dispositivos electromecánicos se caracterizan por su capacidad de mantener una posición angular específica y ajustarla según las señales de control recibidas. En el mercado actual, existe una amplia variedad de servomotores que se distinguen por sus características técnicas como el esfuerzo de torsión (torque), velocidad de operación, voltaje de funcionamiento y dimensiones físicas. La selección del servomotor adecuado depende de los requerimientos específicos de la aplicación, considerando factores como la carga a mover, la precisión requerida y las restricciones de espacio y peso. A continuación, se presenta la Tabla 4 una comparativa de diferentes modelos de servomotores [17].

Tabla 4. Tipos de servo motores y sus características [18] [19]

Marca	Modelo	Esfuerzo de torsión	Velocidad	Voltaje	Corriente	Peso	Dimensiones (mm)
TopModel	MS2810MG	2.8 kg·cm	0.12 sec/60	4.8 V	1.0 A	21.3 g	35.0 x 29.7 x 12.0
TowerPro	SG90	1.8 kg·cm	0.1 sec/60	4.8 V	0.27 A	9.0 g	29.8 x 22.6 x 11.8
TowerPro	MG90S	1.8 kg·cm	0.12 sec/60	4.8 V	0.14 A	13.4 g	33.0 x 22.6 x 12.0
TowerPro	MG996R	11.0 kg·cm	0.14 sec/60	6 V	0.9 A	55.0 g	43.2 x 40.3 x 20.0
TowerPro	MG995	12.0 kg·cm	0.15 sec/60	6 V	1.2 A	50.0 g	40.0 x 20.0 x 40.0

Los servomotores anteriormente mencionados requieren de un sistema de control preciso y versátil para su correcta operación. En este contexto, los microcontroladores juegan un papel fundamental como cerebro del sistema de automatización, siendo responsables del procesamiento de señales y control de los actuadores. Estos dispositivos programables integran en un solo chip diferentes recursos como memoria, procesador y periféricos de entrada/salida, permitiendo implementar desde sistemas de

control simples hasta aplicaciones complejas de automatización. A continuación, se presenta la Tabla 5, una comparativa de tres microcontroladores ampliamente utilizados en proyectos de automatización y control.

Tabla 5. Tipos de microcontrolador [20] [21] [22]

Marca	Modelo	Microcontrolador	RAM	Velocidad de reloj	Almacenamiento	Entradas/ Salidas
Arduino	Arduino Mega 2560	ATmega2560	8 KB	16 MHz	256 KB Flash	54 digital, 16 analógicas
Microchip	PIC16F877A	PIC16	368 B	20 MHz	14 KB Flash	33 digital, 8 analógicas
Espressif	ESP32	ESP32	4 GB	240 MHz	4 MB Flash	34 digital, 18 analógicas

1.3.3 Características generales de las fresas en Ecuador

Se conocen más de 20 especies de Fragaria, que varían en cuanto al número de cromosomas, mostrando una importante poliploidía. Las especies silvestres más comunes son diploides, exhibiendo dos juegos de siete cromosomas; otras son tetraploides o hexaploides, y los híbridos más resistentes son octoploides y aún decaploides [23].

Las especies con más cromosomas tienden a ser más robustas y producir frutos de mayor tamaño. De entre las fresas y los fresones más comercializados, encontramos tres tipos que se presentan en la Figura 1 [23]:

- La Fresa: Caracterizada por su pequeño tamaño, con un diámetro de sección ecuatorial alrededor de 20 mm.

- El Fresón Camarosa: Caracterizado por su gran tamaño, firme y rojo. Destaca por su buena calidad y su resistencia.

- El Fresón Douglas: Caracterizado por su tamaño más grande y en forma de globo, deformado en su cresta. Tiene un color rojo intenso y brillante, a menudo con la punta rosada.



Figura 1. Fresas y los fresones más comercializados

a. Generalidades de calidad y postcosecha de fresas

El reglamento técnico SAGPYA 1998, señala que las fresas deben cosecharse como mínimo cuando presentan más de un 50% de su superficie color rojo. No obstante, los azúcares continúan acumulándose hasta el final de la maduración, por lo que la cosecha temprana disminuye la calidad organoléptica. Las fresas deben ser cosechadas con el mayor cuidado para poder llegar al mercado en buenas condiciones. En los meses calurosos, es deseable cosechar en las horas más frescas del día. Los frutos cosechados deben colocarse delicadamente en el recipiente de recolección, separando aquellos defectuosos o fuera de tipo [24].

Los parámetros a tener en cuenta incluyen la apariencia (color, forma, tamaño, ausencia de defectos), firmeza, sabor (sólidos solubles, acidez, aroma), y valor nutricional (vitamina C). Se recomienda niveles de sólidos solubles de 7% y de acidez de 0,8%. El cáliz debe presentarse verde y turgente [25].

La fresa presenta una elevada tasa respiratoria. Por este motivo se recomienda su almacenamiento a 0 - 1ºC y un 90 - 95% de humedad relativa. Bajo estas condiciones su vida de postcosecha es de 7 - 10 días [25].

Las fresas se pueden clasificar por Tamaño y Color de la siguiente manera [25]:

Tamaño

- Grande: Diámetro superior a 25 mm.
- Mediano: Diámetro entre 18 y 25 mm..
- Pequeño: Diámetro inferior a 18 mm.

Color

- Rojo Brillante: Indicativo de frescura y madurez adecuada.
- Rojo Oscuro: Puede ser un signo de sobremadurez.
- Descolorido o con manchas: Generalmente considerado de menor calidad debido a posibles daños o enfermedades.

1.3.4 Clasificación de fresas en grados de calidad

a. Clasificación de fresas en grados de calidad según la norma NMX-FF-062-SCFI-2002

El producto objeto de esta norma en sus diferentes grados de calidad debe cumplir con las especificaciones siguientes [26]:

Requerimientos mínimos

- Estar sanas y de aspecto fresco;
- Estar enteras y bien desarrolladas;
- Al menos con tres cuartas partes de la superficie mostrando un color rosa o rojo tenue;
- Estar limpias, exentas de materia extraña visible;
- Ser de forma, sabor y olor característicos de la variedad;
- Tener consistencia firme;
- Tener pedúnculo con una longitud máxima de 1,5 cm antes del envase;

- Estar prácticamente exentas de magulladuras;
- Exentas de daños por sol;
- Exentas de polvo, tierra o materia orgánica
- Exentas de daños causados por plagas;
- Libres de descomposición, pudrición y moho causado por microorganismos;
- Estar exentas de daños por refrigeración y variaciones en la temperatura; y
- Estar exentas de humedad exterior anormal, salvo la condensación consiguiente a su remoción de una cámara frigorífica.

La fresa objeto de esta norma se clasifica en los grados de calidad que se presentan en la Tabla 6.

Tabla 6. Grados de calidad de la fresa según la norma NMX-FF-062-SCFI-2002 [26]

Grados de calidad	Características
Extra (México 1)	• Las fresas de este grado deben ser de calidad superior y presentar la forma y desarrollo típicos o propios de la variedad. Deben ser uniformes en cuanto a la coloración y tamaño, deben cumplir con los requisitos mínimos. • No deben tener defectos, salvo aquellos superficiales muy leves siempre y cuando no afecten: el aspecto general del producto, su calidad, conservación y presentación en el envase. Se permiten los defectos para esta categoría que se indican en la Tabla 8.
Primera (México 2)	• Las fresas de esta categoría deben cumplir con las especificaciones señaladas en los requisitos mínimos y presentar la forma, desarrollo y coloración típicas o propias de la variedad. • Se permiten los siguientes defectos leves, siempre y cuando no afecten: el aspecto general del producto, su calidad, conservación y presentación en el envase. Se permiten los defectos para esta categoría que se indican en la Tabla 8.
Segunda (México 3)	• Esta categoría comprende las fresas que no pueden clasificarse en las categorías superiores, pero que satisfacen las especificaciones mínimas detalladas en los requisitos mínimos. • Deben satisfacer las características de forma, coloración, desarrollo y madurez típicos de la variedad. • Defectos de forma y color; siempre y cuando el producto tenga las características comunes de la variedad. Se permiten los defectos que se indican para esta categoría en la Tabla 8, defectos en la piel debidos a rozaduras, quemaduras de sol, magulladuras u otras que no excedan el 10 % de la superficie total del fruto.

El tamaño de las fresas se determina con base a su diámetro ecuatorial de acuerdo con la Tabla 7.

Tabla 7, Especificaciones de tamaño [26]

Tamaño	Intervalo de diámetro ecuatorial (cm)		
A	3,2	De	mayor
B	2,6	A	3,1
C	2,0	A	2,6
D	1,6	A	1,9

Las tolerancias con respecto a la calidad y tamaño de las fresas que no cumplan con las especificaciones de los grados indicados, se determinan en porcentajes de unidades o de masa sobre el total de productos contenidos en el mismo envase, mediante el conteo de unidades o por pesada (determinación de masa) de las mismas respecto al total del envase, admitiéndose las tolerancias indicadas en la Tabla 8 para cada categoría, estas tolerancias son admitidas previo al envasado final [26].

Tabla 8. Tolerancias de daños permitidos [26]

Especificaciones	Tolerancias					
	Extra		Primera		Segunda	
	P.E.	P.A.	P.E.	P.A.	P.E.	P.A.
Defecto menor	8	10	10	12	10	12
Defecto mayor	5	7	5	7	5	7
Defecto critic0	0	2	2	4	3	5
Pudrición	0	2	0	2	0	2
Total de defectos permitidos	8	10	10	12	10	12

P.E. Punto de embarque

P.A, Punto de arribo

b. Clasificación de fresas en grados de calidad según Instituto Tecnológico Superior de Coalcomán

La elección de los frutos de fresa en su estado óptimo de maduración asegura un producto de alta calidad y prolongada vida útil. Si se recolectan frutos que están fisiológicamente inmaduros, no lograrán desarrollar su color y sabor, resultando en un producto inferior. Por otro lado, si se cosecha un fruto que está sobre maduro, tendrá

una vida comercial reducida y se fermentará con facilidad, afectando a los frutos cercanos. Esta selección se realiza utilizando como referencia el índice de madurez o la tabla de color de la fresa que se presenta en la Figura 2. Se sugiere realizar la recolección cuando el grado de madurez esté entre 3 y 5 [27].

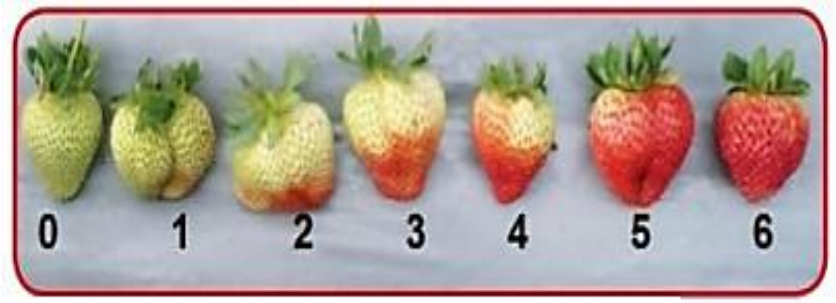


Figura 2. Grados de madurez de las fresas [27]

Para aumentar la rentabilidad tanto del productor, se recomienda clasificar los frutos según criterios de calidad y presentación, asegurando que se cumplan los parámetros mínimos de aceptación o rechazo establecidos por el comercializador. Lo ideal es que la selección se lleve a cabo directamente en el campo, eligiendo los frutos en función de su calidad, grado de madurez y cualquier daño visible [27].

Si la clasificación no se realiza en el campo, sino en el punto de acopio, este debe cumplir con todos los requisitos establecidos por las buenas prácticas de manufactura. Es fundamental contar con mesones de acero inoxidable en buen estado para el proceso de selección, los cuales deben estar previamente lavados y desinfectados. Además, el personal encargado de esta tarea debe estar capacitado en la manipulación de alimentos [27].

Para clasificar los frutos en función de su tamaño y color, se pueden emplear como referencia los requisitos mínimos de calidad definidos por la Norma NTC 4103 de 1997, los cuales son los siguientes [27]:

- Calidad EXTRA: Frutas que presentan una forma característica, son firmes, enteras y tienen un cáliz de color rojo uniforme que cubre entre el 50% y el 80% de su superficie. Deben estar libres de deformaciones, cicatrices y no mostrar hongos ni humedad exterior.

- Calidad I: Frutas con forma característica, firmes, enteras y con un cáliz que presenta un color rojo uniforme entre el 50% y el 80% de su superficie. Pueden tener ligeras deformaciones y cicatrices, pero no deben mostrar hongos ni humedad exterior.

- Calidad II: Frutas de formas irregulares, firmes, enteras y con un cáliz de color rojo uniforme que cubre entre el 50% y el 80% de su superficie. Estas frutas pueden presentar ligeras deformaciones y cicatrices que afecten hasta el 20% de su peso total, y deben estar libres de hongos y humedad exterior.

- Calidad industrial: Frutas de tamaño muy pequeño, de color rojo, con o sin cáliz, que presentan deformaciones y están libres de hongos y humedad.

c. Clasificación de fresas en grados de calidad según el CENICAFE

Al recolectar las fresas, se considera su color como un factor clave. Según CENICAFE, el rango óptimo para la cosecha se encuentra entre los niveles 4 y 5 de su tabla de color, la cual se muestra en la Figura 3. Dado que las fresas no se refrigeran, su tiempo de maduración es muy corto, por lo que es fundamental mantenerlas refrigeradas el mayor tiempo posible. CENICAFE ha establecido seis estados de maduración de la fresa, basados en los cambios en su color externo, que indican la etapa de maduración en la que se encuentran [28]:

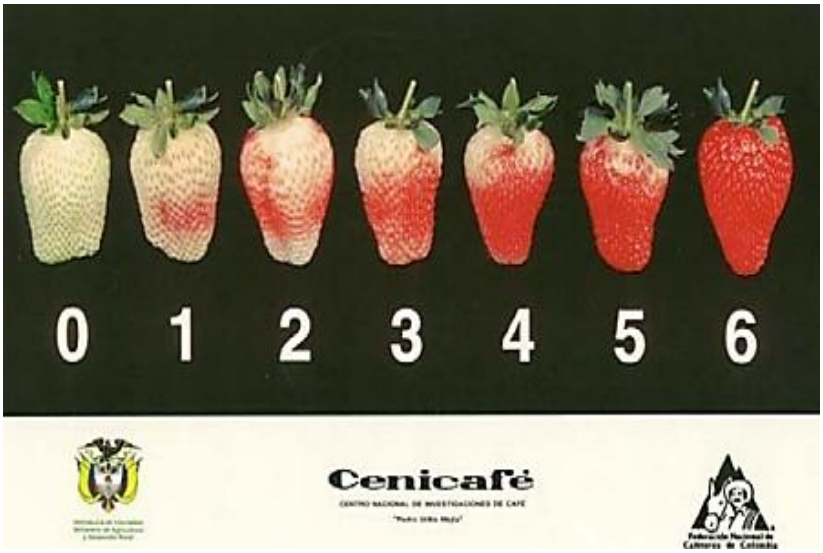


Figura 3. Grados de madures de la fresa según el CENICAFE [28]

- COLOR 0: Fruto de color blanco verdoso, bien desarrollado.
- COLOR 1: El fruto es de color blanco verdoso, con algunas áreas de color rojo en la zona apical.
- COLOR 2: Aumenta el área de color rojo en la zona apical.
- COLOR 3: El color rojo cubre hasta la zona media del fruto y la zona del cáliz presenta visos rosados.
- COLOR 4: Aumenta el área de color rojo hacia la zona del cáliz.
- COLOR 5: La intensidad del color rojo aumenta y empieza a cubrir la región cercana del cáliz.
- COLOR 6: El color rojo es intenso y cubre toda la fruta

Hay tipos diferentes de frutas que se comercializan y en cada uno de estos tipos diferentes categorías.

La fruta fresca para exportación debe ser de la mejor calidad y seleccionarse y empacarse inmediatamente después de la cosecha, considerando su grado de maduración, tamaño, uniformidad y sanidad. No puede ser lavada ni contener suciedad, y se clasifica en tamaños como extragrande (más de 40 mm), grande (35-40 mm), mediana (30-35 mm) y pequeña (25-30 mm). Se empaca en canastitas plásticas de 250 g para el mercado europeo o 400 g para el estadounidense, agrupadas en cajas de cartón de 6 o 12 [28].

Por otro lado, la fruta destinada al mercado nacional incluye aquellas con pequeños defectos o más de 3/4 de maduración que no califican para exportación. Este mercado es menos exigente, permitiendo la venta de frutas pequeñas, especialmente en temporadas de baja cosecha. El empaque para el mercado nacional utiliza canastitas plásticas, pero se transporta en cajas de madera con capacidad para 30 canastitas. Algunas frutas se venden en bolsas plásticas sin selección y en estado avanzado de deterioro, considerándose de uso industrial. La fruta fresca debe mantenerse en cámara

fría entre 0 y 20 °C con 85-90% de humedad relativa, y no puede conservarse más de 4 días [28].

Finalmente, la fruta de industria incluye aquellas por excesiva maduración, defectos menores o tamaño pequeño, a las que se les quita el cáliz y se empacan en bolsas de 5 kg para el mercado, pudiendo ser congeladas y almacenadas para uso posterior [28].

1.3.5 Inteligencia Artificial

La Inteligencia Artificial (IA) es la capacidad de las máquinas para usar algoritmos, aprender de los datos y tomar decisiones de manera similar a los humanos. A diferencia de las personas, la IA puede trabajar sin descanso y manejar grandes volúmenes de información con menor margen de error. Su capacidad para aprender y tomar decisiones ha permitido que realice tareas antes exclusivas de los humanos, mejorando la eficiencia en muchos ámbitos. La IA impactará en la salud, la educación y los negocios, permitiendo que los humanos se concentren en tareas más complejas. En la Figura 4, se muestran las ramas en las que se divide la Inteligencia Artificial [29].

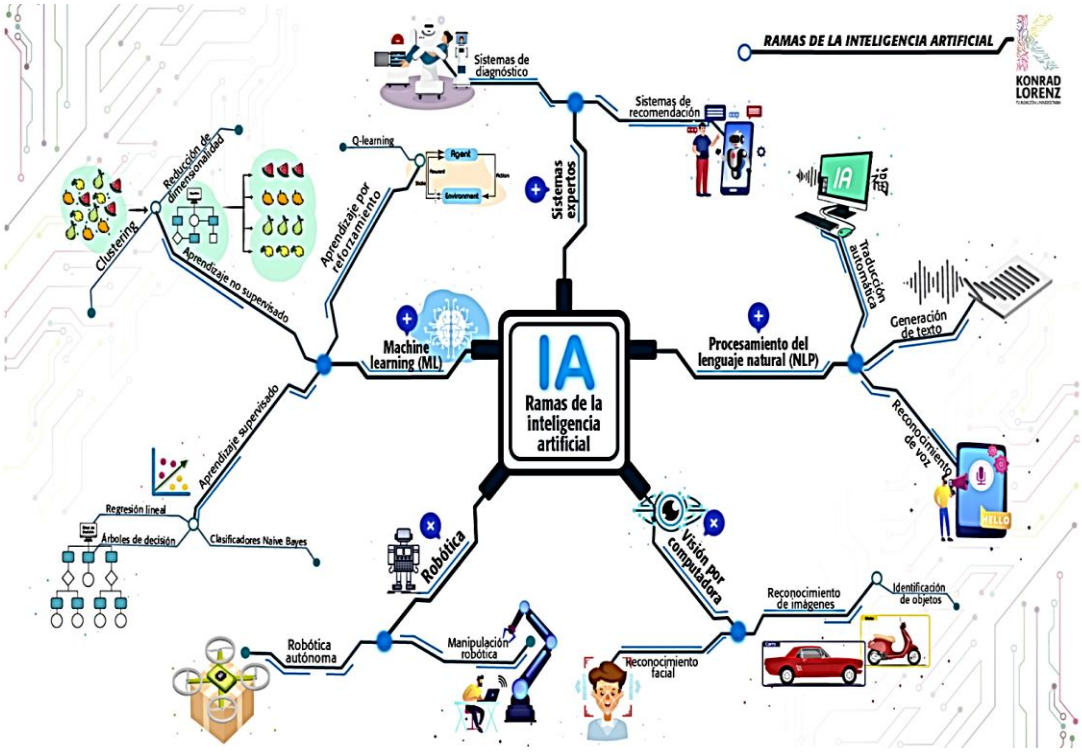


Figura 4. Ramas de la inteligencia artificial [30]

a. Aprendizaje automático

El aprendizaje automático, o machine learning, es un enfoque fundamental de la inteligencia artificial que permite a las máquinas aprender de los datos sin estar explícitamente programadas para ello. A diferencia de los primeros ordenadores que requerían instrucciones específicas, los dispositivos basados en aprendizaje automático adquieren conocimientos a partir de la experiencia y el uso, proporcionando una experiencia personalizada al usuario. Un ejemplo actual es la personalización en redes sociales como Facebook o los resultados de búsqueda de Google. El aprendizaje automático utiliza algoritmos para identificar patrones en los datos, como los filtros de spam que identifican y separan correos no deseados, demostrando cómo puede aplicarse para tomar decisiones basadas en el conocimiento adquirido. En la Tabla 9 se presenta los tres tipos de aprendizaje automático [31].

Tabla 9. Tipos de aprendizaje automático [32]

Tipos de aprendizaje automático	Definición
Aprendizaje supervisado	Los algoritmos usan datos que ya han sido etiquetados u organizados previamente para indicar cómo tendría que ser categorizada la nueva información. Con este método, se requiere la intervención humana para proporcionar retroalimentación.
Aprendizaje no supervisado	Los algoritmos no usan ningún dato etiquetado u organizado previamente para indicar cómo tendría que ser categorizada la nueva información, sino que tienen que encontrar la manera de clasificarlas ellos mismos. Por tanto, este método no requiere la intervención humana.
Aprendizaje por refuerzo	Los algoritmos aprenden de la experiencia. En otras palabras, tenemos que darles «un refuerzo positivo» cada vez que aciertan.5 La forma en que estos algoritmos aprenden se puede comparar con la de los perros cuando les damos «recompensas» al aprender a sentarse, por ejemplo.

b. Redes Neuronales

Las redes neuronales artificiales (RNA) constituyen un campo de la inteligencia artificial inspirado en el funcionamiento del cerebro humano. Estas redes buscan

replicar la capacidad del cerebro para aprender y procesar información mediante la interconexión de unidades básicas denominadas neuronas artificiales. Modelos matemáticos que procesan información a través de una arquitectura compuesta por capas [33]. Cada capa contiene un conjunto de neuronas artificiales que transforman los datos de entrada en salidas mediante una función de activación, además existen diferentes tipos de redes neuronales especializadas en una tarea en específico las cuales se presentan en la Tabla 10.

La estructura básica de una red neuronal se compone de [33]:

- Capa de entrada: Recibe los datos iniciales.
- Capas ocultas: Realizan cálculos intermedios para descubrir patrones complejos.
- Capa de salida: Genera los resultados finales.

Tabla 10. Tipos de Redes Neuronales [34] [35]

Tipos	**Descripción**
Redes Neuronales Perceptrón Multicapa (MLP)	Son el modelo más básico, compuesto por varias capas densamente conectadas. Se utilizan para tareas como clasificación y regresión.
Redes Neuronales Convolucionales (CNN)	Diseñadas para analizar datos estructurados en grillas, como imágenes. Utilizan capas convolucionales para extraer características locales.
Redes Neuronales Recurrentes (RNN)	Adecuadas para secuencias de datos, como texto o series temporales. Cuentan con bucles que permiten almacenar información previa.
Redes Generativas Antagónicas (GAN	Compuestas por un generador y un discriminador que trabajan de forma competitiva para crear datos sintéticos.

c. Herramientas para entrenar Redes Neuronales

Entrenar modelos de aprendizaje automático en servidores en línea ofrece varias ventajas clave. Primero, el acceso a recursos potentes permite utilizar hardware de alto rendimiento, como múltiples GPUs y TPUs, lo que acelera el entrenamiento de modelos grandes y complejos. La escalabilidad es otra ventaja, ya que se pueden ajustar los recursos según las necesidades del proyecto sin invertir en hardware físico [36].

La colaboración se facilita gracias a la infraestructura en la nube, permitiendo que equipos geográficamente dispersos trabajen juntos en el mismo entorno. Además, los proveedores de servicios en la nube se encargan del mantenimiento y actualizaciones, asegurando que los sistemas estén siempre optimizados [36].

Las GPU son esenciales para realizar cálculos paralelos, procesando miles de operaciones simultáneamente y mejorando la precisión de los modelos. Por otro lado, las TPU están diseñadas específicamente para aprendizaje automático, ofreciendo un rendimiento superior y mayor eficiencia energética en comparación con las GPU. Esto no solo reduce costos operativos, sino que también permite escalar el entrenamiento de modelos complejos de manera más efectiva. En la Tabla 11, se presentan una comparación entre las herramientas más utilizadas para ejecutar entrenamientos en la nube.

Tabla 11. Herramientas para ejecutar código en nube [37] [38]

Característica	**Google Colab**	**Google Colab Pro**	**Kaggle Kernels**
Costo	Gratuito	$9.99/mes (aproximadamente)	Gratuito
Acceso a GPU	Limitado (hasta 12 horas)	Acceso a GPU más potente y prolongado	Acceso a GPU (limitado)
Acceso a TPU	Sí	Sí	No
Tiempo de Ejecución	Sesiones de hasta 12 horas	Sesiones de hasta 24 horas	Sesiones de hasta 9 horas
Almacenamiento	Integración con Google Drive	Integración con Google Drive	Almacenamiento en Kaggle
Colaboración	Colaboración en tiempo real	Colaboración en tiempo real	Colaboración en tiempo real
Recursos de Memoria	Limitados	Mayor capacidad de RAM	Limitados
Bibliotecas Preinstaladas	Sí	Sí	Sí
Interfaz	Jupyter Notebook	Jupyter Notebook	Interfaz de Kaggle
Uso Común	Aprendizaje automático, análisis de datos	Proyectos más intensivos de ML	Competencias de ciencia de datos

1.3.6 Visión artificial

La visión artificial, es una disciplina dentro de la inteligencia artificial que se enfoca en desarrollar sistemas capaces de interpretar y entender información visual obtenida a través de imágenes o vídeos. Su objetivo es emular la capacidad humana de ver y comprender el entorno, en la Figura 5 se presenta el diagrama básico de la visión

artificial. Esta tecnología se compone de diversas etapas y elementos, los cuales se detallan en la Tabla 12 y Tabla 13, respectivamente. [39].

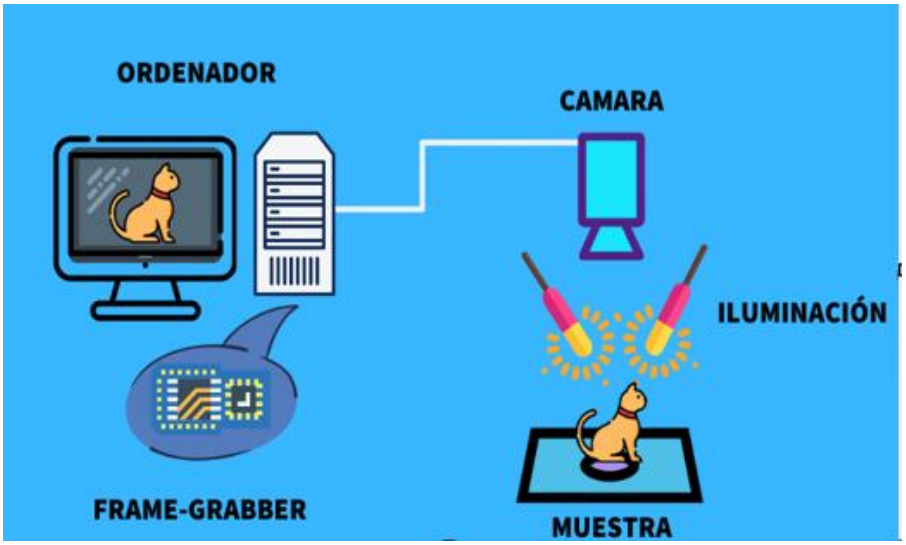


Figura 5. Diagrama básico de la visión artificial [39]

Tabla 12. Etapas de un proceso de visión artificial [40]

Etapa	Descripción
Adquisición de Imágenes	Es la etapa inicial donde se capturan las imágenes o vídeos del entorno utilizando dispositivos como cámaras, sensores o escáneres.
Preprocesamiento	En esta etapa, se realizan operaciones para mejorar la calidad de las imágenes, como corrección de iluminación, eliminación de ruido o ajuste de contraste.
Segmentación	Se divide la imagen en regiones o segmentos significativos, identificando áreas de interés que puedan contener objetos de interés.
Extracción de Características	Se identifican y se extraen características relevantes de las regiones segmentadas, como bordes, formas, texturas o colores.
Reconocimiento o Clasificación	Se clasifican los objetos o patrones presentes en la imagen utilizando algoritmos de aprendizaje automático o técnicas de comparaciones basadas en características previamente extraídas.
Interpretación	Se interpreta la información obtenida para tomar decisiones o realizar acciones específicas, como el control de procesos industriales, la detección de objetos en entornos robóticos o el reconocimiento de patrones en aplicaciones médicas.

Tabla 13. Componentes de la visión artificial [41]

Componente	Descripción
Sensores	Dispositivos electrónicos que capturan imágenes o vídeos del entorno, como cámaras digitales, cámaras de vídeo o escáneres.
Hardware de Procesamiento	Equipos informáticos que procesan y analizan las imágenes, como ordenadores personales, sistemas embebidos o dispositivos móviles.
Software de Procesamiento	Programas y algoritmos diseñados para procesar y analizar imágenes, incluyendo bibliotecas de visión por computadora como OpenCV, TensorFlow o PyTorch.
Algoritmos de Visión por Computadora	Conjunto de técnicas y algoritmos utilizados para realizar tareas específicas de procesamiento de imágenes, como filtrado, segmentación, detección de bordes, extracción de características, reconocimiento de patrones, entre otros.
Interfaz de Usuario	Componente que permite la interacción entre el sistema de visión artificial y el usuario final, proporcionando herramientas para la visualización, configuración y control del proceso.

a. Importancia de las microcomputadoras en la visión artificial

Las microcomputadoras son dispositivos compactos que integran procesador, memoria e interfaces de entrada/salida, desempeñando un papel crucial en el ámbito de la visión artificial. Estos dispositivos ofrecen múltiples beneficios en sistemas basados en inteligencia artificial que se presentan en la Tabla 14.

Tabla 14. Importancia de las microcomputadoras en la visión artificial [42] [43] [44]

Aspecto	Descripción
Procesamiento Local	Permiten el procesamiento de datos en el dispositivo, reduciendo la latencia y mejorando la rapidez en aplicaciones críticas, como vehículos autónomos y sistemas de vigilancia.
Costo-Efectividad	Son más asequibles que estaciones de trabajo o servidores, facilitando la implementación de soluciones de visión artificial para investigadores y pequeñas empresas, democratizando el acceso a la tecnología.
Eficiencia Energética	Diseñadas para consumir poca energía, son ideales para aplicaciones móviles y dispositivos IoT, permitiendo un funcionamiento prolongado en drones y cámaras de seguridad.
Flexibilidad y Personalización	Ofrecen múltiples opciones de conectividad y expansión, permitiendo a los desarrolladores adaptar sus sistemas a necesidades específicas mediante la integración de diversos sensores y módulos.
Comunidad y Recursos	Cuentan con comunidades activas y recursos en línea, facilitando el aprendizaje y desarrollo de proyectos de visión artificial a través de tutoriales y bibliotecas de software.
Capacidades de IA	Equipadas para ejecutar modelos de aprendizaje profundo, permiten implementar algoritmos complejos de visión artificial, mejorando la eficiencia y rapidez en el procesamiento de imágenes.

A continuación, se presenta la Tabla 15, que detalla diferentes tipos de microcomputadoras y sus características específicas. Estas microcomputadoras son ampliamente utilizadas en aplicaciones de visión artificial, cada una con sus propias capacidades en términos de procesador, memoria, almacenamiento y conectividad.

Tabla 15. Tipos de microcomputadoras y sus características [45] [46]

Marca	Modelo	Procesador	RAM	Almacenamiento	Conectividad	Dimensiones (mm)
Raspberry Pi	Raspberry Pi 4 Model B	Broadcom BCM2711, Quad-core Cortex-A72	2GB/4GB/8 GB	microSD	Ethernet, Wi-Fi, Bluetooth	85.6 x 56.5 x 19.5
NVIDIA	Jetson Nano	Quad-core ARM Cortex-A57	4 GB	16 GB eMMC	Ethernet, Wi-Fi (opcional)	100 x 80 x 29
NVIDIA	GTX 1660 Ti	TU116 (arquitectura Turing)	6 GB	Del equipo	HDMI, DisplayPort, DVI-D	223 x 121 x 40 mm

1.3.7 Procesamiento de imágenes

El procesamiento de imágenes es una rama de la visión por computadora que se encarga de manipular y analizar imágenes digitales con el fin de extraer información

útil o realizar tareas específicas. Dos técnicas importantes en este campo son la segmentación semántica y la detección de objetos.

a. Segmentación semántica

La segmentación semántica es una técnica que consiste en dividir una imagen en regiones semánticamente significativas, asignando una etiqueta a cada píxel según su clase o categoría. En la segmentación semántica se clasifican todos los píxeles de la imagen en categorías como carretera, peatón, vehículo, árbol, etc. Esto permite comprender la escena de manera más detallada y precisa, en la Figura 6 se presenta el proceso básico que se realiza con la segmentación semántica. Algunos enfoques comunes de segmentación semántica incluyen las redes neuronales convolucionales (CNN) y técnicas basadas en grafos, como los modelos de Markov aleatorio (MRF) y campos aleatorios condicionales (CRF) [47].

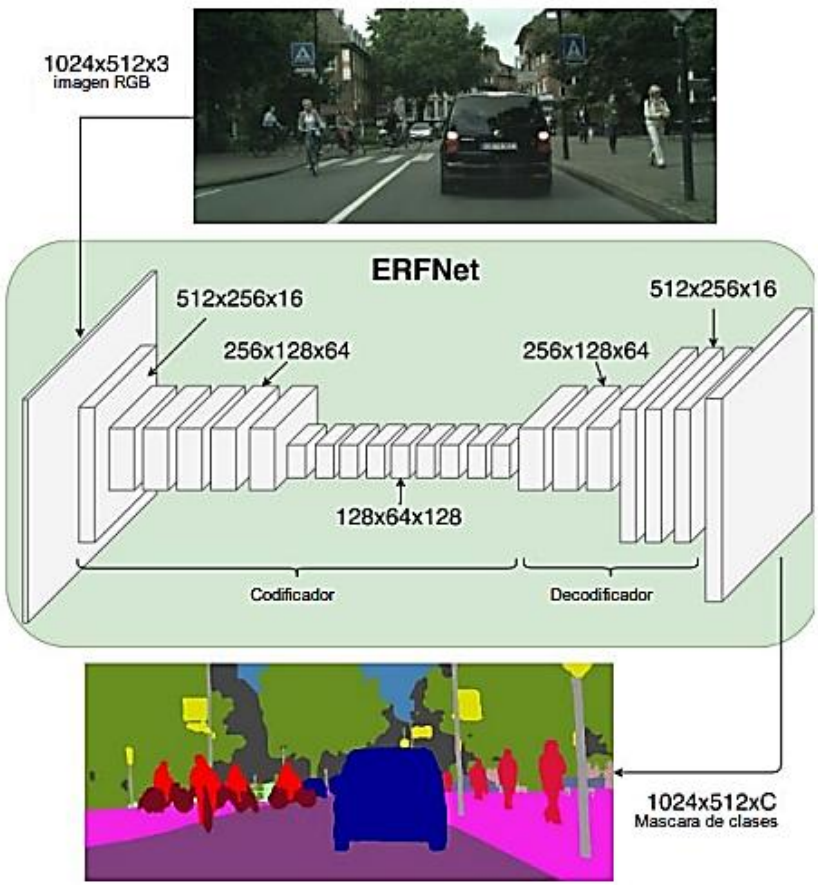


Figura 6. Segmentación semántica [48]

b. Detección de objetos

La detección de objetos es una técnica que consiste en identificar y localizar la presencia de objetos específicos dentro de una imagen. La detección de objetos localiza y delimita cada objeto individualmente mediante un cuadro delimitador (bounding box) y le asigna una etiqueta. Los algoritmos de detección de objetos suelen basarse en el uso de características visuales, como bordes, texturas y colores, junto con algoritmos de aprendizaje automático, como las redes neuronales convolucionales (CNN) y métodos de extracción de características, como los descriptores de características basados en puntos de interés, como SIFT (Scale-Invariant Feature Transform) o SURF (Speeded-Up Robust Features). En la Figura 7 se muestra la manera en que se realiza a detección de objetos en el caso de frutas en un árbol [49].

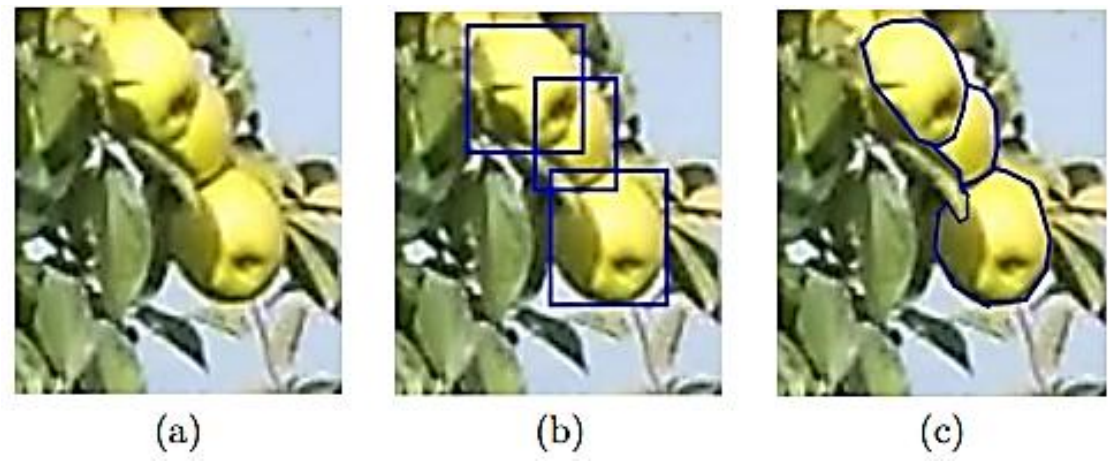


Figura 7. Oclusión en fruta. a) Imagen Original b) Etiquetas rectangulares c) Etiquetas de segmentos [50]

c. Lenguajes de programación utilizados en Inteligencia Artificial

La inteligencia artificial (IA) es un campo multidisciplinario que involucra diversas técnicas y enfoques para desarrollar sistemas capaces de realizar tareas que normalmente requieren inteligencia humana. La elección del lenguaje de programación es crucial, ya que influye en la eficiencia, la facilidad de implementación y la accesibilidad de las herramientas y bibliotecas disponibles. A continuación, En la Tabla 16 se presentan algunos de los lenguajes de programación más utilizados en IA y sus características.

Tabla 16. Lenguajes de programación utilizados en inteligencia artificial [51] [52] [53]

Lenguaje	Facilidad de Aprendizaje	Bibliotecas/Frameworks	Ventajas	Desventajas
Python	Alta	TensorFlow, PyTorch, Keras, Scikit-learn)	• Sintaxis clara y legible • Gran comunidad • Amplia variedad de bibliotecas	• Rendimiento más bajo en comparación con C/C++
Java	Moderada	Weka, Deeplearning4j, MOA	• Portabilidad • Buen rendimiento en sistemas grandes	• Verboso en comparación con Python
C++	Baja	Dlib, Shark	• Alto rendimiento • Control sobre recursos	• Curva de aprendizaje empinada

1.3.8 Adquisición y digitalización de imágenes

La adquisición y procesamiento de imágenes es un componente crucial en el desarrollo de modelos de inteligencia artificial (IA), especialmente en el ámbito del aprendizaje profundo (deep learning). A continuación en la Tabla 17 se presenta las etapas fundamentales de la adquisición y el procesamiento de imágenes.

Tabla 17. Etapas fundamentales de la adquisición y el procesamiento de imágenes [54] [55]

Etapas	Subetapas
Adquisición de Imágenes	Dispositivos de Captura: Estos pueden ser cámaras digitales, cámaras térmicas, utilizadas para capturar imágenes en diversas condiciones de iluminación y escenarios.
	Conversión Analógica-Digital: Las señales se convierten en datos digitales mediante un convertidor analógico-digital (CAD), que realiza el muestreo y cuantificación
	Formato de Almacenamiento: Las imágenes se almacenan en formatos digitales (JPEG, PNG, TIFF) que permiten la compresión y el manejo eficiente de datos.
Procesamiento de Imágenes	Preprocesamiento:Se ajustan las dimensiones de las imágenes para que sean consistentes y adecuadas para el modelo de IA.

	Segmentación: Separar los objetos de interés del fondo en la imagen. Esto para la detección de objetos y el reconocimiento de patrones.
	Extracción de Características: Identificar y extraer características relevantes de las imágenes que ayudarán al modelo a aprender patrones significativos. Esto puede incluir bordes, texturas y formas.
	Representación: Las imágenes se representan como matrices de píxeles, donde cada píxel contiene información sobre su color y brillo. La representación puede ser en diferentes espacios de color (RGB, HSV)

a. Dispositivos utilizados en la adquisición de imágenes.

Es fundamental comprender las características técnicas de los diferentes dispositivos disponibles para la adquisición de imágenes. En Tabla 18, se presenta una comparación de diversas cámaras web y dispositivos móviles, detallando sus especificaciones clave como resolución, velocidad de captura (FPS), tipo de conexión y rango de precios.

Tabla 18. Tipos de Cámaras [56] [57] [58]

Cámara	Resolución	FPS	Tipo de conexión	Precio aproximado
Logitech HD Pro C920	1080p (1920x1080)	30 FPS	USB	$70 - $100
Microsoft LifeCam HD-3000	720p (1280x720)	30 FPS	USB	$50 - $70
Samsung A10 (Cámara trasera)	1080p (1920x1080)	30 FPS	N/A	$80 - $100
Logitech StreamCam	1080p (1920x1080)	60 FPS	USB-C	$150 - $170

b. Software de etiquetado de imágenes

El software de etiquetado de imágenes representa una herramienta fundamental en el campo de la visión artificial y el aprendizaje automático, constituyendo un elemento crítico en la preparación de datos para el entrenamiento de modelos de inteligencia artificial. Este proceso sistemático de anotación permite la identificación y marcado de elementos específicos dentro de las imágenes, estableciendo así la base para el desarrollo de sistemas de reconocimiento visual avanzados [59].

En el contexto del aprendizaje automático, el etiquetado de imágenes abarca diversos tipos de anotaciones, cada uno diseñado para satisfacer diferentes necesidades de los

modelos de IA. Estas incluyen la clasificación simple, donde se asignan etiquetas generales a imágenes completas; la detección de objetos, que implica la localización precisa de elementos mediante cuadros delimitadores; la segmentación semántica, que divide la imagen en regiones significativas; y la anotación de puntos clave, utilizada para identificar ubicaciones específicas de interés dentro de la imagen [59].

La arquitectura de los sistemas de etiquetado modernos se compone de tres elementos principales: la interfaz de usuario, que proporciona las herramientas necesarias para la visualización y anotación; el motor de etiquetado, que gestiona los algoritmos y procesos de anotación; y el sistema de almacenamiento, responsable de la gestión y exportación de datos. Esta estructura permite un flujo de trabajo eficiente y la gestión efectiva de proyectos de etiquetado a diferentes escalas [59].

Los formatos de anotación juegan un papel crucial en la interoperabilidad y utilidad de los datos etiquetados. Estándares como PASCAL VOC, COCO y YOLO han emergido como formatos predominantes, cada uno con sus propias ventajas y casos de uso específicos. La elección del formato apropiado depende de factores como el tipo de tarea de aprendizaje automático, los requisitos del modelo y las necesidades de integración del sistema [60].

El mercado actual ofrece diversas soluciones de software para el etiquetado de imágenes, cada una con características específicas que se adaptan a diferentes necesidades. Estas herramientas varían desde opciones gratuitas y de código abierto hasta plataformas empresariales completas basadas en la nube. A continuación en la Tabla 19, se presenta comparativa de las principales herramientas de etiquetado disponibles.

Tabla 19. Tipos de software para etiquetado de imágenes [61] [62] [63] [64]

Software	Características principales	Formatos de salida	Ventajas	Desventajas
LabelImg	• Open source • Etiquetado de imágenes manual • Compatible con XML y TXT	XML (PASCAL VOC), YOLO (TXT)	• Gratuito y de código abierto • Soporte para imágenes comunes	• Requiere etiquetado manual • Sin herramientas avanzadas
Supervisely	• Plataforma en la nube • Anotación colaborativa • Soporte para proyectos complejos	JSON, COCO, VOC, YOLO	• Ideal para proyectos grandes • Funciones avanzadas de edición	• Costo elevado para grandes equipos • Requiere internet
Labelme	• Open source • Compatible con múltiples tipos de etiquetas (polígonos, líneas, puntos, etc.)	JSON	• Ligero y de fácil uso • Compatible con Python para flujos automatizados	• No soporta anotaciones de video • Sin colaboración en equipo
Roboflow	• Plataforma en la nube • Anotación colaborativa • Preprocesamiento y aumento de datos • Interfaz gráfica amigable	YOLO, COCO, VOC, TFRecord, entre otros	• Exporta a múltiples formatos • Flujo completo en un solo lugar • Ideal para IA	• Algunas funciones son de pago • Requiere conexión a internet

1.3.9 Análisis comparativo sobre las técnicas de visión artificial empleados en la clasificación de frutas

En la Tabla 20, se presenta un análisis comparativo de diversas técnicas de visión artificial utilizadas en la clasificación de frutas. En esta tabla se presentan los equipos, algoritmos se utilizaron en los trabajos de investigación, incluyendo la efectividad y precisión de cada sistema en la tarea de clasificación, destacando la relevancia de la visión por computadora en la mejora del control de calidad en la industria frutal.

Tabla 20. Técnicas de visión artificial empleados en la clasificación de frutas [65] [66] [67] [68] [69] [70]

Autores	Título	Equipos	Algoritmo	Exactitud
León León, Ryan Abraham; Boy Diaz, Carlos Sebastian; Gonzalez Palacios, José Jesús	Desarrollo de un Algoritmo de visión artificial para la Clasificación de Fresas de Exportación o Consumo Nacional	Lenovo con un procesador Intel i7 y Windows 11; cámara Logitech c920	Programación realizada en Python. Utilizando la red YOLOv5 en su modelo YOLOv5x; entrenado en Google Colab	Eficiencia general en ambas clasificaciones del 97.14% con un margen de error del 2.86%.
Patricia Nataly Constante Procel; Andres Marcelo Gordon Garces	Diseño e implementación de un sistema de visión artificial para clasificación de al menos tres tipos de frutas	Cámara Logitech HD Pro C920	Software elegido OpenCV; Entrenamiento RNA con imágenes 40 de fresas, moras, uvillas	Porcentaje de reconocimiento del 87.5%
Juan José Camargo Robles	Sistema Clasificador de Fresa Usando Técnicas de Inteligencia Artificial	AMD Ryzen 5 3500U; GPU: AMD Radeon™ Vega 8; RAM de 12 GB	YOLOv8; el aumento de datos en el proyecto se implementó utilizando la herramienta RoboFlow	Precisión sin aumento de datos 96.3%; precisión con aumento de datos 97.08%
García Pérez, Luis Alberto; Martínez López, Ana Sofía	Clasificación Automática de Fresas Mediante Redes Neuronales Convolucionales	PC con procesador Intel i5, GPU NVIDIA GTX 1660 Ti; cámara Sony Alpha 6000	Modelo CNN personalizado implementado con TensorFlow y Keras; el dataset incluyó 2000 imágenes de fresas	Precisión del 94.8% en la clasificación de fresas maduras y no maduras.
Rodríguez Gómez, María Fernanda; Hernández Ruiz, Carlos Eduardo	Sistema de Clasificación de Fresas Basado en visión artificial y Aprendizaje Profundo	Raspberry Pi 4 con módulo de cámara oficial; procesador ARM Cortex-A72	YOLOv4 Tiny entrenado con un conjunto de datos de fresas recolectadas manualmente; preprocesamiento realizado con OpenCV	Precisión del 92.5% en la detección de fresas aptas para consumo y rechazo.
Liao Yue, Li Lerong, Xiao Huiqiang, Feijiano Xu, Shan Bochen y Yin Hua	YOLO-MECD: Algoritmo de Detección de Cítricos Basado en YOLOv11	cámara Honor 70	YOLOv11, YOLO-MECD	Detección de objetos cítricos, con una precisión (P) del 84,4% una tasa de y una precisión media media (mAP) elevada del 81,6% con mejoras mAP de 3.8%, 3.2% y 5.5% en comparación con YOLOv8s, YOLOv9s y YOLOv10s

1.3.10 Redes Convolucionales

Las redes convolucionales (CNNs) son un tipo de arquitectura de red neuronal especialmente eficaz para tareas de procesamiento de datos con estructura de cuadrícula, como las imágenes. Las CNNs son ampliamente utilizadas en el campo del reconocimiento de imágenes y el procesamiento de video. A continuación, se enumera algunos tipos de redes convolucionales en la Tabla 21 .

Tabla 21. Tipos de Redes Convolucionales [71] [33] [72] [73] [74]

Red convolucional	Descripción	Características técnicas
LeNet	Una de las primeras CNNs, desarrollada por Yann LeCun para la tarea de reconocimiento de dígitos escritos a mano (MNIST). Consiste en una arquitectura simple con dos capas de convolución seguidas de capas de pooling y dos capas completamente conectadas.	• Capas: 2 capas de convolución, 2 capas de pooling, 2 capas completamente conectadas. • Activación: Función de activación sigmoid. • Tamaño de entrada: 32x32.
AlexNet	Introducida por Alex Krizhevsky en 2012. Tiene más capas de convolución que LeNet y utiliza ReLU, normalización y dropout.	• Capas: 5 capas de convolución, 3 capas completamente conectadas. • Activación: ReLU. • Normalización: Normalización local de respuesta. • Dropout: Se utiliza para prevenir el sobreajuste. • Tamaño de entrada: 227x227 píxeles.
ResNet (Residual Networks)	Presenta conexiones residuales o "skip connections" quepermiten que las capas aprendan correcciones a la identidad, facilitando el entrenamiento de redes muy profundas (comoResNet50, ResNet101).	• Capas: Hasta 152 capas en ResNet152. • Conexiones residuales: Facilitan el flujo de gradientes. • Batch Normalization: Se utiliza en cada bloque residual. • Tamaño de entrada: 224x224 píxeles.
YOLO (You Only Look Once)	Es una familia de redes convolucionales para la detección de objetos en tiempo real. YOLO divide la imagen en una cuadrícula y predice bounding boxes y probabilidades de clase para cada celda de la cuadrícula.	• Arquitectura: Una única red convolucional que predice bounding boxes y clases. • Velocidad: Detección en tiempo real (hasta 45 FPS en versiones anteriores). • Tamaño de entrada: 416x416 píxeles (varía según la versión). • Versiones: YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, YOLOv8, YOLOv9, YOLOv10, YOLOv11.

1.3.11 YOLO (You Only Look Once)

Es una arquitectura de red neuronal de tipo convolucional (CNN) utilizada para la detección de objetos en imágenes y vídeos en tiempo real. A diferencia de otros métodos tradicionales de detección, como los que utilizan sistemas de región propuesta, YOLO trata la detección de objetos como un problema de regresión única. Esto significa que el modelo predice directamente las clases de los objetos y las cajas delimitadoras correspondientes en una sola ejecución de la red, en lugar de realizar Múltiples etapas de procesamiento. Se han desarrollado diversas versiones de YOLO, cada una incorporando mejoras significativas en el rendimiento y la precisión para la detección de objetos. A continuación, se presenta la Tabla 22 con las versiones de YOLO y sus características principales [74].

a. Características principales

Dentro de las características más importante que tiene Yolo esta las siguientes [74]:

- Rapidez: YOLO es capaz de procesar imágenes en tiempo real debido a su diseño unificado, que reduce el tiempo de inferencia.
- Enfoque unificado: Combina la localización de objetos (bounding boxes) y la clasificación de objetos en una sola red neuronal.
- Detección multiescala: Las versiones más recientes (a partir de YOLOv3) son capaces de detectar objetos de diferentes tamaños en múltiples escalas.
- Generalización: YOLO generaliza bien en datos desconocidos, lo que lo hace útil para aplicaciones prácticas.
- Anclas (Anchors): Desde YOLOv2, se introdujeron las anclas para mejorar la precisión en la predicción de bounding boxes.
- No Maxima Suppression (NMS): Se utiliza para eliminar bounding boxes

Tabla 22. Tabla comparativa de las versiones de Yolo [75] [76]

Versión	Año	Características Técnicas	Ventajas	Limitaciones
YOLOv1	2016	• Arquitectura básica de detección. • Divide la imagen en una cuadrícula S×S. • Cada celda predice 2 bounding boxes y una clase.	• Rapidez (45 FPS). • Arquitectura simple.	• Baja precisión en objetos pequeños. • Problemas con la detección de objetos cercanos.
YOLOv2	2017	• Introducción de anclas (anchors). • Uso de Batch Normalization. • Entrenamiento en múltiples conjuntos de datos (YOLO9000). • Backbone: Darknet-19.	• Mejor precisión que YOLOv1. • Capacidad de detección de 9000 clases.	• Aún presenta problemas con objetos muy pequeños.
YOLOv3	2018	• Backbone: Darknet-53. • Detección a múltiples escalas. • Uso de Sigmoid para la predicción de clases. • Mejor manejo de objetos pequeños.	• Mejor rendimiento en objetos pequeños. • Detección más precisa.	• Más pesado y lento en comparación con YOLOv2.
YOLOv4	2020	• Backbone: CSPDarknet-53. • Introducción de técnicas como Mosaic Data Augmentation, DropBlock y CIoU Loss. • Uso de CSPNet para mejorar la eficiencia.	• Alta precisión y velocidad. • Mejor rendimiento en GPUs modernas.	• Mayor complejidad en el entrenamiento.
YOLOv5	2020	• Desarrollado por Ultralytics (no oficial de los creadores de YOLO). • Variantes: small, medium, large, xlarge. • Optimización para facilidad de uso. • Compatible con PyTorch.	• Fácil de usar. • Variantes ligeras para dispositivos con recursos limitados.	• No es una versión oficial. • Controversia sobre su denominación como YOLO.
YOLOv6	2022	• Enfoque en eficiencia y precisión.	• Mejor relación entre precisión y velocidad que YOLOv5.	• Menor adopción en comparación con YOLOv5 y YOLOv7.

		• Adaptaciones para aplicaciones específicas. • Mejora en la inferencia y el entrenamiento.		
YOLOv7	2022	• Introducción de técnicas avanzadas de optimización. • Mejoras en detección de objetos pequeños y medianos. • Más eficiente que YOLOv6 en términos de precisión y velocidad.	• Precisión y velocidad superiores. • Más eficiente en recursos computacionales.	• Aún en desarrollo y mejora continua.
YOLOv8	2023	• Última versión desarrollada por Ultralytics. • Mejora en la arquitectura para tareas de detección, segmentación y clasificación. • Optimización para hardware moderno.	• Versátil (detección, segmentación y clasificación). • Mejor rendimiento general.	• Complejidad creciente en la arquitectura.
YOLOv9	2024	• Atención basada en Transformers. • Mejor comprensión contextual. • Backbone optimizado para visión.	• Precisión superior en condiciones complejas. • Mejor generalización.	• Mayor demanda computacional.
YOLOv10	2024	• Atención híbrida (CNN + Transformer). • Escalado adaptativo. • Optimizado para Edge A.	• Precisión mejorada para defectos sutiles. • Buen rendimiento en dispositivos de bajo consumo.	• Requiere ajuste fino por aplicación.
YOLOv11	2025	• Atención multiescala. • Aprendizaje adaptativo. • Arquitectura ligera y robusta.	• Excelente en diversas telas e iluminación. • Rendimiento en tiempo real superior.	• Aún en etapa de validación y adopción.

b. Tipos de versiones dentro de Yolov11

YOLOv11 es la versión más reciente de la serie Ultralytics YOLO para detección de objetos en tiempo real, y marca un nuevo estándar en precisión, velocidad y eficiencia. Aprovechando los avances logrados en versiones anteriores, esta iteración incorpora importantes mejoras en su arquitectura y técnicas de entrenamiento. Gracias a estas

innovaciones, YOLOv11 se posiciona como una herramienta altamente adaptable para diversas aplicaciones en el campo de la visión por computadora. En la Tabla 23, se presenta una comparación detallada de las cinco variantes principales de YOLOv11 (nano, small, medium, large y extra-large), destacando sus características distintivas, capacidades y casos de uso más apropiados. Esta diversificación permite a los desarrolladores e investigadores seleccionar la versión más adecuada según sus requisitos específicos de velocidad, precisión y recursos computacionales disponibles [77].

Tabla 23. Comparación de las versiones de Yolov11 [77]

Tipo de YOLOv11	**Descripción**	**Características Clave**	**Uso Común**
YOLOv11n	Versión más ligera y rápida	• Atención eficiente • Bajo consumo de recursos	Dispositivos móviles, IoT
YOLOv11s	Optimizada para velocidad y precisión básica	• Buen rendimiento • Arquitectura compacta	Sistemas embebidos, vigilancia
YOLOv11m	Equilibrio entre precisión y eficiencia	• Precisión intermedia • Más recursos que v11s	Industria y robótica
YOLOv11l	Alta precisión con buen rendimiento	• Excelente para imágenes complejas	Aplicaciones médicas o agrícolas
YOLOv11x	Máxima capacidad y detalle	• Atención multiescala avanzada • Alta demanda computacional	Investigación avanzada, imágenes satelitales

1.3.12 Metodología para Machine Learning: CRISP-DM

RISP-DM (Cross-Industry Standard Process for Data Mining) es una metodología ampliamente utilizada en proyectos de ciencia de datos y machine learning. Fue desarrollada en 1996 y se compone de seis fases estructuradas que guían el desarrollo de soluciones basadas en datos [78]:

- Comprensión del negocio: Se definen los objetivos del proyecto y se identifican los problemas que se desean resolver.

- Comprensión de los datos: Se recopilan, describen y analizan los datos disponibles para evaluar su calidad y relevancia.

- Preparación de los datos: Se seleccionan, limpian y transforman los datos para su uso en modelos de machine learning.
- Modelado: Se seleccionan y entrenan algoritmos adecuados según el problema y los datos preparados.
- Evaluación: Se mide el rendimiento del modelo para asegurar que cumple con los objetivos del negocio.
- Despliegue: El modelo se implementa en el entorno real para generar valor práctico.

CRISP-DM es flexible, iterativa y adaptable, lo que permite mejorar continuamente los resultados.

1.4 Objetivos

1.4.1 Objetivo general

Implementar un sistema de clasificación automático de fresas según su calidad empleando Inteligencia Artificial.

1.4.2 Objetivos específicos

- Identificar las características que determinan la calidad de la fresa.
- Determinar las técnicas actuales de Inteligencia Artificial aplicado en cultivos.
- Implementar un algoritmo de procesamiento de imágenes para clasificar las fresas basándose en sus características de calidad.
- Realizar pruebas de validación del sistema de clasificación automático de fresas.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

En el proyecto de investigación se emplearon diversos tipos de hardware y software para desarrollar el sistema de clasificación automática de fresas según su calidad utilizando inteligencia artificial. Durante la etapa de adquisición de datos, se utilizó la cámara trasera del celular Samsung A10 para capturar imágenes y videos de las fresas, los cuales serían analizados posteriormente. Para el proceso de etiquetado de datos, se empleó la plataforma Roboflow, que permitió generar los archivos necesarios en el formato adecuado para el entrenamiento del modelo. Posteriormente, en la etapa de entrenamiento, se utilizó la red neuronal YOLOv11l en conjunto con Google Colab, aprovechando sus recursos de procesamiento como GPUs y TPUs para acelerar el proceso. Además, se utilizó Google Drive como herramienta de almacenamiento en la nube, lo cual facilitó tanto el acceso como la gestión eficiente de los archivos a lo largo del proyecto.

En el prototipo clasificador de fresas se utilizará una laptop HP OMEN, aprovechando su tarjeta gráfica integrada NVIDIA GeForce GTX 1660 Ti para el procesamiento de datos y la ejecución del modelo de inteligencia artificial, permitiendo realizar inferencias en tiempo real con eficiencia. Además, se utilizó un microcontrolador Arduino Mega para el control de los actuadores, encargados de realizar la clasificación física de las fresas. A continuación, en la Tabla 24 y la Tabla 25 se presentan los materiales utilizados en el proyecto.

Tabla 24. Hardware utilizado en el proyecto de investigación.

Materiales	Características	Descripción
Laptop HP OMEN	• Procesador Intel Core i7 • GPU NVIDIA GeForce GTX 1660 Ti • 16 GB RAM	Ordenador portátil de alto rendimiento, utilizado para ejecutar modelos de inteligencia artificia.
Servo Motor SG90 9GR	• Torque: 1.2 kg/cm • Velocidad: 0.1 s/60° • Tamaño compacto	Servo motor pequeño y ligero, utilizado en robótica y proyectos de automatización para movimientos precisos.
Mg995 Servo Motor Towerpro 12kg	• Torque: 12 kg/cm • Velocidad: 0.2 s/60° • Mayor tamaño y peso	Servo motor de alto torque, adecuado para aplicaciones que requieren mayor fuerza y estabilidad en el movimiento.
Pulsadores	• Alimentación: 5V	Dispositivos que permiten enviar señales al sistema al ser presionados.
Arduino MEGA 2560	• 54 pines digitales • 16 pines analógicos • 256 KB de memoria flash	Microcontrolador versátil y potente, ideal para proyectos que requieren múltiples conexiones y procesamiento.
Samsung A10 (Cámara trasera)	• Resolución: 1080p • Autofoco	Cámara de celular de alta definición, utilizada para video y captura de imágenes; excelente para proyectos de visión artificial.

Tabla 25. Software utilizado en el proyecto de investigación.

Materiales	Características	Descripción
Roboflow	• Herramienta de anotación de imágenes • Interfaz gráfica fácil de usar	Software online utilizado para etiquetar imágenes y crear conjuntos de datos para entrenamiento de modelos de visión artificial.
Python	• Lenguaje de programación de alto nivel • Soporte para bibliotecas de IA	Lenguaje ampliamente utilizado en ciencia de datos, aprendizaje automático y desarrollo de aplicaciones..
Google Drive	• Almacenamiento en la nube • Compartición fácil de archivos	Servicio de almacenamiento en línea que permite guardar y compartir archivos.
Google Colab	• Acceso a GPUs y TPUs • Entorno de desarrollo en la nube	Plataforma que permite ejecutar código Python en la nube, ideal para proyectos de aprendizaje profundo y análisis de datos.
Yolov11l	• Modelo de detección de objetos • Basado en redes neuronales convolucionales	Algoritmo avanzado para la detección y clasificación de objetos en imágenes y videos, utilizado en aplicaciones de visión artificial.

2.2 Métodos

2.2.1 Modalidad de la investigación

En la ejecución del proyecto de investigación, se utilizó la modalidad de investigación aplicada, ya que se buscaba brindar una solución efectiva al problema planteado. Para ello, fue esencial aplicar tanto los conocimientos prácticos como teóricos adquiridos a lo largo de la formación académica universitaria, con el objetivo de desarrollar el sistema propuesto.

La investigación también adoptó un enfoque bibliográfico, sustentándose en antecedentes de libros, tesis, artículos y revistas científicas. Esto garantizó la recopilación de información veraz y actualizada sobre la tecnología empleada, así como de los parámetros requeridos para el desarrollo del proyecto de investigación.

Además, se llevó a cabo una investigación cuantitativa para analizar la eficiencia del sistema de clasificación automática de fresas. Este estudio permitió evaluar métricas clave, como el tiempo de clasificación y el porcentaje de errores, comparando los resultados del prototipo con los métodos de clasificación manual. Esta evaluación posibilitó identificar áreas de mejora y validar la viabilidad del prototipo en un entorno operativo real.

La investigación de campo desempeñó un papel esencial en el proyecto, ya que facilitó el desarrollo, prueba y optimización de los algoritmos de IA en un entorno real. A través de este enfoque, se logró recolectar datos directamente en el contexto operativo, ajustar los modelos mediante iteraciones sucesivas y compararlos con métodos tradicionales para validar la eficacia del sistema. Todo esto para asegurar que el sistema pudiera ser optimizado y probado en condiciones reales, garantizando así su precisión y eficiencia en el entorno práctico de aplicación.

2.2.2 Recolección de información

La recolección de información se basó en los repositorios de las Universidades, información relevante de artículos científicos, libros, revistas científicas, etc. La

investigación se basó en los criterios que se presentan en la Tabla 26. La información fue tomada de fuentes verificadas y actualizadas para tener una buena base teórica para el desarrollo del proyecto de investigación.

Tabla 26. Criterios de recolección de información.

CRITERIO	**DESCRIPCIÓN**
¿Para qué?	Mejorar la clasificación según la calidad de las fresas cosechadas
¿De qué personas u objetos?	De los cultivos de fresas
¿Cómo?	Utilizando la Inteligencia Artificial y procesamiento de imágenes
¿Con qué?	Un sistema de cámaras que implementen Inteligencia Artificial
¿Cuándo?	Periodo académico septiembre 2024 – febrero 2025
¿Dónde?	Huachi Grande

2.2.3 Procesamiento y análisis de datos

Una vez recolectados los datos, se procedió a realizar un proceso que incluyó los siguientes pasos:

- Recolección de la información.
- Revisión de la información obtenida.
- Implementación de los algoritmos y tecnologías encontrados para construir el sistema.
- Análisis de la eficiencia del sistema de clasificación automática de fresas

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

En la Figura 8, se presenta el esquema general del sistema de clasificación automática de fresas que está constituido por cuatro etapas principales. A continuación, se detallan cada una de estas etapas para ofrecer una comprensión más clara de su funcionamiento.

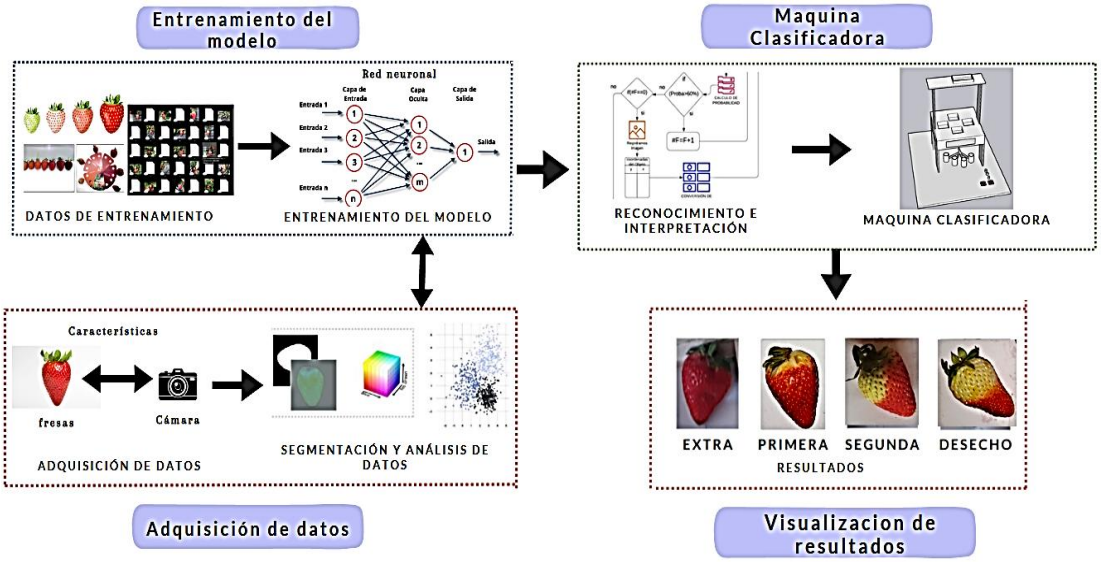


Figura 8. Esquema general de del sistema de clasificación automática de fresas

- **Adquisición de Datos:** En esta etapa, se recopilan datos relevantes sobre las fresas que se van a clasificar. Esto incluye imágenes de las fresas para comparar su color y calidad. Los datos se obtienen a través de una cámara que capturan imágenes en diferentes condiciones de iluminación y desde varios ángulos. Esta información es crucial para entrenar el modelo de inteligencia artificial, ya que proporciona ejemplos de las características que se desean clasificar.

- **Entrenamiento de modelo con IA**: Una vez que se han recopilado los datos, se procede a entrenar un modelo de inteligencia artificial. Este proceso implica la utilización de técnicas de aprendizaje automático para enseñar al modelo a reconocer y clasificar las fresas, basándose en las características extraídas durante la adquisición de datos. El entrenamiento se llevó a cabo utilizando un conjunto de datos compuesto por 1429 imágenes etiquetadas en fresas de calidad "Extra", "Primera", "Segunda" y "Desecho".

- **Máquina Clasificadora:** Una vez que el modelo ha sido entrenado, se implementa en una máquina clasificadora. Esta máquina utiliza el modelo de IA para procesar las imágenes de fresas, a medida que las fresas pasan por la máquina, se capturan imágenes y se analizan utilizando el modelo previamente entrenado. La máquina clasifica automáticamente cada fresa en función de su calidad, esta automatización aumenta la eficiencia y reduce el margen de error en comparación con la clasificación manual.

- **Aplicación de Usuario:** La aplicación de usuario es la interfaz gráfica que presenta el resultado de la clasificación que permite a los operarios o administradores del sistema interactuar con el sistema de clasificación automática. Los usuarios pueden monitorear el funcionamiento de la máquina clasificadora, visualizar estadísticas sobre la producción y la calidad de las fresas clasificadas, y realizar ajustes en los parámetros del modelo si es necesario.

Para llevar a cabo las dos primeras etapas del proyecto, que son la adquisición de datos y el entrenamiento de la inteligencia artificial, se aplicó la metodología CRISP-DM (Cross Industry Standard Process for Data Mining). Esta metodología fue escogida por su enfoque estructurado y flexible, ideal para proyectos de análisis de datos e inteligencia artificial. CRISP-DM permite desarrollar modelos eficientes al dividir el proceso en fases claras: comprensión del negocio, comprensión de la información, preparación de los datos, entrenamiento y validación. Esto facilitó una planificación ordenada, asegurando la calidad del conjunto de datos y un entrenamiento adecuado del modelo de clasificación.

3.1 Comprensión del negocio: Análisis de la producción de fresa en el Ecuador

La comprensión del negocio es el primer paso esencial en la metodología CRISP-DM, ya que permite identificar los objetivos del proyecto en función del contexto real. En este caso, se analizó la producción de fresa en Ecuador, un cultivo de gran importancia económica y agrícola en varias regiones del país, especialmente en la Sierra. A pesar de su demanda, el proceso de clasificación de la fruta sigue siendo manual y subjetivo, lo que puede afectar la calidad final del producto y su valor en el mercado.

3.1.1 Zonas y épocas donde la producción de la fresa es mayor

Las fresas se cultivan principalmente en la región Sierra, destacándose las siguientes provincias: Pichincha, donde Cayambe, Tabacundo y Pedro Moncayo son las zonas principales donde se lleva a cabo la producción de este fruto; Imbabura, con cantones como Otavalo y Cotacachi también tienen una producción significativa; Cotopaxi, donde La Maná es un punto clave para el cultivo; y Tungurahua, donde Ambato y sus alrededores participan en la producción en menor escala. Estas áreas tienen altitudes y climas templados que favorecen el desarrollo de fresas de alta calidad [79].

En cuanto a la época de cosecha, la primera temporada que va de octubre a diciembre, cuando las condiciones climáticas son óptimas y se produce la mayor cantidad de fresas. La segunda temporada ocurre de febrero a abril, en la cual, aunque la producción es menor, sigue siendo significativa en ciertas zonas [79].

3.1.2 Producción de la fresa en el Ecuador

La producción de fresas en Ecuador tiene un enfoque principalmente destinado al mercado interno, aunque también se realizan exportaciones en pequeña escala. En el año 2023, Ecuador exportó $3,800 USD en fresas frescas, posicionándose como el exportador número 96 de 118 países en esta categoría. A nivel nacional, las fresas frescas ocuparon el puesto 2109 entre los productos más exportados (de un total de 2,937) [80].

Los principales destinos de estas exportaciones fueron Estados Unidos con $3,720 USD, seguido por Luxemburgo ($58 USD) y España ($26 USD). Cabe destacar que el mercado de exportación con el crecimiento más acelerado entre 2022 y 2023 fue Estados Unidos, con un aumento de $2,040 USD en las compras de fresas ecuatorianas durante ese periodo [80].

Aunque la magnitud de las exportaciones aún es modesta, estos datos reflejan un potencial de crecimiento, particularmente si se fortalecen las cadenas de producción, certificación fitosanitaria y acceso a mercados internacionales [80].

3.2 Comprensión de los datos: Análisis de las características de calidad de la fresa

En esta etapa se realiza la comprensión de los datos, enfocándose en el análisis de las características visuales que determinan la calidad de la fresa. Se identificaron atributos clave como el color, forma y estado de la superficie, siendo el color rojo el principal indicador de madurez y calidad. Estos datos fueron recolectados mediante imágenes capturadas con la cámara de un smartphone y etiquetados con la herramienta Roboflow. El análisis previo de estas características permitió establecer una base sólida para entrenar el modelo de inteligencia artificial y garantizar una clasificación precisa según los niveles "Extra", "Primera", "Segunda" o "Desecho".

Las fresas en Ecuador tienen una importancia significativa tanto en términos económicos como sociales. Este cultivo se ha destacado por su creciente demanda en el mercado local e internacional debido a su alto valor nutricional, versatilidad en la cocina y usos industriales. Las fresas ecuatorianas se caracterizan por su sabor, frescura y calidad, lo que ha permitido que el país se convierta en un exportador relevante de esta fruta, especialmente hacia mercados como Estados Unidos, Europa y países de América Latina.

En el ámbito local, el cultivo de fresas representa una fuente significativa de empleo en las zonas rurales, beneficiando especialmente a pequeños y medianos productores. A continuación, se presenta la Tabla 27, que detalla las calidades de la fresa y sus características, basándose en lo investigado en la sección 1.3.4 Clasificación de fresas en grados de calidad del Capítulo I. Esta tabla muestra los resultados obtenidos tras analizar la clasificación de fresas según la norma NMX-FF-062-SCFI-2002, así como la clasificación del Instituto Tecnológico Superior de Coalcomán y del CENICAFE.

Tabla 27. Diferentes calidades de la fresa [26] [27] [28]

Categoría	Características	Imagen
Extra	Tamaño: diámetro mayor a 3,2 cm o igual Color: color hacia la zona de cáliz, el rojo es intenso y cubre toda la fruta, el fruto tiene más del 75% de superficie de color rojo	
Primera	Tamaño: diámetros de 3,1 a 2,6 cm Color: color rojo cubre la zona media del fruto y la zona del cáliz presenta intensidades rozadas, entre el 60 % y 75 % de la superficie del fruto es roja.	
Segunda	Tamaño: diámetro de 2,0 a 2,6 cm Color: entre el 55 % y 60 % de la superficie es roja.	
Desecho	Tamaño: menor a 1,9 cm Color: menos del 55 % de la superficie es roja.	

En el entrenamiento del modelo de inteligencia artificial se decidió utilizar únicamente el color como parámetro principal de clasificación, descartando el tamaño del fruto. Esta decisión se basó en que el tamaño de la fresa puede variar visualmente dependiendo del ángulo en que se capture la imagen, generando inconsistencias en la medición durante el proceso automatizado. Las diferencias en perspectiva, distancia a la cámara y posición de la fruta pueden afectar la precisión del tamaño percibido, lo cual podría confundir al modelo. En cambio, el color, especialmente en términos de intensidad y porcentaje de cobertura en la superficie del fruto, es una característica más estable, representativa y confiable para determinar la calidad. Por esta razón, se priorizó el análisis de color como criterio principal para asegurar un sistema de clasificación automático más preciso y robusto.

3.3 Preparación de los datos: Etiquetado de las fresas según tipos

En esta etapa, se realizó la preparación de los datos, paso esencial para el correcto entrenamiento del modelo de inteligencia artificial. Consistió en etiquetar las imágenes

de fresas según su nivel de calidad: “Extra”, “Primera”, “Segunda” o “Desecho”. Para ello, utilizando un software de etiquetado de imágenes que permitió marcar de forma precisa las regiones correspondientes a cada fresa y asignarles su respectiva categoría. Este proceso aseguró una base de datos bien estructurada, lo que facilita que el modelo aprenda a diferenciar visualmente las características propias de cada tipo de fresa, mejorando así su capacidad de clasificación.

3.3.1 Selección de cámara para la adquisición de imágenes

Para la selección del dispositivo de captura de imágenes, se realizó un análisis comparativo de diferentes cámaras disponibles en el mercado, considerando factores críticos como resolución, velocidad de captura (FPS), tipo de conexión y costo. En base a la comparación que realizada en la Tabla 18, la cámara que se selecciona para el sistema de clasificación automática de fresas es la cámara trasera del Samsung A10 ya que, permite obtener imágenes de alta definición, y su velocidad de captura de 30 FPS, lo que asegura un procesamiento eficiente en tiempo real. Además, la conexión mediante USB-C facilita la integración con sistemas de visión artificial, y su costo competitivo la convierte en una opción accesible para proyectos de clasificación automática de fresas.

3.3.2 Adquisición de imágenes para el Dataset

Para la elaboración del conjunto de datos, se utilizó un dispositivo móvil Samsung A10 con cámara principal de 13 megapíxeles, capturando un total de 935 imágenes de fresas. Estas imágenes fueron clasificadas visualmente según su color y calidad, organizándolas en carpetas correspondientes a cuatro categorías: Extra (307 imágenes), Primera (242), Segunda (151) y Desecho (235), como se muestra en la Figura 9. Posteriormente, las imágenes se distribuyeron en conjuntos de entrenamiento, validación y prueba, procurando mantener un equilibrio representativo de todas las categorías en cada uno, con el fin de garantizar un entrenamiento robusto y preciso del modelo de clasificación.

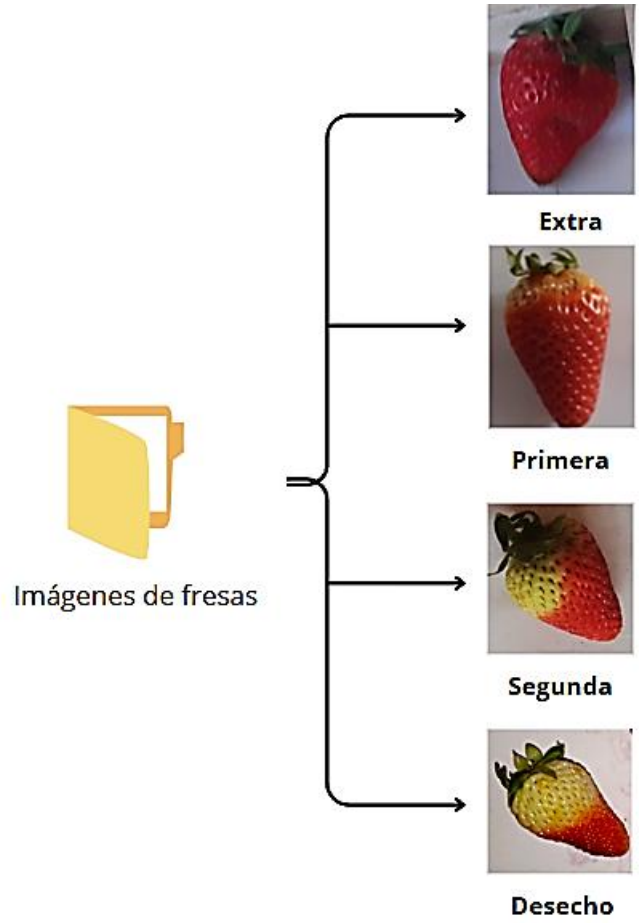


Figura 9. Estructura de carpetas del dataset de fresas

3.3.3 Selección de software de etiquetado de imágenes

Se realizó una detallada comparación de las características de varios softwares de etiquetado de imágenes utilizados en visión artificial que se presentan en la Tabla 19, donde se evaluó aspectos como la facilidad de uso, compatibilidad con formatos de salida, herramientas avanzadas de anotación y posibilidades de colaboración. Como resultado, se seleccionó Roboflow para el etiquetado de imágenes en este proyecto. Esta decisión se fundamentó en su interfaz intuitiva, su compatibilidad con múltiples formatos de exportación (como YOLO, COCO y VOC) y sus herramientas integradas de preprocesamiento y aumento de datos. Además, su entorno en la nube y su integración con lenguajes como Python permiten automatizar el flujo de trabajo y facilitar la preparación de datos para el entrenamiento de modelos de inteligencia artificial.

3.3.4 Etiquetado utilizando la herramienta online Roboflow

El etiquetado de imágenes se llevó a cabo utilizando la herramienta en línea Roboflow, que con la que se asignó cuatro etiquetas de calidad a las imágenes de fresas: “Extra” (más del 75 % de superficie roja), “Primera” (entre 60 %–75 % de superficie roja), “Segunda” (Entre 55 %–60 % de superficie roja) y “Desecho” (menos del 55 % de superficie roja). Este proceso consiste en cargar las imágenes de fresas en la interfaz online de Roboflow y dibujar manualmente contornos alrededor de cada fresa para clasificarlas según su nivel de madurez, tal como se ilustra en la Figura 10. Cada etiqueta creada se asigna a una de las cuatro categorías de calidad, lo que facilita la posterior identificación automatizada.

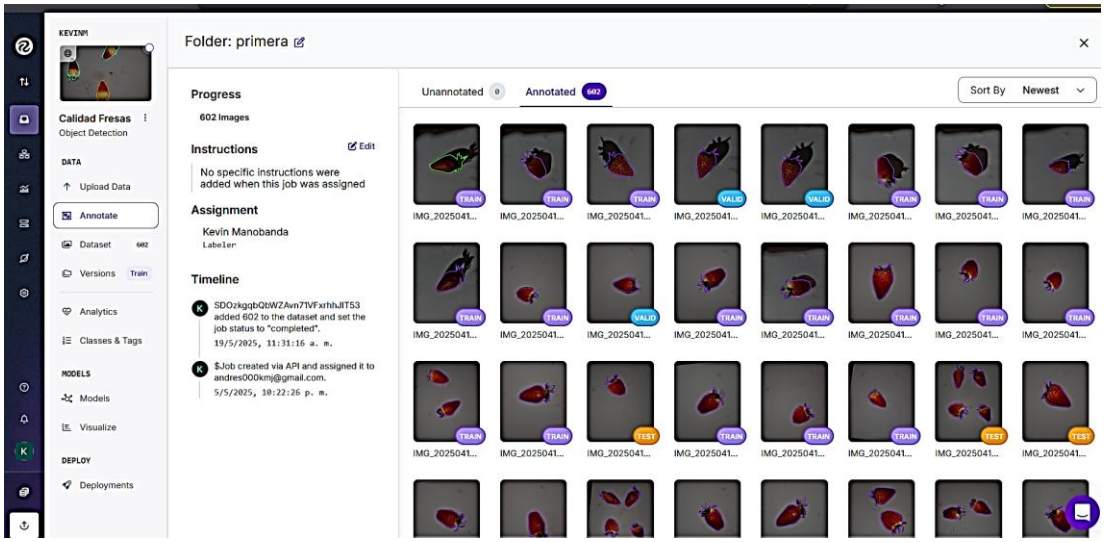


Figura 10. Etiquetado de imágenes en Roboflow

Una vez completado el etiquetado, con las herramientas que ofrece Roboflow incrementamos la base de datos para el entrenamiento utilizando “versions” y se creó un dataset aumentado con la opción “Augmentation” seleccionando la característica de rotación en -15° y +15°, lo que resulto en una base de datos con 1429 imágenes, donde 1248 son para el entrenamiento, 121 son para validación y 60 son para prueba como se puede observar en la Figura 11.

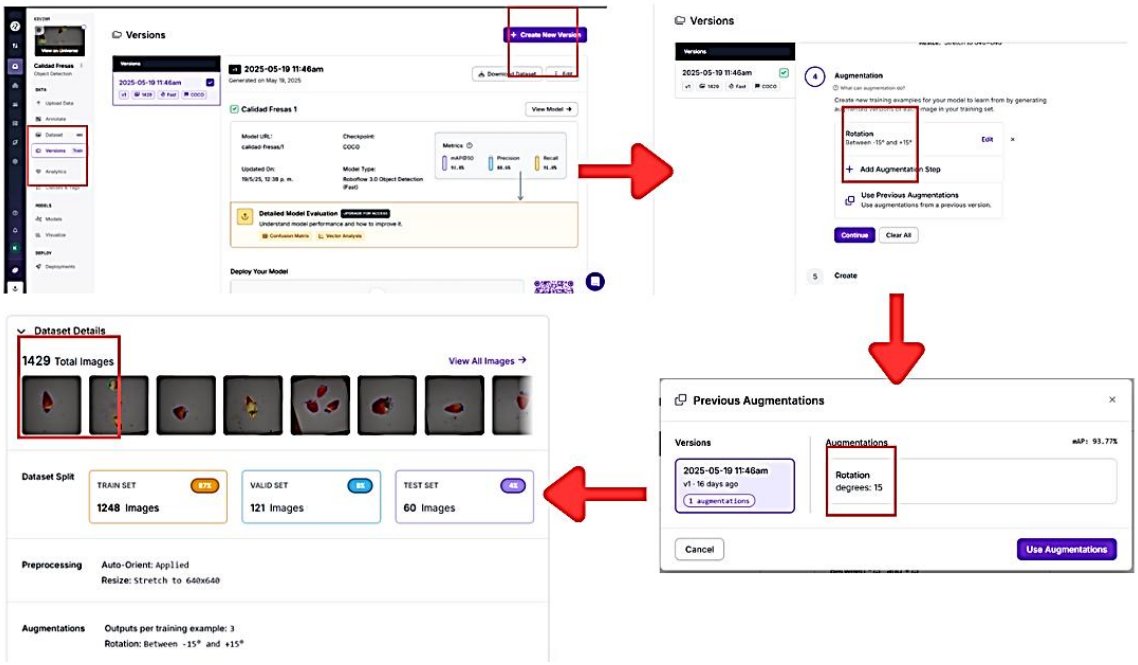


Figura 11. Proceso para incrementar los datos del Dataset

Para utilizar la base de datos creada, se empleó la opción “Download Dataset” seleccionando el formato compatible con YOLOv11. Con el objetivo de optimizar el almacenamiento, se utilizó la opción “Show download code”, lo que permitió acceder directamente a la base de datos desde Roboflow mediante código, sin necesidad de descargar los archivos localmente. Este procedimiento facilitó el entrenamiento del modelo en Google Colab. El proceso se resume en la Figura 12.

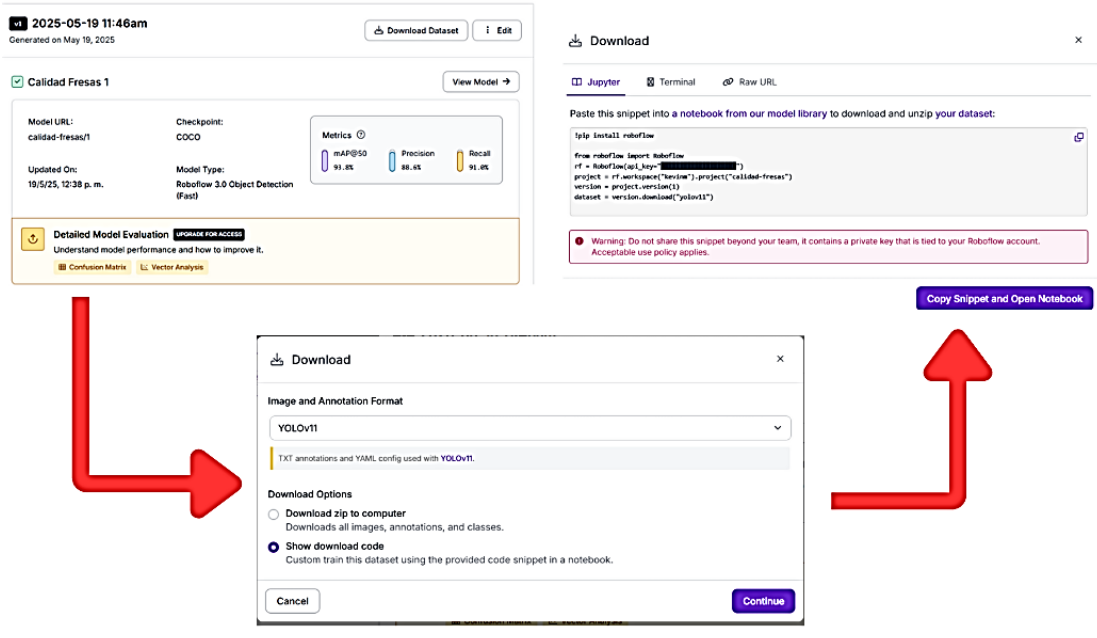


Figura 12. Procedimiento para utilizar el Dataset creado en Roboflow

Este código, ejecutado en una plataforma como Google Colab, permite acceder al dataset previamente creado en Roboflow. Al ejecutarlo, se genera automáticamente una carpeta denominada "Calidad-Fresas-1", que contiene todos los archivos necesarios para el entrenamiento del modelo, incluyendo subcarpetas con imágenes y etiquetas organizadas para las fases de entrenamiento, validación y prueba. Además, se incluye un archivo clave llamado data.yaml, el cual especifica las rutas de acceso a dichos archivos, así como el número total de clases y los nombres asignados a cada etiqueta como se puede ver en la Figura 13. Esta configuración facilita una integración ordenada y eficiente con el modelo YOLO.



Figura 13. Contenido del archivo data.yaml

3.4 Modelado: Entrenamiento del modelo de Inteligencia Artificial

En esta etapa, se lleva a cabo el modelado del sistema de clasificación de fresas mediante la configuración de un entorno de entrenamiento basado en la nube, el cual proporciona recursos computacionales adecuados para procesar grandes volúmenes de datos e implementar modelos de inteligencia artificial. Para el entrenamiento, se utiliza una de las versiones más adecuadas del modelo YOLO, reconocida por su capacidad de realizar detección de objetos en tiempo real con alta precisión. La red neuronal se entrena utilizando el dataset previamente etiquetado, optimizando sus parámetros para lograr una detección y clasificación eficiente de las fresas según su calidad.

3.4.1 Selección de algoritmo de Inteligencia Artificial

Al analizar las técnicas de visión artificial empleadas en la clasificación de frutas, como se muestra en la Tabla 20, se observa una variedad de enfoques y resultados. Por ejemplo, el estudio de León León et al. utilizó YOLOv5x en su modelo, logrando una eficiencia general del 97.14% en la clasificación de fresas. Por otro lado, el sistema desarrollado por Patricia Nataly Constante Procel y Andrés Marcelo Gordon Garces, que utilizó OpenCV, obtuvo un reconocimiento del 87.5%, lo que indica que aunque es una técnica válida, podría no ser tan robusta como los modelos basados en YOLO.

En el caso del estudio de Juan José Camargo Robles, que implementó YOLOv8, se obtuvo una precisión del 97.08%, lo que demuestra la eficacia de YOLOv8 en comparación con otros métodos, como las redes neuronales convolucionales (CNN) utilizadas por García Pérez y Martínez López, que lograron un 94.8%. Ademas, el sistema basado en Raspberry Pi con YOLOv4 Tiny desarrollado por Rodríguez Gómez y Hernández Ruiz alcanzó una precisión del 92.5%, pero su rendimiento es inferior al de YOLOv8, especialmente en términos de velocidad y eficiencia en dispositivos de bajo rendimiento. Finalmente, Liao Yue, Li Lerong, Xiao Huiqiang, Feijiano Xu, Shan Bochen y Yin Hua con el algoritmo de detección de cítricos basado en YOLOv11y YOLO-MECD alcanzo una precisión (P) del 84,4% con una tasa de y una precisión media media (mAP) elevada del 81,6% con mejoras mAP de 3.8%, 3.2% y 5.5% en comparación con YOLOv8s, YOLOv9s y YOLOv10s.

La elección de YOLOv11 se justifica por su superioridad en precisión, velocidad y eficiencia de recursos. Comparado con versiones anteriores, YOLOv11 ofrece un modelo más optimizado. Dentro de Yolov11 se encuentran más versiones que se presentan en la Tabla 23, después de un análisis se determinó que la versión más adecuada para la elaboración del proyecto es la versión YOLOv11l principalmente porque es ligera, rápida y tienen un bajo consumo de recursos a diferencia del YOLOv11x. Además, tiene mejoras en la generalización de modelos, lo que facilita su uso en una amplia gama de tareas, desde clasificación de imágenes hasta segmentación de objetos, adaptándose a proyectos complejos y de gran escala.

3.4.2 Selección de herramienta para ejecutar código en nube

En base a la comparación realizada en la Tabla 11, se ha elegido Google Colab como la herramienta para ejecutar código en la nube en este proyecto debido a sus numerosas ventajas que facilitan el desarrollo y la implementación de modelos de aprendizaje automático. Ofrece acceso a recursos computacionales, incluyendo GPUs de alto rendimiento, lo que permite un entrenamiento más eficiente de modelos complejos. Además, su integración con Google Drive y su interfaz basada en Jupyter Notebook facilitan la colaboración en tiempo real y la gestión de datos, lo que resulta ideal para equipos que trabajan en proyectos de ciencia de datos. En la Tabla 28, se presentan las características técnicas de Google Colab.

Tabla 28. Característica de la máquina usada para entrenamiento [37]

Característica	**Descripción**
CPU	Hasta 2 núcleos de CPU (Intel Xeon)
GPU	Acceso a GPUs como NVIDIA Tesla K80, T4, P4 o A100 (según disponibilidad)
TPU	Acceso a TPUs de Google
RAM	Hasta 12 GB de RAM
Almacenamiento	Temporal en /content y acceso a Google Drive
Tiempo de ejecución	Sesiones de hasta 12 horas
Entorno de ejecución	Python 3 preinstalado con bibliotecas como NumPy, TensorFlow, PyTorch, etc.
Acceso a recursos	Acceso a recursos de hardware mejorados
Soporte para entornos virtuales	Compatible con entornos virtuales (venv, conda)

3.4.3 Configuración del entrenamiento en Google Colab

Para empezar con el entrenamiento se importa las bibliotecas necesarias para el entrenamiento, para utilizar los recursos de Roboflow, para graficar y presentar los resultados como se muestra en la Figura 14.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import matplotlib.pyplot as plt
from PIL import Image

[ ] !pip install split-folders pillow ultralytics
    !pip install roboflow
```

Figura 14. Líneas código para importar las librerias en Google Colab

A continuación, se incluyen los comandos para importar la base de datos de Roboflow a Google Colab para utilizar la base de datos creada en el entrenamiento como se puede ver en la Figura 15.

```
[ ] from roboflow import Roboflow
    rf = Roboflow(api_key="w26pXrUbyM22c37pHOvR")
    project = rf.workspace("kevinm").project("calidad-fresas")
    version = project.version(1)
    dataset = version.download("yolov11")

    loading Roboflow workspace...
    loading Roboflow project...
    Downloading Dataset Version Zip in Calidad-Fresas-1 to yolov11:: 100%| | 35098/35098 [00:00<00:00, 44180.81it/s]

    Extracting Dataset Version Zip to Calidad-Fresas-1 in yolov11:: 100%| | 2870/2870 [00:00<00:00, 9073.35it/s]
```

Figura 15. Líneas de código para importar el Dataset de Roboflow

Se ejecutó el código presentado en la Figura 16 con el fin de verificar que el Dataset exportado desde Roboflow se haya importado correctamente. Adicionalmente, se eliminó la carpeta innecesaria denominada “sample_data” para mantener el entorno de trabajo más organizado.

```
[ ]  # Specify the path to the file
     file_path = '/content/Calidad-Fresas-1/data.yaml'

     # Check if the file exists
     if os.path.exists(file_path):
         print(f"The file at {file_path} exists.")
     else:
         print(f"The file at {file_path} does not exist.")

     The file at /content/Calidad-Fresas-1/data.yaml exists.

[ ]  # Get the current working directory
     cwd = os.getcwd()
     print(f"Current working directory: {cwd}")

     # List all files and directories in the current directory
     files = os.listdir(cwd)
     print(f"Files and directories in '{cwd}': {files}")

     Current working directory: /content
     Files and directories in '/content': ['.config', 'Calidad-Fresas-1', 'sample_data']

[ ]  dir_path = cwd + "/sample_data"
     for files in os.listdir(dir_path):
       os.remove(dir_path + "/" + files)
     os.rmdir(dir_path)
```

Figura 16. Líneas de código para verificar la exportación del Dataset

Por último se configura los parámetros del entrenamiento como se muestra en la Figura 17. Se crea una instancia del modelo YOLO al cargar un modelo pre entrenado denominado 'yolov11l.pt'. La línea model = YOLO('yolov11l.pt') carga los parámetros del modelo, permitiendo su uso en tareas como entrenamiento, evaluación o predicción.

En la línea de código siguiente, se entrena el modelo utilizando la función train(), que ajusta el modelo a los datos específicos del usuario. Los parámetros utilizados incluyen:

- data='/content/Calidad-Fresas-1/data.yaml', que indica la ruta al archivo de configuración que describe la ubicación de las imágenes de entrenamiento y validación, así como las clases a detectar.

- epochs=50 especifica que el modelo verá el conjunto de datos completo 50 veces durante el entrenamiento para mejorar su precisión.

- imgsz=640 define el tamaño de las imágenes a 640x640 píxeles, asegurando uniformidad en las entradas.

- batch=16 establece que el modelo procesará 16 imágenes a la vez, lo que influye en la velocidad y el uso de memoria durante el entrenamiento.

- Finalmente, name='entrenamiento_fresas' asigna un nombre al experimento, permitiendo organizar y almacenar los resultados, como modelos y gráficos, de manera eficiente.

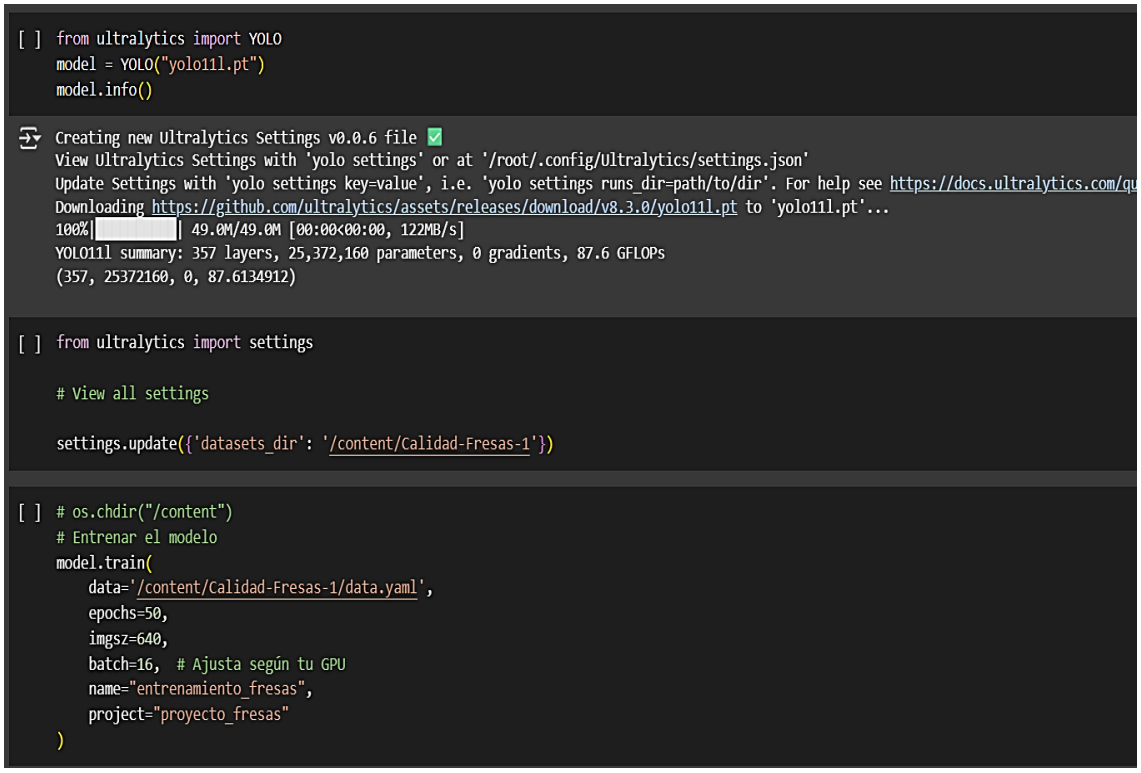


Figura 17. Configuración del entrenamiento con YOPLOv11l (50 épocas)

Este proceso se llevó a cabo en tres ocasiones, modificando principalmente el parámetro de las épocas de entrenamiento a 75 y 100 épocas, como se muestra en la Figura 18.

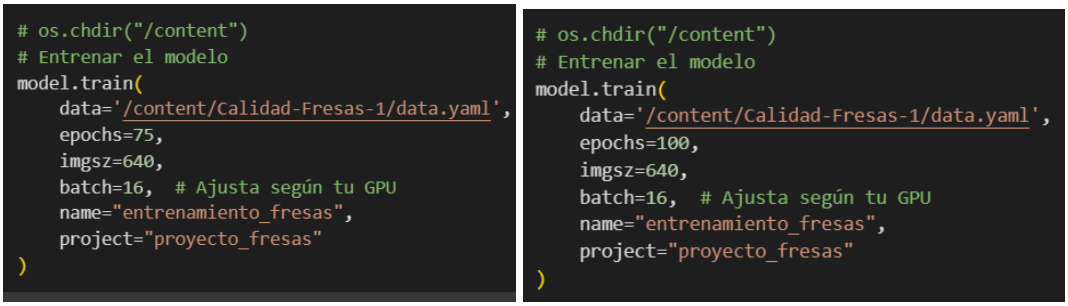


Figura 18. Configuración del entrenamiento con YOPLOv11l (75 y 100 épocas)

3.4.4 Visualización de resultados del entrenamiento con Google Colab

a. Entrenamiento con 50 épocas

Durante el entrenamiento del modelo con 50 épocas, el tiempo promedio por época es de aproximadamente 57 segundos, como se ilustra en la Figura 19.

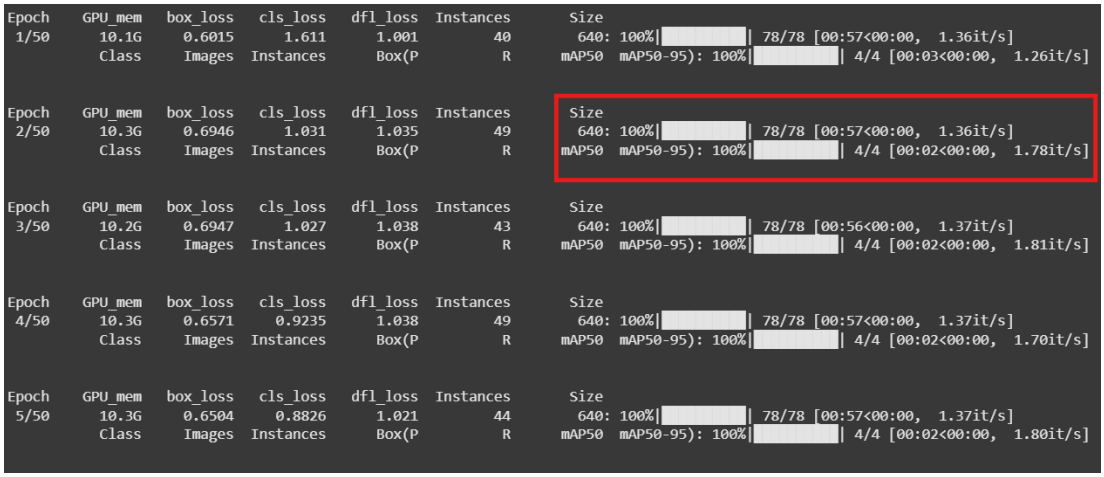


Figura 19. Inicio de la ejecución del entrenamiento con 50 epocas

Al finalizar el entrenamiento, este proceso específico tomó 0.86 horas en el entorno de Google Colab. Los archivos generados como resultado del entrenamiento se almacenan en Google Drive, incluyendo los archivos best.pt y last.pt, como se ilustra en la Figura 20.

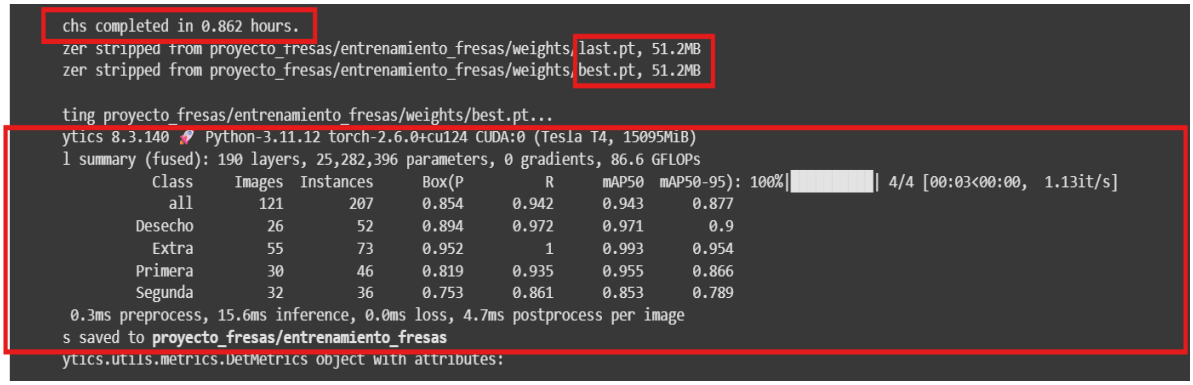


Figura 20. Finalización de la ejecución del entrenamiento con 50 épocas

b. Entrenamiento con 75 épocas

Durante el entrenamiento del modelo con 75 épocas, el tiempo promedio por época es de aproximadamente 59 segundos, como se ilustra en la Figura 21.

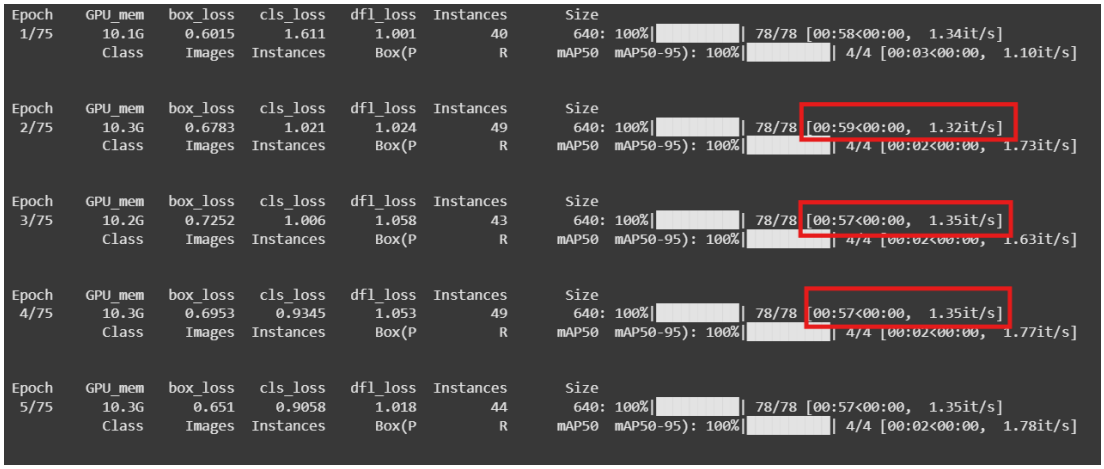


Figura 21. Inicio de la ejecución del entrenamiento con 75 épocas

Al finalizar el entrenamiento, este proceso específico tomó 1.3 horas en el entorno de Google Colab. Los archivos generados como resultado del entrenamiento se almacenan en Google Drive, incluyendo los archivos best.pt y last.pt, como se ilustra en la Figura 22.

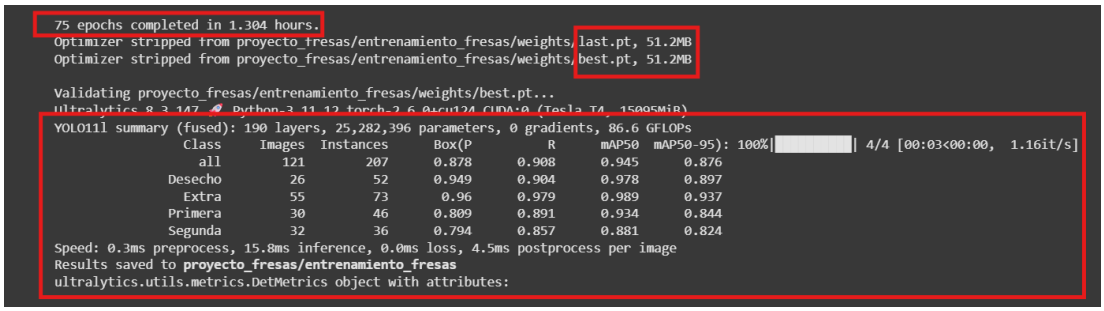


Figura 22. Finalización de la ejecución del entrenamiento con 75 épocas

c. Entrenamiento con 1000 épocas

Durante el entrenamiento del modelo con 100 épocas, el tiempo promedio por época es de aproximadamente 59 segundos, como se ilustra en la Figura 23.

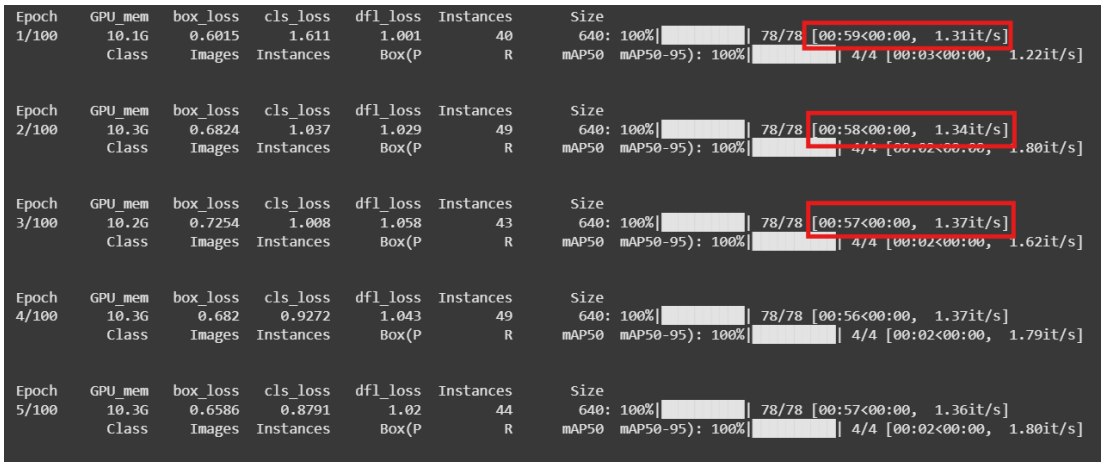


Figura 23. Inicio de la ejecución del entrenamiento con 100 épocas

Al finalizar el entrenamiento, este proceso específico tomó 1.7 horas en el entorno de Google Colab. Los archivos generados como resultado del entrenamiento se almacenan en Google Drive, incluyendo los archivos best.pt y last.pt, como se ilustra en la Figura 24.

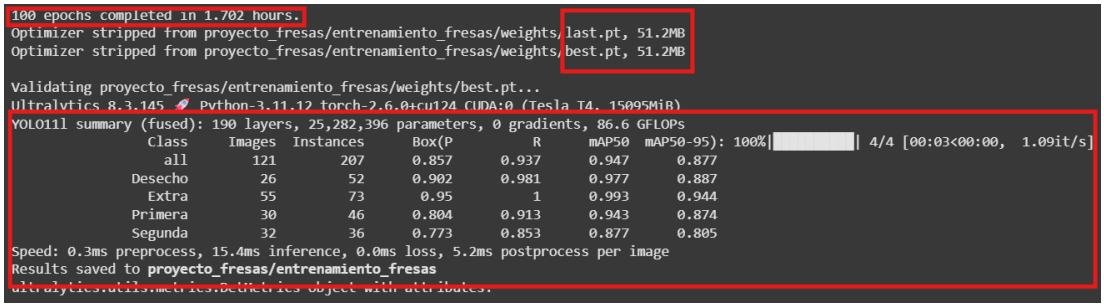


Figura 24. Finalización de la ejecución del entrenamiento con 100 épocas

3.5 Evaluación: Resultados del entrenamiento del modelo de inteligencia artificial

En la fase de evaluación, se analizan los resultados obtenidos tras el entrenamiento del modelo de inteligencia artificial, con el objetivo de medir su precisión, confiabilidad y capacidad para clasificar correctamente las fresas según su calidad. Se utilizan métricas como la exactitud, la pérdida (loss), y la matriz de confusión para verificar el rendimiento del modelo entrenado con YOLOv11.

3.5.1 Entrenamiento 1 con 50 etapas

a. Métricas de rendimiento

En la Figura 25, se muestran las métricas de rendimiento donde en la gráfica de "train/box_loss", "train/cls_loss" y "train/dfl_loss" se observa una disminución consistente de las pérdidas de entrenamiento a lo largo de las 50 épocas. Esto indica que el modelo mejoró progresivamente en la predicción de posiciones, tamaños y clasificación de los objetos durante el entrenamiento.

Las métricas de "metrics/precision(B)" y "metrics/recall(B)" muestran un comportamiento similar, con un aumento gradual de la precisión de 0.760700 a 0.870210 y del recall de 0.849658 a 0.946470. Esto sugiere que el modelo redujo la tasa de falsos positivos y falsos negativos, mejorando su capacidad de identificar correctamente los objetos.

En cuanto a las métricas de validación, las gráficas de "val/box_loss", "val/cls_loss" y "val/dfl_loss" también muestran una disminución constante, lo que indica una mejora en el rendimiento del modelo en el conjunto de validación.

Las métricas "metrics/mAP50(B)" y "metrics/mAP50-95(B)" finalizan en valores altos, alrededor de 0.944040 y 0.877360 respectivamente. Esto demuestra que el modelo logró un buen desempeño general en la detección de objetos, tanto en el umbral de IoU del 50% como en el rango entre 50% y 95%.

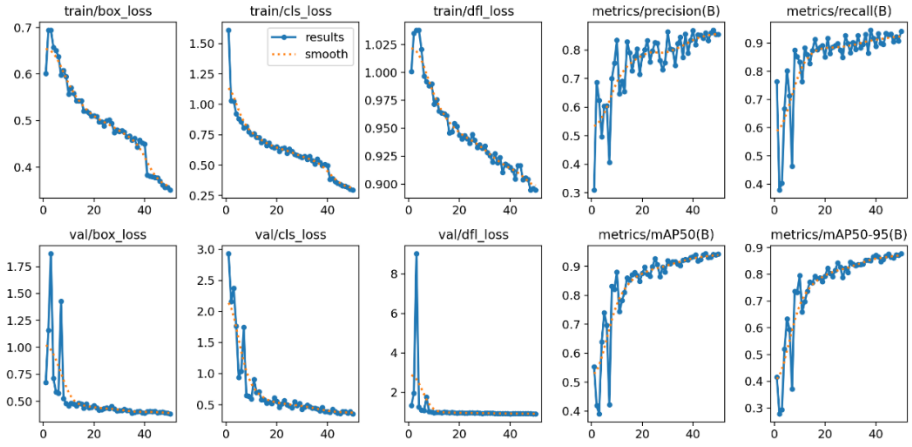


Figura 25. Métricas de rendimiento del modelo entrenado 1

b. Matiz de confusión

Basándose en la matriz de confusión que se presenta en la Figura 26, muestra el desempeño de un modelo de clasificación binaria que distingue entre cinco clases: "Extra", "Primera", "Segunda", "Desecho" y "background" (fondo).

Los valores en la matriz están normalizados, lo que significa que representan proporciones o probabilidades en lugar de recuentos absolutos.

En la celda de “Desecho”, se observa un valor de 0.94, lo que indica que el modelo clasificó correctamente el 94% de las instancias de la clase "Desecho" (verdaderos positivos).

En la celda de “Extra”, el valor de 0.99 significa que el modelo clasificó correctamente el 99% de las instancias de la clase "Extra" (verdaderos positivos).

En la celda de “Primera”, el valor de 0.89 muestra que el modelo clasificó correctamente el 89% de las instancias de la clase "Primera" (verdaderos positivos).

En la celda de “Segunda”, el valor de 0.78 muestra que el modelo clasificó correctamente el 78% de las instancias de la clase "Segunda" (verdaderos positivos).

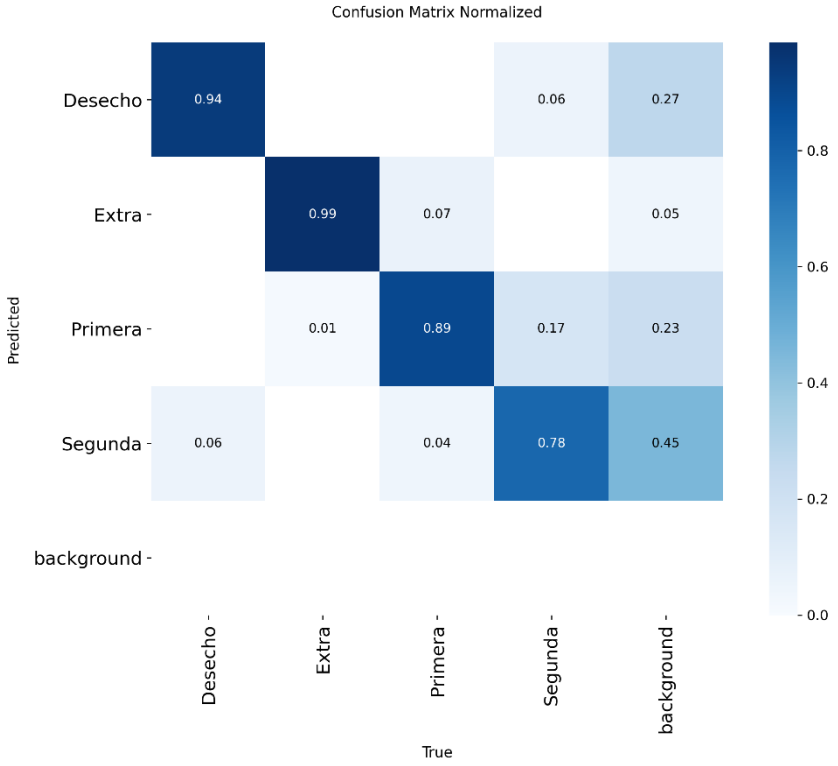


Figura 26. Matriz de confusión del modelo entrenado 1

En conclusión, el modelo tiene un buen rendimiento, como lo evidencia la correcta clasificación en la diagonal principal de la matriz. Esto indica una alta precisión en la predicción de las clases Extra, Primera, Segunda y Desecho.

Con los datos obtenidos de la matriz de confusión, se determina la precisión del sistema. En la Tabla 29, se presentan los valores totales de verdaderos positivos, falsos positivos y falsos negativos para las clases Extra, Primera, Segunda y Desecho. Los verdaderos positivos se producen cuando el modelo identifica correctamente una instancia dentro de su clase correspondiente. En cambio, los falsos positivos surgen cuando el sistema clasifica erróneamente una instancia como perteneciente a una clase cuando no lo es, lo que puede deberse a similitudes visuales entre categorías, iluminación deficiente o superposición entre frutos. Por otro lado, los falsos negativos ocurren cuando el modelo no reconoce una instancia que sí pertenece a una clase, lo cual puede originarse por variaciones en la forma, color atípico, o por condiciones visuales que dificultan la detección.

En este sentido, el sistema presenta debilidades principalmente al diferenciar entre clases con características menos contrastadas, como “Primera” y “Segunda”, lo que afecta la precisión en estas categorías específicas.

Tabla 29. Datos de la matriz de confusión del modelo entrenado 1

Clases	Verdaderos Positivos (VP)	Falsos Positivos (FP)	Falsos Negativos (FN)
Desecho	49	3	8
Extra	72	1	4
Primera	41	5	12
Segunda	28	8	15
Total	190	17	39

La exactitud general del modelo entrenado se calcula mediante el promedio de los valores de verdaderos positivos y falsos positivos de cada clase, usando la Ecuación (1):

$$Precisión = \frac{Vp}{Vp\ +\ Fp} \tag{1}$$

Sustituyendo los valores:

$$Precisión\ = \frac{190}{190\ +\ 17}$$

$$Precisión\ = 0.91$$

Por lo tanto, la exactitud del modelo es **0.91 (91%)**, lo que indica que todas las instancias fueron clasificadas correctamente casi sin errores.

c. Precisión del entrenamiento

La Figura 27, se presenta la curva Precisión-Confianza donde se observa que las clases “Desecho” y “Extra” presentan una alta precisión a lo largo de diferentes niveles de confianza, lo cual indica un buen desempeño del modelo en estas categorías. Las clases “Primera” y “Segunda” muestran menor precisión, especialmente “Segunda”, que presenta mayor variabilidad e inestabilidad. A pesar de ello, la línea azul gruesa, que representa el promedio de todas las clases, alcanza una precisión de 1.00 a un nivel de confianza de 0.983, lo cual evidencia un rendimiento global aceptable del modelo.

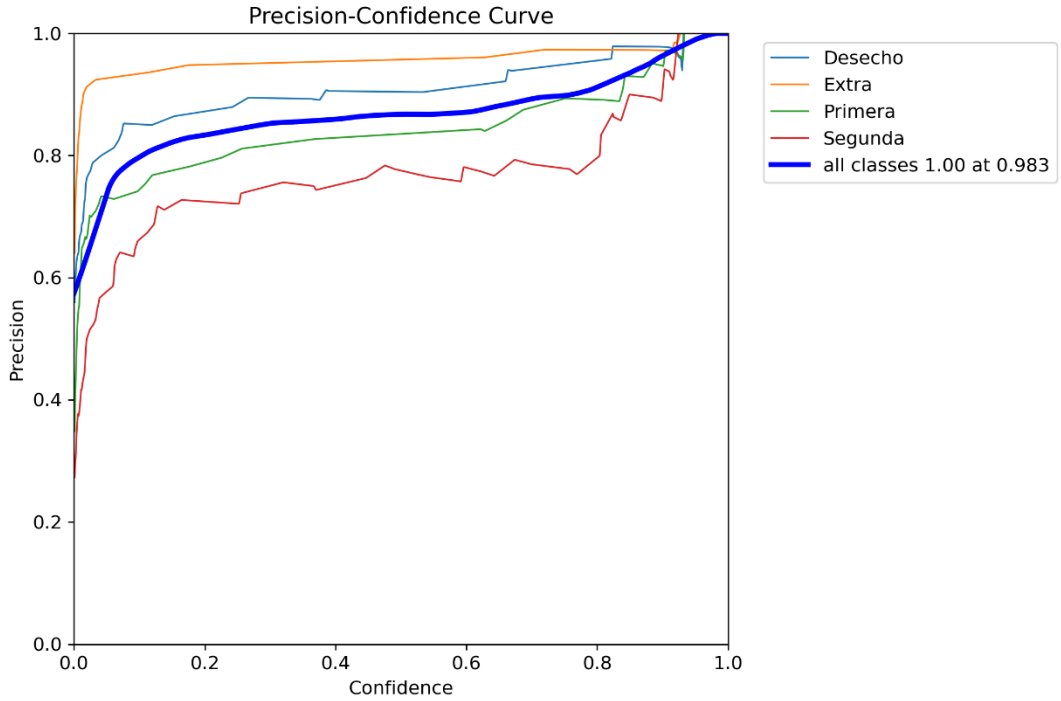


Figura 27. Precisión del modelo 1

3.5.2 Entrenamiento 2 con 75 etapas

a. Métricas de rendimiento

En la Figura 28, se muestran las métricas de rendimiento donde en la gráfica de "train/box_loss", "train/cls_loss" y "train/dfl_loss" se observa una disminución consistente de las pérdidas de entrenamiento a lo largo de las 75 épocas. Esto indica que el modelo mejoró progresivamente en la predicción de posiciones, tamaños y clasificación de los objetos durante el entrenamiento.

Las métricas de "metrics/precision(B)" y "metrics/recall(B)" muestran un comportamiento similar, con un aumento gradual de la precisión de 0.779257 a 0.903110 y del recall de 0.864740 a 0.958900. Esto sugiere que el modelo redujo la tasa de falsos positivos y falsos negativos, mejorando su capacidad de identificar correctamente los objetos.

En cuanto a las métricas de validación, las gráficas de "val/box_loss", "val/cls_loss" y "val/dfl_loss" también muestran una disminución constante, lo que indica una mejora en el rendimiento del modelo en el conjunto de validación.

Las métricas "metrics/mAP50(B)" y "metrics/mAP50-95(B)" finalizan en valores altos, alrededor de 0.945230 y 0.875690 respectivamente. Esto demuestra que el modelo logró un buen desempeño general en la detección de objetos, tanto en el umbral de IoU del 50% como en el rango entre 50% y 95%.

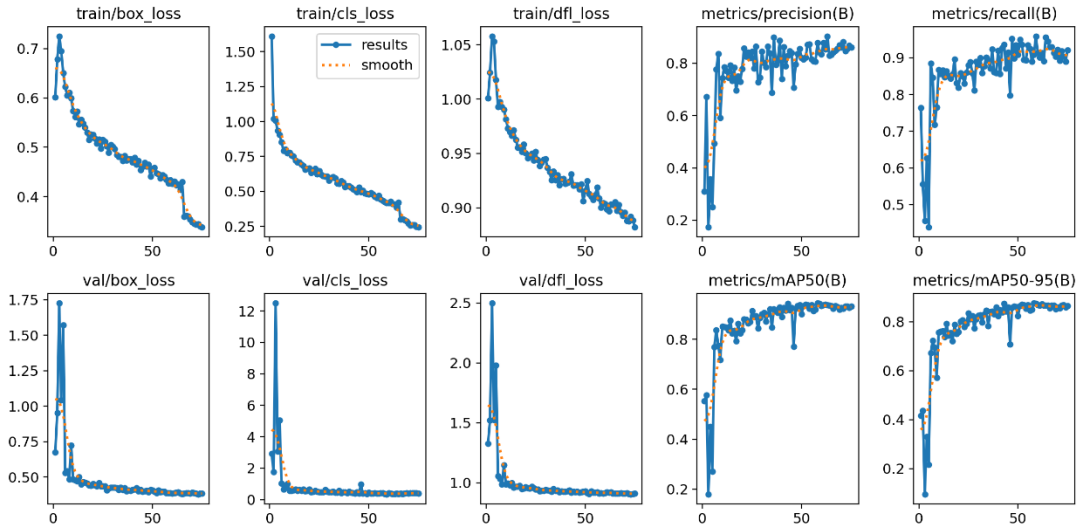


Figura 28. Métricas de rendimiento del modelo entrenado 2

b. Matiz de confusión

Basándose en la matriz de confusión que se presenta en la Figura 29, muestra el desempeño de un modelo de clasificación binaria que distingue entre cinco clases: "Extra", "Primera", "Segunda", "Desecho" y "background" (fondo).

Los valores en la matriz están normalizados, lo que significa que representan proporciones o probabilidades en lugar de recuentos absolutos.

En la celda de “Desecho”, se observa un valor de 0.83, lo que indica que el modelo clasificó correctamente el 83% de las instancias de la clase "Desecho" (verdaderos positivos).

En la celda de “Extra”, el valor de 0.96 significa que el modelo clasificó correctamente el 96% de las instancias de la clase "Extra" (verdaderos positivos).

En la celda de “Primera”, el valor de 0.87 muestra que el modelo clasificó correctamente el 87% de las instancias de la clase "Primera" (verdaderos positivos).

En la celda de “Segunda”, el valor de 0.75 muestra que el modelo clasificó correctamente el 75% de las instancias de la clase "Segunda" (verdaderos positivos).

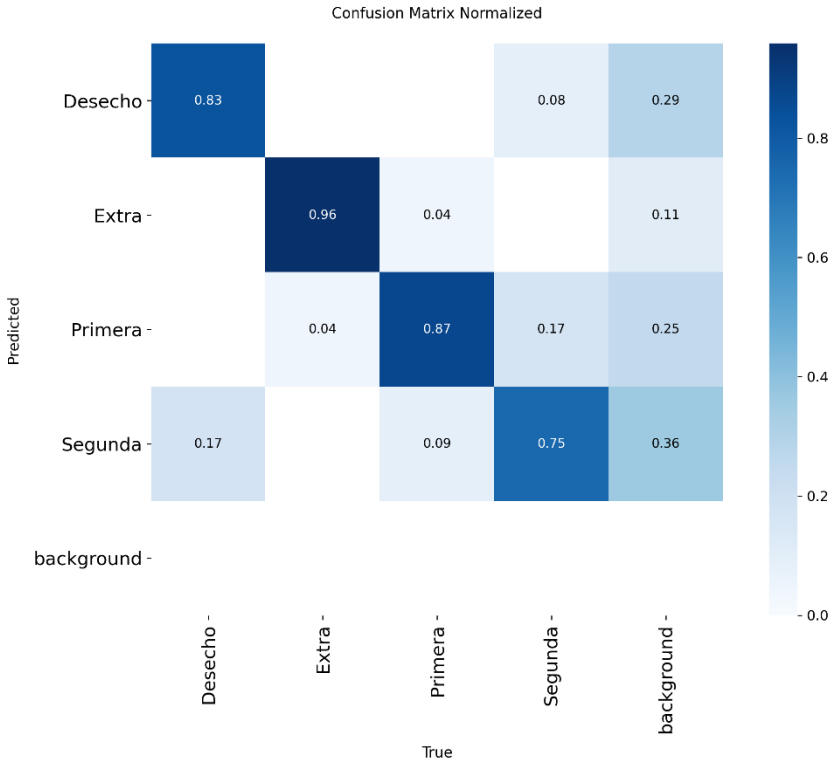


Figura 29. Matriz de confusión del modelo entrenado 2

En conclusión, la precisión de este modelo sigue siendo aceptable; sin embargo, al compararlo con el modelo entrenado con 50 épocas, se observa una menor precisión en la clasificación de las clases.

Con los datos obtenidos de la matriz de confusión, se determina la precisión del sistema. En la Tabla 30, se presentan los valores totales de verdaderos positivos, falsos positivos y falsos negativos para las clases Extra, Primera, Segunda y Desecho. En este segundo entrenamiento, los verdaderos positivos reflejan un correcto aprendizaje del modelo, mientras que los falsos positivos y negativos se deben principalmente a similitudes visuales entre clases, condiciones de iluminación y superposición de objetos. La clase “Segunda” presenta mayores dificultades de identificación, lo que evidencia una menor capacidad del modelo para generalizar en categorías con diferencias sutiles. Aun así, el rendimiento global se mantiene aceptable.

Tabla 30. Datos de la matriz de confusión del modelo entrenado 2

Clases	Verdaderos Positivos (VP)	Falsos Positivos (FP)	Falsos Negativos (FN)
Desecho	43	9	11
Extra	70	3	5
Primera	40	6	16
Segunda	27	9	23
Total	180	27	55

La exactitud general del modelo entrenado se calcula mediante el promedio de los valores de verdaderos positivos y falsos positivos de cada clase, usando la fórmula:

$$Precisión \ = \frac{Vp}{Vp \ + \ Fp}$$

Sustituyendo los valores:

$$Precisión \ = \frac{180}{180 \ + \ 27}$$

$$Precisión \ = 0.86$$

Por lo tanto, la exactitud del modelo es **0.86 (86%)**, lo que indica que todas las instancias fueron clasificadas correctamente casi sin errores.

c. Precisión del entrenamiento

La Figura 30 muestra la curva Precisión-Confianza para cada clase del modelo. Se observa que las clases “Desecho” y “Extra” mantienen una alta precisión (>0.9) incluso con niveles bajos de confianza, lo que indica una clasificación sólida. La clase “Primera” presenta una precisión intermedia y más variable, mientras que “Segunda” muestra menor precisión y mayor inestabilidad, lo que sugiere dificultad para el modelo en identificarla correctamente. La línea azul gruesa representa el promedio de todas las clases, alcanzando una precisión máxima de 1.00 a un nivel de confianza de 0.981, lo cual es positivo. En conjunto, el modelo es más confiable para ciertas clases que para otras.

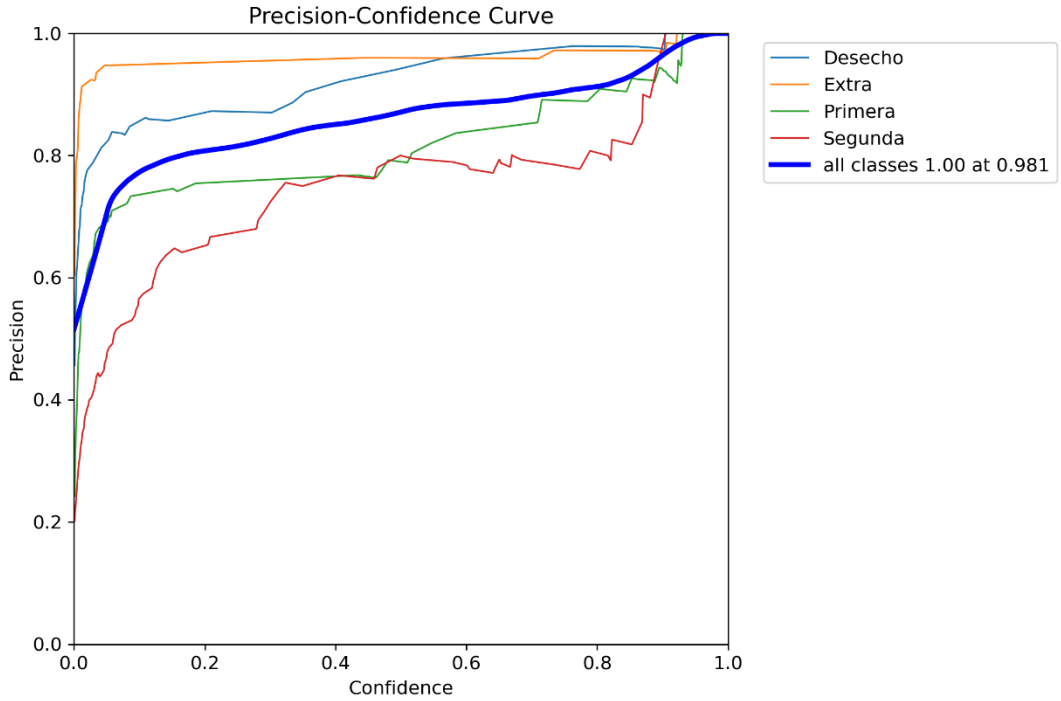


Figura 30. Precisión del modelo 2

3.5.3 Entrenamiento 3 con 100 etapas

a. Métricas de rendimiento

En la Figura 31, se muestran las métricas de rendimiento donde en la gráfica de "train/box_loss", "train/cls_loss" y "train/dfl_loss" se observa una disminución consistente de las pérdidas de entrenamiento a lo largo de las 75 épocas. Esto indica que el modelo mejoró progresivamente en la predicción de posiciones, tamaños y clasificación de los objetos durante el entrenamiento.

Las métricas de "metrics/precision(B)" y "metrics/recall(B)" muestran un comportamiento similar, con un aumento gradual de la precisión de 0.818214 a 0.919880 y del recall de 0.858627 a 0.976090. Esto sugiere que el modelo redujo la tasa de falsos positivos y falsos negativos, mejorando su capacidad de identificar correctamente los objetos.

En cuanto a las métricas de validación, las gráficas de "val/box_loss", "val/cls_loss" y "val/dfl_loss" también muestran una disminución constante, lo que indica una mejora en el rendimiento del modelo en el conjunto de validación.

Las métricas "metrics/mAP50(B)" y "metrics/mAP50-95(B)" finalizan en valores altos, alrededor de 0.947310 y 0.877280 respectivamente. Esto demuestra que el modelo logró un buen desempeño general en la detección de objetos, tanto en el umbral de IoU del 50% como en el rango entre 50% y 95%.

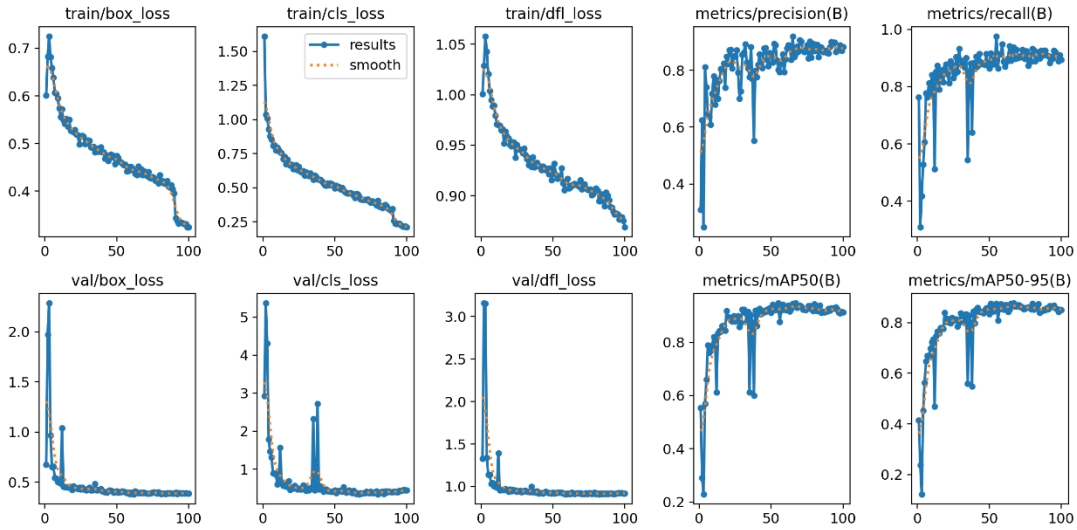


Figura 31. Métricas de rendimiento del modelo entrenado 3

b. Matiz de confusión

Basándose en la matriz de confusión que se presenta en la Figura 32, muestra el desempeño de un modelo de clasificación binaria que distingue entre cinco clases: "Extra", "Primera", "Segunda", "Desecho" y "background" (fondo).

Los valores en la matriz están normalizados, lo que significa que representan proporciones o probabilidades en lugar de recuentos absolutos.

En la celda de “Desecho”, se observa un valor de 0.90, lo que indica que el modelo clasificó correctamente el 90% de las instancias de la clase "Desecho" (verdaderos positivos).

En la celda de “Extra”, el valor de 0.95 significa que el modelo clasificó correctamente el 95% de las instancias de la clase "Extra" (verdaderos positivos).

En la celda de “Primera”, el valor de 0.96 muestra que el modelo clasificó correctamente el 96% de las instancias de la clase "Primera" (verdaderos positivos).

En la celda de “Segunda”, el valor de 0.78 muestra que el modelo clasificó correctamente el 78% de las instancias de la clase "Segunda" (verdaderos positivos).

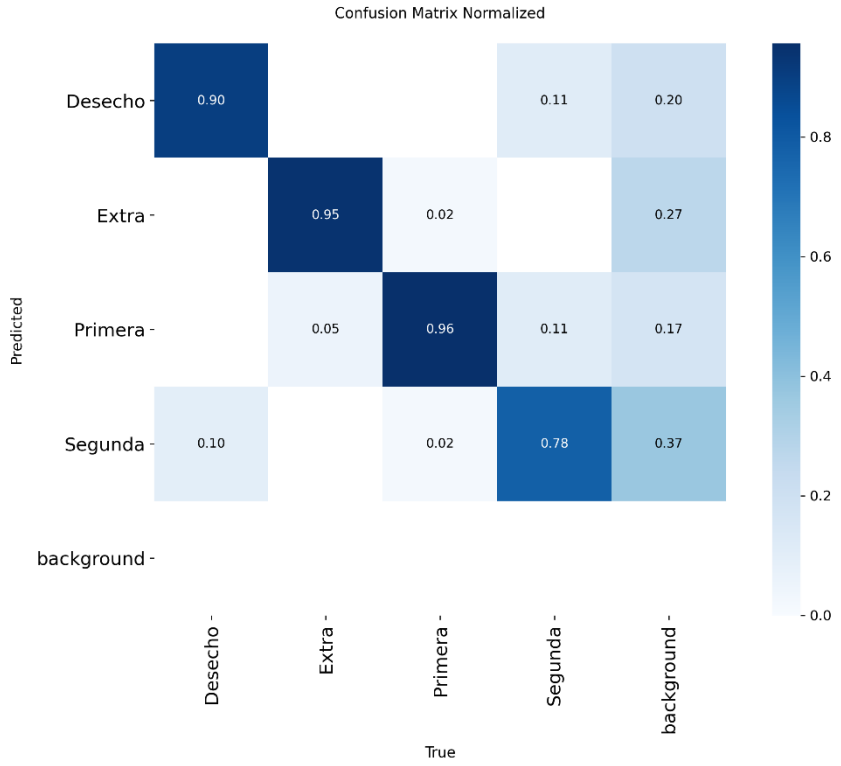


Figura 32. Matriz de confusión del modelo entrenado 3

En conclusión, la matriz de confusión normalizada muestra que las clases “Desecho”, “Extra” y “Primera” tienen altos niveles de precisión (90%, 95% y 96%, respectivamente), lo que indica que el modelo las clasifica correctamente en la mayoría de los casos. Sin embargo, la clase “Segunda” presenta confusiones importantes, siendo mal clasificada como “Desecho” (10%) y “Primera” (11%), y especialmente como “background” (37%). Esto sugiere que el modelo tiene dificultades para distinguir correctamente la clase “Segunda”.

Con los datos obtenidos de la matriz de confusión, se determina la precisión del sistema. En la Tabla 31, se presentan los valores totales de verdaderos positivos, falsos positivos y falsos negativos para las clases Extra, Primera, Segunda y Desecho. En este tercer entrenamiento, los verdaderos positivos reflejan una adecuada capacidad del modelo para identificar correctamente las clases, mientras que los falsos positivos y

negativos continúan concentrándose en clases con menor diferenciación visual, como “Segunda”, la cual sigue mostrando dificultades de clasificación.

Tabla 31. Datos de la matriz de confusión del modelo entrenado 3

Clases	Verdaderos Positivos (VP)	Falsos Positivos (FP)	Falsos Negativos (FN)
Desecho	47	5	10
Extra	69	4	9
Primera	44	2	13
Segunda	28	8	17
Total	188	19	49

La exactitud general del modelo entrenado se calcula mediante el promedio de los valores de verdaderos positivos y falsos positivos de cada clase, usando la fórmula:

$$Precisión \ = \frac{Vp}{Vp \ + \ Fp}$$

Sustituyendo los valores:

$$Precisión \ = \frac{188}{188 \ + \ 19}$$

$$Precisión \ = 0.90$$

Por lo tanto, la exactitud del modelo alcanzó un **0.90 (90%)**, lo que indica un buen desempeño general con un bajo margen de error. A pesar de la persistencia de falsos positivos y negativos, se observa una leve mejora en la precisión respecto a los modelos anteriores. Esto sugiere que el aumento en la cantidad de épocas favoreció un mejor ajuste del modelo, especialmente en las clases “Primera” y “Desecho”, donde la clasificación se volvió más precisa y estable.

c. Precisión del entrenamiento

La Figura 33 muestra la relación Precisión-Confianza que el modelo de clasificación tiene un desempeño general muy bueno, alcanzando una precisión promedio cercana a 1.00 cuando la confianza es aproximadamente 0.981. La clase "Extra" destaca con una precisión consistentemente alta, incluso a bajos niveles de confianza, lo que indica que el modelo la reconoce con gran certeza. Las clases "Desecho" y "Primera" también presentan buenos niveles de precisión, aunque ligeramente inferiores. En contraste, la

clase "Segunda" muestra el rendimiento más bajo, con menor precisión especialmente a niveles bajos de confianza, lo que sugiere que el modelo tiene más dificultad para identificarla correctamente.

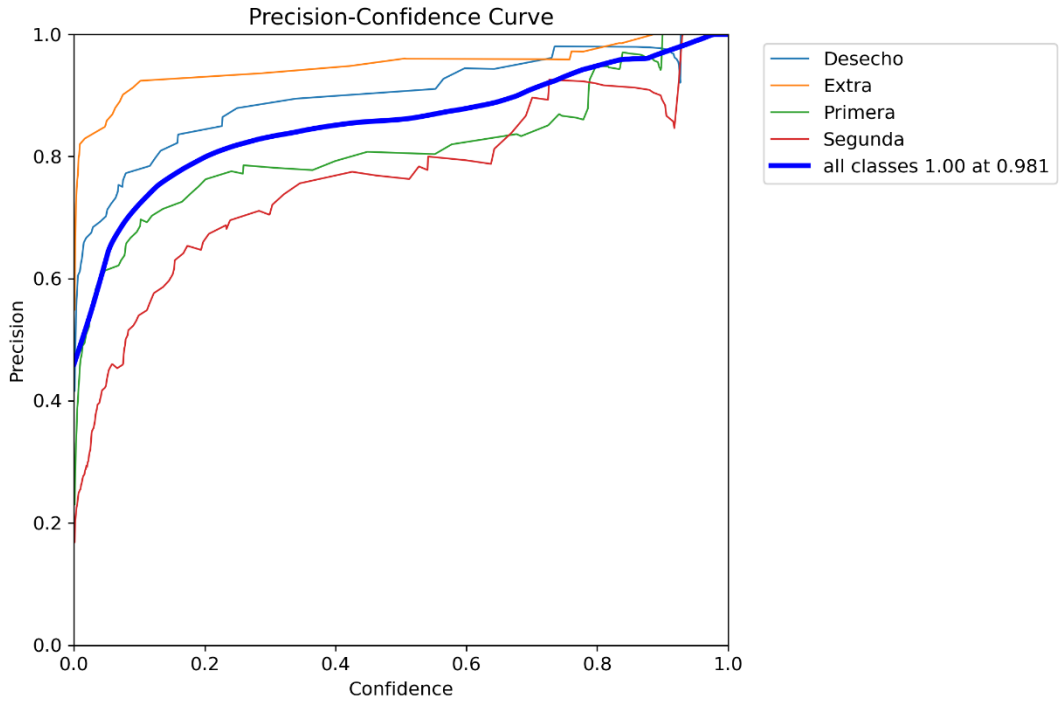


Figura 33. Precisión del modelo 3

3.5.4 Comparación de modelos

Se seleccionó el modelo YOLOv11l entrenado con 100 épocas debido a que presentó el mejor rendimiento entre las tres configuraciones evaluadas como se puede visualizar en la Tabla 32. Alcanzó un mAP50 de 0.947310, superando al modelo con 75 épocas (0.945230) y al de 50 épocas (0.944040). En cuanto al mAP50-95, mantuvo una precisión estable de 0.877280, ligeramente superior al modelo de 75 épocas (0.875690) e igual al de 50 épocas (0.877360). Además, la precisión general fue de 0.919880, significativamente mayor que la del modelo de 75 épocas (0.903110) y mucho más alta que la del modelo de 50 épocas (0.870210). Estas mejoras reflejan que el modelo entrenado con 100 épocas posee una mayor capacidad de detección y clasificación precisa de las fresas, lo que lo convierte en la mejor opción para ser implementado en el sistema de clasificación automatizado.

Tabla 32. Comparación de resultados obtenidos del entrenamiento.

Modelo	**mAP50(B)**	**mAP50-95(B)**	**Precisión (B)**
Yolov11l con 50 épocas	0.944040	0.877360	0.870210
Yolov11l con 75 épocas	0.945230	0.875690	0.903110
Yolov11l con 100 épocas	0.947310	0.877280	0.919880

3.6 Despliegue: Diseño del sistema clasificador de fresas según su calidad

La fase de despliegue, última etapa del modelo CRISP-DM, corresponde a la implementación práctica de la solución desarrollada. En este proyecto, dicha fase abarca las dos etapas finales del esquema general del sistema: la máquina clasificadora y la aplicación de usuario. Se diseñó un sistema automatizado capaz de clasificar fresas en tiempo real según la coloración de su superficie, utilizando un modelo previamente entrenado con YOLOv11l. Este modelo fue integrado en una interfaz gráfica amigable, permitiendo su uso por productores o empacadores sin conocimientos técnicos. Además, se incorporó comunicación serial con hardware externo, como Arduino, con el fin de habilitar su conexión a sistemas físicos de selección automatizada. Esta integración facilita la toma de decisiones rápida y mejora la eficiencia del proceso postcosecha.

3.6.1 Selección de lenguaje de programación

Para la selección del lenguaje de programación, se llevó a cabo un análisis exhaustivo de las características de varios lenguajes utilizados en el ámbito de la inteligencia artificial, como se presenta en la Tabla 16. En esta evaluación, se consideraron aspectos clave como la facilidad de aprendizaje, la disponibilidad de bibliotecas y frameworks especializados, así como las ventajas y desventajas de cada lenguaje. Tras un análisis detallado, se decidió optar por Python como el lenguaje de programación

para este proyecto, debido a su sintaxis clara y concisa, que facilita el desarrollo rápido de prototipos. Además, Python ofrece una integración fluida con herramientas de análisis de datos, lo que lo convierte en una opción ideal para proyectos de inteligencia artificial.

3.6.2 Algoritmo del sistema de clasificación

Es fundamental desarrollar un código en Python junto con una interfaz gráfica para utilizar eficientemente el archivo best.pt por esto se desarrolló un programa en Python el cual se resume en el diagrama de flujo que se muestra en la Figura 34 que ilustra el flujo de trabajo de una aplicación que clasifica fresas mediante una interfaz gráfica, un modelo YOLO, Arduino y video. El usuario inicia la app, se configura la ventana, se carga el modelo y se abre el puerto serial. Al agregar el video, se muestra en la interfaz. Al presionar "Clasificar", se captura un fotograma, se analiza con el modelo YOLO y se obtienen resultados que se muestran en la tabla e imagen. Si hay resultados, se envían por serial al Arduino; si no, se muestra un error. El proceso puede detenerse con el botón "Detener".

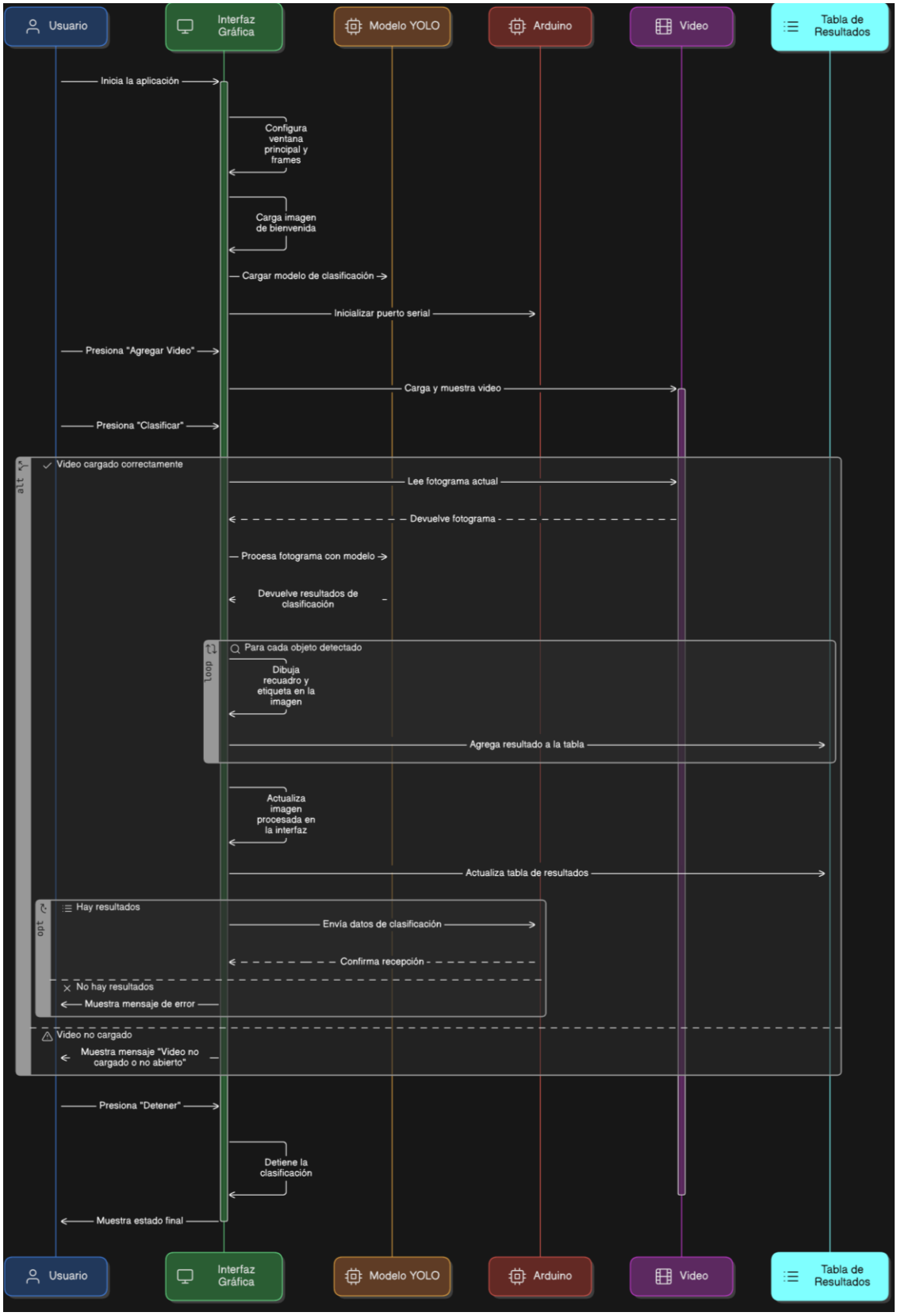


Figura 34. Diagrama de flujo del código de Python

El código comienza importando varias bibliotecas necesarias, como OpenCV, NumPy, Ultralytics (para el modelo YOLO), Matplotlib, Tkinter y Serial como se muestra en la Figura 35.

```
1  import tkinter as tk
2  from PIL import Image, ImageTk
3  import os
4  import cv2
5  from ultralytics import YOLO
6  import serial
7  import time
```

Figura 35. Código de Python: Importación de librerias

Como se observa en la Figura 36, la clase FruitClassificationGUI es la principal del programa y se encarga de crear la interfaz gráfica y sus funciones. En el método __init__, se configura la ventana con fondo beige y un título, se carga el modelo YOLO entrenado (best.pt), se establece la conexión con el puerto serial y se organizan las diferentes áreas de la interfaz: un título superior verde, la imagen de referencia, el cuadro del video en vivo, la imagen procesada, la tabla de resultados con ID y calidad, y los botones para clasificar, detener o agregar video.



```python
class FruitClassificationGUI:
    def __init__(self, master):
        self.master = master
        master.title("Sistema clasificador de fresas según su calidad")
        # Cambiar el color de fondo a beige
        master.configure(bg="#FCFCFC")  # Beige
        # Cargar modelo de IA (YOLOv11l)
        self.model = self.load_model()
        # Inicializar la conexión serial
        self.serial_port = self.initialize_serial()
        # Crear el contenedor principal
        self.main_frame = tk.Frame(master, bg="#dbc483")  # Beige
        self.main_frame.place(x=0, y=0, width=900, height=700)  # Posición y tamaño del contenedor principal
        # Crear el rectángulo en la parte superior para el título
        self.title_frame = tk.Frame(self.main_frame, bg="#006400")  # Verde oscuro
        self.title_frame.place(x=0, y=0, width=900, height=50)  # (x=0, y=0) - 900px de ancho, 50px de alto
        # Etiqueta del título
        self.title_label = tk.Label(self.title_frame, text="Sistema clasificador de fresas según su calidad",
                                    font=("Arial", 16, "bold"), bg="#006400", fg="white", padx=20, pady=10)
        self.title_label.pack(fill="both")  # Llenar el espacio disponible
        # Crear la sección de imágenes
        self.image_frame = tk.Frame(self.main_frame, bg="#dbc483")  # Beige
        self.image_frame.place(x=150, y=55, width=590, height=190)  # (x=150, y=55) - 590px de ancho, 190px de alto
        # Cargar y mostrar la imagen de las fresas (inicial)
        self.image_path = r"C:\Users\USUARIO\Desktop\Tesis\Prueba_Entrenamiento\interfazusar.png"
        self.image = Image.open(self.image_path)  # Abrir la imagen
        self.image_tkinter = ImageTk.PhotoImage(self.image)  # Convertir a formato compatible con Tkinter
        self.image_label = tk.Label(self.image_frame, image=self.image_tkinter)  # Crear etiqueta para la imagen
        self.image_label.pack(side="top", pady=10)  # Mostrar la imagen en el marco
        # Crear el cuadrado para el video
        self.video_frame = tk.Frame(self.main_frame, width=450, height=250, bg="gray")  # Ajustar altura
        self.video_frame.place(x=20, y=250, width=400, height=400)  # (x=20, y=250) - 400px de ancho, 400px de alto
        # Etiqueta "Video en vivo" encima del cuadro de video
        self.video_label = tk.Label(self.video_frame, text="VIDEO INGRESADO",
                                    font=("Arial", 12, "bold"), bg="#006400", fg="white")  # Verde oscuro
        self.video_label.pack()  # Mostrar la etiqueta
        # Crear el cuadro para la imagen procesada
        self.additional_image_label1 = tk.Label(self.processed_image_frame)
        self.additional_image_label1.pack()  # Mostrar la etiqueta
        # Crear la sección de tabla
        self.table_frame = tk.Frame(self.main_frame, bg="#f5f5dc")  # Beige
        self.table_frame.place(x=760, y=60, width=115, height=110)  # (x=760, y=60) - 115px de ancho, 110px de alto
        # Crear la tabla
        self.table = tk.Frame(self.table_frame, bg="#f5f5dc")  # Beige
        self.table.pack(fill="both", expand=True)  # Llenar el espacio disponible
        # Encabezado de la tabla (solo se agrega una vez)
        self.table_header = tk.Frame(self.table, bg="#f5f5dc")  # Beige
        self.table_header.pack(fill="x")  # Llenar el espacio horizontal
        # Etiquetas del encabezado de la tabla
        tk.Label(self.table_header, text="ID", font=("Arial", 12, "bold"), bg="#f5f5dc").pack(side="left", padx=10)
        tk.Label(self.table_header, text="Calidad", font=("Arial", 12, "bold"), bg="#f5f5dc").pack(side="left", padx=10)
        # Crear un marco para los botones
        self.button_frame = tk.Frame(self.main_frame, bg="#dbc483")  # Beige
        self.button_frame.place(relx=1.0, rely=1.0, anchor='se', x=-20, y=-20)  # Posición en la esquina inferior derecha
        # Crear los botones
        self.start_button = tk.Button(self.button_frame, text="Clasificar", command=self.start_classification)
        self.start_button.pack(side="left", padx=10)  # Botón para iniciar la clasificación
        self.stop_button = tk.Button(self.button_frame, text="Detener", command=self.stop_classification)
        self.stop_button.pack(side="left", padx=10)  # Botón para detener la clasificación
        self.add_video_button = tk.Button(self.button_frame, text="Agregar Video", command=self.add_video)
        self.add_video_button.pack(side="left", padx=10)  # Botón para agregar un video
        # Inicializar variables
        self.classification_running = False  # Estado de la clasificación
        self.video_capture = None  # Variable para la captura de video
        self.video_label = None  # Variable para la etiqueta del video
        self.video_path = r"C:\Users\USUARIO\Desktop\Tesis\Prueba_Entrenamiento\video2.mp4"  # Ruta del video
        self.video_width = 400  # Ancho del video
        self.video_height = 400  # Altura del video
        # Inicializar la tabla de resultados
        self.results_table = []  # Lista para almacenar los resultados de la clasificación
```

Figura 36. Código de Python: Función __init__

La función que le presenta en la Figura 37, load_model, carga el modelo entrenado YOLOv11l para detectar fresas.

```python
def load_model(self):
    # Cargar el modelo YOLOv11l
    model_path = r"C:\Users\USUARIO\Desktop\Tesis\Prueba_Entrenamiento\best100.pt"
    model = YOLO(model_path)  # Cambiado a la nueva forma de cargar el modelo
    return model
```

Figura 37. Código de Python: Función para cargar el modelo YOLOv11l

La función mostrada en la Figura 38, initialize_serial, establece conexión serial con Arduino por puerto COM4.

```python
def initialize_serial(self):
    # Inicializar la conexión serial
    try:
        ser = serial.Serial('COM4', 9600, timeout=1)  # Cambia 'COM4' al puerto correcto
        time.sleep(2)  # Esperar a que se establezca la conexión
        return ser
    except serial.SerialException as e:
        print(f"Error al abrir el puerto serial: {e}")
        return None
```

Figura 38. Código de Python: Función para establecer conexión con serial

La función que se muestra en la Figura 39, start_classification lee un fotograma del video y lo analiza usando el modelo YOLO a través de process_frame_with_model. Luego, muestra la imagen procesada en la interfaz gráfica, actualiza la tabla de resultados con los ID y etiquetas detectadas usando update_results_table, y finalmente envía estos datos al Arduino mediante la función send_data_to_arduino.

```python
def start_classification(self):
    # Iniciar el proceso de clasificación
    if self.video_capture is None or not self.video_capture.isOpened():
        print("Video no cargado o no abierto.")
        return

    # Capturar un fotograma actual
    ret, frame = self.video_capture.read()
    if not ret:
        print("No se pudo capturar el fotograma del video.")
        return

    # Procesar con IA
    self.results_table.clear()  # Limpiar la tabla antes de cada clasificación
    annotated_frame = self.process_frame_with_model(frame)

    # Convertir a imagen a mostrar en Tkinter
    image_tk = ImageTk.PhotoImage(image=Image.fromarray(annotated_frame).resize((400, 400)))

    # Mostrar en el recuadro imagen para procesar
    self.additional_image_label1.configure(image=image_tk)
    self.additional_image_label1.image = image_tk

    # Actualizar la tabla con los resultados
    self.update_results_table()

    # Enviar datos por serial
    self.send_data_to_arduino()

    print("Clasificación y anotación completadas.")
```

Figura 39. Código de Python: Función para iniciar la clasificación de fresas

En la Figura 40, se presenta la función process_frame_with_model, detecta fresas en el fotograma usando el modelo YOLO, dibuja cajas con etiquetas como "1: buena" y filtra objetos muy cercanos para evitar duplicados. Solo guarda hasta 4 fresas por fotograma y ordena los resultados según su posición en la imagen.

```python
def process_frame_with_model(self, frame):
    # Detectar objetos en el fotograma
    results = self.model(frame)[0]
    # Lista para guardar objetos únicos
    detected_objects = []
    for i, box in enumerate(results.boxes):
        x1, y1, x2, y2 = map(int, box.xyxy[0])  # Coordenadas de la caja delimitadora
        conf = float(box.conf[0])  # Confianza de la detección
        cls = int(box.cls[0])  # Clase del objeto detectado
        label = results.names[cls]  # Nombre de la clase
        center_x = (x1 + x2) // 2  # Coordenada X del centro
        center_y = (y1 + y2) // 2  # Coordenada Y del centro
        # Verificar si ya hay un objeto muy cerca (evita encimados)
        too_close = False
        for obj in detected_objects:
            ox, oy = obj[3]
            if abs(center_x - ox) < 30 and abs(center_y - oy) < 30:
                too_close = True
                break
        if not too_close:
            detected_objects.append((center_x, label, (x1, y1, x2, y2), (center_x, center_y)))

        if len(detected_objects) >= 4:  # Limitar a 4 objetos detectados
            break
    # Ordenar de izquierda a derecha
    detected_objects.sort(key=lambda x: x[0])
    # Dibujar las cajas y etiquetas en el fotograma
    for idx, (center_x, label, (x1, y1, x2, y2), _) in enumerate(detected_objects, start=1):
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)  # Dibujar la caja delimitadora
        cv2.putText(frame, f"{idx}: {label}", (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0), 2)  # Dibujar la etiqueta
        # Agregar a la tabla de resultados
        self.results_table.append((idx, label))
    return frame
```

Figura 40. Código Python: Función para marcar y enumerar los frames

En la Figura 41, se presenta la función update_results_table, que actualiza la tabla en la interfaz con los resultados detectados: ID y calidad.

```python
def update_results_table(self):
    # Limpiar la tabla existente (sin eliminar los encabezados)
    for widget in self.table.winfo_children():
        if widget not in self.table_header.winfo_children():  # No eliminar encabezados
            widget.destroy()

    # Agregar resultados a la tabla
    for id, label in self.results_table:
        row = tk.Frame(self.table, bg="#f5f5dc")  # Beige
        row.pack(fill="x")  # Llenar el espacio horizontal
        tk.Label(row, text=id, bg="#f5f5dc").pack(side="left", padx=10)  # Mostrar ID
        tk.Label(row, text=label, bg="#f5f5dc").pack(side="left", padx=10)  # Mostrar calidad
```

Figura 41. Código Python: Función para agregar los resultados detectados a una tabla

En la Figura 42, se presenta la función send_data_to_arduino que verifica si existe conexión serial y si hay datos disponibles. Luego, envía al Arduino los resultados de clasificación en el formato ID:Calidad, separados por comas, como por ejemplo: 1:Extra,2:Primera.

```
def send_data_to_arduino(self):
    # Enviar datos de clasificación al Arduino
    if self.serial_port and self.results_table:
        # Convertir la tabla de resultados a un formato adecuado para enviar
        data = ','.join([f"{id}:{label}" for id, label in self.results_table])
        self.serial_port.write(data.encode('utf-8') + b'\n')  # Enviar datos y agregar un salto de línea
        print(f"Enviando por serial: {data}")  # Imprimir los datos enviados
```

Figura 42. Código Python: Función para enviar los datos al Arduino

En la Figura 43, se presenta la función stop_classification, detiene el proceso automático de clasificación al cambiar la bandera classification_running a False y muestra un mensaje en la consola indicando que la clasificación de fresas ha sido detenida.

```
def stop_classification(self):
    # Detener la clasificación
    self.classification_running = False
    print("Clasificación de fresas detenida")
```

Figura 43. Código Python: Función para detener la clasificación de fresas

En la Figura 44, se presenta la función add_video que carga un video desde la ruta definida, verifica que se abra correctamente, ajusta su tamaño según los parámetros establecidos, lo muestra en la interfaz gráfica y llama a play_video() para iniciar su reproducción continua.

```
def add_video(self):
    # Agregar video para la clasificación
    self.video_capture = cv2.VideoCapture(self.video_path)

    if not self.video_capture.isOpened():
        print("Error al abrir el video.")
        return

    self.video_capture.set(cv2.CAP_PROP_FRAME_WIDTH, self.video_width)  # Establecer ancho del video
    self.video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT, self.video_height)  # Establecer altura del video

    if self.video_label is None:
        self.video_label = tk.Label(self.video_frame)  # Crear etiqueta para el video
        self.video_label.pack()

    self.play_video()  # Iniciar la reproducción del video
```

Figura 44. Código Python: Función para agregar el video para analizar

En la Figura 45, se presenta la función play_video lee fotogramas del video uno por uno, los convierte a formato RGB y los muestra en la interfaz. Se ejecuta de forma repetitiva cada 33 milisegundos para simular una reproducción de aproximadamente 30 fps. Cuando el video termina, libera la captura y limpia la imagen mostrada.

```python
def play_video(self):
    # Reproducir el video en el cuadro de video
    if self.video_capture is not None and self.video_label is not None:
        ret, frame = self.video_capture.read()  # Leer un fotograma del video

        if ret:
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)  # Convertir BGR a RGB
            photo = ImageTk.PhotoImage(image=Image.fromarray(frame_rgb).resize((self.video_width, self.video_height)))

            self.video_label.configure(image=photo)  # Actualizar la etiqueta del video
            self.video_label.image = photo  # Mantener una referencia a la imagen

            self.current_frame = frame  # Guardar para usar en clasificación

            self.video_label.after(33, self.play_video)  # Repetir cada 33 ms
        else:
            self.video_capture.release()  # Liberar la captura de video
            self.video_label.configure(image=None)  # Limpiar la etiqueta del video
            self.video_label.image = None  # Limpiar la referencia a la imagen
```

Figura 45. Código Python: Función para reproducir el video que se va a analizar

Con la implementación del código en Python, se obtienen los resultados presentados en la Figura 46, donde se optimiza el modelo entrenado para que clasifique las fresas en cuatro categorías de calidad para las fresas: “Extra”, “Primera”, “Segunda” y “Desecho”.

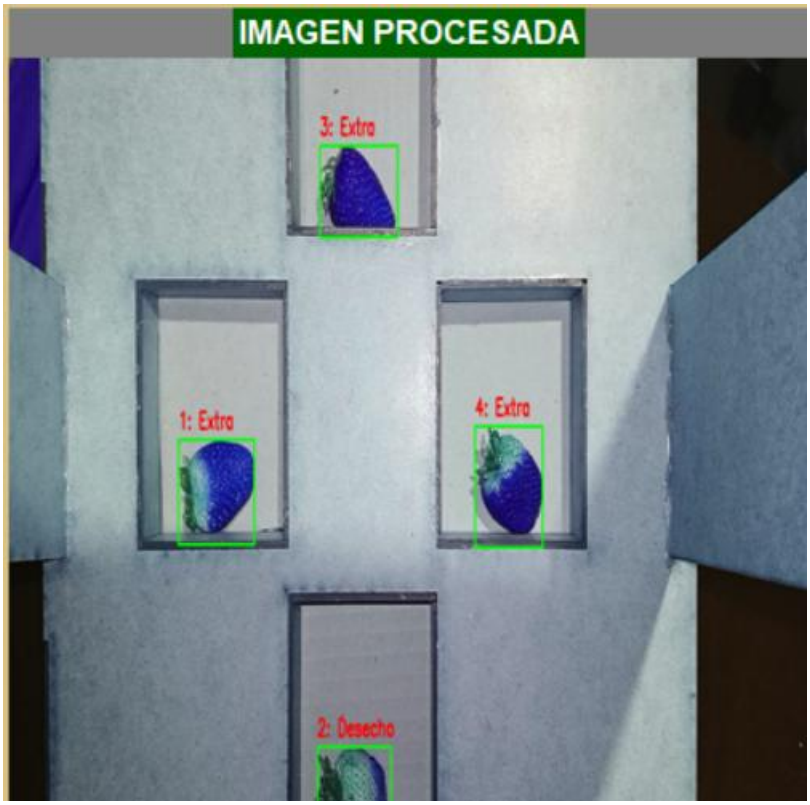


Figura 46. Resultados del sistema de clasificación

3.6.3 Selección de microcontrolador

En base a la comparación realizada en la Tabla 5, el Arduino Mega es una excelente opción como microcontrolador para desarrollar una máquina clasificadora debido a su gran capacidad de entradas y salidas, lo que permite controlar múltiples actuadores simultáneamente. Además, su amplio espacio de memoria (256 KB de memoria Flash) y su procesador ATmega2560 permiten manejar tareas complejas de control y procesamiento de señales, asegurando un rendimiento estable y confiable en sistemas que dependen de la interacción con distintos componentes.

3.6.4 Algoritmo de clasificación para el microcontrolador

El diagrama de flujo que se presenta en la Figura 47, representa un programa de Arduino que gestiona la activación de servomotores según comandos enviados desde el monitor serial. Al iniciar, se conecta cada servo y se posiciona en 0°. Luego, el

sistema espera un comando del usuario, como "1:Extra,2:Primera", que se procesa y distribuye por tipo. Según el grupo recibido (Extra, Primera, Segunda o Desecho), el servo direccional se mueve a una posición específica (0°, 90°, 180° o 270°), y se activan los servos correspondientes, que giran a 90° y regresan a 0°. El proceso termina con la restauración de los valores iniciales.

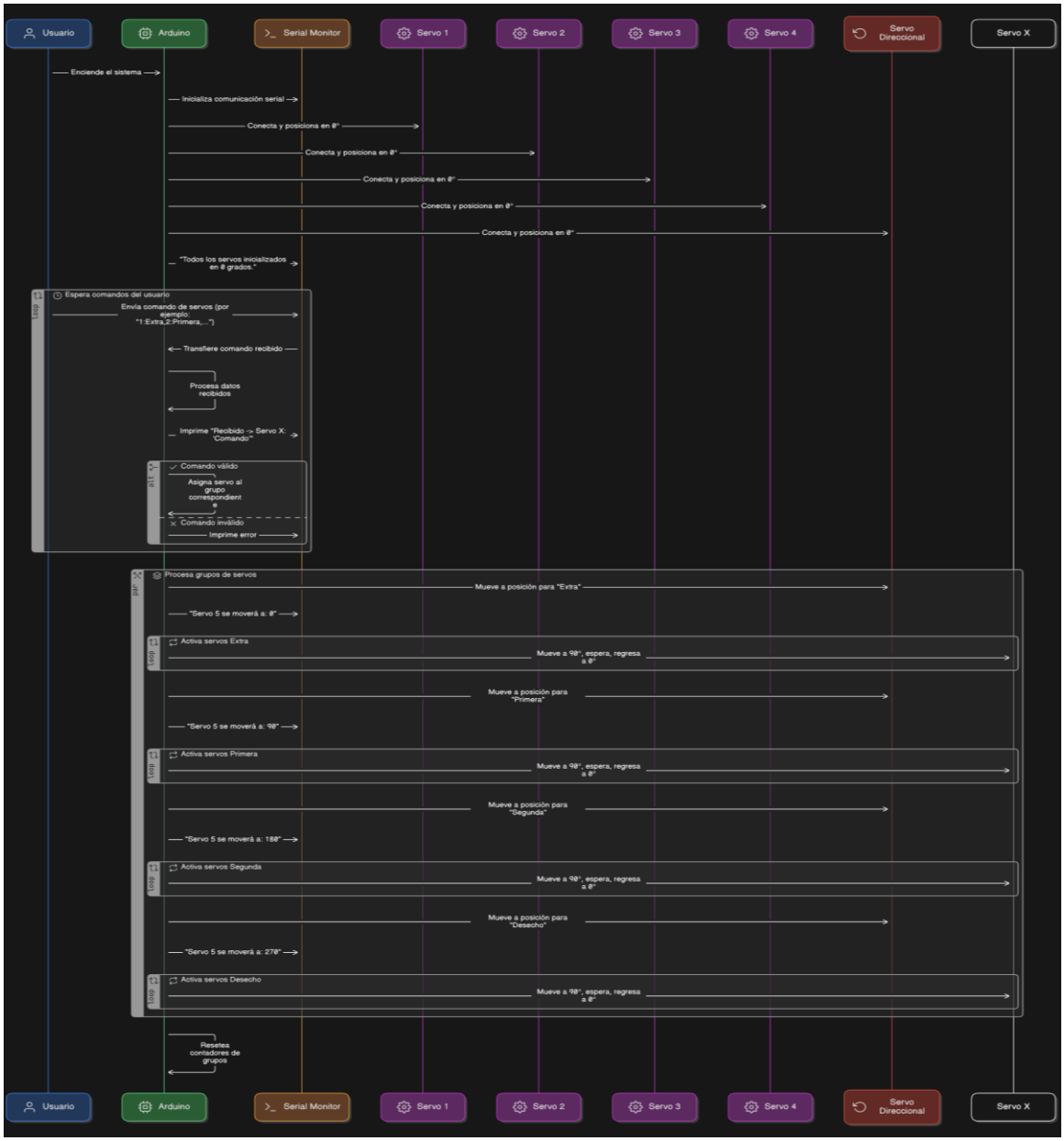


Figura 47. Diagrama de flujo del condigo de Arduino

El código mostrado en la Figura 48 comienza con la inclusión de la librería **Servo.h**, necesaria para controlar cinco servomotores. Cuatro de ellos se encargan de clasificar los objetos según su calidad, mientras que el quinto indica visualmente la categoría

detectada. Para organizar su funcionamiento, se crean arreglos que agrupan los índices de los servos según las categorías "Extra", "Primera", "Segunda" y "Desecho". Además, se establece la comunicación serial y se definen los pines del Arduino donde se conectan los servomotores.

```
#include <Servo.h>

Servo servos[5];  // Ahora hay 5 servos

// Arrays para guardar los índices de servos por comando
int extraServos[4], primeraServos[4], segundaServos[4], desechoServos[4];
int extraCount = 0, primeraCount = 0, segundaCount = 0, desechoCount = 0;

// Último comando recibido
String lastCommand = "";

void setup() {
  Serial.begin(9600);
  servos[0].attach(3);
  servos[1].attach(5);
  servos[2].attach(6);
  servos[3].attach(9);
  servos[4].attach(10);  // Nuevo servo agregado
}
```

Figura 48. Código de Arduino: Definición de constantes, inclusión de bibliotecas y comunicación serial.

La función loop() que se presenta en la Figura 49, espera recibir datos desde la PC en formato como "1:Extra,2:Primera,3:Primera, 4:Primera ". Una vez recibido el mensaje, llama a la función processData() para interpretar y organizar los datos según la calidad detectada y preparar los servos correspondientes para su activación.

```
17  void loop() {
18    if (Serial.available()) {
19      String data = Serial.readStringUntil('\n');
20      processData(data);
21
22      activateGroup(extraServos, extraCount, "Extra");
23      delay(3000);
24      activateGroup(primeraServos, primeraCount, "Primera");
25      delay(3000);
26      activateGroup(segundaServos, segundaCount, "Segunda");
27      delay(3000);
28      activateGroup(desechoServos, desechoCount, "Desecho");
29
30      // Reset counts
31      extraCount = primeraCount = segundaCount = desechoCount = 0;
32    }
33  }
```

Figura 49. Código Arduino: Función loop().

En la función processData() que se presentada en la Figura 50, lee el mensaje recibido, separa cada par servo:comando, guarda el número de servo en el array correspondiente y también guarda el último comando recibido.

```
35  void processData(String input) {
36    input.trim();
37    int start = 0;
38
39    while (start < input.length()) {
40      int comma = input.indexOf(',', start);
41      if (comma == -1) comma = input.length();
42      String pair = input.substring(start, comma);
43      int colon = pair.indexOf(':');
44      if (colon != -1) {
45        int servoNum = pair.substring(0, colon).toInt();
46        String command = pair.substring(colon + 1);
47        command.trim();
48
49        Serial.print("Recibido -> Servo ");
50        Serial.print(servoNum);
51        Serial.print(": '");
52        Serial.print(command);
53        Serial.println("'");
54
55        if (servoNum >= 1 && servoNum <= 4) {
56          int index = servoNum - 1;
57
58          if (command == "Extra") extraServos[extraCount++] = index;
59          else if (command == "Primera") primeraServos[primeraCount++] = index;
60          else if (command == "Segunda") segundaServos[segundaCount++] = index;
61          else if (command == "Desecho") desechoServos[desechoCount++] = index;
62        }
63      }
64      start = comma + 1;
65    }
66  }
```

Figura 50. Código: Función processData().

En la sección del código que se muestra en la Figura 51, activa los servos pertenecientes a un grupo específico según la calidad detectada (Extra, Primera, Segunda o Desecho). Además, llama a activateFifthServo() para mover el servo 5 a un ángulo que representa visualmente dicha calidad, reforzando así la clasificación con un movimiento indicativo adicional.

```
77  void activateGroup(int group[], int count, String name) {
78    Serial.print("Activando servos para comando: ");
79    Serial.println(name);
80    for (int i = 0; i < count; i++) {
81      int index = group[i];
82      Serial.print("  Servo ");
83      Serial.println(index + 1);
84      servos[index].write(180);
85    }
86    activateFifthServo(name);  // Activar el nuevo servo según la calidad
87    delay(1000);
88    for (int i = 0; i < count; i++) {
89      servos[group[i]].write(0);
90    }
91  }
92  void activateFifthServo(String quality) {
93    int angle = 0;
94
95    if (quality == "Extra") angle = 0;
96    else if (quality == "Primera") angle = 90;
97    else if (quality == "Segunda") angle = 180;
98    else if (quality == "Desecho") angle = 270;
99
100   Serial.print("Servo 5 se moverá a: ");
101   Serial.println(angle);
102
103   servos[4].write(angle);
104   delay(1000);
105   servos[4].write(0);  // Regresa a posición inicial
106 }
```

Figura 51. Código de Arduino: Función activateGroup() y activateFifthServo()

3.6.5 Prototipo de la máquina clasificadora

En esta etapa se recibirá la información captada por la cámara, que adquiere datos sobre el color y tamaño de las fresas. Esta información se procesará mediante algoritmos de inteligencia artificial, que evaluarán la calidad de las fresas en tiempo real. Utilizando un microcontrolador, se enviarán señales a los actuadores, que

realizarán el proceso de clasificación, separando las fresas en sus respectivos tipos de calidad. Este enfoque automatizado garantiza una clasificación eficiente y precisa, optimizando la selección de las fresas de acuerdo con cada tipo de calidad que estas tengan.

a. Selección de actuadores

En base a la comparación realizada en la Tabla 4 se ha decidido emplear servomotores de la marca TowerPro SG90 y Mg995 ya que estos son ideales para proyectos pequeños que requieren un esfuerzo de torsión moderado, lo cual es suficiente para la aplicación que va a tener en el sistema de clasificación automática de fresas.

b. Construcción del prototipo

Con base en el diseño en 3D presentado en la Figura 52, se procedió a la construcción del prototipo físico del sistema clasificador. Para ello, se utilizó MDF como material principal, seleccionado por su versatilidad, resistencia y facilidad de trabajo. Una de las principales ventajas del MDF es que permite cortar con láser las piezas que conforman la estructura, lo que facilitó un ensamblaje más preciso y limpio. Este proceso permitió replicar fielmente el modelo digital, asegurando que las dimensiones y funcionalidades del diseño se mantuvieran durante la fabricación del prototipo, el cual será clave para validar su funcionamiento real.

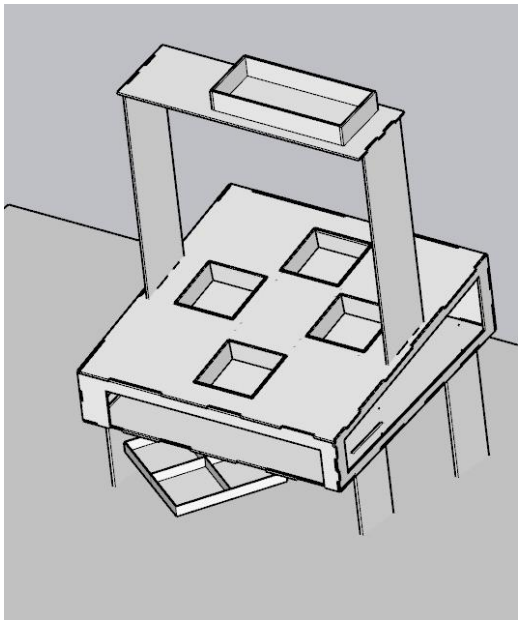


Figura 52. Modelado en 3D del prototipo

Las conexiones de los actuadores se han realizado siguiendo las directrices establecidas en la Figura 55, donde se presenta cómo deben conectarse todos los componentes, incluyendo los actuadores, indicadores y pulsadores, a su respectiva interfaz con Arduino.

Cada actuador y pulsador ha sido cuidadosamente posicionado en el prototipo como se muestra en la Figura 53, para maximizar la accesibilidad y la ergonomía, permitiendo una operación intuitiva y efectiva.

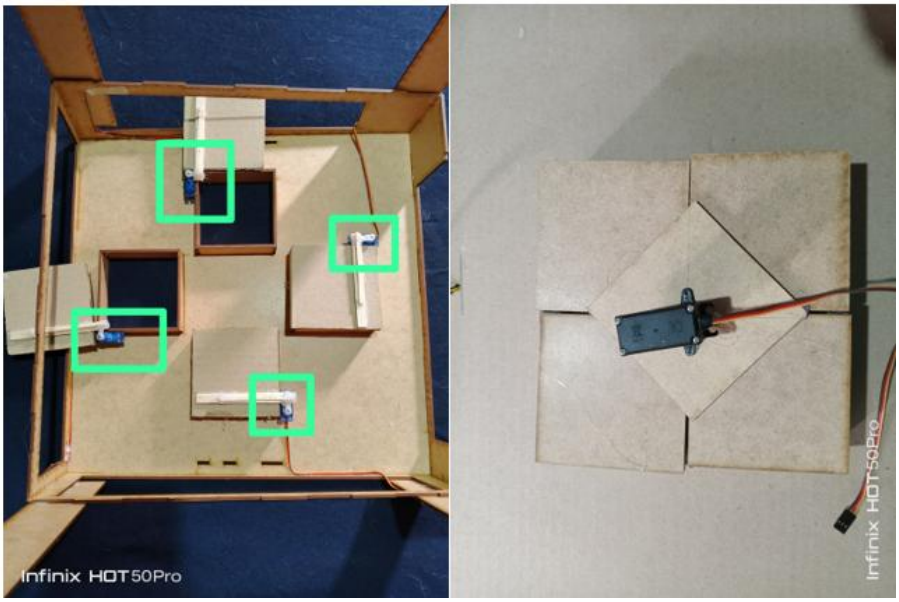


Figura 53. Colocación de los actuadores en el prototipo.

En la Figura 54, se presenta el prototipo finalizado, que, a través del microcontrolador, recibirá datos mediante comunicación serial. Este sistema está diseñado para llevar a cabo la clasificación de las fresas de acuerdo con las cuatro calidades diferentes previamente establecidas.

El microcontrolador actuará como el cerebro del sistema, procesando la información recibida y ejecutando las acciones físicas necesarias para clasificar las fresas de manera eficiente.



Figura 54. Prototipo del sistema de clasificación automática de fresas en base a su calidad.

Una de las principales limitaciones del prototipo radica en las condiciones específicas que deben cumplirse para garantizar su correcto funcionamiento. Para que el sistema de clasificación opere de forma precisa, es fundamental contar con una iluminación adecuada, ya que variaciones en la luz pueden afectar negativamente el análisis visual de las fresas. Además, es crucial que las frutas se coloquen correctamente en el área de detección; una posición inadecuada podría impedir que el modelo capture las características necesarias para su clasificación. Otro aspecto para considerar es el ángulo en que se posiciona la fresa, dado que el prototipo solo cuenta con una cámara que visualiza una sola cara del fruto. Esto limita la evaluación a una perspectiva

parcial, lo que podría generar errores si la calidad visible no representa el estado general del producto.

c. Diagrama del circuito a implementar

En la Figura 55, se presenta el circuito diseñado para la máquina de clasificación automática, este circuito es fundamental para el funcionamiento de la máquina, ya que permitirá clasificar de forma física las fresas de diferentes calidades. La integración de los servomotores garantiza un movimiento preciso y controlado, mientras que los pulsadores proporcionan una interfaz intuitiva para el usuario. Los indicadores, por su parte, ofrecerán información visual sobre el estado del sistema, facilitando su monitoreo y operación.

El microcontrolador, como cerebro del sistema, coordinará todas las acciones, asegurando que cada componente trabaje en armonía. Este diseño no solo optimiza el rendimiento de la máquina, sino que también mejora la calidad del proceso de clasificación, permitiendo una separación eficiente de las fresas según su calidad.

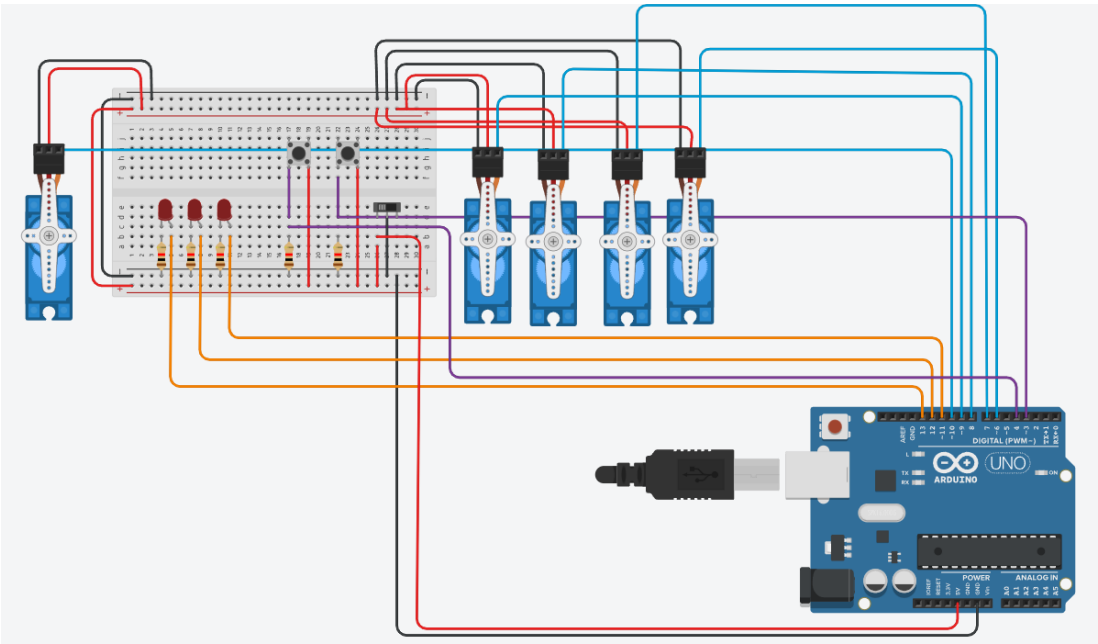


Figura 55. Circuito de la máquina clasificadora automática de fresas.

3.6.6 Visualización de los resultados

En la etapa final del sistema de clasificación automática de fresas, se visualizan y analizan los resultados obtenidos durante el proceso de clasificación basado en la calidad de las fresas. Esta fase permite evaluar el rendimiento del sistema, enfocándose en aspectos clave como la precisión en la clasificación correcta, la tasa de errores o fallos del sistema, y la eficiencia general del proceso. Esta etapa es crucial para validar el éxito del proyecto y realizar los ajustes necesarios antes de su implementación.

En este contexto, se desarrolló una interfaz gráfica de usuario que se observa en la Figura 56 que facilita la interacción con el prototipo, mostrando en tiempo real el video capturado por la cámara. La interfaz cuenta con tres botones principales: "Agregar video", "Clasificar" y "Detener". Al presionar el botón "Agregar video", se inicia la captura del video desde la cámara del sistema; luego, al seleccionar "Clasificar", se toma una imagen del video en tiempo real, la cual se visualiza en la misma interfaz y se envía al algoritmo de inteligencia artificial para su procesamiento. Este algoritmo analiza la imagen y determina la calidad de la fresa, clasificándola en una de las cuatro categorías establecidas. Finalmente, los datos obtenidos son enviados al microcontrolador, que activa las acciones necesarias para ejecutar la clasificación automática. Por otro lado, el botón "Detener" permite finalizar la ejecución del sistema, deteniendo tanto la captura del video como el funcionamiento general del prototipo.



Figura 56. Interfaz gráfica de usuario

3.7 Pruebas de funcionamiento del sistema

Durante las pruebas de funcionamiento, se utilizaron fresas de diferentes calidades, incluyendo fresas de calidad “Extra”, “Primera”, “Segunda” y “Desecho”. Esta diversidad en las muestras permitió contrastar los resultados en todos los escenarios posibles, validando la precisión del sistema en la clasificación.

En la Figura 57 y en la Figura 58, se presenta el resultado del procesamiento en tiempo real de la imagen capturada, donde se evidencia la implementación del sistema de detección y clasificación de fresas. Cada fresa detectada es delimitada por un rectángulo. El sistema muestra, para cada espécimen identificado, información crucial como la etiqueta de clasificación y un identificador único (ID), permitiendo una verificación inmediata de la precisión del sistema y facilitando el control de calidad en el proceso automatizado de selección de fresas.



Figura 57. Resultado 1



Figura 58. Resultado 2

En la Figura 59 y Figura 60, muestran la transmisión de datos por comunicación serial al microcontrolador, donde cada mensaje contiene el identificador y la categoría de calidad asignada a cada fresa detectada. Esta información es procesada para controlar los actuadores del sistema, permitiendo la clasificación física automática de las fresas según su calidad determinada.

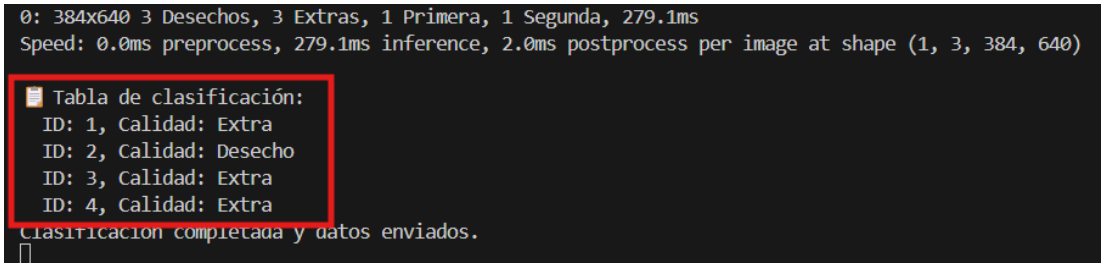


Figura 59. Mensaje serial que se envía al microcontrolador del resultado 1

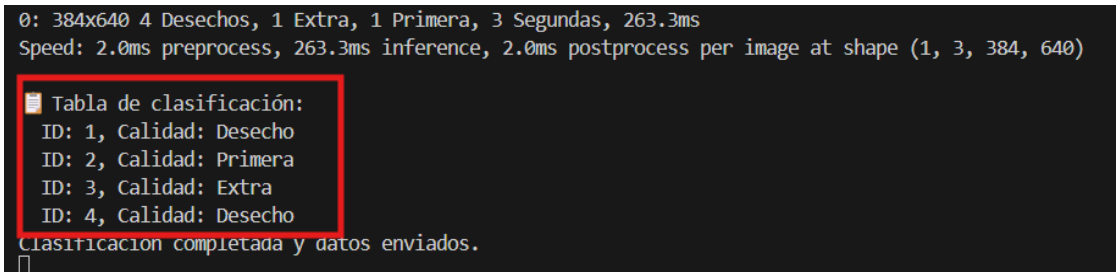


Figura 60. Mensaje serial que se envía al microcontrolador del resultado 2

En la Figura 61 y Figura 62, se muestran las acciones que dicta el microcontrolador a los actuadores en base a los resultados recibidos por el puerto serial. El código recibe el mensaje serial 1:Extra, 2:Desecho, 3:Extra, 4:Extra y clasifica los servos según la calidad indicada. Primero, mueve el servo 5 (direccional) a 0° porque hay servos tipo “Extra”, luego activa los servos 1, 3 y 4 llevándolos a 90° y regresándolos a 0°. Después mueve el servo 5 a 90° (Primera) y a 180° (Segunda), pero como no hay servos asignados a esas categorías, no se activan. Finalmente, mueve el servo 5 a 270° para “Desecho” y activa el servo 2. Todo el proceso se imprime en el monitor serial.

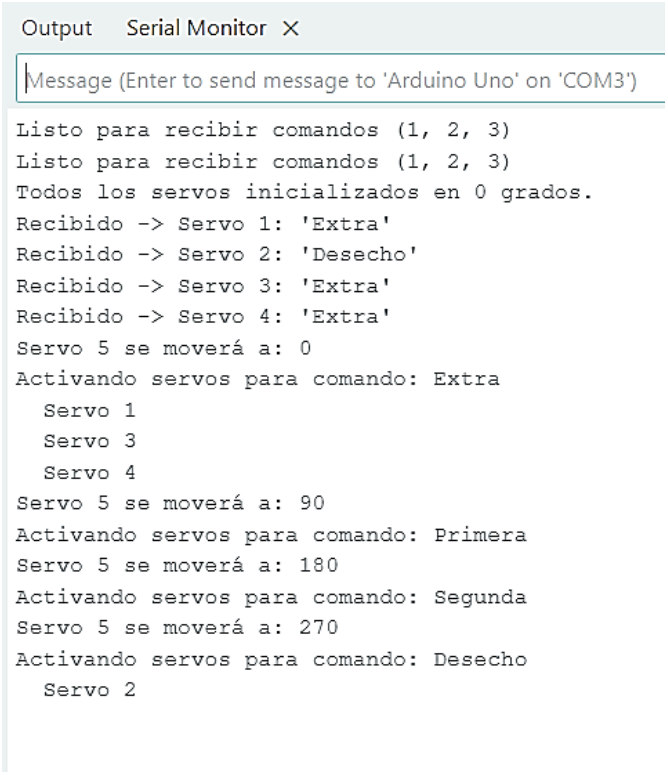


Figura 61. Acciones que realiza el microcontrolador con los datos del resultado 1

```
Recibido -> Servo 1: 'Desecho'
Recibido -> Servo 2: 'Segunda'
Recibido -> Servo 3: 'Extra'
Recibido -> Servo 4: 'Primera'
Activando servos para comando: Extra
  Servo 3
Servo 5 se moverá a: 0
Activando servos para comando: Primera
  Servo 4
Servo 5 se moverá a: 90
Activando servos para comando: Segunda
  Servo 2
Servo 5 se moverá a: 180
Activando servos para comando: Desecho
  Servo 1
Servo 5 se moverá a: 270
```

Figura 62. Acciones que realiza el microcontrolador con los datos del resultado 2

3.7.1 Resultados de las pruebas de funcionamiento

Para evaluar la efectividad del sistema de control de calidad de fresas en etapa de precosecha, se llevó a cabo un análisis comparativo entre los resultados obtenidos mediante inspección visual tradicional basa en los parámetros mencionados en la Tabla

27 y la inspección artificial implementada por el sistema. Como se observa en la Tabla 33.

Tabla 33. Pruebas de funcionamiento

Control de Calidad		Inspección visual humana				Inspección artificial			
		Grado de Calidad				Grado de Calidad			
# Pruebas	#Fresas	Extra	Primera	Segunda	Desecho	Extra	Primera	Segunda	Desecho
Prueba 1	4	3	1	0	0	2	2	0	0
Prueba 2	4	1	1	2	0	1	1	2	0
Prueba 3	4	1	1	1	1	1	1	1	1
Prueba 4	4	0	3	1	0	0	3	1	0
Prueba 5	4	2	0	1	1	2	0	2	0
Prueba 6	4	2	2	0	0	2	1	1	0
Prueba 7	4	0	0	4	0	0	0	4	0
Prueba 8	4	0	2	2	0	0	2	2	0
Prueba 9	4	0	0	3	1	0	0	3	1
Prueba 10	4	1	1	0	2	1	1	1	1
Prueba 11	4	0	2	2	0	0	1	3	0
Prueba 12	4	1	1	1	1	1	2	0	1
Prueba 13	4	0	3	0	1	0	3	0	1
Prueba 14	4	1	0	2	1	1	0	2	1
Prueba 15	4	2	0	2	0	2	0	2	0
Prueba 16	4	0	2	1	1	0	2	1	1
Prueba 17	4	0	0	4	0	0	0	4	0
Prueba 18	4	1	1	1	1	1	1	1	1
Prueba 19	4	0	3	1	0	0	3	1	0
Prueba 20	4	1	1	0	2	1	1	0	2
Prueba 21	4	0	0	0	4	0	1	0	3
Prueba 22	4	1	1	1	1	1	1	1	1
Prueba 23	4	1	2	0	1	1	2	0	1
Prueba 24	4	0	2	2	0	0	1	3	0
Prueba 25	4	1	2	1	0	1	2	1	0
Prueba 26	4	0	2	2	0	0	2	2	0
Prueba 27	4	1	2	0	1	1	2	0	1
Prueba 28	4	0	0	4	0	0	1	3	0
Prueba 29	4	1	2	1	0	0	2	2	0
Prueba 30	4	0	2	2	0	0	1	3	0

3.7.2 Análisis de los resultados del sistema

a. Cálculo del error

El cálculo del error para el sistema clasificador de fresas se realizó mediante pruebas sistemáticas, colocando fresas de diferentes estados de calidad en el prototipo. Para cada prueba se analizó un lote de 120 fresas, distribuidas en las cuatro categorías de calidad (Extra, Primera, Segunda y Desecho).

La determinación del error absoluto se realizó comparando los resultados del sistema automático de clasificación contra la clasificación manual, empleando la Ecuación (2):

$$\boldsymbol{Error\ absoluto = Fresas\ detectadas\ correctamente - Fresas\ colocadas} \quad (2)$$

El error relativo en el sistema de clasificación de fresas se calcula mediante la relación entre el error absoluto y el valor real, expresado en forma porcentual. Se emplea la Ecuación (3):

$$\boldsymbol{ER = \left|\frac{Valor\ Teorico - Valor\ medio}{Valor\ Teorico}\right| \times 100} \quad (3)$$

Al aplicar la fórmula del error relativo en el sistema de clasificación de fresas, se generó una tabla de resultados comparativos que se muestra en Tabla 34.

Tabla 34. Eficiencia del sistema

Grado de calidad	Fresas colocadas	Detectadas correctamente	Error Absoluto	Error Relativo	Error relativo porcentual %
Extra	21	19	2	0,0952381	9,52380952
Primera	39	31	8	0,20512821	20,5128205
Segunda	41	32	9	0,2195122	21,9512195
Desecho	19	16	3	0,15789474	15,7894737
Promedio					16,9443308

El análisis del error relativo porcentual reveló que el sistema de clasificación de fresas alcanzó diferentes niveles de eficiencia según la categoría: 90,47% para calidad Extra, 79,48% para Primera, 78,04% para Segunda y 84,21% para Desecho. Durante las pruebas se evaluaron 120 fresas distribuidas en cuatro categorías (21 Extra, 39

Primera, 41 Segunda y 19 Desecho), donde el sistema demostró su mejor rendimiento en la clasificación de fresas de Primera calidad, mientras que la categoría Segunda presentó la mayor tasa de error. La validación del sistema arrojó un error relativo porcentual promedio del 1%, lo que confirma un nivel aceptable de precisión en la clasificación automatizada de fresas.

b. Confiabilidad del prototipo

La confiabilidad del sistema de clasificación de fresas se establece mediante el promedio ponderado de los errores porcentuales individuales obtenidos en cada categoría, utilizando la Ecuación (4).

$$\boldsymbol{Confiabilidad = 100\% - \text{error promedio del sistema}} \quad (4)$$

$$Confiabilidad = 100\% - 16{,}94\%$$

$$Confiabilidad = 83{,}05\%$$

La confiabilidad del prototipo se fundamenta en el desempeño observado durante la fase de entrenamiento y la precisión alcanzada por el algoritmo de visión artificial YOLOv11l implementado para la clasificación de calidad en fresas. La validación del algoritmo demuestra que el sistema alcanza una tasa de precisión del 83,05%, lo que confirma que el prototipo es confiable para su aplicación. Este nivel de precisión se logró mediante la implementación de técnicas avanzadas de visión artificial, respaldando su efectividad como herramienta para la clasificación automatizada de fresas según sus diferentes grados de calidad (Extra, Primera, Segunda y Desecho), lo que lo convierte en un sistema viable para su aplicación en el sector agrícola.

Considerando los errores relativos porcentuales de 9,52% para calidad Extra, 20.51% para Primera, 21,95% para Segunda y 15,78% para Desecho, se obtiene un error promedio del 16,94%. Por lo tanto, la confiabilidad global del sistema se determina en 83,05% (100% - 16,94%) lo que indica que el sistema de visión artificial mantiene un nivel aceptable de precisión en la clasificación automática de fresas según sus características de calidad, demostrando ser una herramienta efectiva para la automatización del proceso de clasificación.

3.8 Presupuesto

El presupuesto sistema de clasificación automático de fresas según su calidad se determinó tomando en cuenta los costos de diseño y fabricación. En el proceso de diseño, se consideran las horas invertidas en la fabricación y el salario de un ingeniero en telecomunicaciones, cuyo salario base, según el Ministerio de Trabajo de Ecuador, es de 925 dólares. Para este cálculo, se utilizan 21 días laborables para establecer el salario diario con la Ecuación (5).

$$\boldsymbol{Salario_{dia}} = \frac{\boldsymbol{Salario_{mes}}}{\boldsymbol{Dias_{laborales}}} \tag{5}$$

$$Salario_{dia} = \frac{925}{21} = 44.04\$$$

Para determinar el salario por hora se toma a consideración la jornada laboral por día que son 8 horas con la Ecuación (6).

$$\boldsymbol{Salario_{xhora}} = \frac{\boldsymbol{Salario_{dia}}}{\boldsymbol{Horas_{laborales}}} \tag{6}$$

$$Salario_{xhora} = \frac{44.04}{8} = 5.50\ \$$$

El costo total del diseño se calcula con la Ecuación (7) asumiendo 170 horas de trabajo destinadas al estudios, diseño, construcción e implementación del sistema. El valor por hora de trabajo es de $5.50.

$$\boldsymbol{Presupuesto_{diseño}} = \boldsymbol{Horas_{ximplementacion}} \times \boldsymbol{Salario_{xhora}} \tag{7}$$

$$Presupuesto_{diseño} = 170 \times 5.50 = 935\ \$$$

En la Tabla 35, se detalla los costos relacionados para la implementación del sistema.

Tabla 35. Costos de la elaboración del prototipo

Recursos	**Cantidad**	**Precio Unitario**	**Precio Total**
Arduino Mega	1	$30	$30
Servomotores	6	$12	$72
Pulsadores	2	$1	$2
Indicadores Led	3	$0.50	$2
Fuente de poder	1	$20	$20
Materiales de construcción	1	$100	$100
Subtotal			$226
IVA 15%			$34
Imprevistos 10%			$23
Total			$283

El presupuesto total para la implementación del sistema es:

$$\boldsymbol{Presupuesto_{total}} = \boldsymbol{Presupuesto_{diseño}} + \boldsymbol{Presupuesto_{implementacion}} \quad \textbf{(8)}$$

$$Presupuesto_{total} = 935 + 283 = 1218$$

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La identificación precisa de las características que determinan la calidad de la fresa, como el color y el tamaño, es fundamental para el desarrollo de un sistema de clasificación efectivo. A través del análisis de la producción de fresa en Ecuador, se ha evidenciado que las fresas se cultivan principalmente en la región Sierra, en provincias como Pichincha, Imbabura, Cotopaxi y Tungurahua, donde las condiciones climáticas y de altitud favorecen el desarrollo de fresas de alta calidad. La clasificación determinada de los cuatro tipos: Extra, Primera, Segunda y Desecho, se realizó en base a los resultados obtenidos tras analizar la clasificación de fresas según la norma NMX-FF-062-SCFI-2002, así como la clasificación del Instituto Tecnológico Superior de Coalcomán y del CENICAFE. De esta manera, se establecieron las características de las fresas que el sistema de clasificación considera, siendo las más relevantes el color y, en segundo plano, el tamaño.

- La revisión de las técnicas actuales de Inteligencia Artificial aplicadas en cultivos revela un amplio espectro de métodos, que van desde el aprendizaje automático hasta el procesamiento de imágenes. Al analizar las técnicas de visión artificial empleadas en la clasificación de frutas, se observó una variedad de enfoques y resultados derivados de la implementación de diversos modelos, como YOLOv5x, OpenCV, YOLOv4, YOLOv8 y YOLOv11. El mejor resultado se logró con la implementación del modelo YOLOv11, alcanzando una precisión (P) del 84,4% con una tasa de y una precisión media (mAP) elevada del 81,6% con mejoras mAP de 3.8%, 3.2% y 5.5% en comparación con YOLOv8s, YOLOv9s y YOLOv10s. En este contexto, se concluyó que YOLOv11l es la versión más adecuada para la elaboración del proyecto, principalmente por ser la más ligera, rápida y de bajo consumo de recursos. Además, presenta mejoras significativas en la generalización de modelos, lo que facilita su aplicación en una amplia gama de tareas, desde la clasificación

de imágenes hasta la segmentación de objetos, adaptándose así a proyectos complejos y de gran escala.

- La implementación de un algoritmo de procesamiento de imágenes resultó efectiva para clasificar fresas según su calidad, automatizando el proceso y reduciendo el tiempo y esfuerzo frente a los métodos manuales. Se utilizó el modelo YOLOv11l, entrenado con 1429 imágenes durante 100 épocas, logrando mejoras significativas. Las métricas evidenciaron una disminución constante de la pérdida de entrenamiento, indicando un aprendizaje progresivo. La precisión aumentó de 0.818214 a 0.919880, y el recall de 0.858627 a 0.976090, reflejando una menor tasa de falsos positivos y negativos. El modelo alcanzó un mAP50 de 0.947310, superando a los modelos entrenados por 75 (0.945230) y 50 épocas (0.944040). Además, mantuvo un mAP50-95 estable de 0.877280, siendo ligeramente superior o igual a los otros modelos. En conclusión, el modelo de 100 épocas demostró mejor capacidad de detección y clasificación, siendo el más adecuado por ofrecer mayor confiabilidad en las predicciones.

- La implementación de un algoritmo en Python, que utiliza el modelo entrenado en YOLOv11l, ha permitido procesar imágenes para clasificar las fresas en cuatro categorías: Extra, Primera, Segunda y Desecho. Los datos generados son enviados a un algoritmo en Arduino, el cual proporciona las instrucciones necesarias a los actuadores para realizar la clasificación física de las fresas. El análisis del error relativo porcentual ha revelado que el sistema alcanza diferentes niveles de eficiencia, con un 90,52% para calidad Extra, 79,48% para Primera, 78,04% para Segunda y 84,21% para Desecho. Esto determina una confiabilidad global del 83,05%, lo que respalda la efectividad del sistema como herramienta para la clasificación automatizada de fresas. A diferencia del método tradicional, que depende del juicio humano y puede estar sujeto a fatiga, errores subjetivos o variabilidad en los criterios de calidad, este sistema ofrece una evaluación más objetiva, constante y rápida. Además, reduce la necesidad de intervención manual directa, lo cual mejora la productividad y permite estandarizar el proceso en contextos agrícolas a mayor escala.

4.2 Recomendaciones

- Es fundamental llevar a cabo una investigación exhaustiva de las características que determinan la calidad de las fresas, ya que este es un paso crucial en el desarrollo de un sistema de clasificación automático. Este proceso implica un análisis detallado de diversos factores que influyen en la percepción de calidad. Entre los aspectos más relevantes se encuentran el color, que indica diferentes grados de madurez; el tamaño, que se refiere al diámetro y peso de la fruta, siendo las fresas más grandes generalmente preferidas. Esta evaluación resulta indispensable para la implementación del sistema de clasificación, ya que es en esta etapa donde se definirán los parámetros que se analizarán mediante inteligencia artificial para determinar a qué categoría de calidad pertenece cada fresa.

- Se sugiere contar con una base de datos robusta que incluya al menos 1000 imágenes de fresas, las cuales se utilizarán para el entrenamiento del modelo. Esta cantidad de datos es esencial para garantizar una precisión rápida y efectiva en la clasificación, ya que un conjunto de datos más amplio permite al modelo aprender de manera más variada y generalizar mejor a nuevas imágenes. Además, es crucial evaluar y seleccionar el modelo de inteligencia artificial más adecuado para el entrenamiento, teniendo en cuenta las características específicas que se desean detectar en las imágenes, como color, tamaño y defectos visibles. También es importante considerar las épocas de entrenamiento; un número inadecuado de épocas puede llevar a resultados no favorables, como el sobreajuste o el subajuste del modelo. Por lo tanto, se recomienda realizar pruebas y ajustes en el número de épocas para encontrar el equilibrio óptimo que maximice la efectividad del sistema de clasificación.

- Se sugiere utilizar tiempos adecuados e instrucciones precisas para la comunicación serial, que se empleará para enviar los datos obtenidos sobre la calidad de las fresas al microcontrolador. Es crucial establecer tiempos de espera y de actuación apropiados para garantizar que la clasificación se realice de manera correcta y eficiente. Un manejo adecuado de los tiempos permite que el microcontrolador procese la información de forma óptima, evitando

pérdidas de datos y asegurando que cada clasificación se ejecute en el momento adecuado. De esta manera, se mejora la sincronización entre la adquisición de datos y la respuesta del sistema, lo que es esencial para el éxito del proceso de clasificación automatizada.

- Se sugiere realizar varias pruebas del funcionamiento del sistema para identificar errores y determinar la eficiencia general del mismo. Estas pruebas son esenciales para evaluar cómo se está llevando a cabo la clasificación de las fresas y para detectar posibles fallos en el proceso. Al analizar los resultados de estas pruebas, se pueden identificar áreas de mejora que permitan optimizar el sistema y lograr una clasificación más precisa y efectiva. Implementar un ciclo de pruebas y ajustes no solo ayuda a corregir errores existentes, sino que también proporciona información valiosa para futuras actualizaciones y refinamientos del sistema, asegurando así un rendimiento óptimo en la clasificación de fresas.

REFERENCIAS BIBLIOGRÁFICAS

[1] OEC, «Fresas "frutillas", frescas,» 2022. [En línea]. Available: https://oec.world/es/profile/hs/strawberries-fresh. [Último acceso: 3 Enero 2024].

[2] W. Verdugo, «Introducción de dos variedades de fresa (Fragaria vesca) y tecnica de fertirrigación empleando cuatro biofertilizantes liquidos en Pablo Sexto– Morona Santiago,» 2011. [En línea]. Available: http://repositorio.uta.edu.ec/bitstream/123456789/1677/1/tesis004%20Gesti%C3%B3n%20de%20la%20prod.%20de%20flores%20y%20Frut.pdf. [Último acceso: 10 Enero 2025].

[3] AME, «Tisaleo, principal productor de mora y fresa,» 2022. [En línea]. Available: https://ame.gob.ec/2021/05/14/tisaleo-principal-productor-de-mora-y-fresa/#:~:text=El%20cant%C3%B3n%20Tisaleo%20es%20un,y%20fresas%20con%20marca%20tungurahuense.. [Último acceso: 11 Enero 2024].

[4] S. B. PROAÑO QUISHPE, «ANÁLISIS DEL MANEJO POSCOSECHA DE LA FRESA Fragaria ananassa EN LOS CULTIVOS DE LA COMUNIDAD DE PAQUIESTANCIA, CANTÓN CAYAMBE,» 2022. [En línea]. Available: https://repositorio.utn.edu.ec/bitstream/123456789/11947/2/03%20EIA%20540%20TRABAJO%20GRADO.pdf. [Último acceso: 30 Enero 2025].

[5] S. W. Sidehabi, A. Suyuti, I. S. Areni y I. Nurtanio, «The Development of Machine Vision System for Sorting Passion Fruit using Multi-Class Support Vector Machine.,» *Journal of Engineering Science & Technology Review,* vol. 11, 2018.

[6] P. Neira, «Implementación de una estación prototipo con visión artificial, aplicado a la agricultura de precisión,» 2019. [En línea]. Available: https://dspace.ucuenca.edu.ec/items/aebb7ae3-2481-4669-9186-37d95b560125. [Último acceso: 16 Enero 2025].

[7] D. O. Orbes Padilla, «Implementación de un sistema de detección por visión artificial en la etapa de recolección del cultivo de fresas,» 2022. [En línea]. Available: https://repositorio.utn.edu.ec/handle/123456789/12524. [Último acceso: 18 Enero 2025].

[8] L. A. Gutiérrez Sánchez, «Sistema inteligente de selección de cosecha en cultivo de pitahaya mediante visión artificial,» 2024. [En línea]. Available: https://repositorio.uta.edu.ec/items/8e48eeb6-edaa-4774-b4e0-d6953f3bbb3e. [Último acceso: 20 Enero 2025].

[9] J. A. Chisaguano Tapia, «Sistema de visión artificial para el control de thrips en rosas de exportación,» 2024. [En línea]. Available: https://repositorio.uta.edu.ec/items/446ef570-2fc5-4425-9803-69cd20312994. [Último acceso: 22 Enero 2025].

[10] G. E. Ibarra, «Agricultura de Precisión: La integración de las TIC en la producción Agrícola,» *Journal of computer and electronic science, theory and applications,* vol. 3, p. 34–38, 2022.

[11] Q. Zhang, M. Wang y Y. Zhang, «Precision agriculture: A worldwide overview,» *Journal of Agricultural Science,* vol. 8, pp. 1-10, 2016.

[12] E. García y F. Flego, «Agricultura de precisión,» *Revista Ciencia y Tecnología,* p. 99–116, 2008.

[13] A. T. Balafoutis, B. Beck, S. Fountas, Z. Tsiropoulos, J. Vangeyte, T. van der Wal y S. M. Pedersen, «Smart farming technologies–Description, taxonomy and economic impact,» *Precision Agriculture: Technology and Economic Perspectives,* p. 21–77, 2017.

[14] S. Fountas, N. Mylonas, I. Malounas, E. Rodias, C. H. Santos y E. Pekkeriet, «Agricultural robotics for field operations,» *Sensors,* vol. 20, p. 2672, 2020.

[15] M. Paustian y L. Theuvsen, «Adoption of precision agriculture technologies by German crop farmers,» *Precision Agriculture,* vol. 18, p. 701–716, 2017.

[16] B. Zhang, W. Huang, J. Li, C. Zhao, S. Fan, J. Wu y C. Liu, «Principles, developments and applications of computer vision for external quality inspection of fruits and vegetables: A review,» *Food Research International,* vol. 62, p. 326–343, 2018.

[17] X. Chen, Y. Xun, W. Li y J. Zhang, «Smart agricultural automation and robotics: A review,» *Computers and Electronics in Agriculture,* vol. 157, p. 21–39, 2020.

[18] TopModel, «MS2810MG,» 2023. [En línea]. Available: https://www.topmodelcz.cz/index.php?&desktop_back=eshop&action_back=zbozi_detail&id_back=6994&desktop=eshop&action=zbozi_detail&id=94838. [Último acceso: 5 Enero 2025].

[19] TowerPro, «Servos & Parts – Tower Pro,» 2024. [En línea]. Available: https://towerpro.com.tw/product-category/servos-parts/. [Último acceso: 1 Enero 2025].

[20] Arduino, «Arduino Mega 2560,» 2024. [En línea]. Available: https://store.arduino.cc/products/arduino-mega-2560-rev3?srsltid=AfmBOooKY-bVbbkuUY7M6FN4j5PTOF-u78mvZVQyDifHLGJYrytrs1gI. [Último acceso: 2 Febrero 2025].

[21] Inc., Microchip Technology, «PIC16F877A,» 2024. [En línea]. Available: https://www.microchip.com/en-us/product/pic16f877a. [Último acceso: 5 Febrero 2025].

[22] Systems, Espressif, «ESP32,» 2023. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Último acceso: 5 Febrero 2025].

[23] G. M. Darrow, The Strawberry: History, Breeding and Physiology, vol. 1, New York: Rinehart and Winston, 1996.

[24] SAGPYA, «Reglamento Técnico del Mercado Común del Sur (MERCOSUR) para la Fijación de Identidad y Calidad de,» *Georgia Institute of Technology,* vol. vol. Resolución 85/98, 1998.

[25] SENA, «Manual Tecnico del Cultivo de Fresa Bajo Buenas Practicas Agricolas».

[26] Ministerio de Economía de Mexico, «NMX-FF-062-SCFI-2002: Especificaciones para la producción, comercialización y uso de fertilizantes,» 2002. [En línea]. Available: http://www.economia-nmx.gob.mx/normas/nmx/2002/nmx-ff-062-scfi-2002.pdf. [Último acceso: 14 Mayo 2025].

[27] Coalcomán, Instituto Tecnológico Superior, «Manual para cultivo de fresa en Coalcomán,» n.d.. [En línea]. Available: https://www.itscoalcoman.edu.mx/content/descargas/vinculacion/MANUAL%20PARA%20CULTIVO%20DE%20FRESA%20EN%20COALCOMAN.pdf. [Último acceso: 9 Febrero 2025].

[28] C. Gomez, «CENICAFE,» n.d.. [En línea]. Available: https://biblioteca.cenicafe.org/bitstream/10778/828/16/Fresa%20variedad%20Chandler.pdf. [Último acceso: 20 Febrero 2025].

[29] Euroinnova, «Ramas de la inteligencia artificial,» 2024. [En línea]. Available: https://www.euroinnova.com/blog/ramas-de-la-inteligencia-artificial. [Último acceso: 3 Marzo 2025].

[30] J. H. Sanabria Muñoz, «Ramas de la inteligencia artificial,» *Diplomado en Aplicación de Inteligencia Artificial,* 2023.

[31] L. Rouhiainen, «Inteligencia artificial,» *Madrid: Alienta Editorial,* p. 20–21, 2018.

[32] J. DÍAZ-RAMÍREZ, «Aprendizaje automático y aprendizaje profundo,» *Ingeniare. Revista chilena de ingeniería,* vol. 29, nº 2, pp. 180-181, 2021.

[33] Alex Krizhevsky y G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks,» *Communications of the ACM,* vol. 60, p. 84–90, 2017.

[34] I. Goodfellow, Y. Bengio y A. Courville, Deep Learning, MIT Press, 2016.

[35] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» *Nature,* vol. 521, p. 436–444, 2015.

[36] Pacioni, Elia y F. Fernández de Vega, «EduCode: herramienta de soporte al aprendizaje de la programación,» 2023.

[37] Canesche, Michael y e. al, «"Google Colab cad4u: Laboratorios prácticos en la nube para diseño digital",» *Simposio Internacional IEEE sobre Circuitos y Sistemas (ISCAS) de 2021,* 2021.

[38] Kaggle, «Kaggle Notebooks Documentation,» Kaggle, 2024. [En línea]. Available: https://www.kaggle.com/docs/notebooks. [Último acceso: 15 Marzo 2025].

[39] R. Szeliski, «Computer Vision: Algorithms and Applications,» *Springer,* 2010.

[40] R. C. Gonzalez y R. E. Woods, Digital Image Processing, 4th ed., London: Pearson, 2018.

[41] D. Forsyth y J. Ponce, Computer Vision: A Modern Approach, Prentice Hall, 2002.

[42] V. Sze, Y. Chen, T. J. Yang y J. Emer, «Efficient Processing of Deep Neural Networks: A Tutorial and Survey,» *Proceedings of the IEEE,* vol. 105, p. 2295–2329, 2017.

[43] M. A. Khan y M. Alshahrani, «A Survey on Edge Computing: Challenges, Opportunities, and Future Directions,» *IEEE Access,* vol. 8, p. 123456–123478, 2020.

[44] Y. Zhang y Y. Wang, «The Role of Edge Computing in Artificial Intelligence Applications,» *Journal of Computer Networks and Communications,* vol. 2021, p. 1–10, 2021.

[45] E. Upton y H. Gareth, uía del usuario de Raspberry Pi, John Wiley & Sons, 2016.

[46] S. Mittal, «Un estudio sobre la implementación optimizada de modelos de aprendizaje profundo en la plataforma Nvidia Jetson,» *Journal of Systems Architecture 97,* pp. 428-442, 2019.

[47] MathWorks, «Procesamiento de imágenes y Visión artificial,» 2024. [En línea]. Available: https://la.mathworks.com/solutions/image-video-processing.html. [Último acceso: 2024].

[48] O. A. Soto-Orozco, A. D. Corral-Sáenz, C. E. Rojo-González y J. A. Ramírez-Quintana, «Análisis del desempeño de redes neuronales profundas para segmentación semántica en hardware limitado,» *ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica,* vol. 8, 2019.

[49] Z.-Q. Zhao, P. Zheng, S.-T. Xu y X. Wu, «Object Detection with Deep Learning: A Review,» *IEEE Transactions on Neural Networks and Learning Systems,* vol. 30, pp. 3212-3232, 2019.

[50] L. G. Cossio Montefinale, «Metodologías modernas de aprendizaje profundo para la detección de fruta,» 2023.

[51] R. Van, Guido y D. Fred L, Manual de la API de Python/C - Python 3, CreateSpace, 2009.

[52] Oracle, «Java SE at a Glance,,» Oracle, 2024. [En línea]. Available: https://www.oracle.com/java/technologies/java-se-glance.html. [Último acceso: 1 Abril 2025].

[53] Stroustrup, Bjarne, «Una visión general del lenguaje de programación C++,» *Manual de tecnología de objetos,* vol. 72.

[54] C. Shorten y T. M. Khoshgoftaar, «A survey on image data augmentation for deep learning,» *Journal of Big Data,* vol. 6, pp. 1-48, 2019.

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, V. Vanhoucke y I. Radosavovic, «Going Deeper with Convolutions,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1-9, 2015.

[56] Logitech, «Logitech HD Pro C920,» Logitech, 2024. [En línea]. Available: https://www.logitech.com/es-es/shop/p/c920-pro-hd-webcam. [Último acceso: 10 Abril 2025].

[57] Microsoft, «LifeCam HD-3000,» Microsoft Community, 2019. [En línea]. Available: https://answers.microsoft.com/es-es/windows/forum/all/lifecam-hd-3000/80390a32-fa32-4b8a-9cbd-61309bf28307. [Último acceso: 12 Abril 2025].

[58] Samsung, «Samsung Galaxy A10,» Samsung Latinoamérica, 2024. [En línea]. Available: https://www.samsung.com/latin/smartphones/galaxy-a/galaxy-a10-blue-32gb-sm-a105mzbktpa/. [Último acceso: 15 Abril 2025].

[59] Z. Zou, Z. Shi y Y. Guo, «A Survey on Image Data Annotation for Computer Vision Applications,» *Pattern Recognition,* vol. 96, p. 106954, 2019.

[60] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn y A. Zisserman, «The Pascal Visual Object Classes (VOC) Challenge,» *International Journal of Computer Vision,* vol. 88, p. 303–338, 2010.

[61] Tzutalin, «LabelImg,» Python Package Index (PyPI), 2024. [En línea]. Available: https://pypi.org/project/labelImg/. [Último acceso: 20 Abril 2025].

[62] Supervisely, «Supervisely – The platform for computer vision,» Supervisely, 2024. [En línea]. Available: https://supervisely.com/. [Último acceso: 25 Abril 2025].

[63] LabelMe, «LabelMe: Image Polygonal Annotation with Python,» LabelMe.io, 2024. [En línea]. Available: https://labelme.io/. [Último acceso: 30 Abril 2025].

[64] Roboflow, «Roboflow: Give your software the eyes it needs,» 2024. [En línea]. Available: https://roboflow.com/. [Último acceso: 2 Mayo 2025].

[65] R. A. L. León, C. S. B. Diaz y J. J. G. Palacios, *YOLOv6,* GitHub, 2023.

[66] P. N. Constante Procel y A. M. Gordon Garces, «Diseño e implementación de un sistema de visión artificial para clasificación de al menos tres tipos de frutas,» *Journal of Computer Vision and Image Processing,* vol. 15, p. 78–89, 2023.

[67] J. J. Camargo Robles, «Sistema Clasificador de Fresa Usando Técnicas de Inteligencia Artificial,» *International Journal of Artificial Intelligence and Machine Learning,* vol. 10, p. 22–34, 2023.

[68] L. A. García Pérez y A. S. Martínez López, «Clasificación Automática de Fresas Mediante Redes Neuronales Convolucionales,» *Journal of Applied Machine Learning,* vol. 8, p. 100–112, 2024.

[69] M. F. Rodríguez Gómez y C. E. Hernández Ruiz, «Sistema de Clasificación de Fresas Basado en Visión Artificial y Aprendizaje Profundo,» *Journal of Agricultural Robotics and Automation,* vol. 5, p. 15–27, 2024.

[70] R. Khanam y M. Hussain, «Yolov11: An overview of the key architectural enhancements,» *arXiv preprint arXiv:2410.17725,* 2024.

[71] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner, «Gradient-Based Learning Applied to Document Recognition,» *Proceedings of the IEEE,* vol. 86, p. 2278–2324, 1998.

[72] C. Szegedy, V. Vanhoucke, S. Ioffe, O. Vinyals y A. Alemi, «Rethinking the Inception Architecture for Computer Vision,» de *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[73] K. He, X. Zhang, S. Ren y J. Sun, «Deep Residual Learning for Image Recognition,» de *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[75] J. Redmon y A. Farhadi, «YOLOv3: An Incremental Improvement,» *arXiv preprint arXiv:1804.02767,* 2018.

[76] A. Vina, «Comparación de Ultralytics YOLO11 con los modelos YOLO anteriores,» Ultralytics Blog (en español), 2025. [En línea]. Available: https://www.ultralytics.com/es/blog/comparing-ultralytics-yolo11-vs-previous-yolo-models. [Último acceso: 10 Mayo 2025].

[77] A. F. Joseph Redmon, «You Only Look Once: Unified, Real-Time Object Detection,» 2023. [En línea]. Available: https://arxiv.org/abs/1506.02640. [Último acceso: 3 Abril 2025].

[78] P. Chapman, E. Clare, T. Khabaza, T. Reinartz, C. Shearer y R. Wirth, «CRISP-DM 1.0: Step-by-step Data Mining Guide,» 2000. [En línea]. Available:

https://www.the-modeling-agency.com/crisp-dm.pdf. [Último acceso: 23 Abril 2025].

[79] El Comercio, «La frutilla es un cultivo rentable,» 2023. [En línea]. Available: https://www.elcomercio.com/actualidad/negocios/frutilla-cultivo-rentable/. [Último acceso: 20 Mayo 2025].

[80] The Observatory of Economic Complexity (OEC), «Fresas, frescas en Ecuador,» 2023. [En línea]. Available: https://oec.world/es/profile/bilateral-product/strawberries/reporter/ecu/. [Último acceso: 1 Junio 2025].

ANEXOS

Anexo A. Diseño del prototipo clasificador de fresas según su calidad.

En la Figura A1, se muestra el diseño del prototipo parte por parte y como fue construido.

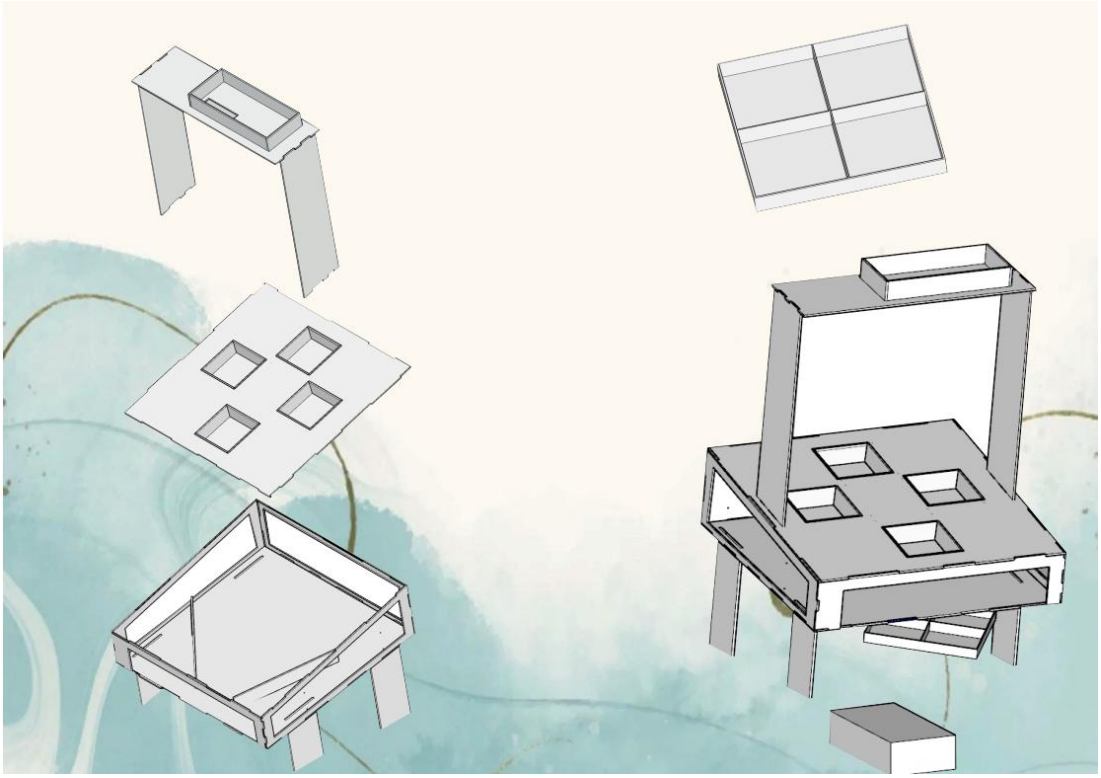


Figura A1. Diseño del prototipo clasificador

Anexo B. Laptop OMEN by HP 15-dh1xxx

En la Figura B1 se muestra las especificaciones técnicas del computador portátil utilizado en el sistema.

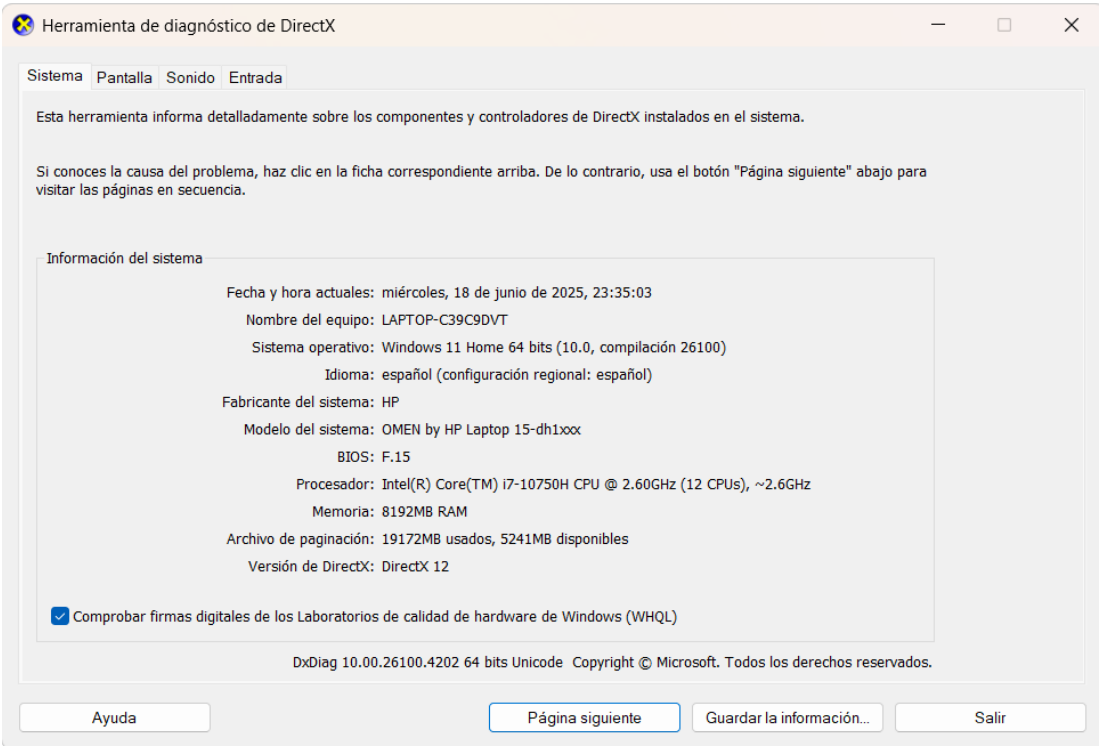


Figura B1. D Características de Laptop OMEN by HP 15-dh1xxx

Anexo C. NVIDIA GeForce GTX 1660 Ti

En la Figura C1 se muestra las especificaciones técnicas de la tarjeta gráfica integrada en la laptop que se va a ocupar en el sistema.

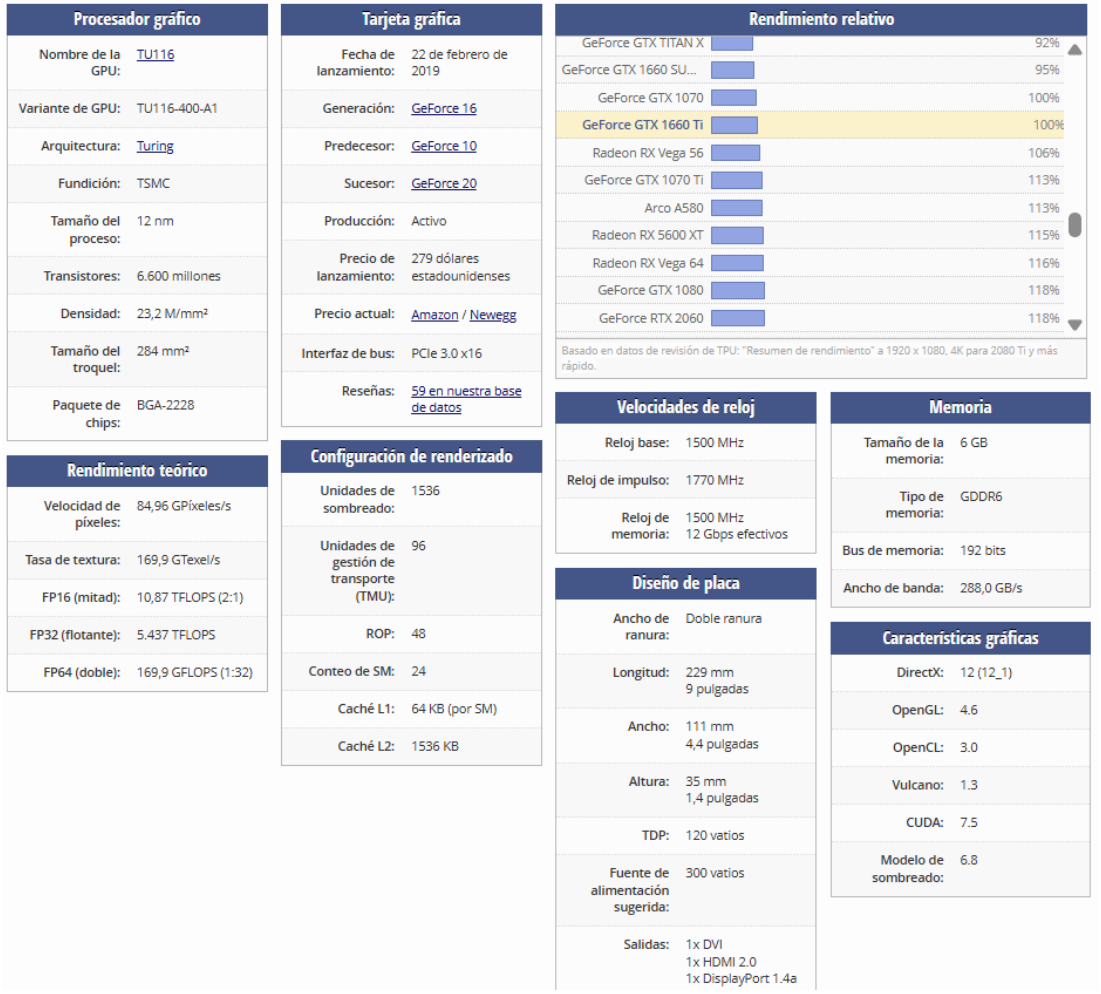


Figura C1. Características de la NVIDIA GeForce GTX 1660 Ti

Anexo D. Arduino Mega 2560

En la Figura D1 se muestra las especificaciones técnicas del microcontrolador usado en el sistema.

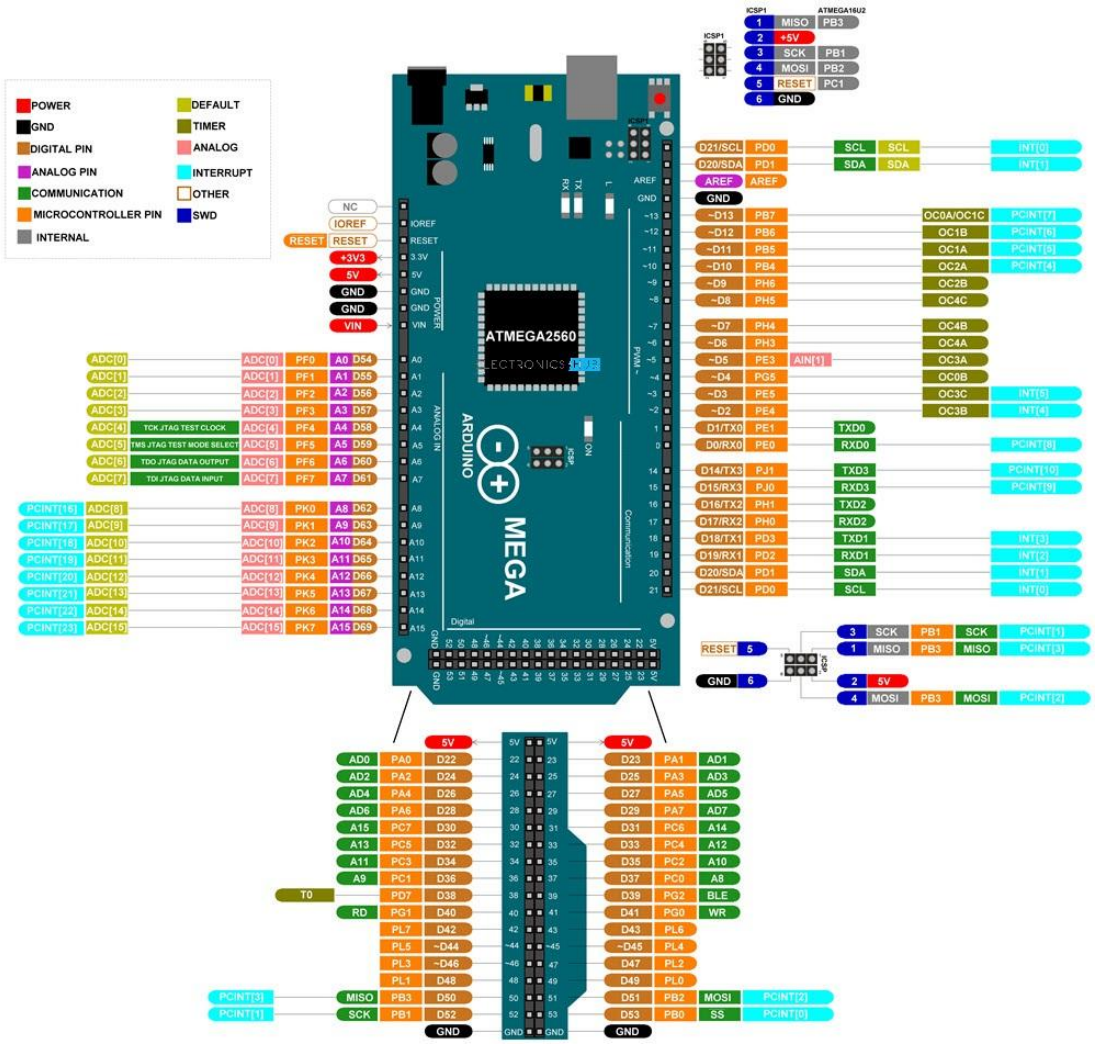


Figura D1. Características del Arduino Mega 2560

Anexo E. Algoritmo de clasificación en Python

```
import tkinter as tk
from PIL import Image, ImageTk
import os
import cv2
from ultralytics import YOLO
import serial
import time

class FruitClassificationGUI:
    def __init__(self, master):
        self.master = master
```

```
        master.title("Sistema clasificador de fresas según su calidad")
        # Cambiar el color de fondo a beige
        master.configure(bg="#FCFCFC")  # Beige
        # Cargar modelo de IA (YOLOv11l)
        self.model = self.load_model()
        # Inicializar la conexión serial
        self.serial_port = self.initialize_serial()
        # Crear el contenedor principal
        self.main_frame = tk.Frame(master, bg="#dbc483")  # Beige
        self.main_frame.place(x=0, y=0, width=900, height=700)  #
Posición y tamaño del contenedor principal
        # Crear el rectángulo en la parte superior para el título
        self.title_frame = tk.Frame(self.main_frame, bg="#006400")  #
Verde oscuro
        self.title_frame.place(x=0, y=0, width=900, height=50)  # (x=0,
y=0) - 900px de ancho, 50px de alto
        # Etiqueta del título
        self.title_label = tk.Label(self.title_frame, text="Sistema
clasificador de fresas según su calidad",
                                    font=("Arial", 16, "bold"),
bg="#006400", fg="white", padx=20, pady=10)
        self.title_label.pack(fill="both")  # Llenar el espacio
disponible
        # Crear la sección de imágenes
        self.image_frame = tk.Frame(self.main_frame, bg="#dbc483")  #
Beige
        self.image_frame.place(x=150, y=55, width=590, height=190)  #
(x=150, y=55) - 590px de ancho, 190px de alto
        # Cargar y mostrar la imagen de las fresas (inicial)
        self.image_path =
r"C:\Users\USUARIO\Desktop\Tesis\Prueba_Entrenamiento\interfazusar.png"
        self.image = Image.open(self.image_path)  # Abrir la imagen
        self.image_tkinter = ImageTk.PhotoImage(self.image)  #
Convertir a formato compatible con Tkinter
        self.image_label = tk.Label(self.image_frame,
image=self.image_tkinter)  # Crear etiqueta para la imagen
        self.image_label.pack(side="top", pady=10)  # Mostrar la imagen
en el marco
        # Crear el cuadrado para el video
        self.video_frame = tk.Frame(self.main_frame, width=450,
height=250, bg="gray")  # Ajustar altura
        self.video_frame.place(x=20, y=250, width=400, height=400)  #
(x=20, y=250) - 400px de ancho, 400px de alto
        # Etiqueta "Video en vivo" encima del cuadro de video
        self.video_label = tk.Label(self.video_frame, text="VIDEO
INGRESADO",
                                    font=("Arial", 12, "bold"),
bg="#006400", fg="white")  # Verde oscuro
```

```
        self.video_label.pack()  # Mostrar la etiqueta
        # Crear el cuadro para la imagen procesada
        self.processed_image_frame = tk.Frame(self.main_frame,
width=400, height=400, bg="gray")  # Cuadro para imagen procesada
        self.processed_image_frame.place(x=475, y=250, width=400,
height=400)  # (x=475, y=250) - 400px de ancho, 400px de alto
        # Etiqueta "Imagen procesada" encima del cuadro de imagen
        self.process_label = tk.Label(self.processed_image_frame,
text="IMAGEN PROCESADA",
                                      font=("Arial", 12, "bold"),
bg="#006400", fg="white")  # Verde oscuro
        self.process_label.pack()  # Mostrar la etiqueta
        # Etiqueta para mostrar la imagen procesada
        self.additional_image_label1 =
tk.Label(self.processed_image_frame)
        self.additional_image_label1.pack()  # Mostrar la etiqueta
        # Crear la sección de tabla
        self.table_frame = tk.Frame(self.main_frame, bg="#f5f5dc")  #
Beige
        self.table_frame.place(x=760, y=60, width=115, height=110)  #
(x=760, y=60) - 115px de ancho, 110px de alto
        # Crear la tabla
        self.table = tk.Frame(self.table_frame, bg="#f5f5dc")  # Beige
        self.table.pack(fill="both", expand=True)  # Llenar el espacio
disponible
        # Encabezado de la tabla (solo se agrega una vez)
        self.table_header = tk.Frame(self.table, bg="#f5f5dc")  # Beige
        self.table_header.pack(fill="x")  # Llenar el espacio
horizontal
        # Etiquetas del encabezado de la tabla
        tk.Label(self.table_header, text="ID", font=("Arial", 12,
"bold"), bg="#f5f5dc").pack(side="left", padx=10)
        tk.Label(self.table_header, text="Calidad", font=("Arial", 12,
"bold"), bg="#f5f5dc").pack(side="left", padx=10)
        # Crear un marco para los botones
        self.button_frame = tk.Frame(self.main_frame, bg="#dbc483")  #
Beige
        self.button_frame.place(relx=1.0, rely=1.0, anchor='se', x=-20,
y=-20)  # Posición en la esquina inferior derecha
        # Crear los botones
        self.start_button = tk.Button(self.button_frame,
text="Clasificar", command=self.start_classification)
        self.start_button.pack(side="left", padx=10)  # Botón para
iniciar la clasificación
        self.stop_button = tk.Button(self.button_frame, text="Detener",
command=self.stop_classification)
        self.stop_button.pack(side="left", padx=10)  # Botón para
detener la clasificación
```

```
        self.add_video_button = tk.Button(self.button_frame,
text="Agregar Video", command=self.add_video)
        self.add_video_button.pack(side="left", padx=10)  # Botón para
agregar un video
        # Inicializar variables
        self.classification_running = False  # Estado de la
clasificación
        self.video_capture = None  # Variable para la captura de video
        self.video_label = None  # Variable para la etiqueta del video
        self.video_path =
r"C:\Users\USUARIO\Desktop\Tesis\Prueba_Entrenamiento\video2.mp4"  #
Ruta del video
        self.video_width = 400  # Ancho del video
        self.video_height = 400  # Altura del video
        # Inicializar la tabla de resultados
        self.results_table = []  # Lista para almacenar los resultados
de la clasificación

    def load_model(self):
        # Cargar el modelo YOLOv11l
        model_path =
r"C:\Users\USUARIO\Desktop\Tesis\Prueba_Entrenamiento\best100.pt"
        model = YOLO(model_path)  # Cambiado a la nueva forma de cargar
el modelo
        return model

    def initialize_serial(self):
        # Inicializar la conexión serial
        try:
            ser = serial.Serial('COM4', 9600, timeout=1)  # Cambia
'COM4' al puerto correcto
            time.sleep(2)  # Esperar a que se establezca la conexión
            return ser
        except serial.SerialException as e:
            print(f"Error al abrir el puerto serial: {e}")
            return None

    def start_classification(self):
        # Iniciar el proceso de clasificación
        if self.video_capture is None or not
self.video_capture.isOpened():
            print("Video no cargado o no abierto.")
            return

        # Capturar un fotograma actual
        ret, frame = self.video_capture.read()
        if not ret:
            print("No se pudo capturar el fotograma del video.")
```

```
            return

        # Procesar con IA
        self.results_table.clear()  # Limpiar la tabla antes de cada
clasificación
        annotated_frame = self.process_frame_with_model(frame)

        # Convertir a imagen a mostrar en Tkinter
        image_tk =
ImageTk.PhotoImage(image=Image.fromarray(annotated_frame).resize((400,
400)))

        # Mostrar en el recuadro imagen para procesar
        self.additional_image_label1.configure(image=image_tk)
        self.additional_image_label1.image = image_tk

        # Actualizar la tabla con los resultados
        self.update_results_table()

        # Enviar datos por serial
        self.send_data_to_arduino()

        print("Clasificación y anotación completadas.")

    def process_frame_with_model(self, frame):
        # Detectar objetos en el fotograma
        results = self.model(frame)[0]
        # Lista para guardar objetos únicos
        detected_objects = []
        for i, box in enumerate(results.boxes):
            x1, y1, x2, y2 = map(int, box.xyxy[0])  # Coordenadas de la
caja delimitadora
            conf = float(box.conf[0])  # Confianza de la detección
            cls = int(box.cls[0])  # Clase del objeto detectado
            label = results.names[cls]  # Nombre de la clase
            center_x = (x1 + x2) // 2  # Coordenada X del centro
            center_y = (y1 + y2) // 2  # Coordenada Y del centro
            # Verificar si ya hay un objeto muy cerca (evita encimados)
            too_close = False
            for obj in detected_objects:
                ox, oy = obj[3]
                if abs(center_x - ox) < 30 and abs(center_y - oy) < 30:
                    too_close = True
                    break
            if not too_close:
                detected_objects.append((center_x, label, (x1, y1, x2,
y2), (center_x, center_y)))
```

```
            if len(detected_objects) >= 4:  # Limitar a 4 objetos
detectados
                break
        # Ordenar de izquierda a derecha
        detected_objects.sort(key=lambda x: x[0])
        # Dibujar las cajas y etiquetas en el fotograma
        for idx, (center_x, label, (x1, y1, x2, y2), _) in
enumerate(detected_objects, start=1):
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)  #
Dibujar la caja delimitadora
            cv2.putText(frame, f"{idx}: {label}", (x1, y1 - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 0, 0),
2)  # Dibujar la etiqueta
            # Agregar a la tabla de resultados
            self.results_table.append((idx, label))
        return frame

    def update_results_table(self):
        # Limpiar la tabla existente (sin eliminar los encabezados)
        for widget in self.table.winfo_children():
            if widget not in self.table_header.winfo_children():  # No
eliminar encabezados
                widget.destroy()

        # Agregar resultados a la tabla
        for id, label in self.results_table:
            row = tk.Frame(self.table, bg="#f5f5dc")  # Beige
            row.pack(fill="x")  # Llenar el espacio horizontal
            tk.Label(row, text=id, bg="#f5f5dc").pack(side="left",
padx=10)  # Mostrar ID
            tk.Label(row, text=label, bg="#f5f5dc").pack(side="left",
padx=10)  # Mostrar calidad

    def send_data_to_arduino(self):
        # Enviar datos de clasificación al Arduino
        if self.serial_port and self.results_table:
            # Convertir la tabla de resultados a un formato adecuado
para enviar
            data = ','.join([f"{id}:{label}" for id, label in
self.results_table])
            self.serial_port.write(data.encode('utf-8') + b'\n')  #
Enviar datos y agregar un salto de línea
            print(f"Enviando por serial: {data}")  # Imprimir los datos
enviados

    def stop_classification(self):
        # Detener la clasificación
        self.classification_running = False
```

```
        print("Clasificación de fresas detenida")

    def add_video(self):
        # Agregar video para la clasificación
        self.video_capture = cv2.VideoCapture(self.video_path)

        if not self.video_capture.isOpened():
            print("Error al abrir el video.")
            return

        self.video_capture.set(cv2.CAP_PROP_FRAME_WIDTH,
self.video_width)  # Establecer ancho del video
        self.video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT,
self.video_height)  # Establecer altura del video

        if self.video_label is None:
            self.video_label = tk.Label(self.video_frame)  # Crear
etiqueta para el video
            self.video_label.pack()

        self.play_video()  # Iniciar la reproducción del video

    def play_video(self):
        # Reproducir el video en el cuadro de video
        if self.video_capture is not None and self.video_label is not
None:
            ret, frame = self.video_capture.read()  # Leer un fotograma
del video

            if ret:
                frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)  #
Convertir BGR a RGB
                photo =
ImageTk.PhotoImage(image=Image.fromarray(frame_rgb).resize((self.video_
width, self.video_height)))

                self.video_label.configure(image=photo)  # Actualizar
la etiqueta del video
                self.video_label.image = photo  # Mantener una
referencia a la imagen

                self.current_frame = frame  # Guardar para usar en
clasificación

                self.video_label.after(33, self.play_video)  # Repetir
cada 33 ms
            else:
```

```
                self.video_capture.release()  # Liberar la captura de
video
                self.video_label.configure(image=None)  # Limpiar la
etiqueta del video
                self.video_label.image = None  # Limpiar la referencia
a la imagen

if __name__ == "__main__":
    root = tk.Tk()  # Crear la ventana principal
    app = FruitClassificationGUI(root)  # Crear la aplicación
    root.mainloop()  # Iniciar el bucle principal de la interfaz
gráfica
```

Anexo F. Algoritmo de clasificación para el microcontrolador.

```
#include <Servo.h>

Servo servos[5];  // Ahora hay 5 servos

// Arrays para guardar los índices de servos por comando
int extraServos[4], primeraServos[4], segundaServos[4],
desechoServos[4];
int extraCount = 0, primeraCount = 0, segundaCount = 0, desechoCount
= 0;

// Último comando recibido
String lastCommand = "";

void setup() {
  Serial.begin(9600);
  servos[0].attach(3);
  servos[1].attach(5);
  servos[2].attach(6);
  servos[3].attach(9);
  servos[4].attach(10);  // Nuevo servo agregado
}

void loop() {
  if (Serial.available()) {
    String data = Serial.readStringUntil('\n');
    processData(data);
```

```
    activateGroup(extraServos, extraCount, "Extra");
    delay(3000);
    activateGroup(primeraServos, primeraCount, "Primera");
    delay(3000);
    activateGroup(segundaServos, segundaCount, "Segunda");
    delay(3000);
    activateGroup(desechoServos, desechoCount, "Desecho");

    // Reset counts
    extraCount = primeraCount = segundaCount = desechoCount = 0;
  }
}

void processData(String input) {
  input.trim();
  int start = 0;

  while (start < input.length()) {
    int comma = input.indexOf(',', start);
    if (comma == -1) comma = input.length();

    String pair = input.substring(start, comma);
    int colon = pair.indexOf(':');

    if (colon != -1) {
      int servoNum = pair.substring(0, colon).toInt();
      String command = pair.substring(colon + 1);
      command.trim();

      Serial.print("Recibido -> Servo ");
      Serial.print(servoNum);
      Serial.print(": '");
      Serial.print(command);
      Serial.println("'");

      if (servoNum >= 1 && servoNum <= 4) {
        int index = servoNum - 1;

        if (command == "Extra") extraServos[extraCount++] = index;
        else if (command == "Primera") primeraServos[primeraCount++]
= index;
        else if (command == "Segunda") segundaServos[segundaCount++]
= index;
        else if (command == "Desecho") desechoServos[desechoCount++]
= index;

        lastCommand = command;  // Guardar último comando
      }
```

```
    }

    start = comma + 1;
  }
}

void activateGroup(int group[], int count, String name) {
  Serial.print("Activando servos para comando: ");
  Serial.println(name);
  for (int i = 0; i < count; i++) {
    int index = group[i];
    Serial.print("  Servo ");
    Serial.println(index + 1);
    servos[index].write(180);
  }
  activateFifthServo(name);  // Activar el nuevo servo según la
calidad
  delay(1000);
  for (int i = 0; i < count; i++) {
    servos[group[i]].write(0);
  }
}
void activateFifthServo(String quality) {
  int angle = 0;

  if (quality == "Extra") angle = 0;
  else if (quality == "Primera") angle = 90;
  else if (quality == "Segunda") angle = 180;
  else if (quality == "Desecho") angle = 270;

  Serial.print("Servo 5 se moverá a: ");
  Serial.println(angle);

  servos[4].write(angle);
  delay(1000);
  servos[4].write(0);  // Regresa a posición inicial
}
```

Anexo G. Pruebas del sistema

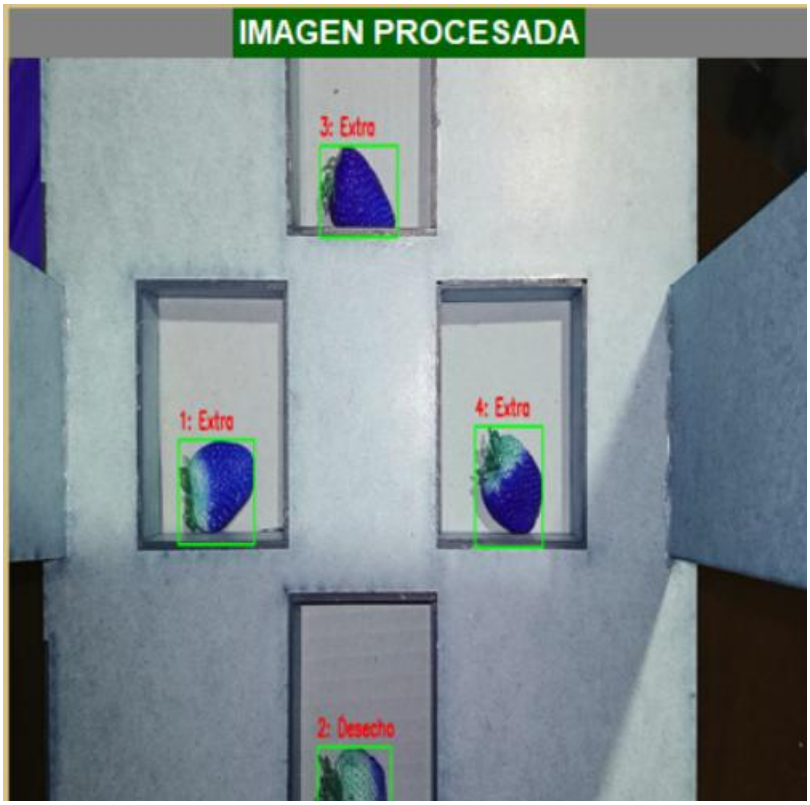


Figura G1. Resultado de las pruebas de funcionamiento