



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial  
y Diseño Industrial

Desarrollo e implementación de un sistema de gestión de  
inventario de plantas en un vivero mediante el uso de  
tecnología RFID y una base de datos.

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Mesado González, Enrique

Tutor/a: Guasque Ortega, Ana

CURSO ACADÉMICO: 2022/2023

---

**Resumen:**

El proyecto consiste en el diseño y construcción de un sistema de gestión de inventario de plantas en un vivero utilizando tecnología RFID (Identificación por Radiofrecuencia). El objetivo principal es mejorar la eficiencia de la gestión del inventario y reducir el número de errores humanos en el proceso. El sistema se basa en etiquetas RFID que se aplicarán a diferentes plantas en un vivero, lectores RFID y un software de gestión de inventario. Los lectores RFID se instalarán en lugares estratégicos del vivero para capturar la información de las etiquetas de manera automática y sin contacto. El software de gestión de inventario permitirá visualizar y actualizar la información de cada planta en tiempo real, lo que facilitará el proceso de seguimiento y control del inventario. Este software será accesible desde cualquier dispositivo para poder gestionar el inventario en cualquier momento y lugar.

---

**Palabras clave:**

RFID; Etiquetas RFID; Sensorización; Bases de datos; Trazabilidad.



---

**Abstract:**

The project consists of the design and construction of a management system inventory of plants in a nursery using RFID technology (Identification by Radio Frequency). The main objective is to improve the efficiency of the inventory management and reduce the number of human errors in the process. The system is based on RFID tags that will be applied to different plants in a nursery, RFID readers and inventory management software. RFID readers They will be installed in strategic places in the nursery to capture information of labels automatically and without contact. The inventory management software will allow you to view and update the information on each plant in real time, which will facilitate the process of inventory monitoring and control. This software will be accessible from any device to be able to manage inventory anytime, anywhere.

---

**Keywords:**

RFID; RFID tags; Sensorization; Databases; Traceability.



---

**Resum:**

El projecte consisteix en el disseny i construcció d'un sistema de gestió de inventari de plantes en un viver utilitzant tecnologia RFID (Identificació per Radiofreqüència) . L'objectiu principal és millorar l'eficiència de la gestió de l'inventari i reduir el nombre d'errors humans en el procés. El sistema es basa en etiquetes RFID que s'aplicaran a diferents plantes en un viver, lectors RFID i un programa de gestió d'inventari. Els lectors RFID s'instal·laran en llocs estratègics del viver per a capturar la informació de les etiquetes de manera automàtica i sense contacte. El programari de gestió d'inventari permetrà visualitzar i actualitzar la informació de cada planta en temps real, la qual cosa facilitarà el procés de seguiment i control de l'inventari. Aquest programari serà accessible des de qualsevol dispositiu per a poder gestionar l'inventari en qualsevol moment i lloc.

---

**Paraules clau:**

RFID; RFID etiquetes; Sensorització; Bases de dades; Traçabilitat.



---

### **Agradecimientos:**

Agradezco a todas las personas que contribuyeron de manera significativa a la realización de este proyecto. Sus apoyos, consejos y esfuerzos han hecho posible este proyecto.

Ya en primer lugar, quiero agradecer a mis padres y familia por su inquebrantable apoyo a lo largo de este viaje. A mi hermano, quien ha mostrado su gran interés durante todo el desarrollo de este proyecto.

Otro reconocimiento especial va dirigido a mi padre, en particular, le debo un agradecimiento especial por su dedicación y apoyo en la implementación de este proyecto. Su conocimiento y experiencia han sido la base de todo esto.

Una mención destacada merecen mis amigos, quienes siempre han estado ahí para ofrecer su apoyo y ánimo en todo momento, se lo agradezco de todo corazón.

Bajo la tutela de mi tutora, Ana, he aprendido y avanzado enormemente. Su guía constante, sus consejos expertos y su disposición para seguir de cerca el progreso de este trabajo han sido fundamentales.

A todos los que contribuyeron, directa o indirectamente, a este proyecto, les doy las gracias. Esto no habría sido posible sin su generosidad y apoyo.

---





# Índice general

Índice de figuras	XIII
Índice de tablas	XV
Listado de acrónimos empleados	XVII
<b>I Memoria</b>	<b>1</b>
<b>1. Introducción y objetivos</b>	<b>3</b>
1.1. Introducción . . . . .	3
1.2. Objetivos . . . . .	3
1.3. Entorno de trabajo . . . . .	4
<b>2. Estado del arte</b>	<b>7</b>
2.1. Tecnología RFID . . . . .	7
2.1.1. Introducción a los sistemas RFID . . . . .	7
2.1.2. Antecedentes la tecnología RFID . . . . .	8
2.1.3. Funcionamiento de un sistema RFID . . . . .	9
2.1.4. Aplicaciones y casos de uso . . . . .	10
2.1.5. Elementos de un sistema RFID . . . . .	12
Etiquetas RFID . . . . .	12
Lector RFID . . . . .	13
Impresoras RFID . . . . .	14
Antenas RFID . . . . .	15
2.1.6. Frecuencias de funcionamiento en sistemas RFID . . . . .	17
2.1.7. Ventajas y desventajas del uso de RFID . . . . .	18
2.2. Sistemas de identificación . . . . .	19
<b>3. Planteamiento de soluciones alternativas y justificación de la solución adoptada</b>	<b>25</b>

3.1. Estudio y selección de los componentes . . . . .	25
3.1.1. Tecnología de identificación . . . . .	25
3.1.2. Microcontrolador . . . . .	26
3.1.3. Módulo lector RFID . . . . .	27
3.1.4. Método de intercambio de datos con el servidor . . . . .	28
<b>4. Descripción detallada de la solución adoptada</b>	<b>29</b>
4.1. Descripción del sistema . . . . .	29
4.1.1. Hardware . . . . .	29
ESP32-DevKitC V4 . . . . .	29
RFID-RC522 . . . . .	31
4.1.2. Software . . . . .	32
Arduino . . . . .	33
Scripts PHP . . . . .	33
Base de datos . . . . .	33
Visual Studio . . . . .	33
4.2. Implementación del hardware . . . . .	34
4.2.1. Diagrama de conexiones . . . . .	34
4.3. Implementación del software . . . . .	35
4.3.1. Arduino . . . . .	35
4.3.2. Servidor PHP . . . . .	40
4.3.3. Base de datos . . . . .	43
4.3.4. Visual Studio . . . . .	47
<b>5. Resultados</b>	<b>55</b>
5.1. Implementación . . . . .	55
5.2. Ampliación del trabajo . . . . .	58
5.3. Conclusiones y valoración personal . . . . .	64
<b>6. Estudio económico</b>	<b>65</b>
6.1. Revisión de costes . . . . .	65
6.1.1. Costes del proyecto realizado . . . . .	65
6.1.2. Mano de obra . . . . .	66
6.1.3. Costes totales . . . . .	66
6.1.4. Costes de la idea de proyecto futuro . . . . .	66

<b>II</b>	<b>Pliego de condiciones</b>	<b>69</b>
<b>7.</b>	<b>Pliego de condiciones</b>	<b>71</b>
7.1.	Objeto . . . . .	71
7.2.	Condiciones técnicas . . . . .	71
7.3.	Control de calidad . . . . .	71
7.4.	Condiciones de ejecución . . . . .	73
7.5.	Manual de usuario . . . . .	73
	<b>Bibliografía</b>	<b>75</b>
<b>III</b>	<b>Anexos</b>	<b>79</b>
<b>A.</b>	<b>Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2030.</b>	<b>81</b>
<b>B.</b>	<b>Programación</b>	<b>83</b>
B.1.	Arduino . . . . .	83
B.2.	Scripts PHP . . . . .	85
B.3.	Base de datos . . . . .	88
B.4.	Visual Studio . . . . .	89



# Índice de figuras

1.1. Esquema visual del sistema a implementar . . . . .	4
1.2. Logo de Viveros Mesado . . . . .	5
2.1. Esquema RFID . . . . .	7
2.2. Acoplamiento inductivo en RFID . . . . .	10
2.3. Acoplamiento capacitivo en RFID . . . . .	11
2.4. Acoplamiento por propagación de ondas en RFID . . . . .	11
2.5. Composición Etiqueta RFID . . . . .	12
2.6. Lector RFID fijo . . . . .	14
2.7. Lector RFID portátil . . . . .	14
2.8. Ejemplo impresora RFID . . . . .	15
2.9. Antenas en un sistema RFID . . . . .	15
2.10. Comparación ganancia y ancho de haz de una antena . . . . .	16
2.11. Comparación antena en función de las espirales . . . . .	17
2.12. RFID frecuencias y rango de distancia . . . . .	17
2.13. RFID tamaño antena en función de frecuencias . . . . .	18
2.14. Comparación gráfica RFID y código de barras . . . . .	22
4.1. ESP32-DevKitC V4 . . . . .	29
4.2. PINOUT ESP32 . . . . .	30
4.3. Lector RFID RC522 . . . . .	31
4.4. Pineado del lector RFID RC522 . . . . .	32
4.5. Diagrama de conexiones del sistema . . . . .	34
4.6. Diagrama de conexiones de software . . . . .	35
4.7. Diagrama de flujo Arduino . . . . .	40
4.8. Crear nuevo sitio web en 000webhost . . . . .	40
4.9. Crear nuevo sitio web en 000webhost 2 . . . . .	41
4.10. Scripts dentro de los archivos del servidor . . . . .	41
4.11. Crear nueva base de datos . . . . .	43
4.12. Información base de datos . . . . .	44

---

4.13. mySQL Tabla invernaderos . . . . .	45
4.14. mySQL Tabla plantas . . . . .	46
4.15. mySQL Tabla invernadero_planta . . . . .	46
4.16. Main Window . . . . .	48
4.17. Ventana Gestionar Inventario Visual Studio . . . . .	48
4.18. Mensaje de error puerto COM Visual Studio <b>Fuente:</b> Propia . . . . .	50
4.19. Ventana Ver Inventario Visual Studio . . . . .	52
4.20. Ventana Invernadero 1 Visual Studio . . . . .	52
5.1. Máquina trabajando con el modo agregar . . . . .	56
5.2. Lector RFID en la cinta transportadora . . . . .	57
5.3. Plantel y macetas preparadas para sembrar . . . . .	58
5.4. Proceso de plantación de esquejes . . . . .	59
5.5. Transporte de los carros hasta el invernadero . . . . .	60
5.6. Colocación de las plantas en el invernadero . . . . .	61
5.7. Visualización de las plantas en un invernadero . . . . .	62
5.8. Escáner RFID identificando múltiples plantas a la vez . . . . .	63

# Índice de tablas

2.1. Comparación de las Principales Tecnologías de Identificación . . . . .	21
2.2. Comparación de las tecnologías de Identificación Automática . . . . .	22
2.3. Comparación entre Tarjeta Magnética y RFID . . . . .	22
2.4. Comparación entre Códigos de Barras y Tecnología RFID . . . . .	23
3.1. Comparación entre ESP32 Dev Kit y ESP8266 . . . . .	26
6.1. Costes del proyecto . . . . .	65
6.2. Coste humano . . . . .	66
6.3. Coste Total . . . . .	66
6.4. Costes de la idea de proyecto . . . . .	67
A.1. Objetivos de Desarrollo Sostenibles . . . . .	82





# Listado de acrónimos empleados

**RFID** Radio Frequency Identification. [7](#)

**IOT** Internet Of The Things. [8](#)

**NFC** Near Field Communication. [27](#)

**PHP** Hypertext Preprocessor. [33](#)

**ID** Identification. [33](#)

**DBMS** DataBase Management System. [33](#)

**IDE** Entorno de desarrollo integrado. [33](#)

**UID** Identificador Único. [36](#)

**GUI** Interfaz gráfica de usuario. [47](#)

**ODS** Objetivos de Desarrollo Sostenible. [81](#)



Parte I

Memoria



# Capítulo 1

## Introducción y objetivos

### 1.1. Introducción

Los viveros juegan un papel crucial en la conservación y el desarrollo sostenible de los recursos naturales, así como en la promoción de la biodiversidad y la producción agrícola.

Algunas de las principales razones que destacan la importancia de los viveros son la conservación de especies, la reforestación y restauración ecológica, la producción de alimentos y la mejora genética de plantas.

Por tanto, los viveros son esenciales para la conservación de la biodiversidad, la mitigación del cambio climático, la investigación científica y la mejora de la calidad de vida de las comunidades humanas. Su importancia radica en su capacidad para proporcionar y mantener una amplia variedad de plantas, desde especies amenazadas hasta cultivos esenciales, y para contribuir al desarrollo sostenible y al bienestar humano.

Para la operación eficaz de los viveros y garantizar una gestión eficiente de sus recursos, es fundamental el control exacto y actualizado del inventario de plantas. Es en este sentido en el que se enmarca este trabajo.

### 1.2. Objetivos

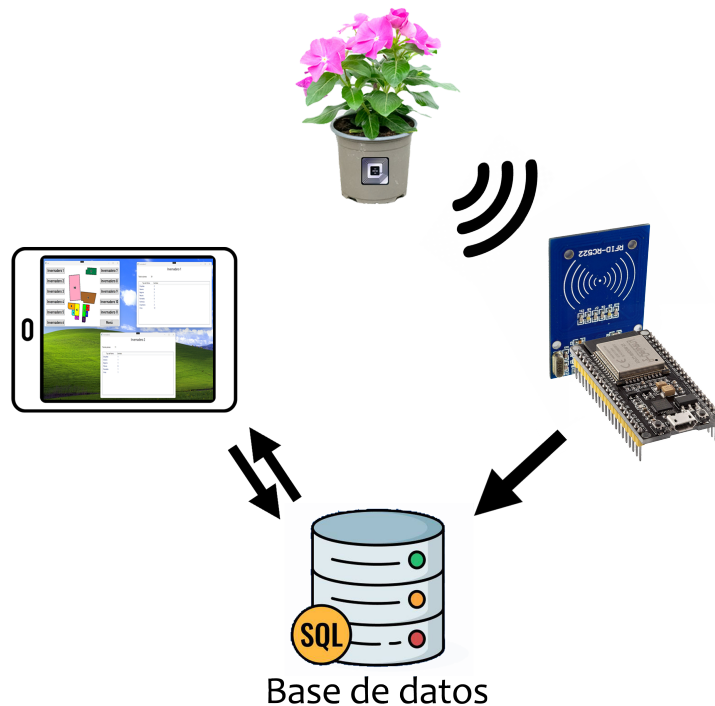
El objetivo de este proyecto es la creación y puesta en marcha de un sistema de gestión de inventario en tiempo real para un vivero, fundamentado en una base de datos central y accesible desde cualquier dispositivo con conexión a Internet. El proyecto se enfocará en el diseño e implementación de una aplicación que facilite la administración del inventario de plantas, permitiendo añadir, eliminar y modificar los registros de plantas, así como la visualización detallada del inventario de cada invernadero. Para conseguir esto, se integrarán componentes de hardware en las plantas, como tarjetas RFID, para etiquetarlas y rastrearlas individualmente.

Se establecerán protocolos de comunicación entre el hardware y la base de datos para asegurar una actualización en tiempo real de la información. Este sistema contribuirá a mejorar la productividad y rentabilidad del vivero, minimizando los errores y la pérdida de tiempo asociados con la gestión manual de inventarios, y facilitando la toma de decisiones operativas y estratégicas.

El sistema es capaz de leer etiquetas RFID que llevan las plantas para gestionarlas

dentro del inventario, estos datos son recibidos por un microprocesador y se suben a una base de datos alojada en un servidor en la nube. Para ver estos datos se ha realizado una aplicación de escritorio la cual se puede acceder desde cualquier dispositivo para ver y gestionar el inventario en tiempo real (Figura 1.1).

Se realiza un proyecto ajustado a consideraciones económicas y de recursos disponibles en el momento de su desarrollo. Aunque el sistema es altamente efectivo en su implementación actual, es importante destacar que existen oportunidades de mejora y expansión en el futuro.



**Figura 1.1:** Esquema visual del sistema a implementar  
**Fuente:** Propia

### 1.3. Entorno de trabajo

Para la puesta en marcha de los resultados obtenidos con este proyecto se han realizado las pruebas en unos viveros (Figura 1.2).

Viveros Mesado es una sociedad agraria de transformación con más de 50 años de experiencia. Su actividad es el cultivo y comercialización de planta ornamental de temporada. Las instalaciones están ubicadas en Torrent (Valencia) y cuentan con una superficie de 66000 m<sup>2</sup> dedicados a la producción de plantas de calidad, que cumplen con la normativa vigente de seguridad vegetal. Se cultivan unas 100 especies de plantas diferentes, cuyas variedades y formatos de presentación están seleccionados para satisfacer una amplia variedad de requerimientos climáticos y ornamentales.



**Figura 1.2:** Logo de Viveros Mesado  
**Fuente:** Propia





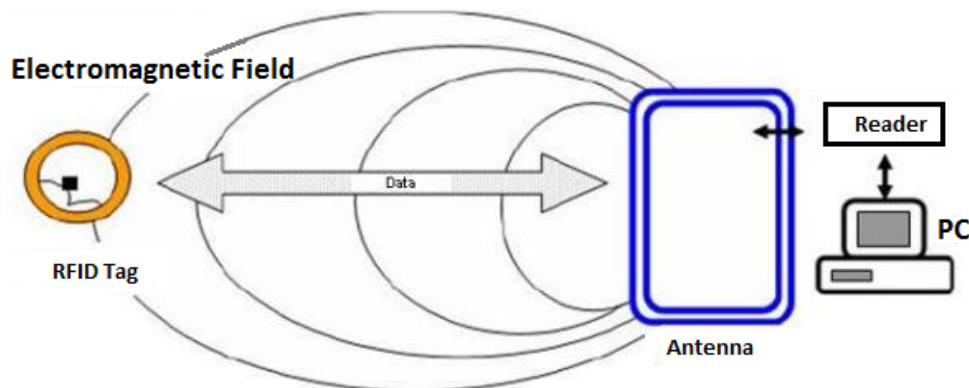
# Capítulo 2

## Estado del arte

### 2.1. Tecnología RFID

#### 2.1.1. Introducción a los sistemas RFID

Las siglas **RFID** provienen del inglés Radio Frequency Identification o identificación por radiofrecuencia. La base fundamental de esta tecnología se basa en la capacidad de rastrear un activo a través de diversos sistemas de software y hardware de manera remota, eliminando la necesidad de un contacto visual directo o una proximidad extrema con los dispositivos de recopilación de datos [1]. El empleo de este está dirigido a la identificación de objetos o personas o la localización de los mismos en un espacio concreto.



**Figura 2.1:** Esquema RFID

Fuente: [2]

En términos generales, un sistema RFID (Figura 2.1) necesita como mínimo estos elementos:

- **Etiquetas RFID:** Son dispositivos pequeños que contienen un chip y una antena. Estas etiquetas almacenan información y pueden ser adheridas o integradas en objetos o productos que se desean rastrear. Hay diferentes tipos de etiquetas RFID, como etiquetas pasivas, semiactivas y activas, que varían en función de su capacidad de energía y alcance de lectura.
- **Lectores RFID:** Son dispositivos que se utilizan para leer y escribir en las etiquetas RFID. Los lectores pueden ser fijos, portátiles o integrados en otros

sistemas. Se encargan de enviar señales de radiofrecuencia y recibir las respuestas de las etiquetas. Además, los lectores pueden estar conectados a una red para transmitir los datos recolectados.

- **Antenas RFID:** Son componentes esenciales para la comunicación entre los lectores y las etiquetas. Las antenas emiten y reciben señales de radiofrecuencia y determinan el alcance de lectura. La selección adecuada de las antenas depende del entorno y los requisitos específicos del sistema.
- **Software:** Generalmente se trata de un "middleware", encargado de administrar y procesar los datos recolectados por el sistema RFID. Puede realizar tareas como el seguimiento de activos, inventario, control de accesos, entre otros.

### 2.1.2. Antecedentes la tecnología RFID

La tecnología RFID ha estado presente en la historia de la humanidad desde sus inicios, ya que se basa en el uso de la energía electromagnética, que se considera uno de los elementos fundamentales del universo. [3]

El desarrollo de la comprensión científica de la energía electromagnética avanzó significativamente en los siglos XVII y XVIII, con figuras destacadas como *Benjamin Franklin* contribuyendo a la investigación eléctrica. En el siglo XIX, científicos como *Michael Faraday* y *James Clerk Maxwell* formularon teorías fundamentales sobre campos electromagnéticos y ondas, allanando el camino para el uso práctico de la energía electromagnética.

El verdadero avance en la tecnología RFID tuvo lugar en el siglo XX. En 1948, *Harry Stockman* escribió un artículo pionero sobre la comunicación mediante el poder reflejado, que es esencialmente el principio detrás de la tecnología RFID. Sin embargo, pasarían más de treinta años antes de que sus ideas se hicieran realidad.

La década de 1970 fue testigo del despegue de la tecnología RFID, con investigaciones activas, invenciones y desarrollos de compañías y laboratorios académicos. En esta época, se exploraron aplicaciones para el seguimiento de animales, vehículos y la automatización industrial.

En los años 80, se vieron los primeros despliegues significativos de la tecnología RFID, especialmente en el ámbito de la recolección electrónica de peaje en carreteras. Grandes empresas y laboratorios, como *Los Alamos Scientific Laboratory*, participaron en investigaciones y desarrollos clave.

En la década de 1990, la tecnología RFID comenzó a implementarse a gran escala, con sistemas de recolección de peajes electrónicos y aplicaciones de acceso en parques de estacionamiento y comunidades cerradas. Se realizaron avances tecnológicos, como la fabricación de diodos Schottky de microondas en circuitos integrados CMOS, lo que permitió etiquetas RFID más pequeñas y funcionales.

A medida que avanza el siglo XXI, la tecnología RFID sigue creciendo, con aplicaciones en la gestión de artículos y la convergencia con códigos de barras. Las perspectivas de futuro para RFID son prometedoras, con aplicaciones en telemática y comercio móvil, así como un mayor crecimiento en el IOT (Internet de las cosas).

### 2.1.3. Funcionamiento de un sistema RFID

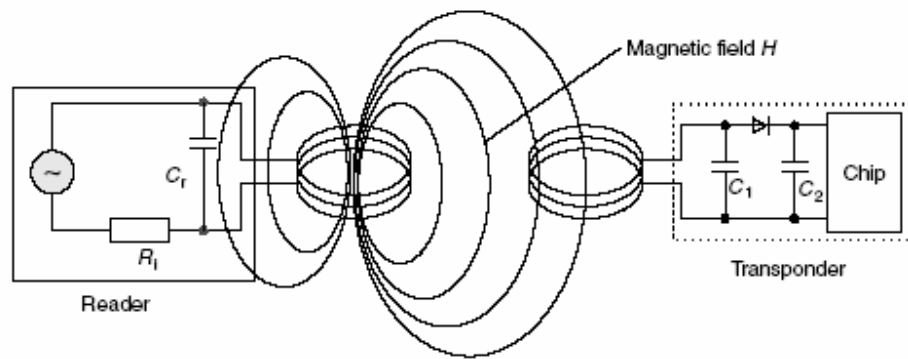
Un sistema de comunicación RFID se basa en la comunicación bidireccional entre un lector (interrogador) y una etiqueta (transponder), por medio de ondas de radiofrecuencia. Las etiquetas se encargan de almacenar los datos del activo mientras que los lectores son los encargados de recepcionar estos datos.

Para establecer una comunicación efectiva entre el lector y las etiquetas RFID, se siguen los siguientes tres pasos:

1. El lector emite constantemente señales de radiofrecuencia, en espera de que sean detectadas por alguna etiqueta. Las etiquetas activas tienen la capacidad de enviar información de forma autónoma, mientras que las etiquetas pasivas requieren ser activadas mediante una señal externa de radiofrecuencia.
2. La antena de la etiqueta RFID recibe la señal del lector y envía de vuelta al lector la información almacenada en el chip de la etiqueta. Es la antena la encargada tanto de recibir como de transmitir datos.
3. El lector recibe la señal de vuelta de la etiqueta y envía los datos al sistema informático encargado de gestionar esta información. En el sistema informático, se realizan acciones como el almacenamiento de datos y la actualización de la ubicación y otros detalles relacionados con el activo identificado por la etiqueta.

La potencia recibida por el lector que llega desde las etiquetas suele ser muy débil, mucho menor que la emitida por el lector. Por ello es importante asegurar una correcta comunicación entre etiqueta y el lector. El sistema de transmisión de información varía según la frecuencia en la que trabaja. Para las frecuencias más bajas se utiliza el acoplamiento inductivo y el acoplamiento capacitivo, mientras que para las frecuencias más altas es el sistema de propagación de ondas.<sup>[4]</sup>

- **Acoplamiento inductivo:** Se basa en la inducción electromagnética. Cuando una corriente eléctrica pasa a través de una bobina (antena del lector, en este caso), crea un campo magnético alrededor de la bobina. Si una segunda bobina (antena de la etiqueta RFID) está cerca, este campo magnético variable puede inducir una corriente eléctrica en la segunda bobina. Muy común en sistemas RFID de baja frecuencia (LF) y frecuencia intermedia (HF), como tarjetas de acceso, tarjetas de pago sin contacto y etiquetas para seguimiento de objetos en entornos controlados. Como ventajas de este método destacar que dado que se basa en campos magnéticos, el acoplamiento inductivo es menos susceptible a interferencias de materiales conductores y líquidos. Y como limitaciones hay que decir que debido a la naturaleza de los campos magnéticos, el rango es limitado y las antenas suelen ser bobinas de alambre, lo que puede influir en el diseño y tamaño de las etiquetas y lectores. (Figura 2.2)
- **Acoplamiento capacitivo:** Este método de acoplamiento utiliza campos eléctricos para transferir datos y energía entre el lector y la etiqueta RFID. Es similar a cómo funciona un condensador, en el que se almacena energía en un campo eléctrico entre dos placas conductoras, cuando el lector genera un campo eléctrico fluctuante, induce una tensión en la antena de la etiqueta, permitiendo que la etiqueta se alimente y comunique. El acoplamiento capacitivo



**Figura 2.2:** Acoplamiento inductivo en RFID

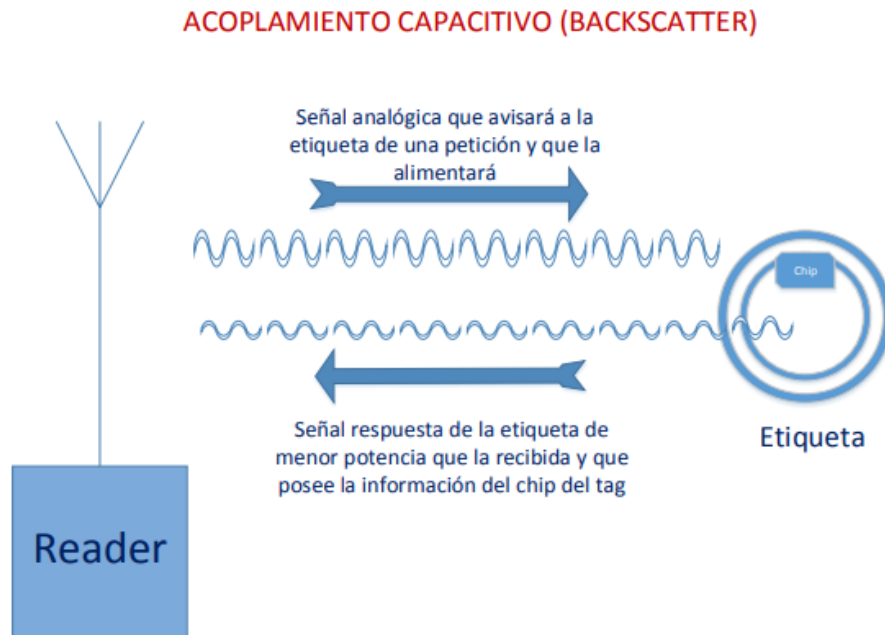
Fuente: [4]

es típicamente usado en sistemas RFID de corto alcance, donde las distancias entre el lector y la etiqueta son mínimas. Debido a la naturaleza de los campos eléctricos, su eficacia disminuye rápidamente con la distancia, haciendo que este método sea menos adecuado para aplicaciones de largo alcance. Una ventaja del acoplamiento capacitivo es que puede funcionar eficientemente en entornos con proximidad física cercana entre el lector y la etiqueta. Sin embargo, su alcance es limitado, y puede ser afectado por materiales dieléctricos cercanos o por la presencia de líquidos (Figura 2.3).

- Acoplamiento por propagación de ondas electromagnéticas:** Este método se basa en la transmisión y recepción de ondas electromagnéticas a través del espacio libre, permitiendo la comunicación entre el lector y la etiqueta RFID a distancias más largas. Cuando el lector emite una señal, genera ondas electromagnéticas que se propagan en todas direcciones. La etiqueta RFID, al encontrarse dentro del alcance de estas ondas, puede captarlas y extraer tanto la energía como la información contenida en ellas. Luego, la etiqueta puede usar esta energía para alimentarse y enviar una respuesta al lector. Este tipo de acoplamiento es típico en sistemas RFID UHF (Ultra High Frequency) y microondas, que están diseñados para operar a mayores distancias, incluso más allá de un metro. Estos sistemas son comunes en aplicaciones como la gestión de inventario. La ventaja de poder actuar a mayores distancias hace también que sea una desventaja, ya que las ondas electromagnéticas pueden ser afectadas por obstáculos o interferencias, y esto podría limitar la eficacia del sistema en ciertos entornos (Figura 2.4).

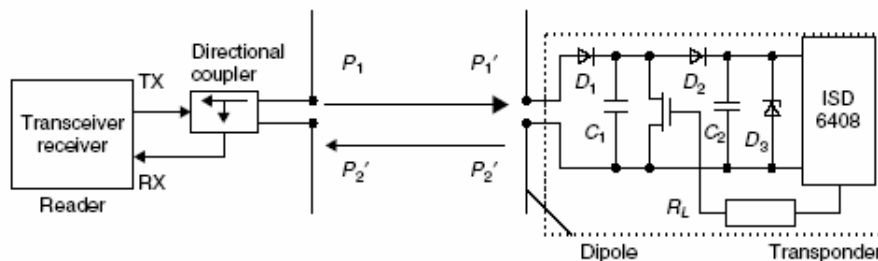
#### 2.1.4. Aplicaciones y casos de uso

La tecnología RFID ha ganado notable popularidad en todo el mundo, su uso se ha visto aumentado de forma exponencial durante los últimos años. Hoy en día, se aprecia su presencia en múltiples ámbitos, desde la cadena de suministro y la industria alimentaria hasta el cuidado de la salud, la agricultura y la gestión de accesos, ya que aporta una recogida y almacenamiento de datos de forma rápida y segura. Esta tecnología brinda numerosos beneficios, como aumentar la productividad, minimizar gastos, contrarrestar fraudes y fortalecer la seguridad de los artículos etiquetados con este método. En secciones previas, se abordaron algunas aplicaciones de la tecnología RFID; a continuación, se presenta un



**Figura 2.3:** Acoplamiento capacitivo en RFID

Fuente: [5]



**Figura 2.4:** Acoplamiento por propagación de ondas en RFID

Fuente: [6]

resumen de ellas [7]:

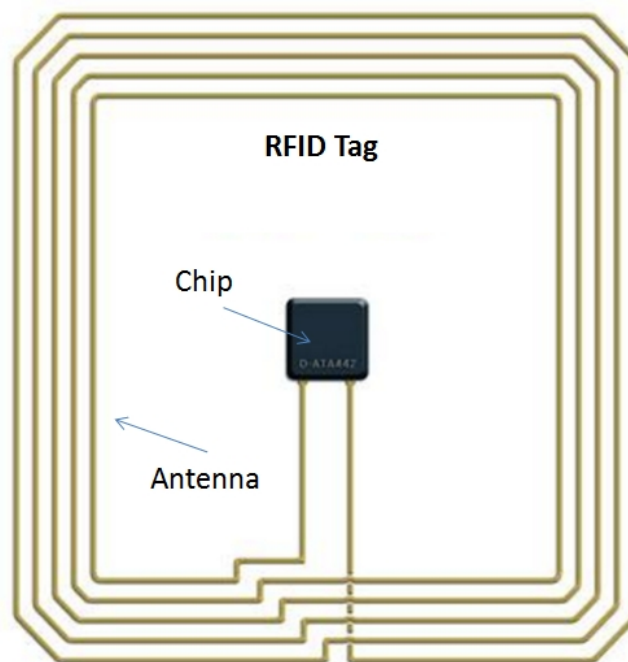
- **Control de accesos:** El RFID juega un papel crucial en la supervisión y control del acceso a áreas específicas [8]. A través de identificadores, como tarjetas, se puede verificar la autenticidad de una persona antes de concederle el acceso. Por ejemplo, el acceso podría limitarse a quienes hayan adquirido una entrada o a empleados autorizados.
- **Trazabilidad en la cadena de suministros:** La capacidad del RFID para rastrear automáticamente mercancías hace que el flujo y gestión de productos en el complejo entramado global sea más sencillo y fiable. Es esencial para coordinar entre diferentes entidades, como proveedores, distribuidores y transportistas, garantizando al mismo tiempo la seguridad de los productos y su entrega puntual [9].
- **Trazabilidad de productos comerciales:** Las etiquetas facilitan el rastreo y la gestión de inventarios de productos en establecimientos. Ayudan a prevenir pérdidas, ya sea por robo o extravío, y a mantener una visión actualizada del inventario. Esto no solo reduce el desabastecimiento, sino que también optimiza el proceso de facturación, evolucionando del tradicional código de barras.

- **Rastreo de Animales:** Tanto en la ganadería como en el cuidado de mascotas, los chips RFID se utilizan para almacenar y actualizar información sobre cada animal, incluyendo su salud y su historial [10].
- **Supervisión de Medicamentos y Cosméticos:** Dada la naturaleza delicada de los medicamentos y cosméticos, el RFID asegura un rastreo detallado y único de cada producto, proporcionando datos como fechas de caducidad, origen y más. Esto ayuda a evitar falsificaciones y a garantizar la seguridad del consumidor.
- **Pagos sin Contacto:** Es probablemente una de las aplicaciones más populares del RFID, permiten establecer la seguridad necesaria para realizar pago con ellas. El pago se realiza mediante teléfono móvil o con tarjetas especializadas para ello. En cuanto a las empresas, permite la recogida de tendencia y gustos por parte de los clientes, por lo que supone una ventaja para vendedores y para compradores, haciendo más cómodo, rápido y seguro el pago para estos últimos.
- **Cronometraje en Eventos Deportivos:** Las etiquetas RFID en atletas o participantes de competiciones hacen más eficiente el proceso de medición de tiempos y seguimiento de recorridos, optimizando costes y garantizando resultados precisos.

### 2.1.5. Elementos de un sistema RFID

#### Etiquetas RFID

Una etiqueta o tag está formada sencillamente por un transductor electromagnético, una antena impresa y un chip. Se utilizan para transmitir información a través de ondas de radio a un lector RFID, sin necesidad de contacto directo (Figura 2.5).



**Figura 2.5:** Composición Etiqueta RFID

**Fuente:** [11]

La antena se ocupa de la transmisión y recepción de los datos que serán enviados desde o hacia el chip. Éste último posee una memoria de unos cuantos Kb, según el tipo de etiqueta. Se pueden encontrar varios tipos de tags:

- **Etiquetas Pasivas:** No tienen fuente de energía propia. Se alimentan de la energía proveniente del lector RFID. Son las más comunes debido a su bajo costo y tamaño reducido. Tienen un alcance limitado, generalmente de pocos centímetros a varios metros.
- **Etiquetas Activas:** Tienen una batería incorporada que alimenta el chip y la antena, lo que les permite transmitir señales más fuertes y a mayores distancias (hasta 100 metros o más). Son más caras y grandes que las pasivas.
- **Etiquetas Semi-Activas:** Tienen baterías, pero solo se activan cuando están en el rango de un lector.

## Lector RFID

El lector al igual que los tag, se trata de uno de los ejes centrales de un sistema RFID, puesto que un tag sin un dispositivo que recepcione su información no tendría ninguna utilidad. La función principal de un lector RFID se basa en emitir señales de radiofrecuencia para identificar las etiquetas que se encuentren dentro de su alcance. Son los encargados de alimentar las etiquetas a través de las antenas, al mismo tiempo que capturan sus datos, los decodifican y los transmiten al software correspondiente para su interpretación.

En su fabricación podemos separar los lectores dos tipos. Primero, sistemas de bobina simple, en el cual la misma bobina sirve para transmitir la energía y los datos. Son más simples y más baratos, pero tienen menos alcance. Y por otro lado los que tienen dos bobinas, una para transmitir energía y otra para transmitir datos. Son más caros, pero consiguen mayores prestaciones.

Podemos dividir los lectores RFID en 2 tipos:

- **Lectores fijos:** Son dispositivos diseñados para su instalación en ubicaciones permanentes, como almacenes, puntos de venta o áreas de producción. Estos lectores suelen estar conectados a una fuente de alimentación constante y a una red o sistema centralizado. Debido a su configuración y antenas de mayor potencia, los lectores fijos pueden tener un alcance de lectura más amplio. Además, al estar instalados en una ubicación fija, este tipo de lectores pueden realizar lecturas continuas y en tiempo real de las etiquetas que se encuentren en su rango de alcance (Figura 2.6).
- **Lectores portátiles:** Son dispositivos compactos y móviles que se pueden transportar fácilmente. Estos lectores ofrecen flexibilidad y portabilidad para llevar a cabo tareas de lectura de etiquetas en diferentes ubicaciones. Suelen contar con baterías recargables, lo que les proporciona energía independiente y no requieren una conexión constante a una fuente de alimentación externa. Debido a su tamaño y configuración más compacta, los lectores portátiles pueden tener un alcance de lectura más limitado en comparación con los lectores



fijos. Los lectores portátiles suelen tener interfaces de usuario más amplias, como pantallas táctiles y botones, lo que permite una mayor interacción manual con el dispositivo (Figura 2.7).



**Figura 2.6:** Lector RFID fijo  
**Fuente:** [12]



**Figura 2.7:** Lector RFID portátil  
**Fuente:** [13]

El funcionamiento de un lector es el siguiente: Primero el lector genera una señal de radiofrecuencia que se utiliza para activar las etiquetas cercanas, esta señal se emite a través de una antena incorporada en el lector. Cuando una etiqueta RFID está dentro del rango de alcance del lector y recibe la señal de radiofrecuencia, se activa y responde al lector. El lector RFID procesa los datos capturados y los transmite a un sistema o aplicación.

## Impresoras RFID

Las impresoras RFID (Figura 2.8) son dispositivos especiales que permiten la impresión de etiquetas RFID de manera rápida y eficiente. A diferencia de las impresoras convencionales, las impresoras RFID están equipadas con tecnología especializada para imprimir información en etiquetas RFID. Existen dos formas de impresión en este tipo de impresoras:

- **Impresión directa:** El material de impresión, que suele ser una tinta térmica sensible al calor, se aplica directamente sobre la superficie de la etiqueta. La impresora aplica calor selectivamente a través de su cabezal térmico en las áreas deseadas de la etiqueta, lo que hace que la tinta reaccione y se vuelva visible. Es una impresión que destaca por ser rápida y sencilla, ideal para etiquetas que tienen una vida útil relativamente corta. Como desventaja cabe destacar que tiene una resistencia y durabilidad inferior comparada con la impresión de transferencia térmica.
- **Transferencia térmica:** Se utiliza una cinta de transferencia de tinta entre el cabezal de impresión y la etiqueta. El cabezal térmico aplica calor selectivamente en la cinta, lo que hace que la tinta se transfiera de la cinta a la etiqueta. La tinta se adhiere a la etiqueta mediante presión y calor, creando

una impresión duradera y de alta calidad. Permite la impresión en una variedad de materiales de etiquetas, incluyendo papel, poliéster y polietileno. Las limitaciones de este proceso son la lentitud y el elevado coste respecto a la transferencia directa.

Las impresoras RFID suelen ser compatibles con diferentes tipos de etiquetas RFID, como etiquetas pasivas, semiactivas o activas. Esto permite una mayor flexibilidad y adaptabilidad en función de los requisitos específicos del sistema RFID utilizado.



**Figura 2.8:** Ejemplo impresora RFID  
**Fuente:** [14]

## Antenas RFID

Las antenas RFID juegan un papel fundamental en la tecnología RFID. Son responsables de transmitir la señal de radiofrecuencia entre el lector y la etiqueta. Hay que tener en cuenta que en un sistema RFID se utilizan mínimo 2 antenas, la situada en la etiqueta y la situada en el lector (Figura 2.9). Después es posible añadir una antena externa más potente que sirva de enlace entre los 2 elementos principales.

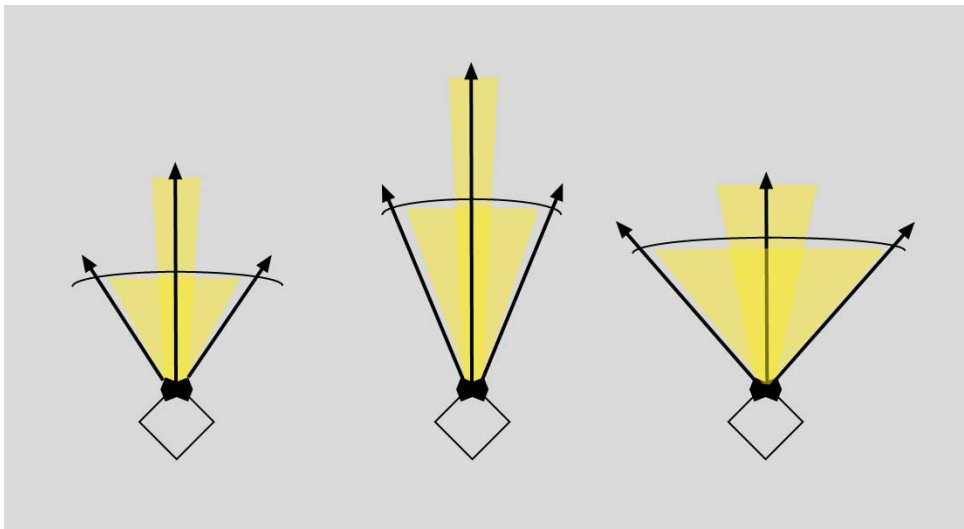


**Figura 2.9:** Antenas en un sistema RFID  
**Fuente:** Propia

La antena en una etiqueta RFID tiene la tarea de captar las ondas de radiofrecuencia (RF) y, a continuación, transmitir la información que está almacenada en el microchip de la etiqueta utilizando esas mismas ondas. La etiqueta obtiene la energía que necesita para funcionar de las ondas RF que están presentes en su entorno. Esta interacción y transferencia de energía entre las ondas y la antena es lo que se llama "acoplamiento". Por otro lado, la antena del dispositivo lector emite señales y también detecta las señales de retorno de las etiquetas RFID. El término "acoplamiento" se refiere a cómo se transfiere la energía entre dos sistemas, en este contexto, cómo las ondas de aire se conectan y transfieren energía a la antena.

Una antena RFID genera una zona de influencia alrededor de sí misma, que se puede visualizar como una forma tridimensional, a menudo referida como "haz" o "patrón". La principal ventaja de estas antenas es su habilidad para expandir este área de influencia lo más que puedan y también intensificar la potencia del campo electromagnético en esa área. En esencia, cuanto mayor y más potente sea esta zona, más eficientemente podrá detectar y leer las etiquetas RFID. Es importante destacar que el diseño y orientación de la antena, así como las características del entorno, pueden influir en este alcance y eficiencia.

La ganancia y la anchura del haz, dos especificaciones relativas a la parte eléctrica de la antena y estrechamente relacionadas. Una ganancia mayor crea un área de cobertura más estrecha, pero la distancia de lectura será mayor, es decir, la antena será más direccional. En contraposición, una menor ganancia crea un área de cobertura mayor, pero una distancia de lectura menor, es decir, la antena será más omnidireccional (Figura 2.10). La anchura del haz y la ganancia son inversamente proporcionales.



**Figura 2.10:** Comparación ganancia y ancho de haz de una antena  
**Fuente:** [15]

Por lo general, a mayor superficie de la antena del tag, tiene la capacidad de recopilar más energía y dirigirla al chip, lo que amplía su rango de lectura. Es esencial que la antena del tag sea pequeña, económica y de sencilla fabricación para su producción a gran escala. En muchos contextos, es preferible que la antena del tag posea una radiación omnidireccional o una cobertura hemisférica. Habitualmente, la resistencia del chip del tag no alcanza los 50 ohmios; por lo tanto, es vital que la antena esté adecuadamente alineada con el chip para proporcionarle la energía máxima. Esta antena del tag puede

presentarse en una sola espiral o en múltiples, tal como se muestra en (Figura 2.11).

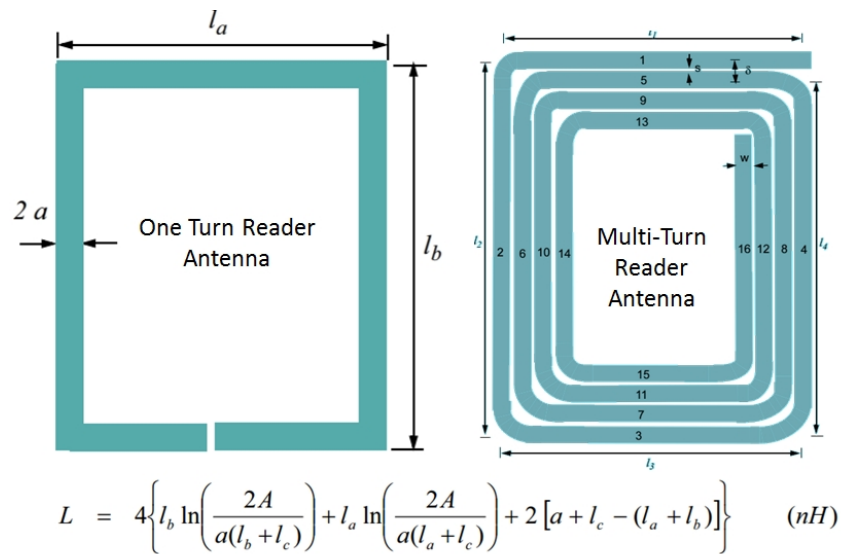


Figura 2.11: Comparación antena en función de las espirales  
Fuente: [11]

### 2.1.6. Frecuencias de funcionamiento en sistemas RFID

Para que un sistema RFID pueda comunicarse entre si es necesario que todos los dispositivos RFID de un sistema estén trabajando a la misma frecuencia, en función de la frecuencia a la que trabajen se pueden distinguir los siguientes modos (Figura 2.12) [7]:

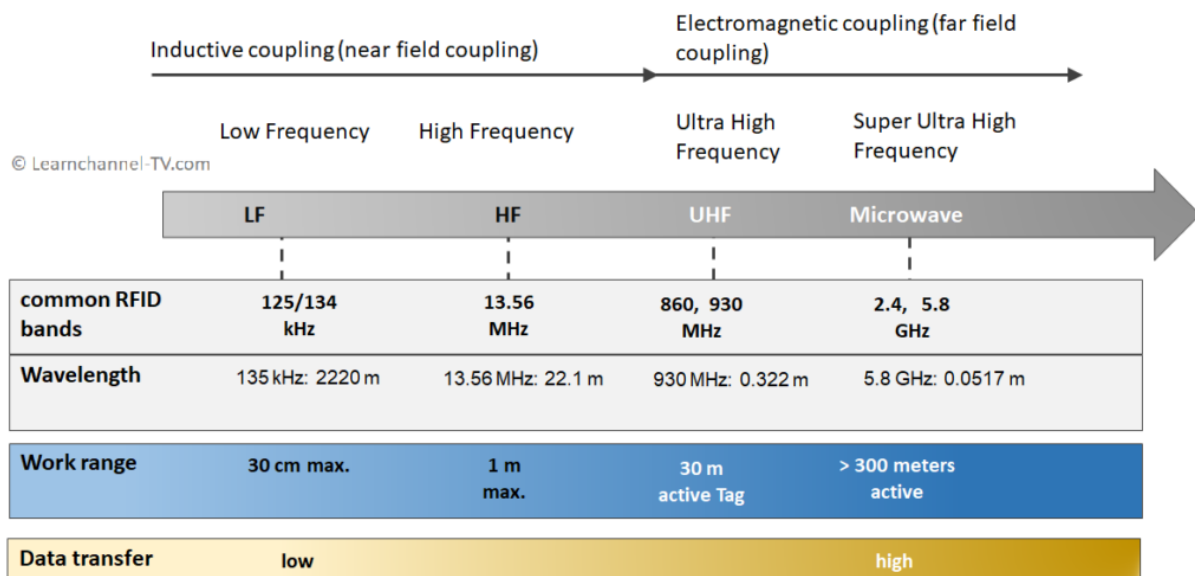


Figura 2.12: RFID frecuencias y rango de distancia  
Fuente: [16]

- **Baja Frecuencia (LF, "Low Frequency"):** Trabaja alrededor de los 125 a 134.2 kHz. Estas etiquetas tienen un alcance corto, usan menos energía y tienen un rango de aproximadamente medio metro. Son muy usadas en aplicaciones industriales ya que penetran

muy bien en metales y líquidos. Son ideales para aplicaciones donde el tag está cerca del lector, como el acceso controlado o el control de inventarios.

- **Frecuencia Alta (HF, "High Frequency"):** Opera alrededor de los 13.56 MHz, estas etiquetas tienen un alcance de lectura un poco más amplio que las LF, llegando hasta un poco menos de un metro aproximadamente. Son comunes en aplicaciones como seguimiento de activos.
- **Frecuencia Ultra Alta (UHF, "Ultra High Frequency"):** Funcionan entre los 860 y 960 MHz, ofrecen mayor alcance, pudiendo ser leídas a distancias de hasta 12 metros en condiciones óptimas. Pueden identificar gran cantidad de etiquetas a la vez y son más sensibles a interferencias. Los metales y líquidos afectan negativamente cuando se trabaja a estas frecuencias, los metales reflejan las señales y las moléculas de agua las absorben.
- **Microwave:** Esta frecuencia trabaja en 2.45 GHz aproximadamente y tiene una gran velocidad de lectura, más rápida incluso que las etiquetas UHF. Esta frecuencia produce mejores resultados en aplicaciones como el seguimiento de vehículos (dentro y fuera de barreras), con aproximadamente 1 metro de rango.

Estas frecuencias son fijas y los componentes no son intercambiables. Esto también se debe a que cada rango de frecuencia requiere su propia antena (Figura 2.13). En general se puede decir: cuanto mayor es la frecuencia, más pequeña es la antena:

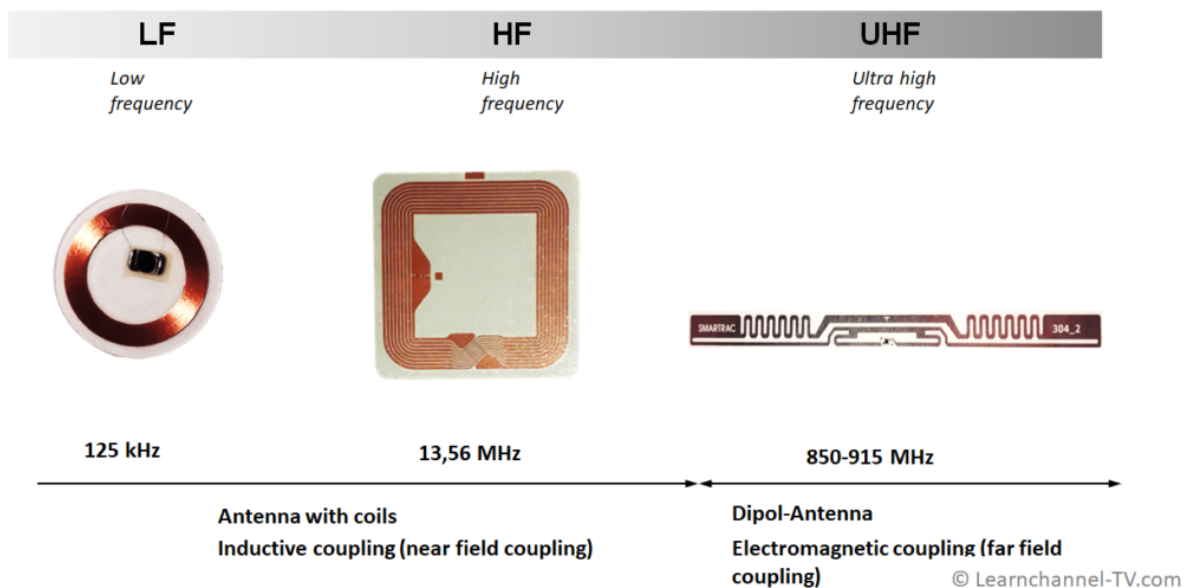


Figura 2.13: RFID tamaño antena en función de frecuencias

Fuente: [16]

### 2.1.7. Ventajas y desventajas del uso de RFID

A través de la comunicación entre etiquetas y lectores, es posible identificar y registrar información sin necesidad de contacto directo. No obstante, como cualquier tecnología, presenta tanto ventajas como desventajas que deben ser consideradas al momento de su implementación.

**Ventajas:**

- **Capacidad de lectura múltiple:** Un lector RFID es capaz de leer varias etiquetas casi simultáneamente, esto permite agilizar mucho algunos procesos.
- **Capacidad de reescritura:** Los códigos de barras solo se pueden escribir con datos una vez, pero las etiquetas RFID se pueden actualizar tantas veces como sea necesario.
- **Sin necesidad de línea de visión:** Las etiquetas no requieren una línea de visión directa con el lector, permitiendo el rastreo y la lectura incluso cuando están ocultas o incrustadas dentro de objetos. A diferencia de los códigos de barras por ejemplo.
- **Durabilidad:** Estas etiquetas son por norma general más robustas y duraderas que las etiquetas con códigos de barras, ya que pueden resistir condiciones adversas como humedad o suciedad.
- **Seguridad:** Las etiquetas pueden ser equipadas con medidas de seguridad, como cifrado, para proteger la información almacenada.

#### Desventajas:

- **Costo:** El coste inicial para la implementación de un sistema RFID, incluyendo lectores y etiquetas, es significativamente mayor que otros sistemas, como el de códigos de barras.
- **Interferencias:** La presencia de metales o líquidos puede interferir con la señal RFID (dependiendo de la frecuencia), lo que dificulta la lectura de las etiquetas en ciertos escenarios.
- **Limitaciones de rango:** Aunque el RFID puede tener un gran alcance en comparación con otras tecnologías, aún tiene limitaciones, especialmente con etiquetas pasivas.
- **Problemas de privacidad:** Existe la preocupación de que los sistemas RFID puedan ser utilizados para rastrear indebidamente a individuos o recolectar datos sin consentimiento.
- **Compatibilidad:** Puede haber problemas de compatibilidad entre las diferentes frecuencias RF, lo que complica la integración y el intercambio de información entre sistemas.

## 2.2. Sistemas de identificación

En el mundo contemporáneo, la necesidad de identificar, rastrear y gestionar objetos, personas o información ha llevado al desarrollo de diversas tecnologías de identificación. Estas tecnologías juegan un papel muy importante en muchas aplicaciones que abarcan desde la simple identificación de productos en tiendas hasta la gestión de una cadena de suministro o el control de acceso a instalaciones. El proceso de identificación es fundamental en muchas operaciones y negocios tanto por seguridad como por organización. A medida que la demanda tanto de precisión como de seguridad o incluso velocidad en la identificación ha aumentado, también lo ha hecho la diversidad de tecnologías disponibles.

Para analizar las tecnologías líderes en identificación, se elaboró una tabla de contraste entre las soluciones más populares del sector. Estos criterios de análisis provienen del estudio "Aspectos de Seguridad y Aplicaciones Prospectivas", ejecutado por la Oficina Federal para la Seguridad de la Información de Alemania, con la cooperación del Instituto de Estudios Futuros y Evaluación Tecnológica y los Laboratorios Federales Suizos para Pruebas de Materiales e Investigación. Los parámetros de evaluación fueron los siguientes[17]:

- **Modificación de Datos:** Posibilidad de ajustar o añadir nueva información, así como su reutilización.
- **Seguridad de Datos:** Habilidad de cifrar el contenido almacenado.
- **Cantidad típica de datos (byte):** Volumen de datos que puede contener.
- **Coste, cuánto cuesta:** Más allá del coste individual, es crucial considerar inversiones en conectividad, sistemas de protección, mantenimiento y actualización de dispositivos.
- **Estándares:** Preferir normas generalizadas aceptadas por múltiples productores y consumidores, evitando soluciones exclusivas de un solo proveedor.
- **Desgaste:** Periodo en el que la información es accesible, determinando su vida útil.
- **Distancia de lectura:** La necesidad o no de una línea de visión directa y el alcance efectivo entre el dispositivo lector y el etiquetado.
- **Interfaz:** Modalidad de interacción entre el lector y el etiquetado.
- **Susceptible a la suciedad/líquidos:** El grado en que estos factores pueden afectar a la detección.
- **Influencia en la dirección y posición:** Factores externos que podrían interferir y causar inconvenientes en el sistema.

La Tabla 2.2 presenta una evaluación realizada con la colaboración de un conjunto de profesores y alumnos avanzados de FISU, se basó en un cuestionario aplicado a 44 estudiantes avanzados, que trabajan en distintas entidades del sector de tecnologías de la información. En la tabla, se exponen los resultados de una comparativa entre Tarjetas Magnéticas, Biometría y Memoria de Contacto. Considerando una escala del 1 al 10, siendo 10 la puntuación más alta por cada aspecto evaluado y con un máximo acumulable de 50 puntos (distribuidos en 5 aspectos), las Tarjetas Magnéticas alcanzaron 32 puntos, equivalentes al 64 %. La Biometría logró 29 puntos, representando un 58 %. Mientras que la Memoria de Contacto obtuvo 23 puntos, es decir, un 46 %. Estos resultados sugieren una inclinación hacia el uso de las Tarjetas Magnéticas debido a su mayor valoración. Sin embargo, la Tabla 2.3 destaca una preferencia aún mayor hacia la tecnología RFID, superando incluso a las tarjetas magnéticas en puntuación.

Actualmente el código de barras sigue siendo el sistema de identificación más conocido, sobre todo en las empresas, pero poco a poco la identificación por radiofrecuencia va ganando terreno. Aunque ambos sistemas están diseñados para automatizar el proceso de identificación de objetos y personas, operan de manera fundamentalmente diferente. Mientras que los códigos de barras han sido una solución confiable y económica durante décadas, el RFID ofrece capacidades avanzadas, pero a menudo a un costo más alto. Para

	Código de Barras	Memorias de contacto	Biometría	OCR	Tarjetas Magnéticas	Tarjetas Inteligentes	RFID	
							RFID Pasivo	RFID Activo
<b>Modificación de Datos</b>	No Modificable	Modificable	No Modificable	No Modificable	Parcialmente Modificable	Modificable	Modificable	Modificable
<b>Seguridad de Datos</b>	Seguridad mínima	Altamente seguro	Altamente seguro	Seguridad media	Seguridad media	Alta Seguridad	Rango de baja a alta seguridad	Alta Seguridad
<b>Cantidad típica de datos (byte)</b>	1 a 100	De 8Mb en adelante	Ninguno	Ninguno	16 a 64k	1 MB	Alrededor de 64 KBytes	Alrededor de 8 Mb
<b>Coste</b>	Muy Bajo	Alto (más de 1 \$ por etiqueta)	Ninguno	Medio	Medio	Medio	Medio (Unos 0.25 \$ por tag)	Muy alto (más de 10 \$ por tag)
<b>Estándares</b>	Estable e implantado	Propietario, sin estándar	Ninguno	Estable	Estable e implantado	Estable e implantado	Con estándares en fase de implantación	Propietario y estándares abiertos
<b>Desgaste</b>	limitado	limitado	Indefinido	Limitado	limitado	limitado	Ninguno	Ninguno
<b>Distancia de lectura</b>	Pocos centímetros	Contacto necesario	Contacto Directo	Contacto Directo	Contacto Directo	Contacto Directo	Del orden de 1 metro	Del orden de 100 metros
<b>Interfaz</b>	Lectura óptica directa	Contacto	Contacto	Contacto	Contacto	Contacto	Sin barreras aunque puede haber interferencias	Sin barreras aunque puede haber interferencias
<b>Susceptible a la Suciedad/liquidos</b>	Alto	Alto	Ninguno	Medio	Posible	Posible	Ninguno	Ninguno
<b>Influencia en la dirección y posición</b>	Ligero	Ligero	Muy Alto	Alto	Muy Alto	Ligero	Ninguno	Ninguno

**Tabla 2.1:** Comparación de las Principales Tecnologías de Identificación

**Fuente:** [17]

compararlos es mejor hacerlo con una tabla, para poder ver las diferencias rápidamente (Tabla 2.4).

A pesar de las diferencias mostradas anteriormente, ambos sistema son utilizados a día de hoy ya que el código de barras sigue siendo un gran sistema y no se ha quedado aún obsoleto (Figura 2.14).



#	CRITERIO	TARJETA MAGNÉTICA	BIOMETRÍA	MEMORIA DE CONTACTO
1	Costo	6	3	3
2	Rendimiento	6	8	6
3	Relación Costo Beneficio	7	3	3
4	Fiabilidad	8	7	6
5	Seguridad de la Información	5	8	5
Total:		32	29	23
%		64%	58%	46%

**Tabla 2.2:** Comparación de las tecnologías de Identificación Automática  
**Fuente:** [17]

#	CRITERIO	RFID	TARJETA MAGNÉTICA
1	Costo	2	6
2	Rendimiento	9	6
3	Relación Costo Beneficio	9	7
4	Fiabilidad	10	8
5	Seguridad de la Información	10	5
Total:		40	32
%		80%	64%

**Tabla 2.3:** Comparación entre Tarjeta Magnética y RFID  
**Fuente:** [17]

### código de barras

El código de barras solo contiene el "UPC" o código del fabricante



Define el producto (SKU) pero no diferencia entre ellos

### etiqueta RFID

La etiqueta RFID contiene el "UPC" del producto más un número serial que es único para cada artículo



+ Serial único del artículo  
1234567...678901



Define a ambos productos (SKU) e identifica a un producto específico

**Figura 2.14:** Comparación gráfica RFID y código de barras  
**Fuente:** Propia

<b>Criterio</b>	<b>Códigos de Barras</b>	<b>RFID</b>
Mecanismo de funcionamiento	Basado en la lectura óptica de datos representados en patrones de líneas.	Basado en la radiofrecuencia para transmitir datos de una etiqueta a un lector.
Distancia de lectura	Requiere línea directa de visión para escanear.	Puede leerse a distancia y sin línea directa de visión.
Durabilidad	Puede deteriorarse con el tiempo o dañarse fácilmente.	Robusto y resistente a daños externos.
Capacidad de datos	Almacena una cantidad limitada de datos.	Puede almacenar información extensa y ser regrabable.
Costo	Generalmente más económico.	Más costoso, pero ofrece capacidades avanzadas.
Eficiencia	Lectura individual de cada código.	Lectura simultánea de cientos de etiquetas a la vez.
Seguridad	Baja seguridad ya que cualquier persona es capaz de ver visualmente el código de barras impreso.	Alta seguridad debido a la información se encuentra protegida en chip y es posible cifrar los datos.
Aplicaciones comunes	Seguimiento de paquetes, identificación de productos.	Seguimiento de activos, control de acceso, logística avanzada.

**Tabla 2.4:** Comparación entre Códigos de Barras y Tecnología RFID

**Fuente:** Propia



# Capítulo 3

## Planteamiento de soluciones alternativas y justificación de la solución adoptada

En este apartado se plantean las diferentes opciones de herramientas que se han tenido en cuenta para la creación del proyecto, comparándolas y seleccionando la más adecuada para el trabajo que se va a realizar.

### 3.1. Estudio y selección de los componentes

En esta sección se presenta el análisis llevado a cabo para identificar los componentes más apropiados para la realización de este proyecto. Para cada categoría de componente, se han examinado varias opciones y se han elegido los más apropiados basándonos en criterios de calidad, sencillez y compatibilidad.

#### 3.1.1. Tecnología de identificación

Para la realización de este proyecto se han pensado múltiples tecnologías que podían funcionar y adecuarse a lo que se busca. La idea de este es sustituir al código de barras tradicional para poder digitalizarlo todo y hacer un gran seguimiento. Las tecnologías que más se adecuan a lo que se busca son la RFID y la tecnología NFC. RFID ofrece un rango de lectura amplio, que puede ir desde centímetros hasta varios metros. Esto permite escanear plantas y etiquetas de inventario a distancias variables sin necesidad de un contacto cercano. NFC en cambio tiene un rango de lectura muy limitado, generalmente inferior a 10 centímetros. Por otro lado esta la línea de visión, la radiofrecuencia no necesita una línea de visión directa entre el lector y las etiquetas RFID. Puede detectar etiquetas incluso si están dentro de cajas o estanterías mientras la tecnología NFC necesita una línea de visión directa y una proximidad cercana entre el dispositivo NFC y la etiqueta para funcionar. Otro factor importante es la eficiencia a la hora de escanear, mientras NFC está limitado a escanear una etiqueta NFC a la vez, la tecnología RFID es capaz de escanear múltiples etiquetas a la vez, esto es esencial en un vivero donde hay cientos de miles de plantas.

En resumen, la tecnología RFID es más adecuada para el seguimiento de inventario

en un vivero debido a su mayor alcance, capacidad de lectura sin contacto y eficiencia en la lectura de múltiples etiquetas a la vez. NFC es más adecuado para aplicaciones que requieren interacción cercana y bidireccional, como pagos móviles o autenticación. Además, las etiquetas RFID son más resistentes a condiciones ambientales adversas, como el polvo o la humedad, en comparación con las etiquetas NFC.

### 3.1.2. Microcontrolador

Un microcontrolador es un circuito integrado que incluye en su interior las tres unidades fundamentales de cualquier computadora: CPU (Unidad Central de Procesamiento), memoria y periféricos de entrada/salida. Está diseñado para ser el “cerebro” de dispositivos electrónicos y es capaz de ejecutar un software almacenado en su memoria. A menudo se utiliza en dispositivos y sistemas automáticos como electrodomésticos, automóviles, dispositivos móviles y otros sistemas de control embebidos debido a su tamaño compacto, bajo consumo de energía y costo relativamente bajo.

Para este proyecto se han tenido en cuenta varias opciones posibles que usar como microcontrolador, la primera idea fue el uso de alguna variante de Arduino, como por ejemplo Arduino Uno o Arduino Nano. Esto es debido a su sencillo y familiar lenguaje a la hora de programar, además cuenta con muchas bibliotecas interesantes y con extenso soporte en internet gracias a su amplia comunidad. Pero para este proyecto es más conveniente utilizar un microprocesador más potente y con más características. Como por ejemplo el uso de Wi-Fi, ya que este proyecto está en línea en todo momento para asegurar el correcto valor de los diferentes datos y por lo tanto es necesaria su conexión a internet.

Por lo tanto las alternativas más competentes son la ESP32 y la ESP8266 [18]. Estas son placas bastante similares, y es importante destacar que ambas placas pueden ser programadas usando el entorno de programación IDE de Arduino. Aunque hay que tener en cuenta que estas puede que no sean compatibles con las mismas bibliotecas y funciones, hay algunas bibliotecas que son solo compatibles con determinadas placas, por lo que la mayoría de veces un mismo código en ESP8266 no será compatible con el ESP32. Para descubrir sus diferencias principales podemos usar la tabla 3.1:

Características	ESP32 Dev Kit	ESP8266
CPU	32 bits, Dual-core, 160-240 MHz	32 bits, Single-core, 80-160 MHz
RAM	520 KB	160 KB
Pines GPIO	36 (no todos disponibles para uso general)	17 (dependiendo del módulo)
WiFi	2.4 GHz y 5 GHz	2.4 GHz
Almacenamiento Flash	4 MB - 16 MB	512 KB - 4 MB
Bluetooth	Sí (Bluetooth y BLE)	No
Periféricos de Hardware	ADC, DAC, touch sensors, etc.	Limitado
Consumo de Energía	Modo de bajo consumo disponible	Mayor consumo de energía
Precio	14€	8€

**Tabla 3.1:** Comparación entre ESP32 Dev Kit y ESP8266

**Fuente:** Propia

Por lo tanto si hay que decantarse por una de las dos hay que tener en cuenta que el coste del ESP8266 es menor en comparación con el ESP32. A pesar de que no ofrece tantas características, es adecuado para la mayoría de los proyectos básicos. No obstante, presenta ciertas restricciones en cuanto a la configuración de los pines GPIO, ya que el ESP32 tiene mayor número de pines.

El ESP32 es considerablemente más robusto que el ESP8266, dispone de más pines GPIO multifuncionales, una conexión Wi-Fi más rápida y también es compatible con Bluetooth. Este también consume más energía que el ESP8266 pero este proyecto al no estar alimentado por baterías es un aspecto despreciable.

El ESP32 también tiene algunos aspectos negativos. Su precio es mayor que el del ESP8266. Por lo tanto, si se está desarrollando un proyecto sencillo de entrada/salida, el ESP8266 podría ser suficiente y más económico. Además, dado que el ESP8266 ha estado en el mercado más tiempo que el ESP32, algunas librerías y características están más desarrolladas para el ESP8266, y hay más recursos disponibles (foros, soluciones a problemas comunes, etc.). No obstante, con el tiempo, el ESP32 está ganando popularidad, y estas diferencias en cuanto a desarrollo y librerías se están volviendo menos notables.

Uno de los aspectos más importantes es el procesador de doble núcleo con el que cuenta el ESP32 lo que significa que puede manejar tareas múltiples más eficientemente que el ESP8266, que tiene un procesador de un solo núcleo.

En resumen, para este proyecto, el ESP32 [19] [20] parece ser la opción más adecuada debido a su mayor número de pines GPIO y capacidad de procesamiento.

### 3.1.3. Módulo lector RFID

La identificación por Radiofrecuencia permite la identificación automática de objetos, personas y animales utilizando ondas de radio. Estas ondas reflejan la información contenida en una etiqueta RFID, por lo tanto es muy importante contar con un módulo que cumpla con los requerimientos planteados.

Dentro de las opciones consideradas para el proyecto, resaltan dos módulos RFID por sus capacidades: el RFID-RC522 y el LECTOR RFID/NFC PN532.

El módulo RFID-RC522 es conocido por operar en la banda de 13.56 MHz. Se ha ganado una reputación como una solución económica y eficiente para sistemas de identificación. Su rango de lectura puede llegar hasta los 10 cm, se basa en la norma de comunicación SPI, cuenta con un bajo consumo energético y ofrece una amplia disponibilidad de librerías y soporte para plataformas como Arduino.

Por otro lado, el lector PN532 no es solo un módulo RFID, sino que es capaz de comunicarse con dispositivos mediante NFC (Near Field Communication), lo que le brinda una versatilidad adicional. Tiene un rango de lectura de hasta 7 cm, pero supera al RC522 en cuanto a interfaces de comunicación, ya que es compatible con I2C, SPI y UART. Este cuenta con un amplio respaldo de la comunidad dada su adopción en múltiples proyectos y aplicaciones comerciales.

Por lo tanto, la decisión entre estos dos módulos se basa en varios aspectos. Ambos módulos son robustos y aptos para este proyecto, por un lado este puede beneficiarse de la versatilidad del PN532 si se requiere NFC, permitiendo una gama más amplia de aplicaciones. No obstante, el RFID-RC522 tiene un rango de lectura ligeramente superior, lo que es un gran punto a favor en este tipo de proyectos. En cuanto a la interfaz de

comunicación, el PN532 brinda una flexibilidad superior al ofrecer I2C, SPI y UART, y el RC522 solo SPI, dado a que se va a trabajar con entorno Arduino la comunicación se establecerá vía SPI por lo tanto es indiferente en este sentido la elección.

Sin embargo, un factor determinante puede ser el coste. El RFID-RC522 suele tener un precio más asequible que el PN532. Tras tener en cuenta todas estas consideraciones, y sabiendo las necesidades específicas del proyecto, se ha decidido que el RFID-RC522 es la elección más adecuada, principalmente por su rango de lectura, su relación costo-beneficio y la amplia documentación y soporte disponible. [21] [22]

### 3.1.4. Método de intercambio de datos con el servidor

Para el intercambio de datos entre el servidor y el cliente se plantearon 2 alternativas. El método *GET* y el método *POST*. Estos son dos de los métodos más utilizados para enviar datos desde un cliente a un servidor en el contexto web [23]. Ambos tienen sus propias características y usos típicos.

1. **GET:** Los datos enviados mediante GET se añaden a la URL en la forma de parámetros de consulta. Por ejemplo: `http://ejemplo.com/script.php?rfid=123456`. Esto significa que los datos son visibles para cualquiera que vea la URL. Debido a que los datos se envían en la URL, hay una limitación en la cantidad de datos que puedes enviar. Aunque el límite exacto puede variar según el navegador y el servidor, generalmente se recomienda no superar los 2.048 caracteres. Se utiliza principalmente para solicitar datos donde no se requiere privacidad o seguridad, por ejemplo, al abrir una página web.
2. **POST:** Los datos enviados mediante POST se transmiten en el cuerpo de la solicitud HTTP y no son visibles en la URL. Esto es más seguro en comparación con GET cuando se envían datos sensibles. Por lo tanto, al no escribirse en la url no hay una limitación en la cantidad de datos que se pueden enviar, esto hace a este método más adecuado a la hora de enviar grandes cantidades de datos. Se utiliza principalmente para enviar datos que deben ser procesados por el servidor, como enviar un formulario que tiene información del usuario, especialmente cuando la información es sensible o privada.

Como se puede ver al comparar ambos métodos, el método POST es mejor tanto en seguridad como en longitud de los datos. Para la elaboración de este proyecto finalmente se ha decidido usar el método GET, esto es debido a que este es más fácil de implementar y la seguridad no es necesaria en este caso ya que no hay información privada de ningún tipo, por otro lado los datos no tienen mucho volumen por lo tanto con el GET es suficiente.

# Capítulo 4

## Descripción detallada de la solución adoptada

### 4.1. Descripción del sistema

En este punto se va a desarrollar todos los elementos tanto a nivel de hardware como de software que se utilizan en la realización del proyecto.

#### 4.1.1. Hardware

##### ESP32-DevKitC V4

Tal y como se ha mencionado en la sección 3.1.2, el microcontrolador escogido es el ESP32 V4, debido en gran parte a su gran capacidad de procesamiento y su mayor número de pines GPIO.

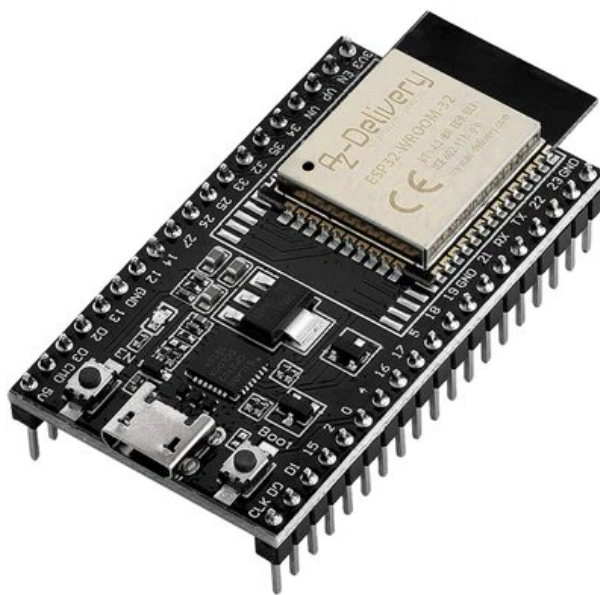


Figura 4.1: ESP32-DevKitC V4

Fuente: [24]



Esta tarjeta de desarrollo es el cerebro del proyecto.[19]

**Especificaciones y características:**

- Microcontrolador: ESP32 de Espressif Systems.
- CPU: Dual core Tensilica LX6 (32 bit)
- Velocidad del reloj: Hasta 240 MHz.
- Memoria flash: 4 MB.
- Memoria RAM: 520 KB estática y 8 MB de memoria SPI PSRAM (opcional).
- Conectividad inalámbrica: Wi-Fi, Bluetooth 4.2 y Bluetooth Low Energy (BLE).
- Interfaces: UART, SPI, I2C, I2S, ADC, DAC, PWM, GPIO.
- Conexión USB: Puerto micro USB para alimentación y comunicación con la computadora.
- Botones y LED: Botones de reinicio y de usuario, y LED de indicación de estado.
- Antena: Antena integrada en la placa.
- Alimentación: Se puede alimentar mediante el puerto USB o con una fuente de alimentación externa de 5V.
- Dimensiones: Aproximadamente 54mm x 28mm.
- Software de programación: MicroPython, LUA y Arduino.

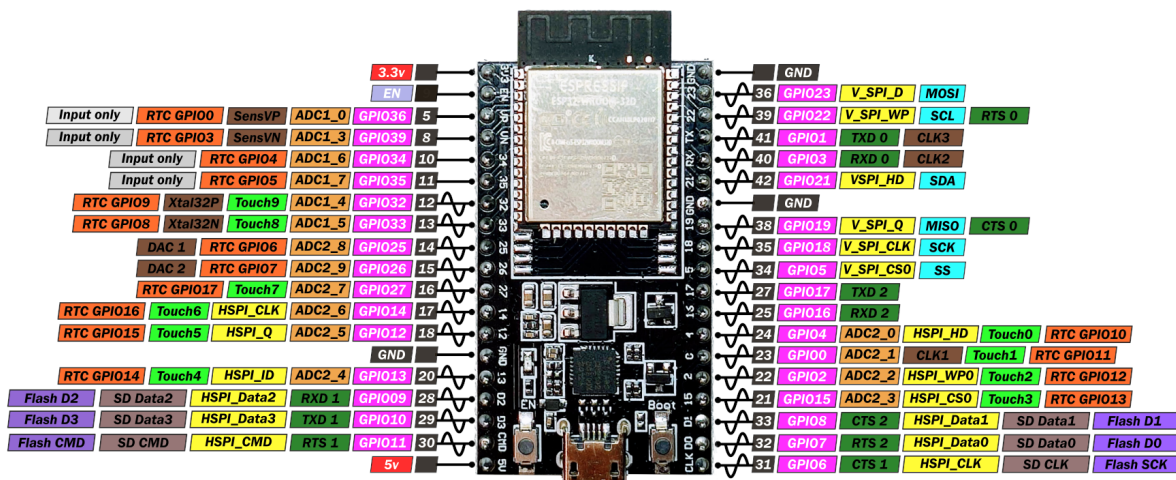


Figura 4.2: PINOUT ESP32  
Fuente: [25]

## RFID-RC522

Tal y como se ha mencionado en la sección 3.1.3, el microcontrolador escogido es el RFID-RC522, debido en gran parte a su económico precio y sus características técnicas.



**Figura 4.3:** Lector RFID RC522

**Fuente:** [26]

Es el lector encargado de leer todos los tags RF[27].

### Especificaciones y características:

- Interfaz de comunicación: SPI (Serial Peripheral Interface).
- Frecuencia de operación: 13.56 MHz.
- Soporte para los estándares RFID: ISO/IEC 14443A.
- Distancia de lectura: Aproximadamente 3-5 cm.
- Protocolo de comunicación: ISO/IEC 14443A y compatible con MIFARE Classic 1K y MIFARE Classic 4K.
- Modulación: ASK (Amplitude Shift Keying).
- Velocidad de transmisión: Hasta 10 Mbps (SPI).
- Memoria interna: 64 bytes de memoria de usuario.
- Alimentación: Voltaje de 3.3V (requiere regulador de voltaje si se utiliza con una placa de 5V).
- Consumo de energía: 13-26 mA en operación.

- Antena: Antena incorporada en la placa.
- Dimensiones: Aproximadamente 40mm x 60mm.
- Compatibilidad: Compatible con varias plataformas de desarrollo, como Arduino, Raspberry Pi, entre otras.

El lector RFID RC522 tiene 8 pines para conectarse a un microcontrolador:

- **SDA (NSS):** Serial Data Signal. Es el pin de selección de esclavo utilizado por el bus SPI para iniciar la comunicación con el módulo RC522.
- **SCK:** Serial Clock. Pin del reloj utilizado por el bus SPI.
- **MOSI:** Master Out Slave In. Pin de datos de salida desde el microcontrolador maestro hacia el módulo esclavo.
- **MISO:** Master In Slave Out. Pin de datos de entrada al microcontrolador maestro desde el módulo esclavo.
- **IRQ:** Interruption Request. Es un pin que puede ser utilizado para manejar interrupciones.
- **GND:** Ground. Pin de conexión a tierra.
- **RST:** Reset. Pin para reiniciar el módulo.
- **3.3V:** Alimentación. Se debe alimentar con 3.3V para asegurar la operación adecuada del módulo.

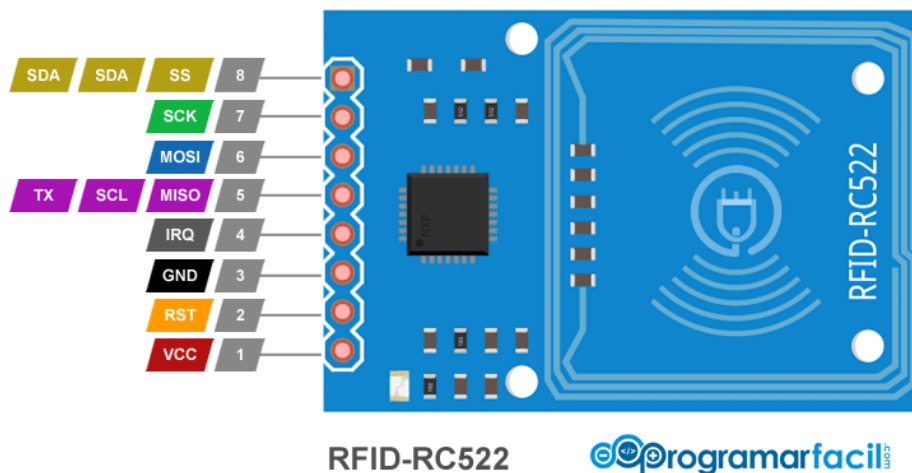


Figura 4.4: Pineado del lector RFID RC522

Fuente: [26]

### 4.1.2. Software

El software desarrollado para este proyecto se compone de una aplicación de escritorio, scripts PHP, una base de datos y un programa para el microcontrolador. Todos estos elementos han sido cuidadosamente diseñados e implementados para trabajar de manera conjunta y garantizar una gestión eficaz y en tiempo real del inventario de plantas de los invernaderos.

## Arduino

El microcontrolador ESP32, programado con el lenguaje Arduino, se utiliza para leer las etiquetas RFID de las plantas. Estos dispositivos están configurados para identificar cada planta y transmitir esta información a la base de datos central. Para lograr esto, se ha desarrollado un programa que permite leer la información de las etiquetas RFID y enviarla a la base de datos a través de una conexión a Internet. Además, este programa también se encarga de traducir los diferentes identificadores de cada etiqueta a una planta específica.

## Scripts PHP

Los scripts [PHP](#) (Hypertext Preprocessor) actúan como intermediarios entre la base de datos y la aplicación de escritorio. Estos scripts están alojados en un servidor web y se encargan de recibir las peticiones HTTP de la aplicación, procesarlas, y realizar las operaciones correspondientes en la base de datos, como consultar, insertar, modificar o eliminar datos, para luego devolver la respuesta a la aplicación. En este proyecto, se han desarrollado dos scripts principales. El primero se encarga de recibir datos del microcontrolador y almacenarlos en la base de datos. Por otro lado, el segundo script se encarga de consultar y devolver la información relacionada con un invernadero específico de la base de datos.

## Base de datos

La base de datos es el componente central donde se almacena toda la información relativa a las plantas e invernaderos. Es fundamental para la gestión eficaz y en tiempo real del inventario de plantas. La base de datos está diseñada para almacenar información como el [ID](#) (Identification) de la planta, el tipo de planta, la cantidad de plantas y en qué invernadero se encuentran. También registra los datos de las etiquetas RFID asociadas a cada planta. Se ha utilizado un sistema de gestión de bases de datos [DBMS](#) (DataBase Management System) para crear, recuperar y administrar la base de datos. Las operaciones en la base de datos son realizadas por los scripts PHP mencionados anteriormente, que interactúan con la base de datos para consultar, insertar, modificar o eliminar datos según sea necesario.

## Visual Studio

Visual Studio es el [IDE](#) (entorno de desarrollo integrado) utilizado para desarrollar la aplicación de escritorio. Facilita la creación, el desarrollo y la prueba de la aplicación. La aplicación de escritorio es la interfaz gráfica que utiliza el personal del vivero para gestionar el inventario de plantas en tiempo real. Permite realizar operaciones como consultar el inventario actual, añadir o eliminar plantas, y verificar el estado de cada invernadero. La aplicación de escritorio se comunica con los scripts PHP alojados en el servidor web para realizar operaciones en la base de datos y reflejar los cambios en tiempo real. Visual Studio proporciona herramientas y servicios que hacen que el desarrollo de esta aplicación sea más eficiente y fácil de manejar.

## 4.2. Implementación del hardware

Esta sección describe el conexionado de los componentes a nivel de hardware para el funcionamiento óptimo del trabajo.

### 4.2.1. Diagrama de conexiones

Para la realización del proyecto, el conexionado es muy simple. En la figura 4.5 se puede apreciar el microcontrolador ESP32 conectado al lector RFID-RC522.

El módulo RC522 RFID utiliza el protocolo SPI para comunicarse con el ESP32. El pinout es el siguiente (lado izquierdo RC522, lado derecho ESP32):

- Vcc <->3V3
- RST (Reset) <->D0
- GND (Masse) <->GND
- MISO (Master Input Slave Output) <->19
- MOSI (Master Output Slave Input) <->23
- SCK (Serial Clock) <->18
- SS/SDA (Slave select) <->5

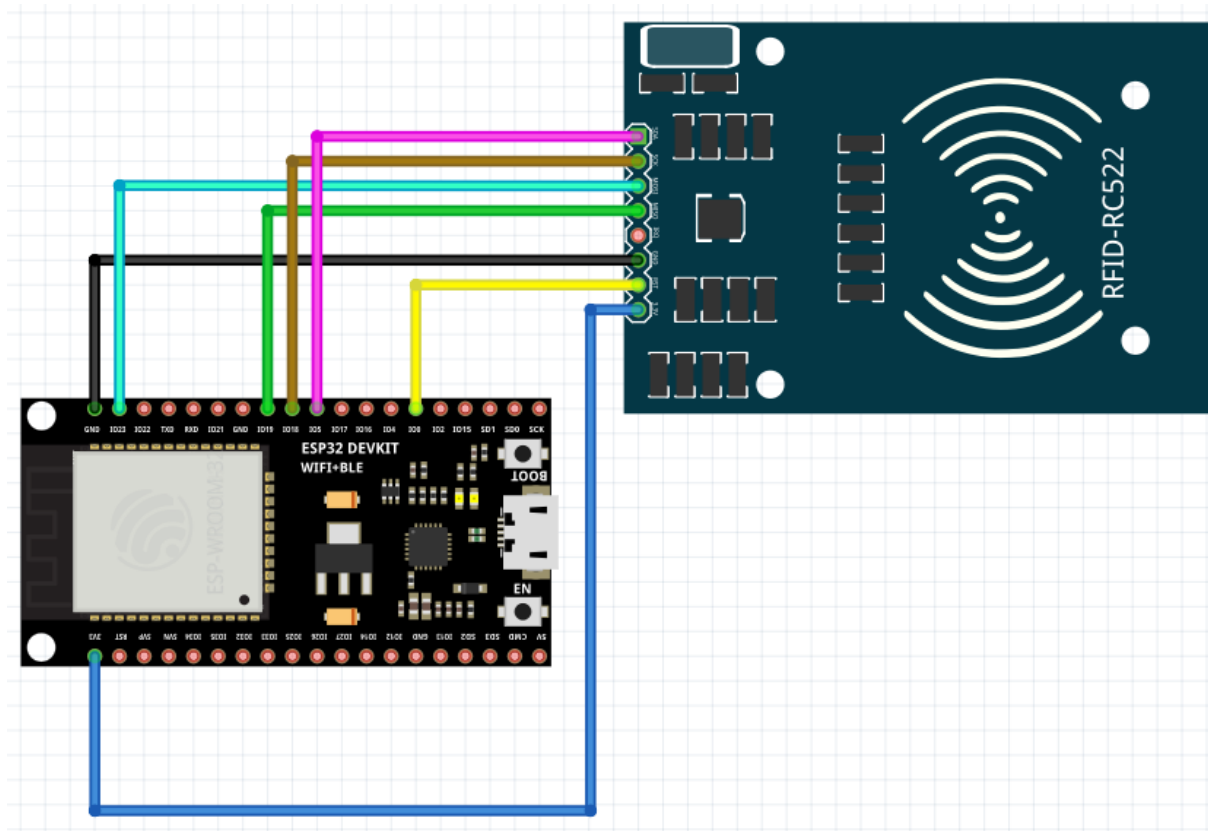


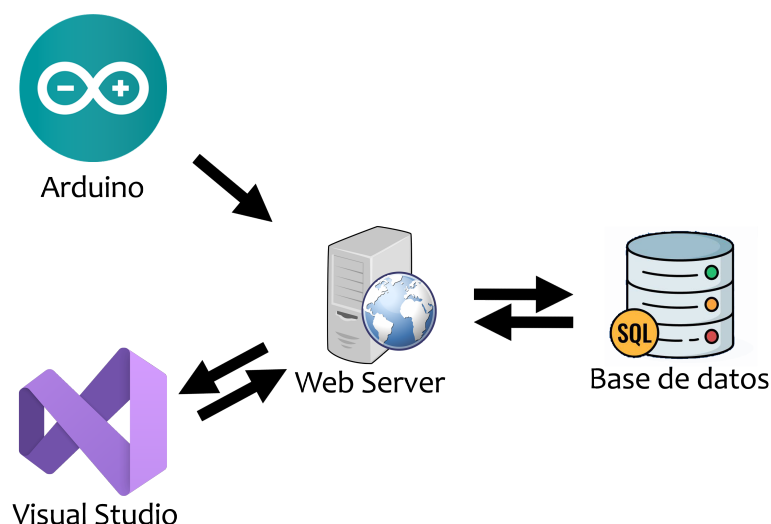
Figura 4.5: Diagrama de conexiones del sistema

Fuente: Propia

### 4.3. Implementación del software

El sistema de software es un componente crucial de este proyecto, ya que facilita la gestión eficiente y en tiempo real del inventario de plantas en los invernaderos. Este sistema está compuesto por varios elementos interconectados, incluyendo un microcontrolador programado con el lenguaje Arduino, scripts PHP alojados en un servidor, una base de datos y una aplicación de escritorio desarrollada en Visual Studio. Cada uno de estos elementos desempeña un papel vital en la recopilación, procesamiento, almacenamiento y visualización de la información relacionada con el inventario de plantas. Esta sección tiene como objetivo proporcionar una descripción detallada de cada uno de estos componentes, explicando su función, cómo fueron implementados y cómo interactúan entre sí para formar un sistema cohesivo.

Para entender de forma visual como está organizado el software del proyecto se puede seguir el diagrama de conexiones (Figura 4.6).



**Figura 4.6:** Diagrama de conexiones de software  
**Fuente:** Propia

#### 4.3.1. Arduino

Todo el proceso comienza en Arduino, la idea de este es usarlo para escanear las etiquetas RF B.1. Lo primero que se hace en el código de arduino es incluir las bibliotecas necesarias: *WiFi* y *MFRC522*. La biblioteca WiFi proporciona la capacidad de conectarse a una red WiFi y la biblioteca *MFRC522* contiene funciones y métodos para interactuar con tarjetas y etiquetas RFID usando el módulo RFID-RC522.

Luego, se definen varias constantes y variables que se utilizarán en todo el programa.

*RST\_PIN* y *SS\_PIN* son definidos como los pines a los que están conectados los pines de *reset* y *SS* (*slave select*) del módulo RFID, respectivamente. También se definen dos constantes, *AGREGAR* y *RETIRAR*, que son utilizadas para indicar los dos modos de operación diferentes en el programa. La variable *modoActual* se inicializa como *AGREGAR*, para que el programa comience en un modo donde se pueden agregar nuevas tarjetas RFID. La variable *invernaderoActual* se inicializa como “1”, lo que indica que el programa comienza operando en el invernadero número 1.

A continuación, se definen las credenciales de la red WiFi a la que el ESP32 intentará conectarse. *ssid* y *password* se inicializan con el nombre y la contraseña de la red, respectivamente. Es importante tener en cuenta que estas credenciales están codificadas en el programa, lo que significa que cualquier persona con acceso al código puede obtenerlas.

Finalmente, se crean instancias de las clases *MFRC522* y *WiFiClient*. *rfid* es una instancia de la clase *MFRC522* y se inicializa con los pines *SS\_PIN* y *RST\_PIN*. Esta instancia se utilizará para interactuar con el módulo RFID, leer tarjetas y etiquetas RFID y obtener su **UID** (Identificador Único). Por otro lado, *client* es una instancia de la clase *WiFiClient*, que proporciona la capacidad de crear un cliente que puede conectarse a una dirección IP y puerto especificados en internet o en una red local.

El fragmento de código correspondiente a la función *setup* del programa, se encarga de inicializar las comunicaciones y conectar el dispositivo a la red WiFi. Primero, se inicializan las comunicaciones serie y *SPI*, necesarias para la interacción con el módulo RFID y el monitor serie. Luego, se inicializa el módulo RFID y se inicia el proceso de conexión a la red WiFi con las credenciales *ssid* y *password*. El programa entra en un bucle que espera hasta que la conexión WiFi esté establecida, imprimiendo “*Conectando a WiFi...*” en el monitor serie cada segundo. Una vez establecida la conexión, imprime “*Conectado a WiFi*” en el monitor serie.

Después crea una función llamada *rfidDatabase* (Código: 4.1), esta toma el string llamado *rfidCode* como argumento y devuelve otro *String* (el string *rfidCode* es el que contiene el último id escaneado). La función verifica el *rfidCode* proporcionado y devuelve el nombre de la planta asociada a ese código. Si el *rfidCode* proporcionado no coincide con ninguno de los códigos definidos en la función, devolverá “*Desconocida*”. Esto significa que si se lee un código RFID que no está en la lista, el sistema lo identificará como una planta desconocida.

```

1 // Se define el codigo RF que tiene asociada cada planta
2 String rfidDatabase(String rfidCode) {
3   if (rfidCode == "F189B51B") return "Geranio";
4   if (rfidCode == "FA686B0") return "Begonia";
5   if (rfidCode == "E11A9F1B") return "Orquídea";
6   if (rfidCode == "D08B7C1D") return "Petunia";
7   if (rfidCode == "C09B831D") return "Poinsettia";
8   if (rfidCode == "C0312A1D") return "Hortensia";
9   if (rfidCode == "9271551D") return "Crisantemo";
10  if (rfidCode == "D09E9A1D") return "Vinca";
11  return "Desconocida";
12 }

```

Código 4.1: Asignación plantas con RFID

Después se encuentra el void loop del programa (Código: 4.2), esta es la función principal que se ejecuta continuamente en un sketch de arduino. En este caso, realiza varias tareas importantes: Primero, verifica si hay datos disponibles en el puerto serie. Si hay

datos, los lee y realiza una acción dependiendo del contenido de los datos. Si los datos son “AGREGAR” o “RETIRAR”, cambia el *modoActual* a *AGREGAR* o *RETIRAR*, respectivamente, y luego imprime un mensaje en el monitor serie confirmando el cambio. Si los datos comienzan con “INVERNADERO:”, cambia el *invernaderoActual* al número de invernadero especificado en los datos y luego imprime un mensaje en el monitor serie confirmando el cambio. Si los datos comienzan con “TOTAL\_PLANTAS:”, llama a la función *consultarTotalPlantas* con el número de invernadero especificado en los datos y luego sale de la función *loop*.

Luego, el código verifica si hay una nueva tarjeta RFID presente y si se puede leer su número de serie. Si no, sale de la función *loop*. Si sí, construye el código RFID a partir del número de serie de la tarjeta, convierte el código a mayúsculas, busca el nombre de la planta asociada a ese código en la función *rfidDatabase* y luego llama a la función *enviarDatos* con el código RFID y el nombre de la planta.

```

1 void loop() {
2   // Verificar si hay datos disponibles en el puerto serie
3   if (Serial.available() > 0) {
4     String input = Serial.readStringUntil('\n');
5     input.trim();
6
7     // Cambiar el modo de operación a AGREGAR
8     if (input == "AGREGAR") {
9       modoActual = AGREGAR;
10      Serial.println("Modo cambiado a AGREGAR");
11
12     // Cambiar el modo de operación a RETIRAR
13     } else if (input == "RETIRAR") {
14       modoActual = RETIRAR;
15       Serial.println("Modo cambiado a RETIRAR");
16
17     // Cambiar el invernadero actual
18     } else if (input.startsWith("INVERNADERO:")) {
19       invernaderoActual = input.substring(12); // Extraer el número del
20       invernadero
21       Serial.print("Cambiado a Invernadero: ");
22       Serial.println(invernaderoActual);
23
24     // Consultar el total de plantas en un invernadero
25     } else if (input.startsWith("TOTAL_PLANTAS:")) {
26       consultarTotalPlantas(input.substring(13)); // Extraer el número del
27       invernadero para consulta
28       return;
29     }
30   }
31
32   // Proceso de lectura RFID y envío de datos
33   if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())
34     return;
35
36   String rfidCode = "";
37   for (byte i = 0; i < rfid.uid.size; i++) {
38     rfidCode += String(rfid.uid.uidByte[i], HEX);
39   }
40   rfidCode.toUpperCase();
41
42   String nombrePlanta = rfidDatabase(rfidCode);
43   enviarDatos(rfidCode, nombrePlanta);

```



```

42  delay(1000);
43  }

```

Código 4.2: Loop del programa

La función *enviarDatos* toma dos argumentos, *rfidCode* y *nombrePlanta*, y envía estos datos, junto con el *modoActual* y el *invernaderoActual*, a un servidor. (Código: 4.3)

Primero, construye la URL de la solicitud GET añadiendo *rfidCode*, *nombrePlanta*, *modoActual* e *invernaderoActual* a la URL base del servidor. Luego, intenta conectar al servidor usando el método *connect* de la clase *WiFiClient*. Si la conexión es exitosa, envía la solicitud GET al servidor. La solicitud GET se compone de varias líneas: la primera línea contiene el método HTTP (GET), la ruta del recurso en el servidor y la versión de HTTP; la segunda línea contiene el host; la tercera línea indica que la conexión debe cerrarse después de recibir la respuesta; y la cuarta línea está vacía, lo que indica el final de la solicitud.

Luego, lee la respuesta del servidor línea por línea. Si una línea comienza con “*Datos recibidos:*”, imprime esa línea en el monitor serie. Esto indica que el servidor ha recibido los datos correctamente.

Finalmente, cierra la conexión al servidor usando el método *stop* de la clase *WiFiClient*.

En resumen, la función *enviarDatos* envía el *rfidCode*, el *nombrePlanta*, el *modoActual* y el *invernaderoActual* a un servidor, lee la respuesta del servidor y cierra la conexión.

```

1 void enviarDatos(String rfidCode, String nombrePlanta) {
2   // Construir la URL para enviar los datos al servidor
3   String serverPath = "http://pruebaviverotfg.000webhostapp.com/
4     almacenar_rfid.php?";
5   serverPath += "rfid=" + rfidCode;
6   serverPath += "&nombre_planta=" + nombrePlanta;
7   serverPath += "&modo=" + String(modoActual);
8   serverPath += "&invernadero_id=" + invernaderoActual;
9
10  // Enviar la solicitud al servidor
11  if (client.connect("pruebaviverotfg.000webhostapp.com", 80)) {
12    client.println("GET " + serverPath + " HTTP/1.1");
13    client.println("Host: pruebaviverotfg.000webhostapp.com");
14    client.println("Connection: close");
15    client.println();
16
17    // Leer la respuesta del servidor
18    while (client.connected()) {
19      if (client.available()) {
20        String line = client.readStringUntil('\n');
21        if (line.startsWith("Datos recibidos:")) {
22          Serial.println(line);
23        }
24      }
25    }
26  }
27  client.stop();
28 }

```

Código 4.3: Función enviar datos

La función *consultarTotalPlantas* toma un argumento, *invernadero\_id*, y envía una solicitud al servidor para obtener el total de plantas en ese invernadero. (Código: 4.4)

Primero, construye la URL de la solicitud GET añadiendo `invernadero_id` a la URL base del servidor. Luego, intenta conectar al servidor usando el método `connect` de la clase `WiFiClient`. Si la conexión es exitosa, envía la solicitud GET al servidor. La solicitud GET se compone de varias líneas: la primera línea contiene el método HTTP (GET), la ruta del recurso en el servidor y la versión de HTTP; la segunda línea contiene el host; la tercera línea indica que la conexión debe cerrarse después de recibir la respuesta; y la cuarta línea está vacía, lo que indica el final de la solicitud.

Luego, lee la respuesta del servidor línea por línea. Si una línea comienza con “*Total Plantas:*”, imprime esa línea en el monitor serie. Esto indica que el servidor ha respondido con el total de plantas en el invernadero especificado.

Finalmente, cierra la conexión al servidor usando el método `stop` de la clase `WiFiClient`.

En resumen, la función `consultarTotalPlantas` envía el `invernadero_id` a un servidor, lee la respuesta del servidor, que debería ser el total de plantas en ese invernadero, e imprime esa respuesta en el monitor serie.

```

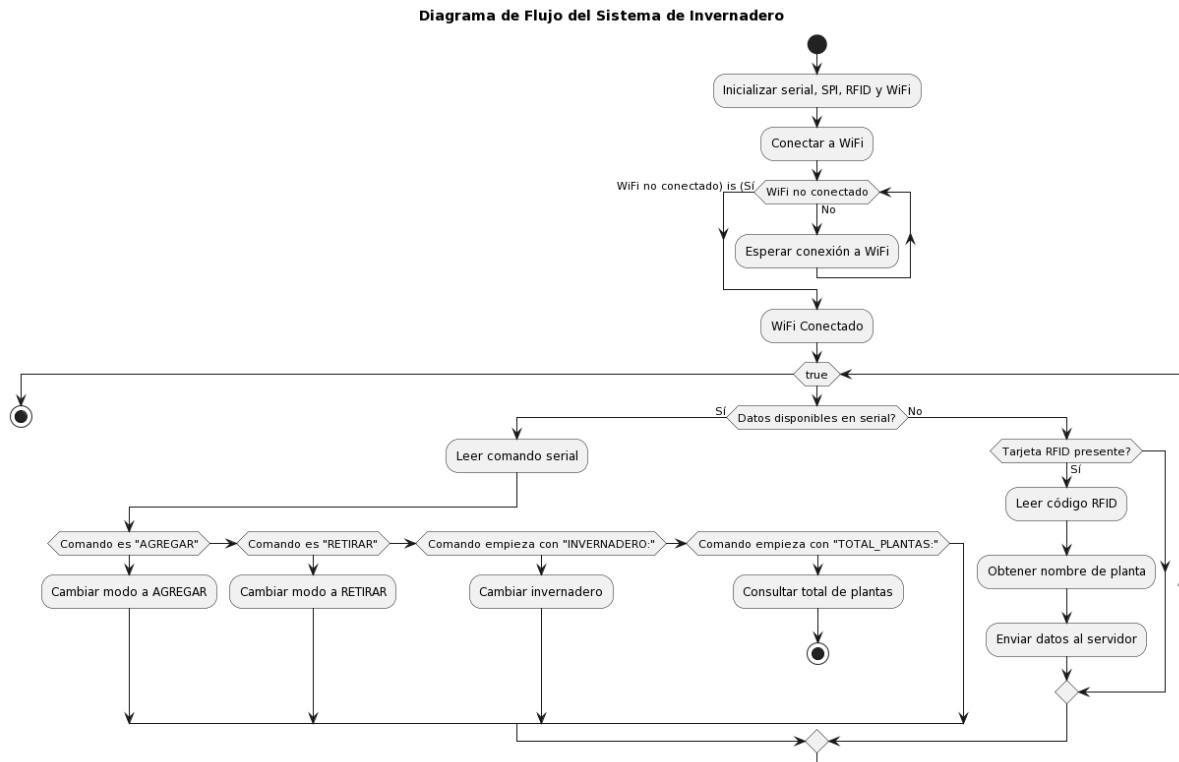
1 void consultarTotalPlantas(String invernadero_id) {
2   // Construir la URL para consultar el total de plantas en un invernadero
3   String serverPath = "http://pruebaviverotfg.000webhostapp.com/
4     almacenar_rfid.php?action=obtener_total&invernadero_id=" + invernadero_id;
5
6   // Enviar la solicitud al servidor
7   if (client.connect("pruebaviverotfg.000webhostapp.com", 80)) {
8     client.println("GET " + serverPath + " HTTP/1.1");
9     client.println("Host: pruebaviverotfg.000webhostapp.com");
10    client.println("Connection: close");
11    client.println();
12
13   // Leer la respuesta del servidor
14   while (client.connected()) {
15     if (client.available()) {
16       String line = client.readStringUntil('\n');
17       if (line.startsWith("Total Plantas:")) {
18         Serial.println(line);
19       }
20     }
21   }
22   client.stop();
23 }

```

Código 4.4: Función consultar datos

La estructura de `consultarTotalPlantas` es similar a la de `enviarDatos`, pero tienen diferentes propósitos y envían diferentes datos al servidor. Básicamente `enviarDatos` envía datos al servidor para almacenar, mientras que `consultarTotalPlantas` solicita información al servidor.

Para comprender el funcionamiento del código de arduino se ha realizado un diagrama de flujo detallado del código. (Figura 4.7)

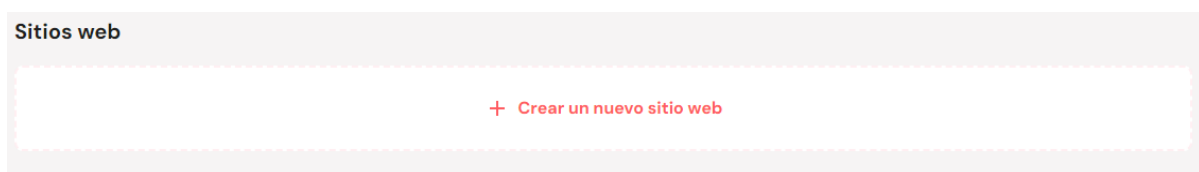


**Figura 4.7:** Diagrama de flujo Arduino  
**Fuente:** Propia

### 4.3.2. Servidor PHP

El siguiente paso es el servidor PHP, para alojar este proyecto se ha utilizado es.000webhost.com Este es un servicio de alojamiento web gratuito que permite a los usuarios crear sitios web sin costo alguno. Ofrece características básicas de alojamiento, como un subdominio gratuito, espacio en disco y ancho de banda limitados, y soporte para PHP y MySQL. Se ha utilizado esta web ya que es una de las mejores webs de hosting gratuitas y destaca por su facilidad para administrar sus archivos, cosa que va a ser muy útil para este proyecto ya que se usarán scripts PHP como intermediarios entre la base de datos y la aplicación de escritorio. Para la realización de este proyecto se usa esta web de hosting, si se desea poner en marcha en el futuro este proyecto habría que usar un servidor con mejores características además de añadir seguridad tanto en el servidor como en los scripts.

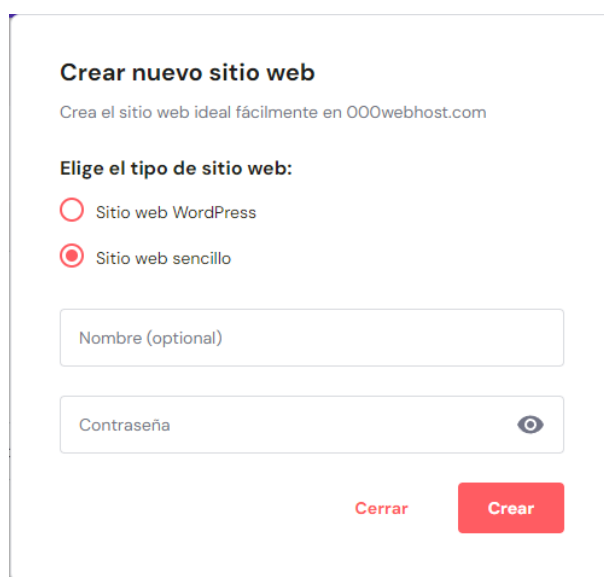
Lo primero que hay que hacer en la página es registrarse, una vez registrados habrá que crear un nuevo sitio web.(Figura 4.8).



**Figura 4.8:** Crear nuevo sitio web en 000webhost  
**Fuente:** Propia

Después habrá que elegir el tipo de sitio web y establecer un nombre y una contraseña

para este. (Figura 4.9).

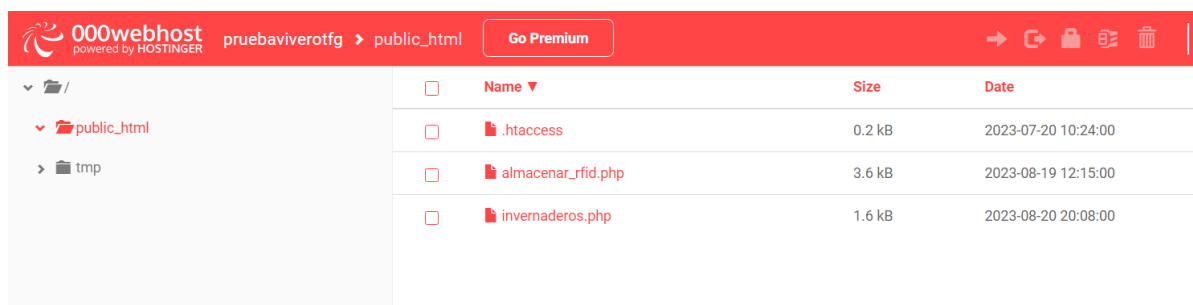


The screenshot shows a web form titled "Crear nuevo sitio web" (Create new website) on the 000webhost.com platform. The form prompts the user to "Crea el sitio web ideal fácilmente en 000webhost.com". Under the heading "Elige el tipo de sitio web:" (Choose the type of website:), there are two radio button options: "Sitio web WordPress" (WordPress website) and "Sitio web sencillo" (Simple website), with the latter being selected. Below these options are two input fields: "Nombre (optional)" (Name) and "Contraseña" (Password), which includes a toggle for password visibility. At the bottom right, there are two buttons: "Cerrar" (Close) and "Crear" (Create).

**Figura 4.9:** Crear nuevo sitio web en 000webhost 2  
**Fuente:** Propia

Ahora ya estará creado el servidor, si no se establece ninguna dirección web o dominio la propia página establecerá por defecto el nombre que se ha asignado a la hora de crear la página + .000webhostapp.com. En este caso la url de la web quedaría como: “*pruebaviverotfg.000webhostapp.com*”. Este es el nombre del host que se ha puesto en el programa de arduino para que se conecte.

Después de tener todo esto claro podemos entrar al administrador de archivos del servidor. Dentro de este vienen por defecto 2 carpetas: “*public\_html*” y “*tmp*”. Dentro de la carpeta “*public\_html*” es donde crearemos los scripts que servirán de intermediario entre la base de datos y la aplicación de escritorio (Figura 4.10). Para el correcto funcionamiento de este proyecto se han creado 2 scripts. El primero “*almacenar\_rfid.php*” se encarga de recibir datos del microprocesador y almacenarlos en la base de datos y el segundo “*invernaderos.php*” se encarga de consultar y devolver la información relacionada con un invernadero específico de la base de datos. Todo el código se encuentra disponible en el Anexo B.2.



The screenshot shows the file manager interface for a 000webhost account. The breadcrumb path is "pruebaviverotfg > public\_html". A table lists the files in the "public\_html" directory:

Name	Size	Date
.htaccess	0.2 kB	2023-07-20 10:24:00
almacenar_rfid.php	3.6 kB	2023-08-19 12:15:00
invernaderos.php	1.6 kB	2023-08-20 20:08:00

**Figura 4.10:** Scripts dentro de los archivos del servidor  
**Fuente:** Propia

El primer script es “almacenar\_rfid”, este script esta diseñado para gestionar la actualización de una base de datos de invernaderos. Establece una conexión con la base de

datos, recoge los datos enviados mediante una solicitud GET, y verifica y actualiza la base de datos según sea necesario. Dependiendo del modo (agregar o retirar) proporcionado en la solicitud, el script actualizará la cantidad de una planta específica en un invernadero y, finalmente, actualizará el total de plantas en el invernadero antes de cerrar la conexión con la base de datos.

Al comenzar el código se definen las variables necesarias para conectar al servidor de base de datos y luego se crea una conexión utilizando la función *mysqli*. Si hay un error en la conexión, el script se detiene y muestra un mensaje de “*Conexión fallida*”. Después se recogen los datos enviados en la solicitud GET. Si alguno de los datos necesarios no se ha recibido, el script se detiene y muestra un mensaje de “*No se recibieron todos los datos necesarios*”. Si todos los datos necesarios se han recibido, se imprime un mensaje indicando los datos recibidos.

En esta parte se verifica si existe una planta con el RFID proporcionado en la base de datos. Si no existe, se ejecuta un comando SQL para insertar una nueva planta con el nombre y RFID proporcionados. Si hay un error al insertar la nueva planta, se imprime un mensaje de error.

Después, se ejecuta un comando SQL para obtener el “*id*” de la planta asociada con el RFID proporcionado. Se asume que la planta existe en la base de datos en este punto, ya que se habría insertado en el paso anterior si no existía previamente.

Lo siguiente es verificar si ya existe una relación entre el invernadero y la planta proporcionados en la tabla “*invernadero\_planta*”. Si no existe, y estamos en “*modo agregar*” (modo = 1), se inserta una nueva relación con “*cantidad*” igual a 1. Si la relación ya existe, se actualiza la “*cantidad*” de plantas en la relación existente, incrementando o decrementando la cantidad según el modo proporcionado. Si hay un error al insertar o actualizar la relación, se imprime un mensaje de error.

Finalmente la variable *\$total\_plantas* es inicializada a 0 antes de calcular el total de plantas en un invernadero específico. Después se calcula el “*total\_plantas*” sumando la “*cantidad*” de todas las relaciones en “*invernadero\_planta*” para el “*invernadero\_id*” proporcionado. Luego, se actualiza la tabla “*invernaderos*” con este “*total\_plantas*”. Si hay un error al actualizar la tabla “*invernaderos*”, se imprime un mensaje de error. Al final del script se cierra la conexión a la base de datos.

Así es como queda el primer script. El segundo llamado *invernaderos.php* se conecta a la base de datos, obtiene información sobre el total de plantas y la cantidad de cada tipo de planta en un invernadero especificado, y luego envía esa información como respuesta en formato JSON.

El primer segmento de código del script establece el tipo de contenido de la respuesta como JSON e inicializa las variables que se utilizarán para la conexión a la base de datos. *\$host* es la dirección del servidor de la base de datos, *\$db* es el nombre de la base de datos, *\$user* es el nombre de usuario, *\$pass* es la contraseña y *\$charset* es el conjunto de caracteres que se utilizará para la conexión.

Después se configuran las opciones para la conexión PDO y se intenta establecer la conexión a la base de datos. *\$dsn* es una cadena que contiene la información necesaria para conectar con la base de datos y *\$options* es un array asociativo de opciones para la conexión PDO.

Se está obteniendo el valor de *id\_invernadero* desde la URL. Si *id\_invernadero* no

se proporciona en la URL, se usará un valor predeterminado de 1. Se utiliza `isset()` para verificar si `id_invernadero` está presente en la URL y se hace un casting a `int` para asegurarse de que el valor sea un número entero.

Se prepara consulta SQL para obtener el total de plantas en el invernadero especificado. Se utiliza `prepare()` para preparar la consulta SQL y `execute()` para ejecutarla. Se pasa `$invernaderoId` a `execute()` para reemplazar el marcador de posición `?` en la consulta SQL. Luego, se utiliza `fetchColumn()` para obtener el resultado de la consulta como un valor único. Este valor se almacena en `$totalPlantas`.

Luego se prepara y ejecuta otra consulta SQL para obtener la cantidad de cada tipo de planta en el invernadero especificado. Se utiliza `INNER JOIN` para combinar las tablas `plantas` e `invernadero_planta` basándose en los valores de `id` y `planta_id`, respectivamente. Luego, se utiliza `fetchAll()` para obtener todos los resultados de la consulta como un array de arrays asociativos. Este resultado se almacena en `$plantas`.

En el último segmento, se construye un objeto JSON con `$totalPlantas` y `$plantas` y se está enviando como respuesta usando `echo` y `json_encode()`. Si ocurre algún error durante la ejecución del script, se captura la excepción y se envía un mensaje de error en formato JSON.

### 4.3.3. Base de datos

Para la creación de la base de datos se ha usado la web también, esta base de datos estará programada en `mySQL`. Lo primero es crear la base de datos estableciendo un nombre para esta, un nombre de usuario y una contraseña. (Figura 4.11).

**Crear nueva base de datos**

Nombre  
vivero\_db

Nombre de usuario  
user\_vivero

Contraseña  
.....

✓ One number      ✓ One symbol  
✓ One lowercase letter      ✓ One uppercase letter  
✓ Use 8-50 characters      ✓ Only Latin letters

Cerrar      Crear

**Figura 4.11:** Crear nueva base de datos  
**Fuente:** Propia

Se ha creado esta base de datos con estas credenciales, es importante guardarlas ya que

son necesarias para que el script pueda acceder a esta y modificar los datos de forma automática. Para la gestión y programación de la base de datos se usa PhpMyAdmin.(Figura 4.12).

Nombre	Usuario	Host	Mesas	Tamaño	1 / 2
id21058636_vivero_db	id21058636_user_vivero	localhost	3	0MB	...

PhpMyAdmin  
 Cambiar contraseña  
 Eliminar

**Figura 4.12:** Información base de datos  
**Fuente:** Propia

phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando un navegador web. Una vez dentro se crean 3 tablas: *invernaderos*, *plantas* y *invernadero\_planta*. Todo el código se encuentra disponible en el Anexo B.3.

La cantidad invernaderos es fija y conocida, por lo tanto se ha pre-poblado la tabla *invernaderos* con estos registros. Para el nombre se les ha asignado “Invernadero X”, siendo X el número que le corresponda a cada invernadero.

- **id:** Es un entero *INT* que se incrementa automáticamente *AUTO\_INCREMENT* con cada nueva fila insertada. Esta columna es la clave primaria *PRIMARY KEY* de la tabla, lo que significa que cada valor en esta columna debe ser único.
- **nombre:** Es una cadena de caracteres de longitud variable *VARCHAR* con un máximo de 255 caracteres. Esta columna no puede tener valores nulos *NOT NULL*, lo que significa que cada fila en la tabla debe tener un valor para esta columna.
- **total\_plantas:** Es un entero *INT* que tiene un valor predeterminado de 0 *DEFAULT 0*, lo que significa que si no se especifica un valor para esta columna al insertar una nueva fila, se utilizará 0.

Después está la tabla *plantas*, esta tabla almacena información sobre los distintos tipos de plantas (sin referenciar a ningún invernadero en particular). La idea de esta tabla es asociar cada planta con el valor de su etiqueta RF y asignarles un ID.

- **id:** Similar a la columna id en la tabla invernaderos, es un entero *INT* que se incrementa automáticamente *AUTO\_INCREMENT* y es la clave primaria *PRIMARY KEY* de la tabla.
- **nombre\_planta:** Es una cadena de caracteres de longitud variable *VARCHAR* con un máximo de 255 caracteres. Esta columna no puede tener valores nulos *NOT NULL*.
- **rfid\_asociado:** Es una cadena de caracteres de longitud variable *VARCHAR* con un máximo de 255 caracteres. Esta columna no puede tener valores nulos *NOT NULL* y cada valor en esta columna debe ser único *UNIQUE*, lo que significa que no puede haber dos filas en la tabla con el mismo valor.

En este punto se genera una relación de muchos a muchos entre plantas e invernaderos. En otras palabras, una planta puede estar en múltiples invernaderos y, a la vez, un invernadero puede tener múltiples tipos de plantas. Para representar esta relación en la base de datos, se crea una tabla intermedia (también conocida como tabla de unión o tabla puente). Esta tabla intermedia contendrá claves foráneas que hagan referencia a las tablas que se desean relacionar. Una clave foránea es una columna (o un conjunto de columnas) en una tabla que se utiliza para establecer una relación con otra tabla. Hace referencia a la clave primaria de otra tabla.

Así es como surge la última tabla, la tabla intermedia llamada `invernadero_planta`.

- **invernadero\_id:** Es un entero *INT* que no puede tener valores nulos *NOT NULL*. Esta columna es una clave foránea *FOREIGN KEY* que hace referencia a la columna `id` en la tabla `invernaderos`.
- **planta\_id:** Es un entero *INT* que no puede tener valores nulos *NOT NULL*. Esta columna es una clave foránea *FOREIGN KEY* que hace referencia a la columna `id` en la tabla `plantas`.
- **cantidad:** Es un entero *INT* que no puede tener valores nulos *NOT NULL* y tiene un valor predeterminado de 0 *DEFAULT 0*.
- **PRIMARY KEY (invernadero\_id, planta\_id):** Esto especifica que la combinación de `invernadero_id` y `planta_id` debe ser única para cada fila en la tabla y juntas sirven como clave primaria de la tabla.

Así ya estaría completa la base de datos, desde phpMyAdmin se pueden ver los datos de cada tabla en tiempo real, así es como se ve:

				id	nombre	total_plantas			
<input type="checkbox"/>		Editar		Copiar		Borrar	1	Invernadero 1	39
<input type="checkbox"/>		Editar		Copiar		Borrar	2	Invernadero 2	11
<input type="checkbox"/>		Editar		Copiar		Borrar	3	Invernadero 3	1
<input type="checkbox"/>		Editar		Copiar		Borrar	4	Invernadero 4	6
<input type="checkbox"/>		Editar		Copiar		Borrar	5	Invernadero 5	10
<input type="checkbox"/>		Editar		Copiar		Borrar	6	Invernadero 6	6
<input type="checkbox"/>		Editar		Copiar		Borrar	7	Invernadero 7	1
<input type="checkbox"/>		Editar		Copiar		Borrar	8	Invernadero 8	6
<input type="checkbox"/>		Editar		Copiar		Borrar	9	Invernadero 9	3
<input type="checkbox"/>		Editar		Copiar		Borrar	10	Invernadero 10	9
<input type="checkbox"/>		Editar		Copiar		Borrar	11	Invernadero 11	4

**Figura 4.13:** mySQL Tabla `invernaderos`

Fuente: Propia



				id	nombre_planta	rfid_asociado			
<input type="checkbox"/>		Editar		Copiar		Borrar	1	Orquídea	E11A9F1B
<input type="checkbox"/>		Editar		Copiar		Borrar	2	Geranio	F189B51B
<input type="checkbox"/>		Editar		Copiar		Borrar	3	Begonia	FA686B0
<input type="checkbox"/>		Editar		Copiar		Borrar	4	Petunia	D08B7C1D
<input type="checkbox"/>		Editar		Copiar		Borrar	5	Poinsettia	C09B831D
<input type="checkbox"/>		Editar		Copiar		Borrar	6	Hortensia	C0312A1D
<input type="checkbox"/>		Editar		Copiar		Borrar	7	Crisantemo	9271551D
<input type="checkbox"/>		Editar		Copiar		Borrar	8	Vinca	D09E9A1D

**Figura 4.14:** mySQL Tabla plantas  
Fuente: Propia

				invernadero_id	planta_id	cantidad			
<input type="checkbox"/>		Editar		Copiar		Borrar	1	1	2
<input type="checkbox"/>		Editar		Copiar		Borrar	1	2	9
<input type="checkbox"/>		Editar		Copiar		Borrar	1	3	5
<input type="checkbox"/>		Editar		Copiar		Borrar	1	4	2
<input type="checkbox"/>		Editar		Copiar		Borrar	1	5	5
<input type="checkbox"/>		Editar		Copiar		Borrar	1	6	1
<input type="checkbox"/>		Editar		Copiar		Borrar	1	7	3
<input type="checkbox"/>		Editar		Copiar		Borrar	1	8	12
<input type="checkbox"/>		Editar		Copiar		Borrar	2	1	1
<input type="checkbox"/>		Editar		Copiar		Borrar	2	2	1
<input type="checkbox"/>		Editar		Copiar		Borrar	2	3	1
<input type="checkbox"/>		Editar		Copiar		Borrar	2	4	6
<input type="checkbox"/>		Editar		Copiar		Borrar	2	5	1
<input type="checkbox"/>		Editar		Copiar		Borrar	2	8	1
<input type="checkbox"/>		Editar		Copiar		Borrar	3	1	1
<input type="checkbox"/>		Editar		Copiar		Borrar	4	3	2
<input type="checkbox"/>		Editar		Copiar		Borrar	4	7	1
<input type="checkbox"/>		Editar		Copiar		Borrar	4	8	3
<input type="checkbox"/>		Editar		Copiar		Borrar	5	1	1
<input type="checkbox"/>		Editar		Copiar		Borrar	5	2	3
<input type="checkbox"/>		Editar		Copiar		Borrar	5	4	1
<input type="checkbox"/>		Editar		Copiar		Borrar	5	6	3
<input type="checkbox"/>		Editar		Copiar		Borrar	5	8	2
<input type="checkbox"/>		Editar		Copiar		Borrar	6	1	1
<input type="checkbox"/>		Editar		Copiar		Borrar	6	2	1

**Figura 4.15:** mySQL Tabla invernadero\_planta  
Fuente: Propia

### 4.3.4. Visual Studio

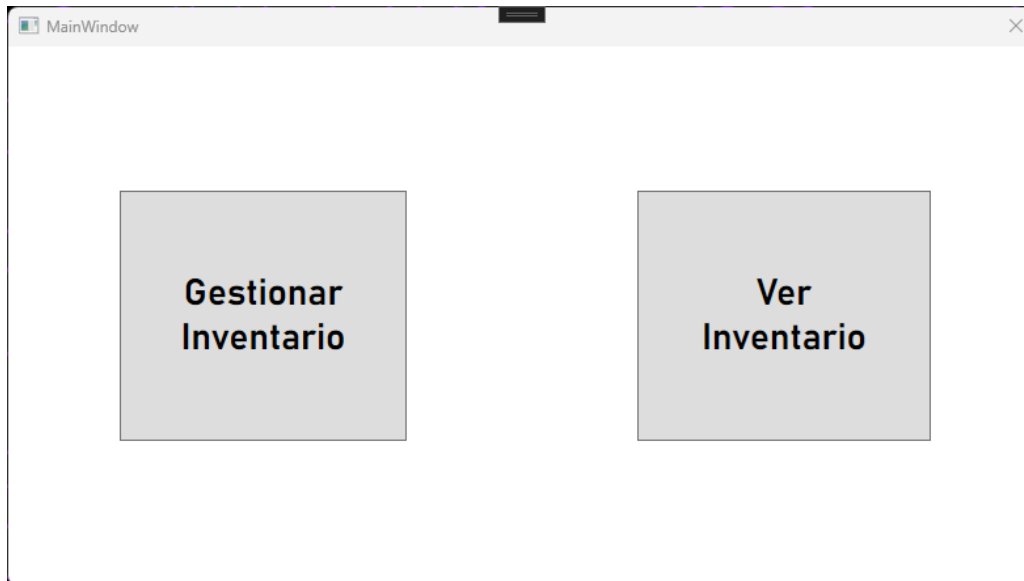
Por último se ha realizado una (GUI) en la aplicación de Visual Studio, esta aplicación cuenta con una simple interfaz de 2 botones, con un botón se puede gestionar el inventario para añadir/eliminar plantas del inventario y con el otro se puede ver el inventario de cada invernadero en tiempo real.

El programa está desarrollado utilizando principalmente dos lenguajes de programación: C# y XAML.

- C# es un lenguaje de programación orientado a objetos, lo que significa que está basado en el concepto de “objetos”, que pueden contener datos y código: datos en forma de propiedades y código, en forma de métodos. En este programa se utiliza para implementar la lógica de la aplicación, como la interacción con la base de datos o la manipulación de datos.
- XAML, que significa eXtensible Application Markup Language, es un lenguaje de marcado desarrollado por Microsoft para definir la interfaz de usuario en aplicaciones basadas en la plataforma .NET. Este es utilizado principalmente para definir la estructura y el aspecto de la interfaz de usuario de la aplicación. En este programa en concreto se utiliza para definir la estructura de las ventanas y controles de la aplicación, como botones, cuadros de texto o listas. Así como su apariencia y disposición.

El reto de esta aplicación es poder conectarse con éxito tanto con la base de datos como con Arduino, para ello son necesarias varias bibliotecas. Primero para conectarse al microcontrolador se ha instalado la biblioteca *System.IO.Ports.SerialPort*. Esta proporciona un conjunto de métodos y propiedades que permiten a la aplicación comunicarse con los puertos serie de un sistema. Para lo conexión con la base de datos se trató de utilizar la biblioteca *MySql.Data.MySqlClient* pero dado que la base de datos está en alojada en la web se descartó esta opción por problemas de compatibilidad. Por lo tanto, en lugar de conectarse directamente a la base de datos desde el código, lo que se ha realizado al final ha sido que el código se conecte a la URL del script de invernaderos y obtenga los datos en formato JSON. Para trabajar con JSON se ha utilizado la biblioteca *Newtonsoft.Json*, esta es utilizada para que la aplicación entienda la respuesta JSON que se recibe de la API web. Todo el código se encuentra disponible en el Anexo B.4.

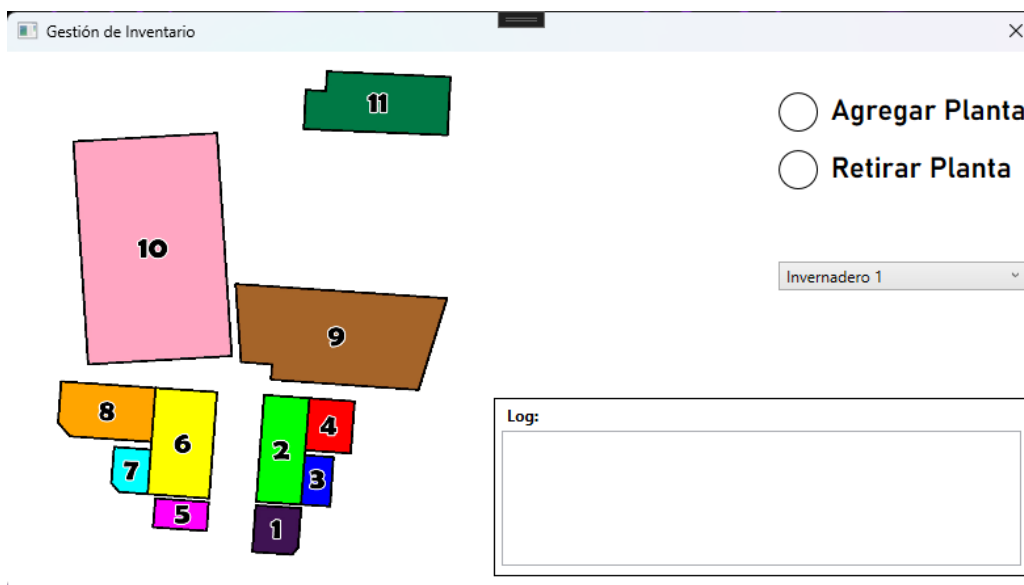
Al ejecutar la aplicación se encuentra la ventana principal *MainWindow*, en esta se encuentran 2 botones. El botón *Gestionar Inventario* conduce a otra ventana en la que poder modificar el inventario de los invernaderos, el botón *Ver Inventario* abre una nueva ventana en la que se podrá ver en tiempo real el inventario de cada invernadero (Figura: 4.16).



**Figura 4.16:** Main Window  
**Fuente:** Propia

El archivo *MainWindow.Xaml* únicamente tiene estos botones programados. Y el archivo *MainWindow.Xaml.cs* contiene la función *Button\_click* de cada botón para abrir una nueva ventana.

En gestionar inventario se ha colocado un plano por colores de los 11 invernaderos del vivero. La ventana cuenta con 2 *RadioButton* para agregar o retirar plantas, un *ComboBox* para seleccionar a que invernadero hay que hacer la modificación de inventario y por último un recuadro que realmente es un *TextBox* que servirá como salida del puerto COM de arduino, de esta forma nos indicará los últimos movimientos en el inventario (Figura: 4.17).



**Figura 4.17:** Ventana Gestionar Inventario Visual Studio  
**Fuente:** Propia

En el archivo *gestionar.xaml.cs* primero se importan varias bibliotecas necesarias para

el código. Se incluyen las bibliotecas básicas de .NET y WPF para la funcionalidad de la interfaz de usuario y eventos, todos los archivos .cs del programa incluyen las mismas librerías generalmente, en este caso se destaca la librería *System.IO.Ports* para la comunicación con el puerto serial.

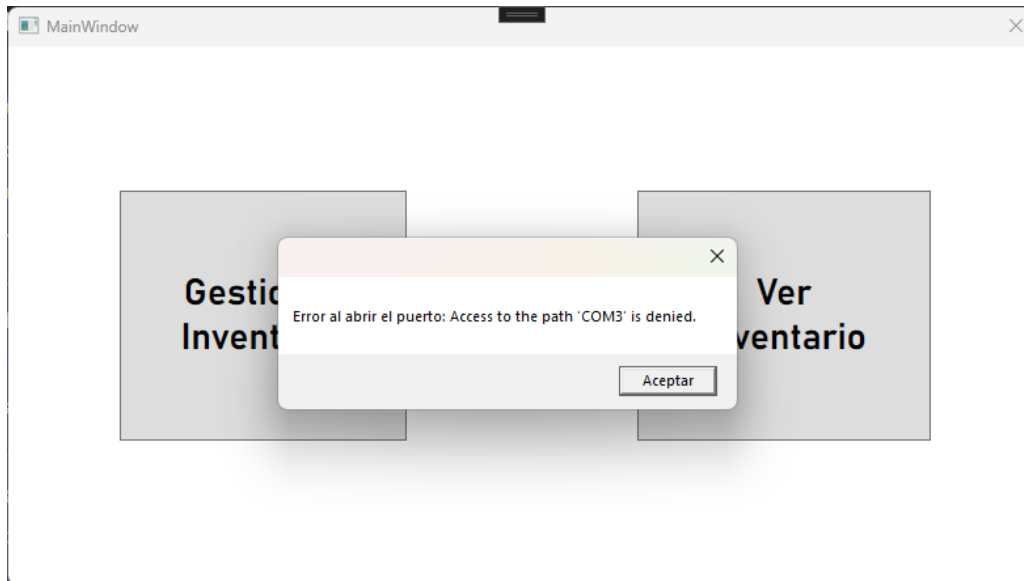
Después dentro de la clase *gestionar*, se declaran y definen algunas variables clave. *arduinoPort* representa la conexión serial al Arduino. *portName* y *baudRate* son constantes que determinan la configuración del puerto serial.

Después dentro de la clase *gestionar*, se declaran y definen algunas variables clave. *arduinoPort* representa la conexión serial al Arduino. *portName* y *baudRate* son constantes que determinan la configuración del puerto serial, especificando el nombre del puerto y la tasa de transmisión en baudios.

El constructor de la clase *gestionar* realiza dos funciones principales. Primero, inicializa los componentes de la interfaz de usuario. Luego, intenta establecer una conexión con el Arduino. Dentro de *InitializeArduinoConnection()*, se configura la conexión serial utilizando los parámetros previamente definidos y se intenta abrir la conexión. Si algo sale mal, por ejemplo, si el puerto no está disponible o hay un conflicto, se atrapa la excepción y se muestra un mensaje al usuario (Código: 4.5) (Figura: 4.18).

```
1 public gestionar()
2 {
3     InitializeComponent();
4     InitializeArduinoConnection();
5 }
6
7 private void InitializeArduinoConnection()
8 {
9     try
10    {
11        arduinoPort = new SerialPort(portName, baudRate);
12        arduinoPort.DataReceived += DataReceivedHandler;
13        arduinoPort.Open();
14    }
15    catch (Exception ex)
16    {
17        MessageBox.Show("Error al abrir el puerto: " + ex.Message);
18    }
19 }
```

Código 4.5: Visual Studio Inicialización de la Conexión Arduino



**Figura 4.18:** Mensaje de error puerto COM Visual Studio  
**Fuente:** Propia

La función *DataReceivedHandler* es crucial para la interactividad en tiempo real de la aplicación. Cuando el Arduino envía datos a través del puerto serial, este se activa. Utiliza *ReadLine()* para leer una línea de datos del puerto serial. Se utiliza *Dispatcher.Invoke* para asegurar que los datos recibidos se muestren en el *logTextBox* de la interfaz gráfica, proporcionando una visualización en tiempo real de la comunicación con el Arduino (Código: 4.6).

```

1 private void DataReceivedHandler(object sender, SerialDataReceivedEventArgs e
2 )
3 {
4     var receivedData = arduinoPort.ReadLine();
5     Dispatcher.Invoke(() =>
6     {
7         logTextBox.AppendText(receivedData + Environment.NewLine);
8         logTextBox.ScrollToEnd();
9     });
10 }

```

**Código 4.6:** Visual Studio Manejo de datos recibidos

Después se encuentran estos dos métodos que gestionan las acciones del usuario en la interfaz de usuario. *RadioButton\_Checked* se activa cuando el usuario selecciona un "radio button". Dependiendo de la opción elegida, se envía una orden específica a Arduino. Por otro lado, *ComboBox\_SelectionChanged* maneja el cambio de selección en el *comboBox*. Cuando se selecciona un invernadero específico, se extrae el número de ese invernadero y se envía un comando a Arduino, permitiéndole saber en qué invernadero realizar una acción específica. (Código: 4.7)

```

1 private void RadioButton_Checked(object sender, RoutedEventArgs e)
2 {
3     var radioButton = sender as RadioButton;
4     if (radioButton != null && arduinoPort != null && arduinoPort.IsOpen)
5     {
6         if (radioButton.Content.ToString() == "Agregar Planta")
7         {

```

```

8         arduinoPort.WriteLine("AGREGAR");
9     }
10    else if (radioButton.Content.ToString() == "Retirar Planta")
11    {
12        arduinoPort.WriteLine("RETIRAR");
13    }
14    }
15 }
16
17 private void ComboBox_SelectionChanged(object sender,
18     SelectionChangedEventArgs e)
19 {
20     if (arduinoPort != null && arduinoPort.IsOpen)
21     {
22         var comboBox = sender as ComboBox;
23         var selectedItem = comboBox.SelectedItem as ComboBoxItem;
24
25         if (selectedItem != null)
26         {
27             string invernadero = selectedItem.Content.ToString();
28             string number = invernadero.Split(' ')[1]; // Extraer número del
29             invernadero
30             arduinoPort.WriteLine("INVERNADERO:" + number);
31         }
32     }
33 }

```

Código 4.7: Visual Studio Acciones Basadas en Interacciones del Usuario

Por último, este segmento se asegura de que la conexión serial se cierre de manera adecuada cuando la ventana de la aplicación se cierra. Si se deja abierto, podría causar conflictos o bloquear el puerto para otras aplicaciones (como arduino). Por ello, al detectar que la ventana se está cerrando, verifica si el puerto está abierto y, en ese caso, procede a cerrarlo (Código: 4.8).

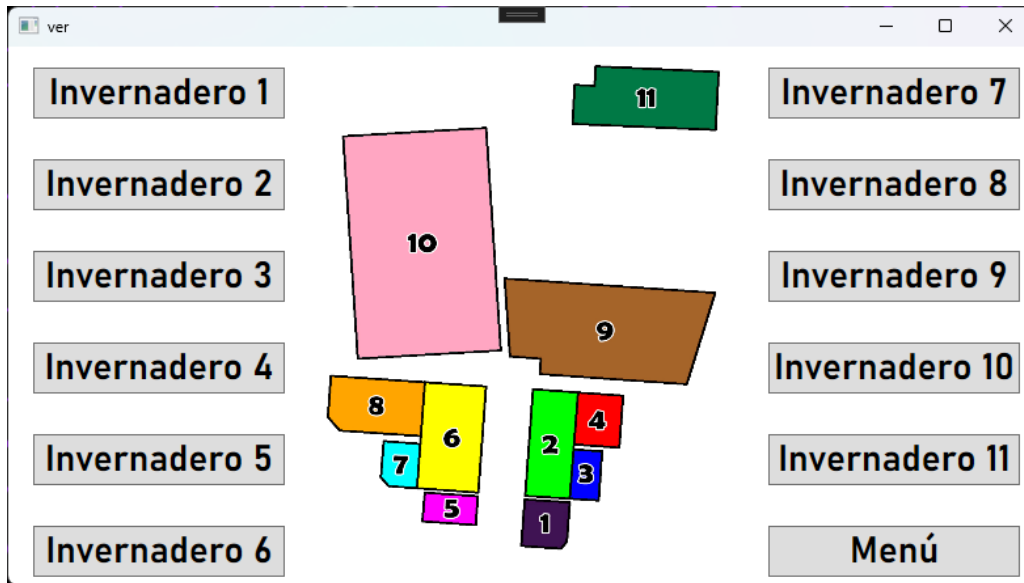
```

1 protected override void OnClosed(EventArgs e)
2 {
3     base.OnClosed(e);
4     if (arduinoPort != null && arduinoPort.IsOpen)
5     {
6         arduinoPort.Close();
7     }
8 }

```

Código 4.8: Visual Studio Cierre de Conexión al Cerrar la Ventana

Por otro lado se encuentra la ventana de *Ver Inventario*, esta cuenta con 12 botones. 11 de estos son para acceder a los diferentes invernaderos y ver en tiempo real su inventarios, y el botón *Menú* sirve para volver al *Main Window*. En medio de la ventana se ha colocado el plano de los invernaderos.



**Figura 4.19:** Ventana Ver Inventario Visual Studio  
**Fuente:** Propia

El archivo *ver.xaml.cs* no tiene mucha complejidad, lo único que hay es una función para cada botón donde se abre una nueva ventana con la información correspondiente de ese invernadero. El archivo *.xaml* de estas ventanas esta formado por un *TextBlock* indicando el nombre del inventario, otro indicando el número total de plantas y por ultimo un *ListView* que tiene una columna para "Tipo de Planta", y otra columna para "Cantidad"(Figura: 4.20).



**Figura 4.20:** Ventana Invernadero 1 Visual Studio  
**Fuente:** Propia

Todas las ventanas de los invernaderos tienen el mismo código, lo único que cambia es la url a la que se conectaran. Para explicarlo se va a desarrollar *Invernadero1.xaml.cs*. Primero se definen las librerías empleadas, son las mismas de siempre pero destaca la incorporación de las siguientes:

- **System.Net.Http**: Se utiliza para hacer solicitudes HTTP y comunicarse con servicios web.
- **Newtonsoft.Json.Linq**: Es esencial para trabajar con estructuras de datos JSON, facilitando su análisis y manipulación.
- **System.Windows.Threading**: Esta es una biblioteca específica para el manejo de temporizadores que trabajan en el hilo principal del UI.

En la clase *Invernadero1* se establecen 2 variables principales: *Url*, que es una constante que almacena la dirección web del servidor desde donde se obtendrán los datos del invernadero, esta url cambiará en función del invernadero que estemos trabajando, si por ejemplo estamos en la ventana del invernadero2 esta url será igual pero acabará en 2 en vez de en 1, uno de los motivos para usar el método GET en vez del método POST fue que con el GET es más fácil implementar cosas parecidas cambiando detalles simples. Por otro lado, *updateTimer*, un temporizador que controla la frecuencia con la que se actualizan los datos en la interfaz (Código: 4.9).

```

1 namespace AppVivero
2 {
3     public partial class Invernadero1 : Window
4     {
5         private const string Url = "http://pruebaviverotfg.000webhostapp.com/
6         invernaderos.php?id_invernadero=1";
7         private DispatcherTimer updateTimer;

```

Código 4.9: Invernadero.xaml.cs Declaraciones de clase y variables

El método *Invernadero1()* es el constructor de la clase. Aquí se inicializan los componentes gráficos de la ventana con el método *InitializeComponent()*, luego se llama al método *CargarDatos()* para obtener los datos iniciales del invernadero. Posteriormente, se configura el temporizador *updateTimer* para que ejecute el método *UpdateTimer\_Tick* cada tres segundos (Código: 4.10).

```

1 public Invernadero1()
2 {
3     InitializeComponent();
4     CargarDatos();
5
6     updateTimer = new DispatcherTimer();
7     updateTimer.Interval = TimeSpan.FromSeconds(3);
8     updateTimer.Tick += UpdateTimer_Tick;
9     updateTimer.Start();
10 }

```

Código 4.10: Invernadero.xaml.cs Constructor de la Clase

Este método es un controlador de eventos que se activa cada vez que el temporizador llega a su intervalo definido, en este caso, cada tres segundos. Su función es simplemente llamar al método *CargarDatos()* para refrescar la información del invernadero en la interfaz (Código: 4.11).

```

1 private void UpdateTimer_Tick(object sender, EventArgs e)
2 {
3     CargarDatos();
4 }

```

Código 4.11: Invernadero.xaml.cs Gestión del Temporizador



Por último se encuentra el método *CargarDatos()*, este gestiona la conexión al servidor mediante la librería *HttpClient* y obtiene los datos en formato JSON. Después, analiza este JSON para extraer el número total de plantas y una lista de las plantas presentes en el invernadero. Con esta información, llena una lista en la interfaz de usuario "lstPlantas" con el nombre y la cantidad de cada planta. También muestra el total de plantas en un campo de texto "txtTotalPlantas". Si ocurre algún problema al conectar con el servidor o al procesar los datos, se muestra un mensaje de error mediante una ventana emergente (Código: 4.12).

```
1 private async void CargarDatos()
2 {
3     try
4     {
5         using (HttpClient client = new HttpClient())
6         {
7             var response = await client.GetStringAsync(Url);
8             JObject jsonObject = JObject.Parse(response);
9
10            int totalPlantas = jsonObject["totalPlantas"].Value<int>();
11            JSONArray plantasArray = (JSONArray)jsonObject["plantas"];
12
13            lstPlantas.Items.Clear();
14
15            foreach (var item in plantasArray)
16            {
17                string tipoPlanta = item["nombre_planta"].ToString();
18                int cantidad = int.Parse(item["cantidad"].ToString());
19                lstPlantas.Items.Add(new { TipoPlanta = tipoPlanta, Cantidad
20                = cantidad });
21            }
22
23            txtTotalPlantas.Text = totalPlantas.ToString();
24        }
25    } catch (Exception ex)
26    {
27        MessageBox.Show($"Ocurrió un error al obtener los datos: {ex.Message}
28        ", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
29    }
```

Código 4.12: Invernadero.xaml.cs Gestión del Temporizador

# Capítulo 5

## Resultados

Este trabajo tiene el objetivo de mejorar la eficiencia en la gestión de inventario de un vivero, para ello se ha realizado un proyecto a escala pequeña del objetivo inicial. Este objetivo no se ha realizado debido al elevado coste que conlleva. En este capítulo se van a mostrar los resultados obtenidos del proyecto realizado y al final de este se muestra la idea del proyecto a gran escala.

### 5.1. Implementación

En este punto se describe la implementación del sistema realizado en un vivero. Para ello se va a detallar paso a paso el proceso de una planta desde que se siembra hasta que llega al invernadero utilizando el sistema realizado.

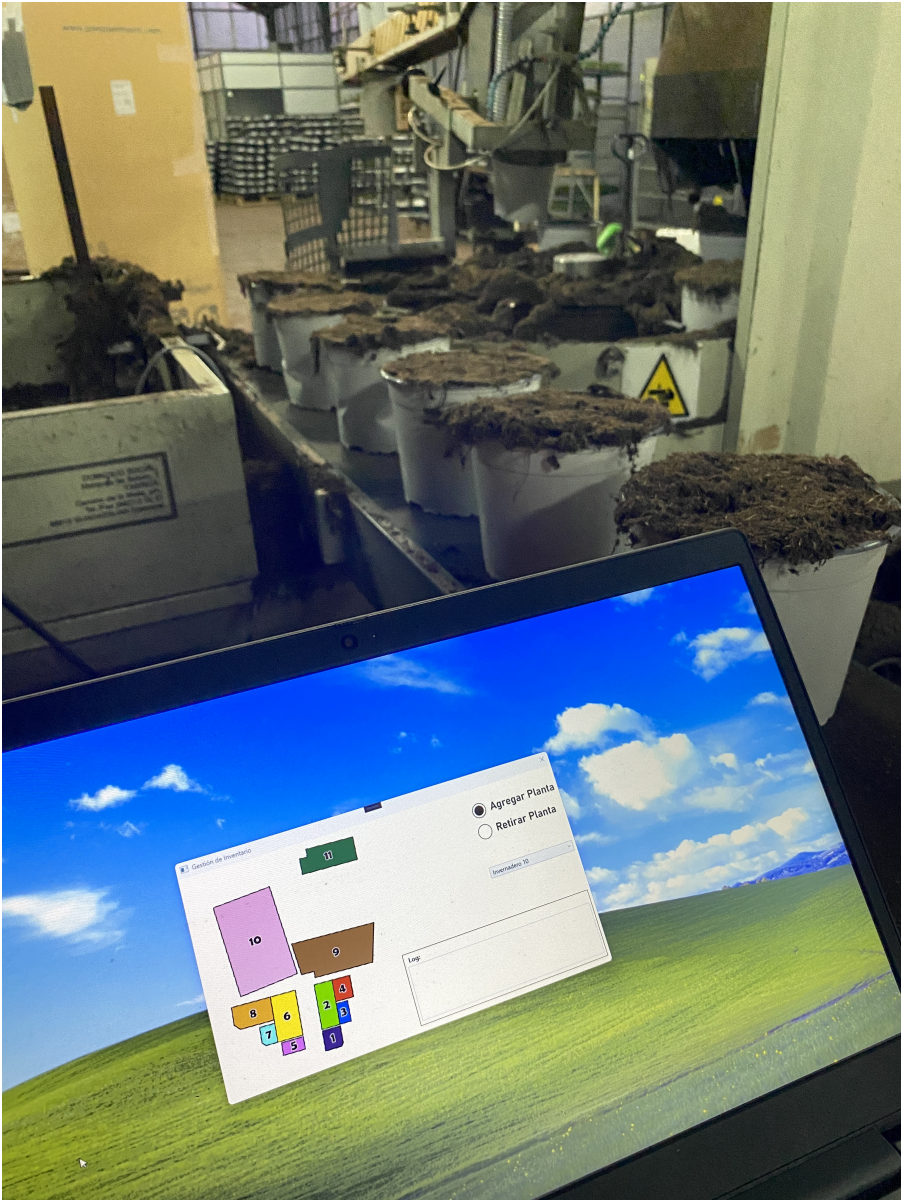
1. El primer paso comienza en la máquina de tierra, en esta hay una persona trabajando continuamente con la máquina para llenar las macetas de tierra y colocarlas en bandejas. La máquina va sacando continuamente macetas llenas de tierra por una cinta transportadora, en el transcurso de esta cinta se coloca el lector RFID en modo agregar y se selecciona a que invernadero van a ir (Figura: 5.1).

Es importante colocar las macetas en el surtidor de la máquina con la etiqueta RFID de la planta que posteriormente se va a sembrar, la máquina siempre saca las macetas en la misma posición, por eso hay que colocarlas en el surtidor de forma que al salir a la cinta estén apuntando hacia el lector RFID (Figura: 5.2).

Por lo tanto, automáticamente en el transcurso por la cinta las plantas se van añadiendo una a una a la base de datos, cuando estas están llegando al final de la cinta son colocadas en bandejas por un empleado. Las bandejas de tierra son apiladas en un palé de madera. Una vez el palé está lleno, se coloca en un lado del invernadero donde se siembran las plantas.

2. El siguiente paso es sembrar las plantas en las macetas. Para esto un equipo de trabajadores se coloca en mesas y utilizan el plantel correspondiente y las macetas que han sido llenadas anteriormente (Figura: 5.3).

Este equipo se encarga de coger las bandejas con las macetas que ya han sido escaneadas y añadidas a la base de datos y sembrar las plantas en ellas. Después cuando la bandeja está llena la coloca en un carro. Se hace esto hasta que el carro



**Figura 5.1:** Máquina trabajando con el modo agregar  
**Fuente:** Propia



**Figura 5.2:** Lector RFID en la cinta transportadora  
**Fuente:** Propia



**Figura 5.3:** Plantel y macetas preparadas para sembrar  
**Fuente:** Propia

está lleno, una vez lleno este está listo para ser desplazado al invernadero que le corresponde. (Figura: 5.4).

3. El siguiente paso es la colocación de las plantas en los invernaderos, para ello otro grupo de empleados se encarga de recoger los carros que están llenos de bandejas con las macetas dentro y los transportan hasta el invernadero correspondiente (Figura: 5.5).

Una vez aquí se encargan de coger las bandejas para sacar las plantas y colocarlas en el invernadero, para su posterior cuidado (Figura: 5.6).

4. Una vez las plantas están colocadas en los invernaderos, se puede acceder a la aplicación desde cualquier sitio y ver así la cantidad de plantas que hay en ese momento en cada invernadero (Figura: 5.7).

## 5.2. Ampliación del trabajo

La idea inicial de este proyecto tenía el objetivo de facilitar al máximo posible la gestión de inventario de un vivero gracias a la tecnología RFID, esta tecnología como ya se ha explicado a lo largo del proyecto tiene innumerables aplicaciones y formas de uso, por lo tanto las posibilidades son infinitas.

Gracias a esta tecnología tienes un control muy preciso de todas las plantas del vivero, por lo tanto puedes usar estos datos no solo para gestionar el inventario del vivero si no para mucho más. La implementación de la tecnología en el vivero abre muchas puertas



**Figura 5.4:** Proceso de plantación de esquejes  
**Fuente:** Propia

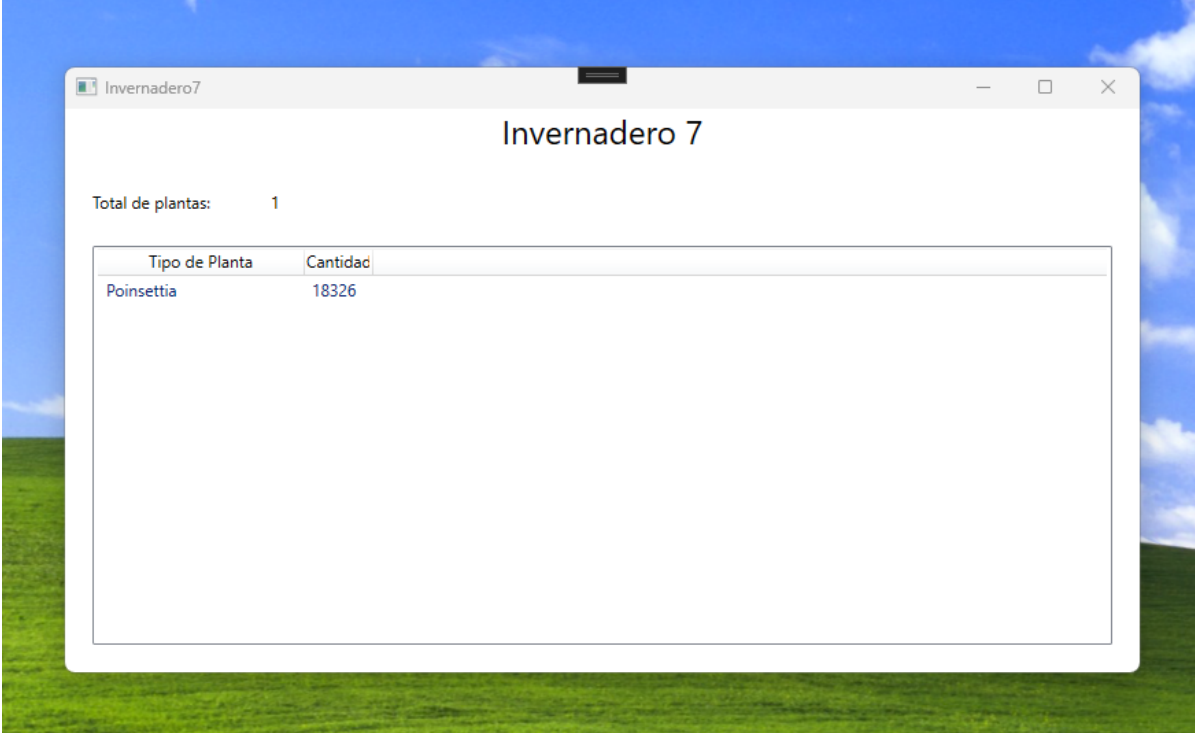


**Figura 5.5:** Transporte de los carros hasta el invernadero  
**Fuente:** Propia



**Figura 5.6:** Colocación de las plantas en el invernadero  
**Fuente:** Propia





Invernadero 7

Total de plantas: 1

Tipo de Planta	Cantidad
Poinsettia	18326

**Figura 5.7:** Visualización de las plantas en un invernadero

**Fuente:** Propia

para mejoras y aplicaciones innovadoras que no han sido implementadas en el proyecto inicial.

La principal diferencia de lo que se ha desarrollado con lo que estaría bien desarrollar es la implementación de etiquetas en todas las plantas y automatizar el proceso de lectura de las etiquetas con lectores no fijos para no tener que pasar las plantas 1 a 1. Ambas no se han desarrollado en el proyecto principal debido a su elevado coste, si se hubiesen llevado a cabo se podrían haber desarrollado muchas más implementaciones.

Algunas ideas y funcionalidades que esto ofrece serían:

- **Monitoreo de Condiciones Ambientales:** El vivero ya cuenta con tecnología para monitorear y ver valores clave en los diferentes invernaderos, pero los invernaderos son muy grandes por lo tanto es muy común que se generen microclimas dentro de estos. La clave aquí sería usar etiquetas RFID que tengan sensores integrados. Estos sensores pueden medir variables como temperatura, humedad, luminosidad, entre otros. Estas etiquetas se colocarían en diferentes puntos estratégicos del invernadero, pudiendo hacer así un mapa de calor de la temperatura o humedad por ejemplo de los diferentes invernaderos y ver como evoluciona esto en tiempo real.
- **Información al Cliente:** Al comprar una planta, el cliente podría escanear la etiqueta RFID y acceder a toda la información sobre su cuidado, historia y condiciones ideales.
- **Historial de Cuidados y Mantenimiento:** Cada vez que una planta recibe agua, fertilizantes o cualquier otro cuidado, el responsable puede registrar la acción con un lector RFID. Esto permitiría tener un historial detallado de todos los cuidados que ha recibido cada planta.
- **Alertas y Notificaciones:** En base a los datos recogidos, el sistema podría generar

alertas. Por ejemplo, si una planta necesita ser regada o si las condiciones ambientales no son las ideales. Al fin y al cabo los invernaderos son muy grandes y dentro de estos puedes estar aplicando un mismo tratamiento para todas las plantas cuando no es lo adecuado, entonces con la tecnología implementada se podría solucionar esto.

- **Ampliación hacia IoT (Internet de las Cosas):** Con la inclusión de más dispositivos inteligentes, el vivero podría convertirse en un espacio totalmente conectado, donde todo está intercomunicado y se pueden automatizar aún más procesos.

Para empezar el desarrollo de la idea habría que comprar una impresora RFID, la idea es etiquetar todas las plantas del vivero por lo tanto es insostenible pedir etiquetas a empresas de terceros para usarlas, haciendo un desembolso inicial grande se acabaría ahorrando y también funcionaría todo más rápido al poder tener las etiquetas cuando uno quiera. La idea es que estas etiquetas sean implementadas antes del proceso de producción de las macetas, cuando estas pasan por la máquina para llenarse de tierra.

Después se mejoraría el proceso de identificación de las etiquetas, para hacerlo de forma autónoma se instalarían en las puertas de los invernaderos lectores automáticos RF, esto actuaría parecido a los escáners anti robo que tienen las tiendas en la actualidad, su funcionalidad en vez de pitar cuando pasa un elemento sin pagar sería registrar en la base de datos todas las etiquetas que entren a un invernadero, de igual manera cuando salen se borrarían de la base de datos ya que detectarían que están saliendo. Para ello se ha diseñado una imagen usando el escáner *Zebra Transition RFID Portal* de como quedaría aplicado a un invernadero del vivero. (Figura: 5.8).



**Figura 5.8:** Escáner RFID identificando múltiples plantas a la vez

**Fuente:** Propia

### 5.3. Conclusiones y valoración personal

Durante la creación de este proyecto se ha conseguido de forma satisfactoria la configuración y puesta en marcha de un sistema basado en RFID dirigido por el microcontrolador ESP32 y gracias al lector RC522. Para la digitalización de los datos se ha logrado interconectar varios elementos de software como son: Arduino, Servidor PHP, base de datos MySQL y la IDE de Visual Studio. La unión de todas estas herramientas ha conseguido que el resultado final sea el deseado.

En el proyecto realizado se ha hecho un trabajo completo de programación, totalmente orientado al producto final, considerando todas sus necesidades y especificaciones, se ha llevado a cabo un exhaustivo análisis de requisitos y posibilidades. Durante este proceso, se identificaron y determinaron cuales eran los objetivos alcanzables del proyecto. Cierto es que me hubiese gustado aplicar la idea a futuro explicada anteriormente pero como ya se ha comentado es inviable económicamente para un proyecto de final de grado de un estudiante.

En lo personal el desarrollo de este proyecto ha sido un gran reto ya que nunca había realizado un trabajo en el cual se tenían que interconectar tantas utilidades para llegar al resultado final. Han sido muchas horas de prueba y error para ir mejorando el trabajo poco a poco hasta conseguir el resultado final.

Ha sido fascinante poder hacer un trabajo aplicado para el vivero, ya que llevo toda la vida creciendo en el entorno de este y conociendo los problemas que puede acarrear, por lo tanto poder hacer un proyecto que mejore la forma de trabajar en este es una gran ilusión.

El resultado final mostrado no ha sido una tarea fácil en absoluto. Detrás de él, se encuentran cientos de horas dedicadas a superar desafíos y resolver problemas a través de un proceso constante de estudio e investigación. Este esfuerzo no solo ha llevado al éxito del proyecto, sino que también me ha permitido adquirir nuevos conocimientos que de otro modo no habrían sido posibles.

# Capítulo 6

## Estudio económico

En este apartado se calculan y detallan los costes asociados al proyecto de fin de grado sobre el desarrollo e implementación de un sistema de gestión de inventario de plantas en un vivero mediante el uso de tecnología RFID y una base de datos.

Se han separado en 2 los costes, por un lado está el costo del proyecto realizado y por otro el presupuesto de la idea del proyecto a mejorar, que no se ha realizado por su alto presupuesto.

### 6.1. Revisión de costes

#### 6.1.1. Costes del proyecto realizado

Para calcular los costes del proyecto realizado se ha tenido en cuenta el microcontrolador ESP-32 Kit de desarrollo C V4, RFID KIT RC522 con módulo RFID-RC522, tarjeta S50, llavero S50, conectores de regleta de pines. (13.56MHz), tarjetas RFID 13,56MHz Tarjeta MF S50 (13,56 MHz), cable micro-usb de 1.5 metros y un pack de protoboards + cables para arduino. (Tabla: 6.1).

En cuanto a los gastos de envío no se han tenido en cuenta ya que dependen de cada página web y de la empresa que gestiona los envíos. El 21 % del IVA ya está aplicado en el importe final de los componentes. Por otro lado, del software no se ha tenido en cuenta nada ya que el IDE de Arduino es código abierto, el servidor de hosting es gratuito y el IDE de Visual Studio es completo, extensible y gratuito.

Producto	Fuente	Cantidad[ud]	Coste uni- dad[€/ud]	Importe[€]
ESP32 V4 DevKit	Az-Delivery	1	13	13
RFID KIT RC522	Az-Delivery	1	5.5	5.5
Cable Micro USB	PC-Componentes	1	7	7
Kit Protoboard + Cables	Amazon	1	15	15
Set tarjetas RFID 10uds	Az-Delivery	1	7	7
<b>Total</b>				<b>47.5</b>

**Tabla 6.1:** Costes del proyecto  
**Fuente:** Propia

### 6.1.2. Mano de obra

En la tabla 6.2 se muestra el gasto de coste humano para la realización del proyecto. El trabajo ha sido realizado solamente por el autor del proyecto, se estima que se ha trabajado entorno a las 360 horas. Como precio se ha puesto el sueldo medio de un ingeniero electrónico recién graduado que equivale a 20€/h. Este dato se ha extraído de la web Jobted: <https://www.jobted.es/salario/ingeniero-electr%C3%B3nico>.

Descripción	Precio[€/hora]	Cantidad [horas]	Total [€]
Documentación e investigación sobre la tecnología	20	50	1000
Diseño del sistema	20	45	900
Aprendizaje sobre el software	20	50	1000
Implementación del sistema	20	100	2000
Pruebas de funcionamiento	20	15	300
Redacción del documento	20	100	2000
Medios auxiliares sobre costes directos	10%	7200€	720€
		<b>Total</b>	<b>7920</b>

**Tabla 6.2:** Coste humano  
**Fuente:** Propia

### 6.1.3. Costes totales

En la Tabla 6.3 se muestra el coste total del proyecto, que incluye tanto los costes de materiales como el coste humano.

Tipo de coste	Precio[€]
Coste de los materiales	47.5
Coste humano	7920
<b>Total</b>	<b>7967.5</b>

**Tabla 6.3:** Coste Total  
**Fuente:** Propia

### 6.1.4. Costes de la idea de proyecto futuro

Para los costes de este proyecto se ha tenido en cuenta la idea desarrollada en el apartado 5.2. Primero se tiene en cuenta la impresora para hacer las etiquetas de las diferentes plantas, se seleccionan también 20 rollos de 5000 etiquetas, aunque esto seria como desembolso inicial ya que el vivero cuenta con muchas mas plantas y además estas son vendidas y hay que generar mas etiquetas para las nuevas. Además se pagaría un servicio de hosting de pago para garantizar el funcionamiento estable en todo momento de la aplicación (Para el precio se ha establecido el servicio de 1 año). Por último, los sensores automáticos para colocar en las puertas de los invernaderos (Tabla: 6.4).

---

Producto	Fuente	Cantidad[ud]	Coste uni- dad[€/ud]	Importe[€]
Impresora RFID ZT400	Zebra technologies	1	3000	3000
Rollo de etiquetas RFID x5000	theRFIDstore.eu	20	575	11500
Zebra Transition RFID Portal	barcodesinc	11	9200	101200
Hosting web 1 año	siteground	1	102	102
<b>Total</b>				115802

---

**Tabla 6.4:** Costes de la idea de proyecto**Fuente:** Propia



## Parte II

### Pliego de condiciones





# Capítulo 7

## Pliego de condiciones

### 7.1. Objeto

El siguiente pliego de condiciones presenta los requisitos necesarios al contratista para el desarrollo e implementación de un sistema de gestión de inventario de plantas en un vivero mediante el uso de tecnología RFID y una base de datos.

### 7.2. Condiciones técnicas

#### 1. Microcontrolador ESP32 V4 Dev Kit

- CPU con 2 núcleos de Tensilica LX6 240MHz
- RAM de 520 KB estática y 8 MB de memoria SPI PSRAM
- Puerto micro USB para alimentación y comunicación con la computadora.
- Conectividad inalámbrica por Wi-fi y bluetooth
- pines GPIO
- El rango de temperatura de funcionamiento típico es de  $-40^{\circ}\text{C}$  a  $85^{\circ}\text{C}$
- Tamaño 54mm x 28mm.

#### 2. Lector RFID RC-522

- Opera en la banda de frecuencia de 13.56 MHz
- Se comunica a través de una interfaz SPI
- Requiere una fuente de alimentación de 3.3V
- Funciona en un rango de temperatura típico de  $-20^{\circ}\text{C}$  a  $85^{\circ}\text{C}$ .
- Tamaño 40 mm x 60 mm.

### 7.3. Control de calidad

Este apartado tiene la función de establecer los criterios y estándares que se utilizarán para asegurar que el proyecto cumple con los requisitos y las expectativas de calidad

establecidos. Para ello se han seguido unas pautas para comprobar que los componentes funcionan correctamente:

### 1. Microcontrolador ESP32 V4 Dev Kit

- **Verificación Funcional:** Se ha comprobado que todas las funcionalidades del ESP32, como Wi-Fi y Bluetooth, funcionen sin errores.
- **Pruebas de Estabilidad:** Se han realizado pruebas a largo plazo para garantizar la estabilidad del microcontrolador.
- **Seguridad:** Se ha evaluado la seguridad del sistema para asegurarse de que no haya vulnerabilidades en la conexión Wi-Fi o Bluetooth.

### 2. Lector RFID RC-522

- **Integración:** Se ha verificado que el lector RFID se integra adecuadamente con el microcontrolador y que la comunicación funciona sin problemas.
- **Precisión de Lectura:** Se ha comprobado que el lector RFID lee tarjetas con precisión y sin errores.
- **Distancia de Lectura:** Se ha asegurado de que la distancia de lectura cumple con los parámetros especificados.

### 3. IDE de Visual Studio y Arduino

- **Compatibilidad:** Confirmar que las IDEs utilizadas sean compatibles con los requisitos del proyecto y funcionen sin problemas.
- **Pruebas de Código:** Realizar pruebas y revisión de código para identificar errores y garantizar la calidad del software.

### 4. Base de Datos

- **Integridad de Datos:** Verificar que la base de datos mantenga la integridad de los datos y evite pérdidas de información.
- **Rendimiento:** Evaluar el rendimiento de la base de datos en velocidad de acceso y consultas.

### 5. Pruebas y Verificación General

- **Compatibilidad:** Se ha comprobado que todos los sistemas se interconectan y entienden entre sí para asegurar el objetivo

Todos los elementos del sistema siguen la normativa de seguridad electrónica que sigue directivas como la Directiva de Baja Tensión y la Directiva de Compatibilidad Electromagnética (CEM). También las normas de calidad como la norma ISO 9001, para garantizar la calidad y la consistencia del producto.

## 7.4. Condiciones de ejecución

1. El circuito electrónico se debe de colocar en una superficie plana para garantizar la correcta conexión de todos los cables y evitar así que se dificulte el funcionamiento
2. Al ejecutar la aplicación es necesario que el circuito esté conectado y el microcontrolador conectado a la red, para evitar problemas al consultar el inventario o al leer las tarjetas RFID.
3. Es necesario que el puerto COM no esté siendo usado por otra aplicación ya que impediría la conexión con la aplicación realizada.
4. Se debe de colocar la placa en un lugar aislado de factores externos como humedad, líquidos o químicos para asegurar el correcto funcionamiento del sistema.

## 7.5. Manual de usuario

Para ser instalado, el sistema completo será entregado cuando haya superado las pruebas mencionadas en la memoria. Este sistema incluye el circuito electrónico con el lector RFID, el software realizado para la gestión de inventario y el correspondiente manual para poder utilizarlo.



# Bibliografía

- [1] Xiaowei Zhu, Samar K. Mukhopadhyay, and Hisashi Kurata. A review of rfid technology and its managerial applications in different industries. *Journal of Engineering and Technology Management*, 29(1):152–167, 2012. ISSN 0923-4748. Creating competitive edge in operations and service management through technology and innovation.
- [2] Electronics Hub. Rfid technology and its applications. Accedido en 03-08-2023 a ,<https://www.electronicshub.org/rfid-technology-and-its-applications/>, 2015. [Online].
- [3] Jerry Landt. Shrouds of time the history of rfid. Accedido en 10-08-2023 a ,[https://web.archive.org/web/20090327005501/http://www.transcore.com/pdf/AIM%20shrouds\\_of\\_time.pdf](https://web.archive.org/web/20090327005501/http://www.transcore.com/pdf/AIM%20shrouds_of_time.pdf). [Online].
- [4] Klaus Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. Wiley Publishing, 2nd edition, 2003. ISBN 0470844027.
- [5] Jose María García Barceló. Análisis y prueba de un sistema en tecnología de identificación por radiofrecuencia. Accedido en 09-08-2023 a ,<https://oa.upm.es/44163/>, Junio 2016. Trabajo Fin de Grado. No publicado. [Online].
- [6] Roberto de Jesús Urbina Ruiz. Tutorial sobre circuitos rfid. Accedido en 11-09-2023 a ,[http://catarina.udlap.mx/u\\_dl\\_a/tales/documentos/lep/urbina\\_r\\_rd/](http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/urbina_r_rd/), Mayo 2011. Trabajo Fin de Grado. [Online].
- [7] Kamran Ahsan, Shah Hanifa, and Paul Kingston. RFID applications: An introductory and exploratory study. *IJCSI International Journal of Computer Science Issues*, 7(3), Enero 2010. ISSN 1694-0784.
- [8] Mohamed El Beqqal and Mostafa Azizi. Review on security issues in RFID systems. *Advances in Science, Technology and Engineering Systems Journal*, 2(6):194–202, 2017.
- [9] Giorgia Casella, Barbara Bigliardi, and Eleonora Bottani. The evolution of RFID technology in the logistics field: a review. *Procedia Computer Science*, 200:1582–1592, 2022. ISSN 1877-0509. 3rd International Conference on Industry 4.0 and Smart Manufacturing.
- [10] Natasha Dean Harrison and Ella L. Kelly. Affordable RFID loggers for monitoring animal movement, activity, and behaviour. *PLOS ONE*, 17(10):1–10, 10 2022.

- 
- [11] Aimee Kalnoskas. analogictips: How do RFID tags and reader antennas work? Accedido en 03-08-2023 a ,<https://www.analogictips.com/rfid-tag-and-reader-antennas/>. [Online].
- [12] Zebra Technologies. Fx9600 rfid fixed reader. (2023). Accedido en 12-07-2023 a ,<https://www.zebra.com/es/es/products/rfid/rfid-readers/fx9600.html>. [Online].
- [13] Aeroexpo. Lector rfid portátil mc3190-z (2023). Accedido en 14-07-2023 a ,<https://www.aeroexpo.online/es/prod/sensormatic/product-175479-25505.html>. [Online].
- [14] Zebra Technologies. Printer zt410. (2023). Accedido en 14-07-2023 a ,<https://www.zebra.com/es/es/support-downloads/printers/industrial/zt410.html>. [Online].
- [15] fqingenieria. Guía para escoger la antena rfid uhf más adecuada. Accedido en 26-07-2023 a ,<https://www.fqingenieria.com/es/conocimiento/guia-para-escoger-la-antena-rfid-uhf-mas-adecuada-114>. [Online].
- [16] Learnchannel-TV. Rfid - frequencies and distance range (2007). Accedido en 02-08-2023 a ,<https://learnchannel-tv.com/en/sensor/rfid-in-automation/rfid-frequencies-and-distance-range/>. [Online].
- [17] Marlene De la Cruz Vélez de Villa, Percy y Reyes Huamán and Daniel Elias Bravo Loayza. Radiofrecuencia de identificación (RFID): microtecnología de gran impacto. *Revista de investigación de Sistemas e Informática*, 7(2):77–86, Diciembre 2010.
- [18] Descubrearduino. ¿cuales son las diferencias entre esp32 y esp8266? Accedido en 09-08-2023 a ,<https://descubrearduino.com/esp32-vs-esp8266/>. [Online].
- [19] ESP32 datasheet. Accedido en 11-09-2023 a ,<https://html.alldatasheet.es/html-pdf/1148023/ESPRESSIF/ESP32/1135/2/ESP32.html>. [Online].
- [20] ESP8266 datasheet. Accedido en 01-09-2023 a ,<https://www.alldatasheet.es/datasheet-pdf/pdf/1132995/ESPRESSIF/ESP8266.html>. [Online].
- [21] RC522 datasheet. Accedido en 20-08-2023 a ,<https://www.alldatasheet.es/datasheet-pdf/pdf/227840/NXP/RC522.html>. [Online].
- [22] PN532 datasheet. Accedido en 10-09-2023 a ,<https://www.alldatasheet.es/datasheet-pdf/pdf/595327/NXP/PN532.html>. [Online].
- [23] Nick Jasuja. Get vs. post. Accedido en 10-07-2023 a ,<https://www.diffen.com/difference/GET-vs-POST-HTTP-Requests>. [Online].
- [24] Az-delivery. Esp-32 dev kit c v4. Accedido en 20-06-2023 a ,<https://www.az-delivery.de/en/products/esp-32-dev-kit-c-v4>. [Online].
- [25] Arduino forum cc. Esp32 can't analogread pin 19 or 22. Accedido en 29-06-2023 a ,<https://forum.arduino.cc/t/esp32-cant-analogread-pin-19-or-22-ugh/949746>. [Online].
- [26] Programar facil. Lector rfid rc522 control de acceso rfid con arduino. Accedido en 08-07-2023 a ,<https://programarfacil.com/blog/arduino-blog/lector-rfid-rc522-con-arduino/>. [Online].

- [27] Luis del Valle Hernández. Lector RFID RC522 control de acceso RFID con arduino. Accedido en 16-07-2023 a ,<https://programarfacil.com/blog/arduino-blog/lector-rfid-rc522-con-arduino/>. [Online].





# Parte III

## Anexos



# Apéndice A

## Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2030.

El contenido de este proyecto contribuye a varios Objetivos de Desarrollo sostenible:

1. **ODS 8 - Trabajo Decente y Crecimiento Económico:** La implementación exitosa del sistema podría generar empleo y crecimiento económico en el sector, especialmente si se adopta en más establecimientos y se convierte en una solución ampliamente utilizada.
2. **ODS 9 - Industria, Innovación e Infraestructura:** Al incorporar tecnología RFID y una base de datos para gestionar el inventario, el proyecto fomenta la innovación y la mejora de la infraestructura de gestión en el sector de los viveros, lo que lleva a una operación más eficiente y sostenible.
3. **ODS 12 - Producción y Consumo Responsables:** Este proyecto permite un control exacto y actualizado del inventario de plantas, el sistema ayuda a reducir el desperdicio y promover una gestión más responsable de los recursos naturales en la producción de estas.
4. **ODS 15 - Vida de Ecosistemas Terrestres:** Al mejorar la gestión de inventario, el proyecto contribuye a la conservación de las plantas y la biodiversidad, esto es debido a que reduce la pérdida de plantas debido a una gestión ineficiente.
5. **ODS 17 - Alianzas para lograr los Objetivos:** Colaborar con tecnologías y soluciones innovadoras como RFID y bases de datos en la nube demuestra cómo las alianzas y la colaboración pueden impulsar el progreso hacia los ODS.

En la tabla [A.1](#) se detalla la relación de este trabajo con los Objetivos de Desarrollo Sostenible (ODS) de la agenda 2030.

<b>Objetivos de Desarrollo Sosteniles</b>	<b>Alto</b>	<b>Medio</b>	<b>Bajo</b>	<b>No procede</b>
ODS 1. Fin de la pobreza				X
ODS 2. Hambre cero				X
ODS 3. Salud y bienestar				X
ODS 4. Educación de calidad				X
ODS 5. Igualdad de género				X
ODS 6. Agua limpia y saneamiento				X
ODS 7. Energía asequible y no contaminante				X
ODS 8. Trabajo decente y crecimiento económico	X			
ODS 9. Industria, innovación e infraestructuras	X			
ODS 10. Reducción de las desigualdades				X
ODS 11. Ciudades y comunidades sostenibles				X
ODS 12. Producción y consumo responsables	X			
ODS 13. Acción por el clima				X
ODS 14. Vida submarina				X
ODS 15. Vida de ecosistemas terrestres	X			
ODS 16. Paz, justicia e instituciones sólidas				X
ODS 17. Alianzas para lograr objetivos		X		X

Tabla A.1: Objetivos de Desarrollo Sostenibles

# Apéndice B

## Programación

### B.1. Arduino

```
1 // Se incluyen las bibliotecas necesarias
2 #include <WiFi.h>
3 #include <MFRC522.h>
4
5 // Se definen algunos pines y variables
6 #define RST_PIN      0
7 #define SS_PIN       5
8 #define AGREGAR      1
9 #define RETIRAR      2
10 byte modoActual = AGREGAR;
11 String invernaderoActual = "1";
12
13 // Se definen las credenciales de la red WiFi a la que se conectará el
14 // dispositivo
15 const char* ssid = "MOVISTAR_3233";
16 const char* password = "simeacuerdo2019";
17
18 // Se inicializan las instancias de las clases MFRC522 y WiFiClient
19 MFRC522 rfid(SS_PIN, RST_PIN);
20 WiFiClient client;
21
22 // En el setup, se inicializan las comunicaciones seriales, SPI y el módulo
23 // RFID, y se conecta a la red WiFi
24 void setup() {
25     Serial.begin(115200);
26     SPI.begin();
27     rfid.PCD_Init();
28
29     WiFi.begin(ssid, password);
30     while (WiFi.status() != WL_CONNECTED) {
31         delay(1000);
32         Serial.println("Conectando a WiFi...");
33     }
34     Serial.println("Conectado a WiFi");
35 }
36
37 // Se define el código RF que tiene asociada cada planta
38 String rfidDatabase(String rfidCode) {
39     if (rfidCode == "F189B51B") return "Geranio";
40     if (rfidCode == "FA686B0") return "Begonia";
41 }
```

```
39  if (rfidCode == "E11A9F1B") return "Orquídea";
40  if (rfidCode == "D08B7C1D") return "Petunia";
41  if (rfidCode == "C09B831D") return "Poinsettia";
42  if (rfidCode == "C0312A1D") return "Hortensia";
43  if (rfidCode == "9271551D") return "Crisantemo";
44  if (rfidCode == "D09E9A1D") return "Vinca";
45  return "Desconocida";
46 }
47
48 void loop() {
49  // Verificar si hay datos disponibles en el puerto serie
50  if (Serial.available() > 0) {
51    String input = Serial.readStringUntil('\n');
52    input.trim();
53
54    // Cambiar el modo de operación a AGREGAR
55    if (input == "AGREGAR") {
56      modoActual = AGREGAR;
57      Serial.println("Modo cambiado a AGREGAR");
58
59      // Cambiar el modo de operación a RETIRAR
60    } else if (input == "RETIRAR") {
61      modoActual = RETIRAR;
62      Serial.println("Modo cambiado a RETIRAR");
63
64      // Cambiar el invernadero actual
65    } else if (input.startsWith("INVERNADERO:")) {
66      invernaderoActual = input.substring(12); // Extraer el número del
invernadero
67      Serial.print("Cambiado a Invernadero: ");
68      Serial.println(invernaderoActual);
69
70      // Consultar el total de plantas en un invernadero
71    } else if (input.startsWith("TOTAL_PLANTAS:")) {
72      consultarTotalPlantas(input.substring(13)); // Extraer el número del
invernadero para consulta
73      return;
74    }
75  }
76
77  // Proceso de lectura RFID y envío de datos
78  if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())
79    return;
80
81  String rfidCode = "";
82  for (byte i = 0; i < rfid.uid.size; i++) {
83    rfidCode += String(rfid.uid.uidByte[i], HEX);
84  }
85  rfidCode.toUpperCase();
86
87  String nombrePlanta = rfidDatabase(rfidCode);
88  enviarDatos(rfidCode, nombrePlanta);
89  delay(1000);
90 }
91
92
93 void enviarDatos(String rfidCode, String nombrePlanta) {
94  // Construir la URL para enviar los datos al servidor
```

```

95 String serverPath = "http://pruebaviverotfg.000webhostapp.com/
    almacenar_rfid.php?";
96 serverPath += "rfid=" + rfidCode;
97 serverPath += "&nombre_planta=" + nombrePlanta;
98 serverPath += "&modo=" + String(modoActual);
99 serverPath += "&invernadero_id=" + invernaderoActual;
100
101 // Enviar la solicitud al servidor
102 if (client.connect("pruebaviverotfg.000webhostapp.com", 80)) {
103     client.println("GET " + serverPath + " HTTP/1.1");
104     client.println("Host: pruebaviverotfg.000webhostapp.com");
105     client.println("Connection: close");
106     client.println();
107
108     // Leer la respuesta del servidor
109     while (client.connected()) {
110         if (client.available()) {
111             String line = client.readStringUntil('\n');
112             if (line.startsWith("Datos recibidos:")) {
113                 Serial.println(line);
114             }
115         }
116     }
117 }
118 client.stop();
119 }
120
121 void consultarTotalPlantas(String invernadero_id) {
122     // Construir la URL para consultar el total de plantas en un invernadero
123     String serverPath = "http://pruebaviverotfg.000webhostapp.com/
    almacenar_rfid.php?action=obtener_total&invernadero_id=" + invernadero_id;
124
125     // Enviar la solicitud al servidor
126     if (client.connect("pruebaviverotfg.000webhostapp.com", 80)) {
127         client.println("GET " + serverPath + " HTTP/1.1");
128         client.println("Host: pruebaviverotfg.000webhostapp.com");
129         client.println("Connection: close");
130         client.println();
131
132         // Leer la respuesta del servidor
133         while (client.connected()) {
134             if (client.available()) {
135                 String line = client.readStringUntil('\n');
136                 if (line.startsWith("Total Plantas:")) {
137                     Serial.println(line);
138                 }
139             }
140         }
141     }
142     client.stop();
143 }

```

Código B.1: Código arduino

## B.2. Scripts PHP

```
1 <?php
```



```
2
3 $servername = "localhost";
4 $username = "id21058636_user_vivero";
5 $password = "PruebaVivero11-";
6 $dbname = "id21058636_vivero_db";
7
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 if ($conn->connect_error) {
11     die("Conexión fallida: " . $conn->connect_error);
12 }
13
14 // Recoger datos desde GET
15 $rfid = $_GET["rfid"];
16 $nombre_planta = $_GET["nombre_planta"];
17 $modo = $_GET["modo"];
18 $invernadero_id = $_GET["invernadero_id"];
19
20 // Verificación de datos recibidos
21 if (!isset($rfid) || !isset($nombre_planta) || !isset($modo) || !isset(
22     $invernadero_id)) {
23     die("Error: No se recibieron todos los datos necesarios.");
24 }
25
26 echo "Datos recibidos: RFID: $rfid, Planta: $nombre_planta, Modo: $modo,
27     Invernadero: $invernadero_id";
28
29 // Verificar si la planta ya existe en la tabla "plantas"
30 $sql_check_planta = "SELECT id FROM plantas WHERE rfid_asociado = '$rfid'";
31 $result_check = $conn->query($sql_check_planta);
32
33 if ($result_check->num_rows == 0) {
34     // Si no existe la planta, la añadimos a la tabla
35     $sql_insert_planta = "INSERT INTO plantas (nombre_planta, rfid_asociado)
36     VALUES ('$nombre_planta', '$rfid')";
37     if (!$conn->query($sql_insert_planta)) {
38         echo "Error: " . $sql_insert_planta . "<br>" . $conn->error;
39     }
40 }
41
42 // Obtenemos el ID de la planta usando el RFID
43 $sql_get_id_planta = "SELECT id FROM plantas WHERE rfid_asociado = '$rfid'";
44 $result_id_planta = $conn->query($sql_get_id_planta);
45 $row = $result_id_planta->fetch_assoc();
46 $planta_id = $row['id'];
47
48 // Ahora vamos a verificar si ya existe una relación entre el invernadero y
49 // la planta en "invernadero_planta"
50 $sql_check_relacion = "SELECT cantidad FROM invernadero_planta WHERE
51     invernadero_id = $invernadero_id AND planta_id = $planta_id";
52 $result_check_relacion = $conn->query($sql_check_relacion);
53
54 if ($result_check_relacion->num_rows == 0) {
55     // Si no existe la relación, creamos una con cantidad = 1 (si estamos en
56     // modo agregar) o no hacemos nada si estamos en modo retirar
57     if ($modo == "1") {
58         $sql_insert_relacion = "INSERT INTO invernadero_planta (
59         invernadero_id, planta_id, cantidad) VALUES ($invernadero_id, $planta_id,
60         1)";
```

```

53     if (!$conn->query($sql_insert_relacion)) {
54         echo "Error: " . $sql_insert_relacion . "<br>" . $conn->error;
55     }
56 }
57 } else {
58     // Si existe la relación, actualizamos la cantidad
59     $row = $result_check_relacion->fetch_assoc();
60     $cantidad_actual = $row['cantidad'];
61
62     if ($modo == "1") {
63         $cantidad_actual++;
64     } elseif ($modo == "2" && $cantidad_actual > 0) {
65         $cantidad_actual--;
66     }
67
68     $sql_update_cantidad = "UPDATE invernadero_planta SET cantidad =
69     $cantidad_actual WHERE invernadero_id = $invernadero_id AND planta_id =
70     $planta_id";
71     if (!$conn->query($sql_update_cantidad)) {
72         echo "Error: " . $sql_update_cantidad . "<br>" . $conn->error;
73     }
74 }
75 // Actualización del total de plantas en la tabla invernaderos después de
76 // cada operación
77 $total_plantas = 0;
78 $sql_total = "SELECT SUM(cantidad) AS total_plantas FROM invernadero_planta
79 WHERE invernadero_id = $invernadero_id";
80 $result_total = $conn->query($sql_total);
81
82 if ($result_total->num_rows > 0) {
83     $row = $result_total->fetch_assoc();
84     $total_plantas = $row['total_plantas'] ?? 0; // El operador '??'
85     devuelve 0 si total_plantas es null
86 }
87
88 $sql_update_invernaderos = "UPDATE invernaderos SET total_plantas =
89 $total_plantas WHERE id = $invernadero_id";
90 if (!$conn->query($sql_update_invernaderos)) {
91     echo "Error al actualizar total de plantas en invernaderos: " . $conn->
92     error;
93 }
94
95 $conn->close();
96 ?>

```

Código B.2: Script php - almacenar\_rfid.php

```

1 <?php
2 header('Content-Type: application/json');
3
4 // Datos de conexión a la base de datos
5 $host = 'localhost';
6 $db = 'id21058636_vivero_db';
7 $user = 'id21058636_user_vivero';
8 $pass = 'PruebaVivero11-';
9 $charset = 'utf8mb4';
10

```

```

11 $dsn = "mysql:host=$host;dbname=$db;charset=$charset";
12 $options = [
13     PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
14     PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
15     PDO::ATTR_EMULATE_PREPARES   => false,
16 ];
17
18 try {
19     $pdo = new PDO($dsn, $user, $pass, $options);
20
21     // Id del invernadero que quieres consultar. Si no se proporciona un
22     // id_invernadero, se establecerá en 1 por defecto.
23     $invernaderoId = isset($_GET['id_invernadero']) ? (int) $_GET['
24     id_invernadero'] : 1;
25
26     // Consulta el total de plantas en el invernadero
27     $stmt = $pdo->prepare("SELECT total_plantas FROM invernaderos WHERE id =
28     ?");
29     $stmt->execute([$invernaderoId]);
30     $totalPlantas = $stmt->fetchColumn();
31
32     // Consulta la cantidad de cada tipo de planta en el invernadero
33     $stmt = $pdo->prepare("SELECT p.nombre_planta, ip.cantidad
34     FROM plantas p
35     INNER JOIN invernadero_planta ip ON p.id = ip.
36     planta_id
37     WHERE ip.invernadero_id = ?");
38     $stmt->execute([$invernaderoId]);
39     $plantas = $stmt->fetchAll();
40
41     // Construye y envía el resultado como JSON
42     echo json_encode([
43         'totalPlantas' => $totalPlantas,
44         'plantas' => $plantas
45     ]);
46 } catch (\PDOException $e) {
47     echo json_encode(['error' => 'Error en la base de datos: ' . $e->
48     getMessage()]);
49 }
50 ?>

```

Código B.3: Script php - invernaderos.php

### B.3. Base de datos

```

1 -- Tabla para los invernaderos
2 CREATE TABLE invernaderos (
3     id INT AUTO_INCREMENT PRIMARY KEY,
4     nombre VARCHAR(255) NOT NULL,
5     total_plantas INT DEFAULT 0
6 );
7
8 -- Insertar los invernaderos
9 INSERT INTO invernaderos (nombre) VALUES
10 ('Invernadero 1'),
11 ('Invernadero 2'),

```

```

12 ('Invernadero 3'),
13 ('Invernadero 4'),
14 ('Invernadero 5'),
15 ('Invernadero 6'),
16 ('Invernadero 7'),
17 ('Invernadero 8'),
18 ('Invernadero 9'),
19 ('Invernadero 10');
20 ('Invernadero 11');
21
22 CREATE TABLE plantas (
23     id INT AUTO_INCREMENT PRIMARY KEY,
24     nombre_planta VARCHAR(255) NOT NULL,
25     rfid_asociado VARCHAR(255) UNIQUE NOT NULL
26 );
27
28 CREATE TABLE invernadero_planta (
29     invernadero_id INT NOT NULL,
30     planta_id INT NOT NULL,
31     cantidad INT NOT NULL DEFAULT 0,
32     PRIMARY KEY (invernadero_id, planta_id),
33     FOREIGN KEY (invernadero_id) REFERENCES invernaderos(id),
34     FOREIGN KEY (planta_id) REFERENCES plantas(id)
35 );

```

Código B.4: MySQL

## B.4. Visual Studio

```

1 <Window x:Class="AppVivero.MainWindow"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6     "
7     xmlns:local="clr-namespace:AppVivero"
8     mc:Ignorable="d"
9     Title="MainWindow" Height="450" Width="800" ResizeMode="NoResize">
10 <Grid>
11     <Button Margin="85,110,480,110" FontFamily="Bahnschrift SemiBold"
12     FontSize="28" Click="Button_Click">
13         <TextBlock Text="Gestionar&#xD;&#xA;Inventario" TextAlignment="
14         Center"/>
15     </Button>
16     <Button Margin="480,110,80,110" FontFamily="Bahnschrift SemiBold"
17     FontSize="28" Click="Button_Click_1">
18         <TextBlock Text="Ver&#xD;&#xA;Inventario" TextAlignment="Center"
19         />
20     </Button>
21
22 </Grid>
23 </Window>

```

Código B.5: MainWindow.xaml

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Windows;
7 using System.Windows.Controls;
8 using System.Windows.Data;
9 using System.Windows.Documents;
10 using System.Windows.Input;
11 using System.Windows.Media;
12 using System.Windows.Media.Imaging;
13 using System.Windows.Navigation;
14 using System.Windows.Shapes;
15
16 namespace AppVivero
17 {
18     /// <summary>
19     /// Interaction logic for MainWindow.xaml
20     /// </summary>
21     public partial class MainWindow : Window
22     {
23         public MainWindow()
24         {
25             InitializeComponent();
26         }
27
28         private void Button_Click(object sender, RoutedEventArgs e)
29         {
30             gestionar window = new gestionar();
31             window.Show();
32         }
33
34         private void Button_Click_1(object sender, RoutedEventArgs e)
35         {
36             ver window = new ver();
37             window.Show();
38         }
39     }
40 }

```

Código B.6: MainWindow.xaml.cs

```

1 <Window x:Class="AppVivero.gestionar"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6     "
7     xmlns:local="clr-namespace:AppVivero"
8     mc:Ignorable="d"
9     Title="Gestión de Inventario" Height="450" Width="800" ResizeMode="
10 NoResize">
11
12 <Window.Resources>
13     <Style TargetType="RadioButton">
14         <Setter Property="Template">

```

```

15         <Setter.Value>
16             <ControlTemplate TargetType="RadioButton">
17                 <StackPanel Orientation="Horizontal">
18                     <Grid Width="30" Height="30">
19                         <Ellipse Fill="Transparent" Stroke="Black"
Width="30" Height="30"/>
20                         <Ellipse x:Name="Dot" Fill="Black" Width="20"
Height="20" Visibility="Collapsed"/>
21                     </Grid>
22                     <ContentPresenter VerticalAlignment="Center"
Margin="10,0,0,0"/>
23                 </StackPanel>
24                 <ControlTemplate.Triggers>
25                     <Trigger Property="IsChecked" Value="True">
26                         <Setter TargetName="Dot" Property="Visibility
" Value="Visible"/>
27                     </Trigger>
28                 </ControlTemplate.Triggers>
29             </ControlTemplate>
30         </Setter.Value>
31     </Setter>
32 </Style>
33
34 </Window.Resources>
35
36 <Grid>
37     <Image Margin="10,10,429,10" Source="/modelocolores.jpg" Stretch="
Fill"/>
38     <RadioButton Content="Agregar Planta" HorizontalAlignment="Left"
Margin="588,31,0,0" VerticalAlignment="Top" Checked="RadioButton_Checked"
FontFamily="Bahnschrift SemiBold" FontSize="22"/>
39     <RadioButton Content="Retirar Planta" HorizontalAlignment="Left"
Margin="588,75,0,0" VerticalAlignment="Top" Checked="RadioButton_Checked"
FontFamily="Bahnschrift SemiBold" FontSize="22"/>
40     <ComboBox Name="InvernaderoComboBox" HorizontalAlignment="Left"
Margin="588,160,0,0" VerticalAlignment="Top" Width="190" SelectionChanged=
"ComboBox_SelectionChanged" Height="22" SelectedIndex="0">
41         <ComboBoxItem Content="Invernadero 1"/>
42         <ComboBoxItem Content="Invernadero 2"/>
43         <ComboBoxItem Content="Invernadero 3"/>
44         <ComboBoxItem Content="Invernadero 4"/>
45         <ComboBoxItem Content="Invernadero 5"/>
46         <ComboBoxItem Content="Invernadero 6"/>
47         <ComboBoxItem Content="Invernadero 7"/>
48         <ComboBoxItem Content="Invernadero 8"/>
49         <ComboBoxItem Content="Invernadero 9"/>
50         <ComboBoxItem Content="Invernadero 10"/>
51         <ComboBoxItem Content="Invernadero 11"/>
52     </ComboBox>
53     <Rectangle HorizontalAlignment="Left" Height="136" Margin="
371,264,0,0" Stroke="Black" VerticalAlignment="Top" Width="407"/>
54     <Label Content="Log:" HorizontalAlignment="Left" Margin="376,264,0,0"
VerticalAlignment="Top" FontWeight="Bold"/>
55     <TextBox Name="LogTextBox" HorizontalAlignment="Left" Margin="
377,289,0,0" VerticalAlignment="Top" RenderTransformOrigin="1.158,0.387"
Height="103" Width="396" IsReadOnly="True" TextWrapping="Wrap"
VerticalScrollBarVisibility="Auto"/>
56
57 </Grid>

```

58 </Window>

## Código B.7: gestionar.xaml

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Windows;
7 using System.Windows.Controls;
8 using System.Windows.Data;
9 using System.Windows.Documents;
10 using System.Windows.Input;
11 using System.Windows.Media;
12 using System.Windows.Media.Imaging;
13 using System.Windows.Shapes;
14 using System.IO.Ports;
15 using System.Net.Http;
16
17
18 namespace AppVivero
19 {
20     public partial class gestionar : Window
21     {
22         private SerialPort arduinoPort;
23         private const string portName = "COM3";
24         private const int baudRate = 115200;
25
26         public gestionar()
27         {
28             InitializeComponent();
29             InitializeArduinoConnection();
30         }
31
32         private void InitializeArduinoConnection()
33         {
34             try
35             {
36                 arduinoPort = new SerialPort(portName, baudRate);
37                 arduinoPort.DataReceived += DataReceivedHandler;
38                 arduinoPort.Open();
39             }
40             catch (Exception ex)
41             {
42                 MessageBox.Show("Error al abrir el puerto: " + ex.Message);
43             }
44         }
45
46         private void DataReceivedHandler(object sender,
47             SerialDataReceivedEventArgs e)
48         {
49             var receivedData = arduinoPort.ReadLine();
50
51             Dispatcher.Invoke(() =>
52             {
53                 logTextBox.AppendText(receivedData + Environment.NewLine);
54
55                 logTextBox.ScrollToEnd();
```

```

56     });
57     }
58
59     private void RadioButton_Checked(object sender, RoutedEventArgs e)
60     {
61         var radioButton = sender as RadioButton;
62         if (radioButton != null && arduinoPort != null && arduinoPort.
IsOpen)
63     {
64         if (radioButton.Content.ToString() == "Agregar Planta")
65         {
66             arduinoPort.WriteLine("AGREGAR");
67         }
68         else if (radioButton.Content.ToString() == "Retirar Planta")
69         {
70             arduinoPort.WriteLine("RETIRAR");
71         }
72     }
73     }
74
75     private void ComboBox_SelectionChanged(object sender,
SelectionChangedEventArgs e)
76     {
77         if (arduinoPort != null && arduinoPort.IsOpen)
78         {
79             var comboBox = sender as ComboBox;
80             var selectedItem = comboBox.SelectedItem as ComboBoxItem;
81
82             if (selectedItem != null)
83             {
84                 string invernadero = selectedItem.Content.ToString();
85                 string number = invernadero.Split(' ')[1]; // Extraer nú
mero del invernadero
86                 arduinoPort.WriteLine("INVERNADERO:" + number);
87             }
88         }
89     }
90
91     protected override void OnClosed(EventArgs e)
92     {
93         base.OnClosed(e);
94
95         if (arduinoPort != null && arduinoPort.IsOpen)
96         {
97             arduinoPort.Close();
98         }
99     }
100 }
101 }

```

Código B.8: gestionar.xaml.cs

```

1 <Window x:Class="AppVivero.ver"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6     xmlns:local="clr-namespace:AppVivero"
7     mc:Ignorable="d"

```



```

8      Title="ver" Height="450" Width="800">
9      <Grid>
10     <Image Margin="215,10,225,10" Source="/modelocolores.jpg" Stretch="
11     Fill"/>
12     <Button Content="Invernadero 1" HorizontalAlignment="Left" Margin="
13     19,16,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
14     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_1"/>
15     <Button Content="Invernadero 2" HorizontalAlignment="Left" Margin="
16     19,86,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
17     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_2"/>
18     <Button Content="Invernadero 3" HorizontalAlignment="Left" Margin="
19     19,156,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
20     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_3"/>
21     <Button Content="Invernadero 4" HorizontalAlignment="Left" Margin="
22     19,226,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
23     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_4"/>
24     <Button Content="Invernadero 5" HorizontalAlignment="Left" Margin="
25     19,296,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
26     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_5"/>
27     <Button Content="Invernadero 6" HorizontalAlignment="Left" Margin="
28     19,366,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
29     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_6"/>
30
31     <Button Content="Invernadero 7" HorizontalAlignment="Left" Margin="
32     580,16,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
33     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_7"/>
34     <Button Content="Invernadero 8" HorizontalAlignment="Left" Margin="
35     580,86,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
36     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_8"/>
37     <Button Content="Invernadero 9" HorizontalAlignment="Left" Margin="
38     580,156,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
39     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_9"/>
40     <Button Content="Invernadero 10" HorizontalAlignment="Left" Margin="
41     580,226,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
42     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_10"/>
43     <Button Content="Invernadero 11" HorizontalAlignment="Left" Margin="
44     580,296,0,0" VerticalAlignment="Top" Height="39" Width="192" FontFamily="
45     Bahnschrift SemiBold" FontSize="28" Click="Button_Click_11"/>
46     <Button Content="Menú" HorizontalAlignment="Left" Margin="580,366,0,0
47     " VerticalAlignment="Top" Height="39" Width="192" FontFamily="Bahnschrift
48     SemiBold" FontSize="28" Click="Button_Click"/>
49     </Grid>
50 </Window>

```

Código B.9: ver.xaml

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Windows;
7 using System.Windows.Controls;
8 using System.Windows.Data;
9 using System.Windows.Documents;
10 using System.Windows.Input;
11 using System.Windows.Media;
12 using System.Windows.Media.Imaging;
13 using System.Windows.Shapes;
14

```

```
15 namespace AppVivero
16 {
17     /// <summary>
18     /// Lógica de interacción para ver.xaml
19     /// </summary>
20     public partial class ver : Window
21     {
22         public ver()
23         {
24             InitializeComponent();
25         }
26
27         private void Button_Click(object sender, RoutedEventArgs e)
28         {
29             this.Close();
30         }
31
32         private void Button_Click_1(object sender, RoutedEventArgs e)
33         {
34             Invernadero1 window = new Invernadero1();
35             window.Show();
36         }
37
38         private void Button_Click_2(object sender, RoutedEventArgs e)
39         {
40             Invernadero2 window = new Invernadero2();
41             window.Show();
42         }
43
44         private void Button_Click_3(object sender, RoutedEventArgs e)
45         {
46             Invernadero3 window = new Invernadero3();
47             window.Show();
48         }
49
50         private void Button_Click_4(object sender, RoutedEventArgs e)
51         {
52             Invernadero4 window = new Invernadero4();
53             window.Show();
54         }
55
56         private void Button_Click_5(object sender, RoutedEventArgs e)
57         {
58             Invernadero5 window = new Invernadero5();
59             window.Show();
60         }
61
62         private void Button_Click_6(object sender, RoutedEventArgs e)
63         {
64             Invernadero6 window = new Invernadero6();
65             window.Show();
66         }
67
68         private void Button_Click_7(object sender, RoutedEventArgs e)
69         {
70             Invernadero7 window = new Invernadero7();
71             window.Show();
72         }
73     }
```

```

74     private void Button_Click_8(object sender, RoutedEventArgs e)
75     {
76         Invernadero8 window = new Invernadero8();
77         window.Show();
78     }
79
80     private void Button_Click_9(object sender, RoutedEventArgs e)
81     {
82         Invernadero9 window = new Invernadero9();
83         window.Show();
84     }
85
86     private void Button_Click_10(object sender, RoutedEventArgs e)
87     {
88         Invernadero10 window = new Invernadero10();
89         window.Show();
90     }
91
92     private void Button_Click_11(object sender, RoutedEventArgs e)
93     {
94         Invernadero11 window = new Invernadero11();
95         window.Show();
96     }
97 }
98 }

```

Código B.10: ver.xaml.cs

```

1 <Window x:Class="AppVivero.Invernadero1"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6     "
7     xmlns:local="clr-namespace:AppVivero"
8     mc:Ignorable="d"
9     Title="Invernadero1" Height="450" Width="800">
10 <Grid>
11     <TextBlock Text="Invernadero 1" FontSize="24" HorizontalAlignment="
12     Center"/>
13     <TextBlock Text="Total de plantas:" Margin="20,60,0,0"/>
14     <TextBlock Name="txtTotalPlantas" Margin="150,60,0,0"/>
15
16     <ListView Name="lstPlantas" Margin="20,100,20,20">
17         <ListView.View>
18             <GridView>
19                 <GridViewColumn Header="Tipo de Planta" Width="150"
20                 DisplayMemberBinding="{Binding TipoPlanta}"/>
21                 <GridViewColumn Header="Cantidad" Width="50"
22                 DisplayMemberBinding="{Binding Cantidad}"/>
23             </GridView>
24         </ListView.View>
25     </ListView>
26 </Grid>
27 </Window>

```

Código B.11: Invernadero.xaml

```
1 using System;
2 using System.Net.Http;
3 using System.Threading.Tasks;
4 using Newtonsoft.Json.Linq;
5 using System.Windows;
6 using System.Windows.Threading;
7
8 namespace AppVivero
9 {
10     public partial class Invernadero1 : Window
11     {
12         private const string Url = "http://pruebaviverotfg.000webhostapp.com/
13         invernaderos.php?id_invernadero=1";
14         private DispatcherTimer updateTimer;
15
16         public Invernadero1()
17         {
18             InitializeComponent();
19             CargarDatos();
20
21             updateTimer = new DispatcherTimer();
22             updateTimer.Interval = TimeSpan.FromSeconds(3);
23             updateTimer.Tick += UpdateTimer_Tick;
24             updateTimer.Start();
25         }
26
27         private void UpdateTimer_Tick(object sender, EventArgs e)
28         {
29             CargarDatos();
30         }
31
32         private async void CargarDatos()
33         {
34             try
35             {
36                 using (HttpClient client = new HttpClient())
37                 {
38                     var response = await client.GetStringAsync(Url);
39                     JObject jsonObject = JObject.Parse(response);
40
41                     int totalPlantas = jsonObject["totalPlantas"].Value<int
42 >();
43
44                     JSONArray plantasArray = (JSONArray)jsonObject["plantas"];
45
46                     lstPlantas.Items.Clear();
47
48                     foreach (var item in plantasArray)
49                     {
50                         string tipoPlanta = item["nombre_planta"].ToString();
51                         int cantidad = int.Parse(item["cantidad"].ToString());
52
53                         lstPlantas.Items.Add(new { TipoPlanta = tipoPlanta,
54 Cantidad = cantidad });
55                     }
56
57                     txtTotalPlantas.Text = totalPlantas.ToString();
58                 }
59             }
60         }
61     }
62 }
```

```
56     }
57     catch (Exception ex)
58     {
59         MessageBox.Show($"Ocurrió un error al obtener los datos: {ex.
60         Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
61     }
62 }
63 }
```

Código B.12: Invernadero.xaml.cs