PROYECTO FIN DE GRADO

TÍTULO: Desarrollo de un sistema de inteligencia artificial aplicado a la agricultura de precisión

AUTOR/A: María Mirabela Toma Toma

TITULACIÓN: Grado en Ingeniería Electrónica de Comunicaciones

TUTOR/A: Cristóbal Tapia García

DEPARTAMENTO: DTE

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: René Pernas Salomón

TUTOR/A: Cristóbal Tapia García

SECRETARIO/A: Néstor Lucas Martínez

Fecha de lectura:

Calificación:

El Secretario/La Secretaria,

Resumen

El presente Proyecto Fin de Grado (PFG) tiene como propósito fundamental aprender y aplicar técnicas de Inteligencia Artificial (IA) y aprendizaje automático (ML) para dispositivos basados en microcontroladores, culminando en el desarrollo de un prototipo funcional. Como caso concreto, se ha implementado un sistema de riego inteligente utilizando un microcontrolador ESP32. El objetivo es optimizar el consumo hídrico en la agricultura mediante la predicción de la necesidad de riego y la probabilidad de lluvia, empleando un enfoque de IA en el borde. Esta iniciativa demuestra la viabilidad de desplegar algoritmos complejos de IA en hardware de bajo coste y con recursos limitados, contribuyendo al avance tecnológico y sostenibilidad agrícola.

El diseño del sistema está condicionado por las limitaciones de memoria y procesamiento del ESP32, lo que ha exigido la optimización del modelo de IA mediante TensorFlow Lite para la inferencia en tiempo real. Se ha priorizado el uso de componentes accesibles como sensores DHT22 y de humedad del suelo capacitivo, y una placa ESP32 con pantalla TFT ST7789 integrada.

La metodología incluye la recopilación de datos de sensores y fuentes históricas, la normalización con MinMaxScaler. El modelo, un Perceptrón Multicapa se ha diseñado y entrenado en Python con TensorFlow y Keras, con salidas binarias para lluvia y riego. Tras su optimización y conversión a TFLite, se integró en el firmware de Arduino del ESP32, gestionando la adquisición de datos, normalización en el dispositivo, sincronización horaria vía NTP y visualización de resultados.

Los resultados demuestran una alta fiabilidad para la toma de decisiones de riego. Sin embargo, la predicción de lluvia presenta limitaciones debido al desequilibrio de datos, lo que subraya la importancia de la calidad y representatividad del conjunto de datos y la necesidad de mecanismos de seguridad adicionales.

Este PFG valida el inmenso potencial de aplicar ML en dispositivos embebidos accesibles. La experiencia y las lecciones aprendidas establecen una base sólida y abren un camino prometedor para futuras investigaciones y ampliaciones en la agricultura de precisión.

Abstract

This Final Degree Project (FDP) aims to learn and apply Artificial Intelligence (AI) and Machine Learning (ML) techniques for microcontroller-based devices, culminating in the development of a functional prototype. As a concrete application, an intelligent irrigation system using an ESP32 microcontroller has been developed. The goal is to optimize agricultural water consumption by predicting irrigation needs and rain probability, employing an Edge AI approach. This initiative demonstrates the feasibility of deploying complex AI algorithms on low-cost, resource-constrained hardware, contributing to technological advancement and agricultural sustainability.

The system's design was constrained by the ESP32's memory and processing limitations, which required optimizing the AI model with TensorFlow Lite for real-time inference. Accessible components like DHT22 and capacitive soil moisture sensors, and an ESP32 board with an integrated TFT ST7789 display, were prioritized.

The methodology involved collecting data from sensors and historical sources, normalizing it with MinMaxScaler. The model, a Multilayer Perceptron was designed and trained in Python using TensorFlow and Keras, with binary outputs for rain and irrigation. After optimization and TFLite conversion, it was integrated into the ESP32's Arduino firmware, managing data acquisition, normalization, NTP-based time synchronization, and result display.

The results show high reliability for irrigation decision-making. However, rain prediction has limitations due to data imbalance, highlighting the importance of dataset quality and representativeness, as well as the need for additional safety mechanisms.

This FDP validates the immense potential of applying ML to accessible embedded devices. The experience and lessons learned establish a solid foundation and open a promising path for future research and extensions in precision agriculture.

Índice de figuras

Figura 1. Cómo se relacionan la IA, el ML, el deep learning y la IA generativa[3] 6
Figura 2 Diagrama de la estructura fundamental de una neurona artificial[8] 9
Figura 3. Modelo simplificado de una neurona biológica[9] 10
Figura 4. Fórmula del perceptrón [10] 11
Figura 5. Función de activación lineal[12] 13
Figura 6. Función de activación sigmoide[12] 14
Figura 7. Función de activación tanh[12] 15
Figura 8. Función de activación de ReLu [12] 16
Figura 9. Función de activación Softmax[12] 17
Figura 10. Ejemplo sobreajuste y subajuste[16] 19
Figura 11. Adaptado de [17] 20
Figura 12. Gráfica del descenso del gradiente[19] 21
Figura 13. Estructura perceptrón multicapa[20] 22
Figura 14. Campo Inteligente[27] 26
Figura 15. Frontal y trasera del microcontrolador ESP32 35
Figura 16. Frontal y trasera del sensor de temperatura y humedad DHT22 37
Figura 17. Frontal y trasera del sensor capacitivo de humedad del suelo 38
Tabla 1. Detalle de recopilación de datos para el primer modelo 40
Figura 18. Detalle de la impresión por Monitor Serie 42
Tabla 2. Detalle de recopilación de datos para el modelo final 43
Figura 19. Visualización pantalla 43
Figura 20. Prototipo del sistema de riego 44
Figura 21. Arquitectura del Sistema 44
Figura 22. Captura de resultados primer modelo predictivo de riego 53
Figura 23. Captura de resultados modelo final predicción de riego 54
Figura 24. Captura de resultados modelo final predicción de lluvia 55
Figura 25. Gráfica de perdida y precisión en el modelo final 56
Figura 26. Análisis del flujo de datos del modelo en tiempo real (1) 58
Figura 27. Análisis del flujo de datos del modelo en tiempo real (2) 59
Tabla 3. Presupuesto total del proyecto. 61

Lista de acrónimos

ADC – Conversor Analógico-Digital

AEMET – Agencia Estatal de Meteorología

API – Interfaz de Programación de Aplicaciones

CSV – Comma Separated Values (Valores Separados por Comas)

GPIO – General Purpose Input/Output (Entrada/Salida de Propósito General)

IA – Inteligencia Artificial

IDE – Integrated Development Environment (Entorno de Desarrollo Integrado)

IoT – Internet de las Cosas

ML – Machine Learning (Aprendizaje Automático)

MLP – Multilayer Perceptron (Perceptrón Multicapa)

NTP – Network Time Protocol

ODS – Objetivos de Desarrollo Sostenible

RAM – Random Access Memory (Memoria de Acceso Aleatorio)

ReLU – Rectified Linear Unit (Unidad Lineal Rectificada)

RH – Relative Humidity (Humedad Relativa)

RNA – Red Neuronal Artificial

SPI – Serial Peripheral Interface

SPIFFS – SPI Flash File System

TFT – Thin-Film Transistor (Transistor de Película Fina)

TFLite – TensorFlow Lite

Índice de contenidos

Resumen ..... **i**

Abstract ..... **iii**

Índice de figuras ..... **iv**

Lista de acrónimos ..... **v**

1. Introducción **1**

1.1. Marco y motivación del proyecto 1

1.2. Objetivos técnicos y académicos 2

1.3. Estructura del resto de la memoria 2

2. Marco tecnológico **5**

2.1. Introducción al Marco Tecnológico 5

2.2. Inteligencia Artificial 5

2.3. Machine Learning (Aprendizaje Automático) 7

2.3.1. Aprendizaje supervisado 7

2.3.2. Aprendizaje no supervisado 7

2.3.3. Aprendizaje por refuerzo 7

2.4. Deep Learning (Aprendizaje Profundo) 8

2.4.1. Neurona de McCulloch y Pitts (1943) 8

2.4.2. Perceptrón de Rosenblatt (1958) 11

2.4.3. Funciones de Activación en Redes Neuronales 13

2.4.4. Redes Neuronales 17

2.4.5. Entrenamiento de la Red 18

2.4.6. El Perceptrón Multicapa (MLP) 21

2.4.7. Tensor 22

2.4.8. Escalado y normalización 23

2.4.9. Funciones de pérdida para la clasificación 24

2.5. Inteligencia Artificial Generativa 25

2.6. Agricultura de precisión 25

3. Especificaciones y restricciones de diseño **28**

3.1. Especificaciones funcionales 28

3.1.1. Adquisición de datos ambientales 28

3.1.2. Procesamiento y predicción 28

3.1.3. Interfaz de usuario 28

3.1.4. Sincronización horaria y programación de evaluaciones 28

3.1.5. Normalización de datos y escalado: 28

3.1.6. Almacenamiento y formato del modelo: 29

3.2. Restricciones técnicas 29

3.2.1. Capacidad limitada del hardware: 29

3.2.2. Precisión y ruido en sensores: 29

3.2.3. Consumo energético: 29

3.2.4. Conectividad y dependencias externas: 29

3.2.5. Compatibilidad y soporte de librerías: 29

3.2.6. Limitaciones del modelo predictivo: 30

4. Descripción de la solución propuesta 31
4.1. Bibliotecas y Herramientas Utilizadas 31
4.1.1. Python 31
4.1.2. TensorFlow y Keras 31
4.1.3. Pandas 32
4.1.4. NumPy 32
4.1.5. Scikit-learn 32
4.1.6. TensorFlow Lite para microcontroladores (TFLite Micro) 33
4.1.7. Librerías para visualización en pantalla TFT ST7789 33
4.1.8. Biblioteca para Sensor DHT22 34
4.1.9. Biblioteca para la gestión de conectividad Wi-Fi y hora 34
4.1.10. Biblioteca para sensor de humedad del suelo (capacitivo) 34
4.2. Componentes hardware y sus características 34
4.2.1. Microcontrolador ESP32 con pantalla TFT ST7789 integrado 35
4.2.2. Sensor de temperatura y humedad DHT22 (AM2302) 36
4.2.3. Sensor de humedad del suelo (capacitivo) 38
4.3. Transmisión y Adquisición de Datos 39
4.3.1. Adquisición de datos 41
4.3.2. Almacenamiento y transmisión de datos 41
4.3.3. Uso para modelo final 42
4.4. Arquitectura general del sistema 44
4.5. Justificación de decisiones de diseño 46
4.6. Aportaciones propias y recursos externos 47
4.6.1. Aportaciones Propias 47
4.6.2. Recursos Externos y Herramientas Utilizadas 49

5. Resultados 51
5.1. Metodología de Pruebas 51
5.1.1. Pruebas del Sistema Físico 51
5.1.2. Metodología de Evaluación del Modelo Predictivo 51
5.2. Resultados Obtenidos 52
5.2.1. Pruebas del Sistema Físico 52
5.2.2. Resultados del Primer Modelo Predictivo 52
5.2.3. Resultados modelo final 54
5.3. Interpretación General de Resultados y Validación del Diseño 57

6. Presupuesto 61

7. Impacto del proyecto 63
7.1. Impacto 63
7.1.1. Impacto social 63
7.1.2. Impacto ambiental 63
7.1.3. Impacto en salud y seguridad 64
7.1.4. Impacto económico 64
7.1.5. Impacto tecnológico e industrial 64
7.2. Alineación con los Objetivos de Desarrollo Sostenible (ODS) 64

8. Conclusiones 67
8.1. Logros y Resultados Obtenidos: 67
8.2. Conocimiento Nuevo y Aspectos Detectados: 68

8.3. Líneas Futuras de Trabajo: 68

9. Referencias 71

Anexo 75

Manual de usuario 77

A.1 Descripción General del Sistema y su Propósito 77

A.2 Componentes y Preparación del Entorno 77

A.2.1. Contenido del Sistema (Componentes Hardware) 77

A.2.2. Preparación del Entorno de Desarrollo (Software y Librerías) 77

A.2.3. Conexiones Físicas (Cableado) 78

A.2.4. Carga del Programa (Firmware) 79

A.3 Proceso de Desarrollo y Despliegue de la Inteligencia del Sistema 80

A.3.1. Recopilación y Preparación de Datos 80

A.3.2. Entrenamiento y Conversión del Modelo de red neuronal 80

A.3.3. Integración y Carga del Modelo en el Microcontrolador 80

A.4 Funciones del Prototipo en Operación 81

A.4.1. Visualización en Pantalla 81

A.4.2. Evaluación del Modelo y Toma de Decisión 82

A.5 Mantenimiento y Calibración 82

A.6 Posibles Problemas y Soluciones 82

A.6.1. Problemas de Funcionamiento del Sistema 83

A.6.2. Problemas de Rendimiento del Modelo de IA 84

1. Introducción

1.1. Marco y motivación del proyecto

En las últimas décadas, el campo de la IA y el ML ha experimentado un crecimiento exponencial, revolucionando diversas industrias y ámbitos de la vida cotidiana. Estas tecnologías permiten a las máquinas aprender de los datos, identificar patrones y tomar decisiones sin intervención humana directa, lo que abre un abanico enorme de aplicaciones. Tradicionalmente, estos algoritmos se ejecutan en entornos de alto rendimiento, como servidores o sistemas en la nube, debido a su elevada demanda computacional. Sin embargo, el desarrollo de frameworks especializados y el incremento de la capacidad de hardware embebido han facilitado la implementación de modelos de IA en dispositivos con recursos limitados, como los microcontroladores, con enormes beneficios para sectores como la agricultura, la domótica o la salud.

Este trabajo se centra en la aplicación práctica del ML en un microcontrolador ESP32, plataforma de hardware abierto reconocida por su accesibilidad, bajo coste y versatilidad. El ESP32 permite la integración directa con los sensores, facilitando la creación de sistemas inteligentes embebidos. En concreto, se ha elegido un caso de uso en agricultura para el control eficiente del riego, un problema relevante en un contexto global de escasez hídrica y necesidad de prácticas sostenibles. Mediante la incorporación de sensores ambientales y un modelo predictivo basado en redes neuronales, se busca anticipar la probabilidad de lluvia y la necesidad de riego, optimizando así el consumo de agua y mejorando la productividad agrícola.

El proceso desarrollado comprende desde la adquisición y preprocesamiento de datos reales, pasando por el diseño y entrenamiento del modelo en un entorno de desarrollo tradicional (Python y TensorFlow), hasta la conversión del modelo a un formato compatible con microcontroladores (TensorFlow Lite). Este modelo se implementa en un ESP32, junto con la integración de sensores DHT22 para medir temperatura y humedad ambiental, y sensores analógicos para medir humedad del suelo. Además, se incorpora un sistema de sincronización horaria vía NTP mediante WiFi para realizar evaluaciones periódicas en horarios predefinidos, evitando evaluaciones redundantes y optimizando recursos.

Una parte fundamental del proyecto es la adaptación del modelo de ML a las limitaciones de memoria y procesamiento del microcontrolador. Esto implica técnicas de normalización de datos (MinMaxScaler), optimización del tamaño del modelo y gestión eficiente de los recursos computacionales para permitir la inferencia en tiempo real. También se ha desarrollado una interfaz sencilla de visualización mediante una pantalla TFT ST7789, que muestra la información relevante sobre temperatura, humedad, probabilidad de lluvia y decisión de riego, permitiendo al usuario interactuar con el sistema de forma directa y práctica.

Este trabajo pretende demostrar la viabilidad y el potencial de integrar técnicas avanzadas de IA en sistemas embebidos de bajo coste, ofreciendo una solución tangible para la agricultura inteligente. Además, se pretende servir como referencia y guía para futuras implementaciones que busquen aplicar ML en dispositivos similares, explorando los desafíos y oportunidades que esta combinación tecnológica presenta. Así, se contribuye al avance en la gestión eficiente de recursos y al impulso de la agricultura de precisión mediante tecnologías accesibles y replicables.

1.2. Objetivos técnicos y académicos

Los objetivos de este proyecto fin de carrera son, desde el punto de vista técnico:

- Aprender y aplicar técnicas de ML en plataformas con recursos limitados.
- Diseñar e implementar un sistema embebido capaz de adquirir datos ambientales y ejecutar inferencias locales.
- Desarrollar un modelo de IA que prediga simultáneamente la probabilidad de lluvia y la necesidad de riego, utilizando datos reales.
- Evaluar la viabilidad y precisión del sistema en un escenario realista de control de riego.
- Promover el uso de soluciones de bajo coste y sostenibles para el sector agrícola.

Desde el punto de vista académico, se adquieren las siguientes competencias y habilidades:

- Dominio en la aplicación de frameworks de ML (TensorFlow, Keras) para el diseño y entrenamiento de modelos predictivos.
- Capacidad para optimizar y desplegar modelos de IA en entornos de hardware con recursos limitados (IA en el borde, TensorFlow Lite para microcontroladores).
- Experiencia en la integración de componentes hardware (sensores, microcontroladores) y software (firmware, librerías) en sistemas embebidos.
- Habilidad para la recopilación, preprocesamiento y análisis de datos para el entrenamiento de modelos de IA, incluyendo estrategias de balanceo de clases.
- Competencia en la evaluación crítica del rendimiento de modelos de ML y la interpretación de métricas de clasificación.
- Desarrollo de habilidades en la resolución de problemas técnicos inherentes a la programación de microcontroladores y la depuración de sistemas complejos.
- Conocimiento sobre la justificación de decisiones de diseño y la consideración de impactos económicos, ambientales y sociales en proyectos tecnológicos.

1.3. Estructura del resto de la memoria

En los próximos capítulos de la memoria se ofrecerá una descripción detallada del proyecto. El Marco Tecnológico explorará las bases teóricas y las tecnologías clave que sustentan el desarrollo, como la IA, el aprendizaje automático y la agricultura de precisión. Las

Especificaciones y Restricciones de Diseño definirán los requisitos funcionales y las limitaciones técnicas que han guiado la implementación del sistema. El Diseño y Desarrollo del Sistema describirá el proceso de creación del prototipo, desde la selección de componentes y herramientas hasta la metodología de entrenamiento del modelo y su despliegue en el microcontrolador. Los Resultados presentarán la evaluación del rendimiento del sistema y del modelo de IA, destacando los logros y los desafíos encontrados. Posteriormente, se abordará el Presupuesto del proyecto, seguido por el Impacto del Proyecto en diversas esferas como la social, ambiental y económica. Finalmente, las Conclusiones resumirán los logros principales, las lecciones aprendidas y propondrán futuras líneas de investigación, mientras que las Referencias listarán todas las fuentes web consultadas.

2. Marco tecnológico

2.1. Introducción al Marco Tecnológico

El desarrollo de sistemas embebidos para aplicaciones concretas como la agricultura inteligente requiere una profunda comprensión de las tecnologías disponibles tanto en hardware como en software. El avance exponencial en microcontroladores, sensores, y algoritmos de IA ha abierto un abanico de posibilidades para implementar soluciones inteligentes y eficientes en entornos donde los recursos computacionales y energéticos son limitados.

En este contexto, el presente proyecto se enfoca en la integración de técnicas de ML en microcontroladores, específicamente usando la plataforma Arduino y el módulo ESP32, con el fin de desarrollar un prototipo funcional para el control inteligente del riego agrícola. Este marco tecnológico presenta un análisis detallado de los componentes tecnológicos que sustentan el proyecto, desde los dispositivos físicos, los sensores empleados, las librerías y software, hasta las metodologías para el desarrollo, entrenamiento e implementación del modelo de IA.

El objetivo de esta sección es proveer una base sólida que explique las tecnologías involucradas, su estado actual y las razones que motivan su elección para el proyecto.

2.2. Inteligencia Artificial

La IA es una rama de la informática que busca crear sistemas capaces de imitar procesos cognitivos humanos, como el razonamiento, la resolución de problemas, la comprensión del lenguaje y el aprendizaje. Su objetivo es dotar a las máquinas de la capacidad de tomar decisiones autónomas, interpretar el entorno y actuar en consecuencia, sin depender exclusivamente de instrucciones preprogramadas.

Esta tecnología ha evolucionado significativamente en las últimas décadas, pasando de tareas simples y estructuradas a aplicaciones altamente complejas. Hoy en día, la IA está presente en asistentes virtuales, sistemas de recomendación, vehículos autónomos, reconocimiento facial y diagnóstico médico, entre otros. Su capacidad para manejar grandes volúmenes de datos y aprender de ellos la convierte en una herramienta esencial para el desarrollo de soluciones innovadoras.

La IA es una tecnología que cambia las reglas del juego para cualquier industria y sociedad. Los beneficios de la IA son transversales a la sociedad y a los países. Esta tecnología consigue mejorar el bienestar de la población, avanzar en los objetivos de sostenibilidad medioambiental, agilizar la acción humanitaria o ayudar a preservar el patrimonio cultural.

También puede ayudar a optimizar el sistema sanitario, facilitando la detección de enfermedades y promoviendo soluciones a medida de cada estudiante o empleado favoreciendo la inclusión y la adaptación a las características del mercado de trabajo.[1]

La IA es una rama de la informática que se dedica al desarrollo de sistemas y dispositivos que pueden realizar tareas que requieren inteligencia humana, como el reconocimiento de patrones, el aprendizaje y el razonamiento. Un ejemplo de IA moderna son los asistentes virtuales como Siri o Alexa, que pueden responder a preguntas y realizar tareas específicas como reproducir música o encender las luces. Otro ejemplo es un coche autónomo con los Tesla de Elon Musk, que utiliza sensores y algoritmos de IA para conducir por las carreteras sin la intervención humana.[2]

Como se observa en la Figura 1, la IA representa el conjunto más amplio, que abarca todas las tecnologías diseñadas para imitar la inteligencia humana y que definiremos en apartados posteriores.

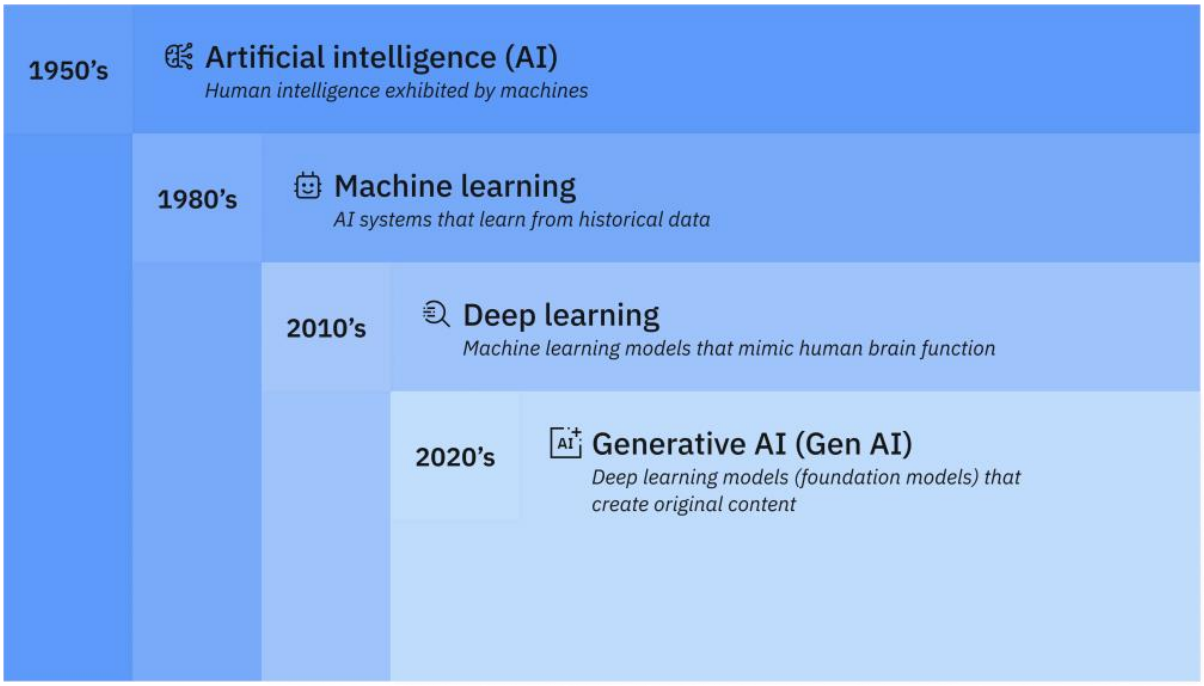


Figura 1. Cómo se relacionan la IA, el ML, el deep learning y la IA generativa[3]

Esta representación ayuda a entender cómo cada concepto se construye sobre el anterior, reflejando el avance desde sistemas que imitan la inteligencia humana hasta aquellos capaces de generar nuevas ideas y soluciones de manera autónoma.

Sin embargo, este avance no está exento de desafíos. Las preocupaciones en torno al uso ético de la IA, la transparencia de los algoritmos, el sesgo en los datos y el impacto en el empleo requieren una reflexión profunda. Por ello, es imprescindible establecer marcos regulatorios que orienten su desarrollo hacia el beneficio común y el respeto de los derechos fundamentales.

2.3. Machine Learning (Aprendizaje Automático)

El ML es una subdisciplina de la IA centrada en el desarrollo de algoritmos que permiten a las máquinas aprender y mejorar a partir de los datos, sin ser programadas explícitamente para cada tarea.

El funcionamiento del aprendizaje automático se basa en la capacidad de las máquinas para aprender de los datos. Al igual que las personas, los algoritmos de ML identifican patrones y relaciones dentro de grandes volúmenes de información para hacer predicciones o tomar decisiones. Para ello, el proceso generalmente comienza con la selección y preparación de un conjunto de datos que servirá para entrenar al modelo. Durante el entrenamiento, el algoritmo analiza los datos y ajusta sus parámetros internos con el objetivo de minimizar los errores y mejorar su precisión.[4]

Existen tres tipos principales de aprendizaje automático:

2.3.1. Aprendizaje supervisado

En este tipo de aprendizaje, el algoritmo se entrena con datos previamente etiquetados, es decir con ejemplos en los que ya se conoce el resultado esperado, ya que a cada entrada le corresponde una salida correcta. A partir de este histórico de datos, el modelo aprende a identificar patrones y relaciones entre las variables de entrada y las etiquetas de salida. De esta manera, cuando se le presentan nuevos datos, puede predecir correctamente el valor o la categoría correspondiente.
El aprendizaje supervisado se suele usar en problemas de clasificación (identificación de dígitos, diagnósticos, o detección de fraude de identidad) y regresión (predicciones meteorológicas, de expectativa de vida, de crecimiento etc).[5]

2.3.2. Aprendizaje no supervisado

En este caso, el algoritmo trabaja con datos sin etiquetar. Su objetivo es encontrar estructuras y patrones ocultos en los datos por sí mismo. Las técnicas de agrupamiento (clustering) son un ejemplo clave, donde el modelo agrupa puntos de datos similares sin una guía previa.

2.3.3. Aprendizaje por refuerzo

Este tipo aprendizaje se basa en mejorar la respuesta del modelo usando un proceso de retroalimentación. El algoritmo aprende observando el mundo que le rodea. Su información de entrada es el feedback o retroalimentación que obtiene del mundo exterior como respuesta a sus acciones. Por lo tanto, el sistema aprende a base de ensayo-error.[6]
Es muy utilizado en robótica y para entrenar agentes en videojuegos.

A pesar de su inmenso potencial, el ML también presenta desafíos, como la necesidad de datos de calidad, como veremos más adelante, la complejidad de los modelos y la posibilidad de que reproduzcan sesgos presentes en los datos de entrenamiento. Por ello, los expertos insisten en que no basta con entrenar modelos eficaces, sino también justos, transparentes y fáciles de revisar.

2.4. Deep Learning (Aprendizaje Profundo)

El deep learning, o aprendizaje profundo, es una subdisciplina avanzada dentro del ML que utiliza redes neuronales artificiales con múltiples capas (deep neural networks) para procesar y analizar grandes volúmenes de datos, especialmente aquellos no estructurados como imágenes, audio, texto o datos temporales. Estas redes están inspiradas en el funcionamiento del cerebro humano, aunque de manera mucho más simplificada, y su objetivo es transformar progresivamente la información a través de sus capas, aprendiendo patrones complejos y jerárquicos sin necesidad de que el programador defina explícitamente cada regla.

En su forma más simple, el aprendizaje profundo se puede considerar como una forma de automatizar el análisis predictivo. Si bien los algoritmos de aprendizaje automático tradicionales son lineales, los algoritmos de aprendizaje profundo se apilan en una jerarquía de complejidad y abstracción cada vez mayores.[7]

Una característica fundamental del deep learning es el aprendizaje jerárquico. En este proceso, las capas inferiores de la red capturan características simples, como bordes o texturas en una imagen, mientras que las capas superiores combinan esta información para identificar patrones más complejos, como objetos, rostros o escenas completas. Este mismo principio se puede aplicar a datos temporales y sensoriales: en tu modelo, las capas iniciales pueden interpretar señales básicas del sensor de humedad o temperatura, mientras que las capas más profundas identifican patrones relacionados con las condiciones que preceden la lluvia o la necesidad de riego.

2.4.1. Neurona de McCulloch y Pitts (1943)

En 1943, Warren McCulloch y Walter Pitts propusieron el primer modelo fundamental de neurona artificial, sentando las bases de las redes neuronales modernas. Este modelo se conoce como Unidad Lógica de Umbral (TLU - Threshold Logic Unit) o Unidad de Umbral Lineal (LTU - Linear Threshold Unit), y es comúnmente denominado Perceptrón en su forma más básica.

El modelo opera con un conjunto de n entradas, representadas como x1,x2,…,xj,…,xn. A estas entradas se les aplica una ponderación lineal y se les suma un umbral, también

conocido como sesgo (bias), denotado por b. La combinación lineal de las entradas y el sesgo se expresa matemáticamente como:

$$z = \sum_{j=1}^{n}(x_j \cdot w_j) + b \qquad (1)$$

En la ecuación (1), wj son los pesos asociados a cada entrada xj.

Posteriormente, a este valor z se le aplica una función de activación no lineal. En el modelo original de McCulloch y Pitts, esta función es de tipo escalonada (también conocida como función de paso binario o Heaviside step function), que produce una salida digital o binaria. La función de activación de la ecuación (2) se define como:

$$a = f(z)\begin{cases} 1 \; si \; z \geq 0 \\ 0 \; si \; z < 0 \end{cases} \qquad (2)$$

En este modelo, las entradas (x1,x2,…,xn) conceptualmente equivalen a las dendritas de una neurona biológica, el parámetro b representa el umbral o sesgo que debe superarse para que la neurona se active, y la salida a simula el axón. Durante el proceso de entrenamiento de una red neuronal, el valor de la salida a se compara con los valores reales esperados (denominados y) para ajustar los pesos y el sesgo, permitiendo que el modelo aprenda a realizar predicciones o clasificaciones.

En la Figura 2 se puede apreciar el modelo simplificado de una neurona artificial, ilustrando sus componentes principales.

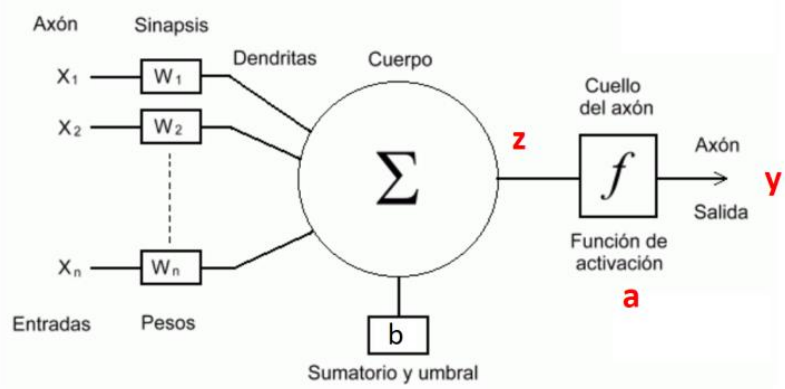


Figura 2. Diagrama de la estructura fundamental de una neurona artificial[8]

La neurona biológica es una célula del sistema nervioso encargada de transmitir información mediante señales eléctricas y químicas. La podemos observar en la Figura 3 y está formada por tres partes principales:

- Dendritas: reciben señales de otras neuronas.
- Soma (cuerpo celular): procesa la información.
- Axón: transmite la señal hacia otras neuronas, músculos o glándulas.

Estas células forman redes complejas que permiten funciones como el pensamiento, la memoria o el movimiento.

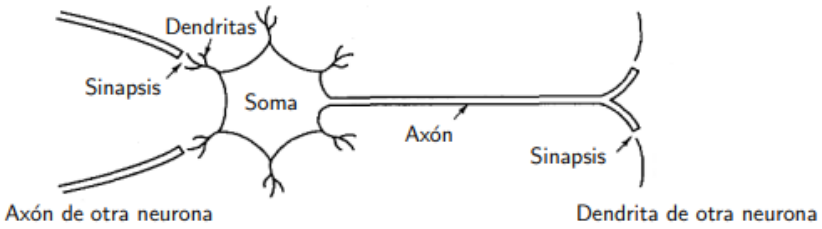


Figura 3. Modelo simplificado de una neurona biológica[9]

Las neuronas artificiales, utilizadas en modelos de aprendizaje profundo, se inspiran en este mismo principio. Al igual que la neurona humana recibe múltiples señales, la neurona artificial recibe varias entradas (datos). Luego, realiza una operación matemática para procesarlas (similar al papel del soma), y finalmente genera una salida, que puede influir en otras neuronas del modelo, como el axón lo hace en el cuerpo humano.

Aunque son mucho más simples que las neuronas biológicas, las neuronas artificiales imitan el comportamiento básico de las humanas: recibir información, procesarla y transmitir una respuesta. Esta semejanza ha sido clave para el desarrollo de las redes neuronales artificiales, que hoy permiten a las máquinas aprender tareas complejas como el reconocimiento de voz, imágenes o la generación de texto.

Pero la neurona artificial de McCulloch y Pitts tenía limitaciones:

- Su modelo era muy simplificado: cada neurona funcionaba como una compuerta lógica, sin posibilidad de aprender.
- No incluía ningún mecanismo de aprendizaje o ajuste de conexiones (pesos).
- No podía resolver problemas que requerían funciones no lineales como XOR, una limitación que se descubriría más adelante.
- Las redes propuestas eran estáticas: no podían adaptarse ni mejorar con experiencia o ejemplos.

Estas limitaciones, particularmente la ausencia de un mecanismo de aprendizaje y la incapacidad de resolver problemas no linealmente separables, representaron un desafío significativo para el avance de las redes neuronales. Sin embargo, estas deficiencias también sentaron las bases para futuras investigaciones, impulsando la búsqueda de modelos neuronales que pudieran aprender de los datos y adaptarse, dando origen a mejoras cruciales en la década siguiente. Es en este contexto que surgiría el Perceptrón, un modelo

que, si bien se basaba en la estructura fundamental de McCulloch y Pitts, incorporaba una capacidad de aprendizaje que revolucionaría el campo.

2.4.2. Perceptrón de Rosenblatt (1958)

A finales de la década de 1950, Frank Rosenblatt introdujo el perceptrón, un modelo que incorporaba por primera vez un mecanismo de aprendizaje supervisado. Inspirado en la neurobiología, el perceptrón podía ajustar sus pesos en función de los errores cometidos durante el entrenamiento, permitiendo así reconocer patrones en los datos y representar funciones más complejas. Este avance marcó un hito fundamental en la evolución de la IA.

El perceptrón funciona como una neurona artificial que recibe múltiples señales de entrada. Según la Regla de Aprendizaje del Perceptrón (Perceptron Learning Rule), el algoritmo ajusta automáticamente los coeficientes de peso óptimo para cada entrada. Para determinar si una neurona “se enciende” o no, las características de los datos de entrada se multiplican por esos pesos.

Como se ilustra en la Figura 4, si la suma de las señales supera un umbral determinado, se produce una señal o, por el contrario, no se emite ningún resultado.

En el marco del método de aprendizaje supervisado de ML, es lo que permite predecir la categoría de una muestra de datos. Por tanto, se trata de un elemento esencial.

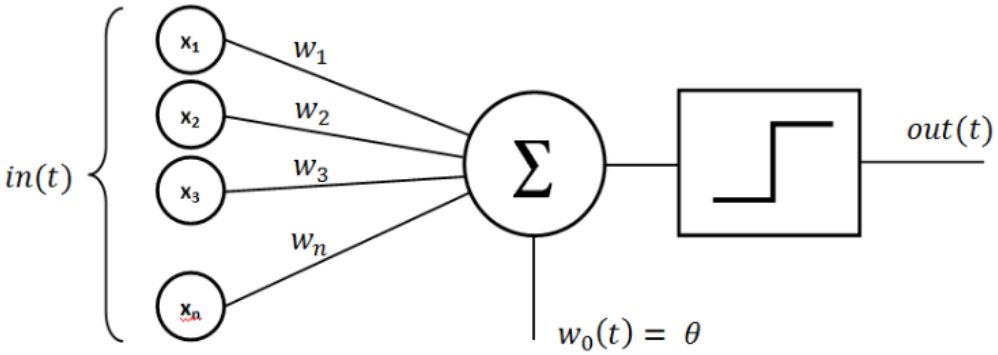


Figura 4. Fórmula del perceptrón [10]

El perceptrón realiza una tarea muy simple pero poderosa: clasifica entradas en una de dos categorías. Es decir, toma una decisión binaria (por ejemplo, "sí" o "no", "spam" o "no spam").

- Recibe varias entradas numéricas (por ejemplo, características de un correo electrónico).
- Multiplica cada entrada por un peso (un valor que representa su importancia).

- Suma todos esos valores.
- Aplica una función de activación (por ejemplo, una función escalón) que decide si el resultado final es 1 (activo) o 0 (inactivo).

El perceptrón se utiliza principalmente para la clasificación de datos simples, como determinar si una imagen representa un círculo o un cuadrado. Además, sirve como base para construir redes neuronales más complejas, como el perceptrón multicapa, que permite resolver problemas más avanzados. También es fundamental para entender cómo las máquinas pueden aprender, ya que demuestra cómo es posible ajustar los pesos a partir de ejemplos mediante un proceso de entrenamiento.

Aunque el perceptrón por sí solo solo puede resolver problemas linealmente separables, su importancia radica en que fue el primer modelo de red neuronal capaz de aprender, y abrió el camino a todo el desarrollo moderno del deep learning.

El perceptrón simple es una red neuronal muy sencilla con una única neurona binaria (función de activación escalón) en la capa de salida que puede entrenarse para clasificar un conjunto de datos de entrada en dos categorías. Como se ha visto el entrenamiento es exitoso si los datos son linealmente separables, es decir pertenecen a uno u otro de los semiespacios delimitados por algún hiperplano cuya orientación y desplazamiento están determinados por el conjunto de valores que toman los pesos sinápticos de las conexiones entre las neuronas de la capa de entrada y la única neurona de la capa de salida y del umbral de ésta última. La red aprende a partir de un conjunto de datos etiquetados (de clases conocidas) mediante un procedimiento iterativo que actualiza los pesos sinápticos y el umbral en cada paso hasta lograr clasificar los datos correctamente. En caso de que no haya separabilidad lineal o para clasificación en más de dos categorías se utilizan redes con arreglos multicapa recurriéndose a métodos de entrenamiento basados en el método del gradiente descendente para encontrar el mínimo de una función de error.[11]

2.4.3. Funciones de Activación en Redes Neuronales

Para que el perceptrón pueda tomar una decisión, necesita convertir la suma ponderada de sus entradas en una salida clara. Aquí entra en juego la función de activación, que es un mecanismo matemático que define si la neurona se "activa" o no. En el perceptrón original, esta función era muy simple: si el resultado superaba un cierto umbral, la salida era 1; de lo contrario, era 0. Esta se conoce como función escalón o función de activación binaria.

Las funciones de activación permiten a la red neuronal introducir decisiones no lineales, lo que es fundamental para que pueda aprender y representar patrones complejos en los datos. Elegir la función de activación adecuada para cada situación facilita un entrenamiento

más eficiente y mejora la capacidad del modelo para generalizar y ofrecer mejores resultados.

- **Función de activación lineal**

La función de activación lineal podemos observarla en la Figura 5. Es la más simple de todas, ya que devuelve la entrada x tal cual como salida.

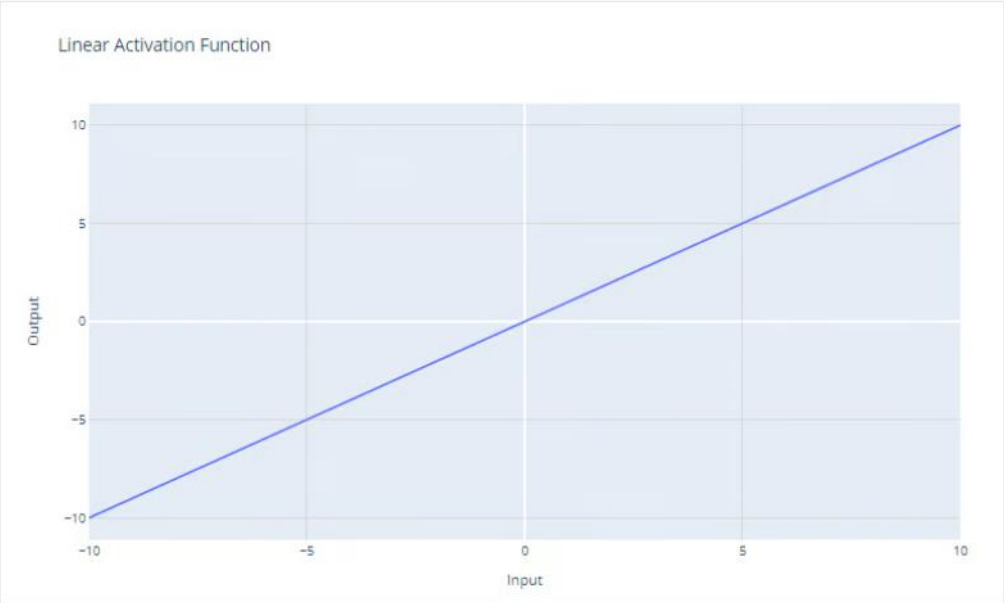


Figura 5. Función de activación lineal[12]

Representada gráficamente, corresponde a una línea recta con pendiente igual a 1. Matemáticamente, se define como en la ecuación (3):

$$f(x) = x \qquad (3)$$

- **Función de activación sigmoide**

La función de activación sigmoide que encontramos en la Figura 6, comúnmente representada como σ(x)\sigma(x)σ(x), ha sido históricamente fundamental en el desarrollo de las redes neuronales. Esta función transforma una entrada de valor real en un valor entre 0 y 1 y su fórmula matemática la encontramos en la ecuación (4).

$$f(x) = \frac{1}{1+e^{-x}} \qquad (4)$$

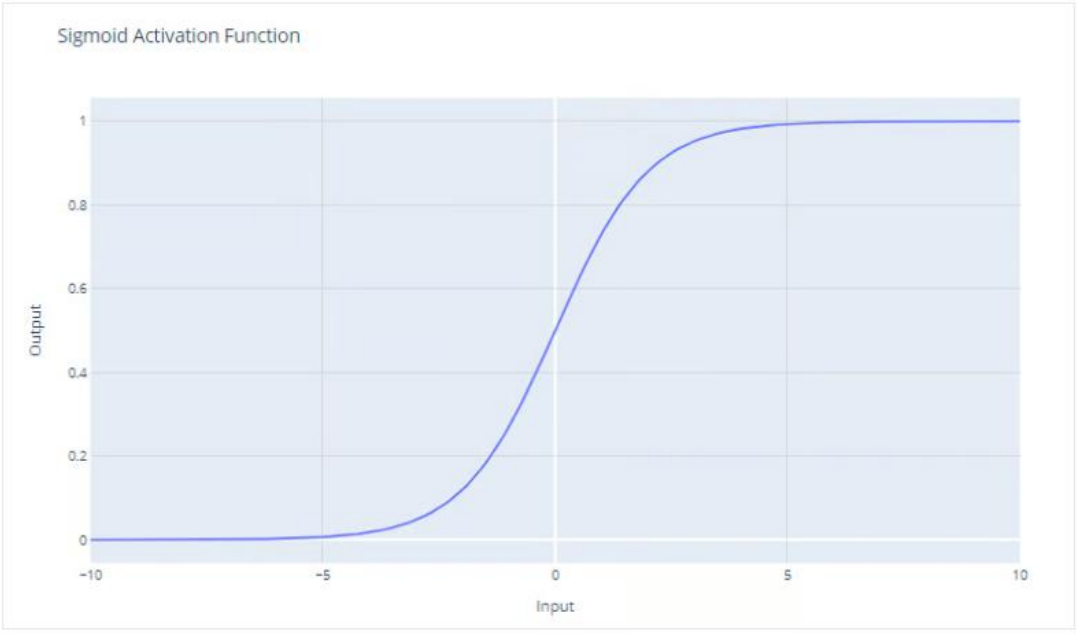


Figura 6. Función de activación sigmoide[12]

Su forma característica en "S" hace que los valores muy negativos se aproximen a 0, mientras que los valores muy positivos se acerquen a 1. Gracias a esta propiedad, sus salidas pueden interpretarse fácilmente como probabilidades, lo que la convierte en una opción natural para problemas de clasificación binaria.

- **Activación de tangente hiperbólica (tanh)**

La función de activación tangente hiperbólica, conocida como *tanh*, se define matemáticamente como en la ecuación (5):

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (5)$$

Esta función genera valores en un rango de -1 a +1, lo que le permite manejar valores negativos de forma más efectiva que la función sigmoide, cuyo rango va de 0 a 1, como podemos observar en la figura 7:

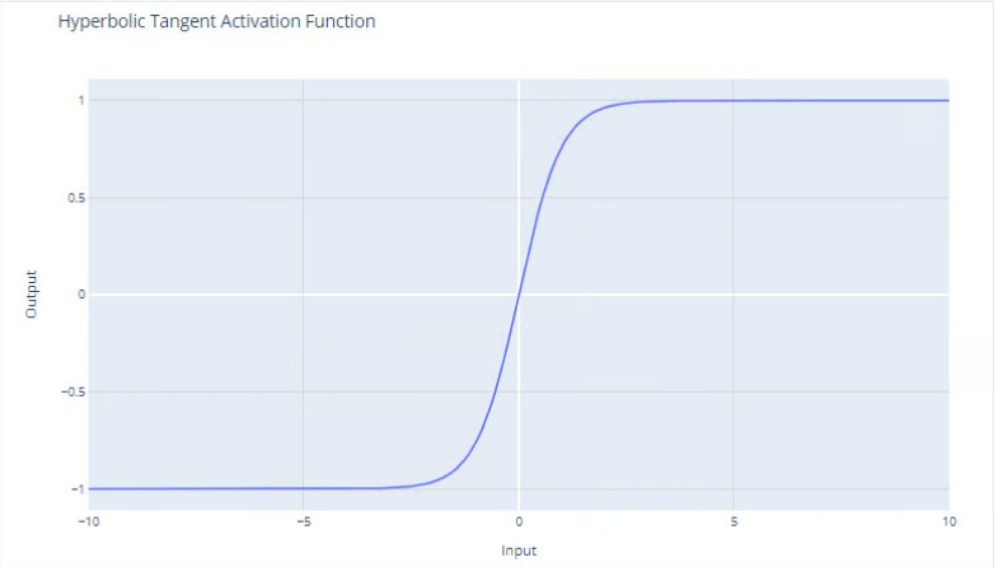


Figura 7. Función de activación tanh[12]

A diferencia de la sigmoide, tanh está centrada en cero, lo que significa que sus salidas son simétricas alrededor del origen. Esta propiedad facilita que los algoritmos de aprendizaje converjan más rápido y mejora la estabilidad durante el entrenamiento.

- **Activación de unidad lineal rectificada (ReLU)**

La función de activación unidad lineal rectificada, conocida como ReLU, se define como en la ecuación (6):

$$f(x) = \max(0, x) \quad (6)$$

Esta función, como podemos apreciar en la Figura 8, devuelve cero para valores de entrada negativos y la misma entrada cuando es positiva. Para valores mayores que cero, ReLU se comporta como una función lineal con pendiente 1, lo que permite que el gradiente se propague sin cambios durante la retropropagación. Esta característica es clave para evitar el problema del desvanecimiento del gradiente y facilita el entrenamiento de redes profundas.

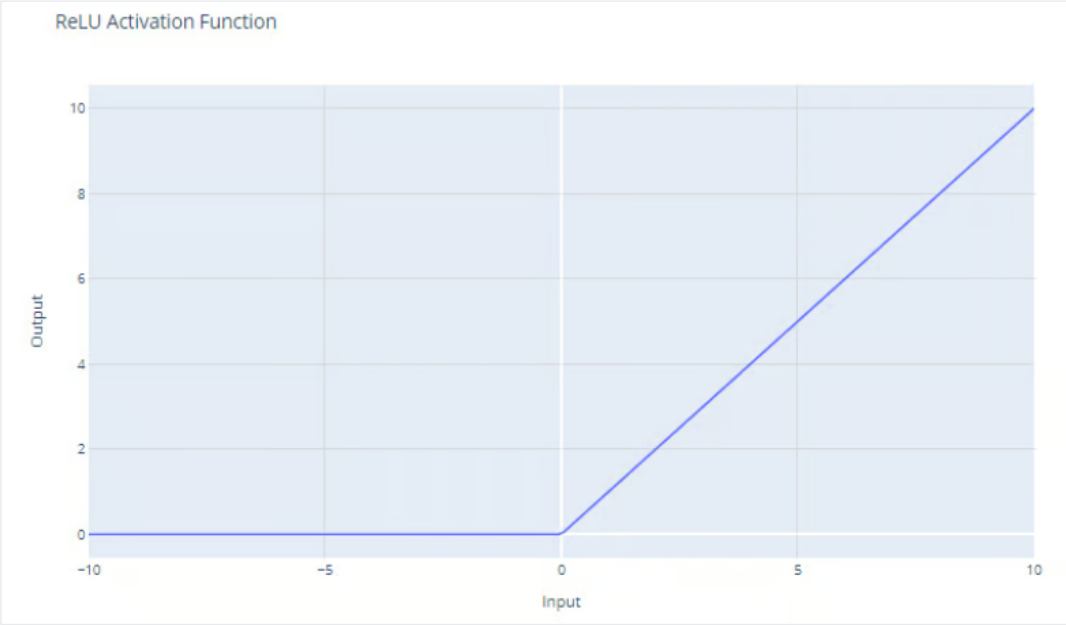


Figura 8. Función de activación de ReLu [12]

- **Activación Softmax**

La función de activación softmax que encontramos en la Figura 9, también llamada función exponencial normalizada, es especialmente útil en problemas de clasificación multiclase. Con la ayuda de la ecuación (7), podemos ver como esta función toma un vector de entrada, conocido como logits, que contiene las puntuaciones brutas para cada clase generadas por las capas previas de la red neuronal.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_i}} \qquad (7)$$

El resultado de softmax es una distribución de probabilidad en la que la suma de todas las salidas es igual a uno. Cada valor en el vector resultante representa la probabilidad de que la entrada pertenezca a una clase específica.

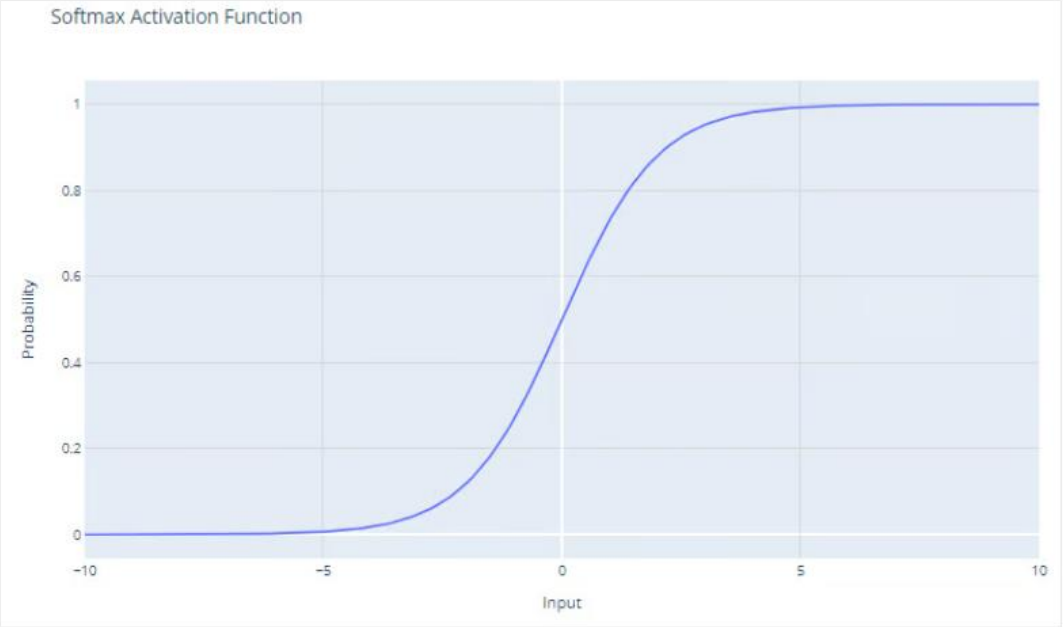


Figura 9. Función de activación Softmax[12]

La elección de la función de activación depende del tipo de problema que se intenta resolver.

2.4.4. Redes Neuronales

Una red neuronal es un modelo computacional inspirado en el funcionamiento del cerebro humano. Una red neuronal se compone de varias capas:

- Capa de entrada: recibe las variables del entorno (ej., temperatura, humedad, mes).
- Capas ocultas: transforman la información mediante operaciones no lineales.
- Capa de salida: genera una predicción o decisión (ej., necesidad de riego).

Independientemente de la capa de la que forme parte, cada nodo realiza algún tipo de tarea o función de procesamiento sobre cualquier entrada que reciba del nodo anterior (o de la capa de entrada). Esencialmente, cada nodo contiene una fórmula matemática, con cada variable dentro de la fórmula ponderada de forma diferente. Si el resultado de aplicar esa fórmula matemática a la entrada supera un determinado umbral, el nodo pasa los datos a la siguiente capa de la red neuronal. Si la salida está por debajo del umbral, no se pasa ningún dato a la capa siguiente.[13]

Las conexiones entre estas neuronas están asociadas a pesos que se ajustan durante un proceso de aprendizaje. La fórmula básica (8) para la activación de una neurona se puede expresar como:

$$y = f(w \times x + b) \quad (8)$$

Donde:

- **Y** es la salida de la neurona.
- **W** son los pesos de las conexiones.
- **X** son las entradas.
- **B** es el sesgo o bias.
- **F** es la función de activación, como la sigmoide o ReLU.[14]

Las redes neuronales biológicas se destacan por su plasticidad, paralelismo y capacidad de aprendizaje, lo que les permite adaptarse a nuevas experiencias.

Las **redes neuronales artificiales** se utilizan en aplicaciones como reconocimiento de imágenes y procesamiento de lenguaje natural, transformando la interacción con la tecnología.[14]

2.4.5. Entrenamiento de la Red

El entrenamiento de una red neuronal artificial consiste en ajustar los pesos sinápticos que conectan las neuronas, con el objetivo de minimizar el error entre la salida que predice la red y el valor esperado. Este proceso es iterativo y se lleva a cabo mediante algoritmos de optimización que permiten a la red aprender de los errores y mejorar su precisión con el tiempo.

Durante una época, el modelo recibe una serie de ejemplos de entrenamiento y ajusta sus parámetros (como los pesos de las conexiones en una red neuronal) en función de los errores cometidos en la predicción de las respuestas correctas. Una vez que todos los ejemplos de entrenamiento han sido vistos por el modelo, se completa una época y se repite el proceso de entrenamiento para tantas épocas como sea necesario para mejorar la precisión del modelo.[15]

Para optimizar el proceso de entrenamiento y hacer que el aprendizaje sea más eficiente, los datos de entrada suelen dividirse en pequeños grupos llamados batches. Cada batch es procesado de forma independiente y los ajustes en los pesos se realizan después de cada uno, permitiendo actualizaciones más frecuentes y estables durante la época. Esta estrategia mejora la convergencia y permite manejar grandes volúmenes de datos sin necesidad de procesarlos todos de una vez.

Sin embargo, durante el entrenamiento pueden surgir problemas.
El subajuste ocurre cuando el modelo es demasiado simple o tiene una capacidad limitada para representar la complejidad de los datos, lo que genera un rendimiento pobre incluso en los datos de entrenamiento.[15]

Por otro lado, el sobreajuste por epoch, también conocido como sobreajuste por iteración, es un fenómeno que ocurre durante el entrenamiento de modelos de aprendizaje automático, como las redes neuronales. Se refiere a una situación en la que el modelo se ha ajustado demasiado a los datos de entrenamiento y ha perdido la capacidad de generalizar correctamente para nuevos datos o ejemplos no vistos previamente.[15]

Por ello, es fundamental controlar el número de épocas y utilizar técnicas de regularización para evitar estos problemas.

En la Figura 10 se ilustran los diferentes escenarios de ajuste de un modelo: subajuste, ajuste apropiado y sobreajuste, fundamentales para evaluar el rendimiento en el aprendizaje automático.

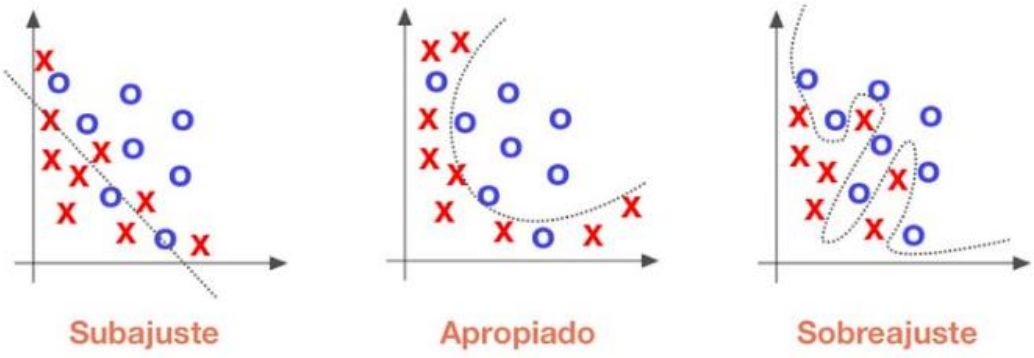


Figura 10. Ejemplo sobreajuste y subajuste[16]

Para que la red neuronal pueda aprender de manera efectiva durante estas épocas y batches, es necesario contar con un mecanismo que ajuste los pesos de forma correcta y eficiente. Aquí es donde entra en juego la retropropagación del error y el descenso del gradiente, dos conceptos fundamentales en el entrenamiento de redes neuronales.

- **Retropropagación del error**

Es un mecanismo que permite a las neuronas aprender por sí mismas. Se entrena a la red para que aprenda de sus propios errores. En la retropropagación se utiliza la regla de la cadena, con la que se calcula la derivada del error de cada capa y luego la derivada de la penúltima capa. A estas derivadas se les conoce por el nombre de gradiente. Se utiliza el descenso del gradiente para ajustar los parámetros o pesos de las neuronas.[17] Un ejemplo de cómo funciona la retropropagación lo podemos encontrar en la Figura 11.

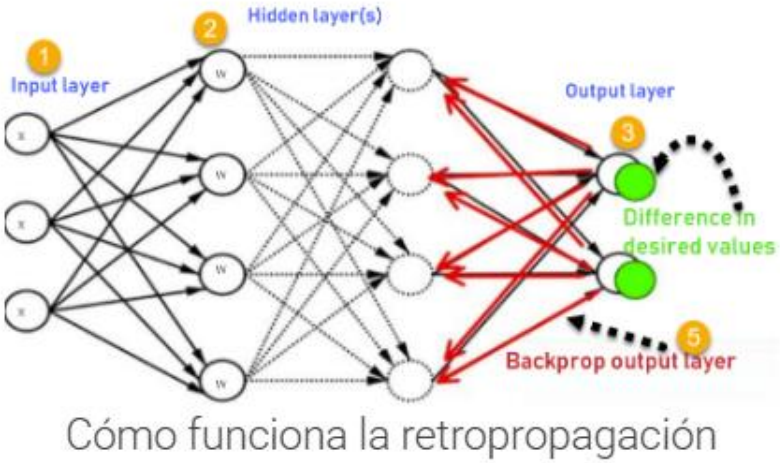


Figura 11. Adaptado de [17]

El propósito de este mecanismo es identificar qué conexiones deben ajustarse y en qué dirección para que, en futuras iteraciones, el error disminuya. La retropropagación no solo ajusta los pesos, sino que también permite que la red generalice mejor cuando se enfrenta a nuevos datos. Sin este algoritmo, el entrenamiento de redes profundas sería prácticamente inviable debido a la complejidad de sus múltiples capas interconectadas.

- **Descenso del gradiente**

Para disminuir el error cometido se realiza una estimación del ajuste que se debería hacer en los valores de entrada. Esta estimación se hace por el método del descenso del gradiente que calcula la pendiente hasta un error cero por medio de las derivadas. Al valor del gradiente o derivada se le resta al peso y, con este cálculo, se va reduciendo el valor de error. En cada iteración el valor de predicción de la red se acerca cada vez más al valor real.[18]

En la Figura 12 se ilustra visualmente el proceso del descenso del gradiente. La red neuronal, al entrenarse, se mueve por la superficie del error buscando la menor altura posible, es decir, el menor valor de pérdida. Cada paso que da está guiado por el cálculo del gradiente,

que indica en qué dirección debe ajustar los pesos. El objetivo es alcanzar un punto óptimo donde el error se minimiza.

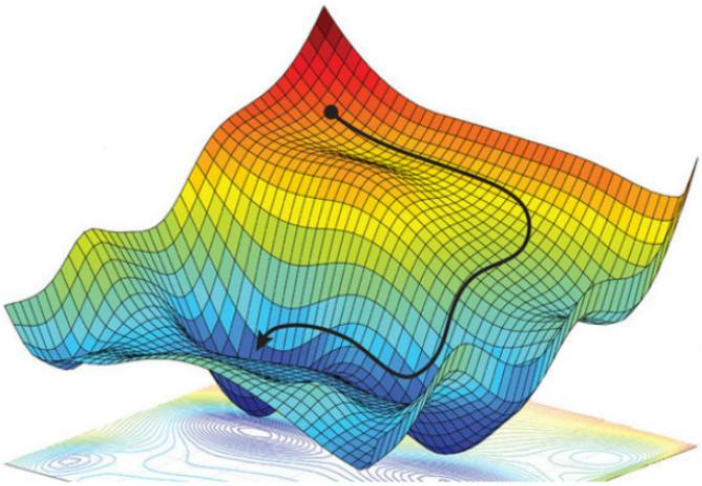


Figura 12. Gráfica del descenso del gradiente[19]

En términos prácticos, el descenso del gradiente es la base del aprendizaje de las redes neuronales: permite que, a través de sucesivas correcciones, la red "aprenda" a aproximarse a la solución correcta. La imagen ayuda a comprender que este proceso no siempre es lineal ni sencillo, ya que la función de error puede tener múltiples mínimos y formas complejas. Por ello, existen variantes del descenso del gradiente (como Momentum**,** RMSProp o Adam) que ayudan a mejorar la eficiencia y evitar quedarse atrapados en soluciones subóptimas.

2.4.6. El Perceptrón Multicapa (MLP)

Como ya hemos comentado, el perceptrón está inspirado en el funcionamiento de las neuronas biológicos. Sin embargo, nuestros cerebros están formados por trillones (entre 100 y 1000) de conexiones entre neuronas. Así pues, si queremos conseguir modelos más "inteligentes" necesitamos conectar perceptrones en capas consecutivas capaces de representar distribuciones de datos no lineales. Esta arquitectura que se ilustra en la Figura 13 se conoce con el nombre de Perceptrón Multicapa, y es la arquitectura básica de las redes neuronales.

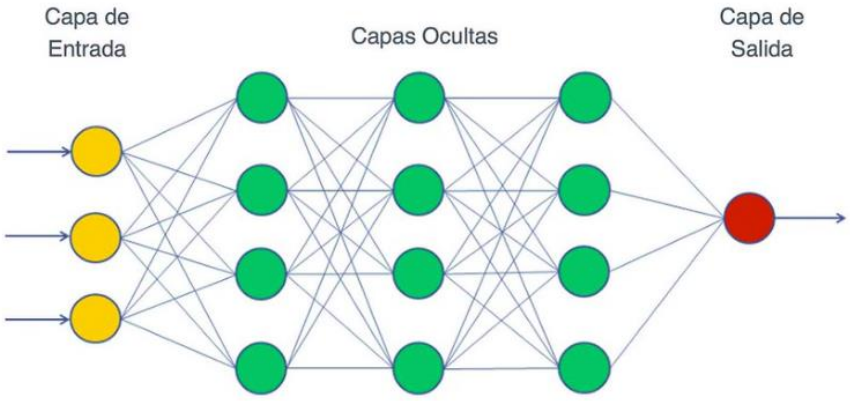


Figura 13. Estructura perceptrón multicapa[20]

Un MLP es un tipo de red neuronal que está formada por varias capas de neuronas. Cada neurona en una capa está conectada a todas las neuronas de la siguiente capa, lo que significa que las conexiones son densas o "fully connected". Estas redes funcionan tomando un conjunto de entradas, aplicándoles una combinación lineal mediante pesos, y luego pasando ese resultado a través de una función de activación no lineal. Esto se repite capa por capa hasta llegar a la capa final, que genera la salida o predicción.

Mientras que las redes de una sola capa, como el perceptrón simple, solo pueden resolver problemas linealmente separables, las MLP pueden aprender y representar funciones altamente no lineales gracias a sus capas ocultas y funciones de activación no lineales.

Las arquitecturas de las MLP pueden variar significativamente en profundidad (número de capas ocultas) y ancho (número de neuronas por capa), dependiendo de la complejidad del problema y del conjunto de datos disponible. Un mayor número de capas y neuronas permite a la red aprender patrones más complejos, pero también aumenta el riesgo de sobreajuste, especialmente si no se cuenta con suficientes datos de entrenamiento.

Es fundamental seleccionar cuidadosamente la arquitectura de la red para equilibrar la capacidad de generalización y el rendimiento en el conjunto de entrenamiento. Técnicas como la regularización y el early stopping pueden ayudar a prevenir el sobreajuste en redes profundas, temas que se discutirán en secciones posteriores.[21]

2.4.7. Tensor

Un tensor es un elemento matemático que almacena un conjunto de valores numéricos. Puede ser unidimensional (vector), bidimensional (matriz) o de más dimensiones. Estos tensores en Python se almacenan en forma de arrays NumPy, normalmente.[22]

Si tenemos un conjunto de datos con varias muestras de entrada, cada muestra puede representarse como un vector (un tensor de una dimensión). Cuando juntas muchas

muestras en un lote (batch), tienes una matriz, que es un tensor 2D: el número de filas es la cantidad de muestras y las columnas son las características de cada muestra.

Durante la propagación hacia adelante, estos vectores o matrices se multiplican por matrices de pesos (también tensores), y se les aplica una función de activación para producir las salidas de cada capa. A medida que las redes se vuelven más complejas, pueden usar tensores de mayor dimensión, pero en redes clásicas normalmente se manejan tensores de 1 o 2 dimensiones.

Entonces, para resumir, los tensores nos permiten manejar eficientemente tanto los datos de entrada, como los parámetros de la red y las activaciones internas, facilitando operaciones matemáticas en lotes, lo que es clave para entrenar las redes de forma rápida y efectiva.

En el aprendizaje profundo los datos son esenciales para que los modelos aprendan patrones complejos y a realizar tareas asombrosas, sin datos los modelos son algoritmos sin dirección que no pueden tomar decisiones. Son fundamentales para que los modelos profundicen en tareas complejas a medida que exponen a más información.[23]

2.4.8. Escalado y normalización

Las redes neuronales aprenden ajustando pesos durante el entrenamiento. Si las entradas tienen diferentes escalas (por ejemplo, una variable va de 0 a 1 y otra de 0 a 1000), eso puede dificultar la convergencia del entrenamiento. Algunas neuronas pueden dominar sobre otras, y el descenso del gradiente se vuelve inestable o muy lento.

Normalización (estandarización): consiste en transformar los datos de forma que todos los predictores estén aproximadamente en la misma escala.

- Normalización Z-score (StandardScaler): dividir cada predictor entre su desviación típica después de haber sido centrado, de esta forma, los datos pasan a tener una distribución normal.
- Estandarización max-min (MinMaxScaler): transformar los datos de forma que estén dentro del rango [0, 1].[24]

2.4.9. Funciones de pérdida para la clasificación

La Pérdida de entropía cruzada binaria (BCE) es una medida de rendimiento de los modelos de clasificación que da como resultado una predicción con un valor de probabilidad normalmente entre 0 y 1, y este valor de predicción corresponde a la probabilidad de que una muestra de datos pertenezca a una clase o categoría. En el caso de la Pérdida de

entropía cruzada binaria, hay dos clases distintas. Pero, en particular, una variante de la pérdida de entropía cruzada, la Entropía Cruzada Categórica, se aplica a escenarios de clasificación multiclase.

➔ **Entropía cruzada (Cross-Entropy Loss)**

- Es la más usada en problemas de clasificación.
- Mide la diferencia entre la distribución de probabilidad predicha y la distribución real (las etiquetas).
- Para clasificación binaria se usa la binary cross-entropy y para clasificación multiclase la categorical cross-entropy.
- La fórmula para clasificación binaria la encontramos en la ecuación (9):

$$L(y, f(x)) = -[y * \log(f(x)) + (1 - y) * \log(1 - f(x))] \quad (9)$$

Donde:

- L representa la función binaria de pérdida de entropía cruzada
- y es la etiqueta binaria verdadera (0 ó 1)
- f(x) es la probabilidad prevista de la clase positiva (entre 0 y 1)[25]

➔ **Función de pérdida de hinge (Hinge Loss)**

- Se usa mucho en máquinas de vectores soporte (SVM), pero también en algunas redes neuronales.
- Para clasificación binaria, fomenta que el margen entre clases sea grande.

➔ **Función de pérdida logarítmica**

- Muy relacionada con la entropía cruzada, usada para problemas de clasificación probabilística.

La función de pérdida es fundamental en las redes neuronales porque actúa como una guía durante el proceso de aprendizaje, ya que proporciona la señal necesaria para ajustar los pesos del modelo mediante algoritmos como el descenso de gradiente. Además, esta función mide el rendimiento del modelo, indicándonos qué tan precisas son sus predicciones en comparación con las etiquetas reales. Otro aspecto importante es que la elección

adecuada de la función de pérdida influye directamente en la convergencia del modelo, facilitando que aprenda más rápido y con mayor precisión.

En conclusión, la función de pérdida no solo evalúa el desempeño del modelo, sino que también es un componente clave que determina la efectividad y eficiencia del entrenamiento en problemas de clasificación.

2.5. Inteligencia Artificial Generativa

La IA generativa es una rama avanzada del deep learning que se enfoca en la creación de contenido nuevo a partir de datos ya existentes. A través de modelos entrenados con enormes cantidades de información, esta tecnología puede generar texto, imágenes, música, código, videos e incluso simulaciones tridimensionales que no existían previamente.

2.6. Agricultura de precisión

La agricultura de precisión es un enfoque moderno para la gestión de cultivos que utiliza tecnologías avanzadas con el objetivo de optimizar el uso de recursos, mejorar el rendimiento y reducir el impacto ambiental. A través del uso de sensores, sistemas de información geográfica (SIG), imágenes satelitales, y modelos predictivos, este enfoque permite tomar decisiones informadas en tiempo real sobre aspectos clave del proceso agrícola, como la fertilización, el control de plagas y el riego.

La sostenibilidad en la agricultura se está convirtiendo en un factor cada vez más importante a medida que la industria se ve afectada por el aumento de los costes de producción y los requisitos de seguridad alimentaria, así como la demanda de los clientes de productos de mejor calidad y más saludables.[26]

Para ello, el desarrollo de infraestructuras basadas en el Internet de las Cosas (IoT), como redes de sensores distribuidos, permite recoger datos en tiempo real sobre variables como temperatura, humedad relativa del aire y humedad del suelo. Estos datos, combinados con modelos predictivos basados en IA, facilitan la automatización de decisiones precisas y localizadas, especialmente en el riego, reduciendo el desperdicio de agua y mejorando la eficiencia energética de los sistemas agrícolas.

El uso de redes neuronales y técnicas de IA se vuelve especialmente relevante en este contexto. Estas herramientas permiten procesar grandes volúmenes de datos agroambientales y generar predicciones precisas sobre fenómenos como la ocurrencia de precipitaciones o las necesidades específicas de cada cultivo. La integración de estas

predicciones en sistemas automatizados de toma de decisiones contribuye a optimizar el uso del agua y minimizar el riesgo de estrés hídrico en las plantas.

La agricultura de precisión brinda un mejor control sobre la producción de cultivos y reduce los riesgos de producción, utilizando la IA y el análisis de imágenes satelitales para realizar mapeos de parcelas y monitorización 360.[26]

La Figura 14 ilustra cómo la agricultura de precisión utiliza drones y sensores para la recolección de datos, un paso fundamental que habilita la aplicación de técnicas de ML para optimizar el rendimiento de los cultivos y la gestión del campo.

Figura 14. Campo Inteligente[27]

La integración de datos provenientes de distintas fuentes, como satélites, sensores en campo y registros manuales, genera un panorama completo y en tiempo real de las condiciones agrícolas. Esta información detallada permite a los agricultores ajustar sus estrategias de manejo de manera dinámica, adaptándose rápidamente a cambios ambientales o imprevistos.

Los algoritmos de IA analizan esta información para detectar patrones y tendencias que serían difíciles de identificar mediante métodos tradicionales. Gracias a estas capacidades predictivas, es posible anticipar fenómenos como el desarrollo de plagas, deficiencias nutricionales o condiciones climáticas adversas, lo que facilita la implementación de medidas preventivas y oportunas. Esto no solo ayuda a mantener la salud de los cultivos, sino que también contribuye a maximizar los rendimientos y la calidad de la producción.

El uso de análisis predictivos en la planificación agrícola permite definir los momentos óptimos para actividades críticas como la siembra, el riego y la fertilización. Esta planificación avanzada contribuye a un uso más eficiente de los insumos y reduce el impacto

ambiental, haciendo que la agricultura sea no solo más productiva, sino también más respetuosa con el entorno natural.

La agricultura de precisión representa una transformación significativa en la forma en que se gestionan los cultivos y recursos naturales. La combinación de sensores, imágenes satelitales, análisis predictivos y modelos de IA permite una agricultura más eficiente, sostenible y resiliente frente a los desafíos actuales como el cambio climático y la escasez de agua. Esta integración tecnológica no solo mejora la productividad y rentabilidad de las explotaciones, sino que también contribuye a la conservación ambiental y a la seguridad alimentaria global. Así, el desarrollo y adopción continua de estas innovaciones será clave para asegurar un futuro agrícola más inteligente y sostenible.

3. Especificaciones y restricciones de diseño

El diseño del sistema se ha planteado teniendo en cuenta tanto las necesidades funcionales del proyecto como las limitaciones inherentes al hardware y las condiciones de uso en un entorno agrícola. A continuación, se detallan las principales especificaciones y restricciones que han guiado el desarrollo y puesta en marcha del prototipo.

3.1. Especificaciones funcionales

3.1.1. Adquisición de datos ambientales

El sistema debe ser capaz de recoger datos en tiempo real de variables ambientales clave para la gestión del riego. Estas variables incluyen: temperatura ambiente, humedad relativa del aire y humedad del suelo. Para ello, se han seleccionado sensores específicos: el sensor DHT22 para temperatura y humedad ambiental, y un sensor analógico para humedad del suelo. Estos sensores deben ser compatibles con el microcontrolador ESP32 y proporcionar datos precisos y estables.

3.1.2. Procesamiento y predicción

Utilizando los datos recogidos, el sistema debe ejecutar un modelo de ML que prediga dos parámetros principales: la probabilidad de lluvia y la necesidad de riego. El modelo debe estar optimizado para correr en el microcontrolador, asegurando un equilibrio entre precisión y eficiencia computacional.

3.1.3. Interfaz de usuario

Se debe mostrar al usuario información relevante a través de una pantalla TFT ST7789, incluyendo fecha y hora, temperatura, humedad, probabilidad de lluvia, probabilidad de riego y la decisión final sobre si es necesario regar o no. Esta interfaz debe ser clara, legible y actualizada en tiempo real con un intervalo de aproximadamente un segundo.

3.1.4. Sincronización horaria y programación de evaluaciones

El sistema debe sincronizarse automáticamente con servidores NTP a través de WiFi para obtener la hora precisa y realizar evaluaciones periódicas en horarios predefinidos, específicamente a las 8:00 y 20:00 horas. Esta restricción permite reducir el consumo energético y evitar evaluaciones redundantes durante el día.

3.1.5. Normalización de datos y escalado:

Para que el modelo pueda funcionar correctamente en el microcontrolador, los datos de entrada deben ser normalizados usando el método MinMaxScaler, con valores mínimos y máximos fijados según el rango real de cada variable medido durante la fase de recopilación de datos.

3.1.6. Almacenamiento y formato del modelo:

El modelo de red neuronal entrenado en Python con TensorFlow debe ser convertido a TensorFlow Lite para microcontroladores (TFLite), adaptado para un tamaño reducido y eficiencia en la ejecución. Este modelo debe ser cargado en la memoria del microcontrolador para realizar inferencias en tiempo real.

3.2. Restricciones técnicas

3.2.1. Capacidad limitada del hardware:

El microcontrolador ESP32 posee recursos limitados en términos de memoria RAM, almacenamiento y potencia de procesamiento. Esto impone restricciones severas sobre el tamaño del modelo, la cantidad de operaciones permitidas y la complejidad del algoritmo que puede ejecutarse. Por ello, se ha optimizado el modelo para que contenga un máximo de 10 operaciones y se ha ajustado la memoria tensor arena a 8 KB.

3.2.2. Precisión y ruido en sensores:

Los sensores analógicos y digitales utilizados pueden presentar variaciones y ruido en las lecturas. Para mitigar esto, se ha implementado un sistema de suavizado de lecturas para el sensor de humedad del suelo, mediante un promedio de varias muestras consecutivas. Esto mejora la estabilidad y fiabilidad de los datos que entran al modelo.

3.2.3. Consumo energético:

La operación continua de sensores, pantalla y conectividad WiFi consume energía, por lo que se limita la evaluación del modelo a dos momentos específicos del día y se apaga la conexión WiFi tras sincronizar la hora, para optimizar el consumo.

3.2.4. Conectividad y dependencias externas:

El sistema depende de la disponibilidad de una red WiFi para sincronizar la hora vía NTP. Esto puede limitar su uso en ubicaciones remotas sin cobertura. Sin embargo, una vez obtenida la hora, el sistema puede operar de manera autónoma.

3.2.5. Compatibilidad y soporte de librerías:

La integración de TensorFlow Lite para microcontroladores, junto con las librerías específicas del ESP32, sensores y pantalla integrada, requiere asegurarse de que las versiones sean compatibles y que la memoria y el rendimiento no se vean comprometidos. Además, la complejidad de la configuración puede ser una barrera para usuarios sin experiencia.

3.2.6. Limitaciones del modelo predictivo:

Al ser un modelo supervisado basado en datos recogidos localmente y con variables limitadas, la precisión del sistema está condicionada a la calidad y diversidad de los datos de entrenamiento. Esto puede influir en la capacidad del sistema para generalizar a condiciones diferentes o extremas no contempladas en el conjunto original.

4. Descripción de la solución propuesta

En este capítulo se detalla el proceso de diseño del sistema cuyo propósito principal ha sido el aprendizaje y la aplicación práctica de técnicas de IA y ML en dispositivos de recursos limitados, como los microcontroladores. El objetivo ha sido desarrollar un prototipo funcional que permita la predicción de lluvia y la necesidad de riego a partir de datos recogidos por sensores ambientales. Para ello, se ha utilizado la plataforma Arduino IDE junto con el microcontrolador ESP32, que recopila información de sensores de temperatura, humedad ambiental y humedad del suelo, complementada con registros históricos de lluvia y riego. A lo largo del capítulo se describen las decisiones técnicas adoptadas, el proceso de recopilación y tratamiento de datos, el entrenamiento del modelo, su implementación en el microcontrolador y la evaluación del sistema resultante. También se abordan los principales retos asociados a la ejecución de modelos de IA en hardware limitado, así como las estrategias aplicadas para optimizar su rendimiento y eficiencia.

4.1. Bibliotecas y Herramientas Utilizadas

El sistema desarrollado combina algoritmos de IA con un entorno embebido para automatizar la toma de decisiones sobre el riego agrícola. Para ello, se han utilizado diversas herramientas de software y bibliotecas, tanto para la etapa de entrenamiento del modelo como para su implementación en el microcontrolador.

4.1.1. Python

Python ha sido el lenguaje principal para el desarrollo del modelo de IA. Gracias a su sintaxis clara, su comunidad activa y su ecosistema de librerías, resulta ideal para tareas de análisis de datos, prototipado rápido y entrenamiento de modelos de ML.

Se ha utilizado para el procesamiento de datos, el diseño del modelo y su exportación al formato compatible con sistemas embebidos.

4.1.2. TensorFlow y Keras

Se ha utilizado TensorFlow, biblioteca de código abierto desarrollada por Google, junto con su API de alto nivel Keras, para diseñar y entrenar un modelo de red neuronal multicapa (MLP).

Estas herramientas permitieron:

- Definir la arquitectura del modelo usando la API Sequential.
- Entrenar el modelo con datos normalizados.

- Evaluar su rendimiento sobre datos de validación.

Este modelo aprende a identificar la necesidad de riego en función de variables como temperatura, humedad ambiental, humedad del suelo y otros factores temporales.

4.1.3. Pandas

La biblioteca Pandas ha sido esencial para la manipulación de datos en formato tabular. Se utilizó para:

- Cargar los registros históricos provenientes de AEMET y Copernicus, así como los datos experimentales.
- Procesar columnas, generar variables temporales (como la estación del año) y gestionar valores ausentes.
- Preparar el conjunto de datos para su uso en el entrenamiento del modelo.

4.1.4. NumPy

NumPy ha complementado las tareas de procesamiento con:

- Operaciones vectorizadas sobre arrays.
- Generación de datos simulados (como la humedad del suelo en fases iniciales).
- Cálculo de estadísticas y funciones auxiliares utilizadas en el preprocesamiento y normalización de los datos antes de alimentar la red neuronal.

4.1.5. Scikit-learn

Scikit-learn se empleó para tareas de:

- Preprocesamiento:
    - En el primer modelo, se usó StandardScaler para transformar las variables a media 0 y desviación estándar 1, apropiado para variables heterogéneas como las fechas o temperaturas.
    - En el segundo modelo, se utilizó MinMaxScaler para escalar los valores al rango [0, 1], aprovechando que los datos de entrada tenían rangos naturales acotados.
- División del conjunto de datos: Se separaron los datos en un 80% para entrenamiento y 20% para evaluación.

- Evaluación del modelo: Se utilizaron métricas clásicas como precisión, recall y matriz de confusión para validar el rendimiento antes de exportarlo.

4.1.6. TensorFlow Lite para microcontroladores (TFLite Micro)

Para ejecutar el modelo en el microcontrolador se utilizó TensorFlow Lite para Microcontroladores, una versión optimizada para entornos con recursos limitados como el ESP32.

TensorFlow Lite es un conjunto de herramientas que ayuda a los desarrolladores a ejecutar sus modelos en dispositivos incorporados, móviles o de IoT, y les permite implementar el aprendizaje automático integrado en el dispositivo.[18]

Una vez entrenado y validado el modelo con TensorFlow y Keras, se procede a su conversión mediante TensorFlow Lite, una herramienta diseñada para ejecutar modelos de ML en dispositivos con recursos limitados. TensorFlow Lite permite transformar modelos estándar en un formato más ligero, optimizado para sistemas embebidos como la ESP32.

En este proyecto, TensorFlow Lite se emplea para:

- Reducir el tamaño del modelo para que sea compatible con microcontroladores.
- Realizar inferencias de forma eficiente en tiempo real.
- Integrar la IA directamente en el hardware.

Esta conversión permite ejecutar predicciones en el dispositivo de forma autónoma, sin necesidad de conexión a la nube o a servidores externos.

➔ **Librerías específicas:**

- tflm_esp32.h: Implementación de TensorFlow Lite adaptada al entorno ESP32. Permite la inferencia directamente en el microcontrolador sin conexión a servidores externos.

4.1.7. Librerías para visualización en pantalla TFT ST7789

- Adafruit_GFX.h: Biblioteca base de gráficos, proporciona funciones para mostrar texto, formas y gráficos básicos.
- Adafruit_ST7789.h: Controlador específico para pantallas TFT ST7789. Maneja la comunicación por SPI, inicialización, control de brillo y actualización de la interfaz visual.

Estas librerías permiten mostrar datos de sensores, resultados del modelo y estado del sistema de riego en tiempo real.

4.1.8. Biblioteca para Sensor DHT22

- DHT.h: Biblioteca específica para sensores DHT (DHT11, DHT22, AM2302), permite inicializar el sensor, obtener lecturas de temperatura y humedad, y manejar errores de comunicación.

Este sensor proporciona dos de las variables principales del modelo de predicción: temperatura y humedad ambiental.

4.1.9. Biblioteca para la gestión de conectividad Wi-Fi y hora

- WiFi.h: Biblioteca estándar del entorno ESP32 que permite conectar el dispositivo a una red inalámbrica.
- time.h: Permite obtener y sincronizar la fecha y hora actual desde servidores NTP. Esto es clave para obtener datos temporales (mes, día, hora, minuto), que son también utilizados como entrada para el modelo.

La sincronización temporal permite que el sistema opere de forma autónoma y realice predicciones ajustadas al momento del día.

4.1.10. Biblioteca para sensor de humedad del suelo (capacitivo)

Este sensor capacitivo se conecta a una entrada analógica del ESP32. La lectura se realiza tomando una media de múltiples muestras para reducir el ruido y luego se mapea ese valor al rango 0–100%, representando el porcentaje de humedad del suelo (donde 0% indica suelo completamente húmedo y 100% suelo seco). Finalmente, este valor se normaliza al rango [0, 1] antes de ser utilizado como entrada en el modelo TinyML. No requiere librería específica; la lectura se realiza mediante funciones estándar de entradas analógicas en el entorno Arduino.

4.2. Componentes hardware y sus características

El diseño e implementación del sistema propuesto para la predicción y automatización del riego se fundamenta en una serie de componentes electrónicos, entre los cuales el microcontrolador juega un papel central. Esta sección describe detalladamente el hardware empleado, sus especificaciones técnicas más relevantes y su papel dentro del sistema. La

correcta selección de estos componentes es esencial para garantizar el funcionamiento eficiente, la capacidad de procesamiento y la recolección precisa de datos ambientales y del suelo.

4.2.1. Microcontrolador ESP32 con pantalla TFT ST7789 integrado

El núcleo del sistema está basado en la placa de desarrollo Ideaspark® ESP32 Development Board, que podemos observar en la Figura 15, y que integra un módulo ESP32 de alto rendimiento junto con una pantalla TFT ST7789 de 1,14 pulgadas con resolución 135x240 píxeles. Esta placa representa una solución compacta y versátil para proyectos de Internet de las Cosas (IoT), proporcionando conectividad inalámbrica y una interfaz visual directa para feedback o control.

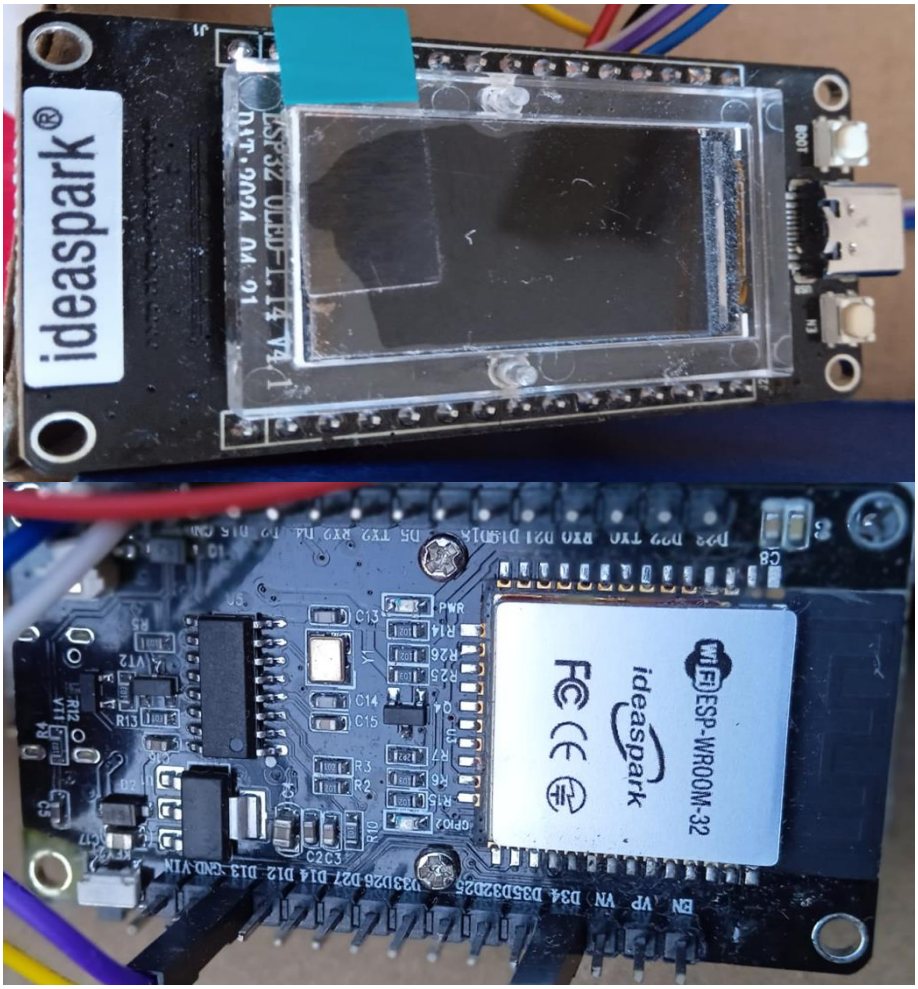


Figura 15. Frontal y trasera del microcontrolador ESP32

➔ **Características técnicas destacadas:**

- Microcontrolador: Módulo ESP32 con procesador Xtensa® 32-bit dual-core LX6 funcionando a hasta 240 MHz, lo que proporciona una potencia suficiente para ejecutar tareas de procesamiento de datos y modelos de ML embebido, como TensorFlow Lite for Microcontrollers.

- Memoria: Incluye 520 KB de SRAM interna y memoria flash integrada (típicamente 4 MB), que permiten almacenar el firmware, el modelo de inferencia y otros datos temporales.
- Conectividad inalámbrica: Wi-Fi de doble banda (2.4 GHz y 5 GHz) y Bluetooth v4.2 (BR/EDR y BLE), lo que facilita la integración con redes locales, dispositivos móviles y servicios en la nube para transmisión o control remoto.
- Pantalla integrada: Pantalla TFT ST7789 de 1,14 pulgadas con resolución 135x240 píxeles, que ofrece una retroalimentación visual clara y compacta, permitiendo mostrar información relevante del sistema (como estados, resultados de predicción o valores de sensores) directamente sin necesidad de un monitor externo.
- Interfaz de conexión: USB Tipo C para programación y alimentación, mejorando la facilidad de uso y compatibilidad con fuentes modernas.
- Convertidor USB a serie: Incorporado mediante el chip CH340, que garantiza una comunicación estable y fiable entre la placa y la computadora para el desarrollo y la depuración.
- Compatibilidad: Compatible con plataformas de desarrollo populares como Arduino IDE y MicroPython, brindando flexibilidad para la programación y experimentación.

➔ **Función en el sistema:**

Esta placa actúa como la unidad central de procesamiento y control del sistema. Recolecta datos de los sensores conectados (temperatura, humedad ambiente y humedad del suelo), procesa esta información ejecutando el modelo entrenado en TensorFlow Lite para determinar cuándo es necesario activar el riego. La pantalla TFT facilita la visualización inmediata de datos y estados, aumentando la usabilidad durante las pruebas y despliegues. Gracias a su conectividad Wi-Fi y Bluetooth, es posible enviar datos para monitoreo remoto o integración con aplicaciones externas, expandiendo la funcionalidad del sistema a plataformas IoT.

4.2.2. Sensor de temperatura y humedad DHT22 (AM2302)

El sensor DHT22, también conocido como AM2302, es un sensor digital que permite medir temperatura y humedad de forma sencilla. Es ampliamente utilizado en aplicaciones de automatización y sistemas de monitorización ambiental debido a su bajo coste, facilidad de uso y buena precisión.

Figura 16. Frontal y trasera del sensor de temperatura y humedad DHT22

En este proyecto, se ha empleado para obtener datos ambientales que luego se utilizan como entrada para el modelo de IA ejecutado en el ESP32.

➔ **Características principales:**

- Salidas digitales: La señal se transmite mediante un único pin de datos digital, facilitando su conexión al microcontrolador.
- Alimentación: Funciona directamente con 3.3 V, por lo que es totalmente compatible con el ESP32 sin necesidad de conversión de niveles.
- Precisión suficiente: Ofrece una precisión de ±0.5 °C para temperatura y ±2 % RH para humedad, adecuada para aplicaciones de control básico.
- Resolución de 16 bits, lo que permite una representación adecuada de los datos sin necesidad de sensores más avanzados.

➔ **Conexión al ESP32:**

El sensor ha sido conectado al pin GPIO 13 del ESP32. La conexión se ha realizado de la siguiente forma:

- Vcc: conectado al pin de 3.3 V del ESP32
- GND: conectado a tierra (GND)

- Datos: conectado al pin digital GPIO 13 del ESP32

Además, se ha incorporado una resistencia pull-up de 10 kΩ entre el pin de datos y Vcc. Esta resistencia es necesaria para garantizar una señal estable, ya que el protocolo de comunicación del DHT22 es de tipo unidireccional (un solo hilo).

➔ **Función dentro del sistema:**

El DHT22 se encarga de proporcionar las variables ambientales de entrada —temperatura y humedad— que permiten al sistema de inferencia tomar decisiones en función de las condiciones del entorno. Estos datos son también los que se usaron para entrenar el modelo de clasificación. La fiabilidad de este sensor es clave para mantener la coherencia entre los datos del entrenamiento y las condiciones reales en las que el modelo se ejecuta.

4.2.3. Sensor de humedad del suelo (capacitivo)

El sensor de humedad del suelo utilizado lo vemos en la Figura 17, es de tipo capacitivo, y es una opción más avanzada frente a los sensores resistivos tradicionales. Este tipo de sensor permite medir la humedad del sustrato de forma más precisa y fiable, ya que no utiliza electrodos expuestos, evitando así la corrosión y aumentando considerablemente su vida útil.

Figura 17. Frontal y trasera del sensor capacitivo de humedad del suelo

Este sensor forma parte de las entradas analógicas del sistema y proporciona información crítica sobre el estado hídrico del suelo, lo cual es fundamental tanto para la recopilación de datos como para la inferencia realizada por el modelo de IA.

➔ **Características principales:**

- Medición capacitiva de humedad: Detecta variaciones en la constante dieléctrica del suelo en lugar de la conductividad, lo que permite mayor precisión y durabilidad.
- Alimentación: Opera a 3.3 V, lo que garantiza compatibilidad directa con el ESP32 sin necesidad de conversores de voltaje.
- Salida analógica: Proporciona un voltaje proporcional al nivel de humedad (entre 0 y 3.0 V).
- Diseño resistente: Su construcción evita la oxidación, lo que lo hace ideal para entornos de uso prolongado o en exteriores.

➔ **Conexión al ESP32:**

El sensor se ha conectado al pin GPIO 34, uno de los pines ADC del ESP32, que permite la lectura de señales analógicas. La conexión se ha realizado de la siguiente manera:

- Vcc: conectado al pin de 3.3 V del ESP32
- GND: conectado a tierra (GND)
- OUT (señal): conectado al pin analógico GPIO 34 del ESP32

No es necesario añadir resistencias externas, ya que el sensor incluye un chip estabilizador de voltaje que asegura un funcionamiento estable dentro del rango especificado.

➔ **Función dentro del sistema:**

Este sensor capacitivo permite detectar los niveles de humedad del sustrato y actúa como una de las variables de entrada del modelo de clasificación, junto con temperatura y humedad ambiental. Los datos recogidos durante la fase experimental se han utilizado para entrenar el modelo, y posteriormente, el sensor continúa proporcionando datos en tiempo real. Su estabilidad y precisión son clave para garantizar la fiabilidad del sistema en condiciones reales.

4.3. Transmisión y Adquisición de Datos

La recopilación de datos es muy importante en el entrenamiento de una red neuronal, debido a que su precisión dependerá de las condiciones atmosféricas y tipo de suelo en que se encuentre nuestro sistema.

Inicialmente se obtuvieron datos de la página de AEMET: temperatura máxima, mínima y media, y la correspondiente hora a la que se registraron esas temperaturas máximas y mínimas, humedad ambiente máxima, mínima y media y las correspondientes horas en las que se registraron los picos, horas de sol, precipitación y de la página de Copernicus se ha obtenido la Humedad del suelo de la zona más cercana en la que nos encontramos. Podemos ver en la Tabla 1 el tipo de formato de los datos recogidos.

Fecha	**Tmax**	**Hora_Tmax**	**Tmin**	**Hora_Tmin**	**Tmed**	**HrMax**	**Hora_HrMax**	**HrMin**	**Hora_HrMin**	**HrMedia**	**Horas_Sol**	**Precipitacion**	**Necesidad_riego**	Humedad_Suelo
01/09/2023	32,8	15:04	15,9	4:57	24,4	54	23:47	22	0:00	31	9	0	1	27,4307
02/09/2023	21,5	0:00	16	4:57	18,8	96	14:00	52	0:08	71	0,4	28,1	0	27,2274
03/09/2023	21,3	22:17	16	0:35	18,6	98	14:57	80	22:08	94	0	36,8	0	27,024
04/09/2023	24,1	17:32	15,8	4:45	20	98	1:30	63	17:34	80	4,6	4,4	0	26,8207
05/09/2023	28,4	15:48	16,3	6:03	22,4	98	6:34	42	15:52	68	7,8	0	0	26,6173
06/09/2023	29,6	16:18	19,3	5:19	24,4	88	3:54	37	16:30	59	6,8	0	0	26,414

Tabla 1. Detalle de recopilación de datos para el primer modelo

Al no tener en un principio datos históricos de la necesidad de riego, para simularla, se ha usado la siguiente fórmula, evaluando si se cumplen ciertas condiciones climáticas:

SI(Y (Tmax >25; HrMax < 60; Precipitacion ≤ 2; Humedad_Suelo < 30); 1; 0) es decir Si(condición_logica; valor_si_verdadero;valor_si_falso). La función Y(…) devuelve VERDADERO si todas las condiciones dentro de ella se cumplen.

Condiciones dentro de Y(…)

- Tmax > 25

Esto sugiere que hace bastante calor, lo que podría aumentar la evaporación del agua en el suelo.

- HrMax <60%

Una humedad menor implica que el ambiente está seco, lo que también favorece la pérdida de agua del suelo.

- Precipitación <=2

Es decir, ha llovido poco o nada, por lo que el suelo no ha recibido suficiente agua.

- Humedad_Suelo < 30

Umbral bajo, indica sequedad crítica.

Si todas estas condiciones se cumplen:

- Hace calor (Tmax > 25)
- El ambiente está seco (HrMax <60)
- Ha llovido muy poco o nada (Precipitación <= 2 mm)
- El suelo está bastante seco (Humedad_Suelo < 30)

entonces la fórmula devuelve un 1, indicando que sí hay necesidad de riego.

Si alguna condición no se cumple, devuelve 0, indicando que no es necesario regar.

Esta fórmula es una regla de decisión para activar o no el riego, a falta de datos históricos de riego, como se ha comentado anteriormente, basada en tres factores climáticos clave: temperatura, humedad del aire y del suelo y lluvia.

Para el desarrollo del modelo final destinado a predecir la necesidad de riego y la ocurrencia de lluvia, se realizó un proceso riguroso de recopilación, organización y preparación de datos. La calidad y relevancia de los datos influyen directamente en la precisión y robustez del modelo de ML, por lo que esta etapa es fundamental dentro del proyecto.

4.3.1. Adquisición de datos

La adquisición de datos se realiza a través de los dos sensores conectados al microcontrolador.

La lectura de los sensores se hace periódicamente (cada segundo para mostrar los valores en pantalla, y cada hora para registrar datos). En cada lectura:

- Se recoge la temperatura en grados Celsius.
- Se recoge la humedad relativa del aire en porcentaje.
- Se lee la humedad del suelo y se convierte a un valor porcentual, usando un rango calibrado entre valores mínimos y máximos detectados al medir con el sensor al aire o empapado.

Además, el sistema sincroniza la hora mediante NTP a través de WiFi, para asignar un sello temporal exacto a cada registro.

4.3.2. Almacenamiento y transmisión de datos

Cada hora, el sistema guarda los datos recogidos en dos sitios:

- Archivo CSV en memoria SPIFFS

SPIFFS significa SPI Flash File System. Es un sistema de archivos que se usa en microcontroladores como el ESP32 para almacenar datos de manera persistente en la memoria flash interna del dispositivo.

Se crea y actualiza un archivo llamado data.csv donde se almacenan las lecturas con la siguiente estructura de columnas:

Ano,Mes,Dia,Hora,Minuto,Temp,Hum_Amb,Hum_Suelo,Precipitacion,Lluvia,Riego

En cada registro, se guardan la fecha y hora actuales, la temperatura, la humedad ambiental y la humedad del suelo. Los campos de Precipitación, Lluvia y Riego quedan reservados para anotaciones manuales o posteriores.

Esto permite tener un historial local, accesible si se conecta el dispositivo por USB o mediante comandos serie.

- Monitor Serie

  Al mismo tiempo que se guarda en el archivo, los datos se imprimen por el puerto serie en formato CSV, para facilitar su captura mediante software externo.

El formato impreso, lo podemos ver en la Figura 18, es una línea CSV, con los mismos campos que el archivo:

2025,7,2,15,0,32.4,48,90.3,?,?,?

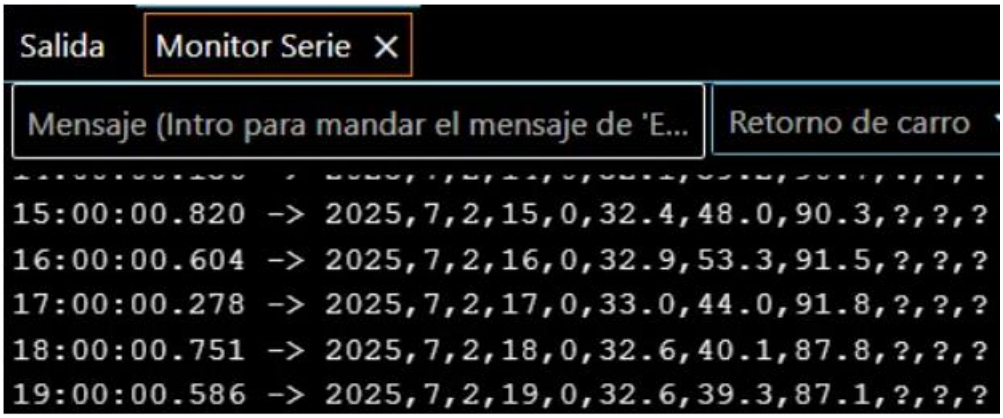


Figura 18. Detalle de la impresión por Monitor Serie

Esta línea indica, por ejemplo, que el 2 de julio de 2025 a las 15:00, la temperatura era 32.4°C, la humedad ambiental 48%, y la humedad del suelo 90.3%. Los signos de interrogación muestran que la información sobre lluvia y riego debe ser completada manualmente.

4.3.3. Uso para modelo final

Los datos obtenidos y guardados en CSV, como podemos apreciar en la Tabla 2 se transfieren luego a un ordenador para su análisis y procesamiento.

La manera más cómoda aunque tediosa de recopilar datos de manera fiable ha sido copiando a mano a excel desde puerto serie y rellenando cada poco tiempo para no perder detalle de lluvia o riego. La columna precipitación que en un principio se iba a completar con datos de la página de AEMET, se descarta, ya que no tenemos un sensor para ello.

01:25:00.618 -> Ano	Mes	Dia	Hora	Minuto	Temp	Hum_Amb	Hum_Suelo	Precipitacion (AEMET)	Lluvia	Riego
20:00:01.061 -> 2025	6	16	20	0	**30.6**	**48.8**	**80.2**	**0**	**1**	**0**
21:00:00.894 -> 2025	6	16	21	0	**30.6**	**46.2**	**80.1**	**0**	**0**	**0**
22:00:00.938 -> 2025	6	16	22	0	**30.7**	**43.2**	**78.9**	**0**	**0**	**0**
23:00:00.787 -> 2025	6	16	23	0	**30.6**	**41.8**	**79.4**	**0**	**0**	**0**
00:00:00.936 -> 2025	6	17	0	0	**30.4**	**47.7**	**91.5**	**0.2**	**0**	**1**

Tabla 2. Detalle de recopilación de datos para el modelo final

Estos datos completos se exportan para alimentar el modelo final de ML, que usa los valores temporales y ambientales para aprender patrones de cuándo regar y cuándo no.

El modelo generado se convierte a TensorFlow Lite y se integra en el firmware final del ESP32, que lee los sensores en tiempo real y predice automáticamente la necesidad de riego.

Se puede observar en la Figura 19 la representación de los datos en la pantalla. Esta interfaz visual es crucial para la monitorización en tiempo real del sistema, mostrando de forma concisa toda la información relevante:

- Línea 1: Fecha y hora sincronizada vía NTP.
- Líneas 2 y 3: Lecturas en tiempo real de los sensores DHT22 (temperatura y humedad ambiental) y del sensor capacitivo (humedad del suelo).
- Línea 4: Predicciones del modelo de IA para la probabilidad de lluvia y riego.
- Línea 5: Decisión final del sistema ("Regar ahora" o "No regar"), que se discute en un apartado posterior.

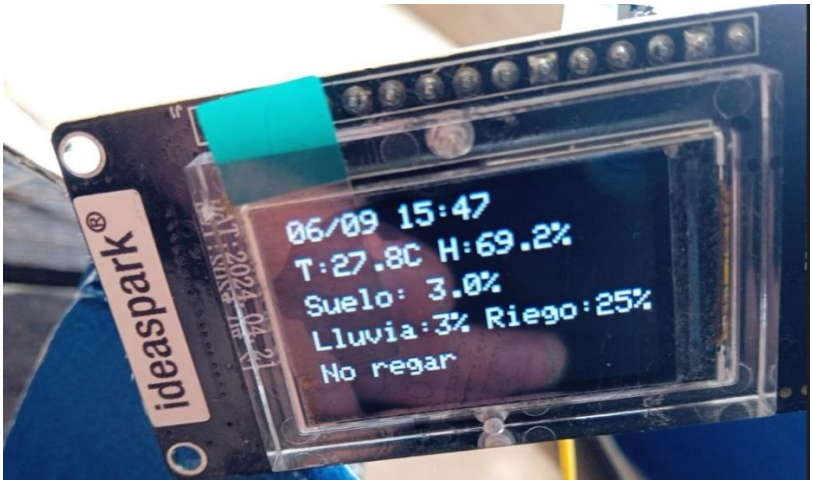


Figura 19. Visualización pantalla

Anteriormente se ha detallado la selección de los componentes de hardware y la lógica de su interconexión. A continuación, en la Figura 20 se muestra la implementación física de estos elementos, conformando el prototipo final utilizado para la validación del sistema.

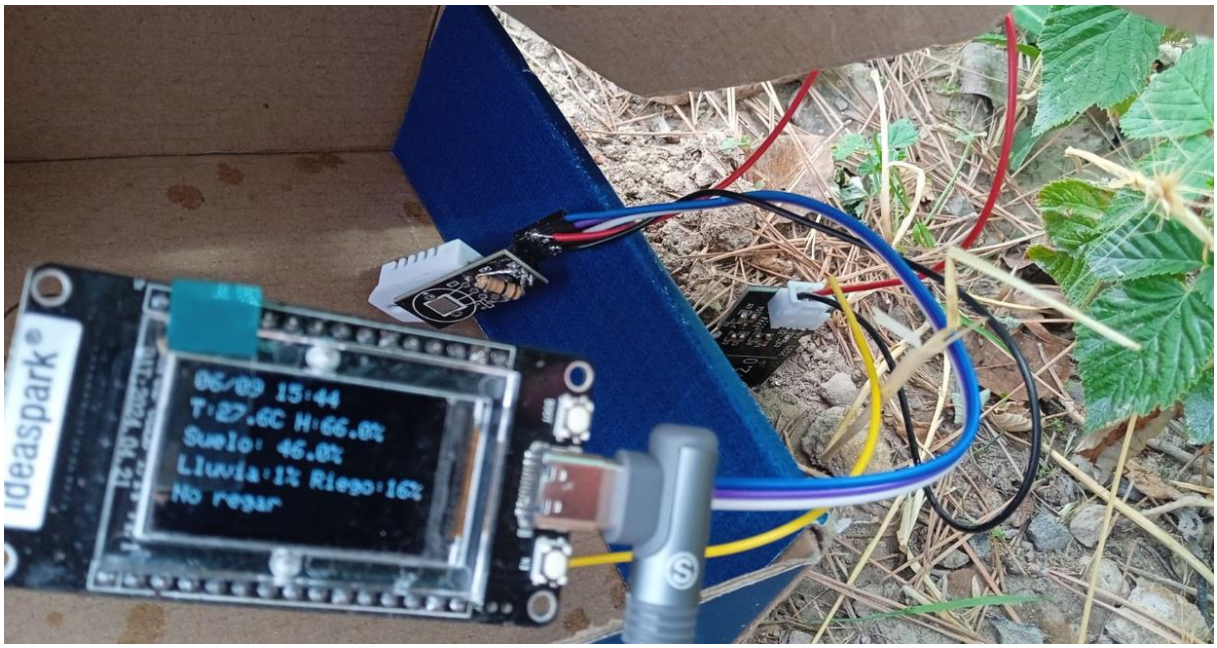


Figura 20. Prototipo del sistema de riego

Vemos el ensamblaje del microcontrolador ESP32 con el sensor de temperatura y humedad DHT22 y el sensor de humedad del suelo. La disposición de los componentes en la protoboard facilita la interconexión y demuestra la viabilidad de la arquitectura diseñada.

4.4. Arquitectura general del sistema

La arquitectura del sistema riego inteligente se concibe como una solución embebida, diseñada para operar de manera autónoma y eficiente en un entorno con recursos limitados. Se representa mediante un diagrama de bloques funcional, representado en la Figura 21, que ilustra la interconexión y el flujo de datos entre sus componentes principales.

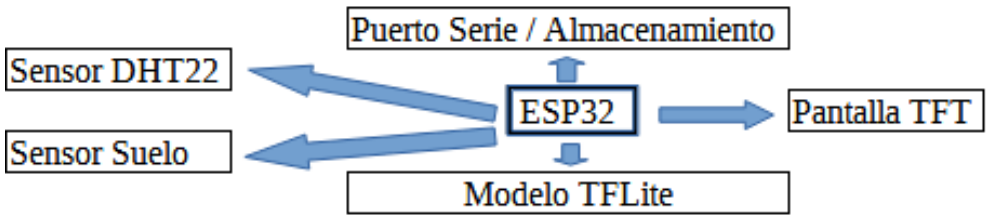


Figura 21. Arquitectura del Sistema

Cada bloque en el diagrama representa una unidad funcional específica que contribuye al objetivo global del sistema. El microcontrolador ESP32 actúa como el núcleo central de la solución, integrando y coordinando la adquisición de datos, la visualización de información, el análisis de las condiciones ambientales y la inferencia del modelo de ML para la toma de decisiones.

A continuación, se detalla la función de cada uno de los bloques y su rol en la operación general del sistema:

- Sensor DHT22: Este componente es responsable de la adquisición de datos ambientales clave: la temperatura y la humedad del aire. Proporciona información esencial sobre las condiciones atmosféricas que influyen en la evaporación y la transpiración de las plantas, datos que son fundamentales para las predicciones del modelo.

- Sensor de Humedad del Suelo: Su función principal es la monitorización directa de la humedad del sustrato. Envía lecturas al microcontrolador, permitiendo evaluar el nivel de sequedad o saturación de la tierra. Este dato es el más crítico para determinar la necesidad de riego.

- ESP32 (Microcontrolador): Constituye el cerebro del sistema. Es el encargado de:

    - Adquisición y preprocesamiento de datos: Recopila las lecturas de ambos sensores, aplicando suavizado si es necesario.

    - Sincronización horaria: Gestiona la conexión Wi-Fi para obtener la hora precisa a través de NTP, lo cual es vital para las variables temporales del modelo y la programación de evaluaciones.

    - Inferencia del modelo: Ejecuta el modelo de ML (TensorFlow Lite) cargado en su memoria, procesando los datos de los sensores para predecir la probabilidad de lluvia y la necesidad de riego.

    - Lógica de decisión: Basándose en las predicciones del modelo y en la lectura directa del sensor de lluvia (como mecanismo de seguridad), decide si activar o no el sistema de riego.

    - Control de periféricos: Gestiona la salida de datos hacia la pantalla TFT y, en una implementación completa, controlaría los actuadores de riego.

    - Visualizar por pantalla: Proporciona una interfaz de usuario local para el sistema. Permite visualizar en tiempo real las lecturas de los sensores (temperatura, humedad ambiental y del suelo), la hora actual, las probabilidades de predicción del modelo y la decisión final sobre el riego ("Regar ahora" o "No regar"). Esto facilita el monitoreo directo del sistema sin necesidad de una conexión externa.

- Modelo TFLite: Este es el componente de IA embebida. Representa el modelo de red neuronal entrenado en Python y optimizado para microcontroladores. Reside en la memoria del ESP32 y es el encargado de realizar las inferencias (predicciones de lluvia y riego) a partir de los datos normalizados de los sensores. Su eficiencia en recursos es clave para el propósito de este proyecto.

- Puerto Serie / Almacenamiento: Este bloque representa la interfaz para la depuración del sistema y el registro de datos. A través del puerto serie, se pueden visualizar en tiempo real las lecturas de los sensores, las probabilidades de predicción del modelo y las decisiones tomadas, lo cual es fundamental durante las fases de desarrollo y prueba. Además, permite el almacenamiento persistente de datos en la memoria SPIFFS del ESP32 para un historial local.

La interacción entre estos bloques sigue un ciclo continuo: los sensores adquieren datos del entorno, el ESP32 los procesa y los alimenta al Modelo TFLite para la inferencia. Las predicciones resultantes informan la lógica de decisión del ESP32, que a su vez gestiona la visualización en la pantalla TFT y la activación de posibles actuadores de riego, mientras se registran los datos para análisis. Esta arquitectura refleja un diseño robusto y eficiente para un sistema de riego inteligente basado en IA en el borde.

4.5. Justificación de decisiones de diseño

Las decisiones de diseño tomadas a lo largo del proyecto se han fundamentado en una serie de criterios clave que buscan optimizar la funcionalidad, la eficiencia y la viabilidad del sistema en el contexto de la agricultura de precisión. A continuación, se justifican las principales elecciones:

- Hardware económico y abierto: permite replicabilidad y adaptación. La elección de componentes como el microcontrolador ESP32 y los sensores DHT22 y capacitivo de suelo se basa en su bajo coste y la naturaleza de hardware abierto. Esta decisión es crucial para el objetivo de promover soluciones accesibles y sostenibles en el sector agrícola. Un sistema económico facilita su adopción por parte de pequeños agricultores o entusiastas, y la disponibilidad de diseños abiertos fomenta la replicabilidad, la adaptación a necesidades específicas y la innovación comunitaria. Esto contrasta con soluciones propietarias de alto coste, haciendo la tecnología de IA embebida más democrática.

- Visualización local: evita necesidad de interfaz web o móvil. La elección de una placa ESP32 con pantalla TFT ST7789 integrada responde a la necesidad de proporcionar una interfaz de usuario inmediata y autónoma. Esta visualización local de los datos de los sensores, las probabilidades de predicción y las decisiones de riego evita la dependencia de una conexión a internet constante, de una aplicación móvil o de una interfaz web externa. Esto simplifica la operación del sistema en entornos rurales o con conectividad limitada, ofreciendo retroalimentación en tiempo real de manera directa y sin complejidades adicionales para el usuario final.

- Modelo de IA embebido: evita dependencia de la nube. La decisión de entrenar y convertir el modelo de ML a TensorFlow Lite para su ejecución directa en el microcontrolador (IA en el borde) es fundamental. Esto permite que las inferencias se

realicen localmente en el ESP32, eliminando la necesidad de enviar datos a la nube o a servidores remotos para su procesamiento. Las ventajas son múltiples:

  - Menor latencia: Las decisiones se toman casi instantáneamente, sin retrasos por la comunicación de red.
  - Mayor privacidad: Los datos de los sensores no necesitan salir del dispositivo.
  - Menor consumo energético: Se reduce el uso de la conectividad inalámbrica, que es un componente de alto consumo.
  - Robustez: El sistema puede operar de forma autónoma incluso sin conexión a internet, lo que es vital en entornos agrícolas remotos.

- Diseño modular: facilita ampliaciones futuras (riego automático, conexión a base de datos, etc.). La arquitectura del sistema se ha concebido con un diseño modular, donde cada componente (sensores, microcontrolador, pantalla, modelo de IA) opera como una unidad funcional bien definida. Esta modularidad es una decisión estratégica que facilita enormemente las futuras ampliaciones y mejoras del sistema. Por ejemplo, la integración de un actuador de riego automático, la conexión a una base de datos para el registro a largo plazo de datos, o la incorporación de nuevos tipos de sensores, se pueden añadir con un impacto mínimo en la estructura existente. Esto asegura la escalabilidad y la adaptabilidad del proyecto a medida que surjan nuevas necesidades o se disponga de más recursos.

4.6. Aportaciones propias y recursos externos

En este proyecto, se ha combinado el uso de herramientas y recursos existentes con un trabajo propio significativo de diseño, integración y optimización. A continuación, se explica qué partes constituyen una contribución original y cuáles son los recursos externos que se han utilizado.

4.6.1. Aportaciones Propias

Las siguientes áreas representan las contribuciones originales y el valor añadido generado en el marco de este Proyecto Fin de Grado:

- Integración y Puesta en Marcha del Sistema Completo (Sensores + Pantalla + Predicción): Se ha logrado la integración funcional de todos los elementos del sistema para que trabajen de forma cohesionada. Esto incluye la conexión de los sensores (temperatura, humedad ambiental y humedad del suelo), el microcontrolador ESP32 (que incorpora una pantalla), el modelo de IA desarrollado, y toda la lógica de control que permite al sistema tomar decisiones. La aportación reside en el diseño de la interacción y el funcionamiento conjunto de estos componentes para crear un sistema de riego inteligente y autónomo. Este proceso abarca desde la adquisición y procesamiento de los datos de los sensores en el

propio dispositivo, la realización de predicciones con el modelo, la toma de decisiones sobre el riego, hasta la visualización de la información en la pantalla integrada.

- Diseño y Optimización de la Lógica de Procesamiento y Toma de Decisiones en el Microcontrolador: Aunque se emplean librerías estándar para la comunicación con los sensores y la gestión del modelo de IA, la lógica de control específica implementada en el firmware del ESP32 es una aportación propia. Esto comprende:
    - Preparación de los datos de los sensores: Se ha definido el método de normalización de los datos (ajuste a una escala de 0 a 1) directamente en el ESP32, utilizando los valores calculados durante el entrenamiento en el ordenador. De esta manera, el modelo interpreta los datos de entrada de forma consistente.
    - Programación de las evaluaciones del modelo: Se ha programado que el sistema realice predicciones en momentos clave del día (por ejemplo, a las 8 de la mañana y a las 8 de la tarde) para optimizar el consumo de energía y la eficiencia operativa.
    - Mejora en la lectura de la humedad del suelo: Se ha implementado una técnica de promedio de múltiples lecturas del sensor de humedad del suelo para obtener un valor más estable y reducir el ruido.
    - Definición de la lógica de decisión de riego: Se ha creado la regla que el sistema sigue para decidir si regar. Esta regla combina la predicción del modelo sobre la necesidad de riego con la información sobre la presencia de lluvia (utilizando un sensor de lluvia si estuviera disponible, o asumiendo ausencia de lluvia si el modelo de predicción de lluvia no es fiable). Esto asegura que no se active el riego si ya está lloviendo.
- Adaptación y Optimización del Modelo de IA para el ESP32: Aunque se han utilizado herramientas estándar (TensorFlow/Keras) para la creación del modelo predictivo, una parte fundamental del trabajo ha sido su adaptación y optimización para funcionar de manera eficiente en un microcontrolador como el ESP32. Para ello:
    - Se ha seleccionado una arquitectura de red neuronal (con 2 capas ocultas, la primera con 16 neuronas, la segunda con 8 neuronas, y una capa de salida, con 2 neuronas para las 2 salidas) que se ajusta a las limitaciones de memoria y capacidad de cálculo del ESP32.
    - Se ha convertido y optimizado el modelo a un formato especial (.tflite) que es más ligero y rápido para dispositivos embebidos.
    - Se ha logrado la integración exitosa de este modelo optimizado dentro del firmware del ESP32, empleando librerías específicas que permiten realizar predicciones en tiempo real directamente en el dispositivo.

4.6.2. Recursos Externos y Herramientas Utilizadas

Para la construcción de este proyecto, se ha recurrido a una variedad de herramientas y recursos existentes, de código abierto o gratuitos, que han sido esenciales para su desarrollo:

- Hardware Estándar y Económico:
    - Microcontrolador ESP32: Plataforma popular y económica, con capacidades Wi-Fi y Bluetooth, idónea para proyectos de este tipo.
    - Sensores (DHT22, capacitivo de suelo): Componentes comerciales de bajo coste para la medición de parámetros ambientales.
    - Pantalla TFT ST7789: Una pantalla común que, en este caso, ya venía integrada en la placa ESP32, lo que simplificó el montaje.
- Datos para Entrenamiento de Acceso Público:
    - Datos históricos de AEMET y Copernicus: Se ha utilizado información meteorológica y de humedad del suelo disponible públicamente, lo que ha contribuido a comprender los patrones relevantes.
    - Datos recogidos por los propios sensores del prototipo: Se han empleado los datos recopilados por el prototipo en funcionamiento, lo cual es crucial para que el modelo aprenda de las condiciones reales del sistema.
- Programas y Librerías de Código Abierto:
    - Python y sus librerías: Se ha utilizado Python para el desarrollo del modelo, con librerías como pandas (para manipulación de datos), NumPy (para cálculos numéricos) y scikit-learn (para preprocesamiento y evaluación).
    - TensorFlow y Keras: Son los frameworks principales de Google para la creación y entrenamiento de modelos de IA.
    - TensorFlow Lite para Microcontroladores (TFLite Micro): Una versión especializada de TensorFlow que permite la ejecución de modelos de IA en dispositivos con recursos limitados.
    - Librerías de Arduino: Se han empleado librerías estándar para la comunicación con el hardware, incluyendo Adafruit_GFX.h, Adafruit_ST7789.h (para la pantalla), DHT.h (para el sensor DHT22), y WiFi.h y time.h (para conectividad y sincronización horaria).
    - Librerías específicas de TinyML: tflm_esp32.h es librería que ha facilitado la integración del modelo de IA en el ESP32.

Al combinar todas estas herramientas y recursos externos con el diseño y la integración propios, se ha conseguido crear un sistema funcional e inteligente. Esto demuestra la viabilidad de aplicar la IA en dispositivos embebidos para resolver problemas prácticos y contribuir a la sostenibilidad ambiental.

5. Resultados

5.1. Metodología de Pruebas

Para validar el correcto funcionamiento del sistema propuesto, se realizaron pruebas estructuradas en varias fases, cubriendo tanto el hardware como el software y el modelo predictivo.

5.1.1. Pruebas del Sistema Físico

Se verificó la funcionalidad del hardware y el firmware en condiciones operativas. Estas pruebas incluyeron:

- Medición de Sensores: Comprobación de que el sensor DHT22 (temperatura y humedad ambiental) y el sensor capacitivo de humedad del suelo proporcionaban lecturas estables y coherentes con las condiciones ambientales.
- Visualización en Pantalla: Verificación del correcto funcionamiento de la pantalla TFT ST7789 integrada, asegurando que mostraba la hora y los valores de los sensores actualizados en tiempo real.
- Registro de Datos: Confirmación de que el sistema registraba correctamente los datos en el archivo CSV en la memoria SPIFFS y los enviaba por el monitor serie, siguiendo la estructura definida.
- Estabilidad Operativa: Monitoreo del microcontrolador durante sesiones prolongadas para detectar posibles bloqueos o reinicios inesperados.
- Sincronización Horaria: Verificación de la correcta conexión Wi-Fi y sincronización horaria mediante NTP para asegurar la precisión de las variables temporales.
- Lógica de Evaluación Programada: Comprobación de que la inferencia del modelo se activaba solo en los horarios predefinidos y con el intervalo mínimo establecido, optimizando el uso de recursos.

5.1.2. Metodología de Evaluación del Modelo Predictivo

La evaluación del modelo de ML se realizó utilizando un conjunto de datos de prueba (20% del conjunto de datos total) que no fue utilizado durante el entrenamiento. Esto garantizó que las métricas de rendimiento reflejaran la capacidad del modelo para generalizar a datos no vistos. Las métricas clave utilizadas para ambas salidas (Lluvia y Riego) fueron:

- Precisión (Accuracy): Porcentaje de predicciones correctas sobre el total de predicciones.

- Reporte de Clasificación: Incluye precision, recall y f1-score para cada clase (0 y 1), así como el support (número de instancias reales de cada clase en el conjunto de prueba). Estas métricas son cruciales para evaluar el rendimiento en clases desbalanceadas.

- Análisis de Errores (Falsos Positivos y Falsos Negativos): Se examinaron los tipos de errores que el modelo cometía para comprender mejor su comportamiento. Los umbrales de decisión para clasificar las probabilidades de salida en binario (0 o 1) se fijaron en 0.50 para ambas predicciones.

5.2. Resultados Obtenidos

5.2.1. Pruebas del Sistema Físico

- Lecturas ambientales: Durante las pruebas de funcionamiento, los sensores integrados en el sistema proporcionaron lecturas consistentes y dentro de los rangos esperados para un entorno de monitorización ambiental. La estabilidad de las mediciones fue clave para asegurar la fiabilidad de los datos de entrada al modelo de inferencia.

- Registro y visualización:

    - El sistema demostró una correcta capacidad de registro de datos, almacenando una línea de información por cada hora en el archivo CSV persistente en la memoria SPIFFS del microcontrolador. Esta funcionalidad es esencial para el posterior análisis del comportamiento del sistema y la recopilación de datos para futuras mejoras.

    - La pantalla TFT ST7789 integrada operó de manera fluida, mostrando la hora y los valores actualizados de los sensores cada segundo sin latencia perceptible o artefactos visuales. Esto valida la interfaz de usuario local como un medio eficaz para el monitoreo en tiempo real.

    - Durante las sesiones de operación prolongadas, no se registraron bloqueos inesperados, reinicios del microcontrolador ni fallos en la comunicación serie. Esta estabilidad operativa es fundamental para la fiabilidad de un sistema embebido autónomo.

5.2.2. Resultados del Primer Modelo Predictivo

Los resultados de la evaluación del modelo en el conjunto de prueba de la Figura 22, utilizando las métricas de clasificación estándar, se detallan a continuación.

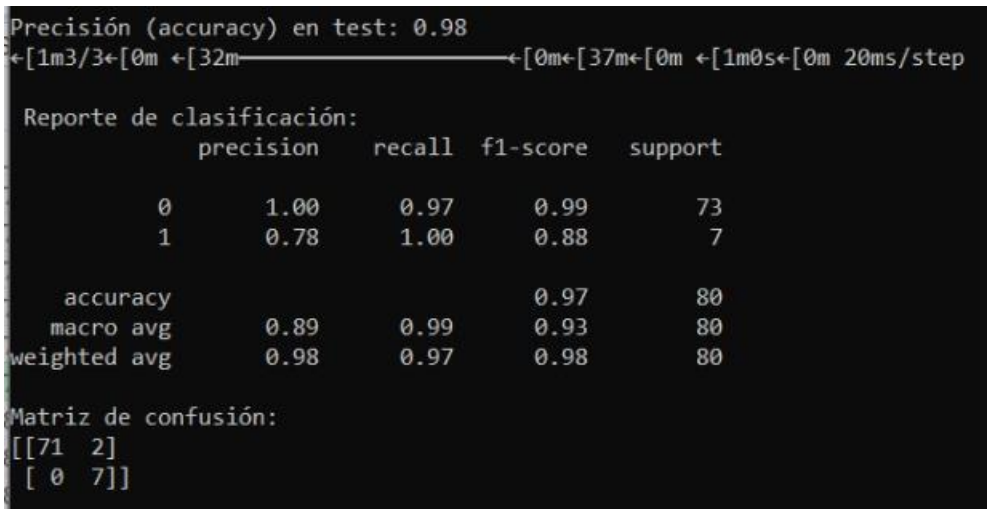


Figura 22. Captura de resultados primer modelo predictivo de riego

El modelo de predicción de necesidad de riego mostró un rendimiento muy sólido en el conjunto de prueba, alcanzando una precisión (accuracy) del 0.97. El reporte de clasificación detalla el desempeño para cada clase:

- Para la clase negativa ("No Riego"), el modelo obtuvo una precisión de 1.00, un recall de 0.97 y un F1-score de 0.99 sobre un total de 73 muestras. Esto indica que el modelo identifica casi perfectamente los casos donde no era necesario regar, con muy pocos falsos positivos.

- Para la clase positiva ("Riego"), el modelo presentó una precisión de 0.78, un recall perfecto de 1.00 y un F1-score de 0.88 en 7 muestras. Esto significa que el modelo fue capaz de detectar correctamente todos los casos reales donde se necesitaba riego (recall = 100%), aunque hubo algunas predicciones erróneas de riego cuando no ocurrió (precisión = 78%).

- El valor global del accuracy fue del 0.97 con 80 muestras evaluadas, mientras que los promedios macro y ponderado del F1-score fueron de 0.93 y 0.98 respectivamente, mostrando un equilibrio adecuado entre la capacidad para evitar falsos negativos y falsos positivos.

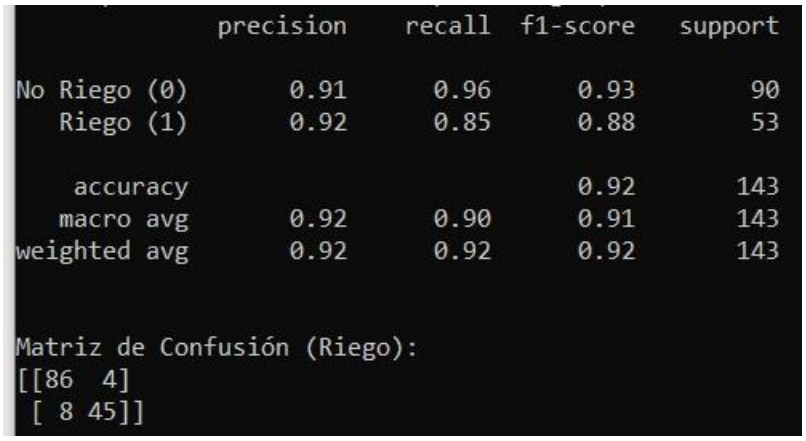
Estos resultados iniciales reflejan que el modelo es especialmente fiable para identificar eventos de necesidad de riego, lo cual es crucial para el sistema, ya que minimizar falsos negativos (no detectar la necesidad real de riego) es más importante para evitar el estrés hídrico en los cultivos. La ligera tendencia a sobreestimar la necesidad de riego (falsos positivos) es menos problemática, ya que podría provocar una pequeña sobrerregulación, preferible a no regar cuando es necesario.

5.2.3. Resultados modelo final

El modelo inicial fue entrenado con los datos recogidos durante un mes, con una frecuencia de una hora. Los resultados de la evaluación para la predicción de lluvia fueron los siguientes:

- **Predicción de la necesidad de riego**

El rendimiento del modelo en la predicción de la necesidad de riego es robusto y altamente aceptable para una aplicación práctica. Se comprueba en la Figura 23.



	precision	recall	f1-score	support
No Riego (0)	0.91	0.96	0.93	90
Riego (1)	0.92	0.85	0.88	53
accuracy			0.92	143
macro avg	0.92	0.90	0.91	143
weighted avg	0.92	0.92	0.92	143

Matriz de Confusión (Riego):
[[86 4]
[8 45]]

Figura 23. Captura de resultados modelo final predicción de riego

Análisis e Interpretación:

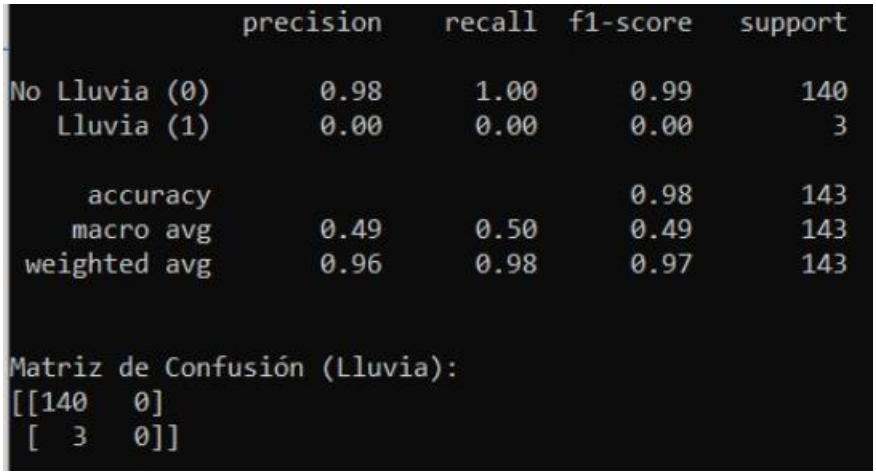
- Alta Precisión en la Decisión de Riego: Con una precisión del 0.92 para la clase "Riego (1)", el modelo demuestra que cuando predice que se necesita regar, es correcto el 92% de las veces. Esto es excelente, ya que minimiza el desperdicio de agua y el riesgo de sobrerriego.
- Buena Capacidad de Detección: El recall del 0.85 para "Riego (1)" indica que el modelo es capaz de identificar correctamente el 85% de las veces que realmente se necesita riego. Esto significa que la planta recibirá agua en la gran mayoría de las ocasiones en que lo requiera.
- Balance Aceptable (F1-score 0.88): El F1-score, que es un promedio armónico de precisión y recall, muestra un buen equilibrio en el rendimiento para la clase de interés (riego).
- Errores Controlados:
    - Solo 4 Falsos Positivos (FP): El modelo predice riego cuando no es necesario en muy pocas ocasiones.
    - Solo 8 Falsos Negativos (FN): El modelo no predice riego cuando sí es necesario en un número limitado de ocasiones. Si bien estos son críticos para la planta, 8 de 53 eventos (aproximadamente 15%) es un compromiso

aceptable para un sistema automatizado, especialmente si se considera que se evita el sobrerriego.

Podemos decir que, para la tarea de riego, el modelo ha aprendido a identificar de manera robusta las condiciones bajo las cuales se debe activar el riego. Este éxito se atribuye a la capacidad del modelo para identificar patrones efectivos en el conjunto de datos procesado, permitiéndole establecer una relación sólida entre las condiciones del suelo y la activación del riego.

- **Predicción de Lluvia**

Los resultados de la Figura 24 para la predicción de lluvia muestran un desafío significativo en el aprendizaje del modelo.



	precision	recall	f1-score	support
No Lluvia (0)	0.98	1.00	0.99	140
Lluvia (1)	0.00	0.00	0.00	3
accuracy			0.98	143
macro avg	0.49	0.50	0.49	143
weighted avg	0.96	0.98	0.97	143

Matriz de Confusión (Lluvia):
[[140 0]
[3 0]]

Figura 24. Captura de resultados modelo final predicción de lluvia

Análisis e Interpretación:

- Desequilibrio Extremo: El conjunto de datos de prueba contiene 140 casos de "No Lluvia" y solo 3 de "Lluvia". Este desequilibrio es la causa principal del comportamiento del modelo.
- Clasificador Constante: El modelo se comporta como un clasificador que siempre predice "No Lluvia".
    - Precision y Recall de 0.00 para "Lluvia (1)": Esto indica que el modelo nunca predice correctamente un evento de lluvia. De los 3 casos reales de lluvia, no detecta ninguno.
    - Perfecto para "No Lluvia": Por otro lado, es casi perfecto prediciendo la ausencia de lluvia (recall 1.00 para "No Lluvia").
- Accuracy Engañosa: La alta accuracy general del 0.98 se debe a que el modelo acierta casi siempre al predecir la clase mayoritaria ("No Lluvia").

- Causa del Comportamiento: Este comportamiento se atribuye a la escasez de datos de eventos de lluvia en el conjunto de entrenamiento. El modelo no tiene suficientes ejemplos para aprender los patrones que distinguen la lluvia de la no lluvia, y por lo tanto, opta por la predicción más segura y frecuente.

Las gráficas de pérdida y precisión del modelo durante el entrenamiento ofrecen una visión clara de su proceso de aprendizaje y las podemos ver en la Figura 25.

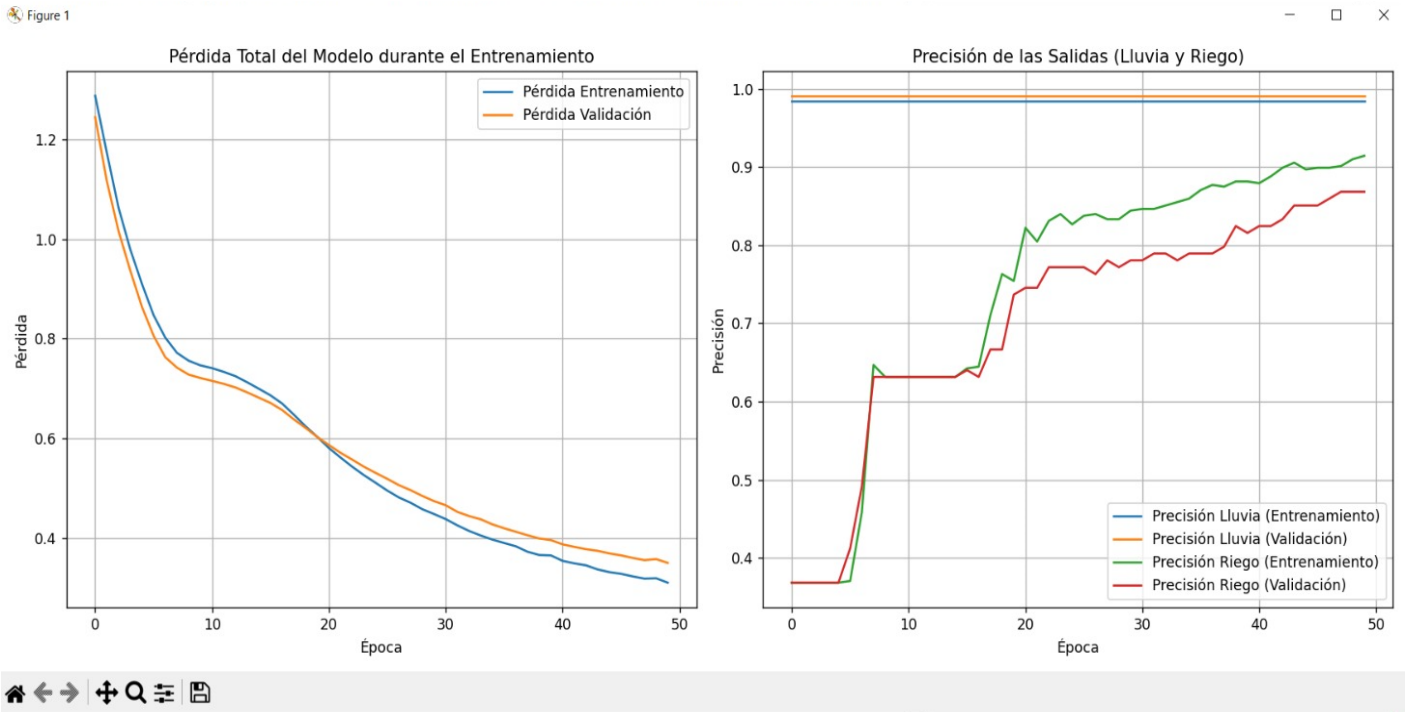


Figura 25. Gráfica de perdida y precisión en el modelo final

La gráfica de "Pérdida Total del Modelo durante el Entrenamiento" muestra que tanto la pérdida de entrenamiento como la de validación disminuyen de manera consistente a lo largo de las 50 épocas. Esto indica que el modelo está aprendiendo eficazmente y reduciendo su error de forma progresiva. La cercanía entre las curvas de pérdida de entrenamiento y validación sugiere una buena capacidad de generalización y no hay indicios de un sobreajuste significativo (overfitting) en el modelo global.

Por otro lado, la gráfica de "Precisión de las Salidas (Lluvia y Riego)" revela el comportamiento específico para cada tarea:

- Predicción de Lluvia: Las curvas de precisión para "Lluvia (Entrenamiento)" y "Lluvia (Validación)" suben abruptamente hasta casi el 100% (1.0) en las primeras épocas y se mantienen en ese nivel. Esto corrobora lo visto en el reporte de clasificación: el modelo es extremadamente preciso al predecir la clase mayoritaria ("No Lluvia"), lo que resulta en una alta precisión general para esta salida, pero no refleja su capacidad para identificar eventos reales de lluvia (clase minoritaria).

- Predicción de Riego: Las curvas de precisión para "Riego (Entrenamiento)" y "Riego (Validación)" muestran una evolución muy positiva. Parten de un valor bajo y

ascienden de forma constante y notable a lo largo de las épocas, alcanzando valores cercanos al 90% y 88% respectivamente. La forma ascendente y la cercanía entre la precisión de entrenamiento y validación para el riego demuestran que el modelo está aprendiendo de forma efectiva y generalizando bien para esta tarea, logrando mejorar su capacidad para discernir la necesidad de riego a medida que procesa más datos. Este comportamiento es un fuerte indicador del éxito del entrenamiento para la salida de riego.

En resumen, las gráficas de entrenamiento confirman que el modelo está aprendiendo de manera efectiva en general y de forma muy robusta para la predicción de riego. Por otro lado, ilustran visualmente el desafío inherente a la predicción de lluvia debido al desequilibrio de clases en los datos.

5.3. Interpretación General de Resultados y Validación del Diseño

La fase de pruebas ha permitido validar el correcto funcionamiento de los componentes hardware y firmware del sistema. Las lecturas de los sensores son estables y coherentes, y el sistema de registro y visualización opera sin fallos, lo que confirma la solidez de la base física y de la programación del microcontrolador.

En cuanto al modelo de ML, los resultados demuestran una validación parcial pero significativa del diseño.

- Validación de la Predicción de Riego: El modelo ha demostrado una excelente capacidad para predecir la necesidad de riego. Los altos valores de precisión y recall para la clase "Riego (1)" confirman que el modelo ha aprendido a identificar de manera robusta las condiciones que requieren la activación del riego. Esto valida la arquitectura de la red neuronal para esta tarea específica, corroborando que el diseño propuesto es una solución efectiva al problema de la automatización inteligente del riego.

- Consideraciones sobre la Predicción de Lluvia: La predicción de lluvia, si bien es una salida del modelo, presentó limitaciones significativas en su capacidad de aprendizaje. Esto se debió principalmente al extremo desequilibrio de la clase "Lluvia" en el conjunto de datos de entrenamiento, agravado por la recolección de datos en un periodo con escasas precipitaciones (mes de junio). El modelo, en la práctica, tendió a predecir la ausencia de lluvia de forma casi constante, lo que si bien resultaba en una alta precisión general (dada la escasez de eventos de lluvia), no permitía una detección fiable de las precipitaciones reales. Esta limitación subraya la importancia crítica de la representatividad de los datos para un aprendizaje exitoso en problemas de clasificación.

- Coherencia con los Supuestos de Diseño: Los resultados son coherentes con los supuestos de diseño, especialmente en lo que respecta a la capacidad del sistema

embebido para ejecutar modelos de IA y gestionar datos de sensores. La limitación en la predicción de lluvia no indica un error en la fase de diseño del modelo en sí, sino una restricción inherente a la disponibilidad y naturaleza de los datos.

- Validación del Diseño como Solución: A pesar de la limitación en la predicción de lluvia, el diseño global del sistema corrobora la validez de la solución propuesta para el control inteligente del riego. La capacidad del modelo para predecir la necesidad de riego con alta fiabilidad, combinada con la robustez del hardware y la posibilidad de implementar un mecanismo de seguridad adicional para la lluvia, asegura que el sistema puede operar de manera efectiva y eficiente en un entorno real. Este proyecto demuestra la viabilidad de aplicar técnicas de ML en dispositivos de bajo coste para la agricultura de precisión.

Podemos observar en la gráfica de la Figura 26 cómo se comporta el circuito.

Figura 26. Análisis del flujo de datos del modelo en tiempo real (1)

Vemos que, por un lado, tenemos los datos de los sensores, que se ha intentado capturar justo en el momento del riego, la temperatura y humedad ambiente vemos que no cambia, y de hecho no tiene que cambiar, y por otra parte observamos cómo las salidas del modelo cambian en consecuencia.

En la Figura 27 también se ha intentado capturar un momento de cambio, para visualizar el funcionamiento de las salidas del modelo.

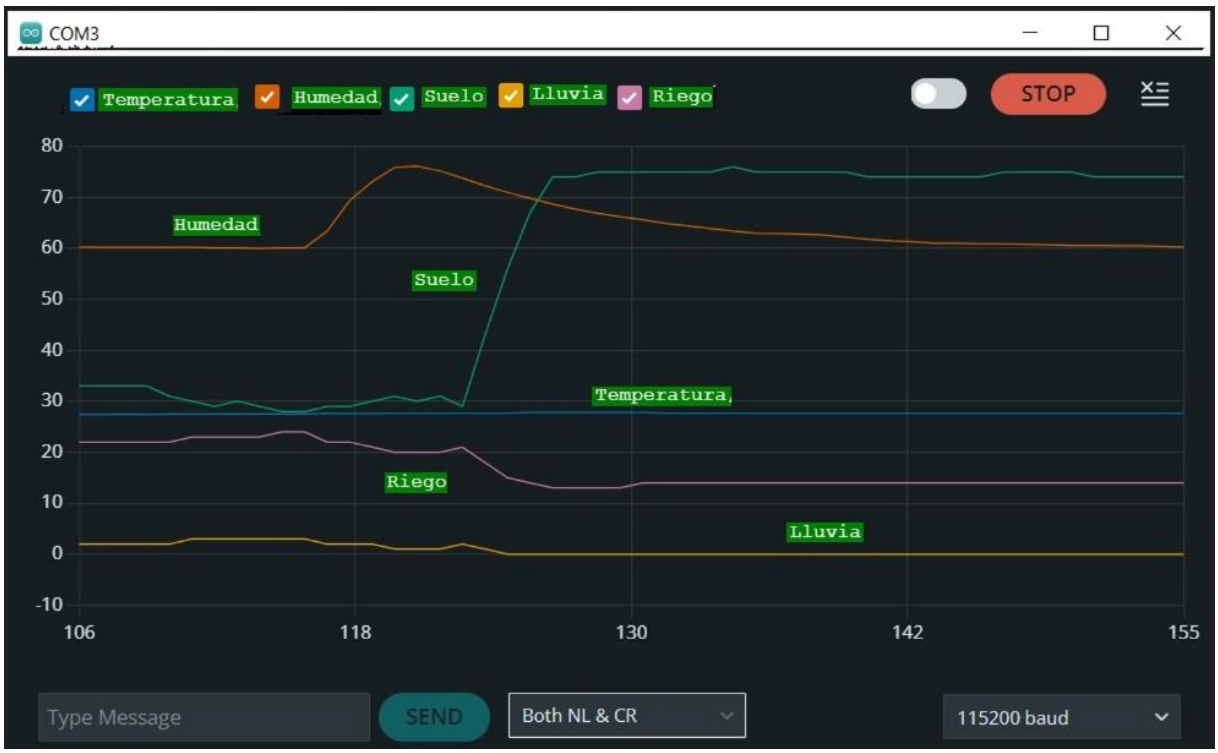


Figura 27. Análisis del flujo de datos del modelo en tiempo real (2)

Se puede observar que la humedad sube y lo hace la humedad del suelo. El resultado del modelo indica una bajada en la previsión de riego como consecuencia, que es lo esperado.

6. Presupuesto

En esta sección se detalla la estimación económica de todos los recursos necesarios para la concepción y materialización de este proyecto, incluyendo tanto los componentes físicos como el esfuerzo humano dedicado.

La filosofía de este trabajo ha sido priorizar el empleo de hardware de bajo coste y herramientas de software de código abierto. Esta elección ha permitido minimizar la inversión monetaria directa en materiales, facilitando así la accesibilidad y difusión de la tecnología desarrollada.

Herramientas de software utilizadas: Todo el software empleado en este proyecto es de acceso libre y de naturaleza abierta, en consonancia con los requisitos establecidos en el capítulo de Diseño y Desarrollo. Esto comprende:

- Entorno de programación: Arduino IDE (0,00 €)
- Lenguaje de codificación: Python (0,00 €)
- Bibliotecas y plataformas de IA/ML: TensorFlow, Keras, Pandas, NumPy, Scikit-learn, TensorFlow Lite para microcontroladores (TFLite Micro), tflm_esp32 (0,00 €)
- Bibliotecas específicas para Arduino: Adafruit GFX, Adafruit ST7789, DHT, WiFi.h, time.h, FS.h, SPIFFS.h (0,00 €)
- Utilidades complementarias: Controladores USB-a-Serie, Git Bash (para la transformación de modelos) (0,00 €)

Costes de personal (desarrollo e investigación): Para la estimación de los costes de personal, se ha considerado una tarifa horaria de 15,60 €, basada en el salario promedio de un profesional recién titulado en el sector.[28] Todo ello lo podemos ver en la Tabla 3.

DESCRIPCIÓN	PRECIO UNITARIO (€)	UNIDADES	PRECIO TOTAL (€)
Microcontrolador ESP32 con pantalla integrada de 1.14”	15,00	1	15
Sensor DHT22	5	1	5
Sensor de humedad del suelo capacitivo	4	1	4
Cables de conexión y Cable micro-USB	5	1	5
Entorno de desarrollo software (licencias)	0	1	0
Personal de investigación / hora	15,6	100 horas	1.560
Personal de desarrollo / hora	15,6	260 horas	4.056
TOTAL PRESUPUESTO			**5.640**

Tabla 3. Presupuesto total del proyecto.

Este análisis presupuestario subraya el compromiso del proyecto con la viabilidad económica y la sostenibilidad, demostrando que es factible implementar soluciones de IA aplicada con una inversión financiera controlada.

7. Impacto del proyecto

El desarrollo de este sistema de predicción meteorológica y monitoreo ambiental basado en sensores físicos y modelos de IA tiene múltiples implicaciones que se extienden más allá del ámbito puramente técnico.

7.1. Impacto

A continuación se detallan los principales impactos que este trabajo puede tener en diferentes ámbitos:

7.1.1. Impacto social

El acceso a información ambiental en tiempo real es un factor clave para mejorar la toma de decisiones tanto en contextos agrícolas como urbanos. La predicción de lluvia y la supervisión de condiciones climáticas permite:

- Optimizar el uso del agua en sistemas de riego, promoviendo una agricultura más eficiente.
- Prevenir riesgos asociados al exceso o la falta de humedad en cultivos.
- Acercar el acceso a tecnología de monitoreo a los pequeños agricultores mediante soluciones de bajo coste y libre acceso al código.

El sistema, por su bajo precio y simplicidad, puede ser utilizado en zonas rurales o en pequeños huertos urbanos, contribuyendo así a la autosuficiencia alimentaria y al empoderamiento de comunidades locales.

7.1.2. Impacto ambiental

Uno de los mayores retos ambientales actuales es la gestión eficiente del agua. Este proyecto contribuye positivamente a este desafío al permitir:

- Evitar el riego innecesario gracias a predicciones basadas en datos históricos y meteorológicos.
- Conservar recursos hídricos al aplicar criterios inteligentes de riego.
- Reducir el uso de energía asociado al bombeo de agua o uso de sistemas automatizados en función de datos reales del entorno.

7.1.3. Impacto en salud y seguridad

Aunque indirecto, el sistema ayuda a minimizar los riesgos de exposición prolongada al calor extremo o a condiciones desfavorables, al proporcionar información útil para proteger cultivos y entornos habitados. En el contexto urbano, puede integrarse para apoyar la creación de microclimas o zonas verdes adaptadas.

7.1.4. Impacto económico

El prototipo está diseñado con componentes de bajo coste (ESP32 con pantalla integrada, sensores comunes), lo que lo hace asequible para pequeños agricultores, centros educativos o makers. Esto puede suponer:

- Una reducción en los costes operativos de riego y mantenimiento de cultivos.
- Una oportunidad de replicación para emprendedores en el ámbito de la agrotecnología.
- Una plataforma educativa para enseñar conceptos de sensórica, IoT e IA de forma práctica.

7.1.5. Impacto tecnológico e industrial

El sistema representa un ejemplo práctico de convergencia entre IoT (Internet of Things), IA y sensorización ambiental. Su implementación demuestra la viabilidad de desplegar modelos de ML en dispositivos embebidos de bajo consumo, promoviendo:

- La adopción de IA en entornos de recursos limitados.
- Procesamiento local con IA para tareas de predicción ambiental.
- La experimentación con tecnologías abiertas y replicables, favoreciendo la innovación.

7.2. Alineación con los Objetivos de Desarrollo Sostenible (ODS)

Este proyecto se alinea de manera significativa con varios ODS definidos por la ONU:

• ODS 6 – Agua limpia y saneamiento: Fomenta el uso eficiente del agua mediante predicción meteorológica y control de humedad del suelo.

• ODS 12 – Producción y consumo responsables: Mejora la gestión de recursos agrícolas mediante datos.

- ODS 13 – Acción por el clima: Contribuye a la adaptación al cambio climático a través del monitoreo climático local.

- ODS 9 – Industria, innovación e infraestructura: Promueve tecnologías accesibles, replicables y sostenibles.

El proyecto no solo demuestra la aplicación efectiva de tecnologías modernas como la IA y el IoT en problemas reales, sino que también aporta un impacto positivo en múltiples dimensiones: ambiental, económica, educativa y social. Su enfoque abierto y su potencial de escalabilidad lo convierten en una herramienta útil para una transición hacia prácticas más sostenibles y tecnológicamente integradas.

8. Conclusiones

El presente Proyecto Fin de Grado ha abordado con éxito el desarrollo y la implementación de un sistema de riego inteligente, cuyo propósito fundamental ha sido la aplicación práctica de técnicas de IA y ML en un microcontrolador ESP32. Se buscaba demostrar la viabilidad de implementar estas técnicas, usualmente empleadas en entornos de alto rendimiento, en dispositivos con recursos limitados, para optimizar el consumo hídrico en la agricultura mediante la predicción de la necesidad de riego y la detección de lluvia, utilizando un enfoque de IA en el borde.

8.1. Logros y Resultados Obtenidos:

- Viabilidad de IA en Hardware Limitado: Se ha demostrado de manera concluyente la viabilidad de implementar y ejecutar modelos de ML (TensorFlow Lite) en un microcontrolador ESP32. Esto valida el objetivo principal de aplicar técnicas de IA, usualmente asociadas a entornos de alto rendimiento, en dispositivos de bajo coste y con capacidades computacionales restringidas.
- Sistema Embebido Funcional: Se ha diseñado e integrado un prototipo funcional que combina la adquisición de datos de sensores ambientales (temperatura, humedad ambiental y del suelo), la sincronización horaria vía NTP, la ejecución local del modelo de IA y la visualización de resultados en una pantalla TFT integrada. La estabilidad operativa del sistema ha sido verificada en pruebas prolongadas, confirmando su robustez.
- Predicción Robusta de la Necesidad de Riego: El modelo de red neuronal desarrollado ha alcanzado un rendimiento altamente satisfactorio en la predicción de la necesidad de riego. Gracias a la estrategia de optimización del conjunto de datos, el modelo ha logrado una precisión del 0.92 y un recall del 0.85 para la clase "Riego (1)", lo que se traduce en una minimización efectiva del desperdicio de agua y una adecuada hidratación de los cultivos. Este es un logro fundamental para la aplicación práctica del sistema.
- Contribución Propia y Adaptación Tecnológica: Las aportaciones originales del proyecto incluyen la integración funcional de todos los componentes, el diseño y optimización de la lógica de procesamiento y toma de decisiones directamente en el microcontrolador (incluyendo la normalización de datos y la lectura suavizada de humedad del suelo), y la adaptación específica del modelo de IA para el entorno ESP32. Esto ha requerido un profundo conocimiento de las herramientas y librerías utilizadas, así como la resolución de desafíos técnicos inherentes a la IA embebida.
- Impacto Positivo: El proyecto se alinea con los Objetivos de Desarrollo Sostenible (ODS 6, 12, 13 y 9), promoviendo la gestión eficiente del agua, la producción responsable, la acción climática y la innovación accesible. Su naturaleza de hardware abierto y bajo coste lo convierte en una solución replicable y con potencial para democratizar el acceso a la tecnología en el sector agrícola.

8.2. Conocimiento Nuevo y Aspectos Detectados:

- Importancia del Balance de Datos: La experiencia con la predicción de lluvia ha resaltado la importancia crítica de contar con conjunto de datos balanceados y representativos para todas las clases. El extremo desequilibrio de la clase "Lluvia" en el conjunto de datos, agravado por la recolección de datos en un periodo con escasas precipitaciones, impidió que el modelo aprendiera a detectar la lluvia de manera fiable, comportándose como un clasificador constante de "No Lluvia". Esto subraya que, incluso con arquitecturas de modelo adecuadas, la calidad y diversidad de los datos son primordiales.
- Mecanismos de Seguridad en IA Embebida: Ante la limitación en la predicción de lluvia, se ha confirmado la necesidad de integrar mecanismos de seguridad basados en lecturas directas de sensores (como un sensor de lluvia físico) para garantizar la fiabilidad del sistema en situaciones críticas, complementando las predicciones del modelo de IA.
- Optimización del conjunto de datos como clave del éxito: La notable mejora en la predicción de riego demuestra que, cuando los datos originales son insuficientes o están sesgados, una estrategia bien definida de aumento y optimización del conjunto de datos puede ser tan o más crucial que la complejidad del propio modelo para lograr un rendimiento práctico.

8.3. Líneas Futuras de Trabajo:

Aunque el proyecto ha alcanzado sus objetivos principales, existen varias líneas de trabajo futuras que podrían expandir y mejorar el sistema:

- Mejora de la Predicción de Lluvia: Recopilar un conjunto de datos mucho más amplio y diverso de eventos de lluvia, abarcando diferentes estaciones y condiciones climáticas, sería fundamental para entrenar un modelo de predicción de lluvia más fiable. Esto podría implicar el uso de sensores de lluvia adicionales o la integración con APIs meteorológicas externas para enriquecer los datos.
- Integración de Actuadores de Riego: La implementación de una válvula o bomba de riego controlada directamente por el ESP32, activándose automáticamente según la decisión del sistema, completaría el ciclo de automatización del riego.
- Monitoreo y Control Remoto: Desarrollar una interfaz de usuario remota (aplicación móvil o plataforma web) que permita al agricultor monitorear los datos, las predicciones y el estado del sistema desde cualquier lugar, así como ajustar parámetros o activar el riego manualmente. Esto requeriría la implementación de protocolos de comunicación IoT (MQTT, HTTP) y una base de datos en la nube.
- Análisis de Datos a Largo Plazo: Implementar un sistema de almacenamiento de datos más robusto (ej., tarjeta SD, base de datos en la nube) para registrar el comportamiento del sistema a lo largo de meses o años, lo que permitiría un análisis

más profundo de la eficiencia del riego y posibles re-entrenamientos del modelo con datos de campo más extensos.

- Modelos Más Complejos o Específicos: Explorar arquitecturas de redes neuronales más avanzadas o modelos específicos para series temporales (ej., LSTMs) si se busca predecir la necesidad de riego con mayor antelación o con una granularidad temporal más fina.
- Calibración y Adaptación Automática: Desarrollar algoritmos de calibración automática para los sensores (especialmente el de humedad del suelo) y mecanismos de adaptación del modelo a diferentes tipos de suelo o cultivos.

Este proyecto establece una base sólida para la aplicación de la IA en la agricultura de precisión, demostrando que las soluciones inteligentes y sostenibles son accesibles incluso con recursos limitados. La experiencia adquirida y las lecciones aprendidas abren un camino prometedor para futuras investigaciones y desarrollos en este campo.

9. Referencias

[1] «Inteligencia Artificial - Telefónica». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://www.telefonica.com/es/nosotros/politicas-publicas-y-regulacion/posicionamiento/inteligencia-artificial/

[2] «Noticias sobre inteligencia artificial: avances, aplicaciones y desafíos en la tecnología actual | Computer Hoy». Accedido: 31 de julio de 2025. [En línea]. Disponible en: https://computerhoy.20minutos.es/tags/inteligencia-artificial

[3] «¿Qué es la IA sensible? | IBM». Accedido: 4 de agosto de 2025. [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/sentient-ai

[4] «Machine Learning: qué es, tipos y aplicaciones». Accedido: 4 de septiembre de 2025. [En línea]. Disponible en: https://www.repsol.com/es/energia-avanzar/innovacion/machine-learning/index.cshtml?utm_source=google&utm_medium=cpc&utm_campaign=crt-aotrafico-unbrand_2502_corp-all&utm_content=all_gad_link_sem_all_machinelearning_all_corp_all_all_trf_cpc_cc&gad_source=1&gad_campaignid=18959993176&gbraid=0AAAAACUhDPPQZCUucKQM6HbM-5YQ_IW5K&gclid=Cj0KCQjwzt_FBhCEARIsAJGFWVkyNRxWKd_fd6pzYScCXBvAiFkq6cjwn11J1PaH4Wezx9vhkTx0iX0aArvNEALw_wcB

[5] «Tipos de aprendizaje en Machine Learning: supervisado y no supervisado». Accedido: 4 de septiembre de 2025. [En línea]. Disponible en: https://telefonicatech.com/blog/que-algoritmo-elegir-en-ml-aprendizaje

[6] «Tipos de aprendizaje en Machine Learning: supervisado y no supervisado». Accedido: 4 de septiembre de 2025. [En línea]. Disponible en: https://telefonicatech.com/blog/que-algoritmo-elegir-en-ml-aprendizaje

[7] «¿Qué es Aprendizaje profundo (deep learning)? - Definición en Computer Weekly». Accedido: 31 de julio de 2025. [En línea]. Disponible en: https://www.computerweekly.com/es/definicion/Aprendizaje-profundo-deep-learning

[8] «05.1 Redes Neuronales Introducción — Introducción al Aprendizaje Automático». Accedido: 31 de julio de 2025. [En línea]. Disponible en: https://dcain.etsin.upm.es/~carlos/bookAA/05.1_RedesNeuronalesIntroduccion.html

[9] «rneuronales | A fine WordPress.com site». Accedido: 4 de agosto de 2025. [En línea]. Disponible en: https://rneuronales.wordpress.com/

[10] «Perceptron : qu'est-ce que c'est et à quoi ça sert ?» Accedido: 31 de julio de 2025. [En línea]. Disponible en: https://datascientest.com/es/perceptron-que-es-y-para-que-sirve

[11] R. García, «EL PERCEPTRÓN: UNA RED NEURONAL ARTIFICIAL PARA CLASIFICAR DATOS», Accedido: 31 de julio de 2025. [En línea]. Disponible en: http://www.economicas.uba.ar/institutos_y_centros/revista-modelos-matematicos/

[12] «Introducción a las funciones de activación en las redes neuronales | DataCamp». Accedido: 31 de julio de 2025. [En línea]. Disponible en: https://www.datacamp.com/es/tutorial/introduction-to-activation-functions-in-neural-networks

[13] «¿Qué es una red neuronal? | Tipos de redes neuronales | Cloudflare». Accedido: 31 de julio de 2025. [En línea]. Disponible en: https://www.cloudflare.com/es-es/learning/ai/what-is-neural-network/

[14] «Red Neural: Definición, Tipos y Ejemplos | StudySmarter». Accedido: 31 de julio de 2025. [En línea]. Disponible en: https://www.studysmarter.es/resumenes/psicologia/estudios-sobre-la-memoria/red-neuronal/

[15] «¿Qué es Epoch en Machine Learning?» Accedido: 31 de julio de 2025. [En línea]. Disponible en: https://ciberseguridadmax.com/epoch/

[16] «Sobreajuste y Subajuste en Machine Learning - AprendeIA». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://aprendeia.com/2018/11/16/sobreajuste-y-subajuste-en-machine-learning/

[17] «Red neuronal de retropropagación: Explicado con un ejemplo sencillo | NCGo». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://ncgovote.org/es/red-neuronal-de-retropropagaci%C3%B3n-explicado-con-un-ejemplo-sencillo/

[18] «Aprendizaje profundo: Método del descenso del gradiente | Aula En Abierto». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://formacion.intef.es/aulaenabierto/mod/book/view.php?id=5077&chapterid=6494#mod_book-chapter

[19] «36. Graph of gradient descent | Download Scientific Diagram». Accedido: 4 de agosto de 2025. [En línea]. Disponible en: https://www.researchgate.net/figure/Graph-of-gradient-descent_fig29_364363862

[20] «¿Qué es el Perceptrón? Perceptrón Simple y Multicapa - AprendeIA». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://aprendeia.com/2021/10/05/que-es-el-perceptron-simple-y-multicapa/

[21] «TensorFlow: Concepto y Aplicaciones de Redes Neuronales de Múltiples Capas». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://certidevs.com/tutorial-tensorflow-arquitectura-redes-neuronales-multiples-capas

[22] «Redes multicapa y ajuste de hiperparámetros - Inteligencia Artificial». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://nachoiborraies.github.io/ia/05c.html#23-el-overfitting

[23] «Aprendizaje profundo: ¿qué es deep learning y cómo utilizarlo con datos sintéticos? - OpenSistemas». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://opensistemas.com/aprendizaje-profundo-con-datos-sinteticos/

[24] «Redes neuronales con Python». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://cienciadedatos.net/documentos/py35-redes-neuronales-python

[25] «Explicación de las funciones de pérdida en el machine learning | DataCamp». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://www.datacamp.com/es/tutorial/loss-function-in-machine-learning

[26] «La agricultura de precisión». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://www.plataformatierra.es/innovacion/la-agricultura-de-precision

[27] «Agricultura de precisión impulsada por IA para rendimientos óptimos de los cultivos». Accedido: 3 de agosto de 2025. [En línea]. Disponible en: https://editverse.com/es/ai-driven-precision-agriculture-for-optimal-crop-yields/

[28] «Ingeniero electronico salario en Madrid - Comprueba ingeniero electronico salario promedio en Jooble». Accedido: 4 de agosto de 2025. [En línea]. Disponible en: https://es.jooble.org/salary/ingeniero-electronico/Madrid

Anexo

Manual de usuario

En este capítulo, se proporciona una guía completa sobre cómo poner en marcha y operar el sistema de riego inteligente desarrollado, con el objetivo de crear y poner en marcha un modelo de ML. Se abordarán todos los pasos necesarios, desde la configuración inicial del entorno de desarrollo hasta la ejecución y el monitoreo de las funciones del prototipo.

A.1 Descripción General del Sistema y su Propósito

El presente trabajo aborda el desarrollo de un sistema inteligente basado en ML aplicado a un microcontrolador ESP32, enfocado en la optimización del riego agrícola. El propósito fundamental de este proyecto es aprender y aplicar técnicas de IA y ML en dispositivos con recursos limitados, como los microcontroladores, hasta obtener un prototipo funcional. Como caso concreto y práctico, se ha empleado un supuesto en agricultura para el control inteligente del riego. El objetivo final es crear un prototipo funcional que pueda predecir la probabilidad de lluvia y la necesidad de riego mediante sensores ambientales.

A.2 Componentes y Preparación del Entorno

Esta sección detalla los componentes hardware necesarios y los pasos esenciales para preparar el entorno de desarrollo y realizar las conexiones físicas.

A.2.1. Contenido del Sistema (Componentes Hardware)

Para la puesta en marcha del sistema, se necesitarán los siguientes componentes hardware:

- Microcontrolador ESP32 Development Board con Pantalla TFT ST7789 integrada: La plataforma principal que aloja el firmware y ejecuta el modelo de IA.
- Sensor DHT22 con cables de conexión: Utilizado para la adquisición de datos de temperatura y humedad del aire.
- Sensor de humedad del suelo capacitivo con cables de conexión: Para medir el nivel de humedad en el sustrato.
- Cable micro-USB: Para alimentar el ESP32 y facilitar la carga del programa desde un ordenador.

A.2.2. Preparación del Entorno de Desarrollo (Software y Librerías)

Para poder instalar y cargar el programa en el ESP32, se necesitará configurar el siguiente entorno de desarrollo y sus librerías asociadas en el ordenador:

- Arduino IDE: El entorno de desarrollo integrado (IDE) es la herramienta principal para programar el microcontrolador ESP32. Se recomienda descargar la última versión estable directamente desde la página oficial de Arduino.

- Soporte para la placa ESP32 en Arduino IDE: Para que el Arduino IDE pueda compilar y cargar código para el ESP32, se debe añadir el soporte para esta placa. Hay que ir a Archivo > Preferencias, en el campo "URLs Adicionales de Gestores de Tarjetas" se añade https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json. Luego, desde el Gestor de Tarjetas (Herramientas > Placa > Gestor de Tarjetas...), se busca e instala el paquete "esp32 by Espressif Systems".

- Bibliotecas de Arduino: Las siguientes bibliotecas son fundamentales para el funcionamiento de los sensores, la pantalla y el modelo de IA. Se deberán instalar a través del Administrador de Librerías (Sketch > Incluir Librería > Administrar Librerías...):

    - Adafruit GFX Library: Biblioteca base para gráficos.
    - Adafruit ST7735 and ST7789 Library: Controlador específico para la pantalla TFT.
    - DHT sensor library: Para la lectura del sensor DHT22.
    - tflm_esp32: Implementación de TensorFlow Lite adaptada al entorno ESP32. Permite la inferencia directamente en el microcontrolador sin conexión a servidores externos.

A.2.3. Conexiones Físicas (Cableado)

Las conexiones entre los sensores y el microcontrolador ESP32 deben realizarse con precisión. Se recomienda consultar el esquema de pines de la placa ESP32 si difiere del modelo de desarrollo estándar.

- Sensor DHT22 (Temperatura y Humedad Ambiental):
    - Pin de Datos conectado a GPIO 13 del ESP32
    - VCC conectado a Pin de 3.3V del ESP32
    - GND conectado a Pin GND del ESP32
    - Nota: Se recomienda una resistencia pull-up de 10 Ω entre el pin de datos y VCC para estabilidad.
- Sensor de Humedad del Suelo Capacitivo:
    - Pin de Salida (OUT) conectado a GPIO 34 (ADC1_CH6) del ESP32
    - VCC conectado a Pin de 3.3V del ESP32

- GND conectado a Pin GND del ESP32

• Pantalla TFT ST7789: Esta pantalla viene integrada en la placa de desarrollo ESP32 utilizada en este proyecto, por lo que no requiere cableado externo. Sus pines internos de conexión son:

- MOSI conectado a GPIO 23
- SCLK conectado a GPIO 18
- CS conectado a GPIO 15
- DC conectado a GPIO 2
- RST conectado a GPIO 4
- BLK conectado a GPIO 32

A.2.4. Carga del Programa (Firmware)

El firmware (apren_automat.ino) es el programa principal que controla el ESP32 y ejecuta el modelo de IA.

• Conectar el ESP32: Se debe conectar el microcontrolador ESP32 al ordenador utilizando el cable micro-USB.

• Abrir el Proyecto: Se abre el archivo .ino del proyecto (por ejemplo, apren_automat.ino) en Arduino IDE.

• Configurar la Red WiFi: Dentro del archivo .ino, se localizan las líneas:

- const char* ssid = "Redmi 9C";
- const char* password = "1234abcd";

• Y se reemplaza "Redmi 9C" y "1234abcd" con el nombre de la red WiFi (SSID) y su contraseña, respectivamente. Esta conexión es necesaria para la sincronización horaria mediante NTP.

• Seleccionar la Placa: En Arduino IDE, se debe ir a Herramientas > Placa y seleccionar ESP32 Dev Module (o la placa ESP32 específica que se esté utilizando).

• Seleccionar el Puerto: En Herramientas > Puerto, se selecciona el puerto COM al que esté conectado el ESP32.

• Cargar el Código: Se hace clic en el botón Cargar (la flecha hacia la derecha) en la barra de herramientas de Arduino IDE. El IDE compilará el código y lo cargará en el ESP32. Se debe esperar a que el proceso finalice.

A.3 Proceso de Desarrollo y Despliegue de la Inteligencia del Sistema

Esta sección describe los pasos fundamentales que se han seguido para obtener los datos, entrenar el modelo de red neuronal y prepararlo para su funcionamiento en el microcontrolador.

A.3.1. Recopilación y Preparación de Datos

La precisión del modelo de IA depende directamente de la calidad y cantidad de los datos con los que se entrena.

- Adquisición de Datos: La recopilación de datos se realiza principalmente a través de los sensores conectados al microcontrolador (temperatura, humedad ambiental y humedad del suelo). Estos datos se leen periódicamente (cada segundo para mostrar en pantalla, y cada hora para registrar).

- Almacenamiento y Transmisión de Datos: Cada hora, el sistema guarda los datos recogidos en un archivo CSV persistente en la memoria SPIFFS del ESP32. Adicionalmente, los datos se envían por el Monitor Serie de Arduino IDE (accesible desde Herramientas > Monitor Serie), lo que permite su visualización y captura en tiempo real en un ordenador.

- Etiquetado y Optimización del conjunto de datos: Los datos recopilados se procesan y se etiquetan para las salidas deseadas (lluvia y necesidad de riego).

A.3.2. Entrenamiento y Conversión del Modelo de red neuronal

Una vez que los datos han sido preparados, se procede al entrenamiento del modelo, seguido de su optimización para el despliegue en el microcontrolador.

- Entrenamiento del Modelo: Se utiliza un script de Python (entrenar_modelo.py) para cargar y preparar los datos. Este script define una red neuronal artificial, la entrena con los datos procesados y evalúa su rendimiento.

- Conversión a TensorFlow Lite: El modelo entrenado se convierte al formato TensorFlow Lite (.tflite). Este formato está optimizado para dispositivos con recursos limitados, lo que permite que el modelo se ejecute de manera eficiente en el ESP32.

A.3.3. Integración y Carga del Modelo en el Microcontrolador

El modelo optimizado se integra directamente en el firmware del ESP32 para su ejecución en tiempo real.

- Inclusión del Modelo en el Código Arduino: El archivo .tflite se convierte en un arreglo de bytes (modelo_final_riego.h) y se incluye directamente en el código de Arduino (apren_automat.ino). Esto permite que el modelo se cargue en la memoria del ESP32.

El proceso de conversión de un archivo .tflite a un archivo .h (cabecera C/C++) que contiene el modelo como un arreglo de bytes se realiza típicamente utilizando herramientas de línea de comandos. Se ha usado Git Bash y su sintaxis es:

xxd -i modelo_riego_lluvia.tflite > modelo_final_riego.h

Este comando lee el archivo binario modelo_riego_lluvia.tflite y genera un archivo modelo_final_riego.h que contiene una variable unsigned char con los bytes del modelo, junto con su tamaño. Este archivo .h puede entonces ser incluido directamente en el código de Arduino.

- Normalización en el Dispositivo: El código de Arduino incluye una función (normalize_input) que replica el escalado de datos realizado durante el entrenamiento, utilizando los valores data_min y data_max obtenidos del script de Python. Esto asegura que los datos de los sensores se preparen correctamente para la inferencia.

- Ejecución de Inferencia: El ESP32 utiliza librerías específicas (tflm_esp32) para ejecutar el modelo TFLite con los datos normalizados de los sensores, obteniendo las probabilidades de lluvia y necesidad de riego.

A.4 Funciones del Prototipo en Operación

Una vez que el programa ha sido cargado exitosamente y el ESP32 está alimentado, el sistema comenzará a operar de forma autónoma.

A.4.1. Visualización en Pantalla

La pantalla TFT integrada se activará y comenzará a mostrar la siguiente información, actualizada cada segundo, proporcionando un monitoreo en tiempo real de las condiciones ambientales y las decisiones del sistema:

- Fecha y Hora actual: Muestra la fecha y hora sincronizadas vía NTP
- Temperatura: La temperatura ambiente en grados Celsius
- Humedad Ambiental: La humedad relativa del aire en porcentaje
- Humedad del Suelo: El nivel de humedad en el sustrato en porcentaje
- Probabilidad de Lluvia: La probabilidad de que esté lloviendo según el modelo de red neuronal artificial
- Probabilidad de Riego: La probabilidad de que sea necesario regar según el modelo

- Decisión de Riego: La decisión final del sistema (Regar ahora o No regar). Esta decisión se actualizará en momentos específicos del día.

A.4.2. Evaluación del Modelo y Toma de Decisión

El modelo se ejecuta automáticamente para proporcionar decisiones de riego:

- Frecuencia de Evaluación: El modelo se ejecutará dos veces al día, a las 8:00 y a las 20:00 horas, para proporcionar la decisión de riego. Fuera de estos intervalos de evaluación, o si no se ha podido realizar una inferencia reciente, la pantalla mostrará "Esperando...".

- Proceso de Inferencia y Toma de Decisión: El sistema toma las lecturas actuales de los sensores, las prepara para el modelo de IA y realiza una predicción. El modelo devuelve probabilidades para la presencia de lluvia y la necesidad de riego. La decisión final de riego se toma combinando la probabilidad de necesitar riego con la información de un sensor de lluvia. Si la probabilidad de necesitar riego supera el umbral de 0.5, el sensor de lluvia indica que NO está lloviendo, el sistema mostrará la decisión de regar.

A.5 Mantenimiento y Calibración

Para asegurar la precisión y el rendimiento óptimo del sistema a lo largo del tiempo, se recomiendan las siguientes prácticas de mantenimiento:

- Limpieza del Sensor de Humedad del Suelo: Para garantizar lecturas precisas a largo plazo, se recomienda limpiar el sensor de humedad del suelo con un paño seco periódicamente. Esto es especialmente importante si el sensor ha estado en contacto prolongado con suelo muy húmedo o con residuos, ya que la acumulación puede afectar su conductividad y, por ende, la precisión de las lecturas.

- Recalibración del Sensor de Humedad del Suelo: Si el sistema se va a utilizar en un tipo de suelo significativamente diferente al original (por ejemplo, de tierra arcillosa a arenosa), o si las lecturas de humedad del suelo parecen inconsistentes con la realidad, puede ser necesario ajustar los valores SOIL_MIN y SOIL_MAX en el código de Arduino. Estos valores definen el rango de lectura para suelo muy húmedo y suelo seco, respectivamente. Su ajuste preciso puede mejorar la exactitud de la medición de humedad relativa del suelo.

A.6 Posibles Problemas y Soluciones

Esta sección describe algunos problemas comunes que pueden surgir durante la operación del sistema y ofrece posibles soluciones. Es importante abordar estos puntos de manera sistemática para diagnosticar y resolver cualquier incidencia.

Problemas de Conexión y Carga del Firmware

Si se encuentran dificultades al conectar el ESP32 al ordenador o al cargar el programa, se debe considerar lo siguiente:

- El ESP32 no es reconocido por el ordenador: Se debe asegurar que el cable micro-USB que se está utilizando es un cable de datos y no solo de carga, ya que algunos cables solo transmiten energía.

- Errores al cargar el código (ej. "Failed to connect to ESP32: Timed out waiting for packet header"): Este tipo de error sugiere que el ESP32 no está en modo de programación o que hay un problema con la comunicación a través del puerto. Primero, se debe verificar que se ha seleccionado el puerto COM correcto en el Arduino IDE (Herramientas > Puerto). En algunas placas ESP32, es necesario presionar y mantener presionado el botón "BOOT" (o "FLASH") en la placa mientras se hace clic en "Subir" en el IDE, y soltarlo una vez que el proceso de carga comienza. Si persiste, se puede intentar desconectar y volver a conectar el ESP32.

- Errores de compilación relacionados con librerías o la placa: Si el Arduino IDE muestra errores durante la compilación que mencionan librerías o la placa, es probable que alguna de las bibliotecas necesarias no esté instalada o que la versión sea incorrecta, o que el soporte para la placa ESP32 no esté configurado. Se debe revisar la sección correspondiente para asegurar que todas las librerías listadas están instaladas correctamente a través del Administrador de Librerías de Arduino IDE y que se ha añadido el soporte para la placa ESP32 en el Gestor de Tarjetas. Para la realización de este trabajo hemos encontrado alguna dificultad con las librerías descargadas a través del gestor de librerías, la librería tflm_esp32 para ser exactos y el problema residía en la duplicación de algunos ficheros de cabecera, exixtentes en la carpeta signal. El problema quedó solucionado al borrar la carpeta src de esta carpeta signal.

A.6.1. Problemas de Funcionamiento del Sistema

Una vez que el firmware está cargado, si el sistema no opera como se espera, se deben revisar estos puntos:

- La pantalla no muestra datos o muestra caracteres extraños: Si la pantalla TFT integrada no funciona correctamente, las causas pueden ser conexiones incorrectas (si la pantalla no viene integrada en la placa y se ha cableado manualmente), problemas en la inicialización de la pantalla en el código Arduino, o una alimentación insuficiente. Se deben verificar las conexiones de los pines de la pantalla según lo comentado anteriormente. Hay que revisar la sección setup() del código Arduino para confirmar que la inicialización de la pantalla (ej. tft.init(), tft.setRotation()) es correcta. Se debe asegurar que el ESP32 recibe suficiente energía.

- Las lecturas de los sensores son inconsistentes o no cambian: Esto puede indicar que un sensor está mal conectado, está defectuoso, o hay un error en la lógica de lectura en el

código. Se deben revisar cuidadosamente las conexiones del sensor (VCC, GND, Datos) según secciones anteriores. Si es posible, se puede probar el sensor individualmente en un circuito simple para verificar su funcionamiento. Finalmente, se debe examinar la parte del código Arduino responsable de leer y procesar los valores de los sensores.

- El sistema no se conecta a la red WiFi: La conectividad WiFi es esencial para la sincronización horaria. Si el ESP32 no logra conectarse, se debe verificar que el SSID (nombre de la red) y la contraseña en el código Arduino (sección 2.4, paso 3) coinciden exactamente con los de la red WiFi, ya que son sensibles a mayúsculas y minúsculas. Se debe asegurar que el ESP32 está dentro del rango de cobertura de la red y que no hay obstáculos significativos.

- La decisión de riego no se actualiza o parece incorrecta: Si la lógica de riego no se activa o proporciona decisiones inesperadas, puede haber un problema con la sincronización horaria, la ejecución del modelo de IA, o los umbrales de decisión. Primero, se debe confirmar que el ESP32 se ha conectado a WiFi y ha sincronizado la hora correctamente (la hora en la pantalla debe ser precisa). Se debe verificar que la función esHoraDeEvaluar() en el código está permitiendo la ejecución del modelo en los momentos esperados (8:00 y 20:00). Hay que revisar los umbrales de decisión para las probabilidades de riego en el código Arduino. Es crucial recordar que, dada la limitación actual del modelo de IA para la predicción de lluvia, la lectura directa de un sensor de lluvia es un mecanismo de seguridad fundamental para evitar el riego durante precipitaciones reales, independientemente de la predicción del modelo.

A.6.2. Problemas de Rendimiento del Modelo de IA

Si las predicciones del modelo de ML no son precisas, se deben considerar estos factores:

- El modelo de riego predice incorrectamente con frecuencia: Esto puede deberse a que los datos de entrenamiento no fueron lo suficientemente variados o representativos, el umbral de decisión en el código Arduino no es óptimo, o el modelo necesita ser reentrenado. Para futuras iteraciones, se recomienda recolectar más datos en diversas condiciones ambientales. Se puede experimentar ajustando el umbral de decisión en el código Arduino para ver si mejora la precisión o el recall según las prioridades. Finalmente, se puede considerar reentrenar el modelo Python con un conjunto de datos más amplio o con diferentes estrategias de aumento de datos.

- El modelo de lluvia no predice lluvia en absoluto: Como se ha detallado anteriormente en el informe técnico, este es un problema conocido debido al desequilibrio severo de clases en el conjunto de datos de entrenamiento (muy pocos eventos de lluvia). Para la implementación actual del prototipo, la solución más robusta es depender de un sensor de lluvia directo como mecanismo de seguridad. Para mejorar la predicción de lluvia con el modelo de IA en el futuro, sería necesario recolectar un conjunto de datos mucho más amplio que incluya una cantidad significativa de eventos de lluvia.