UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería de Computadores

Gemelo digital de un invernadero para optimizar el riego

Digital twin of a greenhouse to optimize irrigation

Realizado por
Esther Almendros Pérez

Tutorizado por
Javier Troya Castilla
Paula Muñoz Ariza

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, junio 2025

UNIVERSIDAD
DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE COMPUTADORES

Gemelo digital de un invernadero para optimizar el riego

Digital twin of a greenhouse to optimize irrigation

Realizado por
Esther Almendros Pérez

Tutorizado por
Javier Troya Castilla
Paula Muñoz Ariza

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2025

Fecha defensa: julio de 2025

Abstract

Climate change and the growing shortage of water resources have created an urgent need to optimize water consumption in agriculture. In this context, the application of smart technologies can offer sustainable and efficient solutions.

This project proposes the development of an intelligent irrigation system based on the concept of a digital twin—a virtual replica of the physical greenhouse that anticipates crop water needs by analyzing real-time environmental data. The system combines sensing, automation, prediction, and IoT technologies to make autonomous irrigation decisions, adapting to the specific conditions of each moment.

Through the integration of physical sensors, a time-series database, and a predictive model trained using machine learning, the system is capable of intelligently managing water supply and contributing to more precise, efficient, and sustainable agriculture.

Keywords: Precision Agriculture, Digital Twin, Smart Irrigation, IoT, Greenhouse, Prediction

Resumen

El cambio climático y la creciente escasez de recursos hídricos han generado una necesidad urgente de optimizar el consumo de agua en la agricultura. En este contexto, la aplicación de tecnologías inteligentes puede aportar soluciones sostenibles y eficientes.

Este trabajo propone el desarrollo de un sistema de riego inteligente basado en el concepto de gemelo digital, una réplica virtual del invernadero físico que permite anticipar las necesidades de agua de los cultivos mediante el análisis de datos ambientales en tiempo real. El sistema combina sensorización, automatización, predicción y tecnologías IoT para tomar decisiones autónomas sobre el riego, ajustándolo a las condiciones concretas de cada momento.

Mediante la integración de sensores físicos, una base de datos de series temporales y un modelo predictivo entrenado con aprendizaje automático, el sistema es capaz de controlar de forma inteligente el suministro de agua y contribuir así a una agricultura más precisa, eficiente y sostenible.

Palabras clave: Agricultura de precisión, Gemelo digital, Riego inteligente, IoT, Invernadero, Predicción

Índice

1 Introducción

Este capítulo introductorio presenta el contexto general del proyecto, sus motivaciones y los objetivos que se persiguen. Se describen las tareas que se han planificado para su desarrollo, así como la metodología de trabajo adoptada, basada en enfoques ágiles adaptados a un entorno académico individual. Además, se ofrece una visión general de la estructura del documento, indicando la organización de los distintos capítulos y el contenido que abordan, con el fin de facilitar la comprensión del lector y ofrecer una visión clara del recorrido realizado a lo largo del proyecto.

1.1. Motivación

El cambio climático y la sobreexplotación de los recursos hídricos han convertido la escasez de agua en un problema global que afecta especialmente a la agricultura. En regiones áridas y semiáridas, donde las precipitaciones son escasas y los periodos de sequía son cada vez más frecuentes, optimizar el uso del agua se ha vuelto una necesidad urgente. La agricultura, al ser el sector que más agua consume a nivel mundial, debe implementar estrategias innovadoras para mejorar su eficiencia hídrica sin comprometer la producción de alimentos [1].

La agricultura de precisión es una estrategia de gestión agrícola que utiliza tecnologías avanzadas (como sensores, sistemas de posicionamiento global, análisis de datos e inteligencia artificial) para monitorizar y optimizar las prácticas agrícolas de forma localizada y eficiente. Su objetivo es aplicar los suministros de manera específica donde y cuando son necesarios, mejorando el rendimiento de los cultivos y reduciendo el impacto ambiental. En este proyecto, la agricultura de precisión es especialmente relevante porque permite tomar decisiones automatizadas y basadas en datos reales sobre el riego, ajustándolo a las condiciones del suelo y del ambiente en tiempo real, lo que se traduce en un uso más sostenible del agua y una mejor salud del cultivo [2].

La automatización del riego mediante el uso de sensores y sistemas inteligentes se ha posicionado como una solución viable para minimizar el desperdicio de agua. Sin embargo, muchos sistemas tradicionales de riego automatizado funcionan con reglas fijas que no siempre se adaptan a las condiciones climáticas cambiantes ni a las necesidades específicas de cada cultivo. Es aquí donde la aplicación de tecnologías avanzadas, como los gemelos digitales y el Internet de las Cosas (IoT), puede marcar una diferencia significativa.

Un gemelo digital es una réplica virtual de un sistema físico que permite monitorizar, analizar y optimizar el funcionamiento de su contraparte física mediante el uso de datos en tiempo real [3]. En este proyecto, se creará un gemelo digital de un invernadero automatizado, donde se recogerán datos de sensores de humedad del suelo, temperatura ambiental y otros factores clave para la gestión del riego. Gracias a este modelo virtual, será posible predecir las necesidades de agua de las plantas y ajustar el riego de manera eficiente para evitar desperdicios.

Por otro lado, el Internet de las Cosas (IoT) facilitará la comunicación entre los sensores del invernadero y el sistema de control. A través de dispositivos conectados, se recopilarán y enviarán datos a la base de datos en tiempo real, lo que permitirá que el gemelo digital haga análisis predictivos y tome decisiones automatizadas [4]. Esto garantizará un uso más eficiente del agua y permitirá una gestión más inteligente del cultivo.

El desarrollo de este proyecto no solo busca mejorar la eficiencia del riego en invernaderos y cultivos controlados, sino también servir como una prueba de concepto para la aplicación de estas tecnologías en la agricultura de precisión [5]. De esta manera, se podrá contribuir a la sostenibilidad del sector agrícola y a la conservación de los recursos hídricos en un contexto de creciente escasez.

1.2. Objetivos

El objetivo principal de este proyecto es el desarrollo de un sistema automatizado de riego basado en tecnologías IoT y gemelos digitales, con el fin de optimizar el uso del agua en cultivos controlados. Para ello, se diseñará un invernadero automatizado capaz de monitorizar y gestionar el riego de forma autónoma, minimizando el desperdicio de agua y mejorando la eficiencia agrícola.

El sistema físico estará compuesto por un conjunto de sensores instalados en el invernadero, encargados de medir variables clave como la temperatura, la humedad del aire y la

humedad del suelo. Estos sensores permitirán recopilar datos en tiempo real sobre el estado del entorno y de los cultivos. Con esta información, el sistema de control, gestionado por un microcontrolador, podrá tomar decisiones automatizadas sobre el riego: cuándo activarlo y durante cuánto tiempo, en función de los niveles de humedad detectados. De esta manera, se evita tanto el riego insuficiente como el exceso de agua, optimizando el consumo hídrico y mejorando la salud del cultivo.

De forma complementaria, se implementará un gemelo digital del invernadero, que actuará como una réplica virtual del sistema físico. Este modelo digital permitirá no solo visualizar el estado actual del invernadero, sino también anticipar sus necesidades mediante técnicas de predicción. Basándose en los datos recopilados por los sensores, así como en patrones históricos y variables climáticas, el gemelo digital utilizará modelos de series temporales e inteligencia artificial para estimar de forma óptima el momento y la duración del riego. Esta capa de análisis predictivo aportará un valor añadido al sistema, permitiendo una gestión más eficiente y proactiva de los recursos hídricos.

En definitiva, el objetivo de este trabajo es la creación de un sistema de riego inteligente que, mediante la automatización y la toma de decisiones basada en datos, permita una gestión más eficiente y sostenible del agua en entornos agrícolas controlados.

1.3. Tareas a desarrollar

A lo largo del desarrollo del presente proyecto se llevarán a cabo una serie de tareas fundamentales, organizadas de forma secuencial pero con cierto grado de solapamiento, en función del avance y las necesidades detectadas en cada fase. Las tareas principales serán:

1. **Diseño y construcción del gemelo físico**: Esta tarea incluye tanto el diseño de la estructura física del invernadero como la implementación del circuito electrónico que lo controla. Se seleccionarán sensores adecuados para la medición de variables ambientales (temperatura, humedad del aire y humedad del suelo) y actuadores como una bomba de agua y un ventilador, todos ellos alimentados y gestionados por una placa microcontroladora.

2. **Creación de la base de datos**: En esta fase se preparará el entorno software en una Raspberry Pi, configurando el sistema operativo y desplegando los servicios necesarios

mediante contenedores Docker. La tarea principal será la instalación y configuración de los servicios necesarios, como la base de datos donde se almacenan los datos recogidos por los sensores para su posterior análisis y visualización.

3. **Implementación del gemelo digital**: Utilizando los datos históricos almacenados en la base de datos, se implementará un sistema de predicción en Python que funciona como gemelo digital del invernadero. Este modelo será capaz de predecir, a partir de las condiciones ambientales actuales, cuántos segundos de riego serán necesarios en un minuto determinado. La implementación se basará en técnicas de regresión y aprendizaje supervisado.

4. **Validación del sistema**: Finalmente, se llevará a cabo un proceso de validación del sistema completo. Se realizarán pruebas tanto del hardware como del software para comprobar la fiabilidad de las mediciones, el correcto funcionamiento de los actuadores, la precisión del modelo predictivo y la estabilidad del sistema de almacenamiento y visualización de datos.

1.4. Metodología

Para la realización de este proyecto se ha seguido una metodología iterativa e incremental, que ha permitido avanzar en el desarrollo de forma flexible, adaptándose a los cambios que han surgido a lo largo del proceso. Dado el carácter académico y unipersonal del trabajo, se ha optado por una adaptación de la metodología ágil SCRUM, ajustada a las particularidades de un Trabajo de Fin de Grado realizado por una única desarrolladora.

SCRUM es un marco de trabajo ágil ampliamente utilizado para el desarrollo de productos complejos. Su enfoque se basa en dividir el proyecto en ciclos cortos llamados *sprints*, y cada *sprint* busca entregar una versión funcional del producto o sistema, fomentando así la mejora continua y la retroalimentación temprana. Aunque en SCRUM tradicional existen roles como el *Scrum Master*, el *Product Owner* y el equipo de desarrollo, en este proyecto dichos roles fueron asumidos de forma distribuida: la autora del TFG desempeñó el papel de desarrolladora y coordinadora del trabajo, mientras que los tutores actuaron como *Product Owners*, proponiendo objetivos y prioridades, y también como revisores del trabajo realizado [6].

La metodología se estructuró en *sprints*, cada uno centrado en el diseño, implementación

y validación de una funcionalidad concreta. Al final de cada *sprint* se celebraba una reunión de seguimiento con los tutores del proyecto, en la que se revisaban los avances, se detectaban posibles mejoras y se definían las tareas a desarrollar en adelante.

1.5. Estructura del documento

A continuación se listan los capítulos en los que se estructura este documento:

- *Capítulo 1. Introducción*
  En este capítulo se describen la motivación de este proyecto y tanto los objetivos generales como las tareas específicas a completar a lo largo de su realización, así como la metodología elegida para éste. También se detalla la estructura de este documento.

- *Capítulo 2. Estado del arte*
  Este capítulo presenta el contexto del proyecto y una breve investigación de otros proyectos ya existentes, y qué valor añadido aporta esta propuesta.

- *Capítulo 3. Tecnologías estudiadas*
  En esta sección se enumeran las tecnologías que se han estudiado para la realización del proyecto, incluyendo las que finalmente no han sido necesarias.

- *Capítulo 4. Diseño del sistema*
  Este capítulo explica de forma más detallada el diseño del sistema, tanto la primera opción que finalmente fue descartada, como la solución propuesta que acabaría siendo el resultado final del proyecto.

- *Capítulo 5. Descripción del sistema físico*
  Esta sección describe la realización del invernadero que constituye el gemelo físico del sistema, desde la fase de construcción del mismo hasta la enumeración de los componentes electrónicos utilizados.

- *Capítulo 6. Implementación*
  En este capítulo se muestra el proceso de implementación del sistema, ilustrado con las partes más importantes del código realizado.

- *Capítulo 7. Experimentación y pruebas*

  En este capítulo se enumeran y se explican algunas de las pruebas manuales que se han realizado en la fase de validación del proyecto.

- *Capítulo 8. Conclusiones y líneas futuras*

  Este capítulo final presenta las reflexiones posteriores a la realización del proyecto, además de diversas propuestas de líneas de extensión del mismo.

2

Estado del arte

Este capítulo recoge el análisis del estado actual de las tecnologías y enfoques relacionados con el proyecto. Se realiza un recorrido por los principales avances en inteligencia artificial aplicados a la agricultura, con especial atención a los gemelos digitales y su potencial en contextos agrícolas. También se estudian diversas iniciativas de agricultura de precisión y huertos inteligentes que han sido desarrolladas en los últimos años, destacando sus logros, limitaciones y grado de madurez. Finalmente, se identifica el valor diferencial que aporta el presente proyecto en comparación con estas propuestas existentes, tanto por su integración de componentes como por su enfoque predictivo basado en datos reales.

2.1. Inteligencia Artificial y agricultura

La Inteligencia Artificial (IA) es un campo de la informática que busca desarrollar sistemas capaces de realizar tareas que, normalmente, requieren inteligencia humana, tales como el razonamiento, el aprendizaje, la percepción y la toma de decisiones. Estas tareas incluyen la clasificación de datos, el reconocimiento de patrones, la percepción de comportamientos y la automatización de procesos complejos [7].

Dentro de la IA, una de las ramas más relevantes es el aprendizaje automático (machine learning), el cual permite que los sistemas informáticos aprendan automáticamente a partir de datos históricos y mejoren su rendimiento con el tiempo, sin necesidad de ser reprogramados explícitamente para cada nueva tarea. Otras ramas destacadas incluyen la lógica difusa, los sistemas expertos, el procesamiento del lenguaje natural (PLN) y las redes neuronales artificiales [8].

El impacto de la IA es especialmente significativo en sectores donde el análisis de grandes volúmenes de datos puede traducirse en mejoras sustanciales en eficiencia, sostenibilidad o rendimiento. En el ámbito agrícola, la aplicación de la IA ha dado lugar a lo que se conoce

como “agricultura inteligente” o “agricultura de precisión”, donde los datos recolectados por sensores se procesan con algoritmos que permiten optimizar los recursos disponibles. Esto resulta fundamental ante la escasez de agua, el cambio climático y la necesidad creciente de producción eficiente. La IA en este sector puede aplicarse a la predicción de plagas, detección de enfermedades, optimización de riego o gestión de nutrientes [9].

2.1.1. Gemelos digitales

Un concepto especialmente relevante en este contexto es el de gemelo digital. Un gemelo digital es una réplica virtual de un sistema físico que permite simular, predecir y optimizar su comportamiento en tiempo real [3]. Se nutre de datos reales obtenidos por sensores instalados en el sistema físico y utiliza modelos computacionales para representar su estado actual, así como sus posibles evoluciones futuras. En proyectos IoT, el gemelo digital permite integrar monitorización, automatización y análisis predictivo en un solo entorno digital, lo cual resulta especialmente útil para optimizar procesos, anticipar fallos o ajustar parámetros de funcionamiento antes de que ocurran en el mundo real.

En los últimos años, los gemelos digitales han comenzado a encontrar aplicaciones también en el ámbito agrícola. Aunque su adopción en este sector aún es incipiente, diversos estudios han identificado su potencial transformador. Los gemelos digitales permiten monitorizar en tiempo real las condiciones de cultivos, animales o maquinaria, simular posibles escenarios y optimizar decisiones operativas. Sin embargo, la mayoría de las implementaciones encontradas hasta ahora en agricultura están aún en fase conceptual o de prototipo, lo que indica una adopción limitada en comparación con sectores como el industrial o el energético. Aun así, destacan casos de uso en ganadería, apicultura, invernaderos y cultivos en vertical, con beneficios que incluyen la mejora de la sostenibilidad, la reducción de costes y la toma de decisiones más fundamentadas. Esto sugiere que, conforme se superen las barreras tecnológicas y económicas, los gemelos digitales pueden desempeñar un papel central en el futuro de la agricultura inteligente [10].

2.2. Agricultura de precisión y huertos inteligentes

Los huertos inteligentes son una implementación práctica de la agricultura de precisión a pequeña escala. Mediante la comunicación de sensores, actuadores y plataformas de análisis, es posible automatizar los cuidados de cultivo en función de las condiciones ambientales. Estos sistemas permiten controlar en tiempo real parámetros como la temperatura, la humedad del suelo, la humedad del aire, la iluminación y el riego. Su implementación mejora la sostenibilidad del cultivo, reduce el desperdicio de agua y nutrientes y disminuye la necesidad de supervisión constante.

Entre los proyectos más relevantes en este ámbito se encuentran iniciativas como OpenAg de MIT Media Lab, que busca democratizar la agricultura mediante cámaras de cultivo controladas digitalmente y la recopilación de datos ambientales para optimizar el crecimiento de las plantas [11]; o PlantVillage Nuru, desarrollada por Penn State University, que emplea Inteligencia Artificial y visión por computador para detectar enfermedades en cultivos mediante imágenes tomadas con smartphones, lo cual resulta especialmente útil en regiones agrícolas con escasos recursos tecnológicos [12].

En comparación, el proyecto que se presenta en esta memoria va un paso más allá al integrar en un único sistema tanto el control físico del riego como un gemelo digital predictivo capaz de anticipar la duración óptima del riego a partir de las condiciones ambientales, usando aprendizaje automático. Esto le otorga una capacidad de toma de decisiones autónoma basada en datos históricos que no suele estar presente en muchos de los sistemas anteriores, más centrados en la monitorización y control en tiempo real.

2.3. Valor añadido del proyecto

Este proyecto propone una solución integral de huerto inteligente que combina sensorización, control automatizado, almacenamiento de datos, visualización en tiempo real de los mismos y predicción de comportamiento futuro. A diferencia de otros trabajos, como los mencionados anteriormente, el valor añadido principal radica en la integración de un gemelo digital que no solo refleja el estado actual del invernadero, sino que, además, realiza predicciones mediante técnicas de regresión sobre cuánto tiempo debe regarse la planta dadas unas condiciones ambientales concretas.

El sistema está diseñado con componentes accesibles, económicos y ampliamente documentados, lo cual facilita su replicabilidad. A esto se suma una arquitectura modular basada en microservicios, que permite actualizar, ampliar o sustituir fácilmente partes del sistema.

3

Tecnologías estudiadas

En este capítulo se presentan las principales tecnologías estudiadas y utilizadas en el desarrollo del proyecto. Se describen tanto los componentes hardware empleados para la sensorización y control del sistema, como las herramientas software que permiten el almacenamiento, visualización y análisis de los datos. Además, se analizan brevemente las razones que justifican su elección y su papel dentro de la arquitectura general. A lo largo del capítulo se expone cómo cada tecnología contribuye a la creación de un sistema de riego inteligente, desde la captura de datos hasta la toma de decisiones mediante predicción.

3.1. Arduino / Wemos D1 R1

Arduino es una plataforma de hardware libre basada en microcontroladores programables mediante un entorno de desarrollo propio [13]. Para este proyecto se ha utilizado una placa Wemos D1 R1, que combina la compatibilidad con el entorno Arduino IDE con un módulo WiFi integrado (ESP8266), lo cual permite enviar datos de sensores de forma inalámbrica sin necesidad de módulos adicionales. Este microcontrolador gestiona la lectura de sensores, el control de la bomba de riego mediante un relé y la ventilación del invernadero, todo ello en tiempo real [14].

3.2. Raspberry Pi

La Raspberry Pi es un microordenador de bajo coste y gran versatibilidad que actúa como nodo central del sistema. En este caso se ha utilizado una Raspberry Pi 3 Model B, que corre un sistema operativo basado en Linux (Raspbian OS) y permite ejecutar múltiples servicios

simultáneamente mediante contenedores Docker. Su bajo consumo energético y facilidad de configuración lo convierten en una plataforma ideal para proyectos IoT. En este proyecto, la Raspberry Pi aloja servicios como InfluxDB, Mosquitto, Grafana, Node-RED y el módulo de predicción en Python [15].

3.3. InfluxDB

InfluxDB es una base de datos NoSQL orientada a series temporales. Es especialmente eficiente para almacenar grandes cantidades de datos con marca temporal, como las lecturas periódicas de sensores. Permite realizar consultas optimizadas por tiempo, agregaciones, y análisis históricos. En este proyecto se utiliza InfluxDB para almacenar temperatura, humedad del aire, humedad del suelo y duración del riego en segundos. Esta base de datos permite luego alimentar al sistema de predicción y visualizar los datos en Grafana [16].

3.4. Grafana

Grafana es una herramienta de visualización de datos en tiempo real. Se conecta con múltiples orígenes de datos, incluido InfluxDB, y permite crear *dashboards* personalizados e interactivos. En este proyecto, se ha configurado un *dashboard* donde el usuario puede consultar en todo momento la evolución de las variables medidas por los sensores, así como los segundos de riego acumulados. Esto no solo facilita el seguimiento del invernadero, sino que también sirve como base para detectar anomalías o validar el comportamiento del modelo predictivo [17].

3.5. Mosquitto (MQTT)

Mosquitto es un broker de mensajería que implementa el protocolo MQTT (*Message Queuing Telemetry Transport*), ampliamente utilizado en sistemas IoT por su eficiencia y bajo consumo de ancho de banda. En este proyecto, los datos de los sensores son enviados desde la placa Wemos a través del protocolo MQTT a la Raspberry Pi. Esto permite una comunicación asíncrona y desacoplada entre dispositivos, garantizando la flexibilidad y robustez del sistema [18].

3.6. Node-RED

Node-RED es una herramienta de desarrollo basada en flujo para la programación visual de sistemas conectados. Permite recibir los mensajes MQTT, transformarlos, visualizar su contenido e insertarlos directamente en InfluxDB mediante nodos configurables. Su enfoque visual facilita el diseño del flujo de datos y la gestión del sistema sin necesidad de escribir código extenso. Además, puede integrarse fácilmente con alertas, interfaces web o acciones automatizadas [19].

3.7. Python

Python es un lenguaje de programación interpretado y de alto nivel que destaca por su simplicidad y versatibilidad. En este proyecto, se ha utilizado para implementar el gemelo digital, encargado de realizar predicciones de duración óptima del riego. Usando bibliotecas como `pandas` para manejo de datos [20], `scikit-learn` para entrenar modelos de regresión [21] y `numpy` para operaciones matemáticas [22], el sistema analiza datos históricos del invernadero y determina, para cada conjunto de condiciones ambientales, cuántos segundos se debería regar en un minuto determinado [23].

3.8. Docker

Docker es una plataforma que permite empaquetar y ejecutar aplicaciones en contenedores ligeros y portables. En este proyecto se ha utilizado para todos los servicios del servidor IoT, como InfluxDB, Mosquitto, Grafana y Node-RED. Gracias a Docker, el sistema puede desplegarse de forma sencilla, garantizando que cada servicio funcione de forma aislada y sin conflictos de dependencias. Además, su uso facilita la replicabilidad y mantenimiento del sistema [24].

4

Diseño del sistema

El sistema desarrollado en este proyecto tiene como objetivo monitorizar las condiciones ambientales dentro de un invernadero y activar de forma automatizada el riego y la ventilación, además de almacenar los datos para su posterior análisis y predicción. Para lograrlo, se diseñó una arquitectura modular que integra sensores, actuadores, una placa microcontroladora, una Raspberry Pi como servidor IoT y un entorno software distribuido. El diseño del sistema propuesto ha evolucionado a lo largo del desarrollo del proyecto, adaptándose a las limitaciones prácticas observadas durante la fase de implementación. Inicialmente, se planteó una arquitectura basada en una placa Wemos D1 R1, que incluye conectividad WiFi integrada y permite enviar datos directamente a la Raspberry Pi a través de MQTT. Sin embargo, durante las pruebas se detectaron problemas de estabilidad y compatibilidad de pines que afectaban al funcionamiento correcto de los sensores y del módulo de riego. Como respuesta, se diseñó una segunda opción, basada en una placa Arduino UNO, sin conectividad inalámbrica, pero que se comunica con la Raspberry Pi mediante un enlace USB Serial. Ambas arquitecturas comparten objetivos funcionales similares: capturar parámetros ambientales en tiempo real, accionar el sistema de riego y transmitir los datos a una base de datos en la Raspberry Pi. A continuación se describen ambas opciones.

4.1. Opción A: Wemos D1 R1 y comunicación WiFi

En esta primera versión, la placa microcontroladora Wemos D1 R1 actúa como núcleo del sistema físico. A ella se conectan dos sensores: un sensor DHT22 para medir la temperatura y la humedad del aire, y un sensor resistivo analógico para medir la humedad del suelo. Además, se controlan dos actuadores: una bomba de agua, activada mediante un relé, y una hélice para ventilación.

La comunicación con la Raspberry Pi se realiza a través de WiFi utilizando el protocolo

MQTT. La placa Wemos se conecta a la red doméstica, establece una conexión con el broker Mosquitto y publica periódicamente los datos en diferentes *topics*. Estos datos incluyen temperatura, humedad del aire, humedad del suelo y duración del riego en segundos acumulados cada minuto. En la figura 1 se muestra un diagrama del flujo de datos diseñado para esta opción.

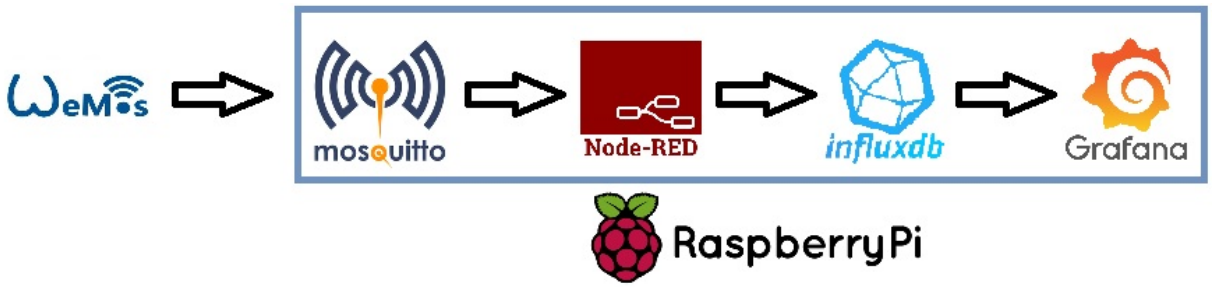

Figura 1: Diagrama del flujo de datos diseñado para la opción A

La implementación se realizó mediante el entorno Arduino IDE, usando bibliotecas específicas como `DHT.h` para el sensor de temperatura y humedad, `ESP8266WiFi.h` para la conexión WiFi, y `PubSubClient.h` para el manejo del protocolo MQTT. El flujo de ejecución de la placa está dividido en tres tareas principales: lectura de sensores, activación de actuadores y envío de datos. La lectura de sensores se realiza cada 5 segundos en el caso del DHT22, y cada 1 segundo para la humedad del suelo. La bomba de agua se activa automáticamente si la humedad del suelo es baja, y se mantiene encendida durante un tiempo determinado. Cada 60 segundos, el sistema acumula los segundos que ha estado regando y los publica junto con el resto de parámetros al servidor. Este fue un comportamiento básico que se usó para las pruebas iniciales, antes de incluir el gemelo digital.

En la Raspberry Pi, los datos recibidos a través de MQTT son tratados con Node-RED. Cada mensaje publicado por el Wemos es recibido por un nodo `mqtt in`, procesado con nodos `function` para adaptarlo al formato de InfluxDB, y enviado a la base de datos mediante un nodo `influxdb out`. En la figura 2 se muestra dicho flujo de datos en Node-RED. Estos datos se almacenan con marca temporal y posteriormente son visualizados con Grafana. En la figura 3 se muestra un ejemplo de cómo es la visualización de datos en Grafana, concretamente de la temperatura. La predicción sobre el tiempo óptimo de riego se realiza en otro módulo independiente en Python.

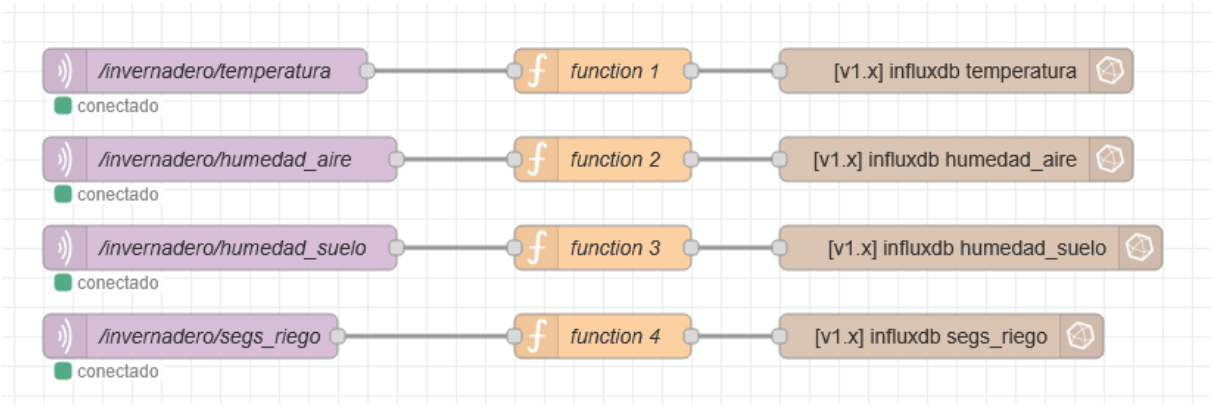

Figura 2: Flujo de datos en Node-RED

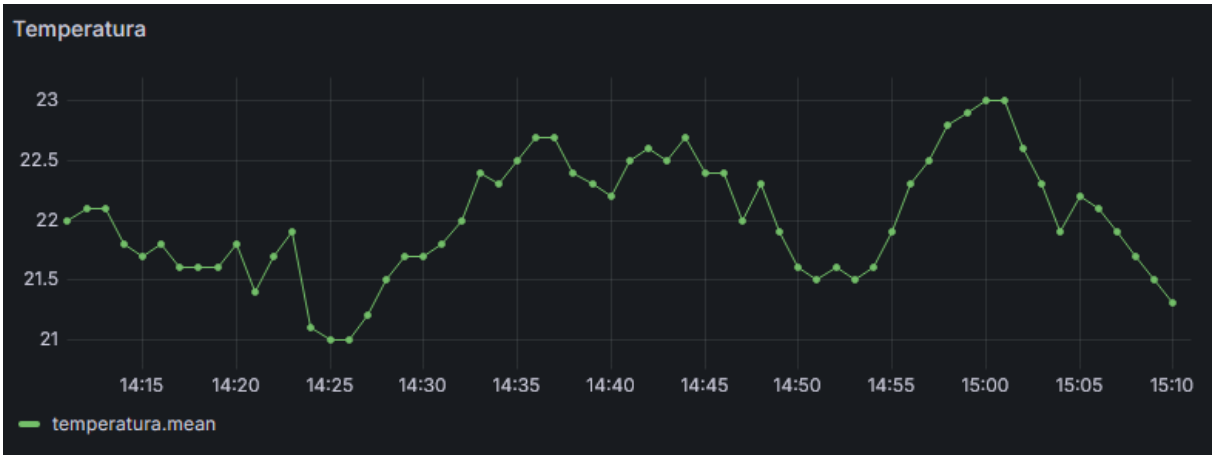

Figura 3: Visualización de datos de temperatura en Grafana

Durante la implementación, esta opción presentó algunos problemas: lecturas erróneas del sensor DHT22 (valores *NaN* o extremos), bloqueos del sistema tras activar el relé, y reinicios aleatorios de la placa. Se detectó que la causa de estos errores podía ser que los pines digitales de la placa Wemos trabajan a 3.3V, mientras que el relé necesita 5V. Es por ello que al activar el relé se bloqueaba el sistema y dejaban de funcionar correctamente el resto de componentes.

4.2. Opción B: Arduino UNO y comunicación por USB Serial

Como alternativa más estable, se rediseñó el sistema utilizando una placa Arduino UNO en lugar de la Wemos. Esta placa no tiene conectividad WiFi, por lo que se implementó una comunicación cableada entre la placa Arduino y la Raspberry Pi mediante el puerto USB, usando el protocolo Serial.

El resto del diseño físico permanece igual: los sensores y actuadores se conectan directamente al Arduino, que ejecuta la lógica de lectura y control. La diferencia clave está en cómo se transmiten los datos. En lugar de usar MQTT, el Arduino envía, cada 60 segundos, una cadena de texto a través del puerto Serial con los valores separados por comas (temperatura, humedad del aire, humedad del suelo, segundos de riego). Esta cadena de texto será posteriormente interpretada por un script de Python, encargado a su vez de enviar los datos a InfluxDB. En la figura 4 se muestra un diagrama del flujo de datos diseñado para esta alternativa.

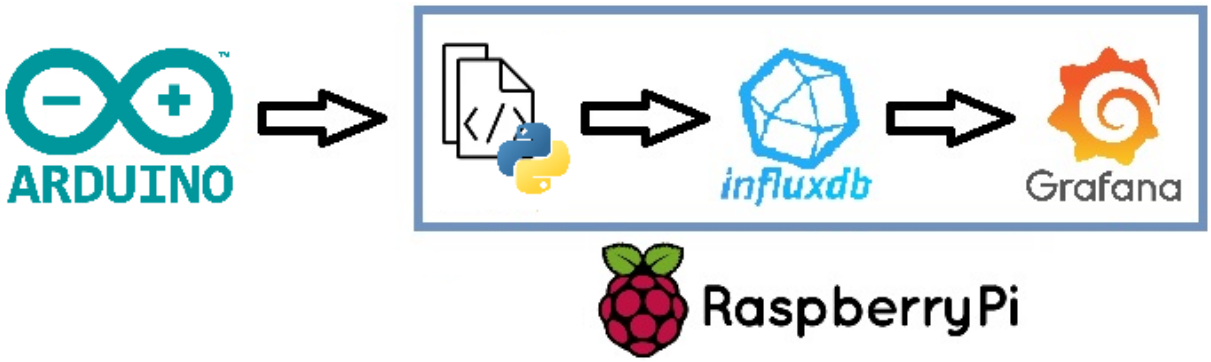

Figura 4: Diagrama del flujo de datos diseñado para la opción B

En esta opción, el programa del Arduino también se desarrolló en el IDE de Arduino, con pequeñas adaptaciones. Se eliminó el uso de las librerías de red y MQTT, y se incorporó `Serial.print()` para transmitir los datos al ordenador. El código mantiene la estructura modular: sensores leídos a intervalos definidos, activación automática del sistema de riego si la humedad es baja, y cálculo de los segundos de riego mediante temporización con `millis()`.

En la Raspberry Pi, un script en Python "escucha" continuamente el puerto Serial, interpreta cada línea de datos recibida y lo inserta en la base de datos InfluxDB. Este script usa las librerías `pyserial` para la lectura desde el puerto [25] e `influxdb` para la escritura en la base de datos [26]. El flujo de datos ya no pasa por Node-RED ni MQTT, aunque estas herramientas pueden seguir empleándose para visualización u otros fines.

Esta implementación resolvió los problemas de estabilidad observados con la Wemos. Al no depender del módulo WiFi ni del sistema de interrupciones de la ESP8266, el Arduino UNO permite un control más fiable de los actuadores y lecturas más estables. Además, la comunicación por USB garantiza una conexión persistente sin necesidad de configuración de red. El único inconveniente es la falta de comunicación inalámbrica, pero esto se compensa con una

mayor robustez y menor complejidad técnica.

5

Descripción del sistema físico

Este capítulo describe la construcción y configuración del gemelo físico del sistema desarrollado, es decir, el invernadero real que alberga los sensores, actuadores y la electrónica necesaria para su automatización. La finalidad de este módulo físico es recoger datos ambientales en tiempo real y ejecutar acciones de riego y ventilación de forma autónoma, en función de las condiciones detectadas y/o de las predicciones del gemelo digital.

5.1. Estructura física del invernadero

El invernadero está construido a partir de una estructura rígida de pequeñas dimensiones (formato maqueta), recubierta con láminas de metacrilato transparente para simular un entorno cerrado. Este espacio alberga una maceta con sustrato vegetal y se ha diseñado específicamente para permitir la integración de componentes electrónicos sin que estos afecten al cultivo.

Dentro del invernadero se han distribuido los sensores y actuadores de forma estratégica:

- El sensor de humedad del suelo se inserta directamente en la tierra para recoger datos reales de humedad.
- El sensor de temperatura y humedad del aire (DHT22) se sitúa en la parte superior, simulando la capa de aire sobre el cultivo.
- La bomba de agua se encuentra fuera del invernadero, y cuenta con una manguera para suministrar agua al sustrato desde un pequeño depósito.

- Un ventilador de pequeño tamaño se ubica en una de las paredes laterales, simulando un sistema de ventilación automática. También se han dispuesto varios LEDs a lo largo de la parte superior del invernadero a modo de iluminación.

En la figura 5 se puede ver una foto real del sistema físico realizado para este proyecto.

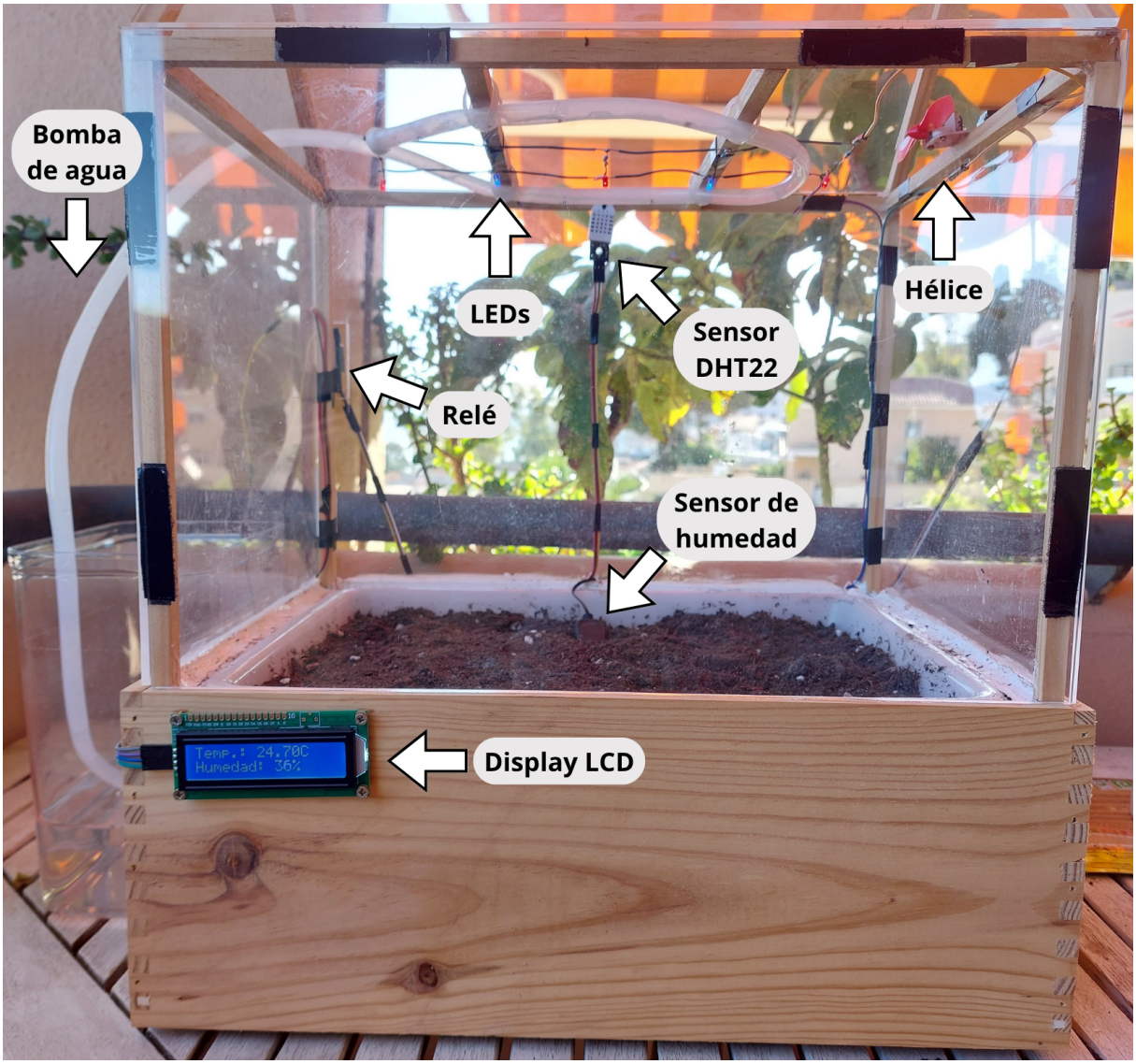

Figura 5: Imagen del invernadero automatizado (indicando los componentes del sistema)

5.2. Componentes electrónicos

En el cuadro 1 se enumeran y describen los principales componentes electrónicos empleados:

Componente	Descripción
Placa Arduino / Wemos	Microcontrolador principal. Se probó con Wemos D1 R1 (WiFi) y Arduino UNO.
Sensor DHT22	Mide temperatura y humedad del aire.
Sensor de humedad resistivo	Mide humedad del suelo a través de la conductividad del sustrato.
Relé de 1 canal	Controla la bomba de agua.
Bomba de agua	Actúa como sistema de riego automatizado.
Ventilador (hélice)	Simula la ventilación automática en función de la temperatura.
Pantalla LCD I2C 16x2	Muestra en tiempo real los valores de los sensores.
Fuente de alimentación	Caja de 4 pilas AA (6V) para alimentar la bomba de forma independiente.

Cuadro 1: Componentes del sistema físico

5.3. Conexiones del circuito

El sistema se ha cableado de forma modular para facilitar su mantenimiento y pruebas. Las conexiones más relevantes son:

- **Sensores**:
  - El DHT22 se conecta al pin digital 8 del Arduino. El pin de datos del sensor se conecta a su vez a 5V mediante una resistencia de 10 KΩ.
  - El sensor de humedad del suelo se conecta a la entrada analógica A0.
- **Actuadores**:
  - El relé de la bomba de agua recibe señal desde el pin digital 7. La bomba se conecta a la fuente de 6V a través del relé (pin “normalmente abierto”).
  - El ventilador se controla mediante los pines INA e INB (modo *forward*/*reverse*, aunque solo se utiliza *forward*), conectados a los pines digitales 5 y 6 del Arduino, respectivamente.

- Los LEDs de la parte superior de la estructura se conectan en paralelo al pin 10.

- **Display LCD**: La pantalla LCD se conecta mediante interfaz I2C (SDA y SCL) a los pines digitales específicos para ello.

- **GND común**: Todos los GND están unificados para garantizar la estabilidad del sistema.

En la figura 6 se muestra un diagrama con todas las conexiones anteriormente mencionadas.

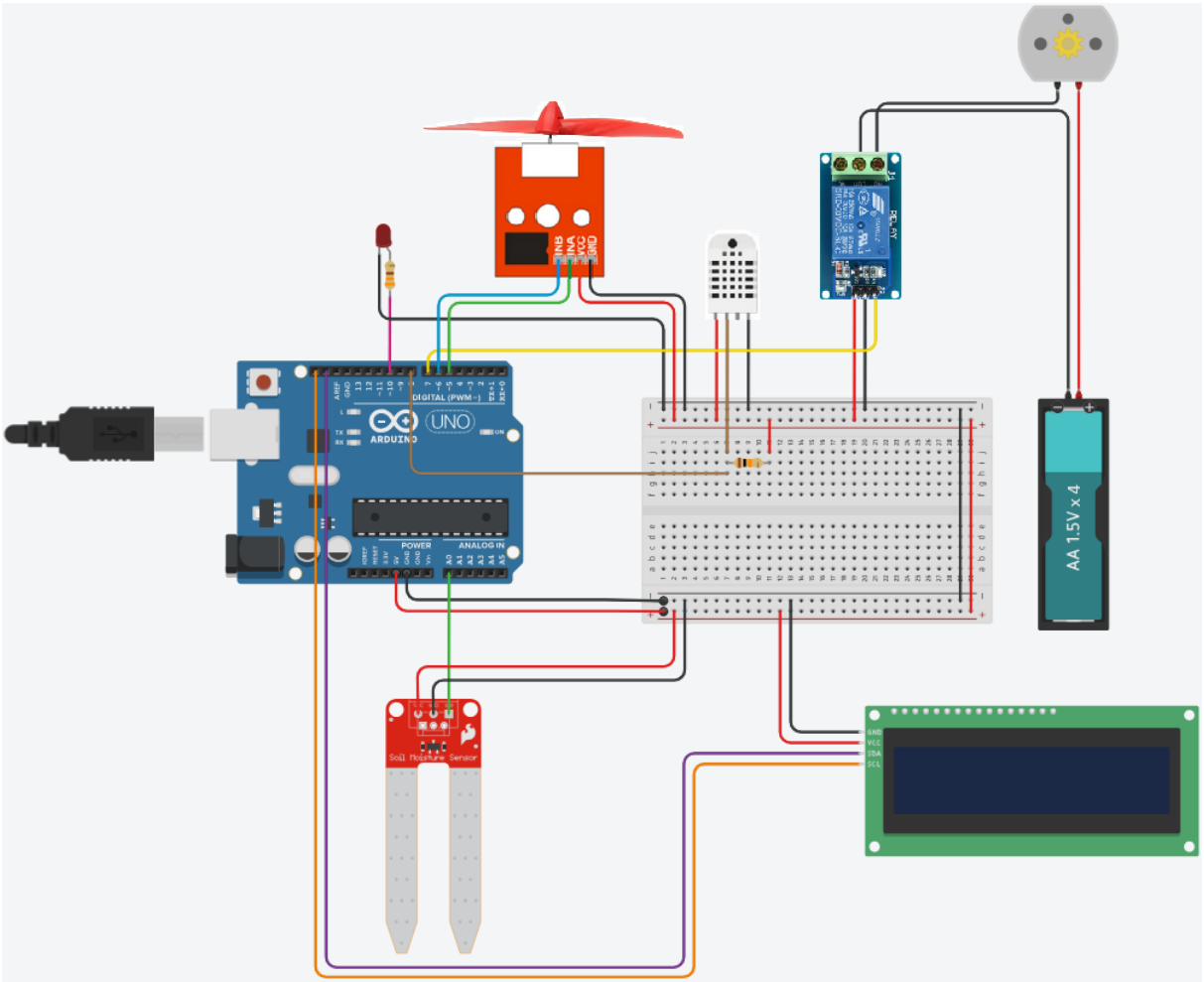

Figura 6: Diagrama de conexiones del sistema físico

5.4. Alimentación

Debido al elevado número de sensores y actuadores presentes en el sistema, la placa Arduino no es capaz de alimentar por sí sola a todos los componentes. Por este motivo, y para

asegurar una alimentación estable y evitar problemas eléctricos, se han tomado las siguientes medidas:

- El Arduino recibe alimentación a través del puerto USB (desde la Raspberry Pi).
- La bomba, al requerir más potencia que la que el Arduino puede proporcionar, se alimenta de forma independiente mediante una caja de 4 pilas AA (6V en total).
- Se ha tenido especial cuidado en evitar bucles de corriente o interferencias mediante la implementación de una masa común entre los distintos circuitos.

6

Implementación

Esta sección describe con detalle cómo se ha llevado a cabo la implementación del sistema propuesto, presentando las dos alternativas desarrolladas a lo largo del proyecto: la opción A, basada en conectividad WiFi mediante la placa Wemos D1 R1; y la opción B, centrada en la comunicación por puerto serie mediante Arduino UNO. Aunque ambas opciones han sido implementadas y probadas, la opción B se ha consolidado como la definitiva debido a su mayor estabilidad en entornos reales.

A lo largo de esta sección se incluirán fragmentos de código representativos, con el objetivo de ilustrar el funcionamiento del sistema, el almacenamiento de datos, y el modelo de predicción del gemelo digital.

6.1. Lógica de control del invernadero

Tanto en la opción A (Wemos y WiFi) como en la opción B (Arduino y USB Serial), el microcontrolador sigue la misma lógica para controlar en invernadero. El objetivo principal es monitorizar en tiempo real las variables ambientales (temperatura, humedad del aire y humedad del suelo), controlar automáticamente los actuadores (ventilador y bomba de agua) y enviar los datos periódicamente al servidor (Raspberry Pi).

El sistema parte de un bloque de inicialización donde se configuran los pines, se inicia la comunicación serie y se inicializan los sensores y la pantalla LCD, como puede verse en el *listing* 1.

```
1 void setup() {
2   Serial.begin(9600);
3
4   pinMode(LED, OUTPUT);
```

```
  pinMode(INA, OUTPUT);
  pinMode(INB, OUTPUT);
  pinMode(MOTOR, OUTPUT);
  digitalWrite(MOTOR, HIGH); // Apagado por defecto

  dht.begin();
  delay(5000); // Estabilización del sensor DHT
  lcd.init();
  lcd.backlight(); // Activar la retroiluminación
}
```

Listing 1: Inicialización del sistema

A partir de ahí, el programa entra en el bucle principal `loop()`, donde se repiten los siguientes bloques fundamentales de forma periódica: la lectura de los sensores y el control de los actuadores, tanto el subsistema de la temperatura como el del riego.

6.1.1. Lectura de sensores

Cada segundo se actualiza la lectura de humedad del suelo usando un sensor resistivo. El valor analógico se transforma a porcentaje (de 0 a 100), como puede verse en el *listing* 2.

```
if (ahora - ultLecturaH >= 1000) {
  ultLecturaH = ahora;
  hum_suelo = analogRead(SENSOR_HUM);
  hum_pct = map(hum_suelo, 1024, 0, 0, 100); // Conversión a %
  ...
}
```

Listing 2: Lectura del sensor de humedad del suelo cada 1 seg.

Cada 5 segundos se actualizan la temperatura y la humedad del aire mediante un sensor DHT22. Es recomendable esperar un tiempo prudencial (unos segundos) entre dos lecturas consecutivas del sensor DHT22 para evitar que se desestabilice, y así asegurar valores fiables. Además, se comprueba que los datos no sean *NaN* antes de usarlos, como se muestra en el *listing* 3.

```
if (ahora - ultLecturaT >= 5000) {
  ultLecturaT = ahora;
  float t = dht.readTemperature(); // Temperatura en ºC
  float h = dht.readHumidity(); // Humedad del aire en %
  if (!isnan(t) && !isnan(h)) {
    temp = t;
    hum_aire = h;
  }
  ...
}
```

Listing 3: Lectura del sensor DHT22 (temperatura y humedad del aire) cada 5 seg.

6.1.2. Control de actuadores

Según la temperatura, se activa o desactiva la hélice de ventilación y los LEDs. En este caso, se ha establecido que el umbral sea de 25ºC, como puede verse en el *listing* 4.

```
void subsTemperatura(float temp) {
  if (temp <= 25.0) {
    digitalWrite(LED, HIGH); // LED encendido
    digitalWrite(INA, LOW); // Hélice apagada
    digitalWrite(INB, LOW);
  } else {
    digitalWrite(LED, LOW); // LED apagado
    digitalWrite(INA, HIGH); // Hélice encendida
    digitalWrite(INB, LOW);
  }
}
```

Listing 4: Subsistema temperatura

Cuando la humedad del suelo baja del 30 %, se activa la bomba de agua durante 5 segundos, tras los cuales se apaga. En el *listing* 5 se muestra también el cálculo de los segundos de que está activada la bomba de agua.

```
void subsRiego(int hum_pct) {
  unsigned long ahora = millis();

  if (hum_pct <= 30 && !motor_ON) {
    digitalWrite(MOTOR, LOW); // Encender bomba
    tiempo_inicio_riego = ahora;
    motor_ON = true;
  }

  if (motor_ON && (ahora - tiempo_inicio_riego >= 5000)) {
    digitalWrite(MOTOR, HIGH); // Apagar bomba
    segs_riego += 5.0; // Añadir 5 segundos al acumulado
    motor_ON = false;
  }
}
```

Listing 5: Subsistema riego

Como ya se ha mencionado, esta lógica es similar en ambas implementaciones del proyecto (opción A y opción B). La principal diferencia entre ellas es el modo en que se envían los datos a la Raspberry, y la forma en que ésta los interpreta. A continuación se explican dichas diferencias.

6.2. Opción A: Wemos D1 R1 y comunicación WiFi

La primera opción implementada utilizaba una placa Wemos D1 R1 (basada en el chip ESP8266) como microcontrolador principal. Esta placa permitía la conexión WiFi directa con la Raspberry Pi, que actuaba como servidor. La lectura de sensores (temperatura, humedad del aire y humedad del suelo) se realizaba desde el firmware de Arduino y los datos se publicaban mediante el protocolo MQTT al broker Mosquitto, alojado en la Raspberry Pi. En el *listing* 6 se muestra cómo se enviaban los datos mediante MQTT.

El flujo de datos se diseñó con Node-RED, que recibía los mensajes MQTT, los transformaba y almacenaba en InfluxDB. A su vez, se visualizaban en un *dashboard* de Grafana.

```
void sendData() {
```

```
  ...

  // Conversión a string:
  char temp_str[6], hum_air_str[6], hum_suelo_str[6], segs_riego_str[6];
  dtostrf(temp, 4, 1, temp_str);
  dtostrf(hum_aire, 4, 1, hum_air_str);
  itoa(hum_pct, hum_suelo_str, 10);
  dtostrf(segs_riego, 4, 1, segs_riego_str);

  client.publish(topic_temp, temp_str);
  client.publish(topic_hum_aire, hum_air_str);
  client.publish(topic_hum_suelo, hum_suelo_str);
  client.publish(topic_segs_riego, segs_riego_str);

  segs_riego = 0.0; // Reinicio del contador de riego
}
```

Listing 6: Envío de datos por MQTT desde Wemos

A pesar de funcionar correctamente en condiciones ideales, esta opción presentaba problemas de estabilidad en el control del relé y consumo de pines del ESP8266, así como reinicios esporádicos bajo ciertas cargas de sensores. Además, se observó comportamiento errático del sistema al activar la bomba de agua, lo cual motivó la exploración de una alternativa cableada más estable.

6.3. Opción B: Arduino y comunicación Serial

La segunda opción consiste en sustituir la Wemos por una placa Arduino UNO, más robusta eléctricamente y con comportamiento más predecible. Dado que Arduino no cuenta con WiFi integrado, la comunicación con la Raspberry Pi se ha implementado mediante USB Serial.

El Arduino se encarga de recopilar datos de sensores y controlar el actuador de riego (bomba), enviando cada minuto un paquete de datos con la estructura mostrada en el *listing* 7.

```
void sendData() {
  ...

```

```
  Serial.print(temp, 1);
  Serial.print(",");
  Serial.print(hum_aire, 1);
  Serial.print(",");
  Serial.print(hum_pct);
  Serial.print(",");
  Serial.println(segs_riego, 1);

  segs_riego = 0.0; // Reinicio del contador de riego
}
```

Listing 7: Envío de datos por Serial desde Arduino

En la Raspberry Pi, se ejecuta un script en Python que lee el puerto serie, procesa los datos recibidos y los inserta en InfluxDB, como puede verse en el *listing* 8.

```
line = ser.readline().decode('utf-8').strip()
temp, hum_aire, hum_suelo, segs_riego = map(float, line.split(","))
json_body = [
    {"measurement": "temperatura", "fields": {"valor": temp}},
    {"measurement": "humedad_aire", "fields": {"valor": hum_aire}},
    {"measurement": "humedad_suelo", "fields": {"valor": hum_suelo}},
    {"measurement": "segs_riego", "fields": {"valor": segs_riego}}
]
client.write_points(json_body)
```

Listing 8: Lectura de datos desde Arduino y almacenamiento en InfluxDB

Este enfoque permite una transmisión más estable de los datos, sin depender del WiFi ni de MQTT. Los datos almacenados en InfluxDB se visualizan igualmente en Grafana.

6.4. Implementación del gemelo digital y modelo predictivo

Una de las partes más relevantes del proyecto es la implementación del gemelo digital, cuyo objetivo es predecir la cantidad de segundos que debe activarse el sistema de riego en función de las condiciones ambientales del invernadero: temperatura, humedad del aire y humedad del suelo. Para lograrlo, se ha desarrollado un script en Python basado en aprendizaje supervisado,

concretamente un modelo de regresión lineal.

6.4.1. Obtención y preparación de los datos

El script comienza conectándose a la base de datos InfluxDB, donde se almacenan los datos históricos del sistema recogidos por los sensores. Estos datos están distribuidos en distintas *measurements* (series temporales): *temperatura*, *humedad_aire*, *humedad_suelo* y *segs_riego*. Como puede verse en el *listing* 9, se ha implementado una función para obtener el *DataFrame* de un *measurement* dado.

```
client = influxdb.InfluxDBClient(host='localhost', port=8086, database='
    invernadero')

def get_df(measurement):
    query = f"SELECT valor FROM {measurement} WHERE time > now() - 90d"
    result = client.query(query)
    data = list(result.get_points())
    df = pd.DataFrame(data)
    df.rename(columns={'valor': measurement}, inplace=True)
    return df.set_index("time")
```

Listing 9: Conexión a InfluxDB y función para obtener el *DataFrame* de un *measurement*

Una vez se obtiene un *DataFrame* para cada variable, se unen todas mediante un `join` interno, asegurando que solo se consideren los instantes de tiempo donde todas las mediciones están presentes simultáneamente. Después, se renombran las columnas para mayor claridad. Todo ello puede verse en el *listing* 10.

```
df_temp = get_df("temperatura")
df_hum_aire = get_df("humedad_aire")
df_hum_suelo = get_df("humedad_suelo")
df_riego = get_df("segs_riego")

df = df_temp.join([df_hum_aire, df_hum_suelo, df_riego], how="inner")
df.columns = ['temperatura', 'humedad_aire', 'humedad_suelo', 'segs_riego']
df.dropna(inplace=True)
```

Listing 10: Generación y unión de los DataFrames

6.4.2. Entrenamiento del modelo predictivo

Una vez preprocesados los datos, se define la variable de salida y (segundos de riego) y las variables de entrada X (condiciones ambientales). Se divide el conjunto de datos en entrenamiento (80 %) y prueba (20 %) y se entrena un modelo de regresión lineal usando `scikit-learn`. Además, el modelo entrenado se guarda en disco con `joblib` para poder reutilizarlo más adelante sin necesidad de reentrenarlo. En el *listing* 11 se muestra todo el proceso de entrenamiento del modelo.

```
X = df[['temperatura', 'humedad_aire', 'humedad_suelo']]
y = df['segs_riego']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

joblib.dump(model, "model.pkl")
```

Listing 11: Entrenamiento y guardado del modelo predictivo

6.4.3. Evaluación del modelo

Tras el entrenamiento, el modelo se evalúa sobre los datos de prueba para conocer su precisión. Se calcula el error absoluto medio (MAE) y el coeficiente de determinación (R^2), que mide qué porcentaje de la varianza del valor de salida está explicado por las variables de entrada, como puede verse en el *listing* 12.

```
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
```

```
r2 = r2_score(y_test, y_pred)

print(f"Error absoluto medio (MAE): {mae:.2f} segundos")
print(f"Coeficiente de determinación (R²): {r2:.2f}")
```

Listing 12: Evaluación del modelo y obtención del MAE y R^2

6.4.4. Predicción y envío del resultado

El gemelo digital realiza una predicción para un conjunto de condiciones de entrada (por ejemplo, 25ºC, 60 % de humedad del aire y 30 % de humedad del suelo). Esta predicción se envía de vuelta a InfluxDB para su posterior visualización o análisis. Finalmente, como se muestra en el *listing* 13, se cierra la conexión con la base de datos.

```
nueva_condicion = np.array([[25.0, 60.0, 30.0]])  # Condiciones de ejemplo
prediccion = model.predict(nueva_condicion)[0]

print(f"Predicción: Se debe regar {prediccion:.2f} segundos")

json_body = [
    {
        "measurement": "predicciones_riego",
        "fields": {
            "segundos_riego_predicho": float(prediccion)
        }
    }
]
client.write_points(json_body)
client.close()
```

Listing 13: Predicción con condiciones ambientales de ejemplo

6.4.5. Resumen

Este script representa el núcleo del sistema de inteligencia del proyecto. Gracias a su entrenamiento con datos reales, el gemelo digital es capaz de realizar recomendaciones de riego

más precisas que un sistema basado únicamente en umbrales fijos. Además, su modularidad permite reentrenarlo fácilmente conforme se disponga de más datos, haciendo que su precisión mejore con el tiempo.

Todo el código implementado puede ser consultado en el repositorio de GitHub creado específicamente para este TFG [27].

7

Experimentación y pruebas

Para verificar el correcto funcionamiento del sistema diseñado e implementado, se realizaron diversas pruebas manuales sobre el prototipo físico. Estas pruebas tuvieron como objetivo validar el comportamiento esperado del sistema en distintos escenarios simulados, centrándose principalmente en el subsistema de riego automático, que constituye el componente clave del proyecto.

Las pruebas se centraron en evaluar si el sistema es capaz de tomar decisiones correctas en función de las condiciones ambientales recogidas por los sensores (humedad del suelo, temperatura y humedad del aire), así como comprobar la estabilidad general del sistema y la fiabilidad del flujo de datos desde Arduino hasta la base de datos en la Raspberry Pi.

A continuación se describen algunas de las pruebas más relevantes llevadas a cabo sobre el sistema final.

7.1. Activación del riego con suelo seco

Condición de partida

El sensor de humedad del suelo se introdujo en tierra seca (valor de humedad del suelo <30 %). La bomba de agua estaba conectada y alimentada, y el sistema Arduino se encontraba en ejecución y enviando datos a la Raspberry Pi.

Comportamiento observado

Al detectar un nivel de humedad bajo, el sistema activó automáticamente la bomba de agua durante un tiempo aproximado de 5 segundos, como estaba programado. Este encendido fue

reconocible tanto por el sonido de la bomba como por el LED indicador que se iluminó en el relé durante la activación. El contador de segundos de riego se incrementó y se transmitió correctamente por USB Serial a la Raspberry, almacenándose en InfluxDB.

Conclusiones

El sistema reacciona adecuadamente ante condiciones de suelo seco, ejecutando el riego de forma automática y precisa, y registrando correctamente los datos. Esta prueba valida el correcto funcionamiento del sensor de humedad del suelo, del relé de activación de la bomba y del proceso de transmisión de datos.

7.2. No activación del riego con suelo húmedo

Condición de partida

El sensor de humedad del suelo se humedeció manualmente con agua antes de introducirlo en la tierra, simulando un escenario de suelo húmedo (valor de humedad >50 %).

Comportamiento observado

El sistema detectó un nivel alto de humedad y no activó la bomba en ningún momento. Tampoco se registraron segundos de riego durante el intervalo de prueba. Los valores se visualizaron correctamente en el LCD y se almacenaron sin errores en la base de datos.

Conclusiones

Se confirma que el sistema no activa innecesariamente la bomba cuando la humedad del suelo está dentro de los valores adecuados, evitando el desperdicio de agua. Este comportamiento demuestra la efectividad del umbral configurado y de la lógica de control.

7.3. Respuesta del sistema ante cambios ambientales

Condición de partida

Se encendió una fuente de calor cerca del sensor DHT22 para elevar la temperatura ambiente, y se midió si el sistema respondía también con la activación del ventilador (hélice), sin

que esto afectara al subsistema de riego si la humedad del suelo era alta.

Comportamiento observado

El sistema detectó el aumento de temperatura (>25ºC) y activó el ventilador, deteniendo el motor en cuanto la temperatura descendió. La bomba de agua no se activó, ya que la humedad del suelo no era elevada. Los datos fueron correctamente mostrados en el LCD y enviados a la Raspberry.

Conclusiones

El sistema es capaz de actuar sobre diferentes elementos (riego y ventilación) de forma independiente y basada en múltiples entradas sensoriales. Se confirma que los subsistemas funcionan de forma autónoma sin interferencias entre ellos.

7.4. Validación de la transmisión de datos

Condición de partida

Se conectó el Arduino a la Raspberry Pi mediante USB Serial y se supervisó durante varios ciclos la transmisión de datos, incluyendo valores de temperatura, humedad del aire, humedad del suelo y segundos de riego. Se utilizó InfluxDB y Grafana para monitorizar el flujo de datos.

Comportamiento observado

Los datos llegaron a la Raspberry con el formato correcto cada 60 segundos, fueron insertados en InfluxDB sin errores y se visualizaron en tiempo real en Grafana. No se detectaron pérdidas de paquetes ni valores anómalos durante la prueba.

Conclusiones

El flujo de datos desde el microcontrolador hasta la base de datos funciona de forma estable y eficiente. Se valida así la comunicación Serial, la recolección periódica de datos y su almacenamiento.

7.5. Evaluación del modelo predictivo

Condición de partida

Se introdujo manualmente una combinación conocida de condiciones ambientales: 25ºC de temperatura, 60 % de humedad del aire y 30 % de humedad del suelo.

Comportamiento observado

El modelo predice un riego de 5,6 segundos. Esta duración coincide con el comportamiento habitual observado anteriormente en condiciones similares.

Conclusiones

El modelo es capaz de realizar una predicción coherente y razonable, lo que confirma que ha aprendido correctamente la relación entre las variables.

8

Conclusiones y líneas futuras

Este capítulo final recoge las reflexiones generales sobre el desarrollo del proyecto, así como las principales conclusiones extraídas durante su implementación. Además, se proponen algunas posibles líneas de mejora y extensión del sistema desarrollado, que podrían explorarse en futuros trabajos relacionados.

8.1. Conclusiones

A lo largo de este Trabajo de Fin de Grado se ha desarrollado un sistema inteligente de riego basado en tecnologías IoT y gemelos digitales, cuyo objetivo principal ha sido optimizar el consumo de agua en un entorno agrícola controlado. El sistema ha sido capaz de recopilar datos ambientales en tiempo real, almacenarlos y analizarlos, y tomar decisiones automáticas sobre la activación y duración del riego. Además, se ha implementado un modelo predictivo que, a partir de los datos históricos, estima la necesidad de riego en función de las condiciones actuales, constituyendo así un gemelo digital funcional del invernadero.

Durante la realización del proyecto se exploraron dos soluciones: una primera opción basada en la placa Wemos D1 R1 y comunicación WiFi con la Raspberry Pi, y una segunda opción más robusta con una placa Arduino UNO y comunicación por USB Serial. La segunda opción resultó ser más estable, debido a las limitaciones de recursos y conectividad de la primera.

Una de las lecciones más significativas extraídas del proyecto ha sido el impacto de la calidad de los componentes hardware en la estabilidad del sistema. Al tratarse de una propuesta de bajo coste, muchos de los dispositivos utilizados presentaron fallos ocasionales, contactos inestables o imprecisión en las mediciones. Estos problemas, aunque solucionables, han dificultado el desarrollo fluido del sistema y han evidenciado la importancia de invertir en hardware de

mayor calidad cuando se buscan soluciones fiables y duraderas. No obstante, haber trabajado con componentes económicos ha permitido diseñar una arquitectura accesible y replicable, lo cual es también un valor añadido del proyecto.

En resumen, el sistema desarrollado cumple con los objetivos planteados: permite una gestión inteligente del riego, reduce el consumo de agua y sienta las bases para el uso de técnicas predictivas en agricultura de precisión.

8.2. Líneas futuras

Existen múltiples vías para continuar y mejorar el sistema implementado. A continuación se destacan algunas propuestas de líneas futuras:

- **Mejora del hardware**: Sustituir sensores y componentes económicos por dispositivos de mayor precisión, robustez y fiabilidad permitiría mejorar la estabilidad del sistema, reducir errores y simplificar el mantenimiento.
- **Ampliación de variables monitorizadas**: Se podrían integrar sensores adicionales como luminosidad, presión atmosférica, o pH del suelo, para hacer el sistema aún más completo y adaptable a distintos tipos de cultivo.
- **Entrenamiento de modelos más complejos**: Aunque el modelo predictivo actual se basa en regresión lineal, se podrían explorar técnicas más avanzadas como redes neuronales o modelos de árboles de decisión, que podrían ofrecer predicciones más precisas en condiciones variables.
- **Interfaz de usuario mejorada**: Actualmente el sistema se consulta a través de Grafana, pero podría complementarse con una aplicación web o móvil personalizada que permita una gestión remota más cómoda y adaptada al usuario final.
- **Escalabilidad del sistema**: Adaptar el sistema a grandes cultivos distribuidos, incluyendo una arquitectura distribuida y más nodos de sensorización, permitiría escalar esta solución a entornos agrícolas de mayor envergadura.
- **Integración con predicciones climáticas externas**: Incorporar servicios de predicción meteorológica podría aumentar la precisión de las decisiones de riego, lo que re-

forzaría el componente preventivo del gemelo digital.

En definitiva, este proyecto abre la puerta a futuras investigaciones y desarrollos en el ámbito de la agricultura inteligente, demostrando cómo la combinación de IoT, análisis de datos y modelos predictivos puede contribuir de forma significativa a la sostenibilidad y eficiencia del sector agrícola.

Referencias

[1] "Manejo proactivo de la sequía". En: *FAO* (2021). URL: https://www.fao.org/climate-smart-agriculture/knowledge/practices/drought/es.

[2] R. Gebbers V. I. Adamchuk. "Science 327". En: 2010. Cap. Precision agriculture and food security.

[3] "Digital twin in industry: State-of-the-art". En: *CIRP Annals - Manufacturing Technology* (2019).

[4] "Internet of Things (IoT): A vision, architectural elements, and future directions". En: *Future Generation Computer Systems* (2013). URL: https://www.sciencedirect.com/science/article/abs/pii/S0167739X13000241.

[5] "Optimización del agua de riego, una herramienta de precisión para la mejora de los cultivos". En: *NEIKER* (2022). URL: https://neiker.eus/es/noticias/optimizacion-del-agua-de-riego-una-herramienta-de-precision-para-la-mejora-de-los-cultivos/.

[6] "Metodología SCRUM: qué es y cómo implementarlo". En: *ILIMIT* (2025). URL: https://www.ilimit.com/es/blog/tecnologico-2/metodologia-scrum-2.

[7] "Qué es la Inteligencia Artificial". En: *PRTR, Gobierno de España* (2023). URL: https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr.

[8] *Artificial Intelligence: A Modern Approach*. University of California, Berkeley, 2021.

[9] S. Wolfert. *Big Data in Smart Farming. Agricultural Systems*. 2017.

[10] Ioannis N. Athanasiadis Christos Pylianidis Sjoukje Osinga. *Introducing digital twins to agriculture*. 2021.

[11] "OpenAg MIT". En: *MIT Media Lab* (2020). URL: https://www.media.mit.edu/groups/open-agriculture-openag.

[12] "PlantVillage Nuru". En: *Diario de Colima* (2019). URL: https://diariodecolima.com/noticias/detalle/2019-05-29-plantvillage-nuru-el-app-que-ayuda-a-agricultores-a-diagnosticar-enfermedades-de-sus-cultivos.

[13] *Arduino IDE*. URL: https://www.arduino.cc/en/software/.

[14] *Wemos*. URL: https://www.wemos.cc/.

[15] *Raspberry Pi Foundation*. URL: https://www.raspberrypi.org/.

[16] *InfluxDB*. URL: https://www.influxdata.com/products/influxdb/.

[17] *Grafana*. URL: https://grafana.com/docs/grafana/latest/datasources/influxdb/.

[18] *Mosquitto MQTT*. URL: https://mosquitto.org/.

[19] *Node-RED*. URL: https://nodered.org/.

[20] *Pandas*. URL: https://pandas.pydata.org/docs/.

[21] *Scikit-learn*. URL: https://scikit-learn.org/.

[22] *NumPy*. URL: https://numpy.org/.

[23] *Python*. URL: https://www.python.org/.

[24] *Docker*. URL: https://docs.docker.com/.

[25] *Pyserial*. URL: https://pyserial.readthedocs.io/en/latest/pyserial.html.

[26] *InfluxDB Python library*. URL: https://influxdb-python.readthedocs.io/en/latest/.

[27] *Repositorio de GitHub de este TFG*. URL: https://github.com/estheralmendros/TFG-Esther.

[28] *Getting started with your Raspberry Pi*. URL: https://www.raspberrypi.com/documentation/computers/getting-started.html.

[29] *Cron Jobs Made Easy: Your Guide to Automating Anything!* URL: https://dev.to/rijultp/cron-jobs-made-easy-your-guide-to-automating-anything-45ac.

Apéndice A
Manual de Instalación

Este manual describe los pasos necesarios para desplegar el sistema de riego inteligente con gemelo digital en un entorno similar al del proyecto desarrollado. Se asume que se dispone de un invernadero con sensores de temperatura, humedad del aire y humedad del suelo, así como actuadores como una bomba de agua controlada mediante un relé. Además, se requiere una Raspberry Pi como servidor central del sistema.

A.1. Requisitos previos

Antes de comenzar, se debe contar con el siguiente equipamiento y software:

- Un sistema físico de riego basado en Arduino (placa, sensores y actuadores).
- Raspberry Pi (3 o superior) con Raspbian OS instalado [28].
- Conexión estable a Internet.
- Docker y Docker Compose instalados en la Raspberry Pi.
- Cable USB para conectar Arduino con la Raspberry Pi.

A.2. Fases de instalación

1. **Preparación del sistema físico (gemelo físico)**

   Se debe montar el invernadero físico siguiendo el esquema eléctrico del proyecto:

   - Instalar y conectar los sensores de temperatura, humedad del aire y humedad del suelo.
   - Conectar la bomba de agua a través de un relé, asegurando una fuente de alimentación externa si es necesario.

- Cargar el archivo `invernadero_serial.ino` en la placa Arduino desde el entorno de desarrollo Arduino IDE.
- Realizar una comprobación básica: al iniciar el sistema, el sensor de humedad del suelo debe ser leído correctamente y la bomba debe activarse si la humedad cae por debajo del umbral definido.

2. **Configuración de la Raspberry Pi**: Para habilitar la recogida y análisis de datos, la Raspberry debe estar correctamente configurada:
   - Crear una carpeta de trabajo donde se organizarán los ficheros del proyecto.
   - Instalar los servicios necesarios mediante IoTStack. Para ello:
     ```
     curl -fsSL https://raw.githubusercontent.com/SensorsIot/IOTstack/master/install.sh
     | bash
     sudo shutdown -r now
     cd IOTstack/
     ./menu.sh
     ```
     En este menú, seleccionar los servicios se desean instalar. En este caso, InfluxDB y Grafana.
   - Iniciar los servicios con `docker-compose up -d`.

3. **Verificación de conexión Serial**: Una vez conectada la placa Arduino a la Raspberry mediante USB, se debe verificar que aparece en `/dev/serial/by-id/`. Este nombre se usará más adelante en el script de lectura, y se puede comprobar con el comando:
   `ls /dev/serial/by-id/`

4. **Configuración de la base de datos InfluxDB**: Es necesario crear una base de datos que contenga las métricas del sistema:
   - Acceder al contenedor de InfluxDB:
     `docker exec -it influxdb influx`
   - Crear la base de datos *invernadero*.
   - Se utilizarán las siguientes *measurements* (mediciones): *temperatura*, *humedad_aire*, *humedad_suelo* y *segs_riego*.

5. **Ejecución del script de lectura de datos**: Este script recibe los datos del Arduino y los almacena en la base de datos:

   - Configurar el script `arduino_a_influx.py` con el nombre del puerto serial correcto detectado en `/dev/serial/by-id/`.
   - Ejecutar el script: `python3 arduino_a_influx.py`.

   Para mantenerlo activo, se recomienda ejecutarlo como servicio o emplear un gestor de procesos como `cron` o `systemd` [29].

6. **Visualizar datos en Grafana**: El sistema permite visualizar los datos en tiempo real:

   - Acceder a Grafana desde cualquier navegador mediante: `http://<IP-Raspberry>:3000`.
   - Añadir InfluxDB como fuente de datos y crear un panel de visualización (*dashboard*) con las variables del sistema.

7. **Entrenamiento y ejecución del gemelo digital**: El sistema predictivo se activa desde un script Python:

   - Ejecutar `gemelo_digital.py` una vez exista un volumen adecuado de datos históricos.
   - El modelo entrenado se guardará como `model.pkl`.
   - Se recomienda automatizar este script mediante `cron` o `systemd` [29].

Apéndice B
Manual de Usuario

Una vez completada la instalación, el sistema de riego inteligente funciona de forma autónoma. El objetivo de este manual es explicar cómo interpretar el comportamiento del sistema y acceder a los datos generados por los sensores, así como a las predicciones del gemelo digital.

B.1. Funcionamiento general

El sistema está compuesto por dos componentes principales:

- El **gemelo físico**, que se encarga de leer los datos del entorno y activar la bomba de agua cuando la humedad del suelo está por debajo de un determinado umbral.
- La **Raspberry Pi**, que actúa como servidor y centro de análisis, almacenando los datos y ejecutando el modelo de predicción.

El Arduino envía los valores de los sensores cada minuto, incluyendo la temperatura, la humedad del aire, la humedad del suelo y los segundos que ha estado activa la bomba en ese intervalo.

B.2. Visualización en tiempo real

La herramienta Grafana permite al usuario consultar fácilmente la información generada por el sistema:

- Desde un navegador web, se puede acceder a `http://<IP-Raspberry>:3000`.
- Se mostrará un *dashboard* con gráficas actualizadas en tiempo real para cada una de las variables.
- El panel puede personalizarse para mostrar intervalos de tiempo concretos, comparar variables y detectar anomalías.

B.3. Interpretación de las predicciones

El gemelo digital calcula cuántos segundos de riego serían óptimos bajo determinadas condiciones ambientales. Este valor se guarda en una nueva métrica llamada *predicciones_riego*.

- Si la predicción coincide con el valor real, se considera que el modelo está correctamente ajustado.
- En caso contrario, puede ser necesario volver a entrenar el modelo (`gemelo_digital.py`) una vez se acumulen más datos.

B.4. Comprobaciones recomendadas

De forma periódica, se recomienda:

- Verificar en el *dashboard* que los valores de los sensores están dentro de rangos esperados.
- Comprobar que la bomba se activa cuando el nivel de humedad del suelo es bajo.
- Revisar que las predicciones se actualizan correctamente.
- Realizar copias de seguridad del modelo `model.pkl` y la base de datos *invernadero*.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA