

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

SISTEMA DE MONITOREO Y CONTROL IOT PARA CULTIVOS AGRÍCOLAS BASADO EN LA ARQUITECTURA EDGE CLOUD Y DEEP LEARNING

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: Tecnologías de la información y Sistemas de control

AUTOR: Byron Alexander Amán Caiza

TUTOR: Ing. Víctor Santiago Manzano Villafuerte, Mg.

Ambato - Ecuador

febrero – 2025

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE MONITOREO Y CONTROL IOT PARA CULTIVOS AGRÍCOLAS BASADO EN LA ARQUITECTURA EDGE CLOUD Y DEEP LEARNING, desarrollado bajo la modalidad Proyecto de Investigación por el señor Byron Alexander Amán Caiza, 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, febrero 2025.

Firmado electrónicamente por:
VICTOR SANTIAGO
MANZANO VILLAFUERTE

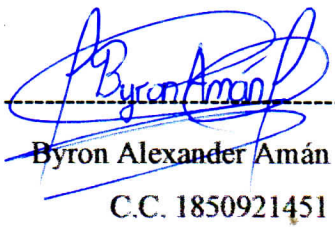
Ing. Víctor Santiago Manzano Villafuerte, Mg.

TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA DE MONITOREO Y CONTROL IOT PARA CULTIVOS AGRÍCOLAS BASADO EN LA ARQUITECTURA EDGE CLOUD Y DEEP LEARNING 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, febrero 2025



Byron Alexander Amán Caiza

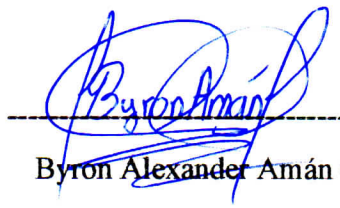
C.C. 1850921451

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, febrero 2025.



Byron Alexander Amán Caiza

C.C. 1850921451

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Byron Alexander Amán Caiza, 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 MONITOREO Y CONTROL IOT PARA CULTIVOS AGRÍCOLAS BASADO EN LA ARQUITECTURA EDGE CLOUD Y DEEP LEARNING, 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, febrero 2025.



Firmado electrónicamente por:
FRANKLIN OSWALDO MAYORGA MAYORGA

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



Firmado electrónicamente por:
FABIAN RODRIGO SALAZAR ESCOBAR

Ing. Fabián Salazar Escobar, PhD.
PROFESOR CALIFICADOR



Firmado electrónicamente por:
MARTHA ESPERANZA SEVILLA ABARCA

Ing. Martha Sevilla Abarca, PhD.
PROFESOR CALIFICADOR

DEDICATORIA

Dedico el presente proyecto de investigación, en primer lugar, a Dios, quien ha sido mi pilar fundamental y mi fortaleza, que me ha permitido superar los momentos más difíciles de mi vida, brindándome protección, sabiduría y permitiéndome llegar hasta este punto de mi vida.

A mis padres, Byron y Rosa, quienes han sido mi apoyo incondicional en todo momento. Mi madre, Rosa, con su amor y dedicación, siempre ha buscado lo mejor para nuestra familia, mientras que mi padre, Byron, con sus consejos y sabiduría, ha sido una fuente constante de orientación y ánimo. Este trabajo de investigación representa una pequeña muestra de mi infinito agradecimiento hacia ellos.

A mi hermana Shirley, por ser la que me ha acompañado en cada etapa de este camino. Su motivación y apoyo constante han sido fundamentales para que pueda continuar y alcanzar mis metas.

A mi abuelita Rosa que a pesar de que ya no esté presente, se le siente igual conmigo y se agradece por todas las cosas que se hizo mientras estaba presente.

Este logro no solo es mío, es de todos los que estuvieron presentes en todo momento, de corazón les digo muchas gracias.

Con cariño Byron Amán.

AGRADECIMIENTO

Agradezco a Dios por tener aquí presente, por la salud permitiendo cumplir todas las metas propuestas.

A mis padres y mi hermana querida que me han apoyado en todo momento, gracias por iluminarme en todo momento.

A mi familia y mi padrino el Ingeniero Patricio Ortega que me han aconsejado, me han expresado el cariño y me han aconsejado en todo momento.

Al Club de Robótica INTROB que me abrió las puertas y me permitió vivir nuevas experiencias.

A mis amigos Danny, Ibeth, Vane, Lizbeth, Faraón, Kevin, María, me han apoyado en el transcurso de la carrera, a Evelyn por motivarme en los momentos más difíciles, a Melanie con quien he pasado toda la vida universitaria, gracias por tantos momentos y tantas emociones, a Sayuri por ayudarme, a Alex, Bryan, Frank, Jorge, Josué que cooperaron en la tesis, a Angy por acompañarme y mis amigos de colegio Daniel y María gracias por estar conmigo.

A mi grupito: Darío, Flavio y Víctor, gracias por estar en todo momento y caminar conmigo, su amistad ha sido fundamental para lograr todo.

Al ingeniero Diego Bustamante por abrirme las puertas y apoyarme en todo el proceso del trabajo y la ingeniera Jessica Arévalo por ayudarme asesorando en las imágenes del proyecto.

A la ingeniera Pamela Castro que ha sido la docente que me ha aportado bastante conocimiento y confió en mí.

Y a mi tutor el ingeniero Santiago Manzano que me brindó la confianza y me apoyó en todo momento para realizar este trabajo.

Muchas gracias, Byron Amán.

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 ..... **xi**

ÍNDICE DE FIGURAS ..... **xiii**

ÍNDICE DE ANEXOS ..... **xvii**

RESUMEN EJECUTIVO ..... **xviii**

ABSTRACT ..... **xix**

CAPÍTULO I. MARCO TEÓRICO **1**

1.1 Tema de investigación 1

1.1.1 Planteamiento del problema 1

1.2 Antecedentes investigativos 4

1.3 Fundamentación teórica 7

1.3.1 Agricultura de precisión 7

1.3.2 Cultivos hidropónicos 10

1.3.3 IoT en la Agricultura 17

1.3.4 Edge Cloud 30

1.3.5 Inteligencia artificial 38

1.3.6 Deep learning 40

1.4 Objetivos 51

1.4.1 Objetivo general 51

1.4.2 Objetivos específicos 51

CAPÍTULO II. METODOLOGÍA 52

2.1 Materiales 52

2.2 Métodos 52

2.2.1 Modalidad de la investigación 52

2.2.2 Recolección de información 54

2.2.3 Procesamiento y análisis de datos 54

CAPÍTULO III. RESULTADOS Y DISCUSIÓN 55

3.1 Arquitectura del sistema 55

3.2 Selección de tecnología 59

3.3 Implementación de Edge Layer 59

3.3.1 Diseño de circuito electrónico 60

3.4 Implementación de Server layer 70

3.4.1 Control de variables 70

3.4.2 Entrenamiento del modelo de inteligencia artificial 74

3.4.3 Implementación del entrenamiento 83

3.5 Implementación de Cloud layer y Display layer 85

3.6 Validación de resultados 87

3.7 Discusión 93

3.8 Costos 94

3.8.1 Insumos del proyecto 94

3.8.2 Costo del desarrollo técnico 95

3.8.3 Costo total 96

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES 97

4.1 Conclusiones 97

4.2 Recomendaciones 98

REFERENCIAS BIBLIOGRÁFICAS 100

ANEXOS 110

ÍNDICE DE TABLAS

Tabla 1. Tipos de lechugas existentes en Ecuador 12

Tabla 2. Principales enfermedades causadas por hongos. 15

Tabla 3. Principales enfermedades causadas por bacterias 16

Tabla 4. Principales enfermedades causadas por virus 17

Tabla 5. Impacto de IoT en la agricultura 20

Tabla 6. Características de IoT en la agricultura. 24

Tabla 7. Sensores de conductividad eléctrica. 26

Tabla 8. Sensores para pH. 26

Tabla 9. Motores a pasos. 26

Tabla 10. Driver de control de motores. 27

Tabla 11. Microcontroladores 27

Tabla 12. Microprocesadores 27

Tabla 13. Cámaras 28

Tabla 14. Lenguajes de programación para el microcontrolador. 28

Tabla 15. Comparativas de protocolos de comunicación 29

Tabla 16. Arquitecturas existentes 30

Tabla 17. Impacto de Edge Cloud en la Arquitectura. 34

Tabla 18. Características Edge Cloud en la agricultura 37

Tabla 19. Comparativa de bases de datos. 38

Tabla 20. Impacto de Deep Learning en la Agricultura. 42

Tabla 21. Características de Deep Learning. 44

Tabla 22. Metodologías..... 45

Tabla 23. Lenguajes de programación para inteligencia artificial 45

Tabla 24. Comparativa de Redes Neuronales Convolucionales 48

Tabla 25. Comparativa de versiones de YOLO 49

Tabla 26. Criterios de recolección de información. 54

Tabla 27. Características del Dataset. 74

Tabla 28. Resultados con 40 épocas. 78

Tabla 29. Resultado con 60 épocas. 78

Tabla 30. Resultado con 94 épocas. 79

Tabla 31. Datos de la matriz de confusión. 82

Tabla 32. Comparativa del sensor pH 89

Tabla 33. Comparativa del sensor de ce..... 90

Tabla 34. Detección mediante imágenes. 92

Tabla 35. Costo de insumos. 95

Tabla 36. Costo en relación con las horas laborales ocupadas. 95

ÍNDICE DE FIGURAS

Figura 1. Cultivos hidropónicos 11

Figura 2. Parámetros de control para una solución nutritiva equilibrada. 14

Figura 3. Dispositivos IoT 21

Figura 4. Ciclo de vida de la agricultura inteligente basada en IoT 21

Figura 5. Resultados de la agricultura inteligente 23

Figura 6. Red de área agrícola basada en IoT 23

Figura 7. Cadena de datos de aplicaciones de Big Data 24

Figura 8. Arquitectura de tres capas para agricultura inteligente. 35

Figura 9. Topología del sistema de red 36

Figura 10. Dispositivos y nodos de borde en relación con la nube 37

Figura 11. Tipos de aprendizaje automático. 39

Figura 12. Modelo AlexNet 43

Figura 13. Tecnologías de Deep learning 47

Figura 14. Arquitectura YOLOv8. 50

Figura 15. Arquitectura general del sistema. 56

Figura 16. Etapas de la inteligencia artificial. 57

Figura 17. Diagrama de flujo del sistema 58

Figura 18. Implementación de los elementos. 59

Figura 19. Diagrama electrónico. 60

Figura 20. Modelo 3D. 62

Figura 21. Pista implementada. 62

Figura 22. Implementación del prototipo. 63

Figura 23. Diseño del prototipo. 63

Figura 24. Dispositivo implementado. 63

Figura 25. Sensor de conductividad eléctrica. 64

Figura 26. Sensor de pH. 64

Figura 27. Diseño para los sensores. 64

Figura 28. Protección de los sensores. 65

Figura 29. Protección al driver. 66

Figura 30. Bomba dosificadora. 66

Figura 31. Diagrama de programación. 67

Figura 32. Conexión a la red. 67

Figura 33. Calibración de sensor pH. 68

Figura 34. Calibración del sensor de ce. 68

Figura 35. Configuración del motor. 68

Figura 36. Envió de datos del sistema. 69

Figura 37. Recibimiento de alertas. 69

Figura 38. Visualización de datos a través de la pantalla. 69

Figura 39. Raspberry Pi camera Rev v1.3. 70

Figura 40. Interfaz Ubuntu 20.04. 70

Figura 41. Phpmyadmin. 71

Figura 42. Estructura de la base de datos. 71

Figura 43. Comando para agregar datos. 71

Figura 44. Ingreso de datos. 72

Figura 45. Obtención de datos para la base de datos. 72

Figura 46. Envío de datos a la base de datos. 73

Figura 47. Parámetros para la alerta. 73

Figura 48. Extracción de cada variable. 73

Figura 49. Envió de datos a la nube. 74

Figura 50. Proceso de etiquetado. 75

Figura 51. Exportación del dataset. 75

Figura 52. Verificación de eficiencia del dataset. 76

Figura 53. Dataset descargado. 76

Figura 54. Dataset importado. 77

Figura 55. Comando de entrenamiento. 77

Figura 56. Proceso de entrenamiento. 78

Figura 57. Métricas de rendimiento del modelo entrenado. 80

Figura 58. Matriz de confusión normalizada. 81

Figura 59. Matriz de confusión. 82

Figura 60. Curvatura de precisión del entrenamiento. 82

Figura 61. Reconocimiento de enfermedad. 83

Figura 62. Envío de datos 84

Figura 63. Asignación de valores a la base de datos. 84

Figura 64. Red configurada. 85

Figura 65. Reglas de seguridad. 85

Figura 66. Configuraciones del servidor. 86

Figura 67. Configuración de nodered. 86

Figura 68. ID de la red. 87

Figura 69. Dispositivos dentro de la red. 87

Figura 70. Implementación en el cultivo. 88

Figura 71. Datos del sensor pH. 89

Figura 72. Datos del sensor de ce. 90

Figura 73. Reconocimiento lechuga sana. 91

Figura 74. Reconocimiento de midiú velloso. 91

Figura 75. Reconocimiento de mancha foliar por Septoria. 91

Figura 76. Reconocimiento de mancha foliar bacteriana. 92

Figura 77. Datos almacenados. 93

Figura 78. Reconocimiento en tiempo real. 93

ÍNDICE DE ANEXOS

Anexo A. Certificado. 110

Anexo B. Esp32. 111

Anexo C. Motor a pasos. 111

Anexo D. Driver para el motor. 111

Anexo E. Sensor para ph. 112

Anexo F. Sensor de conductividad eléctrica. 112

Anexo G. Raspberry pi 4. 113

Anexo H. Pruebas del funcionamiento. 114

Anexo I. Código de programación del dispositivo. 116

Anexo J. Código de programación en Python. 122

RESUMEN EJECUTIVO

La baja productividad agrícola es un desafío significativo que impacta la eficiencia y sostenibilidad del sector, especialmente en países en desarrollo. En el caso de los cultivos hidropónicos, esta problemática se agrava por la falta de sistemas avanzados de monitoreo y control. Para abordar esta limitación, se desarrolló el Sistema de monitoreo y control IoT para cultivos agrícolas basado en Edge-Cloud y Deep Learning, optimizando el crecimiento y gestión de los cultivos.

La implementación del sistema IoT para el monitoreo y control de cultivos agrícolas basado en arquitecturas Edge-Cloud y Deep Learning permitió el procesamiento de datos en tiempo real, optimizando la gestión de recursos y reduciendo la intervención manual. Se desarrolló en cuatro capas. En Edge Layer, dispositivos recopilaron datos de pH y conductividad eléctrica, mientras una cámara capturó las lechugas. En Server Layer, se procesó y almacenó datos, entrenó y aplicó el modelo de inteligencia artificial. En Cloud Layer, una red virtual gestionó la información. Finalmente, en Display Layer, una interfaz permitió la visualización en tiempo real, facilitando la supervisión del sistema.

Los resultados evidenciaron la efectividad del sistema, alcanzando una precisión del 99.76% y 98.80% en la medición de pH y conductividad eléctrica. El modelo de reconocimiento de enfermedades en lechugas logró un 87% en entrenamiento y 84.92% en pruebas reales, permitiendo detección temprana y reduciendo pérdidas. La integración de IoT, Edge-Cloud y Deep Learning optimizó el monitoreo y control, reduciendo costos y mejorando la eficiencia en la aplicación de nutrientes, garantizando un sistema más sostenible.

Palabras clave: Cultivos hidropónicos, lechuga, IoT, Edge-Cloud, Deep Learning.

ABSTRACT

Low agricultural productivity is a significant challenge that impacts the efficiency and sustainability of the sector, especially in developing countries. In the case of hydroponic crops, this problem is aggravated by the lack of advanced monitoring and control systems. To address this limitation, the IoT Monitoring and Control System for Agricultural Crops based on Edge-Cloud and Deep Learning was developed, optimizing crop growth and management.

The implementation of the IoT system for monitoring and control of agricultural crops based on Edge-Cloud and Deep Learning architectures allowed for real-time data processing, optimizing resource management and reducing manual intervention. It was developed in four layers. In the Edge Layer, devices collected pH and electrical conductivity data, while a camera captured the lettuce. In the Server Layer, data was processed and stored, the artificial intelligence model was trained and applied. In the Cloud Layer, a virtual network managed the information. Finally, in the Display Layer, an interface allowed real-time visualization, facilitating system monitoring.

The results demonstrated the effectiveness of the system, reaching an accuracy of 99.76% and 98.80% in the measurement of pH and electrical conductivity. The lettuce disease recognition model achieved 87% in training and 84.92% in real tests, allowing early detection and reducing losses. The integration of IoT, Edge-Cloud and Deep Learning optimized monitoring and control, reducing costs and improving efficiency in the application of nutrients, guaranteeing a more sustainable system.

Keywords: Hydroponic crops, lettuce, IoT, Edge-Cloud, Deep Learning.

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA DE MONITOREO Y CONTROL IOT PARA CULTIVOS AGRÍCOLAS BASADO EN LA ARQUITECTURA EDGE CLOUD Y DEEP LEARNING

1.1.1 Planteamiento del problema

La situación actual durante la próxima década, se espera que la producción mundial de cultivos, ganado y productos pesqueros cubiertos aumente un 1,1% anual, una tasa más lenta que en décadas anteriores. Los reducidos incentivos de crecimiento se deben a un debilitamiento de los rendimientos brutos esperados para los productores tanto por las ventas como por la evolución de los costes. Se espera que los costos de los insumos aumenten, en particular debido al vínculo entre los precios de la energía y los fertilizantes y el endurecimiento de las regulaciones ambientales.

Abordar esta problemática requiere comprender la interconexión entre estos factores y trabajar en soluciones que promuevan la seguridad alimentaria y la sostenibilidad en el sector agrícola a nivel global. En este contexto, el argumento a favor de la inversión en tecnología agrícola se basa en la posible utilidad para el sector privado, siendo que los proveedores y productores solo asumirán el compromiso necesario si los beneficios superan a los costos. Sin embargo, en ciertos casos, los costos de inversión pueden superar los beneficios de los productores, aunque existan importantes beneficios para la sociedad en general [1], [2], [3].

Los cultivos permanentes representan el 26,9 % de la superficie de labor agropecuaria, los cultivos con mayor volumen de producción (de mayor a menor) son: banano, caña de azúcar y palma africana. En el año 2023 la superficie total nacional de Ecuador fue de 12,2 millones de hectáreas, presentando un leve aumento del 0,2 % respecto a la superficie total del año anterior. Esto requiere un enfoque integral que incorpore la digitalización, promueva la resiliencia climática, mejore la salud del suelo y fomente la colaboración entre los actores del sector. La digitalización mejora en 30 % el precio al productor y genera ahorros de hasta 75 % a los consumidores. Estas acciones son

esenciales para aumentar la productividad agrícola y garantizar un desarrollo agrario sostenible en Ecuador, especialmente en las áreas rurales con acceso limitado a la tecnología [4], [5], [6].

En la provincia de Tungurahua las producciones agrícolas, como la papa, posee 2191 hectáreas plantadas y 1987 hectáreas efectivamente cultivadas, evidenciando una brecha entre las proyecciones de producción y la realidad cultivada. Al analizar datos relacionados con el empleo, la inversión y la competitividad en diversos sectores económicos, se logra obtener una visión integral de la dinámica económica en cada región. Estos datos revisten una importancia crucial para orientar la formulación de estrategias de desarrollo económico a nivel local, al proporcionar información fundamental para identificar áreas prioritarias de intervención y para diseñar políticas que fomenten un crecimiento equilibrado y sostenible, con ello, se destaca la posibilidad de emplear tecnología para recopilar y analizar datos climáticos e hidrológicos, lo que podría contribuir significativamente a mejorar la planificación y la toma de decisiones en el sector agrícola de la provincia [7], [8], [9].

La baja productividad agrícola representa un desafío crítico para numerosos países en desarrollo, amenazando la seguridad alimentaria y los medios de vida de millones de personas. Este complejo problema encuentra sus raíces en una constelación de factores interrelacionados que van desde limitaciones tecnológicas hasta prácticas agrícolas deficientes y condiciones ambientales adversas. Para abordar eficazmente esta problemática, es esencial comprender a fondo las causas subyacentes y sus efectos perjudiciales en los sistemas de cultivo. Un análisis crítico de estas relaciones causales revelará puntos de intervención clave y sentará las bases para la formulación de soluciones integrales y sostenibles.

La inadecuada gestión de fertilizantes es una causa importante que contribuye a la degradación del suelo en muchas regiones agrícolas. El uso excesivo o inapropiado de fertilizantes químicos puede alterar el equilibrio de nutrientes en el suelo, acidificarlo y acumular sales en niveles tóxicos para las plantas. Además, la falta de rotación de cultivos y la aplicación desbalanceada de nutrientes agotan el suelo de sus reservas naturales. Como resultado, se produce un declive en la fertilidad del suelo, disminuyendo su capacidad para sostener el crecimiento óptimo de los cultivos. Esta

degradación se manifiesta en la compactación, erosión y pérdida de materia orgánica del suelo, dificultando aún más su productividad agrícola a largo plazo.

La brecha digital prevaleciente en muchos países, caracterizada por el limitado acceso a tecnologías de comunicación e información, representa una barrera significativa para que los agricultores obtengan datos e información actualizada crucial para mejorar su productividad. Sin dispositivos inteligentes y con una cobertura deficiente de Internet y servicios de telecomunicaciones, los productores carecen de los medios para acceder a recursos en línea que brindan conocimientos sobre prácticas agrícolas, pronósticos climáticos y precios de mercado. Esta falta de información oportuna los deja en desventaja para tomar decisiones informadas y adoptar técnicas modernas que impulsen sus rendimientos.

La falta de control climático adecuado en los sistemas de cultivo es un factor que facilita la proliferación de plagas que dañan los cultivos. Cuando las condiciones ambientales como la temperatura, humedad y exposición solar no se regulan de manera óptima, se crean entornos propicios para el desarrollo y reproducción de insectos, hongos y otros organismos perjudiciales. Las plagas encuentran un hábitat favorable para su crecimiento exponencial, alimentándose de los cultivos y propagando enfermedades que pueden devastar cosechas enteras. Esta amenaza se ve exacerbada por la falta de medidas preventivas, monitoreo constante y estrategias de control integrado de plagas ajustadas a las condiciones climáticas locales. En consecuencia, los rendimientos agrícolas se ven gravemente afectados, socavando los esfuerzos por alcanzar una producción sostenible.

Es evidente que la baja producción agrícola es un reto multidimensional que requiere un enfoque holístico. Las causas identificadas, como la falta de sistemas de riego adecuados, el acceso limitado a tecnologías de información, la mala gestión de fertilizantes y el control climático deficiente, se encuentran profundamente entrelazadas y se refuerzan mutuamente. Para romper este ciclo vicioso, es imperativo que los esfuerzos se centren en abordar estas causas de manera coordinada e integrada. Soluciones sostenibles que combinen infraestructura mejorada, transferencia de tecnología, capacitación en prácticas agrícolas y estrategias de mitigación del cambio

climático podrían impulsar un aumento significativo en los rendimientos y la resiliencia de los sistemas de cultivo.

1.2 Antecedentes investigativos

En relación con el tema que se está planteando se revisó antecedentes similares, sirviendo como referencia a la investigación de este plan de tesis.

En el trabajo desarrollado por Jorge Andrés Criollo Navarrete presenta el diseño e implementación de un sistema inteligente basado en visión por computadora y algoritmos de aprendizaje de máquina para optimizar el crecimiento de cultivos en micro granjas, con énfasis en la detección de la enfermedad Erysiphe cichoracerum en cultivos de lechuga. Realizado en Ibarra, Ecuador, en 2023, el sistema integra dos componentes principales: un software basado en Python y Tiny YOLO para el reconocimiento de imágenes, y un hardware que incluye cámaras y aceleradores GPU para garantizar el procesamiento en tiempo real. El sistema se probó tanto en entornos controlados como reales, alcanzando altos niveles de precisión en la detección de enfermedades y condiciones óptimas para el cultivo. La implementación permitió un monitoreo eficiente, reduciendo pérdidas en los cultivos y mejorando la calidad de la producción. Los resultados demostraron que la implementación de este sistema puede incrementar la productividad hasta en un 20% en comparación con métodos agrícolas tradicionales, resaltando su viabilidad como herramienta para la agricultura de precisión y sostenible. Se concluye que este sistema tiene el potencial de revolucionar la agricultura de precisión, al ofrecer una herramienta autónoma y escalable que promueve la sostenibilidad agrícola [10].

Así mismo, el estudio realizado por Montaño, González, Jiménez, Mingo y Carrión presenta el diseño e implementación de un sistema de monitoreo basado en IOT para cultivos hidropónicos. El trabajo se llevó a cabo en Loja, Ecuador en el año 2023, con un enfoque experimental cuantitativo para desarrollar e implementar un sistema IOT para monitorear y predecir variables ambientales como temperatura, humedad y radiación UV. El sistema propuesto cuenta con una arquitectura de cuatro capas: percepción, red, middleware y aplicación, que permite un monitoreo escalable y de bajo costo. Los resultados muestran que el sistema desarrollado es lo suficientemente

inteligente para controlar adecuadamente el entorno hidropónico en función de los múltiples parámetros de entrada recopilados, facilitando una gestión efectiva para los agricultores y mejorando la producción. Además, se implementó un modelo de predicción basado en redes neuronales artificiales que alcanzó una precisión del 89,37% en la predicción de la temperatura en función de la humedad [11].

Por otro lado, el trabajo de Diego Bustamante desarrollado en Ambato, Ecuador, durante su investigación de maestría en 2022, propone un sistema de cultivo aeropónico de precisión para Solanum Tuberosum, supervisado mediante IoT, con el objetivo de optimizar el control de variables críticas y fomentar el mejoramiento genético del tubérculo. El sistema integra sensores avanzados como el DTH21 para humedad y temperatura, el sensor 5A123PH para pH y el DFR0300 para conductividad eléctrica, controlados por un microcontrolador ESP32 utilizando el protocolo MQTT y la plataforma Ubidots para visualización en tiempo real. El prototipo mantuvo parámetros ideales para el cultivo, incluyendo una humedad relativa de 90-95%, temperatura entre 16-19 °C, pH entre 6.5 y 6.8, y conductividad eléctrica entre 1.5 y 2.5 dS/m, asegurando condiciones óptimas para el crecimiento y desarrollo del tubérculo. Los resultados demostraron que el sistema es efectivo para crear un ambiente controlado, mejorando significativamente la eficiencia y sostenibilidad del cultivo, y ofreciendo una herramienta confiable para la toma de decisiones en la agricultura de precisión. Este enfoque representa un avance en la integración de tecnologías IoT con la agricultura, favoreciendo el desarrollo sostenible y productivo de la variedad "Super Chola" [12].

Además, el trabajo realizado por Golondrino, Alejandro Alarcón y Saba en la Universidad de Cartagena en Colombia en 2022, presenta un Panorama sobre el desarrollo de un sistema IOT para el monitoreo y análisis de variables climatológicas de interés en cultivos de agricultura urbana. El sistema propuesto se basa en una arquitectura IOT de cuatro capas (captura, almacenamiento, análisis y visualización), que utiliza herramientas de hardware y software libre para monitorizar variables como temperatura, humedad y luminosidad en cultivos de agricultura urbana, específicamente en el contexto de un cultivo casero de lechuga. Los resultados muestran que el 72.4 % de las instancias o lecturas fueron clasificadas en el clúster 1 y se encuentran alrededor del centroide, mientras que el 27.6 % de las capturas se

encuentran clasificadas en el clúster 2 y se encuentran alrededor del centroide, esto quiere decir que el sistema desarrollado permite una adecuada captura, monitorización y análisis de las variables climáticas, así como la posibilidad de personalizar los modelos de análisis de acuerdo a las características agro-climáticas de cada cultivo, lo cual resulta útil para la toma de decisiones relacionadas con la fisiología de los cultivos [13].

En el mismo sentido, el trabajo realizado por Premkumar y Sigappi en India en el año de 2022 propone un modelo de computación de borde habilitado para IOT para un sistema de riego inteligente. Este estudio se centra en predecir la humedad del suelo en tiempo real y gestionar el uso del agua según las predicciones meteorológicas. Los autores desarrollaron un algoritmo híbrido de predicción de humedad del suelo (SMPHA), combinando técnicas de aprendizaje automático como XGBoost y k-means, demostrando un mejor desempeño que otros enfoques basados en regresión. El modelo implementa un servidor perimetral para el procesamiento y la toma de decisiones a nivel local, mejorando latencia, ancho de banda y tiempo de respuesta en comparación con un modelo basado en la nube. La evaluación del uso de recursos en un nodo de borde Raspberry Pi demuestra la viabilidad de implementar SMPHA en el borde. Los resultados de la evaluación se muestran en cuanto a latencia y tiempos de respuesta en una perspectiva de 10 días. En parámetro de latencia, el servicio de borde ha disminuido en un promedio de 77,85% en comparación con la nube. De la misma manera, el tiempo de respuesta del servicio de borde también se reduce en un 74,09% en comparación con el servicio de nube. En general, el sistema propuesto permite un riego automático y optimizado, resultando en un uso más eficiente del agua [14].

Además, la investigación realizada por Kyuchang, Bhagya y Kijun Han en la Universidad Nacional de Kyungpook en Corea del Sur en el año 2020, propone un algoritmo llamado DLEFN (Deep Learning Entrusted to Fog Nodes) para implementar servicios de Deep Learning en un entorno de computación en la niebla orientado a la agricultura inteligente. El DLEFN decide las capas óptimas del modelo de Deep Learning que deben ejecutarse en cada nodo de niebla, considerando su capacidad de cómputo y ancho de banda disponibles. Esto permite aprovechar mejor los recursos distribuidos en la computación en la niebla, acomodando un mayor número de aplicaciones de Deep Learning y reduciendo la congestión de la red y la carga

computacional en la nube. Los resultados muestran que el método de definir capas fijas para todas las tareas permitió de manera constante hasta 160 aplicaciones. Sin embargo, no pudo dar cabida a más solicitudes. El método fuera de línea permitió constantemente hasta alrededor de 250, esto muestra que el DLEFN es una solución eficiente para implementar Deep Learning en entornos de agricultura inteligente basados en computación en la niebla [15].

1.3 Fundamentación teórica

La agricultura, uno de los sectores más antiguos y fundamentales de la economía global, enfrenta desafíos significativos en la actualidad debido al crecimiento de la población, la escasez de recursos naturales, el cambio climático y la necesidad de aumentar la eficiencia y sostenibilidad de las prácticas agrícolas. Ante estos retos, la adopción de tecnologías emergentes como el Internet de las Cosas (IoT), la arquitectura Edge-Cloud y el Deep Learning ha generado un impacto transformador.

El Internet de las Cosas (IoT) permite la interconexión de dispositivos y sensores que recopilan y transmiten información en tiempo real, optimizando la gestión de cultivos y recursos. La arquitectura Edge-Cloud, por su parte, proporciona una plataforma eficiente para procesar y analizar grandes volúmenes de datos de forma local y en la nube, facilitando decisiones rápidas y precisas, esenciales en el entorno agrícola. Finalmente, el Deep Learning, una rama del aprendizaje automático ha demostrado un gran potencial para mejorar la toma de decisiones mediante el análisis avanzado de patrones complejos y la automatización de procesos.

1.3.1 Agricultura de precisión

La agricultura de precisión (AP) es una práctica que ha transformado la gestión agrícola, integrando tecnologías avanzadas para observar, medir y responder a la variabilidad interna y externa en los cultivos. El concepto de AP surgió como respuesta a la necesidad de optimizar las prácticas agrícolas frente al crecimiento de la población y los desafíos del cambio climático. Desde sus primeras aplicaciones en los años 80, la AP ha evolucionado para convertirse en un enfoque integral que combina ciencias agronómicas, ingeniería y tecnologías de la información [16].

Su principal objetivo es mejorar la eficiencia en el uso de los recursos, incrementar la productividad y reducir el impacto ambiental mediante el uso de herramientas tecnológicas como el GPS, sensores remotos, sistemas de información geográfica (SIG) y modelos predictivos [17].

a. Principales Componentes de la Agricultura de Precisión (AP)

La AP se basa en un ecosistema tecnológico que permite la recolección, análisis y aplicación de datos agrícolas [18]:

- Sistemas de Posicionamiento Global (GPS): Los sistemas GPS son esenciales para el mapeo y la localización precisa dentro de los campos. Facilitan la gestión de operaciones como la siembra, la aplicación de fertilizantes y la cosecha, asegurando una cobertura uniforme y eficiente.

- Sensores Remotos: Los sensores permiten la monitorización continua de variables críticas como la humedad del suelo, la temperatura y la condición de las plantas. Estas tecnologías incluyen imágenes satelitales y drones equipados con cámaras multiespectrales y térmicas, que detectan problemas como estrés hídrico y enfermedades antes de que sean visibles a simple vista.

- Tecnología de Tasa Variable (VRT): La VRT permite la aplicación diferenciada de insumos agrícolas (fertilizantes, pesticidas, agua) según las necesidades específicas de cada área del campo. Esto no solo optimiza el uso de recursos, sino que también reduce costos y minimiza el impacto ambiental.

- Sistemas de Información Geográfica (SIG): El SIG es fundamental para el análisis y visualización de datos geoespaciales. Mediante mapas temáticos, los agricultores pueden identificar patrones de variabilidad en el suelo, las cosechas y las condiciones climáticas, mejorando la toma de decisiones.

b. Beneficios de la Agricultura de Precisión

La adopción de la AP ha demostrado numerosos beneficios [18]:

- Eficiencia en el Uso de Recursos: Al utilizar tecnologías de monitoreo y control, se reducen significativamente el desperdicio de agua, fertilizantes y

pesticidas. Esto es particularmente relevante en regiones con recursos hídricos limitados.

- Incremento en la Productividad: Los datos precisos permiten a los agricultores optimizar la siembra, fertilización y cosecha, lo que se traduce en mayores rendimientos de los cultivos.

- Reducción del Impacto Ambiental: La gestión controlada de insumos y la detección temprana de problemas contribuyen a prácticas agrícolas más sostenibles.

- Mejora en la Gestión del Tiempo: Al automatizar procesos y centralizar la información, los agricultores pueden tomar decisiones rápidas y eficientes.

c. Desafíos y Limitaciones de la Agricultura de Precisión

Aunque los beneficios de la AP son claros, su implementación enfrenta barreras significativas [18]:

- Altos Costos Iniciales: La inversión en equipos, software y capacitación puede ser prohibitivamente alta para pequeños y medianos agricultores.

- Complejidad Técnica: La integración de tecnologías avanzadas requiere habilidades especializadas que no todos los agricultores poseen.

- Conectividad Limitada: En áreas rurales, la falta de acceso a internet o infraestructura tecnológica puede dificultar el uso de plataformas basadas en la nube.

- Manejo de Datos: La recopilación de grandes volúmenes de datos plantea desafíos relacionados con el almacenamiento, procesamiento y análisis eficiente.

d. Aplicaciones Prácticas de la Agricultura de Precisión

La AP tiene aplicaciones prácticas en diversas áreas de la agricultura [18]:

- Monitoreo de Cultivos: Los drones y sensores satelitales permiten un seguimiento detallado de la salud del cultivo, detectando problemas como plagas, enfermedades o deficiencias nutricionales.

- Gestión de Suelos: La AP ayuda a mapear la fertilidad y estructura del suelo, permitiendo prácticas como la rotación de cultivos y la siembra dirigida.

- Irrigación de Precisión: A través del uso de sensores de humedad y sistemas automatizados, los agricultores pueden aplicar agua solo en áreas que lo necesiten, mejorando la eficiencia hídrica.

- Control de Plagas y Enfermedades: Mediante la identificación de áreas afectadas, se pueden aplicar pesticidas de forma localizada, reduciendo el uso generalizado de químicos.

1.3.2 Cultivos hidropónicos

La AP y los cultivos hidropónicos son dos enfoques innovadores que, al integrarse, ofrecen soluciones avanzadas para la producción agrícola sostenible y eficiente. La AP se centra en la gestión detallada de las prácticas agrícolas mediante el uso de tecnologías avanzadas, mientras que la hidroponía es una técnica de cultivo sin suelo que utiliza soluciones nutritivas para el crecimiento de las plantas.

La combinación de AP y sistemas hidropónicos permite una gestión más precisa de las variables críticas en el cultivo, como la nutrición, el riego y las condiciones ambientales. La AP proporciona herramientas y tecnologías que facilitan el monitoreo y control de estos factores, optimizando el crecimiento y la salud de las plantas en sistemas hidropónicos [19].

La Figura 1 muestra una idea de la implementación de los cultivos hidropónicos, destacando cómo las plantas se desarrollan dentro del sistema. Este tipo de cultivo utiliza soluciones nutritivas en lugar de suelo, lo que permite un control más preciso de los nutrientes necesarios para el crecimiento óptimo de las plantas.



Figura 1. Cultivos hidropónicos [19].

Aplicaciones de la agricultura de precisión en hidroponía [19]:

- Monitoreo de Nutrientes y pH: La AP utiliza sensores avanzados para medir en tiempo real los niveles de nutrientes y el pH de las soluciones hidropónicas, permitiendo ajustes inmediatos para mantener condiciones óptimas para el crecimiento de las plantas.

- Control de Temperatura y Humedad: Mediante el uso de sistemas de información geográfica y sensores ambientales, es posible monitorear y controlar la temperatura y humedad en los invernaderos hidropónicos, asegurando un ambiente adecuado para las plantas.

- Detección de Estrés y Enfermedades: La integración de imágenes multiespectrales y análisis de datos permite la detección temprana de estrés hídrico, deficiencias nutricionales y enfermedades en las plantas, facilitando intervenciones oportunas.

Beneficios de Integrar AP en Cultivos Hidropónicos [19]:

- Eficiencia en el Uso de Recursos: La precisión en la aplicación de nutrientes y agua reduce el desperdicio y optimiza el uso de recursos, lo que es especialmente beneficioso en regiones con limitaciones hídricas.

- Incremento en la Productividad: El monitoreo constante y los ajustes basados en datos permiten maximizar el rendimiento de los cultivos, obteniendo productos de mayor calidad.

- Sostenibilidad Ambiental: La reducción en el uso de insumos químicos y la eficiencia en el uso del agua contribuyen a prácticas agrícolas más sostenibles y amigables con el medio ambiente.

a. Selección de producto

La implementación de hortalizas en este tipo de cultivos es lo más común, como tomate, cebolla, ajo, nabo, ente otros. Entre las diversas hortalizas cultivadas, la lechuga (Lactuca sativa) se posiciona como uno de los productos más demandados, tanto por su versatilidad en la gastronomía como por su alto valor nutricional. La producción de lechuga en esta región se ha adaptado a las condiciones climáticas templadas y a las diferentes altitudes que favorecen el crecimiento de variedades específicas. Además, la introducción de tecnologías de cultivo bajo invernadero ha optimizado los rendimientos, permitiendo a los agricultores obtener mejores resultados durante todo el año, en la Tabla 1 se muestra las diferentes variedades que hay en el país.

Tabla 1. Tipos de lechugas existentes en Ecuador [10].

Tipo	**Características**	**Imagen**
Red-Fire	Las hojas de esta planta son un ingrediente esencial en la cocina, agregando un toque único a las preparaciones culinarias.	
Mac	Esta hortaliza es conocida por su resistencia al calor y se utiliza comúnmente en hamburguesas debido a su textura crujiente.	
Crespa	Es una lechuga rizada y pequeña, muy popular en el mercado debido a su versatilidad y durabilidad. Se utiliza comúnmente en ensaladas y como decoración para agregar un toque fresco y visual a los platos.	

Tipo	Características	Imagen
Escarola	Conocida por sus hojas rizadas y abiertas, con una base blanca y arrosetada que se alternan con segmentos de colores verde y amarillo.	
Luana o Salinas	Este repollo es la variedad más popular y demandada, conocido por su sabor agradable y su apariencia atractiva, ideal para preparaciones culinarias.	
Achicoria	Esta lechuga es valorada por su sabor amargo y se utiliza en cocina gourmet. Se recomienda cocinarla hirviendo en agua durante 10 minutos.	

La lechuga crespa es relevante en la agricultura y la alimentación debido a su alto contenido de agua, vitaminas (A, C y K), minerales (como potasio y hierro) y fibra, lo que la convierte en un alimento esencial en dietas saludables. Su cultivo es versátil, adaptándose a diferentes sistemas agrícolas, incluidos métodos sostenibles como la hidroponía, donde es una de las hortalizas más populares por su rápido crecimiento y alto rendimiento.

b. Variables que controlar

En el cultivo hidropónico, el control preciso de ciertos parámetros es esencial para garantizar un crecimiento saludable y un rendimiento óptimo de las plantas. Dos de los factores más críticos que deben ser monitoreados constantemente son el pH y la conductividad eléctrica (CE), ya que ambos influyen directamente en la disponibilidad de nutrientes y el desarrollo de las plantas.

- ***pH:*** El pH de la solución nutritiva en sistemas hidropónicos es un parámetro clave, ya que afecta la absorción de nutrientes esenciales por parte de las plantas. En el caso de la lechuga, se requiere un rango de pH entre 5.8 y 6.2 para maximizar la absorción de nutrientes como el nitrógeno, el fósforo y el potasio. Valores de pH fuera de este rango pueden limitar la disponibilidad de ciertos elementos, causando deficiencias nutricionales o toxicidad en las plantas. Un sistema de monitoreo continuo del pH permitirá ajustar rápidamente la acidez o alcalinidad de la solución, asegurando que las plantas reciban los nutrientes en su forma más accesible. Los niveles de pH muy bajos o ácidos reducen la disponibilidad de nutrientes esenciales como nitrógeno, potasio, calcio, magnesio y fósforo, mientras que los niveles de pH muy altos o alcalinos disminuyen la disponibilidad de la mayoría de los micronutrientes esenciales.

- ***Conductividad Eléctrica (CE):*** La conductividad eléctrica (CE) es una medida de la cantidad de sales disueltas en la solución nutritiva, lo que a su vez indica la concentración de nutrientes disponibles para las plantas. En cultivos de lechuga bajo sistemas hidropónicos, el rango óptimo de CE suele estar entre 1.5 y 2.5 mS/cm. Un valor demasiado bajo puede indicar una insuficiencia de nutrientes, mientras que un valor excesivamente alto podría provocar un estrés osmótico en las plantas, afectando negativamente su crecimiento. La CE debe ser ajustada de acuerdo con la fase de desarrollo de las plantas, y su control continuo es esencial para garantizar una nutrición equilibrada. Una (CE) baja en la lechuga, como 0,5 mS/cm, indica una deficiencia de nutrientes y requiere un ajuste de la dosis de inyección de nutrientes. Por otro lado, una CE alta, como 4 mS/cm, sugiere la acumulación de sales en la solución y puede requerir un recambio del sistema o una reducción de la inyección de nutrientes [20].

El control adecuado de estos dos parámetros asegura que las plantas crezcan en condiciones óptimas, minimizando los riesgos asociados a un desequilibrio en la solución nutritiva, como el estrés fisiológico o el crecimiento desigual. Además, la implementación de sistemas automáticos de monitoreo de pH y ce permite realizar ajustes precisos en tiempo real, lo que mejora la eficiencia del sistema hidropónico y maximiza los rendimientos.

La Figura 2 muestra una referencia a la armonía que se debe tener para obtener una solución nutritiva eficaz para la producción de la lechuga.



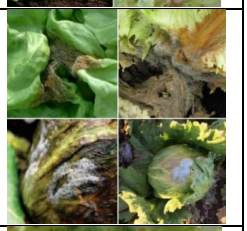
Figura 2. Parámetros de control para una solución nutritiva equilibrada [20].

c. Enfermedades existentes

La lechuga (Lactuca sativa) enfrenta diversas enfermedades que reducen su rendimiento. La aparición y gravedad de estas afecciones dependen del patógeno involucrado, la vulnerabilidad de la planta y las condiciones ambientales. Para cada enfermedad se describen los síntomas y las estrategias de manejo integrado.

Las enfermedades más comunes que se puede presenciar en la producción de lechuga se dan por hongos, bacterias y virus. Se presentan las enfermedades más comunes causadas por hongos en las lechugas y como estas afectan al producto en la Tabla 2.

Tabla 2. Principales enfermedades causadas por hongos [10], [21].

Enfermedad	Descripción	Daños	Prevención	Tratamiento	Imagen
Alternaria	Causado por el hongo Alternaria dauci. Se desarrolla en condiciones húmedas entre 25 a 28 °C	Manchas redondas oscuras necróticas en las hojas.	Evitando la humedad excesiva del follaje	Usando fugicidas con Cimoxanillo, Mancozeb o Zineb	
Añublo lanoso (Downy mildew)	Enfermedad presente en condiciones de temperaturas frescas y alta humedad	Se observan áreas irregulares de color amarillas y necróticas, se ve un crecimiento de un moho blanco lanoso creciendo en la hoja	El riego no debe ser aéreo para evitar un ambiente propicio para la enfermedad.	Usando fugicidas con Cimoxanillo, Mancozeb o Zineb	
Esclerotinia	Vellosidad blanca en la base de la lechuga.	Se ve afectado en el cuello de la planta y se extiende provocando que se pudra.	Aplicaciones de fungicidas.	Metalaxil o tiram	
Botrytis	Hongo más común, tiene un aspecto amarronado a grisáceo con velocidad.	Produce la necrosis y el reblandecimiento de tejidos afectados.	Tiene una buena ventilación evitando que la humedad se quede en las hojas.	Trichoderma harzianum.	
Microdochium panattoniana o Antracnosis en lechuga	Hongo fitopatógeno que produce la enfermedad. Aparece sobre las hojas más viejas de la planta.	Sobre las hojas aparece manchas amarillentas con margen necrótico o rojizo expandiendo al interior.	Eliminar malezas, hojas secas.	Agentes químicos antifúngicos, derivados del cobre.	

Enfermedad	Descripción	Daños	Prevención	Tratamiento	Imagen
Erysiphe cichoracerum u Oidio en Lechuga	Hongo ectoparásito, presencia común por altos niveles de humedad y excesivo consumo de nutrientes.	Presente en el haz como en el envés de la hoja y tallo.	Controlando los niveles de nitrógeno.	Aplicar polvos minerales de azufre o cobre.	
Septoria lactucae o Septoriosis en lechuga	Hongo patógeno apareciente en épocas de humedad.	Afecta inicialmente a las hojas viejas, produciendo manchas irregulares en la hoja, desarrollándose hasta ser necróticas.	Control de la humedad y limpieza del cultivo.	Cimoxanilo, productos cuprosos y Mancozeb.	

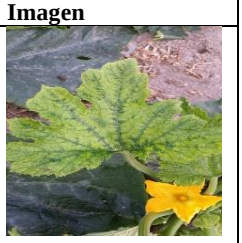
La Tabla 3 presenta una descripción detallada de las principales enfermedades bacterianas que afectan a los cultivos de lechuga.

Tabla 3. Principales enfermedades causadas por bacterias [21].

Enfermedad	Descripción	Daños	Prevención	Tratamiento	Imagen
Tizón bacteriano (“Bacterial blight”)	La enfermedad se ve favorecida por las condiciones cálidas y húmedas.	Produce pequeñas manchas marrones redondas características con lesiones empapadas de agua.	Se recomienda el uso de semilla certificada libre de enfermedades.	La rotación de cultivos, fertilización adecuada y control de la humedad	
Mancha foliar bacteriana	Se pueden observar manchas pequeñas rojizas cerca del márgen de las hojas. Esta bacteria se encuentra en el suelo y puede encontrarse en asociación con P. cichorii.	Necrosis de color marrón en los márgenes de las hojas, que se extiende hacia el centro de la hoja a través de las venas.	Uso de semilla certificada libre de enfermedades.	Fertilización adecuada y control de la humedad	

La Tabla 4 una lista detallada de las principales enfermedades que afectan a las lechugas y que son causadas por diversos tipos de virus presentes en los cultivos del producto.

Tabla 4. Principales enfermedades causadas por virus [21].

Enfermedad	Descripción	Daños	Prevención	Tratamiento	Imagen
Virus de Mosaico del Pepinillo	Enfermedad de poca importancia en climas cálidos en este cultivo	Enanismo, moteado amarillo, distorsión y manchas necróticas en las hojas.	Evite sembrar cerca de siembras abandonadas o recientemente cosechadas de lechuga y cucurbitáceos.	Por ser un problema de menor importancia en lechuga no se siguen medidas de control establecidas.	
Virus del Mosaico de la Lechuga	Se transmite en la semilla y por áfidos como el pulgón verde. Los síntomas varían dependiendo del cultivar, la edad de la planta y las condiciones ambientales existentes.	Las hojas de plantas jóvenes muestran un moteado verde claro o mosaico. Aclaramiento de las venas, rayado necrótico, reducción del crecimiento de la planta y distorsión.	Uso de semilla indicada como libre del virus y cultivares resistentes suplementado con control de las malezas	Fertilización adecuada y control de la humedad	

Entre las patologías que más afectan a la lechuga en cultivo hidropónico se encuentran la mancha foliar por Septoria, mancha foliar bacteriana y el mildiú velloso. La detección temprana y precisa de estas enfermedades es crucial para implementar estrategias de control efectivas y minimizar las pérdidas económicas.

- ***Optimización del Uso de Recursos:*** El IoT permite un uso más eficiente de recursos críticos como el agua y los fertilizantes. Permite a los agricultores controlar sus granjas de forma remota. Facilita la gestión más eficaz de las actividades agrícolas. Mediante sensores de humedad y temperatura, los agricultores pueden monitorizar las condiciones del suelo en tiempo real, ajustando el riego y la fertilización según las necesidades específicas de cada área de cultivo. Estas plataformas integran datos de múltiples fuentes e incluyen sistemas de apoyo a la decisión. Esto no solo reduce el desperdicio de agua y nutrientes, sino que también minimiza el impacto ambiental al evitar la excesiva aplicación de químicos [22].

1.3.3 IoT en la Agricultura

El Internet de las Cosas (IoT) permite la interconexión de dispositivos y sensores que recopilan y transmiten información en tiempo real, optimizando la gestión de cultivos

y recursos. Ha transformado la agricultura mediante la integración de dispositivos inteligentes que recopilan, transmiten y analizan datos en tiempo real. Algunas aplicaciones clave incluyen:

- Monitoreo de cultivos y suelos: Los sensores IoT permiten medir variables como humedad del suelo, temperatura, pH, y conductividad eléctrica, proporcionando datos en tiempo real para optimizar el riego y la fertilización.

- Agricultura de precisión: Gracias a los sensores IoT, los agricultores pueden aplicar recursos como agua o pesticidas solo donde es necesario, lo que mejora la eficiencia y reduce el impacto ambiental.

- Mantenimiento predictivo: Las máquinas agrícolas pueden estar equipadas con sensores para detectar fallos o necesidad de mantenimiento antes de que se produzcan fallos críticos.

a. Impacto de IoT en la agricultura

- ***Mejora en la productividad y rendimiento:*** El monitoreo constante y la recopilación de datos en tiempo real permiten a los agricultores tomar decisiones basadas en información precisa, lo que lleva a un aumento en el rendimiento de los cultivos. La población mundial se espera que llegue a 9.600 millones en 2050, lo que presiona a los agricultores a aumentar la producción de alimentos mediante la optimización de recursos, enfrentándose a cambios climáticos y demanda constante de alimentos. Por ejemplo, el uso de cámaras IoT y drones equipados con sensores puede detectar problemas de salud en los cultivos, como plagas o enfermedades, mucho antes de que sean visibles para el ojo humano. Esto permite una intervención temprana y específica, reduciendo las pérdidas y mejorando la productividad [23].

- ***Sostenibilidad y Reducción del Impacto Ambiental:*** El Internet de las Cosas (IoT) es una herramienta clave que permite la agricultura de precisión y optimizar el uso de insumos agrícolas. Las tecnologías como sensores inalámbricos, drones y robótica mejoran la eficiencia y reducen el impacto ambiental. Sistemas de riego inteligentes y agricultura de precisión reducen significativamente el consumo de agua y agroquímicos, minimizando la degradación del suelo y la contaminación.

Estas tecnologías avanzadas son fundamentales para garantizar un futuro más sostenible y resiliente para la industria agrícola y el medio ambiente. Por ejemplo, mediante el uso de sistemas de riego inteligentes, se puede aplicar agua solo en áreas donde realmente se necesita, reduciendo significativamente el consumo de agua y previniendo la degradación del suelo [24].

- ***Reducción de Costos Operativos:*** La reducción de costos operativos es uno de los beneficios clave de la automatización y el uso de tecnologías avanzadas en la agricultura. Al utilizar tecnologías como los sistemas de riego automatizados, la automatización de la maquinaria agrícola y la automatización de la cadena de suministro, los agricultores pueden reducir significativamente los costos operativos y mejorar la eficiencia en la producción.

Uno de los principales beneficiarios de la reducción de costos operativos es la mano de obra. La automatización de la maquinaria agrícola y los sistemas de riego puede reducir la necesidad de empleados, lo que puede significar ahorros en costos de mano de obra y mejorar la eficiencia en la realización de tareas agrícolas. Además, los sistemas de monitoreo remoto permiten a los agricultores supervisar y controlar sus operaciones desde cualquier lugar, reduciendo la necesidad de presencia constante en el campo y minimizando los costos asociados con el transporte y la estancia en el campo.

Otro beneficio de la reducción de costos operativos es la optimización de la cadena de suministro. La automatización de la cadena de suministro puede ayudar a reducir los tiempos de transporte y la pérdida de alimentos, lo que puede significar ahorros en costos de transporte y almacenamiento. Además, la capacidad de detectar problemas en etapas tempranas permite a los agricultores tomar medidas rápidas y efectivas para abordarlos, reduciendo la necesidad de tratamientos extensivos y la pérdida de cultivos [24].

- ***Trazabilidad y Seguridad Alimentaria:*** La integración de IoT en la cadena de suministro agrícola mejora la trazabilidad de los productos. Los sensores y etiquetas RFID permiten rastrear los productos desde el campo hasta el consumidor final, garantizando que se mantengan en condiciones óptimas durante su transporte y almacenamiento. Esto no solo asegura la calidad del producto, sino que también

facilita la identificación y eliminación rápida de productos contaminados en caso de brotes de enfermedades [25].

La Tabla 5 presenta el resumen de lo mencionado anteriormente.

Tabla 5. Impacto de IoT en la agricultura.

Impacto	**Descripción**
Optimización del Uso de Recursos	Uso eficiente de agua y fertilizantes. Monitoreo remoto y ajuste de riego y fertilización mediante sensores de humedad y temperatura. Reducción del impacto ambiental y desperdicio de insumos.
Mejora en la Productividad y Rendimiento	Monitoreo en tiempo real para decisiones informadas. Uso de drones y cámaras IoT para detectar plagas o enfermedades tempranas. Aumento en el rendimiento de cultivos, enfrentando desafíos como el cambio climático.
Sostenibilidad y Reducción del Impacto	Sistemas de riego inteligentes y agricultura de precisión optimizan el uso de recursos, reducen el consumo de agua y agroquímicos, y minimizan la degradación del suelo.
Reducción de Costos Operativos	Automatización de maquinaria agrícola y sistemas de riego disminuyen la necesidad de mano de obra y optimizan la cadena de suministro, reduciendo costos de transporte y pérdida de alimentos.
Trazabilidad y Seguridad Alimentaria	Uso de sensores y etiquetas RFID para rastrear productos desde el campo hasta el consumidor. Mejora en la calidad del producto y respuesta rápida ante contaminaciones.

b. Características de las Soluciones IoT en Agricultura

- ***Sensores de Monitoreo:*** Los sensores son el componente principal de cualquier solución IoT en agricultura. Capturan datos invaluables relacionados con el suelo y los cultivos, monitorean el crecimiento de los cultivos, detectan el estrés hídrico en partes específicas del campo, evalúan biomasa, rendimiento, superficie, vigor de la vegetación, estrés por sequía y desarrollo fenológico.

Estos dispositivos pueden medir una amplia variedad de parámetros, como humedad del suelo, temperatura, luminosidad, niveles de CO2, y más. Permiten una toma de decisiones temprana y eficiente en fertilización, riego y manejo de plagas. Además, proporcionan información para abordar las incertidumbres derivadas de las variaciones en las condiciones climáticas y las estrategias de gestión. La información recopilada se envía a una plataforma central donde se analiza y se utiliza para tomar decisiones informadas sobre la gestión de los cultivos. La Figura 3 muestra el panorama de un campo agrícola digital equipado con dispositivos IoT para monitorear la humedad del suelo, vehículos robóticos (tanto terrestres como aéreos) y antenas que captan imágenes satelitales [22].



Figura 3. Dispositivos IoT [22].

- ***Plataformas de Análisis y Gestión de Datos:*** Los datos generados por los sensores deben ser procesados y analizados para ser útiles. Las plataformas de análisis, a menudo basadas en la nube, permiten la gestión de grandes volúmenes de datos y la aplicación de algoritmos de inteligencia artificial para identificar patrones y realizar predicciones. Estas plataformas son esenciales para la implementación de agricultura de precisión, permitiendo una toma de decisiones más eficiente y precisa. El resultado es una producción de cultivos más robustos que ofrecen productos de mayor calidad. La Figura 4 muestra el ciclo de vida de la agricultura inteligente basada en IoT, desde la obtención de los datos, análisis y las decisiones que se toma a partir de la gestión de los datos [23].

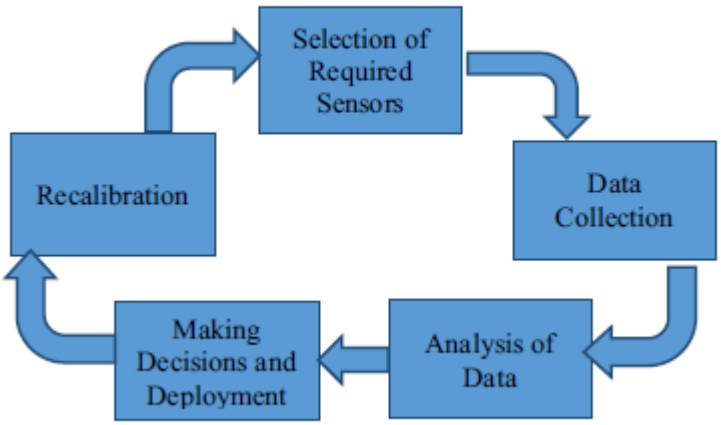


Figura 4. Ciclo de vida de la agricultura inteligente basada en IoT [23].

- ***Automatización de Procesos:*** La automatización es una característica clave del IoT en la agricultura, ya que permite una gestión más eficiente y sostenible de los recursos agrícolas. Los sistemas de riego automatizados, por ejemplo, pueden monitorizar las condiciones del suelo y las plantas, y ajustar automáticamente la cantidad de agua necesaria para garantizar que las plantas reciben el agua adecuada para su crecimiento y desarrollo. Esto reduce la pérdida de agua y minimiza el desperdicio, lo que a su vez ayuda a preservar los recursos hídricos y reducir el impacto ambiental.

Las trampas inteligentes para plagas son otra aplicación del IoT en la agricultura. Estas trampas pueden detectar la presencia de plagas y enfermedades en las plantas y enviar alertas en tiempo real a los agricultores, permitiéndoles tomar medidas rápidas y efectivas para prevenir su propagación. Esto ayuda a reducir la cantidad de pesticidas y productos químicos necesarios, lo que a su vez ayuda a preservar la salud del medio ambiente y los recursos naturales.

La automatización de maquinaria agrícola es otra área en la que el IoT está transformando la agricultura. Las maquinarias agrícolas autónomas pueden realizar tareas como la siembra, el desbroce y la recolección de cultivos de manera más eficiente y precisa, reduciendo la necesidad de mano de obra y aumentando la producción. Además, estas maquinarias pueden ser programadas para trabajar en horarios específicos y en áreas específicas del campo, lo que reduce la pérdida de tiempo y aumenta la eficiencia.

La agricultura inteligente permite calcular con precisión la cantidad de nutrientes necesarios, reduciendo el impacto ambiental. Esto se basa en mediciones detalladas del suelo, que suelen ser costosas y lentas. La Figura 5 presenta un resumen de los insumos clave, los procesos y los resultados principales de la agricultura inteligente. Las tecnologías IoT ayudan a hacer estas estimaciones con mayor exactitud y menos esfuerzo. Herramientas como el NDVI, el GPS, y vehículos autónomos optimizan la aplicación de fertilizantes. Métodos como la fertiirrigación y la fumigación, que utilizan el sistema de riego para aplicar fertilizantes y pesticidas, se han vuelto más efectivos gracias a la integración del IoT [24].



Figura 5. Resultados de la agricultura inteligente [24].

- ***Conectividad y Comunicación:*** La conectividad es un factor crucial para el funcionamiento de las soluciones IoT. En muchas áreas rurales, la falta de infraestructura de red adecuada es un desafío. Sin embargo, se están desarrollando tecnologías como redes de baja potencia (LPWAN) y el uso de satélites para mejorar la conectividad en estas regiones, facilitando la implementación de soluciones IoT en lugares remotos. La Figura 6 muestra una vista instantánea de una red de área agrícola que permite al agricultor visualizar toda la granja en tiempo real [24].

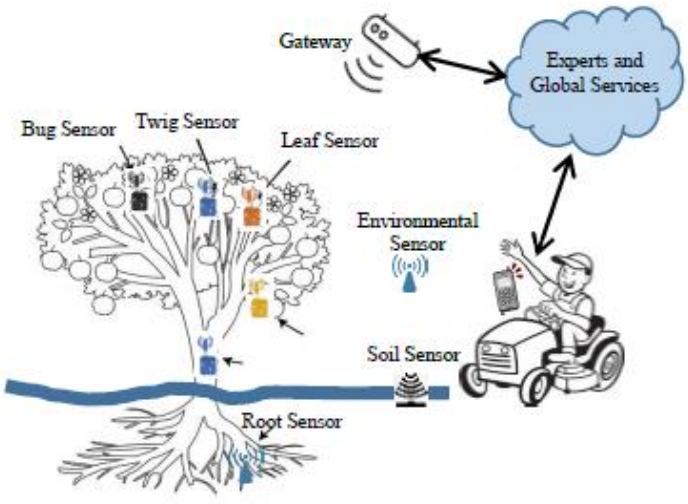


Figura 6. Red de área agrícola basada en IoT [24].

- ***Integración con Inteligencia Artificial y Big Data:*** La combinación de IoT con tecnologías como la inteligencia artificial (IA) y el big data está ampliando las posibilidades en la agricultura. La IA puede analizar grandes volúmenes de datos recopilados por sensores para detectar patrones, predecir tendencias y recomendar acciones. Por ejemplo, se pueden usar algoritmos de aprendizaje automático para prever el rendimiento de los cultivos basándose en datos históricos y en tiempo real. La cadena de datos abarca todas las actividades desde la recolección de datos hasta la toma de decisiones y su comercialización. Este proceso incluye todas las etapas necesarias para gestionar los datos en la agricultura.

La Figura 7 detalla los principales pasos de esta cadena, que es parte fundamental de los procesos empresariales, e incluye una capa técnica encargada de capturar y transformar los datos brutos [25].

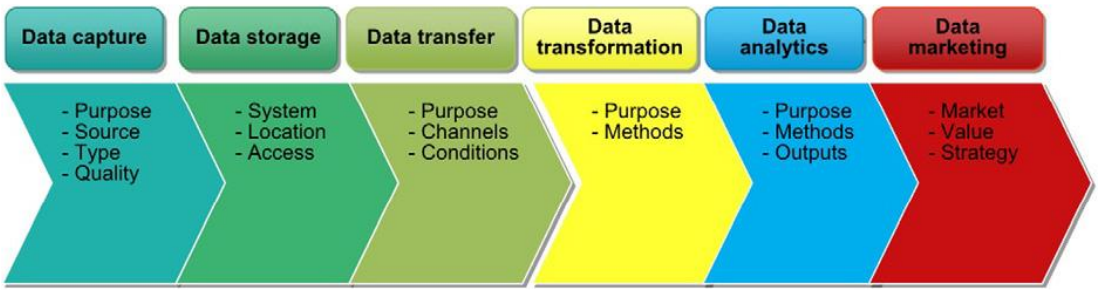


Figura 7. Cadena de datos de aplicaciones de Big Data [25].

La Tabla 6 presenta el resumen de las características.

Tabla 6. Características de IoT en la agricultura.

Carácterísticas	**Descripción**
Sensores de Monitoreo	Miden parámetros clave (humedad, temperatura, luminosidad, CO2, etc.) y permiten decisiones informadas sobre fertilización, riego y manejo de plagas. Envían datos a plataformas centrales para análisis.
Plataformas de Análisis y Gestión de Datos	Procesan grandes volúmenes de datos para identificar patrones y realizar predicciones mediante inteligencia artificial. Permiten implementar agricultura de precisión y mejorar la calidad de los productos.
Automatización de Procesos	Sistemas automatizados para riego, trampas inteligentes para plagas y maquinaria agrícola autónoma mejoran la eficiencia y reducen el impacto ambiental.
Conectividad y Comunicación	Tecnologías como LPWAN y satélites mejoran la conectividad en áreas rurales, permitiendo el monitoreo en tiempo real de toda la granja.
Integración con Inteligencia Artificial	La IA y big data analizan datos históricos y en tiempo real para prever rendimientos, detectar patrones y recomendar acciones. La cadena de datos optimiza la gestión agrícola.

c. Dispositivos IoT

En los cultivos hidropónicos, el monitoreo y control de las variables ambientales y nutricionales es crucial para garantizar el crecimiento óptimo de las plantas. Entre los factores más relevantes se encuentran la conductividad eléctrica, el pH, la cantidad de oxígeno disuelto (DO), la concentración de sólidos disueltos totales (TDS), la humedad y la temperatura del entorno. Para lograr un control preciso y eficiente, se utilizan una variedad de sensores especializados, así como controladores y drivers para la automatización de motores que regulan sistemas de riego, ventilación y otras funciones críticas.

En las tablas comparativa, se presentan los principales sensores y drivers utilizados en sistemas hidropónicos, clasificados por su función. Estos dispositivos han sido seleccionados basados en estudios científicos recientes y su efectividad probada en aplicaciones agrícolas, incluyendo la medición de parámetros químicos y físicos en soluciones nutritivas, así como el control automatizado de motores en sistemas de riego y ventilación.

Cada sensor ha sido evaluado en términos de su rango de medición, precisión, proporcionando una guía clara para la selección de dispositivos según las necesidades de cada sistema hidropónico. Del mismo modo, los drivers de motores paso a paso han sido evaluados por su capacidad de corriente, precisión y adecuación para diferentes niveles de automatización en entornos controlados.

Esta recopilación busca facilitar la identificación de las tecnologías más adecuadas para la implementación de sistemas hidropónicos eficientes y automatizados, asegurando un control preciso de las condiciones ambientales y nutricionales que impactan directamente en la productividad de los cultivos.

Para ello, se presenta la Tabla 7 que muestra los sensores utilizados para medir la conductividad eléctrica, la Tabla 8 para pH, la Tabla 9 los motores a pasos más usados, la Tabla 10 sobre los drivers para controlar motores, Tabla 11 los microcontroladores más frecuentes para el internet de las cosas, la Tabla 12 los microprocesadores y la Tabla 13 cámaras disponibles.

Tabla 7. Sensores de conductividad eléctrica [26], [27], [28].

Sensor	Rango de Medición (mS/cm)	Precisión	Conectividad	Tiempo de Respuesta	Calibración	Compatibilidad
DFRobot Gravity EC Sensor V2.0	0-20	±5% F.S.	Analógica/Digital	1 segundo	Automática o manual	Arduino, esp32
Atlas Scientific EZO-EC	0.07–500	±2%	Digital (UART/I2C)	Instantáneo	Automática o manual	Arduino, esp32, Raspberry
Bluelab Truncheon EC Meter	0.2-3.6	No especificado	Manual (Pantalla LED)	Rápido(manual)	Manual	Manual, no microcontroladores

Tabla 8. Sensores para pH [29], [30], [31].

Dispositivo	Rango de medición	Precisión	Tipo de electrode	Conectividad	Aplicación	Compatibilidad
DFRobot Gravity pH Sensor	0-14 pH	± 0.1 pH	Vidrio	Analógica	Medición del pH en soluciones hidropónicas.	Compatible con Arduino y microcontroladores
Atlas Scientific EZO-pH	0-14 pH	± 0.002 pH	Vidrio	UART/I2C	Control avanzado de pH en hidroponía automatizada.	Compatible con Raspberry Pi, Arduino, IoT
DIY SEMSPr pH-4502C	0-14 pH	± 0.1 pH	Vidrio	Analógica	Sistemas de medición de pH en cultivos pequeños y medianos.	Compatible con Arduino y microcontroladores

Tabla 9. Motores a pasos [32].

Motor	Tamaño de Brida (mm)	Par de Retención (Nm)	Corriente por Fase (A)	Ángulo de Paso (°)	Aplicaciones Comunes
NEMA 17	42 x 42	0.4 - 0.5	1.2 - 2.0	1.8	Impresoras 3D, máquinas CNC pequeñas, robótica ligera.
NEMA 23	57 x 57	1.0 - 3.0	2.8 - 3.0	1.8	Máquinas CNC, impresoras 3D de mayor tamaño, automatización industrial.
NEMA 34	86 x 86	4.5 - 8.0	4.2 - 6.0	1.8	Máquinas CNC de alta potencia, robótica industrial, aplicaciones de alto torque.

Tabla 10. Driver de control de motores [33], [34], [35].

Dispositivo	Tipo	Corriente maxima	Precisión	Voltaje	Aplicación	Compatibilidad
TB6560	Driver de motor paso a paso	0.5-3.5A (3.5A máximo)	Alta precisión en microstepping 1/16	10-35V	Control de motores para sistemas automatizados en hidroponía (válvulas, bandejas móviles, etc.)	NEMA 17, NEMA 23
TB6600	Driver de motor paso a paso	4.5A máximo	Alta precisión en microstepping 1/16	10-45V	Automatización de motores en procesos de hidroponía y otras aplicaciones agrícolas	NEMA 17, NEMA 23, NEMA 34
DRV8825	Driver de motor paso a paso	2.2A máximo	Alta precisión en control de posición microstepping 1/32	8.2-35V	Automatización ligera, sistemas pequeños de movimiento	NEMA 17

Tabla 11. Microcontroladores [36], [37], [38].

Elemento	Arquitectura	Velocidad del CPU	Memoria RAM	Flash Espacio	Conexión inalambrica	Periféricos	Usos Comunes
ESP32	Xtensa LX6 (doble núcleo)	Hasta 240 MHz	520 KB SRAM	Hasta 16 MB (externa)	Wi-Fi 802.11 b/g/n, Bluetooth v4.2	ADC de 12 bits, DAC de 8 bits, UART, SPI, I2C	IoT, domótica, dispositivos portátiles
ESP8266	Tensilica L106 (single core)	80/160 MHz	160 KB SRAM	Hasta 16 MB (externa)	Wi-Fi 802.11 b/g/n	GPIO, PWM, ADC de 10 bits, UART, SPI, I2C	IoT, dispositivos conectados, sistema automatico.
STM32	ARM Cortex-M (varía según modelo)	72 MHz (STM32F103)	20 KB SRAM (STM32F103)	64 KB Flash (STM32F103)	No dispone	ADC de 12 bits, GPIO programables, UART, SPI, I2C, CAN, USB	Industriales, sistemas embebidos, sistema automático.

Tabla 12. Microprocesadores [39], [40], [41].

Dispositivo	CPU	GPU	Memoria RAM	Almacenamiento	Conectividad	Puertos
Raspberry Pi 4	Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GH	Broadcom VideoCore VI @ 500 MHz	2GB, 4GB u 8GB LPDDR4-3200 SDRAM	Ranura para tarjeta Micro-SD	Wi-Fi 802.11ac de doble banda, Bluetooth 5.0, Gigabit Ethernet	2x USB 3.0, 2x USB 2.0, 2x Micro-HDMI (4K @ 60Hz), GPIO de 40 pines

Dispositivo	CPU	GPU	Memoria RAM	Almacenamiento	Conectividad	Puertos
Raspberry Pi 5	Broadcom BCM2712, Quad-core Cortex-A76 (ARM v8.2) 64-bit SoC @ 2.4GHz	Broadcom VideoCore VII @ 800 MHz	4GB u 8GB LPDDR4X-4267 SDRAM	Ranura para tarjeta Micro-SD, soporte para SSD M.2 (con HAT opcional)	Wi-Fi 802.11ac de doble banda, Bluetooth 5.2, Gigabit Ethernet (PoE habilitado con HAT)	2x USB 3.0, 2x USB 2.0, 2x Micro-HDMI (4K @ 60Hz en doble pantalla), GPIO de 40 pines
NVIDIA Jetson Nano	Quad-core ARM Cortex-A57 MPCore Processor @ 1.43 GHz	NVIDIA Maxwell GPU con 128 núcleos CUDA	4GB LPDDR4	Tarjeta Micro-SD	Gigabit Ethernet	4x USB 3.0, HDMI 2.0, DisplayPort 1.2, GPIO de 40 pines

Tabla 13. Cámaras [42], [43], [44].

Nombre de la Cámara	Resolución	Sensor	Características Destacadas	Compatibilidad
Cámara para Raspberry Pi 5MPX 1080P v1.3	5 MP (2592x1944)	OmniVision OV5647	Video Full HD 1080p a 30 fps, enfoque fijo	Raspberry Pi 4, Raspberry Pi 5, NVIDIA Jetson Nano
Raspberry Pi Camera Module V2	8 MP (3280x2464)	Sony IMX219	Video 1080p a 30 fps, lente fija	Raspberry Pi 4, Raspberry Pi 5, NVIDIA Jetson Nano
Raspberry Pi High Quality Camera	12.3 MP	Sony IMX477	Lentes intercambiables (montura CS/C), alta sensibilidad a la luz	Raspberry Pi 4, Raspberry Pi 5, NVIDIA Jetson Nano

d. Lenguajes de programación

Para la programación de los microcontroladores se tiene los diferentes lenguajes de programación mostrado en Tabla 14.

Tabla 14. Lenguajes de programación para el microcontrolador [45], [46], [47].

Lenguaje de Programación	Descripción	Ventajas	Limitaciones	Casos de uso
C/C++ (Arduino IDE)	Lenguaje de programación más común para la ESP32, especialmente utilizando el entorno de desarrollo de Arduino.	Eficiencia de hardware.	Curva de aprendizaje alta para principiantes. Gestión manual de memoria.	Control de hardware de bajo nivel, sistemas de tiempo real.

Lenguaje de Programación	Descripción	Ventajas	Limitaciones	Casos de uso
MicroPython	Versión ligera de Python optimizada para microcontroladores. Facilita la escritura rápida de código en la ESP32.	Facilidad de uso. Gran comunidad de usuarios. Ideal para prototipos rápidos.	Menor rendimiento que C/C++. Limitado en cuanto a control de hardware avanzado	Prototipos, IoT, proyectos educativos.
Lua	Lenguaje ligero, utilizado especialmente en el firmware NodeMCU para control de la ESP32.	Simplicidad de uso. Buena para scripts pequeños y simples.	Rendimiento limitado en comparación con C/C++. Comunidad más pequeña.	Proyectos sencillos, IoT, automatización.

e. Protocolos de comunicación

En el ámbito de Internet de las Cosas (IoT), la selección del protocolo de comunicación adecuado es crucial para garantizar una transmisión eficiente, segura y adaptable a las necesidades del sistema. Cada protocolo presenta diferencias en términos de modelo de comunicación, transporte de datos, consumo energético, latencia y seguridad, lo que los hace más o menos adecuados según la aplicación específica.

La Tabla 15 compara los protocolos más utilizados en IoT, como MQTT, CoAP, HTTP, AMQP, DDS y LoRaWAN, considerando aspectos clave como tipo de transporte, formato de datos, eficiencia energética, tiempos de respuesta y seguridad. Esta comparación permite identificar las ventajas y limitaciones de cada protocolo, facilitando la selección óptima según los requerimientos del sistema implementado.

Tabla 15. Comparativas de protocolos de comunicación [48].

Protocolo	Transporte/Formato	Consumo/Latencia	Seguridad/Uso
MQTT	TCP/Binario	Bajo/Baja	TLS/SSL - Sensores, uso de domótica
CoAP	UDP/Texto	Muy bajo/Muy baja	DTLS - IoT de baja potencia
HTTP	TCP/Texto	Alto/Alta	TLS/SSL – Web, nube
AMQP	TCP/Binario	Medio/Media	TLS/SSL – Empresarial, financiero
DDS	UDP-TCP/Binario	Medio/Media	Propietaria -Tiempo real, automatización de procesos

MQTT es la mejor opción para comunicaciones IoT debido a su eficiencia en ancho de banda, bajo consumo energético y alta fiabilidad. A diferencia de HTTP, que requiere múltiples solicitudes, MQTT utiliza un modelo publicación/suscripción, reduciendo la carga de red. Frente a CoAP, ofrece mayor escalabilidad y compatibilidad con plataformas en la nube como AWS IoT y Google Cloud IoT. Además, su compatibilidad con TLS/SSL lo hace más seguro que protocolos como LoRaWAN, que está limitado a redes de baja velocidad. Su latencia reducida, junto con los niveles de Calidad de Servicio (QoS), garantiza una transmisión confiable incluso en entornos inestables, superando a AMQP y DDS, que requieren mayor infraestructura.

1.3.4 Edge Cloud

En el ámbito de la tecnología moderna, la computación distribuida y en la nube se ha convertido en un pilar fundamental para satisfacer las crecientes demandas de procesamiento, almacenamiento y conectividad en aplicaciones industriales, empresariales y de consumo. Diferentes arquitecturas han emergido para abordar estos desafíos, cada una con características únicas que las hacen ideales para casos de uso específicos. Entre estas, destacan Edge Cloud, Cloud Computing, Fog Computing, Edge Computing y Hybrid Cloud. Estas arquitecturas permiten optimizar el rendimiento, la latencia y la escalabilidad según las necesidades de los usuarios finales y las organizaciones.

La Tabla 16 compara estas arquitecturas clave, destacando sus principales diferencias y casos de aplicación.

Tabla 16. Arquitecturas existentes [49], [50], [51], [52], [53].

Aspecto	**Edge Cloud**	**Cloud Computing**	**Fog Computing**	**Edge Computing**	**Hybrid Cloud**
Ubicación del procesamiento	Procesamiento en el borde con soporte de la nube.	Procesamiento centralizado en grandes centros de datos.	Procesamiento distribuido en múltiples niveles entre el borde y la nube.	Procesamiento completamente en el borde.	Procesamiento en nubes públicas y privadas.
Latencia	Baja, ya que los datos se procesan cerca del usuario.	Mayor, debido a la distancia entre usuarios y centros de datos.	Baja, al realizar procesamiento local o regional.	Muy baja, ya que opera en dispositivos locales.	Depende del diseño de las redes híbridas.

Aspecto	Edge Cloud	Cloud Computing	Fog Computing	Edge Computing	Hybrid Cloud
Casos de uso	IoT, gaming en tiempo real, ciudades inteligentes.	Análisis de datos masivos, almacenamiento a gran escala.	IoT industrial, redes inteligentes, análisis distribuido.	Sistemas críticos locales, sensores IoT.	Escenarios empresariales con seguridad y flexibilidad.
Conectividad	Requiere conexión a la nube para escalabilidad.	Necesita conexión a internet constante.	Puede operar localmente y conectarse a la nube según sea necesario.	Puede operar sin conexión a internet.	Integra servicios conectados y desconectados.
Escalabilidad	Limitada por los nodos Edge, pero optimizada por la nube.	Alta, con recursos virtualizados en la nube.	Escalabilidad intermedia, según la capacidad de los niveles jerárquicos.	Limitada a los recursos del borde.	Alta, combinando recursos públicos y privados.
Complejidad	Menor, con integración directa entre borde y nube.	Baja, diseño centralizado simplificado.	Alta, debido a múltiples niveles de infraestructura.	Baja, con enfoque en dispositivos individuales.	Intermedia, depende de la integración de nubes.
Almacenamiento	En el borde y la nube.	Centralizado en centros de datos.	Local, regional y en la nube.	Local en dispositivos Edge.	Distribuido entre nubes públicas y privadas.

La arquitectura Edge-Cloud proporciona una plataforma eficiente para procesar y analizar grandes volúmenes de datos de forma local (edge) y en el borde de la nube, facilitando decisiones rápidas y precisas, esenciales en el entorno agrícola. El uso de la arquitectura Edge-Cloud en la agricultura está ganando relevancia debido a su capacidad para procesar y analizar grandes volúmenes de datos en tiempo real, mejorando la eficiencia y productividad del sector.

La arquitectura Edge-Cloud combina el procesamiento local en dispositivos periféricos con la computación en la nube para optimizar la gestión de datos y la toma de decisiones en tiempo real. Mientras que la nube central prioriza la uniformidad y el almacenamiento masivo, el procesamiento en el borde reduce la latencia y permite respuestas inmediatas en aplicaciones críticas.

Además de casos del mundo real, la creciente adopción de microservicios ha fortalecido la relación entre la computación en la nube y el borde. Los microservicios son aplicaciones mínimas diseñadas para ejecutarse en contenedores portátiles, lo que permite a los desarrolladores decidir estratégicamente qué procesos deben ejecutarse en la nube o en el borde.

Esta arquitectura es clave en entornos como la agricultura inteligente y la salud, donde el procesamiento local mejora la eficiencia operativa. Además, el uso de microservicios y la expansión de tecnologías como 5G están impulsando su adopción, permitiendo sistemas más escalables y flexibles. Su implementación no solo optimiza el rendimiento, sino que también facilita la integración de dispositivos avanzados en diversas industrias [54].

a. Impacto de la Arquitectura Edge-Cloud en la Agricultura

- ***Procesamiento de Datos en Tiempo Real:*** La integración de tecnologías avanzadas en la agricultura, como la computación en la nube, la computación en niebla y la computación en la orilla, ha revolucionado la industria agrícola. Estas tecnologías permiten a los agricultores optimizar sus prácticas agrícolas, aumentar la producción y mejorar la calidad de los cultivos, mientras reducen los costos y minimizan el impacto ambiental.

Uno de los beneficios clave de estas tecnologías es el procesamiento de datos en tiempo real. El Edge-Cloud permite a los agricultores procesar grandes volúmenes de datos generados por dispositivos IoT en tiempo real, lo que les permite tomar decisiones informadas rápidas. Por ejemplo, al analizar datos actuales de sensores de humedad y temperatura del suelo, los agricultores pueden ajustar el riego y la fertilización de manera eficiente, lo que mejora la eficiencia en el uso de recursos y aumenta el rendimiento de los cultivos.

Esta capacidad de procesamiento en tiempo real también permite a los agricultores monitorear la salud de sus cultivos y animales de granja, lo que les permite tomar medidas preventivas en caso de problemas o enfermedades. Esto puede ayudar a reducir la pérdida de cultivos y a garantizar la seguridad alimentaria. En general, la integración de tecnologías avanzadas en la agricultura, junto con el procesamiento de datos en tiempo real, es clave para garantizar un futuro sostenible y resiliente para la industria agrícola y el medio ambiente [55].

- ***Eficiencia en Costos y Escalabilidad:*** La arquitectura Edge-Cloud ofrece un modelo de procesamiento de datos más eficiente en costos. Al reducir el volumen de datos que se transmite a la nube mediante el procesamiento local en el borde, se

disminuyen los costos de transmisión y almacenamiento. Además, esta arquitectura es altamente escalable, permitiendo la integración de nuevas tecnologías y dispositivos según crezcan las necesidades agrícolas [56].

- ***Mejora en la Confiabilidad y Resiliencia:*** Al procesar los datos en el borde de la red, los dispositivos pueden seguir operando de manera autónoma incluso cuando la conexión con la nube se interrumpe. Esto garantiza que procesos críticos, como la irrigación automática o el control de plagas, sigan funcionando sin interrupciones, minimizando las pérdidas y asegurando la continuidad operativa [57].

- ***Seguridad y Privacidad de los Datos:*** La arquitectura Edge-Cloud ofrece una solución clave para garantizar la seguridad y privacidad de los datos en la agricultura inteligente, permitiendo la gestión local y descentralizada de los datos y la implementación de medidas de seguridad adicionales para proteger la información sensible. Al procesar los datos localmente en el borde de la red, los dispositivos pueden realizar análisis y procesamiento de datos sin enviar la información sensible a la nube, reduciendo así el riesgo de seguridad y privacidad asociado con la transmisión de datos sensibles a través de la red.

La descentralización de la arquitectura Edge-Cloud permite una mayor confiabilidad y resiliencia en la gestión de la seguridad y privacidad de los datos, ya que los dispositivos locales pueden continuar funcionando incluso en caso de interrupciones en la conectividad. Además, la arquitectura Edge-Cloud también permite la implementación de medidas de seguridad y privacidad adicionales, como la encriptación de datos, la autenticación y autorización de usuarios y dispositivos, y la implementación de políticas de seguridad y privacidad robustas.

En resumen, la arquitectura Edge-Cloud es una solución clave para garantizar la seguridad y privacidad de los datos en la agricultura inteligente, permitiendo la gestión local y descentralizada de los datos y la implementación de medidas de seguridad adicionales para proteger la información sensible. La descentralización y las medidas de seguridad adicionales proporcionan una mayor confiabilidad y resiliencia en la gestión de la seguridad y privacidad de los datos, lo que es esencial para garantizar la eficacia y confiabilidad de los sistemas de agricultura inteligente [58].

La Tabla 17 presenta el resumen de mencionando anteriormente.

Tabla 17. Impacto de Edge Cloud en la Arquitectura.

Impacto	**Descripción**
Procesamiento de Datos en Tiempo Real	Integración de Edge-Cloud permite procesar datos generados por IoT en tiempo real, optimizando recursos, aumentando rendimiento y permitiendo monitorear la salud de cultivos y animales. Ayuda a tomar decisiones rápidas y eficientes.
Eficiencia en Costos y Escalabilidad	Reduce costos de transmisión y almacenamiento al procesar datos localmente. Arquitectura altamente escalable para integrar nuevas tecnologías y dispositivos según crezcan las necesidades.
Mejora en Confiabilidad y Resiliencia	Los dispositivos pueden operar autónomamente durante interrupciones en la conexión a la nube, garantizando procesos críticos como irrigación y control de plagas.
Seguridad y Privacidad de Datos	Gestión local de datos en el borde con medidas como encriptación, autenticación y descentralización, asegurando privacidad y reduciendo riesgos en la transmisión.

b. Características de la Arquitectura Edge-Cloud en la Agricultura

- ***Proximidad a la Fuente de Datos:*** La capacidad de procesar datos en el borde de la red, cerca de la fuente de generación, permite que las decisiones basadas en datos se tomen casi instantáneamente. Esto es fundamental en la agricultura para responder rápidamente a cambios en las condiciones ambientales, como variaciones de temperatura o humedad.

Además, la proximidad a la fuente de datos también reduce la carga de datos en la nube, lo que puede ayudar a reducir los costos y mejorar la eficiencia de la gestión de los datos. En resumen, la proximidad a la fuente de datos es una característica clave de la arquitectura Edge-Cloud que puede mejorar la eficacia y confiabilidad de los sistemas de agricultura inteligente [58].

La Figura 8 presenta un modelo de computación distribuida con capas de borde, niebla y nube para la agricultura inteligente. La capa de nube se encarga del almacenamiento masivo de datos y el análisis a gran escala, además de cargar algoritmos en los nodos de niebla y almacenar datos de respaldo para análisis futuros. La capa de niebla, ubicada en las granjas, realiza análisis en tiempo real como predicción de plagas, rendimiento y clima, así como la automatización del monitoreo agrícola. Finalmente, la capa de borde incluye dispositivos como sensores y tractores, responsables de la recopilación y transmisión de datos hacia la capa de niebla [55].

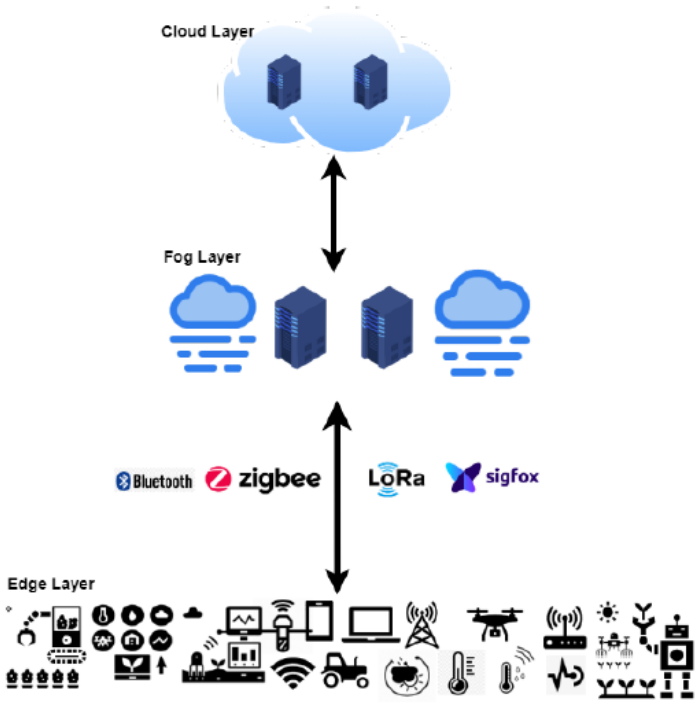


Figura 8. Arquitectura de tres capas para agricultura inteligente [55].

- ***Computación y Almacenamiento Distribuidos:*** La arquitectura Edge-Cloud permite la computación y el almacenamiento distribuidos, lo que facilita la gestión de grandes cantidades de datos agrícolas. Esta capacidad es esencial para aplicaciones avanzadas como el análisis predictivo y el aprendizaje automático, que ayudan a optimizar prácticas agrícolas y aumentar la productividad de los cultivos.

La arquitectura Edge-Cloud permite la computación y el almacenamiento distribuidos, facilitando la gestión de grandes cantidades de datos agrícolas. Esta capacidad es fundamental para aplicaciones avanzadas como el análisis predictivo y el aprendizaje automático, que ayudan a optimizar las prácticas agrícolas y aumentar la productividad de los cultivos. Al distribuir el procesamiento de datos entre los dispositivos locales (borde), la niebla y la nube, se mejora la eficiencia y se reduce la latencia, permitiendo respuestas más rápidas y decisiones basadas en datos en tiempo real [59].

La Figura 9 describe una red que incluye múltiples usuarios finales (UE), un punto de acceso WiFi como nodo de niebla y un servidor en la nube. Los UE se conectan al nodo de niebla vía enlace inalámbrico, mientras que la conexión entre el nodo de niebla y la nube es por fibra óptica. Cada UE puede manejar aplicaciones localmente o descargarlas para su procesamiento en la niebla o en la nube.

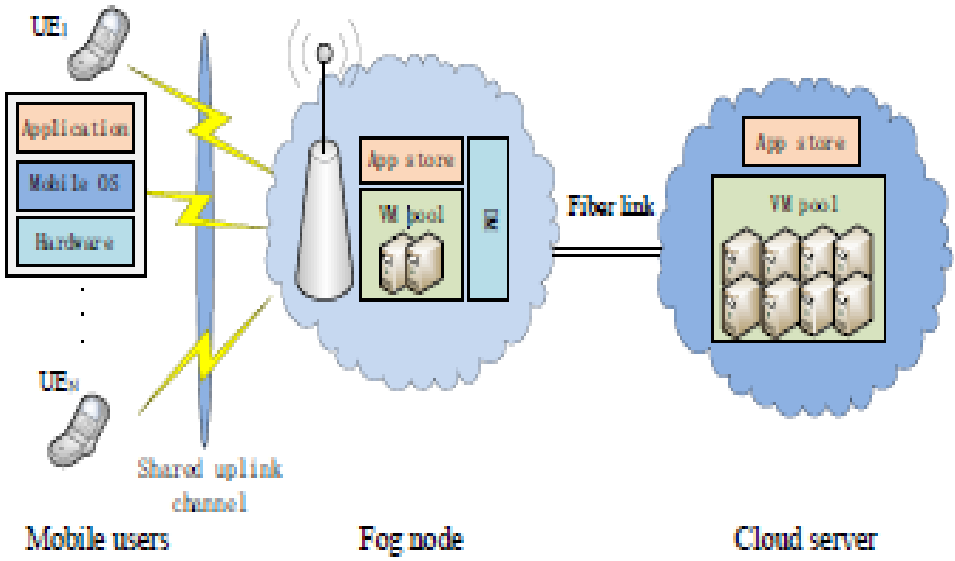


Figura 9. Topología del sistema de red [59].

- ***Escalabilidad y Flexibilidad:*** El Edge-Cloud es altamente escalable, permitiendo adaptarse a condiciones cambiantes y demandas crecientes. La idea de distribuir tareas entre nodos cercanos también implica una arquitectura escalable que puede adaptarse a un aumento en la demanda de procesamiento.

La capacidad de realizar análisis en nodos de borde para manejar datos en tiempo real implica flexibilidad. El sistema puede adaptarse a diferentes condiciones y requerimientos sin depender completamente de una infraestructura centralizada. Durante la temporada de cosecha, por ejemplo, se pueden desplegar nodos adicionales para gestionar el aumento de datos provenientes de equipos de cosecha y sensores, mejorando la eficiencia y la capacidad de respuesta del sistema.

Muchas aplicaciones en la nube, impulsadas por los usuarios, han creado oportunidades para el análisis de grandes volúmenes de datos. Sin embargo, al utilizar la nube como servidor centralizado, se incrementa la comunicación entre los dispositivos de borde (como teléfonos inteligentes y wearables) y centros de datos remotos. Esto resulta ineficaz para aplicaciones que requieren respuestas en tiempo real. Por ello, ha surgido la necesidad de recurrir a la computación en el borde, que realiza procesamiento cerca de los usuarios, reduciendo la latencia y mejorando la eficiencia en dichas aplicaciones., la Figura 10 muestra lo manifestado [60].

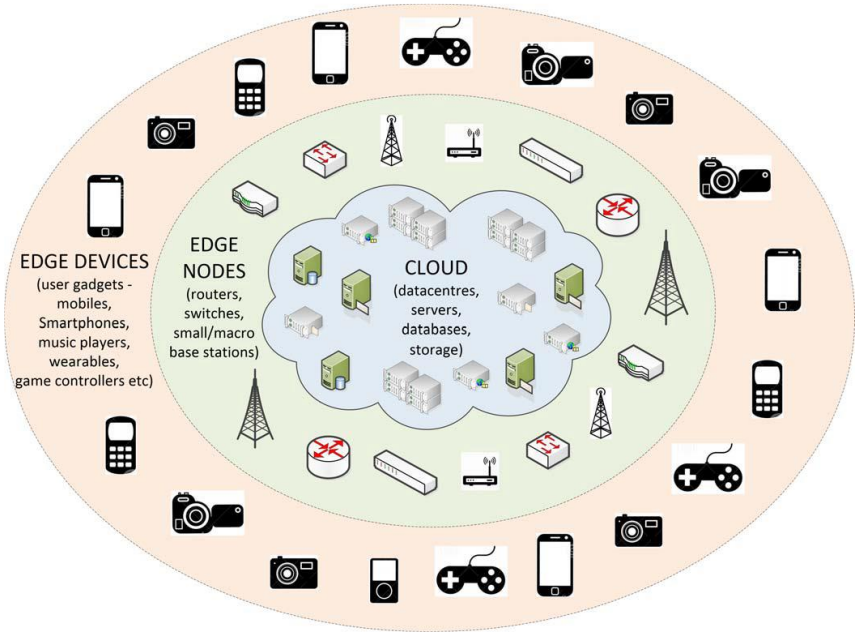


Figura 10. Dispositivos y nodos de borde en relación con la nube [60].

La Tabla 18 muestra el resumen de lo anteriormente mencionado.

Tabla 18. Características Edge Cloud en la agricultura

Características	**Descripción**
Proximidad a la Fuente de Datos	Procesamiento cercano a la fuente permite decisiones inmediatas ante cambios ambientales. Reduce carga de datos en la nube, mejorando la gestión y reduciendo costos.
Computación y Almacenamiento Distribuidos	Gestión eficiente de grandes volúmenes de datos agrícolas. Facilita aplicaciones como análisis predictivo y aprendizaje automático para optimizar prácticas agrícolas y mejorar productividad.
Escalabilidad y Flexibilidad	Permite adaptarse a condiciones cambiantes y demandas crecientes. Posibilidad de desplegar nodos adicionales durante temporadas de alta demanda para mejorar capacidad de respuesta.
Proximidad a la Fuente de Datos	Procesamiento cercano a la fuente permite decisiones inmediatas ante cambios ambientales. Reduce carga de datos en la nube, mejorando la gestión y reduciendo costos.

c. Base de datos

Las bases de datos son una parte fundamental de cualquier aplicación moderna, ya que permiten el almacenamiento, gestión y recuperación eficiente de datos. Con una variedad de opciones disponibles, cada tipo de base de datos ofrece características

únicas que se adaptan a diferentes necesidades y requisitos. En esta sección, se presenta una tabla comparativa de diversas bases de datos, incluyendo tanto sistemas de gestión de bases de datos relacionales como NoSQL. Esta comparación proporciona una visión general de las capacidades, rendimiento y casos de uso de cada base de datos, facilitando la elección de la solución más adecuada para proyectos específicos. La Tabla 19 presenta las bases de datos más comunes y sus usos.

Tabla 19. Comparativa de bases de datos [61].

Base de Datos	**Descripción**	**Tipo de Datos**	**Rendimiento**	**Casos de Uso**
Mysql	Base de datos relacional de código abierto, utilizada para aplicaciones web y análisis de datos.	Estructurada	Alto rendimiento en consultas SQL complejas.	Aplicaciones web, sistemas de gestión de contenido, análisis de datos.
MongoDB	Base de datos NoSQL orientada a documentos, permite almacenamiento flexible de datos JSON.	No estructurada	Muy alto rendimiento en operaciones de lectura/escritura.	Aplicaciones que requieren alta escalabilidad, datos no estructurados, almacenamiento de documentos.
PostgreSQL	Base de datos relacional de código abierto con soporte para SQL avanzado y datos no estructurados.	Estructurada y no estructurada	Alto rendimiento, especialmente en consultas complejas.	Aplicaciones web, análisis de datos, sistemas de información geográfica.

Dando grandes beneficios la base de datos basado en Mysql ya que ayuda mejor en la gestión de datos.

1.3.5 Inteligencia artificial

La inteligencia artificial (IA) puede entenderse como la capacidad de las máquinas para realizar actividades que normalmente requieren inteligencia humana. De forma más precisa, la IA utiliza algoritmos para aprender de los datos y aplicar ese aprendizaje en la toma de decisiones, imitando el razonamiento humano. A diferencia de las personas, los sistemas de IA pueden procesar grandes volúmenes de información de forma continua, sin necesidad de descanso, y con una menor tasa de errores en comparación con los humanos que realizan las mismas tareas.

Lo destacable de la IA es su capacidad para aprender y tomar decisiones, características que están en constante desarrollo y expansión. Esto ha permitido que las máquinas asuman funciones que antes eran exclusivamente humanas, mejorando la eficiencia en diversos ámbitos de la vida cotidiana y profesional.

Sin embargo, aunque estas tecnologías ofrecen numerosos beneficios, como mayor productividad y precisión, su rápido crecimiento también plantea la necesidad de considerar y mitigar posibles consecuencias negativas, tanto directas como indirectas, que podrían surgir de su implementación masiva [62].

El aprendizaje automático, conocido en inglés como Machine Learning, es uno de los pilares fundamentales de la inteligencia artificial. En términos sencillos, se refiere a una rama de la informática en la que las máquinas son capaces de adquirir conocimientos o habilidades sin necesidad de ser programadas específicamente para ello. Un ejemplo común de su aplicación son las recomendaciones o predicciones que los sistemas generan en contextos particulares.

El aprendizaje automático se basa en el uso de algoritmos para identificar y aprender de los patrones presentes en los datos. Un ejemplo sencillo de esta tecnología es el funcionamiento de los filtros de spam en los correos electrónicos, que analizan y detectan mensajes no deseados para clasificarlos y diferenciarlos de los correos legítimos. Este caso ilustra cómo los algoritmos pueden emplearse para reconocer patrones y aplicar el conocimiento obtenido en la toma de decisiones automatizadas. La Figura 11 muestra los distintos tipos de aprendizaje autónomo.



Figura 11. Tipos de aprendizaje automático [62].

El aprendizaje supervisado, no supervisado y por refuerzo son tres enfoques clave en el aprendizaje automático:

- Aprendizaje supervisado: Los algoritmos se entrenan con datos etiquetados previamente, lo que les permite categorizar nueva información basándose en patrones conocidos. Este método requiere intervención humana para proporcionar ejemplos y retroalimentación. Por ejemplo, un algoritmo podría aprender a identificar imágenes de gatos tras ser entrenado con fotos etiquetadas previamente.

- Aprendizaje no supervisado: Aquí, los algoritmos trabajan con datos no etiquetados, descubriendo patrones y categorizaciones por sí mismos, sin intervención humana. En el caso de las fotos, el algoritmo encontraría por su cuenta cómo agrupar las imágenes de gatos en una categoría.

- Aprendizaje por refuerzo: Los algoritmos aprenden a través de la experiencia, recibiendo recompensas o refuerzos positivos cuando toman decisiones correctas. Este proceso es similar al entrenamiento de perros, donde se les premia cuando realizan la acción deseada, como sentarse.

1.3.6 Deep learning

El Deep Learning, una rama avanzada del aprendizaje automático se destaca como una de las aplicaciones más impactantes y en rápido crecimiento de la inteligencia artificial. Este enfoque utiliza redes neuronales organizadas en capas para identificar patrones y relaciones complejas dentro de grandes volúmenes de datos. Para funcionar, requiere cantidades masivas de información y una alta capacidad de procesamiento.

Sus aplicaciones incluyen reconocimiento de voz, procesamiento del lenguaje natural, visión artificial y sistemas avanzados para la conducción asistida. Aunque el término "inteligencia artificial" se utiliza de manera general en muchos contextos, a menudo hace referencia específicamente al aprendizaje profundo o al aprendizaje automático. Sin embargo, se emplea el término amplio para simplificar su uso [62].

El Deep Learning, una rama del aprendizaje automático ha demostrado un gran potencial para mejorar la toma de decisiones mediante el análisis avanzado de patrones complejos y la automatización de procesos. El Deep Learning, una rama avanzada del aprendizaje automático ha transformado la agricultura moderna al permitir la creación de modelos predictivos y de clasificación que pueden analizar datos complejos con gran precisión. Esto ha llevado a mejoras significativas en la gestión de cultivos, la detección de enfermedades y plagas, y la optimización de recursos.

a. Impacto del Deep learning en la Agricultura

El Deep Learning ha tenido un impacto significativo en la agricultura, revolucionando la manera en que los datos se procesan y cómo se toman decisiones a partir de ellos. A continuación, se describen los principales impactos de esta tecnología.

- ***Incremento en la Productividad Agrícola:*** El uso del aprendizaje profundo en la agricultura ha permitido automatizar diversas tareas que tradicionalmente eran manuales, como la detección de enfermedades en los cultivos y la clasificación de frutos. Las redes neuronales convolucionales (CNN) se utilizan para analizar imágenes de cultivos, lo que permite una detección temprana de plagas y enfermedades. Esto reduce la necesidad de inspección humana y permite a los agricultores tomar decisiones más rápidas, lo que aumenta la eficiencia y la productividad de los cultivos [63].

Además, los sistemas de Deep Learning son capaces de prever el rendimiento de los cultivos mediante el análisis de datos históricos y climáticos, lo que permite a los agricultores planificar mejor sus recursos. Por ejemplo, las Redes Neuronales Recurrentes (RNN) pueden modelar series temporales para predecir el crecimiento de los cultivos y optimizar las fechas de siembra y cosecha [64].

- ***Optimización del uso de recursos:*** El Deep Learning también ha permitido optimizar el uso de recursos en la agricultura, lo que se traduce en una mayor sostenibilidad. Modelos avanzados de DL pueden predecir las necesidades de agua, fertilizantes y pesticidas de los cultivos en tiempo real. Esto ha llevado a la creación de sistemas de riego y fertilización de precisión, que aplican solo la cantidad exacta de

recursos necesarios en cada etapa del crecimiento, reduciendo el desperdicio y los costos.

Además, el Deep Learning puede ser utilizado para analizar grandes cantidades de datos de imágenes y sensores para detectar enfermedades y plagas en las plantas, lo que permite implementar estrategias de control preventivo y reducir la cantidad de pesticidas utilizados. Esto no solo beneficia al medio ambiente, sino que también aumenta la productividad y la calidad de las cosechas.

La tecnología del Deep Learning también puede ser utilizada para crear modelos de crecimiento y desarrollo de las plantas, lo que permite a los agricultores anticipar y prepararse para los desafíos que pueden enfrentar en sus cultivos. Esto puede incluir la predicción del clima, la disponibilidad de recursos y la presencia de plagas y enfermedades [65].

La Tabla 20 muestra el resumen de la información presentada.

Tabla 20. Impacto de Deep Learning en la Agricultura.

Impacto	**Descripción**
Incremento en la Productividad Agrícola	Automatización de tareas como la detección de enfermedades y clasificación de frutos mediante redes neuronales convolucionales (CNN). Predicción del rendimiento de cultivos utilizando Redes Neuronales Recurrentes (RNN) para optimizar recursos.
Optimización del Uso de Recursos	Sistemas de riego y fertilización de precisión basados en modelos avanzados de Deep Learning. Análisis de datos para reducir el uso de pesticidas y optimizar recursos, beneficiando al medio ambiente y mejorando la calidad de las cosechas.

b. Características del Deep Learning en Agricultura

- ***Capacidad de Aprendizaje Supervisado y No Supervisado:*** El aprendizaje automático, una rama fundamental de la inteligencia artificial, se divide principalmente en dos enfoques: el aprendizaje supervisado y el no supervisado. El aprendizaje supervisado trabaja con datos etiquetados, donde se proporcionan tanto las variables de entrada como las de salida. Este método busca optimizar los parámetros del modelo para predecir con precisión los resultados basándose en una función de pérdida, y se utiliza comúnmente en tareas de clasificación y regresión.

Por otro lado, el aprendizaje no supervisado opera con datos sin etiquetar, centrándose en descubrir patrones o estructuras ocultas en los datos. Este enfoque incluye técnicas como el análisis de componentes principales (PCA), la agrupación k-means y el t-SNE, y utiliza diversas funciones de pérdida, como la reconstrucción de datos. Mientras que el aprendizaje supervisado se enfoca en la predicción basada en datos etiquetados, el no supervisado busca revelar estructuras intrínsecas en los datos sin una variable objetivo específico. Ambos métodos son cruciales en el campo del aprendizaje automático, ofreciendo diferentes perspectivas y soluciones para el análisis y procesamiento de datos [66].

- ***Adaptabilidad y Personalización de Modelos:*** Los modelos de deep learning pueden adaptarse a diferentes cultivos y enfermedades mediante el uso de técnicas como la transferencia de aprendizaje, donde un modelo preentrenado se ajusta con un nuevo conjunto de datos específico del problema agrícola. Esto reduce el tiempo y el costo de desarrollo de nuevos modelos para diferentes escenarios agrícolas.

La Figura 12 muestra una arquitectura AlexNet, entrenada para identificar enfermedades en plantas, activa de manera eficiente patrones relacionados con manchas enfermas en una hoja afectada por la roya del cedro de la manzana. En el panel A, se muestra una imagen de una hoja afectada, obtenida de una búsqueda en Bing con palabras clave específicas. El panel B ilustra las activaciones en la primera capa de convolución (conv1) del modelo AlexNet cuando se procesa la imagen, demostrando que el modelo ha aprendido a reconocer estas áreas enfermas [67].

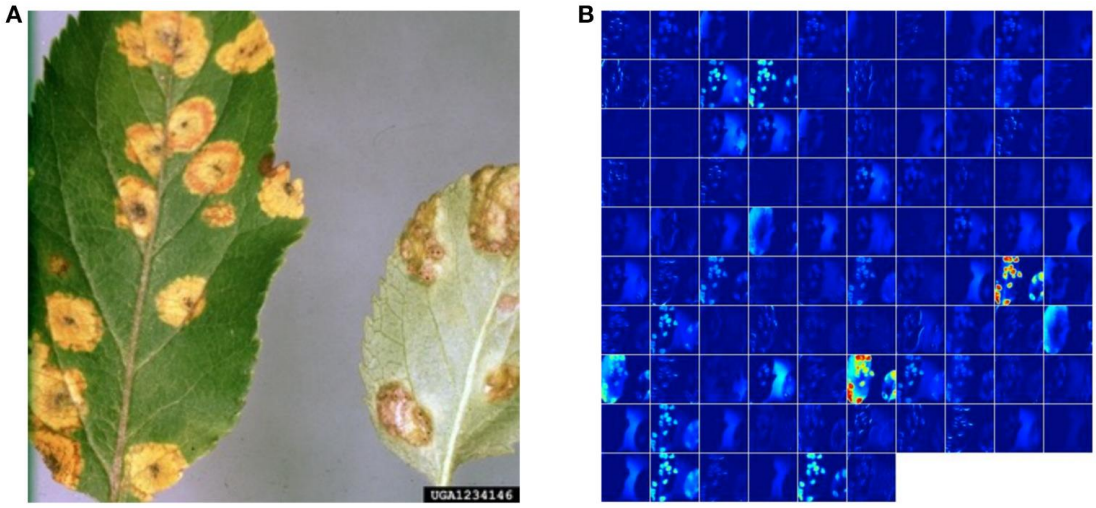


Figura 12. Modelo AlexNet [67].

- ***Adaptabilidad a diferentes escenarios agrícolas:*** Finalmente, una de las características más valiosas del Deep Learning en la agricultura es su capacidad para adaptarse a diferentes entornos y condiciones. Los modelos de DL pueden entrenarse para reconocer patrones específicos de cultivos en distintas regiones geográficas y climas, lo que les permite ser útiles en una amplia variedad de escenarios agrícolas. Esta adaptabilidad es clave para su éxito en una industria tan diversa como la agricultura [65].

La Tabla 21 muestra el resumen de las características presentadas.

Tabla 21. Características de Deep Learning.

Características	**Descripción**
Capacidad de Aprendizaje Supervisado y No Supervisado	Aprendizaje supervisado para tareas de clasificación y regresión con datos etiquetados. Aprendizaje no supervisado para descubrir patrones ocultos en datos no etiquetados, aplicando técnicas como PCA y k-means.
Adaptabilidad y Personalización de Modelos	Uso de técnicas como la transferencia de aprendizaje para adaptar modelos preentrenados a problemas agrícolas específicos, reduciendo tiempo y costos. Por ejemplo, modelos como AlexNet identifican enfermedades en cultivos de forma eficiente.
Adaptabilidad a Diferentes Escenarios Agrícolas	Capacidad de entrenar modelos para patrones específicos en diversas regiones geográficas y condiciones climáticas, haciendo que el Deep Learning sea aplicable a una amplia variedad de escenarios en la agricultura.
Capacidad de Aprendizaje Supervisado y No Supervisado	Aprendizaje supervisado para tareas de clasificación y regresión con datos etiquetados. Aprendizaje no supervisado para descubrir patrones ocultos en datos no etiquetados, aplicando técnicas como PCA y k-means.

c. Tecnologías de Deep learning

El desarrollo e implementación de proyectos de Deep Learning requieren un enfoque metodológico que permita abordar los retos técnicos y prácticos de manera eficiente. Estas metodologías garantizan que los proyectos sean reproducibles, escalables y alineados con los objetivos planteados. Dependiendo de los recursos disponibles, la complejidad del problema y los requisitos del entorno de producción, se pueden aplicar diversas estrategias que abarcan desde la preparación de datos hasta el monitoreo continuo de los modelos en producción. La Tabla 22 resumen tres metodologías ampliamente utilizadas para proyectos de Machine Learning y Deep Learning, cada una diseñada para abordar distintos aspectos del ciclo de vida de un modelo, desde su conceptualización hasta su despliegue en sistemas reales.

Tabla 22. Metodologías [68], [69], [70].

Metodología	**Descripción**
CRISP-DM	Un enfoque estructurado basado en seis etapas: entender el problema, preparar los datos, modelar, evaluar, implementar y monitorear el modelo en producción.
Agile AI	Adapta principios ágiles al Deep Learning, trabajando con iteraciones rápidas, validación continua, desarrollo incremental y retroalimentación constante para mejorar el modelo.
Pipeline	Enfoque lineal que abarca desde la definición del problema hasta el despliegue, pasando por preparación de datos, diseño, entrenamiento, evaluación, optimización y monitoreo.

Para el lenguaje de programación para la inteligencia artificial se muestra en la Tabla 23.

Tabla 23. Lenguajes de programación para inteligencia artificial [71].

Lenguaje de Programación	**Descripción**	**Rendimiento**	**Facilidad de Uso**	**Casos de Uso**
Python	El lenguaje más utilizado en IA y Deep Learning debido a su simplicidad y extensas bibliotecas (TensorFlow, PyTorch, Keras).	Excelente rendimiento con optimización en frameworks como PyTorch y TensorFlow.	Muy fácil de usar para principiantes e investigadores, gran comunidad y recursos.	Entrenamiento y despliegue de modelos de Deep Learning, visión por computadora.
C++	Lenguaje de alto rendimiento, utilizado a menudo en backend de frameworks de Deep Learning.	Alto rendimiento, especialmente para aplicaciones críticas y despliegue en producción.	Más difícil de aprender que Python, pero eficiente en ejecución.	Implementación de modelos en producción, sistemas embebidos.
Java	Utilizado en bibliotecas de IA como Deeplearning4j. Adecuado para sistemas empresariales.	Rendimiento competitivo, aunque no tan optimizado.	Curva de aprendizaje intermedia, menos amigable para desarrollo rápido de IA.	Aplicaciones empresariales, procesamiento en backend, IA en plataformas grandes.

Python es el lenguaje preferido para inteligencia artificial y Deep learning debido a su simplicidad, flexibilidad y una amplia gama de bibliotecas especializadas como TensorFlow, PyTorch y Keras, que facilitan el desarrollo y entrenamiento de modelos avanzados. Su sintaxis intuitiva permite a los desarrolladores enfocarse en la lógica del modelo, mientras que su integración con herramientas científicas (como NumPy y Pandas) y compatibilidad con hardware de alto rendimiento (como GPUs) optimiza el procesamiento de datos y el entrenamiento.

El aprendizaje profundo se caracteriza por emplear estructuras neuronales con múltiples niveles de procesamiento de información. A diferencia de las redes neuronales tradicionales o "superficiales" que operan con una única capa oculta, las redes profundas incorporan múltiples capas ocultas entre sus capas de entrada y salida, lo que les permite realizar análisis más complejos y jerárquicos de la información.

En el campo del aprendizaje profundo, se pueden identificar cuatro enfoques principales de aprendizaje. El aprendizaje supervisado utiliza datos etiquetados y se centra en tareas específicas, mientras que el aprendizaje no supervisado trabaja con datos sin etiquetar, enfocándose en el análisis y descubrimiento de patrones. El aprendizaje semi-supervisado representa un enfoque híbrido que combina elementos de los dos anteriores, aprovechando tanto datos etiquetados como no etiquetados. Por último, el aprendizaje por refuerzo se basa en la interacción con el entorno, donde el modelo aprende mediante un proceso de prueba y error.

Las técnicas de aprendizaje profundo pueden clasificarse en tres categorías principales según su enfoque y aplicación. Las redes discriminativas se centran en el aprendizaje supervisado y se utilizan principalmente para predecir etiquetas o categorías. Las redes generativas, por otro lado, se emplean en el aprendizaje no supervisado y son capaces de generar nuevos datos o descubrir patrones subyacentes en los datos existentes. Finalmente, las redes híbridas combinan elementos de ambos enfoques y pueden incorporar otros métodos relevantes según sea necesario.

La Figura 13 muestra un esquema detallado de las tecnologías y su distribución, destacando la diversidad de técnicas y enfoques aplicados. Esta variedad permite que el aprendizaje profundo aborde una amplia gama de problemas del mundo real, ofreciendo soluciones adaptables a distintos contextos. La elección de una técnica específica dependerá de factores como la naturaleza del problema, el tipo de datos disponibles y los objetivos a alcanzar. Además, la combinación de diferentes enfoques potencia la capacidad del sistema para extraer patrones complejos y optimizar el rendimiento en aplicaciones específicas. La flexibilidad y escalabilidad de estos métodos los convierten en herramientas clave para resolver desafíos en múltiples áreas, desde la agricultura hasta la automatización industrial [72].

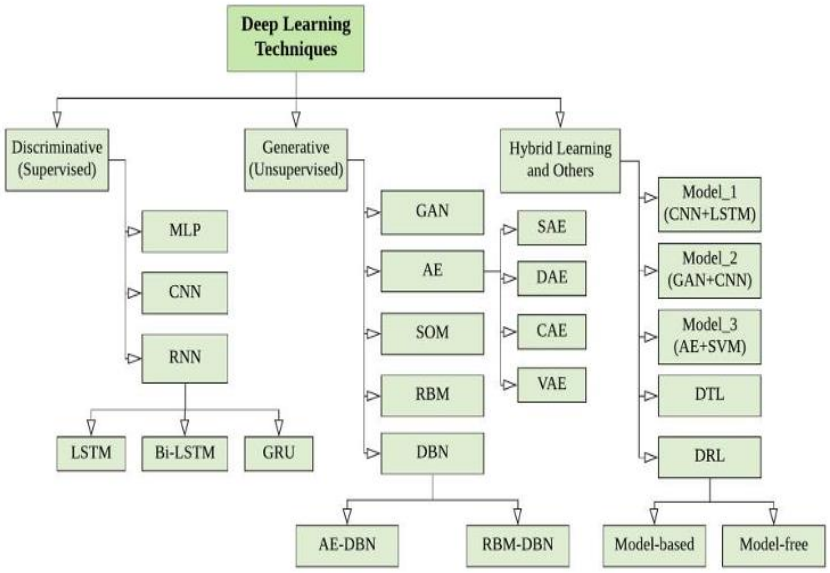


Figura 13. Tecnologías de Deep Learning [72].

Existen varios tipos de algoritmos utilizados en Deep Learning para el procesamiento de imágenes, cada uno con sus propias características y aplicaciones. Los algoritmos más comunes incluyen las Redes Neuronales Convolucionales (CNN), las Redes Generativas Antagónicas (GAN), las Redes de Segmentación Semántica, las Redes Neuronales Recurrentes (RNN), los Vision Transformers (ViT), entre otros. Cada uno de estos algoritmos tiene su utilidad en tareas específicas como la clasificación de imágenes, la detección de objetos, la segmentación y la generación de imágenes.

Las Redes Neuronales Convolucionales (CNN) son especialmente efectivas para tareas de procesamiento de imágenes debido a su capacidad para detectar patrones y características en las imágenes de forma jerárquica. A través de sus capas convolucionales, las CNN pueden identificar características simples como bordes y texturas en las primeras capas, y combinarlas en representaciones más complejas en capas posteriores. Esto permite que las CNN sean muy buenas para reconocer patrones visuales, como la clasificación de objetos en imágenes, a pesar de la variabilidad en su tamaño, orientación o color.

La ventaja de las CNN sobre otros algoritmos, como las redes generativas o las redes recurrentes, radica en su capacidad para manejar datos espaciales. Mientras que otros algoritmos, como las Redes Neuronales Recurrentes (RNN), están diseñados principalmente para procesar secuencias de datos, las CNN están diseñadas para

aprovechar la estructura espacial de las imágenes. Esto significa que las CNN son mucho más eficientes para aprender de las relaciones espaciales entre píxeles, lo que las convierte en la opción ideal para tareas de visión por computadora.

Entre las arquitecturas más conocidas de CNN se encuentran YOLO (You Only Look Once), SSD (Single Shot Multibox Detector) y Faster R-CNN. La Tabla 24 presenta la comparativa y sus características más relevantes.

Tabla 24. Comparativa de Redes Neuronales Convolucionales [73].

Característica	**YOLO (You Only Look Once)**	**SSD (Single Shot MultiBox Detector)**	**Faster R-CNN**
Velocidad	Extremadamente rápida, óptima para uso en tiempo real	Veloz, casi comparable a YOLO	Más lenta debido al proceso de propuestas de región
Precisión	Alta, con mejoras notables en versiones recientes	Elevada, especialmente eficaz con objetos de tamaños variados	Muy alta, referente en pruebas de precisión
Uso de Recursos	Eficiente, optimizado para análisis veloz	Moderado, más eficaz que CNNs tradicionales en tiempo real	Alto, por la necesidad de calcular propuestas de regiones
Complexidad	Menor que Faster R-CNN, manejo sencillo de cuadros	Complicada por el uso de múltiples cajas y escalas	Alta, involucra múltiples etapas y refinamiento de propuestas
Detección en Tiempo Real	Excelente debido a su alta velocidad	Muy buena, capaz de operar en tiempo real	Adecuada para tiempo real gracias a su velocidad de procesamiento
Aplicaciones Típicas	Monitoreo de vehículos, robots autónomos	Robótica móvil, sistemas embebidos, aplicaciones IoT	Investigación académica, usos que requieren alta precisión
Facilidad de Entrenamiento	Relativamente sencillo comparado con otros métodos	Requiere ajuste fino según los tamaños de los objetos	Demanda más tiempo debido a la necesidad de ajustar propuestas de región

La principal ventaja de YOLO (You Only Look Once) es que realiza la detección de objetos en una sola pasada a través de la red, lo que lo convierte en un modelo extremadamente rápido y eficiente, permitiendo su uso en aplicaciones de tiempo real, como vigilancia, conducción autónoma y análisis en video en vivo. Además, YOLO se adapta a diversas necesidades gracias a sus distintas versiones, que han sido mejoradas continuamente a lo largo de los años, incorporando avances en arquitectura y optimización. La Tabla 25 muestra las características de las distintas versiones de YOLO.

Tabla 25. Comparativa de versiones de YOLO [74].

Versión	**Características Principales**	**Métricas (FPS, mAP)**	**Aplicaciones Clave**
YOLOv1	Año: 2016 Innovación: Introdujo detección en un solo paso. Backbone: Darknet Características: Predicción de cajas y clases en una sola pasada; problemas con objetos pequeños.	FPS: 45 FPS mAP: 63.4%	Detección básica en tiempo real.
YOLOv2	Año: 2016 Innovación: Mejoró precisión y localización con anchor boxes Backbone: Darknet-19 Características: Uso de anchor boxes, normalización por lotes, k-means para anclas, entrenamiento multiescala.	FPS: 52 FPS mAP: 69.0%	Mejor localización y escalabilidad.
YOLOv3	Año: 2018 Innovación: Detección a múltiples escalas Backbone: Darknet-53 Características: Redes piramidales de características, detección en tres escalas, mejora en objetos pequeños.	FPS: 34 FPS mAP: 57.9%	Detección en múltiples escalas.
YOLOv4	Año: 2020 Innovación: Optimización para sistemas de producción Backbone: CSPDarknet53 Características: Path Aggregation Network (PAN), Spatial Pyramid Pooling (SPP), CSPDarknet53 backbone.	FPS: 65 FPS mAP: 44.3%	Producción, vigilancia, vehículos autónomos.
YOLOv5	Año: 2020 Innovación: Implementado en PyTorch, más eficiente Backbone: Modified CSP Características: Mosaic, MixUp, AutoAnchor, variantes nano, small, medium, large y extra-large.	FPS: 200 FPS mAP: 50.7%	Dispositivos de bajos recursos.
YOLOv6	Año: 2022 Innovación: Optimizado para aplicaciones industriales Backbone: EfficientRep Características: Cuantización, auto-destilación, EfficientRep backbone, BiC module, AAT strategy.	FPS: 29 FPS mAP: 52.5%	Aplicaciones industriales avanzadas.
YOLOv7	Año: 2022 Innovación: Mejor detección en tiempo real Backbone: E-ELAN Características: E-ELAN para aprendizaje eficiente, RepConvN, etiquetas gruesas y finas.	FPS: 160 FPS mAP: 56.8%	Detección en tiempo real de alta precisión.
YOLOv8	Año: 2023 Innovación: Versatilidad y segmentación semántica Backbone: C2f Características: Segmentación semántica con YOLOv8-Seg, 280 FPS, ajustes en módulos C2f.	FPS: 280 FPS mAP: 53.9%	Segmentación semántica y aplicaciones modernas.

YOLOv8 se considera la mejor versión hasta la fecha debido a su equilibrio entre velocidad, precisión y versatilidad. Introduce capacidades avanzadas de segmentación semántica con el modelo YOLOv8-Seg, lo que permite no solo detectar objetos, sino también identificar áreas específicas dentro de ellos. Estas mejoras, junto con su diseño orientado a facilitar la implementación y el entrenamiento, hacen de YOLOv8 una herramienta poderosa para abordar los desafíos más exigentes en la detección de objetos y la segmentación en escenarios modernos.

La Figura 14 presenta la arquitectura de YOLOv8. El sistema está compuesto por tres elementos principales: Backbone, Head (utilizados para la detección de objetos) y Details. El Backbone emplea múltiples capas convolucionales y módulos C2f organizados en resoluciones que van de P1 a P5, con el propósito de extraer características relevantes de las imágenes. Dentro de este componente se encuentra el módulo SPPF, diseñado para manejar diferentes escalas y resoluciones. Por su parte, el Head se encarga de integrar y refinar las características extraídas por el Backbone, utilizando técnicas como concatenaciones y upsampling, lo que mejora la precisión en la detección de objetos en múltiples escalas. Finalmente, Details describe las capas específicas utilizadas en la arquitectura, como convoluciones, normalización, concatenación y max pooling, las cuales optimizan el flujo de gradientes y aumentan la robustez del modelo [75].

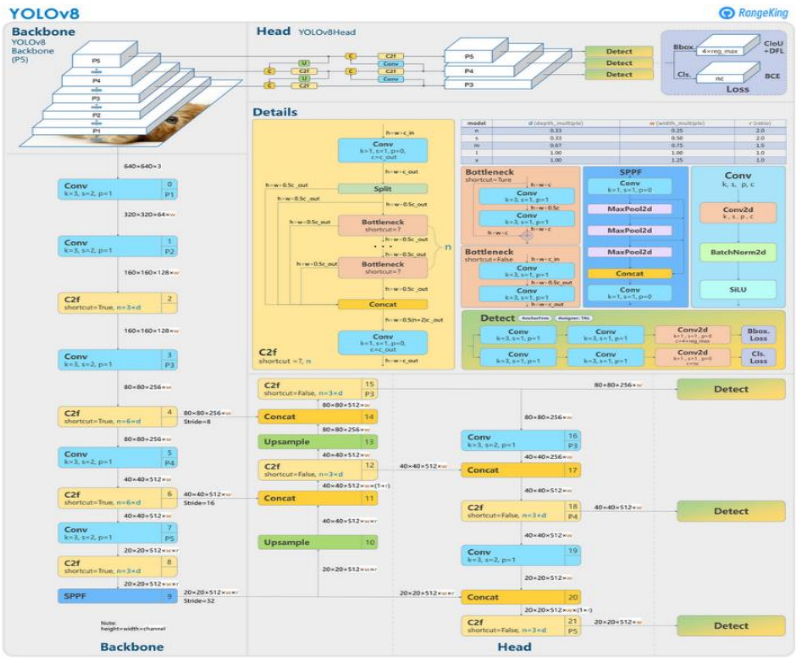


Figura 14. Arquitectura YOLOv8 [75].

YOLOv8, una de las implementaciones más avanzadas para la detección de objetos en tiempo real, ofrece cinco variantes diseñadas para adaptarse a diferentes necesidades y recursos: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large) y YOLOv8x (extra-large).

La elección de la variante adecuada depende de varios factores, como la complejidad del proyecto, el nivel de precisión requerido, y la capacidad computacional disponible. Por ejemplo, YOLOv8n es ideal para dispositivos con recursos debido a su enfoque en priorizar la velocidad y el bajo consumo de recursos a costa de una ligera reducción en la precisión. En contraste, YOLOv8x está diseñado para sistemas más potentes, ofreciendo la mayor precisión posible, aunque requiere un hardware más robusto y tiempos de procesamiento más largos.

1.4 Objetivos

1.4.1 Objetivo general

Implementar un sistema de monitoreo y control IOT de Cultivos Agrícolas mediante la Arquitectura Edge-Cloud y Deep Learning.

1.4.2 Objetivos específicos

- Analizar el impacto de IOT y las características de la arquitectura Edge-Cloud y Deep Learning en el contexto de la agricultura.
- Desarrollar un dispositivo IOT para la adquisición y control de parámetros físicos en plantaciones agrícolas.
- Aplicar un modelo de inteligencia artificial basado en Deep Learning para identificar y analizar las condiciones de los cultivos agrícolas.
- Implementar una arquitectura Edge Cloud para el monitoreo y control de variables existentes en los cultivos agrícolas.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

En el desarrollo de sistemas inteligentes para el monitoreo y control de cultivos hidroónicos, la selección adecuada de componentes y herramientas es fundamental para garantizar un funcionamiento eficiente y preciso. Se usó el sensor ec meter v2.0 ya que tiene una sensibilidad de 0 a 20000us/cm, el sensor de pH DIY SEMSPr pH-4502C, el rango de medición es de 0 a 14, el motor nema 47ls2227 que posee un torque de 0.45Nm, voltaje de 3.3V y un consumo de 1.5A, el driver tb6560 en vista que la corriente que necesita es de 0.5 a 3.5 A. La esp32 al tener Wi-Fi 802.11 b/g/n Dual-core 240 MHz y 520 KB RAM / 4 MB Flash. Para la obtención de imágenes se optó por la Raspberry pi camera v1.3 resolución de 1800x720, ya que es suficiente para el reconocimiento de las lechugas.

Para la programación del dispositivo se usó Arduino IDE y la programación de la inteligencia artificial con Python y en la arquitectura YOLO, usando 1200 imágenes, creando el dataset en Roboflow, el entrenamiento se hizo en Kaggle y la implementación en la Raspberry pi 4, con comunicación MQTT, además de levantar la nube con Open Stack y el servidor con de Ubuntu 20.04, accediendo a través de ZeroTier, permitiendo visualizar la información desde cualquier sitio.

2.2 Métodos

2.2.1 Modalidad de la investigación

La investigación aplicada se desarrolló a través de una exhaustiva exploración y análisis de tecnologías actuales, complementada con el estudio de casos relevantes y la evaluación de soluciones prácticas implementadas en sistemas de monitoreo y control de cultivos hidropónicos. Este enfoque permitió identificar herramientas innovadoras y metodologías efectivas, que han sido probadas y validadas en diversos escenarios agrícolas. Además, se analizaron las tendencias tecnológicas más recientes en el ámbito de la agricultura inteligente, asegurando que las propuestas estuvieran

alineadas con las necesidades actuales del sector y orientadas hacia la sostenibilidad y la eficiencia en la producción.

La investigación bibliográfica se llevó a cabo mediante una revisión exhaustiva de la literatura académica y técnica relacionada con los sistemas de monitoreo y control IoT para cultivos agrícolas, así como con las tecnologías de arquitectura Edge Cloud y Deep Learning. Para ello, se identificaron y analizaron artículos científicos, libros especializados y publicaciones relevantes que abordan temas como el diseño y uso de sensores agrícolas, plataformas IoT para la gestión de datos en tiempo real, algoritmos de aprendizaje automático aplicados a la predicción y optimización de cultivos, y casos prácticos que demuestran la efectividad de estas tecnologías en entornos reales.

La investigación experimental se centró en la implementación y evaluación de un sistema de monitoreo y control IOT para cultivos agrícolas basado en la arquitectura Edge Cloud y Deep Learning. Se diseñó y llevó a cabo experimentos controlados para medir la eficacia del sistema en condiciones reales de campo. Esto incluyó la optimización de algoritmos y la evaluación del rendimiento del sistema en términos de precisión, eficiencia y ahorro de recursos. Los resultados obtenidos de estos experimentos proporcionaron datos cuantitativos esenciales para la validación, la viabilidad y efectividad del sistema aplicado en la agrícola moderna.

La investigación de campo fue esencial para garantizar que el sistema de monitoreo y control IoT funcione en condiciones reales de los cultivos agrícolas. Durante esta etapa, se consideraron factores críticos como la variación de nutrientes, las limitaciones de conectividad, los cuales pueden influir significativamente en el rendimiento del sistema. Estas pruebas permitieron evaluar no solo la eficiencia de la arquitectura Edge-Cloud en el procesamiento y transmisión de datos en tiempo real, sino también su capacidad para operar de manera estable y efectiva en entornos desafiantes. Este enfoque práctico contribuyó a validar el modelo y a identificar oportunidades de mejora para futuras implementaciones.

2.2.2 Recolección de información

Se utilizó información proveniente de diversos repositorios de universidades nacionales, así como libros, revistas científicas en línea y otras bases de datos confiables, con el objetivo de contribuir al desarrollo del proyecto.

Tabla 26. Criterios de recolección de información.

Criterio	**Descripción**
¿Para qué?	Para realizar un monitoreo y control de cultivos agrícolas eficaz.
¿De qué personas u objetos?	De la producción agrícola
¿Cómo?	Realizando un análisis de datos en dispositivos, procesándolos en el servidor local y visualizando en la nube.
¿Con qué?	Con sensores, dispositivos Edge, algoritmos de deep learning y plataformas en la nube.
¿Cuándo?	En el periodo septiembre 2024 – febrero 2025
¿Dónde?	Cultivo Hidropónico ubicado en Tungurahua, cantón Ambato

2.2.3 Procesamiento y análisis de datos

Con la información recopilada se procedió de la siguiente manera:

- Estudio del tipo de información a recolectar.
- Análisis de la información.
- Procesamiento la información.
- Presentación de resultados más relevantes del proyecto de investigación.

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

La implementación del sistema se llevó a cabo en varias etapas clave. Inicialmente, se seleccionaron los sensores adecuados para controlar parámetros críticos como la conductividad eléctrica y el pH del agua en cultivos hidropónicos de lechuga. Para ello, se emplearon el sensor EC Meter V2.0 y el sensor de pH DIY SEMSPr PH-4502C. Además, se diseñó e integró una bomba dosificadora automatizada utilizando un motor a pasos NEMA 47LS2227 y el driver TB6560, lo que permitió la inyección precisa de nutrientes en el sistema.

El dispositivo fue diseñado utilizando el software Proteus e implementado con el microcontrolador ESP32, el cual desempeña un papel central en el procesamiento de datos y su transmisión a un servidor mediante el protocolo MQTT. Este flujo de datos se integró en una arquitectura Edge Cloud, lo que permite la visualización y el control en tiempo real del sistema, garantizando una gestión eficiente y ágil de la información.

Adicionalmente, se implementaron medidas de protección para cada uno de los componentes, incluyendo el dispositivo y los sensores, con el objetivo de asegurar su funcionamiento confiable y protegerlos frente a posibles fallos o daños, fortaleciendo así la seguridad del sistema en su conjunto.En cuanto al reconocimiento de enfermedades en los cultivos de lechuga, se utilizó inteligencia artificial basada en aprendizaje profundo (Deep Learning). El desarrollo se realizó utilizando el lenguaje de programación Python y el modelo YOLO (You Only Look Once). Para la obtención y preparación del conjunto de datos (DataSet), se empleó Roboflowy el entrenamiento en Kaggle. Finalmente, la implementación del sistema de reconocimiento se realizó en una Raspberry Pi 4, permitiendo un procesamiento eficiente y autónomo en tiempo real.

3.1 Arquitectura del sistema

En la Figura 15 se muestra la arquitectura general del sistema, consta de 4 etapas: Edge layer, server layer, cloud layer y display layer.

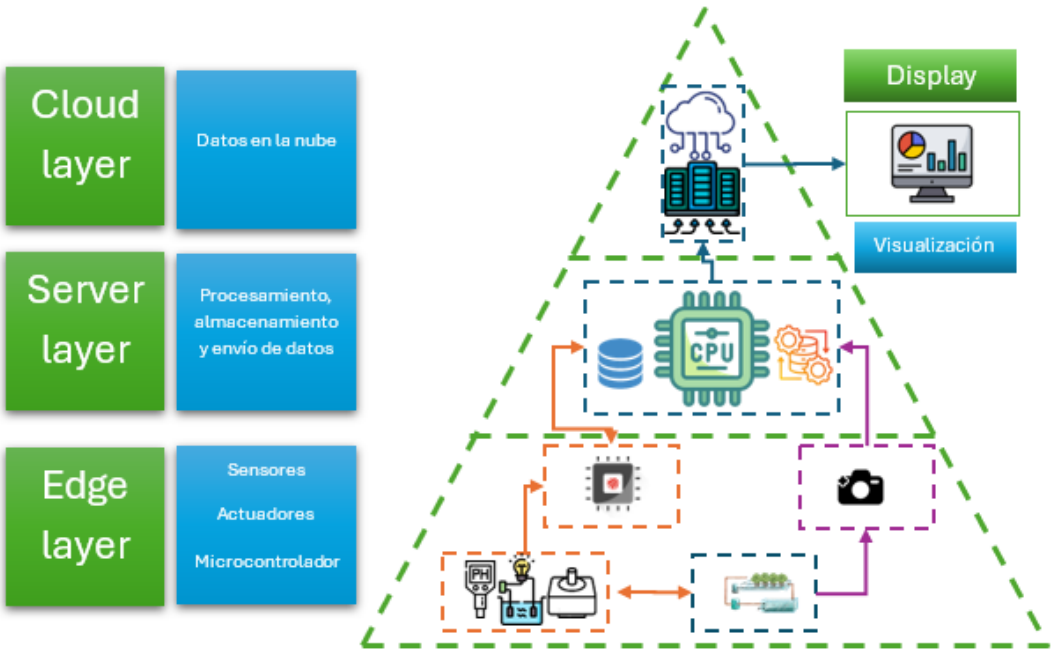


Figura 15. Arquitectura general del sistema.

- Etapa 1 Edge layer: En esta sección se encuentran los dispositivos, sensores y actuadores que son fundamentales para la obtención de datos y el control físico del sistema. Estos componentes trabajan en conjunto para recopilar información en tiempo real y gestionar las operaciones necesarias para el funcionamiento efectivo del sistema.

- Etapa 2 Server layer: En esta sección se encuentra el microprocesador, que desempeña un papel crucial como el enlace para el envío de datos entre los dispositivos del sistema y la nube de borde. Además, permite realizar análisis avanzados mediante algoritmos de inteligencia artificial y aprendizaje automático.

- Etapa 3 Cloud layer: La Capa de Nube es un componente esencial de la arquitectura Edge-Cloud, que proporciona una infraestructura robusta y escalable para el almacenamiento, procesamiento y análisis de datos. Esta capa permite la integración de grandes volúmenes de información recopilados por los dispositivos y sensores en el borde, facilitando su almacenamiento seguro y su análisis detallado.

- Etapa 4 Display layer: Esta etapa es importante ya que permite la interacción entre los usuarios y los datos procesados. Esta capa proporciona un dashboard intuitivo y accesible que facilita la presentación de información relevante y permite a los usuarios finales tomar decisiones informadas basadas en los análisis realizados en las capas anteriores. Esto mejora la comprensión de los datos y permite tomar decisiones más precisas basadas en evidencia.

Para la implementación de la inteligencia artificial, se utilizó la metodología CRISP-DM (Cross Industry Standard Process for Data Mining) mostrado en la Tabla 22. Este enfoque estructurado permitió abordar el problema de manera sistemática, comenzando con una comprensión profunda del problema, que en este caso es el reconocimiento de enfermedades.

Se llevó a cabo la preparación de los datos, asegurando que fueran adecuados y relevantes para el modelo. Posteriormente, se procedió al modelado, como es YOLO. Las etapas siguientes incluyeron la evaluación, donde se analizaron los resultados obtenidos para garantizar la precisión del modelo, y finalmente, la implementación, integrando en el sistema. Se presenta las etapas de la inteligencia artificial en la Figura 16.

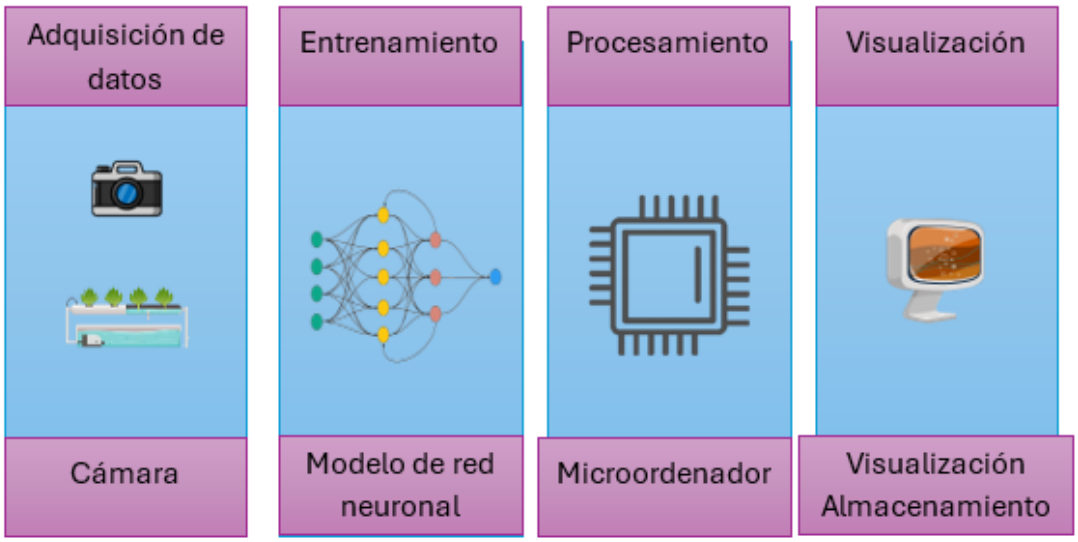


Figura 16. Etapas de la inteligencia artificial.

- Etapa 1 Adquisición de datos: Para obtener las imágenes, se utilizó una cámara a la Raspberry pi 4 para captar las fotos.

- Etapa 2 Fase de entrenamiento: Para el entrenamiento se toma de un dataset ya creado de Robotflow, tiene cargado miles de imágenes, acomodando a lo que se necesita y se entrena en la plataforma de Kaggle.

- Etapa 3 Procesamiento de datos: Una vez aplicado el entrenamiento, se despliega y se procesa en el microprocesador que será encargado de enviar los datos con la arquitectura Edge-Cloud, además de almacenar los resultados obtenidos a través de la base de datos.

- Etapa 4 Visualización y almacenamiento de datos: Una vez entrenado y optimizado el modelo, se introduce las imágenes de las lechugas para el reconocimiento de enfermedades y visualizar el estado del cultivo agrícola.

La Figura 17 presenta el diagrama de todo el sistema

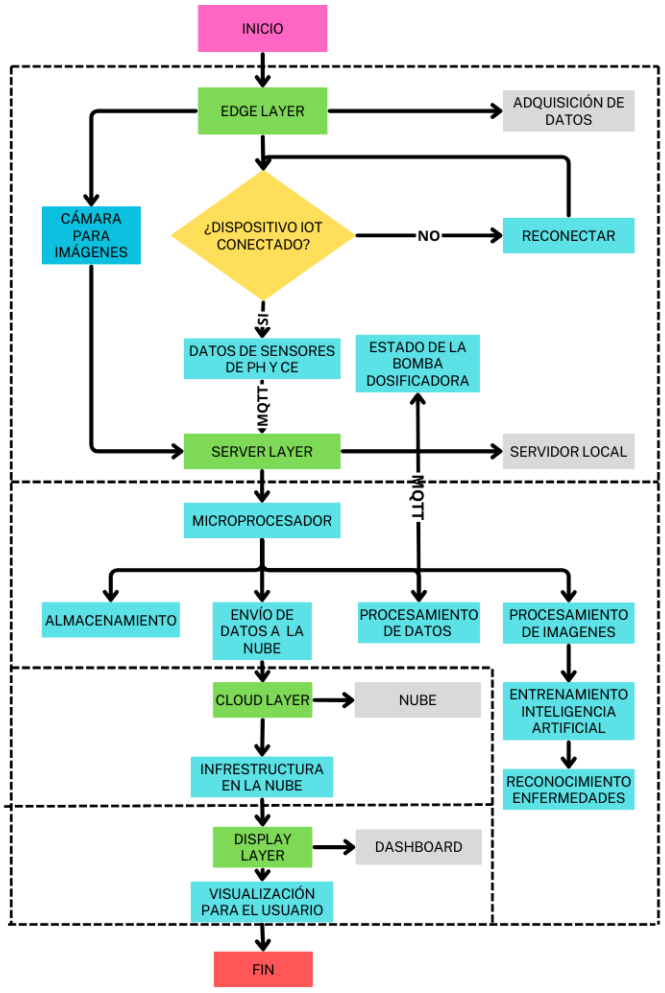


Figura 17. Diagrama de flujo del sistema

3.2 Selección de tecnología

Se usa el sensor ec meter v2.0 mostrado en Tabla 7 ya que tiene una sensibilidad de 0 a 20000us/cm, el sensor de pH DIY SEMSPr pH-4502C, presentado en la Tabla 8, el rango de medición es de 0 a 14, el motor nema 47HD4027-01-A mostrado en la Tabla 9, el driver tb6560 mostrado en la Tabla 10, por el consumo de voltaje que es de 3V y de corriente de 1.5A, además de un rango de movimiento de 1.8°. La esp32 al tener Wi-Fi 802.11 b/g/n Dual-core 240 MHz y 520 KB RAM / 4 MB Flash mostrada en la Tabla 11 , programado con Arduino Ide presentado en la Tabla 14, por la eficiencia de hardware. Para comunicación entre etapas se usó MQTT mostrado en la Tabla 15 por su eficiencia en ancho de banda, bajo consumo energético y alta fiabilidad. Para la obtención de imágenes se optó por la Rapberry pi camera v1.3 en base a la Tabla 13, por la resolución de 5 MP (2592x1944), ya que es suficiente para el reconocimiento de las lechugas. Para el servidor local se usa una Raspberry Pi 4 mostrado en la Tabla 12 por su procesador Quad-Core 1.5 GHz y 4Gb de RAM. Para el entrenamiento se elige YOLO en base a la Tabla 24 por su facilidad de entrenamiento y precisión y YOLOv8 por la Tabla 25 por su segmentación semántica y aplicaciones modernas.

3.3 Implementación de Edge Layer

El diagrama de la implementación del sistema en el cultivo hidropónico se presenta en la Figura 18. En este sistema, el dispositivo recolecta datos del pH y la conductividad eléctrica, los cuales son enviados al controlador principal. Tras recibir la respuesta procesada de la etapa 2, la bomba dosificadora se activa para suministrar los nutrientes necesarios al cultivo, garantizando un balance óptimo para el desarrollo de las plantas.

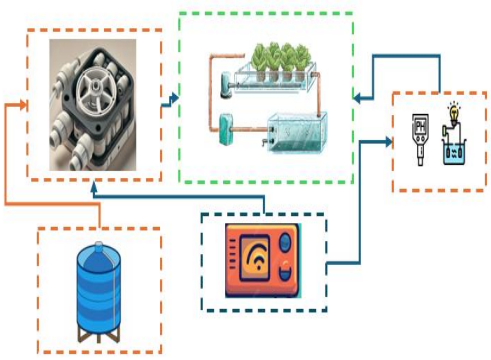


Figura 18. Implementación de los elementos.

3.3.1 Diseño de circuito electrónico

El diagrama electrónico presentado representa la configuración completa del sistema desarrollado, que integra diversos componentes esenciales para la adquisición, procesamiento y control de datos.

a. Dispositivo IoT

El propósito de este diagrama es ofrecer una vista clara y detallada de cómo se interconectan los diferentes módulos, permitiendo comprender el flujo de datos y la operación del sistema, el diseño se realizó en el programa Proteus. La Figura 19 muestra las conexiones y elementos utilizados, para la medición de la conductividad eléctrica (CE), nivel de pH, control del motor para la bomba dosificadora, leds e indicadores externos.

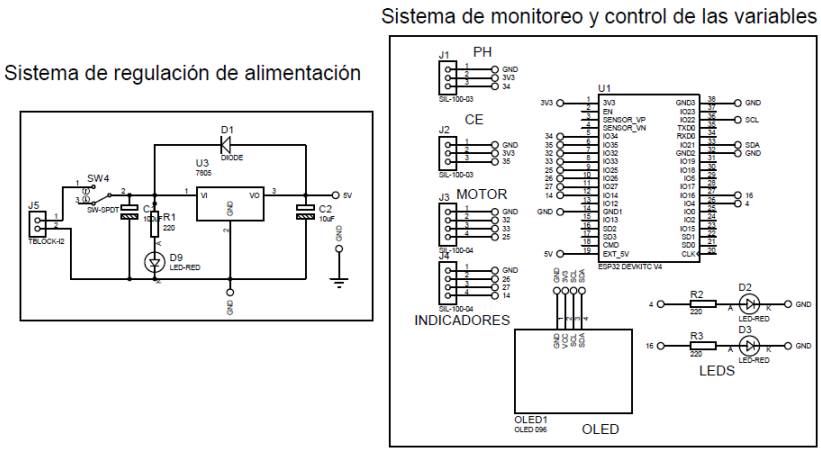


Figura 19. Diagrama electrónico.

El cual consta de dos etapas, el sistema de regulación de alimentación y el de monitoreo y control de variables.

Etapa 1: Sistema de regulación de alimentación

El sistema de regulación de alimentación es una parte fundamental del diseño electrónico, ya que garantiza el suministro de energía estable y seguro a todos los componentes del sistema. Este subsistema se encarga de convertir y distribuir la energía proveniente de la fuente principal de alimentación, adaptándola a los diferentes requerimientos de voltaje y corriente de cada dispositivo.

Se usa una bornera para conectar la fuente de alimentación continua, condensador de 100uf y 10 uf para la amortiguación de la señal, un diodo 1n4007 para protección, el integrado 7805 que regula la fuente de entrada a una salida de 5V.

Para facilitar la operación y monitoreo, el sistema de regulación también integra un LED, que muestran visualmente cuando el sistema está activo y una resistencia de 1.2K proporcionando una referencia rápida del estado del suministro de energía en el circuito.

Etapa 2: Monitoreo y control de variables

En esta etapa, el sistema se enfoca en la adquisición de datos y el control de las variables críticas del entorno. Utilizando una ESP32 como núcleo del sistema, se gestionan tanto la captura de información de los sensores como el control de los actuadores, permitiendo una operación en tiempo real y eficiente. Basando en el Anexo B, se define los pines a usar, el 34 y 35 se establece como pines analógicos de entrada para la medición de pH y CE. Los pines 32,33,25,26,27,14 son pines de salida digital y los pines 4 y 16 funcionan para muestreo por medio de leds.

El sensor pH-4502C recolecta datos en formato analógico y permite medir el nivel de pH en líquidos con precisión. Por su parte, el sensor Analog Electrical Conductivity Sensor/Meter V2 mide, también de forma analógica, la conductividad eléctrica presente en las soluciones.

Los actuadores, como motores y leds, se controlan en base a los parámetros de los sensores. Cuando el nivel de pH es inferior a 6 o el nivel de CE desciende por debajo de 1200 uS se activa el motor con salida.

Para facilitar el seguimiento de estos datos, se incluye la Pantalla Oled 128×64 I2c que permite visualizar en tiempo real los valores monitoreados.

En conjunto, la ESP32, los sensores, los actuadores y la pantalla OLED conforman un sistema de monitoreo y control robusto, que permite la recolección de datos en tiempo real y el ajuste automático de las variables, optimizando el rendimiento del sistema. La Figura 20 muestra una representación 3D del dispositivo electrónico a implementar.

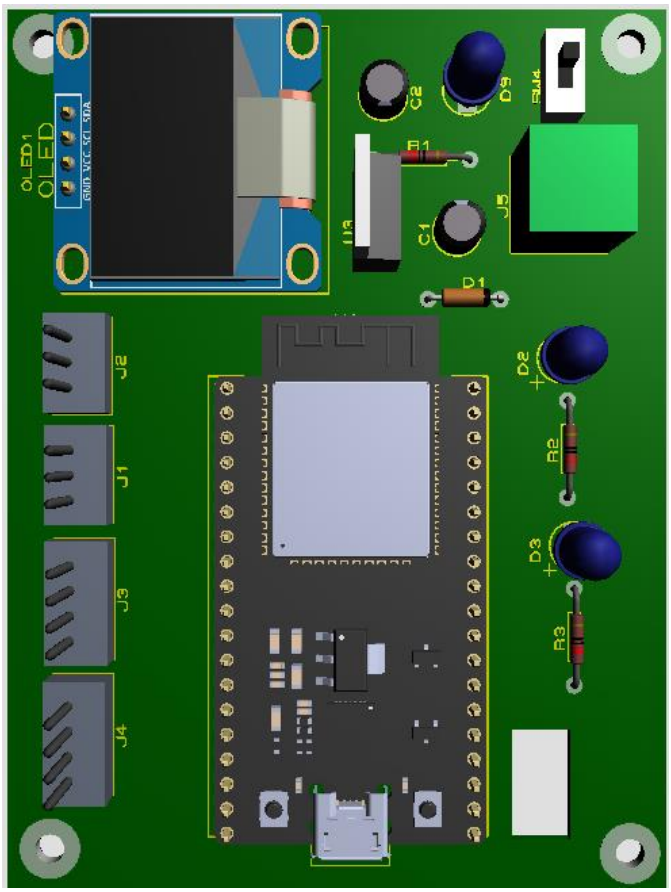


Figura 20. Modelo 3D.

La implementación se llevó a cabo utilizando baquelita como base, y las pistas del circuito fueron elaboradas mediante la técnica de planchado, como se ilustra en la Figura 21.

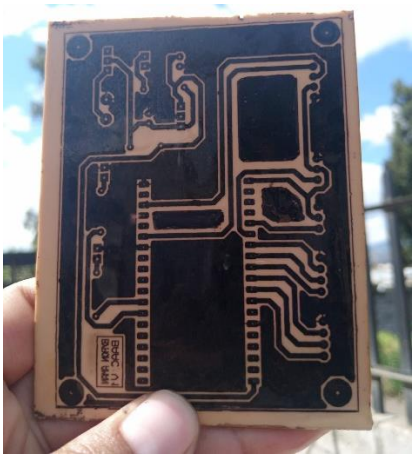


Figura 21. Pista implementada.

La placa finalmente se ve como se muestra en la Figura 22.

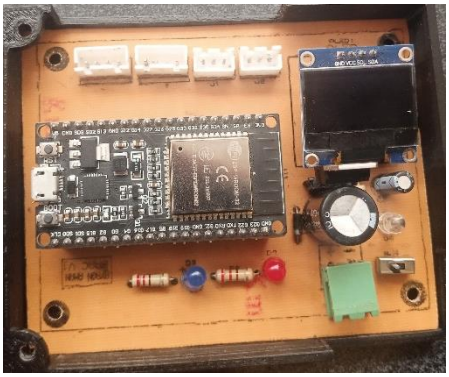


Figura 22. Implementación del prototipo.

Para la protección del dispositivo se diseñó una carcasa que proteja de posibles daños, el diseño se realizó en Solidworks y se estableció a través de las dimensiones de la placa realizada. La Figura 23 muestra el diseño establecido para implementar.

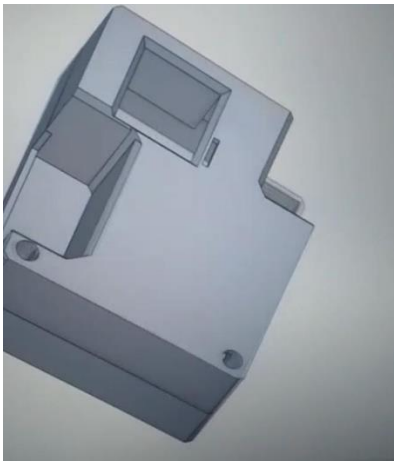


Figura 23. Diseño del prototipo.

La Figura 24 muestra el dispositivo implementado, dejando los espacios para las conexiones de los sensores, junto con el driver con el motor para el actuador, así mismo, la visualización de los datos en la pantalla oled.



Figura 24. Dispositivo implementado.

Los sensores elegidos son: EC Meter V2.0 mostrado en la Tabla 7 y el DIY SEMSPr pH-4502C presentado en la Tabla 8. El Anexo E presenta los pines de conexión del sensor pH y el Anexo F los pines del sensor de conductividad eléctrica. La Figura 25 muestra el sensor de CE y la Figura 26 el sensor de pH usado.



Figura 25. Sensor de conductividad eléctrica.



Figura 26. Sensor de pH.

La protección de los módulos fue diseñada para garantizar la seguridad y el correcto funcionamiento del dispositivo. Este diseño incluye medidas específicas que aseguran la estabilidad y la protección frente a daños externos y salpicaduras de agua como puede apreciarse en detalle en la Figura 27. La implementación se puede apreciar en la Figura 28.

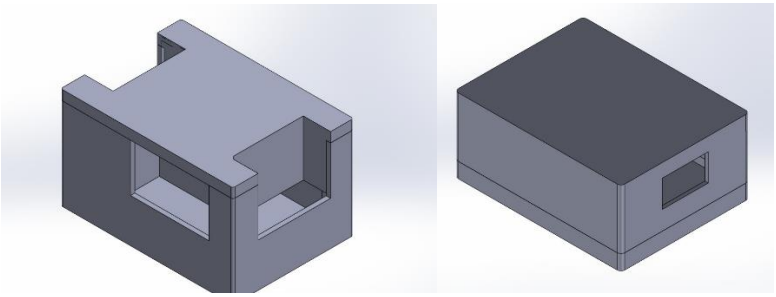


Figura 27. Diseño para los sensores.



Figura 28. Protección de los sensores.

Para la implementación del actuador, se usó una bomba dosificadora que permite la dosificación precisa de fluidos mediante un motor a pasos. Este sistema controla la salida de nutrientes que se distribuyen a toda la producción, asegurando un manejo eficiente y automatizado. La bomba dosificadora está compuesta por un motor a pasos, un driver de control y el diseño estructural de la bomba.

El motor seleccionado es el modelo NEMA 42HD4027-01-A, que opera con un voltaje de 3.3V, una corriente de 1.5A y un ángulo de paso de 1.8°, garantizando precisión en cada movimiento. Para su control, se utilizó el driver TB6560, configurado para una corriente máxima de salida de 1.5A, e integrado con una función de ahorro de energía que se activa cuando no está en uso. Para la conexión de ambos elementos, el Anexo C presenta las conexiones del motor y el para el driver, según las especificaciones del fabricante que se muestra en el Anexo D, muestra las conexiones correspondientes y que la configuración del driver requiere encender únicamente los switches 1 y 3, dejando los demás apagados.

Adicionalmente, se diseñó e implementó la protección para el driver, como se visualiza en la Figura 29.

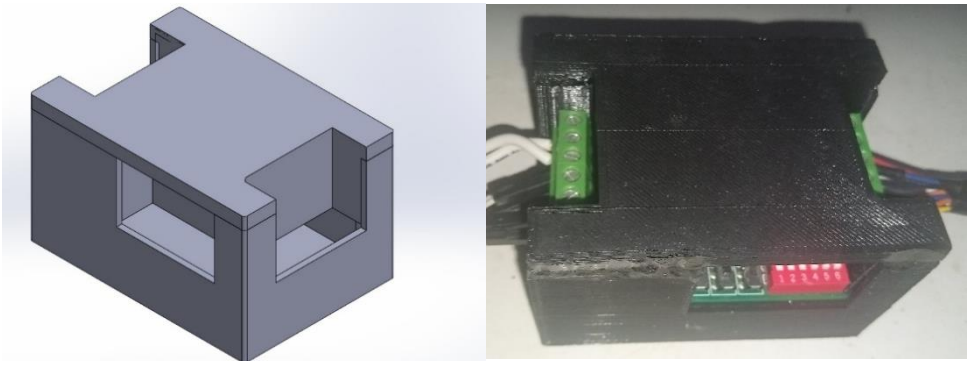


Figura 29. Protección al driver.

La Figura 30 presenta el diseño de la bomba dosificadora, que incorpora una manguera que permite estrangular, un diseño estructural optimizado y un motor a pasos. Este conjunto permite un control preciso de la salida de nutrientes, asegurando una dosificación eficiente y uniforme en el sistema.



Figura 30. Bomba dosificadora.

b. Algoritmo de medición

La Figura 31 muestra el diagrama de programación, siguiendo este orden, conexión a wifi, conexión a mqtt, calibración de sensores, envió de datos, recibimiento de alertas y actuadores en el dispositivo, en el Anexo I se visualiza el código completo.

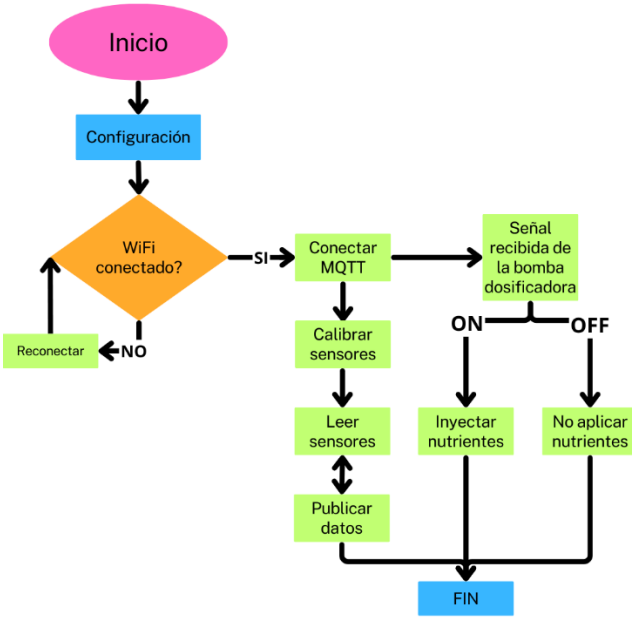


Figura 31. Diagrama de programación.

Se conecta a la red local disponible para posteriormente conectar al broken definido, el broker.hivemq.com que permite enviar y recibir datos, la Figura 32 muestra la configuración hecha.

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Figura 32. Conexión a la red.

Los sensores son industriales por ende los valores que proporciona son bastantes buenos, sin embargo, para un mejor ajuste de los valores se procedió a calibrar los sensores con técnicas matemáticas como regresión lineal para ajustar los valores de

los sensores, además de suavizar los valores, evitando variables muy extendidas, la Figura 33 muestra la aplicación matemática dada del sensor pH y en la Figura 34 de sensor de CE.

```
// Aplicar filtro EMA para suavizar el valor
phCal.phEMA = phCal.alpha_pH * phAnalogValue + (1 - phCal.alpha_pH) * phCal.phEMA;

// Imprimir valor analógico en consola
Serial.print("Valor analógico pH: ");
Serial.println(phAnalogValue);

// Calcular pH según el rango
if (phCal.phEMA <= phCal.bits_pH7) {
    return phCal.slope_acid * phCal.phEMA + phCal.intercept_acid;
} else {
    return phCal.slope_alkaline * phCal.phEMA + phCal.intercept_alkaline;
}
```

Figura 33. Calibración de sensor pH.

```
// Aplicar filtro EMA para suavizar el valor
ecCal.ecEMA = ecCal.alpha_ec * ecAnalogValue + (1 - ecCal.alpha_ec) * ecCal.ecEMA;

// Imprimir valor analógico en consola
Serial.print("Valor analógico Conductividad: ");
Serial.println(ecAnalogValue);

// Calcular conductividad según el rango
float ecValue;
if (ecCal.ecEMA == 0) {
    ecValue = 20;  // Valor mínimo de conductividad
} else if (ecCal.ecEMA <= ecCal.analog_low) {
    ecValue = ecCal.slope_low * ecCal.ecEMA + ecCal.intercept_low;
} else {
    ecValue = ecCal.slope_high * ecCal.ecEMA + ecCal.intercept_high - 100;
}
```

Figura 34. Calibración del sensor de CE.

Para el actuador se configura que al recibir cada alerta se mueva 150 pasos, esto representa unos tres cuartos de vuelta del motor en 100000 microsegundos, la Figura 35 presenta la configuración del mismo.

```
void moveMotor(int steps) {
    for (int i = 0; i < steps; i++) {
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(10000);  // Ajusta para controlar la velocidad
        digitalWrite(stepPin, LOW);
        delayMicroseconds(10000);  // Ajusta para controlar la velocidad
    }
}
```

Figura 35. Configuración del motor.

Para el envío de datos se procedió con el protocolo mqtt por la velocidad y gestión de datos que puede enviar y recibir, se suscribe a un broken público, se envía los datos de los sensores y la cantidad de nutrientes que se aplica a través de una cadena, además

de recibir las alertas de activación del motor, la Figura 36 presenta el envío de datos al servidor y la Figura 37 el recibimiento de las alertas.

```
String topic = "esp32/datos";
String payload = "{\"temperatura\":" + pHString + ",\"humedad\":" + ecString + ",\"voltaje\":" + moString + "}";
client.publish(topic.c_str(), payload.c_str());
```

Figura 36. Envió de datos del sistema.

```
void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Mensaje recibido [");
    Serial.print(topic);
    Serial.print("]: ");
    String message = "";
    for (int i = 0; i < length; i++) {
        message += (char)payload[i];
    }
    Serial.println(message);
```

Figura 37. Recibimiento de alertas.

Para la visualización de datos se publicó a través de la pantalla oled donde se ve la información de los sensores, nutrientes y el estado del motor, la Figura 38 presenta la forma en la que se presenta los datos.

```
// Actualiza la pantalla OLED con los valores medidos
void updateDisplay(float pH, float ec, int motor) {
    display.clearDisplay();
    display.setCursor(0, 0);
    display.print("pH: ");
    display.println(pH);
    display.print("Conductividad: ");
    display.println(ec);
    display.print("uS/cm");
    display.print("Motor: ");
    display.println(motor);
    display.print("estado");
    display.display();
}
```

Figura 38. Visualización de datos a través de la pantalla.

c. Elementos para el sistema de inteligencia artificial

Etapa 1 Adquisición de datos: Para la obtención de imágenes de la producción de lechuga se realizó con la Raspberry Pi camera Rev v1.3 mostrada en la Tabla 13 por su resolución de 2592 x 1944 pixeles, además de tener el serial de cámara MIPI de 15

pines que se conecta directamente a la placa la Figura 39 presenta la cámara seleccionada.

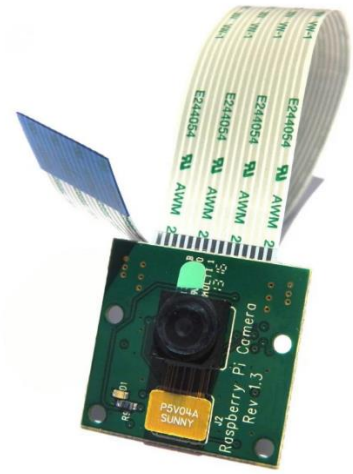


Figura 39. Raspberry Pi camera Rev v1.3.

3.4 Implementación de Server layer

3.4.1 Control de variables

Para la implementación del servidor se realiza en una Raspberry pi 4B que presenta la Tabla 12 por tener un por un procesador ARMv8 de cuatro núcleos de 64 bits a una velocidad de 1.5GHz, GPU Broadcom VideoCore VI y Memoria RAM de 4GB LPDDR4, el Anexo G presenta las especificaciones del mismo. El sistema operativo cargado es Ubuntu 20.04 presentado en la Figura 40.

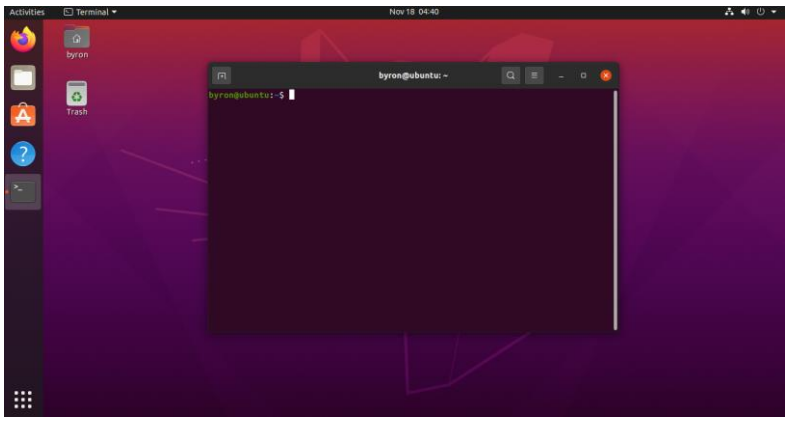


Figura 40. Interfaz Ubuntu 20.04.

Para el procesamiento de datos, junto con el envío, así como, la recepción de información se procedió a instalar Node-RED, mientras que para el almacenamiento se utilizó MySQL con el gestor de base de datos phpMyAdmin. La Figura 41 la interfaz de phpMyAdmin.

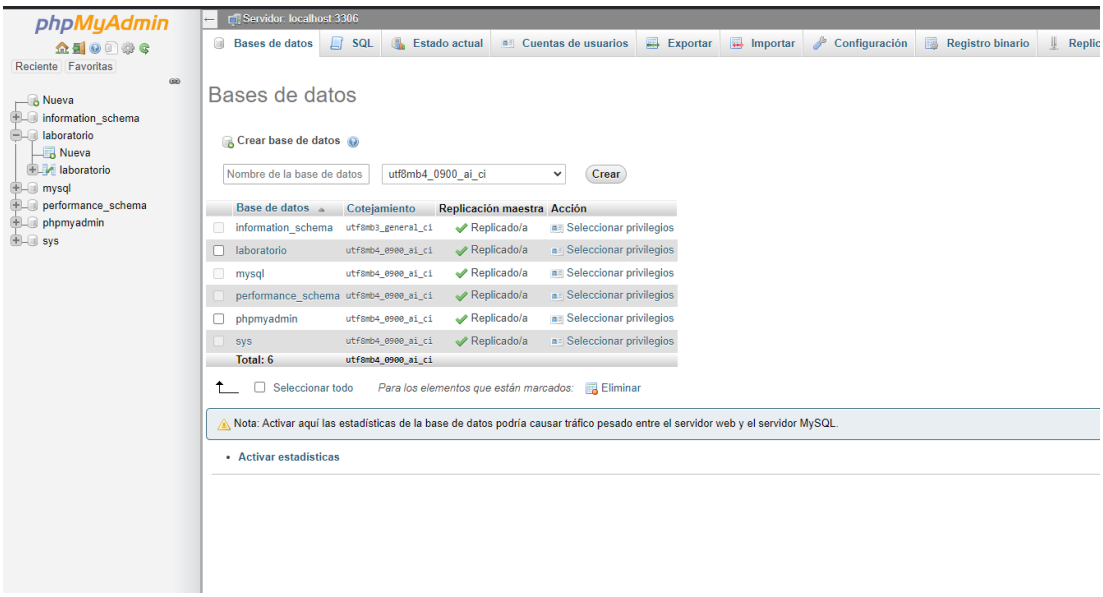


Figura 41. Phpmyadmin.

Se agrega la base de datos con el ID, PH, CE, NUT para guardar los datos del sistema, la Figura 42 muestra la configuración establecida.

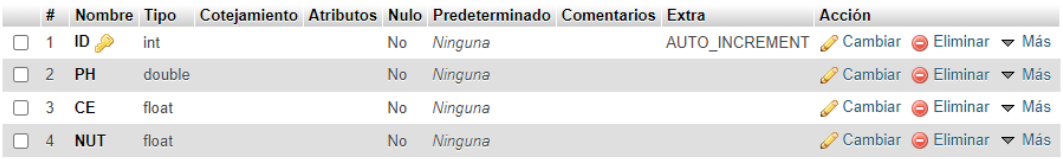


Figura 42. Estructura de la base de datos.

La Figura 43 presenta la forma de agregar los datos, configuración en sql e insert y obtenemos el comando para agregar los datos, que se implementará a través de Node-RED.

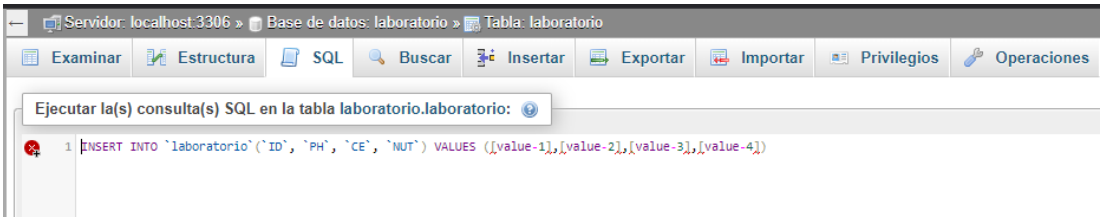


Figura 43. Comando para agregar datos.

Tras instalar Node-RED, se accede a través del puerto 1880. A continuación, se incorporan las librerías necesarias: node-red-contrib-mqtt para gestionar MQTT, node-red-dashboard para el muestreo y la creación de la interfaz de usuario, además de node-red-node-mysql para la conexión con la base de datos. Los datos enviados desde el dispositivo se reciben mediante el nodo mqtt in, configurado con el mismo broker y tema utilizado en el dispositivo. Una vez recibidos, los datos se convierten en objetos de JavaScript y se organizan según lo ilustrado en la Figura 44.

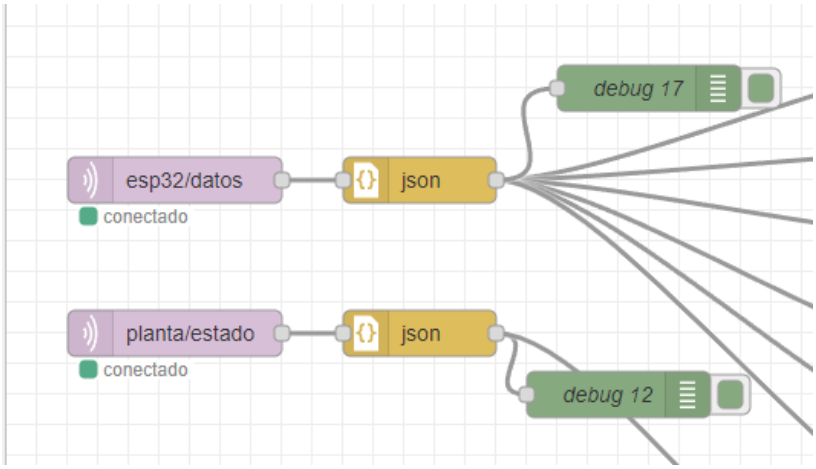


Figura 44. Ingreso de datos.

Para el procesamiento de los datos, primeo se inserta el bloque function y configuramos para obtener los datos de pH, CE y salida de nutrientes en el sistema para enviar los datos a la base de datos. La Figura 45 muestra la configuración de insertar los datos a la base de datos, tal como se mostró en la Figura 43.

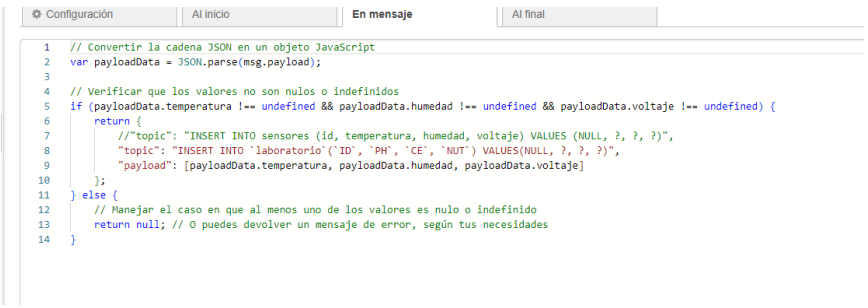


Figura 45. Obtención de datos para la base de datos.

Se conecta la función al bloque mysql y configurar la base de datos, colocando las credenciales. La Figura 46 presenta la conexión de la función y el bloque de datos, además de debug que imprime los valores enviados.

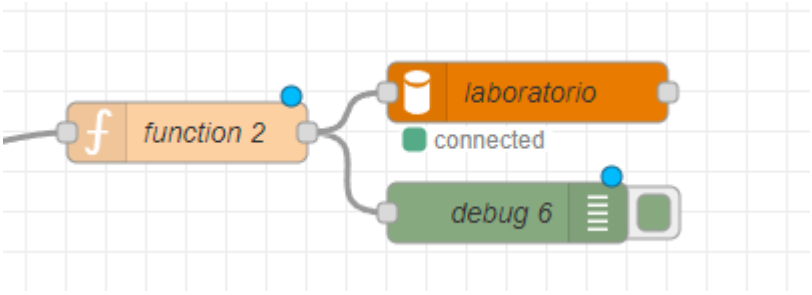


Figura 46. Envío de datos a la base de datos.

Se abre otra función, para obtener los datos de pH y CE y accionar la bomba dosificadora, cuando el pH es menor a 6 o la ce CE menor a 1200, se envía la alerta, la Figura 47 presenta las condiciones establecidas.

```
// Convertir la cadena JSON en un objeto JavaScript
var payloadData = JSON.parse(msg.payload);

// Verificar que los valores no son nulos o indefinidos
if (payloadData.ph !== undefined && payloadData.ce !== undefined && payloadData.nut !== undefined) {
    // Verificar si temperatura y humedad son menores a 5
    if (payloadData.ph < 6 || payloadData.ce < 1200) {
        return {
            "payload": "ON"  // Retorna "ON" si se cumple la condición
        };
    } else {
        return {
            "payload": "OFF"  // Retorna "OFF" si no se cumple la condición
        };
    }
} else {
    // Manejar el caso en que al menos uno de los valores es nulo o indefinido
    return null; // O puedes devolver un mensaje de error si lo prefieres
}

```

Figura 47. Parámetros para la alerta.

La Figura 48 muestra la obtención de datos de CE, lo mismo se realiza para pH, y los nutrientes aplicados.

```
// Convertir la cadena JSON en un objeto JavaScript
var payloadData = JSON.parse(msg.payload);

// Verificar que los valores no son nulos o indefinidos
if (payloadData.ph !== undefined && payloadData.ce !== undefined && payloadData.nut !== undefined) {
    // Verificar

        return {
            "payload": payloadData.ce
        };
    }
 else {
    // Manejar el caso en que al menos uno de los valores es nulo o indefinido
    return null; // O puedes devolver un mensaje de error si lo prefieres
}

```

Figura 48. Extracción de cada variable.

Se envía los datos de cada sensor a la nube de borde a través de cada tópico creado. La Figura 49 presenta los bloques conectados.

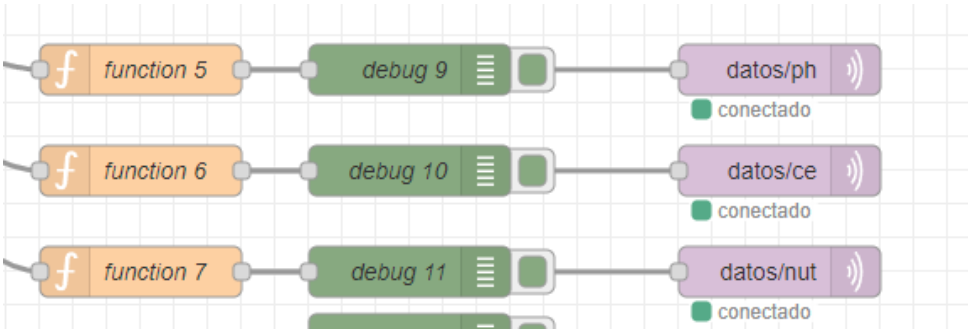


Figura 49. Envió de datos a la nube.

3.4.2 Entrenamiento del modelo de inteligencia artificial

Etapa 2 Fase de entrenamiento: para el dataset se optó por buscar en Roboflow, ya que tiene amplia cantidad de imágenes ya segmentadas. La Tabla 27 presenta las características del dataset. El cual contiene 7830 imágenes clasificadas por el tipo de enfermedad que contiene las lechugas,7 diferentes tipos.

Tabla 27. Características del Dataset.

Parametro	**Descripción**
Nombre	Disease lettuce Dataset
Autor	Yssp
Url	https://universe.roboflow.com/yssp/disease-lettuce-h69zb
Diario	Roboflow Universe
Año	2024
Visita	20 de octubre de 2024
Número de fotos	7830

Por lo visto en el capítulo 1 en la Tabla 2, Tabla 3, y la Tabla 4, se define las enfermedades que se va a encontrar que son: mancha foliar por Septoria, mancha foliar bacteriana y el mildiú velloso, por mayor presencia en ambientes húmedos. Se toma las imágenes disponibles, además de propias que permitan el mejor funcionamiento en el ambiente propio.

Roboflow permite hacer el etiquetado de las imágenes y asignar las clases correspondientes, permite asignar con precisión las zonas que se desea identificar, en primera se sube las imágenes, luego anotar a la clase de cada imagen y finalmente

subir al dataset, la Figura 50 muestra una prueba de etiquetado en una lechuga con mancha foliar bacteriana.

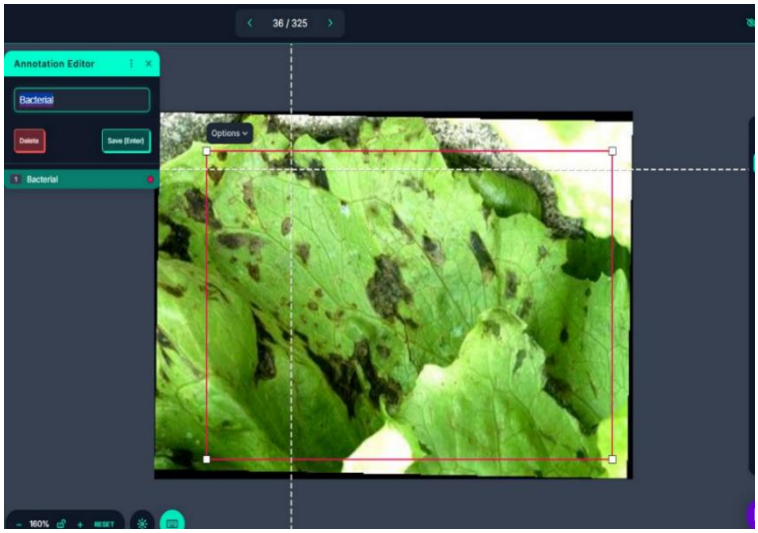


Figura 50. Proceso de etiquetado.

Roboflow además permite la asignación de la base de datos de forma automática, por lo que no se requiere asignar carpetas de imágenes y textos de asignación, además permite la exportación de diferentes modelos para el entrenamiento, como se muestra en la Figura 51.

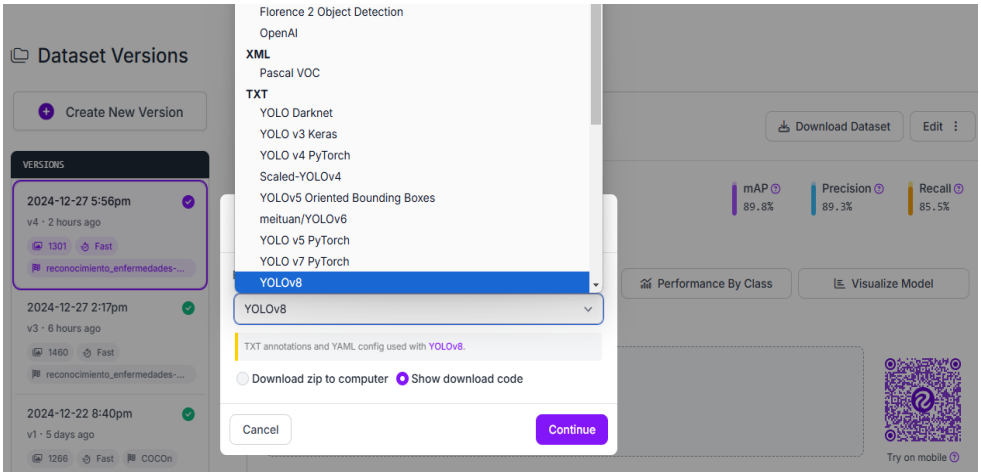


Figura 51. Exportación del dataset.

Además, Roboflow permite hacer un “Try Pre-Trained Model” y muestra cómo reacciona el dataset con un entrenamiento previo, la Figura 52 presenta los resultados obtenidos.



Figura 52. Verificación de eficiencia del dataset.

El resultado proporciona una eficiencia de 89.8 general del entrenamiento, el cual representa una buena eficiencia y veracidad de los datos, también informa que se requiere mínimo 60 epocas para comenzar a tener resultados confiables, proporcionado eso, se descarga para la versión de YOLOv8, disponible en la página para el uso en Kaggle. La Figura 53 presenta el dataset obtenido en la versión de YOLOv8.

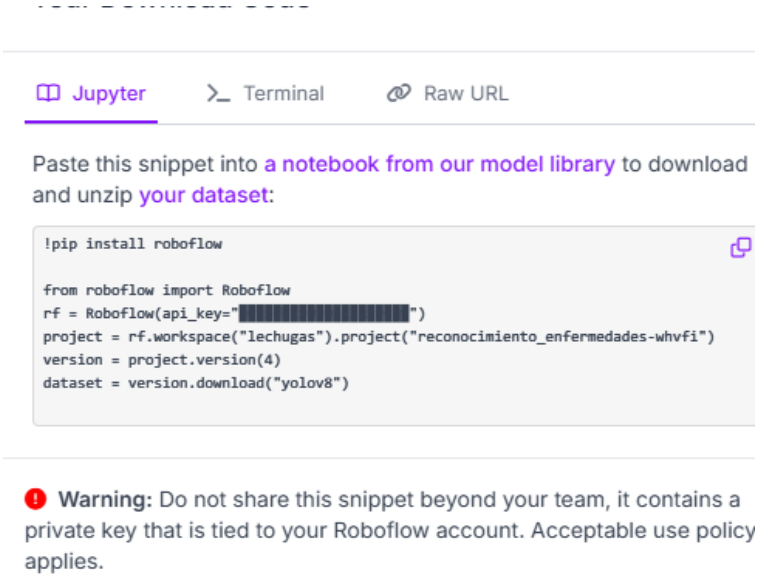


Figura 53. Dataset descargado.

En Kaggle importamos las librerías “!pip install ultralytics” que abre la biblioteca Ultralytics, que permite trabajar con YOLOv8 e ingresamos el dataset, como se muestra en la Figura 54.

```
!mkdir {HOME}/datasets
%cd {HOME}/datasets
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="aCtsf8IxOxOpK3mivf8y")
project = rf.workspace("lechugas").project("reconocimiento_enfermedades-whvfi")
version = project.version(4)
dataset = version.download("yolov8")
```

Figura 54. Dataset importado.

Para el entrenamiento se procede con los siguientes parámetros mostrado en la Figura 55.

```
%cd {HOME}

!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml epochs=40 imgsz=640 plots=True patience=10

/kaggle/working
Ultralytics 8.3.55 🚀 Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15095MiB)
engine/trainer: task=detect, mode=train, model=yolov8s.pt, data=/kaggle/working/datasets/Reconocimiento enfermedades-4/data.yaml, epochs=200, time=
```

Figura 55. Comando de entrenamiento.

Donde:

- model=yolov8s.pt: Utiliza el modelo preentrenado YOLOv8 small (yolov8s.pt) como base.
- data={dataset.location}/data.yaml: Especifica la ubicación del archivo data.yaml que contiene la información del conjunto de datos.
- epochs=40: Define que el entrenamiento se realizará durante 40 épocas.
- imgsz=640: Establece el tamaño de las imágenes a 800 píxeles.
- patience=10: detiene el entrenamiento si el modelo no mejora en 10 épocas consecutivas.

Para la primera prueba se realizó con 40 épocas, la Figura 56 muestra los resultados del primer entrenamiento.

```
                 all        264        328      0.887      0.855      0.905      0.477

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      38/40      4.04G      1.261     0.6923      1.868          7        640: 1
                 Class     Images  Instances      Box(P          R      mAP50  m
                   all        264        328      0.879      0.854      0.901      0.466

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      39/40      3.84G      1.247     0.6585      1.856          9        640: 1
                 Class     Images  Instances      Box(P          R      mAP50  m
                   all        264        328      0.902      0.865      0.913      0.475

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      40/40      4.07G      1.232      0.633      1.856          6        640: 1
                 Class     Images  Instances      Box(P          R      mAP50  m
                   all        264        328      0.886      0.857      0.904      0.476

40 epochs completed in 0.209 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 22.5MB
Optimizer stripped from runs/detect/train/weights/best.pt, 22.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.55 🚀 Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 168 layers, 11,127,132 parameters, 0 gradients, 28.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  m
                   all        264        328      0.888      0.855      0.905      0.478
             Bacterial         48         49        0.9      0.857      0.932      0.439
Downy_mildew_on_lettuce         72         81      0.818      0.852      0.868      0.454
               Healthy         77        124      0.865      0.876      0.898      0.443
Septoria_Blight_on_lettuce         71         74      0.969      0.834      0.922      0.575
Speed: 0.2ms preprocess, 5.2ms inference, 0.0ms loss, 2.6ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

Figura 56. Proceso de entrenamiento.

Los gráficos generados durante el entrenamiento del modelo YOLOv8s ilustran el avance en las fases de entrenamiento y validación. La Tabla 28 presenta los resultados obtenidos en 40 épocas y la Tabla 29 con 60 épocas.

Tabla 28. Resultados con 40 épocas.

Clase	**Imagen**	**Instancia**	**Box(P)**	**R**	**mAP50**	**mAP50-95**
All	264	328	0.888	0.855	0.905	0.478
Bacterial	48	49	0.9	0.857	0.932	0.439
Midiú velloso en lechuga	72	81	0.818	0.852	0.868	0.454
Tizón de Septoriosis en lechuga	71	74	0.969	0.834	0.898	0.575
Normal	77	124	0.865	0.876	0.922	0.443

Tabla 29. Resultado con 60 épocas.

Clase	**Imagen**	**Instancia**	**Box(P)**	**R**	**mAP50**	**mAP50-95**
All	264	328	0.908	0.866	0.914	0.485
Bacterial	48	49	0.954	0.898	0.962	0.456
Midiú velloso en lechuga	72	81	0.874	0.858	0.931	0.489
Tizón de Septoriosis en lechuga	71	74	0.975	0.851	0.936	0.585
Normal	77	124	0.829	0.855	0.828	0.412

Los resultados obtenidos con 30 y 40 épocas muestran que existe un error significativo en la detección de enfermedades. Por eso, se realizó un nuevo entrenamiento con 200 épocas, sin embargo, el programa se detuvo a las 94 épocas, debido que no mejora los resultados, teniendo una duración de 0.451 horas (27 minutos), los resultados se evidencian en la Tabla 30.

Tabla 30. Resultado con 94 épocas.

Clase	**Imagen**	**Instancia**	**Box(P)**	**R**	**mAP50**	**mAP50-95**
All	264	328	0.93	0.855	0.926	0.485
Bacterial	48	49	0.971	0.857	0.955	0.46
Midiú velloso en lechuga	72	81	0.918	0.877	0.936	0.473
Tizón de Septoriosis en lechuga	71	74	0.968	0.865	0.935	0.576
Normal	77	124	0.864	0.822	0.876	0.432

Donde:

- Clase: Representa el nombre de la categoría que se analiza en las imágenes.
- Imágenes: Cantidad total de imágenes empleadas durante el entrenamiento.
- Instancia: Total de objetos pertenecientes a todas las clases presentes en las imágenes.
- P: Precisión obtenida en las predicciones de los cuadros delimitadores.
- R: Sensibilidad alcanzada en las predicciones de los cuadros delimitadores.
- mAP50: Promedio de precisión utilizando un umbral de intersección sobre unión (IoU) de 0.50.
- mAP50-95: Promedio de precisión considerando un rango de umbral de intersección sobre unión desde 0.50 hasta 0.95, con incrementos de 0.05.

La Figura 57 presenta el progreso de varias pérdidas y métricas analizadas durante el proceso de entrenamiento y validación de un modelo de Deep Learning para detección de objetos a lo largo de 96 épocas. Las pérdidas de entrenamiento, que incluyen las de

cajas delimitadoras (train/box_loss), clasificación (train/cls_loss) y distribución focal (train/dfl_loss), disminuyen de forma constante, indicando mejoras en la capacidad del modelo para predecir con mayor precisión las posiciones y tamaños de las cajas, identificar correctamente las clases de los objetos y asignar pesos adecuados dentro de las cajas delimitadoras. Las métricas de precisión (metrics/precision(B)) y recall (metrics/recall(B)) aumentan de manera notable, reflejando una disminución en los falsos positivos y negativos, y mejorando la identificación precisa de los objetos presentes.

Durante la validación, se sigue una tendencia similar, donde las pérdidas (val/box_loss, val/cls_loss, val/dfl_loss) también disminuyen de forma consistente, demostrando que el modelo mantiene un buen desempeño. Asimismo, las métricas de precisión media promedio para IoU al 50% (metrics/mAP50(B)) y entre 50%-95% (metrics/mAP50-95(B)) mejoran de manera significativa, alcanzando altos valores al final del entrenamiento, lo que demuestra un rendimiento sólido en diferentes niveles de umbral.

En conjunto, estos resultados reflejan una mejora continua tanto en las fases de entrenamiento como de validación, indicando que el modelo se ajusta eficazmente a los datos y logra un alto grado de precisión en la detección de objetos.

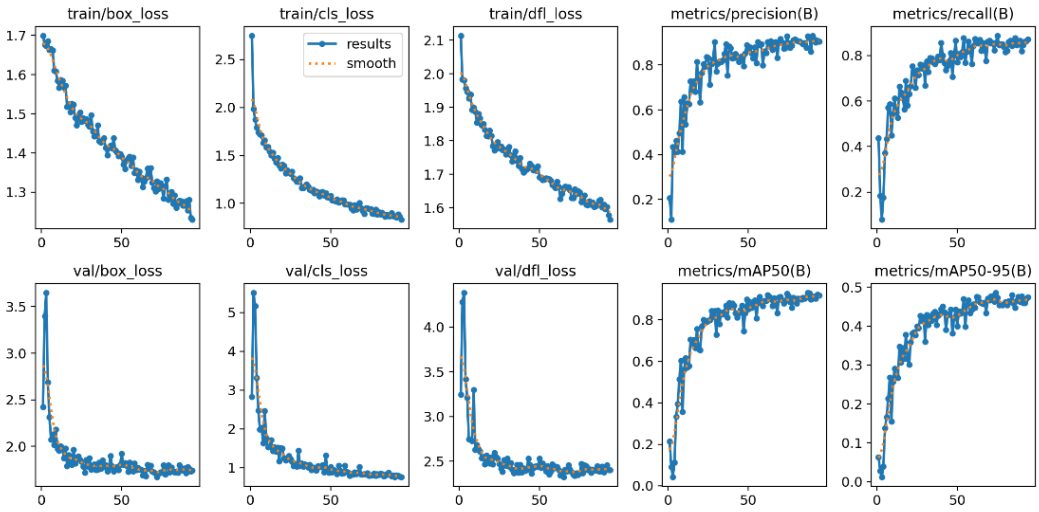


Figura 57. Métricas de rendimiento del modelo entrenado.

La Figura 58 presenta los resultados de la matriz de confusión normalizada, que evalúa el desempeño del sistema en la clasificación de enfermedades en cultivos de lechuga. Los resultados muestran que el modelo alcanza una precisión del 90% con un margen de error del 10% para la enfermedad bacteriana, mientras que el Mildiu velloso en lechuga se clasifica con una efectividad del 94% y un error del 6%. Para el Tizón de septoria, la precisión es del 86% con un margen de error del 14%, y la clasificación de lechugas saludables logra una efectividad del 90% con un error del 10%.

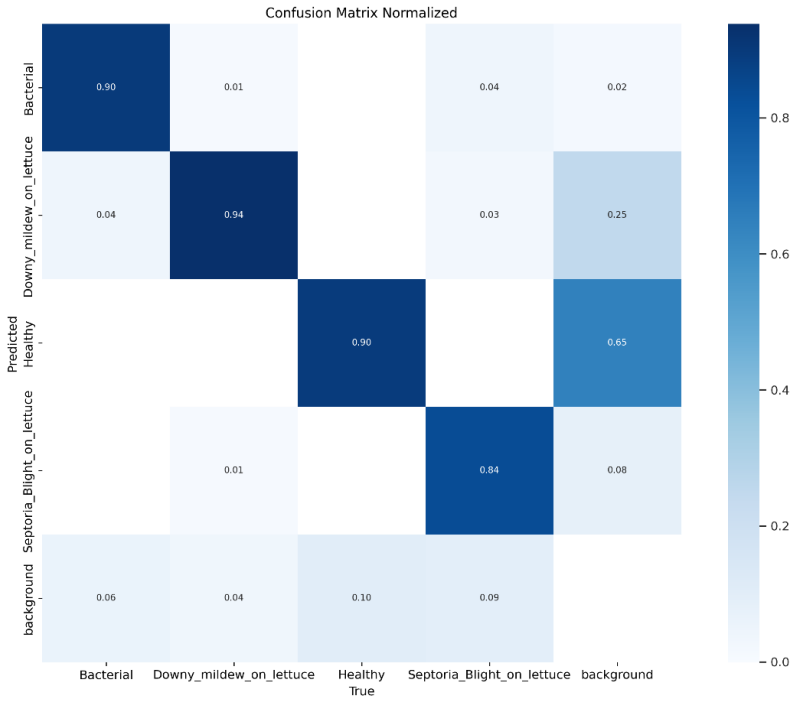


Figura 58. Matriz de confusión normalizada.

La Figura 59 presenta la matriz de confusión, donde se detalla la distribución de la cantidad de imágenes clasificadas en cada una de las clases. Esta herramienta permite visualizar tanto las predicciones correctas como los errores del modelo, proporcionando un análisis claro del desempeño en la clasificación de las diferentes categorías.

La Tabla 31 muestra los valores correspondientes a los aciertos y errores de cada una de las clases, proporcionando un análisis detallado del desempeño del modelo en términos de precisión y fallos en la clasificación.

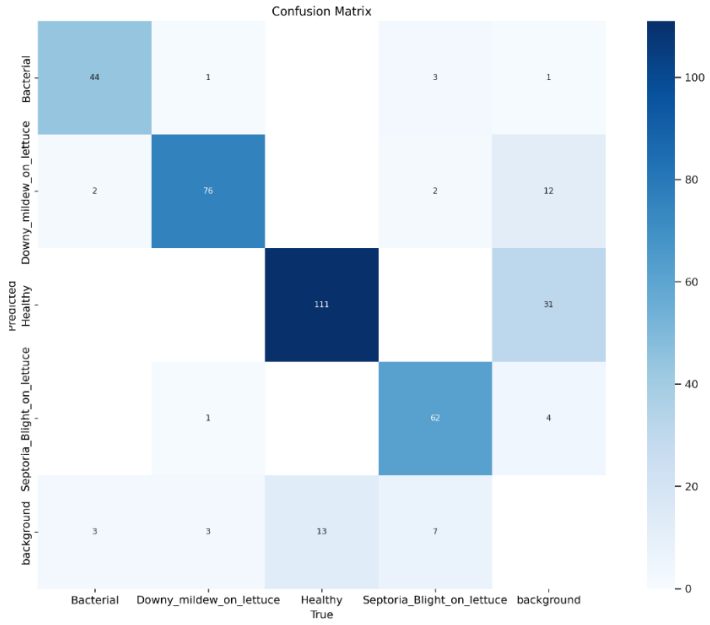


Figura 59. Matriz de confusión.

Tabla 31. Datos de la matriz de confusión.

Clases	**Aciertos**	**Errores**
Bacterial	44	2
Mildiu velloso en lechuga	76	2
Tizón de septoria en lechuga	62	5
Normal	111	0
Promedio	293	9

La Figura 60 muestra la curva de presión del entrenamiento. El gráfico de la curva de precisión-confianza (Precision-Confidence Curve) mostrado refleja el desempeño del modelo YOLOv8 al evaluar la precisión de sus predicciones en función de distintos niveles de confianza. En el eje X se observa la confianza de las predicciones, mientras que el eje Y representa la precisión alcanzada.

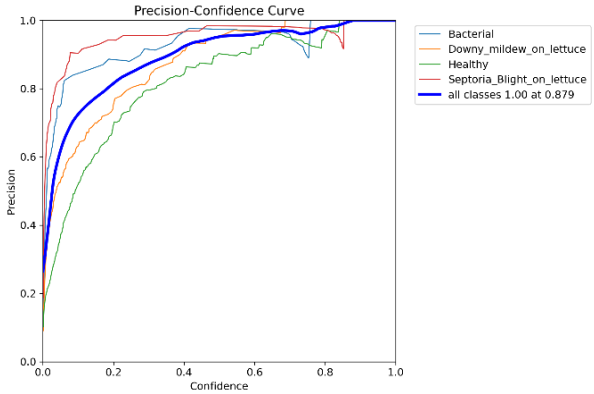


Figura 60. Curvatura de precisión del entrenamiento.

Cada curva en el gráfico corresponde a una clase específica: Bacterial, Downy_mildew_on_lettuce, Septoria_Blight_on_lettuce y normal. La línea azul gruesa indica la precisión general combinada para todas las clases, logrando un rendimiento óptimo con una precisión prefecta de 1.00 con 0.879 de confianza. Las curvas más delgadas ilustran el comportamiento individual por clase, permitiendo analizar variaciones en la precisión según el tipo de predicción.

Este gráfico es fundamental para determinar el umbral de confianza ideal que maximice la precisión, minimizando falsas predicciones y mejorando el balance entre precisión y sensibilidad del modelo. Además, proporciona información clave para identificar áreas de mejora y garantizar un rendimiento adecuado en aplicaciones prácticas relacionadas con la clasificación de enfermedades en lechugas.

3.4.3 Implementación del entrenamiento

Etapa 3 Procesamiento de datos: Para la implementación del sistema en la Raspberry pi 4, se programó en Python y se importó los archivos del entrenamiento, usando las librerías de cv2, ultralytics, mqtt y base64. Cargar el modelo entrenado llamado best.pt y obtener la imagen a través de la cámara conectada. En el Anexo J se presenta el código completo.

Se guarda la imagen original y se convierte a HSV para trabajar con el canal de brillo, ajustar la imagen a 640, y etiquetar los niveles de confianza, dibujando los reconocimientos y observar si contiene alguna anomalía, la toma de imagen se realiza cada hora del día ya que no se requiere de un monitoreo en tiempo real de las plantas, la Figura 61 presenta el etiquetado de la imagen con el reconocimiento realizado.

```
for box in results[0].boxes:
    x1, y1, x2, y2 = map(int, box.xyxy[0].tolist())
    area = (x2 - x1) * (y2 - y1)
    conf = round(float(box.conf.item()), 2)
    label = results[0].names[int(box.cls)]
    if conf > 0.3 and min_area < area < max_area:
        detecciones.append((x1, y1, x2, y2, conf, label))

# Dibujar resultados en la imagen original
for (x1, y1, x2, y2, conf, label) in detecciones:
    print(f"Etiqueta: {label}, Confianza: {conf:.2f}, Coordenadas: ({x1}, {y1}), ({x2}, {y2})")
    cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.putText(image, f"{label} {conf:.2f}", (x1, y1 - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
```

Figura 61. Reconocimiento de enfermedad.

La imagen y el estado se envía por mqtt al servidor donde se guarda en la base de datos el estado de las plantas, además de enviar a la nube para su visualización. La Figura 62 muestra el envío de la imagen captada, la procesada y el estado resultante.

```python
def enviar_a_mqtt(base_image, processed_image, detecciones):
    """Envía las imágenes y detecciones a Node-RED mediante MQTT."""
    # Publicar imagen base en formato base64
    mqtt_client.publish(topic_base_image, base_image)
    print("Imagen base enviada a Node-RED.")

    # Publicar imagen procesada en formato base64
    mqtt_client.publish(topic_processed_image, processed_image)
    print("Imagen procesada enviada a Node-RED.")

    # Agrupar todas las detecciones en un solo mensaje
    detecciones_json = [
        {
            "etiqueta": d[5],
            "confianza": d[4],
            "coordenadas": {"x1": d[0], "y1": d[1], "x2": d[2], "y2": d[3]}
        }
        for d in detecciones
    ]
    mqtt_client.publish(topic_detections, json.dumps(detecciones_json))
    print(f"Publicado: {detecciones_json}")
```

Figura 62. Envío de datos.

En nodered, se configura el recibimiento de datos para el almacenamiento. La Figura 63 presenta la asignación de los datos a las variables creadas en la base de datos.

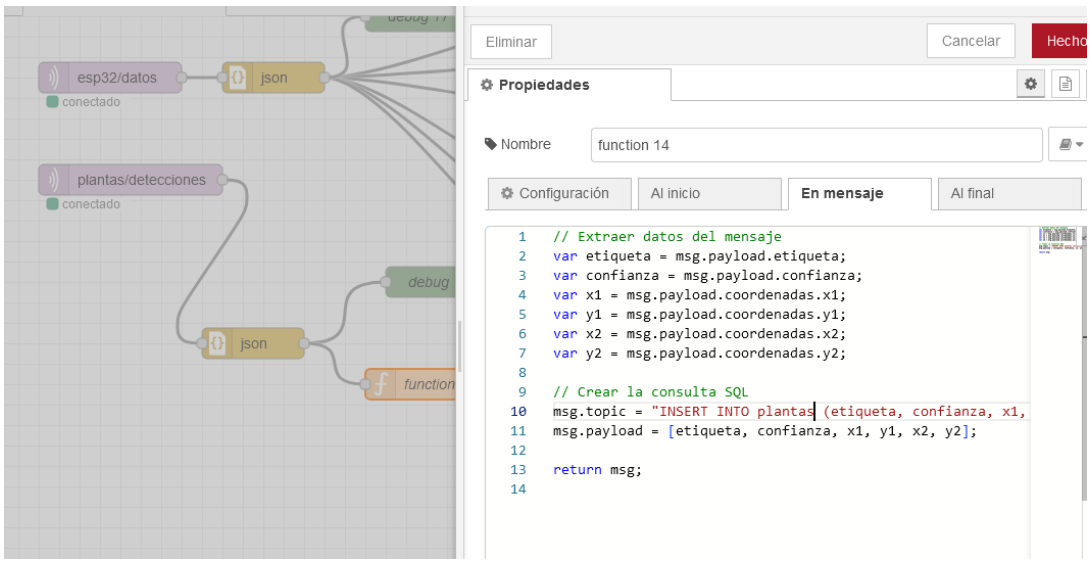


Figura 63. Asignación de valores a la base de datos.

3.5 Implementación de Cloud layer y Display layer

Para la implementación de la nube, se levantó openstack, el cual permite gestionar la red de forma virtual, se levanta un router y servidores, la Figura 64 presenta la red implementada.

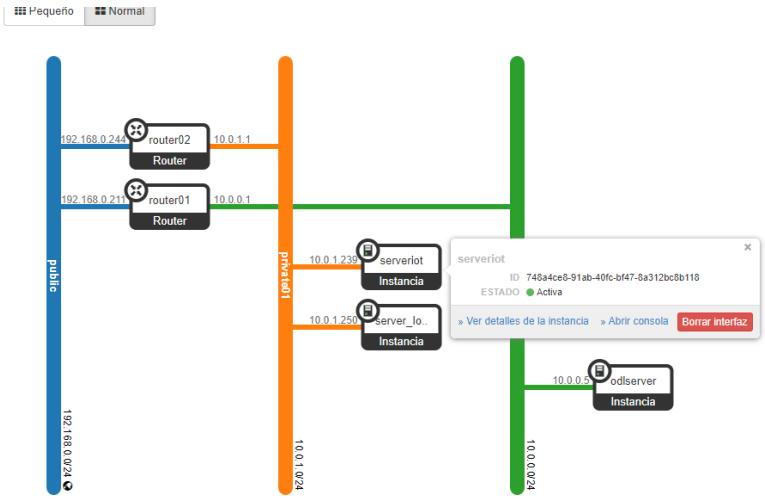


Figura 64. Red configurada.

Para habilitar la máquina virtual en el router se coloca “ssh -i /root/.ssh/id_rsa_new ubuntu@192.168.0.249”, habilitando su uso y con “openstack server start server_IoT_A” la máquina enciende. Para la administración de puertos de entrada y salida se asigna al grupo de seguridad los puertos habilitados para entradas y salidas. Las configuraciones de los puertos se encuentran en secgroup02, aplicando las reglas de grupo, la Figura 65 presenta las reglas aplicadas.

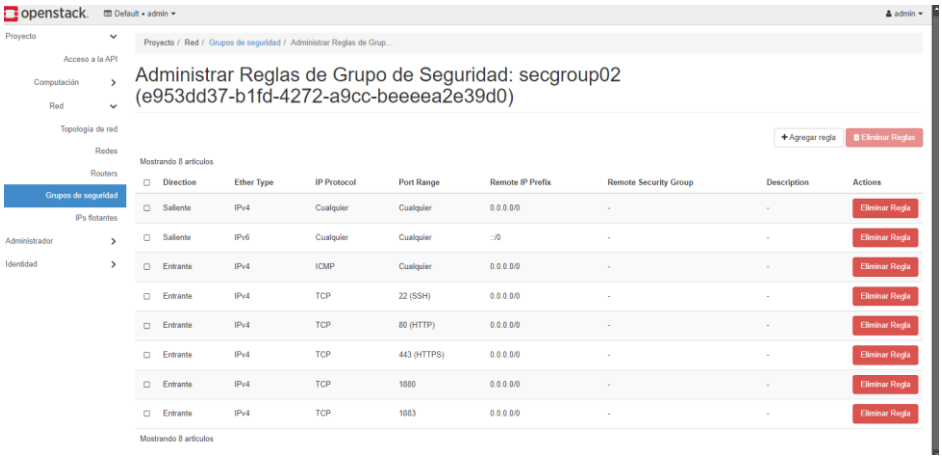


Figura 65. Reglas de seguridad.

La Figura 66 presenta los puertos habilitados para el uso, además del espacio asignado, 2GB de ram y 40GB de almacenamiento, además de la imagen cargada, Ubuntu 20.04 para el servido que se va a usar.



Figura 66. Configuraciones del servidor.

Levantado el servicio de nodered, se configura para la visualización de los datos procesados en la etapa de server layer, la Figura 67 presenta los bloques configurados para la visualización final al usuario.

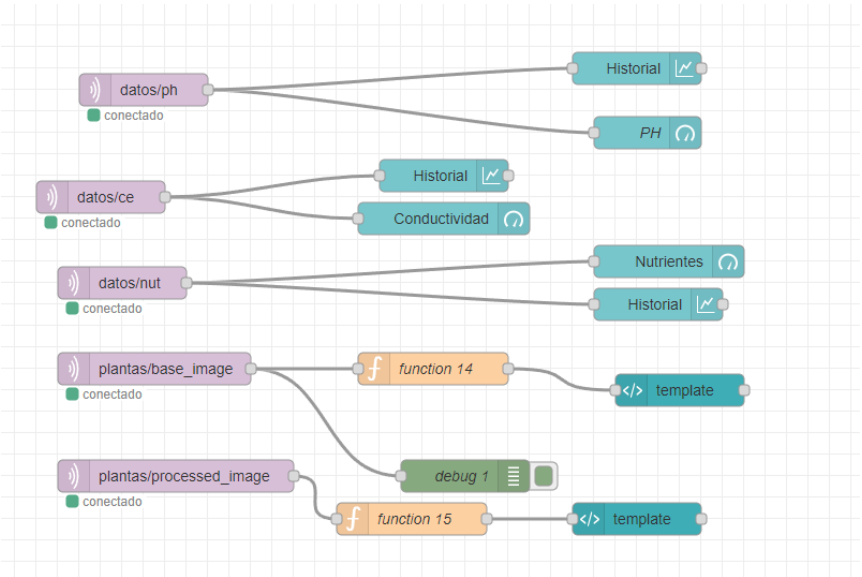


Figura 67. Configuración de nodered.

Para el acceso a visualizar los datos desde cualquier parte con internet, se usa ZeroTier, el cual permite la comunicación de distintas redes, se crea una red virtual para que los dispositivos puedan ingredas ingresar, se instala ZeroTier en los dispositivos que se desea e ingresa el ID de la red creada, como se presenta en la Figura 68.

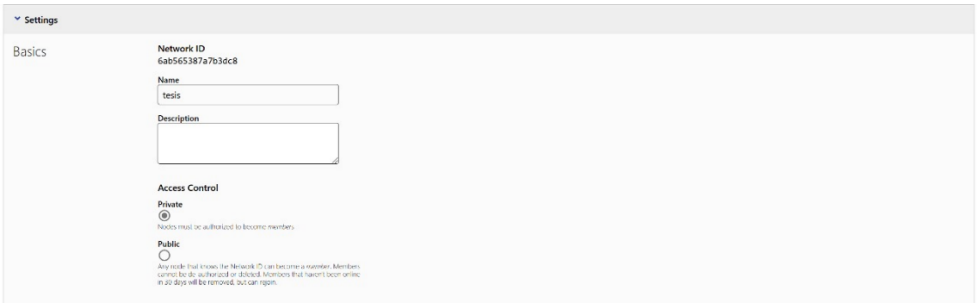


Figura 68. ID de la red.

La dirección IP con la que se desea trabajar puede ser configurada. Esto permite establecer las direcciones IP específicas para los usuarios autorizados, definiendo los puntos de acceso necesarios para la conexión. Además, se pueden gestionar las reglas que determinan qué usuarios están habilitados para comunicarse entre sí dentro de la red, asegurando un control adecuado sobre las interacciones y accesos en el sistema, como se muestra en la Figura 69.

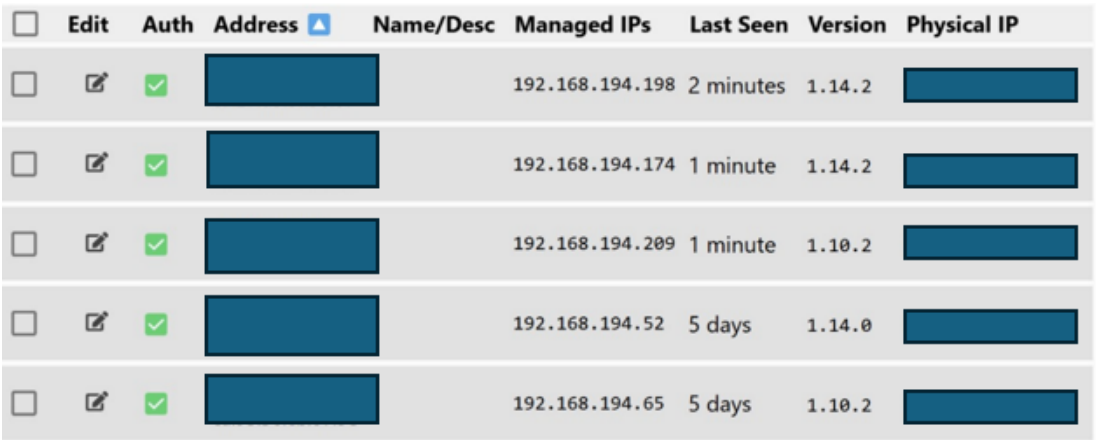


Figura 69. Dispositivos dentro de la red.

3.6 Validación de resultados

Para comprobar el funcionamiento del sistema se realizaron pruebas presentados en el Anexo H. En la Figura 70 se presenta la implementación en el cultivo.

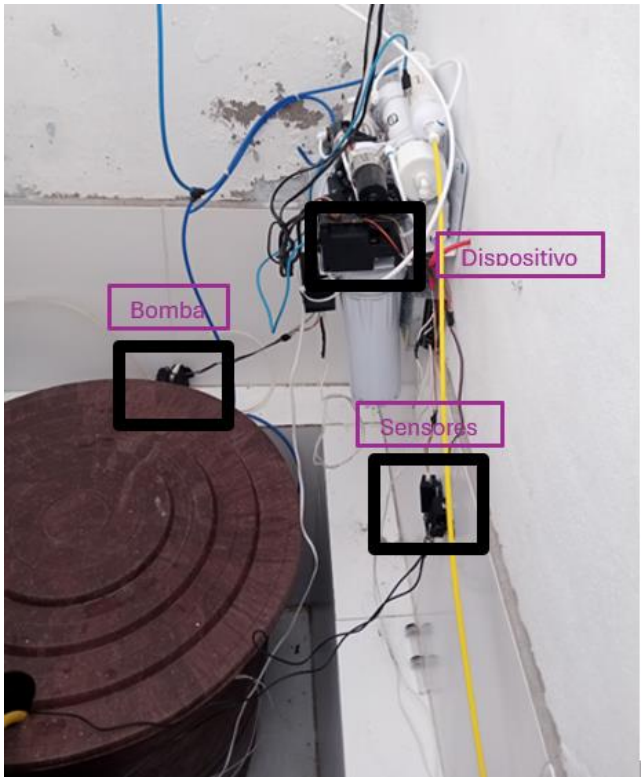


Figura 70. Implementación en el cultivo.

Para evaluar los resultados obtenidos de los sensores, se realiza una comparación con 5sensores comerciales, analizando diversos parámetros de error y fiabilidad. En este análisis, el error absoluto se calcula utilizando la Ecuación (1), el error relativo se determina mediante la Ecuación (2), el porcentaje de error se obtiene con la Ecuación (3) y la fiabilidad se mide aplicando la Ecuación (4).

$$\boldsymbol{Error\ absoluto = |Valor\ exacto - Valor\ real|} \quad (1)$$

$$\boldsymbol{Error\ relativo = \frac{Error\ absoluto}{Valor\ exacto}} \quad (2)$$

$$\boldsymbol{Porcentaje\ Error\ = Error\ relativo * 100} \quad (3)$$

$$\boldsymbol{Fiabilidad\ = 100 - Porcentaje\ error} \quad (4)$$

La Figura 71 presenta la comparación entre el dato obtenido del sensor de pH y el registrado por un sensor comercial, donde el primero muestra un valor de pH de 5.13 y el segundo de 5.11, la Tabla 32 muestra la recolección de los datos en distintos parámetros, con el error absoluto y error relativo entre el sensor usado y el sensor comercial. La fiabilidad del sensor pH es de 99.76%.

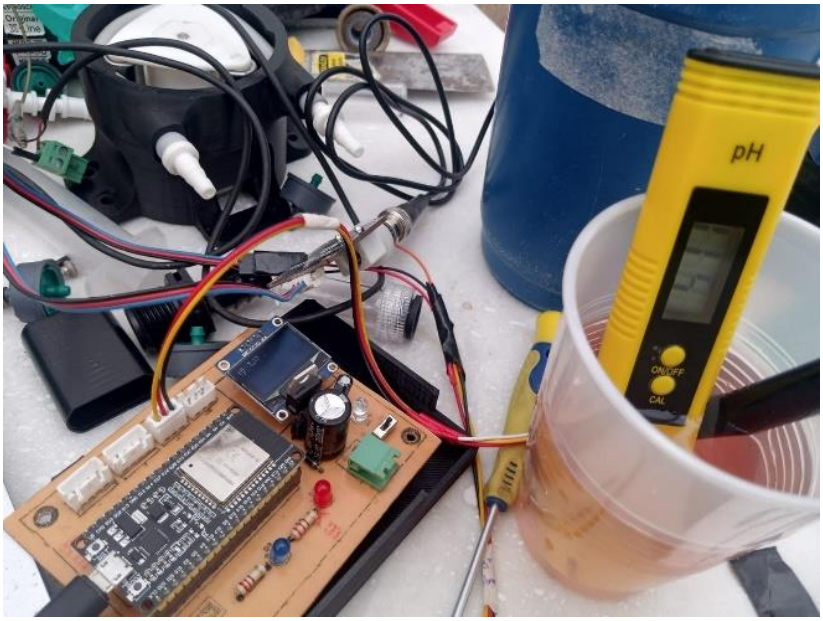


Figura 71. Datos del sensor pH.

Tabla 32. Comparativa del sensor pH

Número de muestra	Valor del sensor [pH]	Valor de sensor commercial [pH]	Error absoluto	Error relativo	Porcentaje error
1	4	4	0	0	0%
2	4.50	4.51	0.01	0.0022	0.22%
3	5.13	5.11	0.02	0.0039	0.39%
4	5.82	5.80	0.02	0.0034	0.34%
5	6.41	6.40	0.01	0.0015	0.15%
6	7	7	0	0	0%
7	7.41	7.40	0.01	0.0013	0.13%
8	7.87	7.85	0.02	0.0025	0.25%
9	8.15	8.10	0.05	0.0061	0.61%
10	9.02	9	0.02	0.0022	0.22%
				PROMEDIO ERROR (%)	0.23%
				FIABILIDIDAD (%)	99.76%

La Figura 72 muestra los datos recopilados por el sensor de conductividad eléctrica (CE) del dispositivo desarrollado y un sensor comercial, donde el dispositivo registra un valor de 1825.43 µS/cm y el sensor comercial un valor de 1858 µS/cm. Por otro lado, la Tabla 33 presenta una comparación detallada entre ambos sensores bajo distintas condiciones, evidenciando la precisión y consistencia del sensor de CE desarrollado. La fiabilidad del sensor se estima en un 98.80%, demostrando su alto nivel de desempeño y confiabilidad en las mediciones.



Figura 72. Datos del sensor de CE.

Tabla 33. Comparativa del sensor de CE.

Número de muestra	Valor del sensor CE [us/cm]	Valor de sensor comercial [us/cm]	Error absoluto	Error relativo	Porcentaje error
1	18	20	1 x 10^-6	0.05	5%
2	820	845	25 x 10^-6	0.295	2.95%
3	1150	1145	5 x 10^-6	0.0043	0.43%
4	1420	1413	7 x 10^-6	0.0049	0.49%
5	1825	1858	33 x 10^-6	0.0177	1.77%
6	2101	2110	9 x 10^-6	0.0026	0.26%
7	3409	3420	11 x 10^-6	0.0032	0.32%
8	8520	8560	40 x 10^-6	0.0046	0.46%
9	10320	10302	18 x 10^-6	0.0017	0.17%
10	12890	12880	10 x 10^-6	0.0007	0.07%
				PROMEDIO ERROR	1.192%
				FIABILIDIDAD	98.808%

Para los datos del reconocimiento de las enfermedades se toma 35 muestras que se encuentran las enfermedades, septoria de lechuga, mudiu velloso, mancha bacterial y normal. La Figura 73 presenta el reconocimiento de la lechuga sana, la Figura 74 sobre el reconocimiento de midui velloso, la Figura 75 la mancha foliar por septoria y la Figura 76 mancha foliar bacteriana.

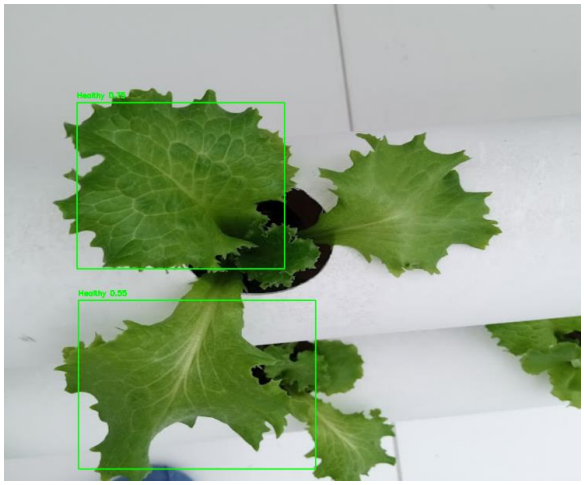


Figura 73. Reconocimiento lechuga sana.



Figura 74. Reconocimiento de midiú velloso.



Figura 75. Reconocimiento de mancha foliar por Septoria.

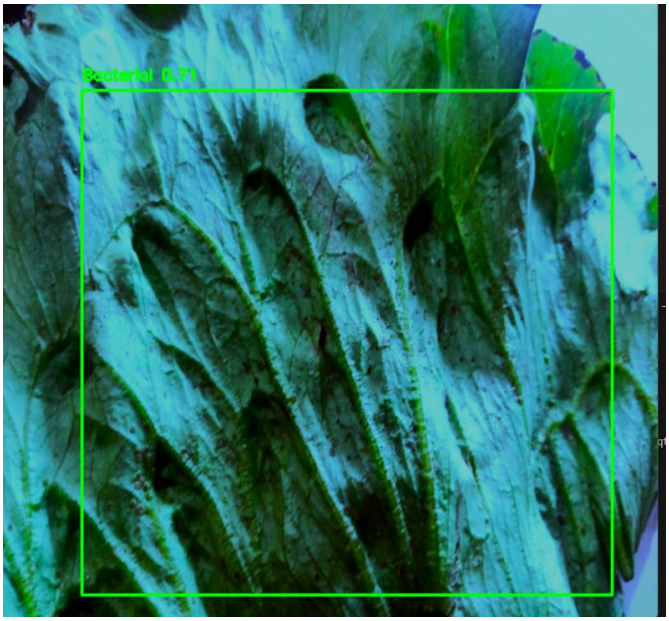


Figura 76. Reconocimiento de mancha foliar bacteriana.

Se realizaron 35 pruebas y la comparación entre resultados reconocidos por el experto mostrado la validez en el Anexo A y el programada diseñado, los datos se muestran en la Tabla 34.

Tabla 34. Detección mediante imágenes.

Clase	**Número de muestras**	**Reconocimiento por Experto**	**Reconocimiento por Ia**	**Error absoluto**	**Error relativo**	**Porcentaje error**
Mildiú velloso	8	8	7	1	0.12	12.5%
Mancha foliar por Septoria	7	7	6	1	0.14	14.28%
Mancha foliar bacteriana	8	8	6	2	0.25	25%
Normal	12	12	11	1	0.083	8.3%
Total muestras	35	**PROMEDIO ERROR**				15.02%
		FIABILIDAD				84.92%

La fiabilidad de cada clase queda: Mildiú velloso con 87.5%, Mancha foliar por con Septoria: 85.72%, Mancha foliar bacteriana con 75% y Normal: 91.7% dando una fiabilidad promedio de 84.92%. Los datos se almacenan en la base de datos en el servidor local. La Figura 77 presenta los datos recolectados a través del tiempo.



	ID	PH	CE	NUT
Editar Copiar Borrar	1	7.15	577	0
Editar Copiar Borrar	2	7.15	549.15	3
Editar Copiar Borrar	3	7.14	522.69	6
Editar Copiar Borrar	4	7.14	497.56	9
Editar Copiar Borrar	5	7.14	473.68	12
Editar Copiar Borrar	6	7.14	451	15
Editar Copiar Borrar	7	7.17	429.45	18
Editar Copiar Borrar	8	7.17	408.98	21
Editar Copiar Borrar	9	7.16	389.53	24

+ Opciones

	ID	etiqueta	confianza	x1	y1	x2	y2
Editar Copiar Borrar	1	Healthy	0.4	439	335	552	402
Editar Copiar Borrar	2	Healthy	0.38	436	535	549	633
Editar Copiar Borrar	3	Healthy	0.35	232	311	305	362
Editar Copiar Borrar	4	Healthy	0.31	361	433	441	515
Editar Copiar Borrar	5	Healthy	0.3	352	267	462	313
Editar Copiar Borrar	6	Healthy	0.3	303	73	381	119
Editar Copiar Borrar	7	Healthy	0.3	292	216	386	261
Editar Copiar Borrar	8	Healthy	0.26	247	175	320	224
Editar Copiar Borrar	9	Healthy	0.26	269	370	379	419

Figura 77. Datos almacenados.

La Figura 78 muestra la interfaz gráfica al usuario para la visualización de los datos de pH, CE, nutrientes aplicados y el estado de las plantas.



Figura 78. Reconocimiento en tiempo real.

3.7 Discusión

El presente estudio, basando con lo realizado por Diego Bustamante en 2022, incorpora mejoras significativas en el diseño del sistema de cultivo, incluyendo la adición de una bomba dosificadora automatizada. Esta innovación permite la inserción precisa y controlada de las soluciones nutritivas necesarias para mantener condiciones óptimas para el desarrollo de los cultivos, reduciendo la intervención humana y aumentando la eficiencia del sistema.

Los resultados obtenidos destacan la alta precisión de los sensores utilizados en el monitoreo de variables críticas. El sensor de pH alcanzó una eficiencia del 99.76%, mientras que el sensor de conductividad eléctrica obtuvo una eficiencia del 98.80%.

Estos valores reflejan la confiabilidad de los componentes del sistema y su capacidad para operar con exactitud en entornos reales.

Además, el sistema diseñado se benefició de la integración de tecnologías IoT avanzadas, lo que permite no solo un monitoreo en tiempo real de parámetros clave, sino también un control dinámico de las condiciones del cultivo mediante ajustes automáticos. Esto, en combinación con la bomba dosificadora, asegura la aplicación oportuna y precisa de nutrientes esenciales, promoviendo un crecimiento uniforme y saludable del cultivo. Este sistema muestra avances tanto en la precisión de los sensores como en la automatización del proceso, demostrando su potencial para mejorar significativamente la sostenibilidad y productividad.

Así mismo, tomando en referencia el trabajo realizado por Jorge Andrés Criollo Navarrete que se centró en la detección de una única enfermedad (Erysiphe cichoracerum), en este estudio se amplió el alcance para abordar tres tipos de enfermedades comunes en cultivos de lechuga: mancha foliar por Septoria, mancha foliar bacteriana y el mildiú velloso. Este enfoque más integral permite un monitoreo más robusto y versátil, adaptándose a los desafíos reales que enfrentan los cultivos.

El sistema desarrollado logró una eficiencia del 84.92% en la detección de enfermedades bajo condiciones reales, lo cual es altamente competitivo dada la complejidad inherente al análisis de múltiples patologías. Este desempeño destaca el potencial del sistema para su implementación en entornos agrícolas reales, donde los cultivos enfrentan una considerable variabilidad en factores como la humedad, la iluminación y los niveles de nutrientes, demostrando su capacidad para adaptarse a condiciones dinámicas y exigentes.

3.8 Costos

3.8.1 Insumos del proyecto

La Tabla 35 presenta el desglose de los costos asociados al monitoreo y control del sistema, proporcionando una visión detallada de los recursos financieros necesarios para su implementación y operación.

Tabla 35. Costo de insumos.

MATERIAL	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
Sensor DFRobot Gravity EC Sensor V2.0	1	60	60
Sensor DIY SEMSPr pH-4502C	1	50	50
NEMA 47hd4027	1	18	18
Driver TB6560	1	16	16
Node MCU Esp32	1	13	13
7805	1	0.75	0.75
Diodo led	3	0.10	0.30
Switch	1	0.30	0.30
Resistencia	2	0.10	0.10
Diodo	1	0.10	0.10
Bornera	5	0.75	3.25
Pantalla oled 128x64	1	6.50	6.50
Capacitor	1	0.50	0.50
Carcasa	5	7.50	32.50
Raspberry pi 4	1	140	140
Raspberry pi camera v1.3	1	28	28
Fuente 24V	1	18	18
Baquelita	1	3	3
Cloruro ferrico	1	1	1
Computadora	1	850	850
Alimentación	4	75	300
Impresiones	20	1	20
Transporte	37.50	4	150
Plan internet	25	4	100
		Total	1811.30

3.8.2 Costo del desarrollo técnico

Para obtener los valores la Tabla 36 presenta el costo en relación con las horas laborales empleadas.

Tabla 36. Costo en relación con las horas laborales ocupadas.

Descripción	Tiempo ocupado
Desarrollo de dispositivo	20 días
Desarrollo de algoritmo	30 días
Total	50 días
Horas laborales	8
Horas totals	400 horas

El salario de un ingeniero en Telecomunicaciones según el Ministerio de Trabajo es de $494.51 en 2024 [76]. El salario mínimo por día es de $24.72, por ende, se gana $3.10 por hora. En base a la Tabla 36 de horas trabajadas se aplica la Ecuación (5)

$$\boldsymbol{Costo\ desarrollo\ técnico = Valor\ por\ hora * horas\ totales} \quad (5)$$

$$\boldsymbol{Costo\ desarrollo\ técnico = 3.10 * 400 = \$1240}$$

3.8.3 Costo total

El costo total resulta de la suma del costo de desarrollo técnico y el costo de insumos como se muestra en la Ecuación (6).

$$\boldsymbol{Costo\ total = Costo\ de\ desarrollo\ tecnico + Costos\ de\ insumo} \quad (6)$$

El resultado final del costo total es:

$$\boldsymbol{Costo\ desarrollo\ técnico = 1240 + 1811.30 = \$3051.30}$$

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La investigación del impacto de IoT, la arquitectura Edge-Cloud y el Deep Learning en la agricultura evidenció su potencial para mejorar la eficiencia en su producción, realizando una investigación bibliográfica y la revisión de casos de estudio relevantes, lo que permitió identificar cómo estas tecnologías optimizan el monitoreo, control y toma de decisiones en tiempo real. Los resultados obtenidos confirman que la implementación de estas herramientas reduce costos operativos, mejora la productividad y facilita la adopción de prácticas agrícolas más sostenibles.

- El desarrollo del dispositivo IoT permitió la adquisición y control de parámetros clave en el cultivo hidropónico, se realizó el diseño, prototipado y pruebas en campo. Este dispositivo, permitió recopilar datos como pH y conductividad eléctrica, alcanzando una eficiencia del 99.76% y 98.808%, respectivamente, permitiendo a los actuadores una aplicación precisa de nutrientes, demostrando ser una herramienta confiable para el monitoreo y control continuo.

- Se implementó YOLOv8, una arquitectura de redes neuronales convolucionales (CNN), como modelo de inteligencia artificial para la detección de objetos y enfermedades en cultivos agrícolas. Esta elección se fundamentó en su equilibrio entre rendimiento y precisión. El entrenamiento con YOLOv8s demostró ser óptimo debido a su balance entre exactitud y eficiencia en el uso de recursos, alcanzando métricas como un Box(P) de 0.93, un R de 0.85, un mAP50 de 0.926 y un mAP50-90 de 0.485 tras 94 épocas. En las pruebas, los tiempos promedio de procesamiento fueron de 22.9 ms, con tiempos de inferencia ponderada de 725.1 ms y de posprocesamiento por imagen de 25.9 ms, dando como resultado un tiempo total de espera de 773.9 ms. La confiabilidad del sistema alcanzó un 84.92%, validando su funcionalidad como prototipo. Estos resultados permitieron identificar

problemas en los cultivos de manera temprana, facilitando la implementación de medidas preventivas y correctivas que contribuyen a reducir significativamente las pérdidas económicas.

- La implementación de la arquitectura Edge-Cloud posibilitó un monitoreo y control eficientes de las variables existentes en los cultivos agrícolas. En el Edge, los sensores y actuadores procesaron datos en tiempo real, permitiendo acciones inmediatas como el ajuste automático de nutrientes. Mientras tanto, el Cloud permitió a los usuarios visualizar datos en tiempo real y centralizar la gestión de información. Además, se incorporaron medidas de seguridad para restringir el acceso a los datos únicamente a usuarios autorizados, garantizando la privacidad y confiabilidad del sistema.

4.2 Recomendaciones

- Se recomienda realizar una revisión sistemática de literatura en bases de datos científicas, contrastando enfoques teóricos y estudios de caso sobre la aplicación de IoT, Edge-Cloud y Deep Learning en la agricultura. Además, es clave analizar modelos computacionales y medidas de seguridad para garantizar la eficiencia y fiabilidad de estos sistemas.

- Para armar el dispositivo se recomienda manipular correctamente la baquelita, evitando sobrecalentamiento o daños en las pistas al perforar y soldar. Es fundamental utilizar productos químicos adecuados en el grabado del circuito y seguir medidas de seguridad. Además, se debe verificar la continuidad de las pistas antes del ensamblaje y emplear estaño de calidad, evitando excesos de calor en la soldadura. Para proteger el dispositivo, se recomienda el uso de encapsulados impermeables y ubicar sensores lejos de fuentes de humedad. Finalmente, se sugiere realizar mantenimiento preventivo para detectar fallas y asegurar su correcto funcionamiento.

- Para garantizar un rendimiento uniforme en los algoritmos de aprendizaje profundo, se recomienda equilibrar el número de imágenes por clase en el dataset de entrenamiento. Un conjunto de datos desbalanceado puede

segmentar los resultados del modelo y reducir su precisión, especialmente en clases minoritarias. Se sugiere realizar un análisis previo de la distribución de clases y ajustar el dataset en consecuencia, mediante técnicas como la recolección adicional de imágenes o el uso de herramientas de aumento de datos.

- Se recomienda optimizar la distribución de carga entre Edge y Cloud, asegurando que las tareas en tiempo real se procesen localmente para reducir la latencia. Es fundamental priorizar hardware eficiente y protocolos de bajo consumo, además de implementar medidas de seguridad en la transmisión y autenticación de datos. Finalmente, se sugiere realizar pruebas en campo para evaluar estabilidad, conectividad y ajustar la capacidad del sistema según las necesidades del cultivo.

- Se recomienda desarrollar un dashboard simple y directa que facilite la comprensión de los datos presentados. Una visualización clara y amigable es fundamental para apoyar la toma de decisiones rápidas y efectivas por parte de los usuarios. Esto puede incluir gráficos intuitivos, alertas visuales configurables y opciones de navegación sencilla.

- Es recomendable realizar pruebas de resistencia de los componentes electrónicos en condiciones reales de operación para identificar posibles vulnerabilidades. Además, se sugiere implementar medidas adicionales de protección, como sellos resistentes al agua, recubrimientos protectores contra corrosión y ventilación adecuada para evitar el sobrecalentamiento en ambientes húmedos.

REFERENCIAS BIBLIOGRÁFICAS

[1] FAO, «OECD-FAO Agricultural Outlook 2023-2032,» FAO, Paris, 2023.

[2] FAO, «Versión resumida de El estado mundial de la agricultura y la alimentación 2022,» FAO, Roma, 2021.

[3] H. Martínez, E. Ramírez y O. Echenique, «Digitalización y cambio tecnológico en las mipymes agrícolas y agroindustriales en América Latina,» FAO, 2021.

[4] O. Maritza, «Encuesta de Superficie y Producción Agropecuaria Continua (ESPAC),» INEC, 2024.

[5] CEPAL, FAO, IICA, «PERSPECTIVAS DE LA AGRICULTURA Y DEL DESARROLLO RURAL EN LAS AMÉRICAS,» IICA, San José, 2021.

[6] SIPA, «Encuesta de Superficie y Producción Agropecuaria Continua,» SIPA, 2021.

[7] SIPA, «INFORMACIÓN PRODUCTIVA TERRITORIAL,» SIPA, 2023.

[8] S. Ana, V. Tatiana, M. Fernando y F. Carolina, «Sector agrícola Ecuador,» Observatorio económico y Social de Tungurahua.

[9] Gobierno provincial de Tungurahua, «ESTUDIO DEFINITIVO TUNGA SAN PEDRO,» HGPT - kfW, San Pedro, 2012.

[10] J. CRIOLLO, «SISTEMA DE MONITOREO DE CONDICIONES ÓPTIMAS DE CRECIMIENTO DE CULTIVOS EN SISTEMAS MICROFARMS POR MEDIO DE VISIÓN POR COMPUTADOR Y ALGORITMOS DE APRENDIZAJE DE MÁQUINA,» UNIVERSIDAD TÉCNICA DEL NORTE, Ibarra, 2023.

[11] B. Montaño, E. González, S. Jiménez, M. Mingo y A. Carrión, «Diseño y despliegue de un sistema de monitoreo basado en IOT para cultivos hidropónicos,» *Ingenius,* vol. 30, nº 1, pp. 9-18, 2023.

[12] D. Bustamante, «Cultivo aeropónico de precisión para Solanum Tuberosum, supervisado mediante IoT en Ambato (Ecuador),» Reunir, Ambato, 2022.

[13] G. Chanchí, E. Gabriel, A. Ospina, A. Manuel y M. Saba, «Sistema IoT para el monitoreo de variables climatológicas en cultivos de agricultura urbana,» *Revista científica,* vol. 44, nº 2, pp. 257-271, 2022.

[14] S. Premkumar, «IoT-enabled edge computing model for smart irrigation system,» *Journal of Intelligent Systems,* vol. 31, nº 1, pp. 632-650, 2022.

[15] L. Kyuchang, S. Bhagya y H. Kijun, «Deep Learning Entrusted to Fog Nodes (DLEFN) Based Smart Agriculture,» *MDPI,* vol. 10, nº 4, pp. 1544-1544, 2020.

[16] B. Rodolfo, C. Evandro, B. Stanley y R. Álvaro, Agricultura de Precisión: Integrando conocimientos para una agricultura moderna y sustentable, Montevideo: INIA, 2006.

[17] G. Emiliano y F. Fernando, Agricultura de Precisión, Tecnología Agropecuaria.

[18] E. García y F. Flego, «Agricultura de precisión,» TECNOLOGIA AGROPECUARIA.

[19] M. Zárate, MANUAL DE HIDROPONIA, México: UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO, 2014.

[20] J. Ayres, R. Grasso y C. Berrueta, CONTROL DE LA SOLUCIÓN NUTRITIVA HIDROPÓNICA (NFT) BASADO EN EL PH Y LA CONDUCTIVIDAD: luces y sombras, Sistema Vegetal Intensivo - INIA, 2022.

[21] W. Almodóvar, «Enfermedades de la Lechuga,» Clínica al Día, 2001.

[22] S. Fountas, B. Espejo, A. Kasimati, M. Mylonas y N. Darra, «The Future of Digital Agriculture: Technologies and Opportunities,» *IEEE,* vol. 22, nº 1, pp. 24-28, 2020.

[23] A. Pawar y S. Deosarkar, «IoT-based smart agriculture: an exhaustive study,» *Springer link,* vol. 29, pp. 2457-2470, 2023.

[24] M. Ayaz, A. Uddin, Z. Sharif y A. Mansour, «Internet-of-Things (IoT)-Based Smart Agriculture: Toward Making the Fields Talk,» *ResearchGate ,* 2019.

[25] S. Wolfert, L. Ge, C. Verdouw y M.-J. Bogaardt, «Big Data in Smart Farming – A review,» *Agricultural Systems,* vol. 153, pp. 69-80, 2017.

[26] «Dfrobot,» [En línea]. Available: https://www.dfrobot.com/product-1123.html. [Último acceso: 26 Diciembre 2024].

[27] «AtlasScientific,» Envioronmental Robotics, [En línea]. Available: https://atlas-scientific.com/embedded-solutions/ezo-conductivity-circuit/. [Último acceso: 26 Diciembre 2024].

[28] «Bluelab Truncheon Nutrient Meter,» Bluelab, [En línea]. Available: https://bluelab.com/usa/bluelab-truncheon-nutrient-meter. [Último acceso: 26 Diciembre 2024].

[29] «Dfrobot,» Dfrobot, [En línea]. Available: https://www.dfrobot.com/product-1782.html. [Último acceso: 26 Diciembre 2024].

[30] «AtlasScientific,» Environmental Robotics, [En línea]. Available: https://atlas-scientific.com/embedded-solutions/ezo-ph-circuit/. [Último acceso: 26 Diciembre 2024].

[31] «DIY MORE,» DIY MORE, [En línea]. Available: https://www.diymore.cc/products/diymore-liquid-ph-value-detection-detect-sensor-module-monitoring-control-for-arduino-m. [Último acceso: 26 Diciembre 2024].

[32] stepIM, «Integrated Closed Loop Stepper Motor,» SERVOTRONIX.

[33] «Uelectronics,» UNIT ELECTRONICS, [En línea]. Available: https://uelectronics.com/producto/driver-tb6560-controlador-motor-a-pasos/. [Último acceso: 26 Diciembre 2024].

[34] «Uelectronics,» UNIT ELECTRONICS, [En línea]. Available: https://uelectronics.com/producto/tb6600-driver-para-motor-a-pasos-4a-nema/. [Último acceso: 26 Diciembre 2024].

[35] «Naylampmechatronics,» NAYLAMP MECHATRONICS, [En línea]. Available: https://naylampmechatronics.com/driver-pap-stepper/126-driver-pap-drv8825.html. [Último acceso: 26 Diciembre 2024].

[36] «Espressif,» Espressif, [En línea]. Available: https://www.espressif.com/en/products/socs/esp32. [Último acceso: 26 Diciembre 2024].

[37] «Espressif,» Espressif, [En línea]. Available: https://www.espressif.com/en/products/socs/esp8266. [Último acceso: 26 Diciembre 2024].

[38] «STM32 32-bit Arm Cortex MCUs,» ST, [En línea]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html. [Último acceso: 26 Diciembre 2024].

[39] «Raspberry pi,» Raspberry pi, [En línea]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. [Último acceso: 26 Diciembre 2024].

[40] «Raspberry pi,» Raspberry pi, [En línea]. Available: https://www.raspberrypi.com/products/raspberry-pi-5/. [Último acceso: 26 Diciembre 2024].

[41] «Nvidia,» Nvidia, [En línea]. Available: https://www.nvidia.com/es-la/autonomous-machines/embedded-systems/jetson-nano/product-development/. [Último acceso: 26 Diciembre 2024].

[42] «Uelectronics,» UNIT ELECTRONICS, [En línea]. Available: https://uelectronics.com/producto/camara-raspberry-pi-rev-1-3-5mp/. [Último acceso: 26 Diciembre 2024].

[43] «Raspberry pi,» Raspberry pi, [En línea]. Available: https://www.raspberrypi.com/products/camera-module-v2/. [Último acceso: 26 Diciembre 2024].

[44] «Raspberry pi,» Raspberry pi, [En línea]. Available: https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/. [Último acceso: 26 Diciembre 2024].

[45] «Arduino,» Arduino, [En línea]. Available: https://forum.arduino.cc/t/how-do-i-use-c-to-code-the-arduino-boards/493409. [Último acceso: 26 Diciembre 2024].

[46] «Micropython,» Micropython, [En línea]. Available: https://micropython.org/. [Último acceso: 26 Diciembre 2024].

[47] «Lua,» lua, [En línea]. Available: https://www.lua.org/. [Último acceso: 26 Diciembre 2024].

[48] Kostiantyn Oliynyk, «webbylab,» 27 Septiembre 2024. [En línea]. Available: https://webbylab.com/blog/mqtt-vs-other-iot-messaging-protocols/. [Último acceso: 20 Noviembre 2024].

[49] G. Shaji y G. Hovan, «Edge Computing and the Future of Cloud Computing: A Survey of Industry Perspectives and Predictions,» *ResearchGate*, pp. 119-144, 2023.

[50] W. Shi y S. Dustdar, «The Promise of Edge Computing,» *Computer,* vol. 49, nº 5, pp. 78-81, 2016.

[51] F. Bonomi, R. Milito, J. Zhu y S. Addepalli, «Fog computing and its role in the internet of things,» *Association for Computing Machinery,* pp. 13-16, 2012.

[52] M. Satyanarayanan, «The Emergence of Edge Computing,» *Computer,* vol. 50, nº 1, pp. 30-39, 2017.

[53] Q. Zhang y L. Cheng, «Cloud computing: state-of-the-art and research challenges,» *Journal of Internet Services and Applications,* nº 1, pp. 7-18, 2010.

[54] Hitachi, «Hitachivantara.com,» 2021. [En línea]. Available: https://www.hitachivantara.com/es-latam/insights/faq/what-is-an-edge-cloud.html. [Último acceso: 15 Noviembre 2024].

[55] Y. Kalyani y R. Collier, « A systematic survey on the role of cloud, fog, and edge computing combination in smart agriculture,» *Sensors,* vol. 21, 2021.

[56] Y. Kalyani y R. Collier, «Multi-agent based cloud-fog-edge computing and digital twin for smart agriculture,» *Springer,* vol. 1089, 2023.

[57] V. Karagiannis y S. Schulte, «Distributed algorithms based on proximity for self-organizing fog computing systems,» *Pervasive and Mobile Computing,* vol. 71, 2021.

[58] K. Wazir, A. Ejaz, H. Saqib, Y. Ibrar y A. Arif, «Edge computing: A survey,» *Future Generation Computer Systems,* vol. 97, pp. 219-235, 2019.

[59] J. Du, L. Zhao, J. Feng y X. Chu, «Computation Offloading and Resource Allocation in Mixed Fog/Cloud Computing Systems With Min-Max Fairness Guarantee,» *IEEE*, vol. 66, nº 4, pp. 1594-1608, 2018.

[60] B. Varghese, N. Wang, S. Barbhuiya y P. Kilpatrick, «Challenges and Opportunities in Edge Computing,» *IEEE*, pp. 20-26, 2016.

[61] «KEEPCODING,» Tech School, [En línea]. Available: https://keepcoding.io/blog/bases-de-datos-mas-comunes/. [Último acceso: 26 Diciembre 2024].

[62] L. Rouhiainen, INTELIGENCIA ARTIFICIAL, España: Alienta editorial, 2018.

[63] K. Andreas y P. Francesc, «Deep learning in agriculture: A survey,» *Computers and Electronics in Agriculture,* vol. 147, pp. 70-90, 2018.

[64] A. Marwan, «A Survey on Deep Learning and Its Impact on Agriculture: Challenges and Opportunities.,» *Agriculture,* vol. 13, nº 3, 2023.

[65] J. Ubbens y I. Stavness, «Deep Plant Phenomics: A Deep Learning Platform for Complex Plant Phenotyping Tasks,» *Frontiers in Plant Science,* vol. 8, 2017.

[66] Z. Gao, Z. Luo, W. Zhang, Z. Lv y Y. Xu, «Deep Learning Application in Plant Stress Imaging: A Review,» *AgriEngineering,* vol. 2, nº 3, 2020.

[67] S. P. Mohanty, D. P. Hughes y M. Salathé, «Using deep learning for image-based plant disease detection,» *Frontiers in plant science,* vol. 7, 2016.

[68] G. Ignacio, «Metodología para Machine Learning (I): CRISP-DM | Ignacio G.R. Gavilán,» Ignacio G.R. Gavilán, 28 Junio 2021. [En línea]. Available: https://ignaciogavilan.com/metodologia-para-machine-learning-i-crisp-dm/. [Último acceso: 2024 Noviembre 20].

[69] N. Hotz, «Agile AI - Data Science PM,» Data Science PM, 31 Agosto 2023. [En línea]. Available: https://www.datascience-pm.com/agile-ai/. [Último acceso: 20 Noviembre 2024].

[70] C. Studio, «¿Qué es un pipeline de aprendizaje automático?,» Purestorage.com, 21 Marzo 2024. [En línea]. Available: https://www.purestorage.com/es/knowledge/what-is-machine-learning-pipeline.html. [Último acceso: 20 Noviembre 2024].

[71] «Los 9 mejores lenguajes de programación para la Inteligencia Artificia,» Wingsoft, [En línea]. Available: https://www.wingsoft.com/blog/mejores-lenguajes-IA. [Último acceso: 26 Diciembre 2024].

[72] I. Sarker, «Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions,» *SN COMPUT*, vol. 2, 2021.

[73] «Faster R-CNN vs YOLO vs SSD — Object Detection Algorithms,» Medium, [En línea]. Available: https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc. [Último acceso: 26 Diciembre 2024].

[74] H. Muhammad, «YOLOv1 to v8: Unveiling Each Variant—A Comprehensive Review of YOLO,» *IEEE*, vol. 12, pp. 42816-42833, 2024.

[75] A. Schcolnik-Elias, S. Martinez, J. Luna y I. Castro, «Detección de armas tipo pistola mediante el uso de redes convolucionales con una arquitectura tipo YOLO y estereoscopía,» *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI,* vol. 11, pp. 196-204, 2023.

[76] Ministerio de Trabajo, «SALARIOS MÍNIMOS SECTORIALES Y TARIFAS 2024,» Quito, 2024.

ANEXOS

Anexo A. Certificado.

Ambato 09/01/2025

Ingeniero

Luis Morales

Presidente

Unidad de Titulación

Carrera de Telecomunicaciones

Facultad de Ingeniería en Sistemas, Electrónica e Industrial

Yo Jessica Arévalo Castro en calidad de Ingeniera en Agronomía, otorgo mi Aval al proyecto de investigación titulado "SISTEMA DE MONITOREO Y CONTROL IOT PARA CULTIVOS AGRÍCOLAS BASADO EN LA ARQUITECTURA EDGE CLOUD Y DEEP LEARNING" desarrollado por el señor Byron Alexander Amán Caiza, portador de la cedula 1850921451, estudiante de la carrera de Telecomunicaciones de la facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, quien ha trabajado bajo mi supervisión y ha demostrado un alto nivel de competencia técnica y científica. Por lo tanto, considero que el trabajo presentado es original y de alta calidad, cumpliendo con los estándares de la agronomía necesarios para el proyecto de titulación.

Atentamente,



.....

Ing. Jessica María Arévalo Castro

1804568119

0997408470

arevalojessica66@gmail.com

Figura A1. Certificado de validación.

Anexo B. Esp32.

En la Figura B1 se muestra las especificaciones de la Esp32 para el uso de pines para el proyecto.

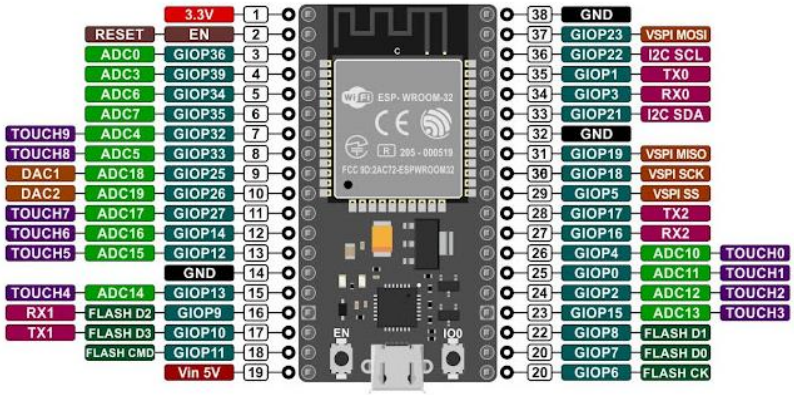
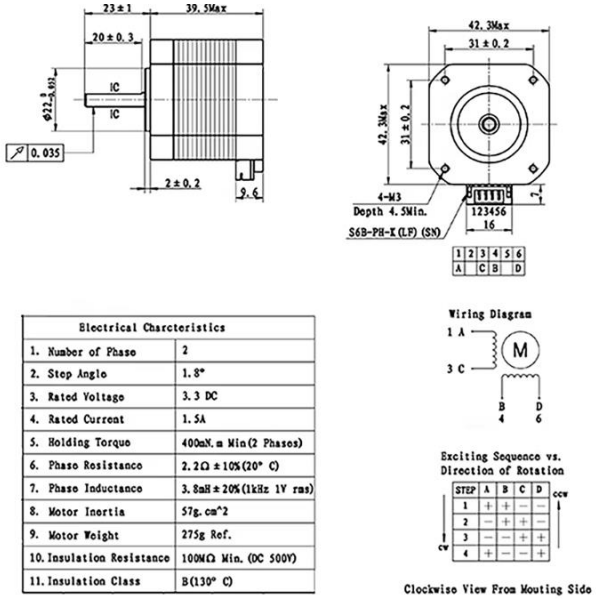


Figura B1. Pinout esp32.

Anexo C. Motor a pasos.

La Figura C1 muestra las especificaciones del motor NEMA utilizado, junto con su conexión y los parámetros de su funcionamiento.



Electrical Charcteristics	
1. Number of Phase	2
2. Step Angle	1.8°
3. Rated Voltage	3.3 DC
4. Rated Current	1.5A
5. Holding Torque	400mN.m Min(2 Phases)
6. Phase Resistance	2.2Ω ±10%(20° C)
7. Phase Inductance	3.8mH ±20%(1kHz 1V rms)
8. Motor Inertia	57g.cm^2
9. Motor Weight	275g Ref.
10. Insulation Resistance	100MΩ Min. (DC 500V)
11. Insulation Class	B(130° C)

Figura C1. Especificaciones motor 47DH4027-01-A.

Anexo D. Driver para el motor.

La Figura D1 presenta las especificaciones del driver tb6560, el encargado para el funcionamiento de la bomba dosificadora.

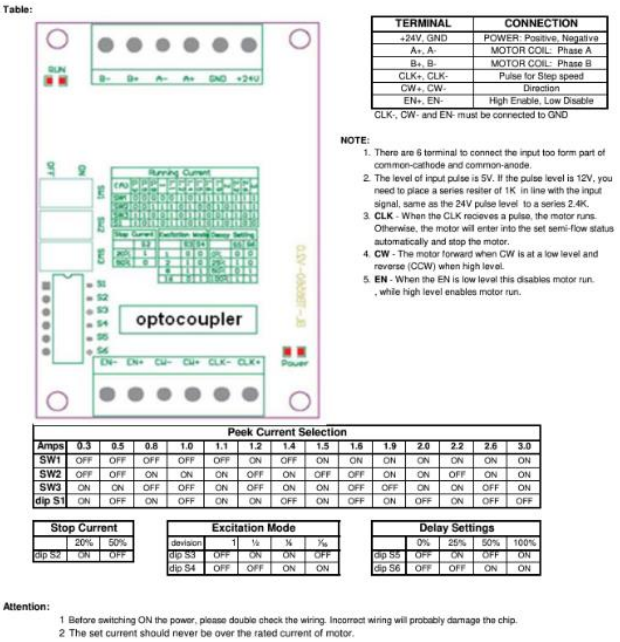


Table:

TERMINAL	CONNECTION
+24V, GND	POWER: Positive, Negative
A+, A-	MOTOR COIL: Phase A
B+, B-	MOTOR COIL: Phase B
CLK+, CLK-	Pulse for Step speed
CW+, CW-	Direction
EN+, EN-	High Enable, Low Disable

CLK-, CW- and EN- must be connected to GND

NOTE:
1. There are 6 terminal to connect the input too form part of common-cathode and common-anode.
2. The level of input pulse is 5V. If the pulse level is 12V, you need to place a series resiter of 1K in line with the input signal, same as the 24V pulse level to a series 2.4K.
3. **CLK** - When the CLK recieves a pulse, the motor runs. Otherwise, the motor will enter into the set semi-flow status automatically and stop the motor.
4. **CW** - The motor forward when CW is at a low level and reverse (CCW) when high level.
5. **EN** - When the EN is low level this disables motor run. , while high level enables motor run.

optocoupler

Peek Current Selection													
Amps	0.3	0.5	0.8	1.0	1.1	1.2	1.4	1.5	1.6	1.9	2.0	2.2	2.6	3.0
SW1	OFF	OFF	OFF	OFF	OFF	ON	OFF	ON	ON	ON	ON	ON	ON	ON
SW2	OFF	OFF	ON	ON	ON	OFF	ON	OFF	OFF	ON	ON	OFF	ON	ON
SW3	ON	ON	OFF	OFF	ON	OFF	ON	ON	OFF	OFF	ON	ON	OFF	ON
dip S1	ON	OFF	ON	OFF	ON	ON	OFF	ON	OFF	ON	OFF	ON	OFF	OFF

Stop Current		
	20%	50%
dip S2	ON	OFF

Excitation Mode				
devision	1	½	¼	1/8
dip S3	OFF	ON	ON	OFF
dip S4	OFF	OFF	ON	ON

Delay Settings				
	0%	25%	50%	100%
dip S5	OFF	ON	OFF	ON
dip S6	OFF	OFF	ON	ON

Attention:
1 Before switching ON the power, please double check the wiring. Incorrect wiring will probably damage the chip.
2 The set current should never be over the rated current of motor.

Figura D1. Especificaciones driver TB6560.

Anexo E. Sensor para ph.

La Figura E1 presenta las conexiones del sensor ph4502C.

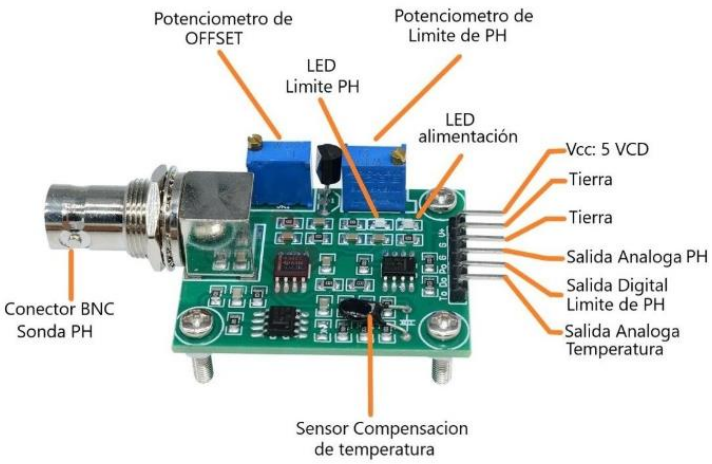
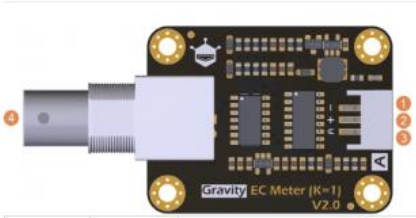


Figura E1. Sensor ph 4502C.

Anexo F. Sensor de conductividad eléctrica.

La Figura F1 presenta las conexiones del sensor EC Sensor V2.0.



Num	Label	Description
1	-	Power GND(0V)
2	+	Power VCC(3.0~5.0V)
3	A	Analog Signal Output(0~3.4V)
4	BNC	Probe Connector

Figura F1. Sensor EC Sensor V2.0.

Anexo G. Raspberry pi 4.

La Figura G1 presenta las especificaciones de la Raspberry pi 4.



Características:

- Raspberry Pi 4 Model B
- Procesador: Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- Memoria RAM: 4GB LPDDR4-2400 SDRAM
- Conectividad Wifi: 2.4 GHz y 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE
- Conectividad ethernet: True Gigabit Ethernet
- 2 puertos USB 3.0
- 2 puertos USB 2.0
- GPIO 40-pinescompatible con versiones anteriores
- 2 puertos micro HDMI con soporte 4K 60fps
- Puerto MIPI DSI/CSI para cámara y pantalla
- Salida jack para audio estéreo y video compuesto
- Zócalo Micro SD para el sistema operativo
- Requiere alimentador de 5.1V, 3A con conector USB Type C o mediante GPIO
- PoE disponible a través de un HAT opcional

Figura G1. Raspberry pi 4.

Anexo H. Pruebas del funcionamiento.

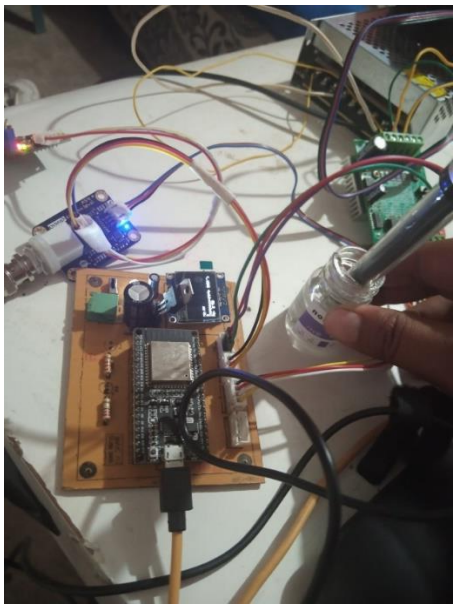


Figura H1. Pruebas de funcionamiento de los sensores.

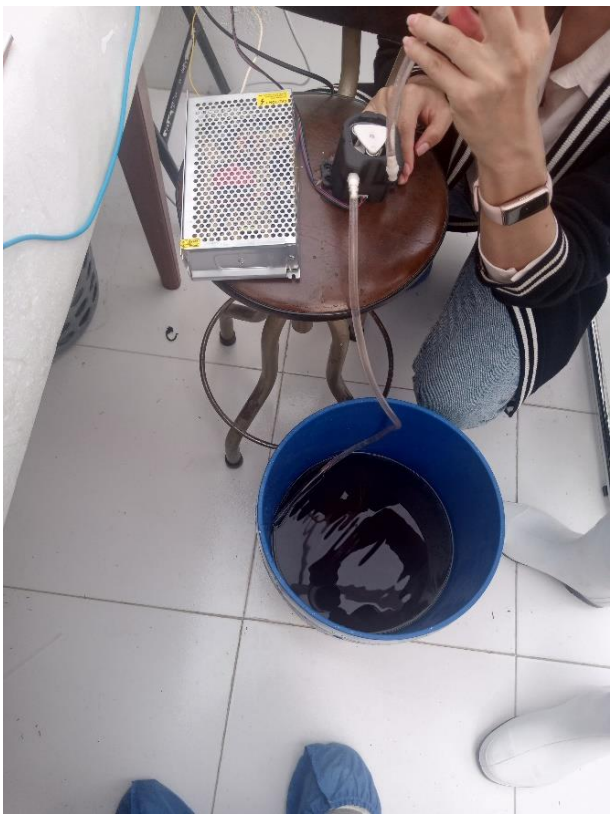


Figura H2. Pruebas de funcionamiento de la bomba dosificadora.



Figura H3. Pruebas de funcionamiento en general.

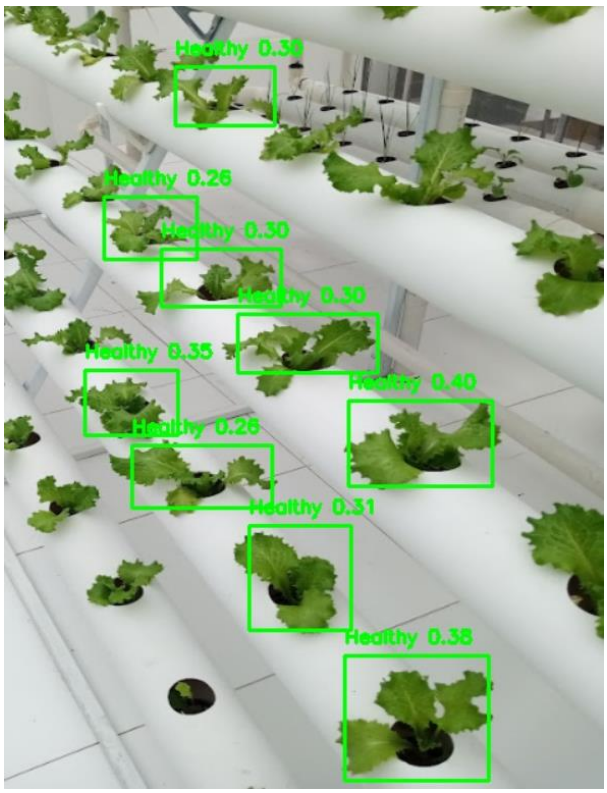


Figura H4. Pruebas de funcionamiento del reconocimiento de enfermedades.

Anexo I. Código de programación del dispositivo.

```
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "nombre de la red";
const char* password = "contraseña";

const char* mqtt_server = "nombre del broken";

// ----- DEFINICIONES DE PINES Y CONSTANTES -----
#define PH_PIN 34
#define EC_PIN 35
#define VOLTAGE_REF 3.3
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Definición de pines para motores paso a paso
const int stepPin = 32;
const int dirPin = 33;
const int enPin = 25;
const int stepPin2 = 26;
const int dirPin2 = 27;
const int enPin2 = 14;

// estado del motor
bool motorActivo = false;
int estado_motor=0;
const int muestreo1=16;
const int muestreo2=4;

// ----- ESTRUCTURAS DE CALIBRACIÓN -----
struct PHCalibration {
   const int bits_pH4 = 600;
   const int bits_pH7 = 1200;
   const int bits_pH9 = 1900;
   float slope_acid, slope_alkaline;
   float intercept_acid, intercept_alkaline;
   float phEMA = 0;
   const float alpha_pH = 0.05;
};

struct ECCalibration {
   const float analog_very_low = 0;
   const float ec_very_low = 20.0;
   const float analog_low = 8;
   const float ec_low = 1413.0;
   const float analog_high = 2050;
   const float ec_high = 13000.0;
   float slope_low, intercept_low;
   float slope_high, intercept_high;
   float ecEMA = 0;
   const float alpha_ec = 0.05;
   const float temperature = 25.0;
};
```

```
// ----- VARIABLES GLOBALES -----
PHCalibration phCal;
ECCalibration ecCal;
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;

// ----- PROTOTIPOS DE FUNCIONES -----
void setup_wifi();
void callback(char* topic, byte* payload, unsigned int length);
void reconnect();
void initializeDisplay();
void initializePHCalibration();
void initializeECCalibration();
void initializeMotor();
float readAndCalculatePH();
float readAndCalculateEC();
void updateDisplay(float pH, float ec);
bool mostrarPh = true;//alternar las visualizaciones
void moveMotor(int steps);
void publishSensorData(float pH, float ec);

// ----- FUNCIÓN SETUP -----
void setup() {
  Serial.begin(115200);

  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  initializeDisplay();
  initializePHCalibration();
  initializeECCalibration();
  initializeMotor();

  Serial.println("Medición pH y Conductividad Eléctrica");
  display.setCursor(0, 0);
  display.println("Medicion pH y CE");
  display.display();
  delay(2000);

}

// ----- FUNCIÓN LOOP -----
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;

    float pHValue = readAndCalculatePH();
    float ecValue = readAndCalculateEC();
```

```
        // Control de motores basado en lecturas
        if (motorActivo) {
           digitalWrite(dirPin, LOW);
           moveMotor(200);
           estado_motor=1;
           digitalWrite(muestreo1,HIGH);
           digitalWrite(muestreo2,LOW);
        }else{
          estado_motor=0;
          digitalWrite(muestreo1,LOW);
          digitalWrite(muestreo2,HIGH);
          }

        // Actualizar pantalla y publicar datos
        updateDisplay(pHValue, ecValue, estado_motor );
        publishSensorData(pHValue, ecValue, estado_motor);
    }
}

// ----- IMPLEMENTACIÓN DE FUNCIONES WIFI Y MQTT -----
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Mensaje recibido [");
    Serial.print(topic);
    Serial.print("]: ");
    String message = "";
    for (int i = 0; i < length; i++) {
        message += (char)payload[i];
    }
    Serial.println(message);

    // Controlar motor por MQTT
    if (String(topic) == "nombre del topic") {
        if (message == "ON") {
            motorActivo = true;
            Serial.println("Motor activado por MQTT");
        } else if (message == "OFF") {
            motorActivo = false;
            Serial.println("Motor desactivado por MQTT");
        }
    }
}
```

```
void reconnect() {
   while (!client.connected()) {
      Serial.print("Attempting MQTT connection...");
      String clientId = "ESP32Client-";
      clientId += String(random(0xffff), HEX);
      if (client.connect(clientId.c_str())) {
         Serial.println("Conectado a MQTT");
         client.publish("nombre del topic", "ESP32 Conectada");
         client.subscribe("nombre del topic");
         client.subscribe("nombre del topic"); // NUEVO: Suscribirse al tópico para control del motor
      } else {
         Serial.print("Error, rc=");
         Serial.print(client.state());
         Serial.println(" intentando de nuevo en 5 segundos");
         delay(5000);
      }
   }
}

void publishSensorData(float pH, float ec, int motor) {
   String pHString = String(pH, 2);
   String ecString = String(ec, 2);
   String nutString = String(motor);

   String topic = "nombre del topic";
   String payload = "{\"ph\":" + pHString + ",\"ce\":" + ecString + ",\"nut\":" + nutString + "}";
   client.publish(topic.c_str(), payload.c_str());

   Serial.print("Published - pH: ");
   Serial.print(pHString);
   Serial.print(" EC: ");
   Serial.println(ecString);
   Serial.print(" NUT: ");
   Serial.println(nutString);
}
// ----- IMPLEMENTACIÓN DE FUNCIONES -----

// Inicializa la pantalla OLED
void initializeDisplay() {
   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
      Serial.println(F("Fallo al iniciar OLED"));
      for (;;);
   }
   display.clearDisplay();
   display.setTextSize(3);
   display.setTextColor(SSD1306_WHITE);
}

// Calcula las pendientes y intersecciones para la calibración de pH
void initializePHCalibration() {
   // Calcular pendientes y offsets para el pH usando bits
   phCal.slope_acid = (7.0 - 4.0) / (phCal.bits_pH7 - phCal.bits_pH4);
   phCal.intercept_acid = 7.0 - (phCal.slope_acid * phCal.bits_pH7);

   phCal.slope_alkaline = (9.0 - 7.0) / (phCal.bits_pH9 - phCal.bits_pH7);
   phCal.intercept_alkaline = 9.0 - (phCal.slope_alkaline * phCal.bits_pH9);

   phCal.phEMA = analogRead(PH_PIN);  // Inicializar EMA con la primera lectura
}
```

```
// Calcula las pendientes y intersecciones para la calibración de conductividad
void initializeECCalibration() {
   // Calcular pendientes e intersecciones para las dos regresiones lineales
   ecCal.slope_low = (ecCal.ec_low - ecCal.ec_very_low) / (ecCal.analog_low - ecCal.analog_very_low);
   ecCal.intercept_low = 20.0;  // Para valor analógico 0, conductividad es 20 µS/cm

   ecCal.slope_high = (ecCal.ec_high - ecCal.ec_low) / (ecCal.analog_high - ecCal.analog_low);
   ecCal.intercept_high = ecCal.ec_low - (ecCal.slope_high * ecCal.analog_low);

   ecCal.ecEMA = analogRead(EC_PIN);  // Inicializar EMA con la primera lectura
}

// Configura los pines del motor paso a paso
void initializeMotor() {
   pinMode(stepPin, OUTPUT);
   pinMode(dirPin, OUTPUT);
   pinMode(enPin, OUTPUT);
   digitalWrite(enPin, LOW);  // LOW activa el driver
   pinMode(stepPin2, OUTPUT);
   pinMode(dirPin2, OUTPUT);
   pinMode(enPin2, OUTPUT);
   digitalWrite(enPin2, LOW);  // LOW activa el driver
   pinMode(muestreo1, OUTPUT);
   pinMode(muestreo2, OUTPUT);
}

// Lee y calcula el valor de pH
float readAndCalculatePH() {
   int phAnalogValue = analogRead(PH_PIN);
   phAnalogValue = map(phAnalogValue, 0, 4096, 4096, 0);  // Invertir la señal

   // Aplicar filtro EMA para suavizar el valor
   phCal.phEMA = phCal.alpha_pH * phAnalogValue + (1 - phCal.alpha_pH) * phCal.phEMA;

   // Imprimir valor analógico en consola
   Serial.print("Valor analógico pH: ");
   Serial.println(phAnalogValue);

   // Calcular pH según el rango
   if (phCal.phEMA <= phCal.bits_pH7) {
      return phCal.slope_acid * phCal.phEMA + phCal.intercept_acid - 0.10;
   } else {
      return phCal.slope_alkaline * phCal.phEMA + phCal.intercept_alkaline + 0.20;
   }
}

// Lee y calcula el valor de conductividad eléctrica
float readAndCalculateEC() {
   int ecAnalogValue = analogRead(EC_PIN);

   // Aplicar filtro EMA para suavizar el valor
   ecCal.ecEMA = ecCal.alpha_ec * ecAnalogValue + (1 - ecCal.alpha_ec) * ecCal.ecEMA;

   // Imprimir valor analógico en consola
   Serial.print("Valor analógico Conductividad: ");
   Serial.println(ecAnalogValue);
```

```
    // Calcular conductividad según el rango
    float ecValue;
    if (ecCal.ecEMA == 0) {
        ecValue = 20;  // Valor mínimo de conductividad
    } else if (ecCal.ecEMA <= ecCal.analog_low) {
        ecValue = ecCal.slope_low * ecCal.ecEMA + ecCal.intercept_low + 400;
    } else {
        ecValue = ecCal.slope_high * ecCal.ecEMA + ecCal.intercept_high - 100;
    }

    // Aplicar compensación de temperatura
    return ecValue / (1.0 + 0.0185 * (ecCal.temperature - 25.0));
}

// Actualiza la pantalla OLED con los valores medidos
void updateDisplay(float pH, float ec, int motor) {
    // Limpiar la pantalla antes de dibujar
    display.clearDisplay();
    display.setCursor(0, 0);

    if (mostrarPh) {
        // Mostrar pH y estado del motor
        display.print("pH:");
        display.println(pH, 2); // Mostrar pH con 2 decimales
        //display.println(" ");   // Espacio
        display.print("Nut:");
        display.println(motor == 1 ? "ON" : "OFF"); // Mostrar ON/OFF
    } else {
        // Mostrar CE y estado del motor
        display.print("CE:");
        display.print(ec, 2);   // Mostrar CE con 2 decimales
        display.println(" uS"); // Añadir unidades
        //display.println(" ");   // Espacio
        display.print("Nut:");
        display.println(motor == 1 ? "ON" : "OFF"); // Mostrar ON/OFF
    }

    // Mostrar datos en la pantalla
    display.display();

    // Alternar la variable para el próximo ciclo
    mostrarPh = !mostrarPh;
}

// Mueve el motor paso a paso un número específico de pasos
void moveMotor(int steps) {
    for (int i = 0; i < steps; i++) {
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(2000);  // Ajusta para controlar la velocidad
        digitalWrite(stepPin, LOW);
        delayMicroseconds(2000);  // Ajusta para controlar la velocidad
    }
}

// Imprime los datos en el puerto serie
void printSerialData(float pH, float ec) {
    Serial.print("Señal analógica (EMA): ");
    Serial.print(ecCal.ecEMA);
    Serial.print(" | Conductividad Promedio: ");
```

```
    Serial.print(ec);
    Serial.println(" uS/cm");

    Serial.print("pH EMA: ");
    Serial.print(phCal.phEMA);
    Serial.print(" | pH Calculado: ");
    Serial.println(pH);
}
```

Anexo J. Código de programación en Python.

```
import cv2
from ultralytics import YOLO
import numpy as np
import base64
import paho.mqtt.client as mqtt
import json
import time
import os

# Configuración de MQTT
broker = "nombre del broken"  # Cambia por la IP o el nombre de tu broker MQTT
port = 1883
topic_base_image = "nombre del topic"
topic_processed_image = "nombre del topic"
topic_detections = "nombre del topic"

# Inicializar cliente MQTT
mqtt_client = mqtt.Client()
mqtt_client.connect(broker, port, 60)

# Cargar el modelo YOLO entrenado
model = YOLO('weights/best.pt')
model.conf = 0.01  # Captura predicciones con confianza muy baja

# ---------- FUNCIONES AUXILIARES ---------- #
def capture_image(output_path):
    """Captura una imagen usando raspistill."""
    os.system(f"raspistill -o {output_path} -w 800 -h 800 -t 1000")
    print(f"Imagen capturada y guardada en {output_path}")

def image_to_base64(image, max_width=250):
    """Redimensiona la imagen si es necesario y la convierte en formato base64."""
    h, w = image.shape[:2]
    if w > max_width:
        # Calcular el factor de escala
        scale = max_width / w
        new_width = int(w * scale)
        new_height = int(h * scale)
        resized_image = cv2.resize(image, (new_width, new_height))
    else:
        resized_image = image

    _, buffer = cv2.imencode('.jpg', resized_image)
```

```
    return base64.b64encode(buffer).decode('utf-8')

def enviar_a_mqtt(base_image, processed_image, detecciones):
    """Envía las imágenes y detecciones a Node-RED mediante MQTT."""
    # Publicar imagen base en formato base64
    mqtt_client.publish(topic_base_image, base_image)
    print("Imagen base enviada a Node-RED.")

    # Publicar imagen procesada en formato base64
    mqtt_client.publish(topic_processed_image, processed_image)
    print("Imagen procesada enviada a Node-RED.")

    # Agrupar todas las detecciones en un solo mensaje
    detecciones_json = [
        {
            "etiqueta": d[5],
            "confianza": d[4],
            "coordenadas": {"x1": d[0], "y1": d[1], "x2": d[2], "y2": d[3]}
        }
        for d in detecciones
    ]
    mqtt_client.publish(topic_detections, json.dumps(detecciones_json))
    print(f"Publicado: {detecciones_json}")

# ---------- DETECCIÓN PRINCIPAL ---------- #
# Capturar imagen desde la cámara
image_path = "captura.jpg"
capture_image(image_path)

# Leer la imagen capturada
image = cv2.imread(image_path)
if image is None:
    print(f"Error: No se pudo cargar la imagen desde {image_path}. Verifica la cámara y el archivo.")
    exit()

# Preprocesar la imagen con ajustes de contraste y nitidez
alpha = 1.2  # Factor de contraste
beta = 10    # Brillo
image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
image_hsv[:, :, 2] = cv2.equalizeHist(image_hsv[:, :, 2])  # Ecualización en el canal V
image_preprocessed = cv2.cvtColor(image_hsv, cv2.COLOR_HSV2BGR)

# Aplicar un filtro de nitidez
kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])  # Filtro de nitidez
image_preprocessed = cv2.filter2D(image_preprocessed, -1, kernel)
image_preprocessed = cv2.convertScaleAbs(image_preprocessed, alpha=alpha, beta=beta)

# Ajustar la resolución de entrada
results = model(image_preprocessed, imgsz=800)

# Filtrar detecciones por área y confianza
min_area = 500  # Área mínima permitida
max_area = 100000  # Área máxima permitida
detecciones = []

for box in results[0].boxes:
    x1, y1, x2, y2 = map(int, box.xyxy[0].tolist())
    area = (x2 - x1) * (y2 - y1)
    conf = round(float(box.conf.item()), 2)
```

```
    label = results[0].names[int(box.cls)]
    if conf > 0.3 and min_area < area < max_area:
        detecciones.append((x1, y1, x2, y2, conf, label))

# Dibujar resultados en la imagen original
for (x1, y1, x2, y2, conf, label) in detecciones:
    print(f"Etiqueta: {label}, Confianza: {conf:.2f}, Coordenadas: ({x1}, {y1}), ({x2}, {y2})")
    cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.putText(image, f"{label} {conf:.2f}", (x1, y1 - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

# Convertir imágenes a base64
base_image_base64 = image_to_base64(image)
processed_image_base64 = image_to_base64(image)

# Enviar datos a Node-RED
enviar_a_mqtt(base_image_base64, processed_image_base64, detecciones)

# Guardar la imagen procesada
cv2.imwrite("imagen_procesada.jpg", image)
print("Imagen procesada guardada en 'imagen_procesada.jpg'")
```