Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Sistema de riego inteligente para jardines verticales

Trabajo Fin de Máster

Máster Universitario en Ingeniería Informática

Autor: Sara Lerma Sánchez

Tutor: Joan Fons Cors

Curso 2019/2020

Agradecimientos

A mi pareja, mi hermana, mi madre y Rosa por apoyar mis decisiones y motivarme hacia nuevos objetivos.
A mi tutor por resolver mis dudas, por estar siempre ayudándome y con entusiasmo y por creer en el desarrollo de este proyecto desde el principio.
Y sobre todo a mi padre que con su entusiasmo, dedicación y paciencia ha aportado siempre todo lo que ha estado en su mano para mi formación en general.

Resumen

Hoy en día gracias a Internet de las Cosas (IoT) se puede controlar cualquier objeto desde cualquier parte a través de internet, sin embargo, cada vez más, se espera que los sistemas sean autónomos y tomen decisiones correctas por sí solos que permitan ahorrar agua, luz, batería, etc. Por otra parte, como es conocido, el mundo agrícola es un sector donde se realiza un mayor consumo de agua, lo que supone la necesidad de un sistema de riego eficiente. Debido a ello, se desea implementar un sistema de riego inteligente para jardines verticales. El proyecto consiste en la creación de un programador de riego que no necesita supervisión humana ya que proporciona a las plantas el agua necesaria en función de la humedad del suelo requerida por cada tipo de planta, logrando así una optimización de agua. El sistema es controlado por un programador que, basándose en los parámetros de humedad de suelo de cada zona, configurados por el usuario, gestiona el riego. La configuración de dichos parámetros, así como la visualización de los valores de los sensores y el control del riego, se puede realizar desde cualquier lugar y desde cualquier dispositivo mediante una página web. Esto se logra gracias a un servidor alojado en la nube que permite la comunicación entre el programador y la página web con la que interactúa el usuario. Para la creación de dicho sistema se desarrolla un programador desde cero y se usan las tecnologías punteras como son React para la creación de páginas web, Strapi para la gestión de la parte *backend* y AWS para el almacenamiento en la nube. Con el fin de validar la propuesta se realiza un prototipo funcional del sistema comentado.

Palabras clave: IoT, optimización agua, sistema riego inteligente, jardines verticales, nube

Resum

Hui en dia gràcies a Internet de les Coses (IoT) es pot controlar qualsevol objecte des de qualsevol part a través d'internet, no obstant això, cada vegada més, s'espera que els sistemes siguen autònoms i prenguen decisions correctes per si sols que permeten estalviar aigua, llum, bateria, etc. D'altra banda, com és conegut, el món agrícola és un sector on es realitza un major consum d'aigua, la qual cosa suposa la necessitat d'un sistema de reg eficient. A causa d'això, es desitja implementar un sistema de reg intel·ligent per a jardins verticals. El projecte consistix en la creació d'un programador de reg que no necessita supervisió humana ja que proporciona a les plantes l'aigua necessària en funció de la humitat del sòl requerida per cada tipus de planta, aconseguint així una optimització d'aigua. El sistema és controlat per un programador que, basant-se en els paràmetres d'humitat de sòl de cada zona, configurats per l'usuari, gestiona el reg. La configuració d'aquests paràmetres, així com la visualització dels valors dels sensors i el control del reg, es pot realitzar des de qualsevol lloc i des de qualsevol dispositiu per mitjà d'una pàgina web. Açò s'aconseguix gràcies a un servidor allotjat en el núvol que permet la comunicació entre el programador i la pàgina web amb què interactua l'usuari. Per a la creació del dit sistema es desenrotlla un programador des de zero i s'usen les tecnologies punteres com són React per a la creació de pàgines web, Strapi per a la gestió de la part *backend* i AWS per a l'emmagatzemament en el núvol. A fi de validar la proposta es realitza un prototip funcional del sistema comentat.

Paraules clau: Iot, optimització aigua, sistema reg intel·ligent, jardins verticals, núvol

Abstract

Nowadays, thanks to the Internet of Things (IoT), any object can be controlled from anywhere through the Internet, however, more and more, systems are expected to be autonomous and make correct decisions on their own that allow saving water, light, battery, etc. On the other hand, as it is known, the agricultural world is a sector where there is a greater consumption of water, which implies the need for an efficient irrigation system. Due to this, it is desired to implement an intelligent irrigation system for vertical gardens. The project consists of the creation of an irrigation programmer that does not need human supervision since it provides the plants with the necessary water based on the soil moisture required by each type of plant, thus achieving water optimization. The system is controlled by a programmer that, based on the soil moisture parameters of each zone, configured by the user, manages the irrigation. The configuration of these parameters, as well as the visualization of the sensor values and the irrigation control, can be done from anywhere and from any device through a web page. This is achieved thanks to a server hosted in the cloud that allows communication between the programmer and the web page with which the user interacts. For the creation of this system, a programmer is developed from scratch and cutting-edge technologies are used such as React for creating web pages, Strapi for managing the backend part, and AWS for cloud storage. In order to validate the proposal, a functional prototype of the commented system is made.

Keywords: IoT, water optimization, smart irrigation system, vertical gardens, cloud

Tabla de contenidos

Tabla de figuras

1. Introducción

1.1 Motivación

El agua es un recurso imprescindible para la supervivencia tanto de los seres vivos como del ecosistema. El cambio climático y los hábitos de vida de la sociedad de hoy en día están haciendo disminuir las reservas de agua a un ritmo preocupante como se puede observar con el aumento del nivel del mar, las sequías, el deshielo de los polos, entre otros factores. Por ello, es muy importante concienciarse de hacer una buena gestión y un uso sostenible del agua.

Uno de los sectores más populares en los que se debe tomar medidas es en el mundo de la agricultura, ya que, en Europa la irrigación puede suponer un porcentaje del consumo de agua para la agricultura hasta un 80%. Por lo que es necesario un sistema de riego eficiente y que permita aprovechar al máximo el agua sin desperdiciarla.

Ante esta situación y teniendo en cuenta que el uso jardines verticales estos últimos años ha aumentado debido a que además de adornar una fachada, refrescan el ambiente y regulan el efecto térmico del clima sobre los muros lo que permite ahorrar hasta un 20% de la energía necesaria para calentar o enfriar un edificio. Por ese motivo, se ha pensado en crear un sistema de riego inteligente para jardines verticales que permita la optimización del agua en el riego.

En un sistema de riego hay muchos factores a considerar que determinan cuándo se debe regar, como por ejemplo, si una zona recibe más el sol que otras puede que necesite más agua o si ha llovido puede que no sea necesario que se riegue ese día. Si se dispone de un sistema de riego programado que únicamente active el riego a ciertas horas del día, se escapan muchos factores que hay que tener en cuenta ya que puede que no se esté atendiendo correctamente las necesidades de la planta, además del desperdicio de agua que supone.

En este proyecto se logra la optimización del agua mediante el suministro justo de agua por cada zona del jardín vertical cuando es necesario. Esto se realiza mediante la medición de la humedad de suelo de cada zona, ya que solamente se activa el riego

de esa zona específica cuando la humedad de suelo sea menor a la necesaria, de esta manera se consigue que la zona de plantas reciba el agua justa y necesaria para vivir sin desperdiciar agua, reduciendo costes y ahorrando tiempo y esfuerzo.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Máster (TFM) es desarrollar un sistema de riego inteligente para jardines verticales que permita la optimización del agua mediante un sistema de irrigación eficiente y así mejorar la calidad de vida de las plantas ubicadas en el jardín vertical. Para ello, el sistema de riego inteligente dispone de un sensor de temperatura externa y de sensores de humedad de suelo, tantos como zonas disponga el jardín vertical. En este proyecto se realiza el sistema para cuatro zonas, pero puede ser extensible. Por otra parte, dispone también de válvulas de agua, que al igual que los sensores de humedad, son necesarias tantas válvulas como zonas tenga el jardín vertical. Todos estos dispositivos son controlados mediante un programador, el cual se comunica en tiempo real mediante *websockets* con el servidor AWS. La página web con la que interactúa el usuario remotamente, sin necesidad de estar en el lugar donde se encuentra el jardín vertical, se comunica directamente con el servidor alojado en la nube, obteniendo y mandando información en tiempo real. Por lo tanto, se debe tener en cuenta los siguientes objetivos específicos a la hora del desarrollo de dicho TFM:

- Informar de la temperatura externa.
- Informar de la humedad de suelo de cada zona en tiempo real.
- El usuario debe poder registrar el programador a una red wifi para poder controlar el programador a distancia.
- El usuario debe poder registrar el programador a una red *cloud* donde el programador pueda enviar toda la información recolectada y recibir información.
- El usuario debe poder acceder a la información del programador, así como controlar el riego en cualquier momento y desde cualquier lugar a través de una página web.
- El usuario debe poder iniciar sesión en una página web para poder monitorizar el estado de su jardín vertical.
- El usuario debe poder especificar mediante una página web los rangos de humedad de suelo entre los que debe estar la humedad de suelo de cada zona.

- El usuario debe poder cambiar la contraseña de su cuenta en una página web.
- El usuario debe poder cerrar sesión de su cuenta en la página web.
- Controlar el riego de cada zona individualmente mediante una válvula de agua por zona dependiendo del rango de humedad de suelo especificada.

1.2.1 Objetivos de Desarrollo Sostenible

Como se ha comentado en este apartado, se desea desarrollar un sistema de irrigación eficiente de manera que se logre la máxima optimización del agua posible. Con este proyecto se contribuye en los Objetivos de Desarrollo Sostenible (ODS). La Agenda 2030, adoptada por la Asamblea General de las Naciones Unidas, plantea los 17 ODS que se muestran en la Figura 1.

Figura 1 - Objetivos de desarrollo sostenible

Con el desarrollo del presente proyecto se contribuye, en mayor o menor medida, con los siguientes objetivos:

6. Agua limpia y saneamiento: Garantizar la disponibilidad de agua y su gestión sostenible y el saneamiento para todos.

13. Acción por el clima: Adoptar medidas urgentes para combatir el cambio climático y sus efectos.

15. Vida de ecosistemas terrestres: Gestionar sosteniblemente los bosques, luchar contra la desertificación, detener e invertir la degradación de las tierras y detener la pérdida de biodiversidad.

Desarrollando un sistema de irrigación eficiente se logra una gestión sostenible del agua, con lo que se ayuda a combatir el cambio climático. No solo por el ahorro de agua sino también por el hecho de que se alienta a que más personas utilicen jardines verticales, consiguiendo así mayores espacios con plantas debido a la comodidad de mantener un jardín que proporciona dicho sistema incorporado.

1.3 Metodología

Para la realización de este proyecto se ha seguido una metodología ágil. La metodología ágil está basada en el desarrollo iterativo e incremental, donde los requisitos y las soluciones van evolucionando a medida que se realiza el proyecto. Se ha utilizado dicha metodología en las diferentes áreas que contiene este proyecto, a continuación, se exponen algunos de los ejemplos del desarrollo iterativo e incremental que se ha seguido en diferentes áreas:

- En el área de hardware se ha desarrollado el sistema de riego en un *protoboard*. En primer lugar, se ha probado el funcionamiento de cada componente por separado, como por ejemplo el funcionamiento del sensor de humedad de suelo con el módulo Arduino, observando que se realizan las lecturas correctamente. Posteriormente, se han interconectado todos los componentes electrónicos entre sí, como los sensores, las válvulas de agua, los módulos y demás componentes electrónicos necesarios y, una vez asegurado que funciona en conjunto, finalmente se ha realizado el diseño de la placa de circuito impreso, se han soldado todos los componentes y se ha comprobado su correcto funcionamiento.
- Otro ejemplo reside en el área de creación de la página web. En primer lugar, solamente se implementó el control sobre las válvulas de agua, en segundo lugar, se implementó la visualización del valor de humedad de suelo de cada zona y, a continuación, se implementó las áreas donde el usuario pueda configurar los rangos de humedad de suelo para cada zona, posteriormente se realizó el inicio/cierre de sesión y, finalmente, se implementó el cambio de contraseña.

Como puede observarse, en las áreas en las que se puede desglosar el proyecto se ha seguido un desarrollo incremental, donde primeramente se dotaban de las funcionalidades más necesarias y posteriormente se han ido añadiendo más funcionalidades.

En este proyecto se ha hecho uso de Clubhouse [1], una plataforma que permite gestionar proyectos para el desarrollo de software. En dicha plataforma se han definido 3 grandes bloques en los que se puede desglosar el proyecto. Estos bloques son, como se puede observar en la Figura 2, *Backend*, Web y *Hardware*.

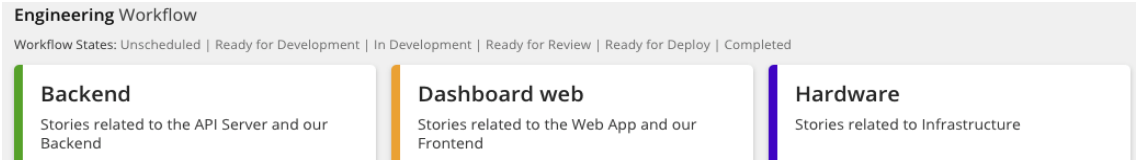


Figura 2 - Épicas definidas en Clubhouse

Dentro de cada proyecto, se han definido una serie de historias, como se puede apreciar en la Figura 3.

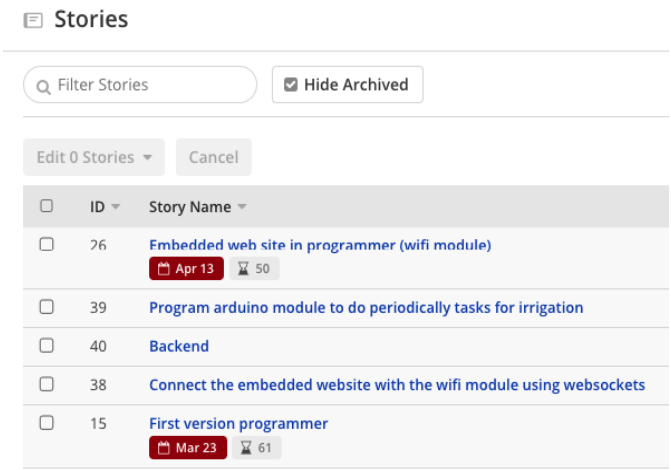


Figura 3 - Historias definidas en Clubhouse

Dichas historias están compuestas por tareas como se observa en la Figura 4.

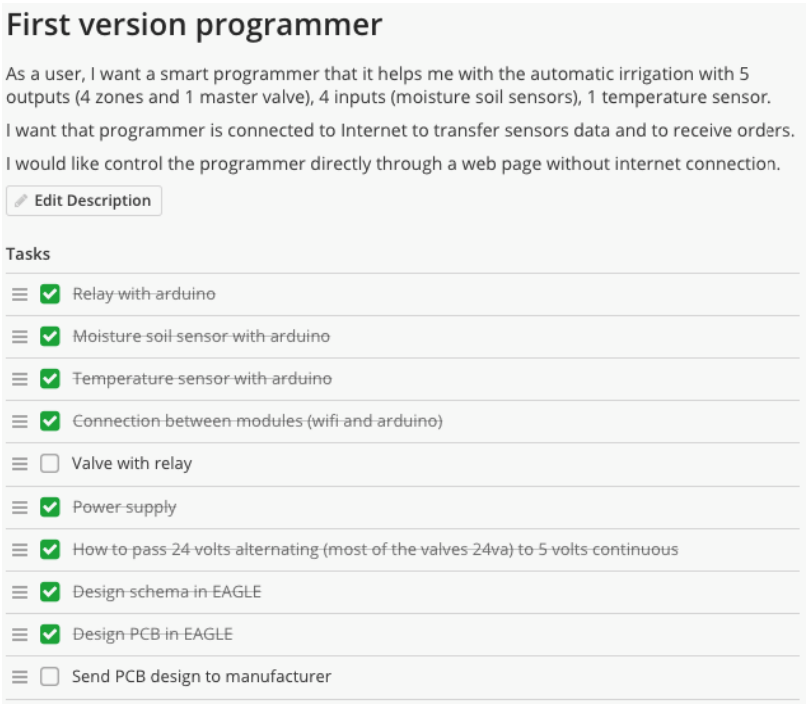


Figura 4 - Tareas definidas en Clubhouse

Gracias a la metodología ágil se ha dispuesto en todo momento, a lo largo del proceso de desarrollo, un producto más o menos desarrollado, pero funcional, que era el objetivo. Por otra parte, gracias a la gestión del proyecto mediante la plataforma ClubHouse [1], se ha podido definir las tareas previamente y tener una mayor gestión del tiempo y del esfuerzo así como una mejor organización.

1.4 Estructura del documento

Esta memoria está compuesta por diez capítulos los cuales siguen un flujo de desarrollo.

Capítulo 1 – Introducción: Se introduce la idea general de proyecto, donde se expone la motivación principal para la creación de este proyecto, los objetivos, la metodología que se ha seguido y la estructura de dicho documento.

Capítulo 2 – Estado del arte: Se exponen otros productos existentes en el mercado que realizan funcionalidades similares al presente proyecto y se expone la propuesta y la diferencia del producto desarrollado con los existentes en el mercado.

Capítulo 3 – Análisis del problema: Se especifican los requisitos, se muestra tanto la solución de hardware propuesta justificando el motivo de su elección en los anexos como la solución de software propuesta y se desglosa el presupuesto calculado.

Capítulo 4 – Diseño de la solución: Se presenta la arquitectura del sistema desarrollado, el diseño detallado de la parte software, como son los diagramas de flujo y diagramas UML, como de la parte hardware, como es el diseño de la placa PCB desde cero. Además, se muestra el conexionado general del hardware y se presentan las tecnologías que se han utilizado.

Capítulo 5 – Desarrollo de la solución: En este capítulo se muestra el desarrollo del software, se desglosa en varias secciones para una mejor comprensión y se expone la conexión del hardware de cada componente, así como su soldadura y el montaje del hardware.

Capítulo 6 – Implantación: Se presentan los pasos a realizar en la instalación del sistema desarrollado, desde la parte del software, la previa configuración del wifi y el registro de un nuevo usuario y desde la parte del hardware, la configuración que habría que realizar en la implantación del proyecto.

Capítulo 7 – Conclusiones: Se comentan las conclusiones a las que se han llegado tras la realización del TFM así como la relación del presente proyecto con las asignaturas del máster MUIINF.

Capítulo 8 – Trabajos futuros: En este capítulo se presentan los trabajos futuros que se podrían realizar a partir de este proyecto.

Capítulo 9 – Referencias: Se comparten los enlaces de las referencias utilizadas, tanto de materiales utilizados para realizar el proyecto como las fuentes donde se han obtenido los componentes hardware.

Anexos: Se encuentran dos documentos, uno que es el Manual de usuario y otro donde se muestran características técnicas de cada dispositivo hardware seleccionado, sus ventajas y desventajas y el motivo de elección de dichos dispositivos para este proyecto.

2. Estado del arte

¿Cuál es la diferencia de un jardín vertical a un jardín horizontal?

La principal diferencia es que, en un jardín vertical, el agua se ve afectada por el fenómeno de la gravedad, por lo que caerá hacia abajo abasteciendo en mayor cantidad a las plantas situadas en la parte inferior que a las situadas en la parte superior. Por ello, es necesario un sensor de humedad en cada zona para asegurarse de que todas las zonas del jardín reciben el agua necesaria. Los sistemas de riego convencionales para jardines horizontales no suelen tener en cuenta este factor puesto que no se da este fenómeno.

Cabe destacar que no se han encontrado sistemas de riego inteligentes para jardines verticales. Por lo que se procede a estudiar los sistemas de riego inteligentes existentes en el mercado para jardines horizontales.

JARDIBRIC

La mayoría de los sistemas del mercado no son inteligentes, es decir, no toman ellos mismos la decisión de regar basándose en unos parámetros de entrada, sino que están programados para activarse a determinadas horas durante un cierto periodo de tiempo. JARDIBIC [2] es un claro ejemplo, ya que es un programador que permite programar una frecuencia de riego desde una hora a una semana y una duración de máximo un minuto a una determinada hora.

Figura 5 - Sistema de riego JARDBIC

Hunter Eco-Logic

A continuación, se muestra programadores de riego que se basa en datos meteorológicos o en pluviómetros para realizar el riego. Por ejemplo, Hunter Eco-Logic [3] es un programador que permite configurar dos programas independientes con cuatro tiempos de arranque cada programa. Dispone de *bypass* del sensor de lluvia, es decir, dispone de un conmutador que controla la señal de entrada del sensor de lluvia y cuando el conmutador está en posición “Bypass”, el programador regará normalmente ignorando la señal del sensor.

Figura 6 - Sistema de riego Hunter Eco-Logic

SensoTimer ST6 Duo eco!ogic

También se encuentran programadores de riego con sensores de humedad. Como es el caso del programador SensoTimer ST6 Duo eco!ogic [4] que, a diferencia de los anteriores, riega en función de las necesidades del jardín dado que realiza el control mediante sensores de humedad del suelo. Permite configurar hasta dos momentos de riego al día con un tiempo de riego máximo de 90 minutos. Por último, se debe destacar que requiere de tres baterías de 9V, una para el panel de mando y otras dos para cada uno de los sensores.

Figura 7 - Sistema de riego SensoTimer ST6 Duo eco!ogic

Fliwer Sense&Water

Finalmente, se encuentra un sistema de riego inteligente que dispone de sensor de humedad de suelo y posee una electroválvula en el interior que se puede utilizar para controlar un sistema de micro-riego. Además, se puede elegir el kit Fliwer Sense&Water WiFi [5] o el kit de 3G que incluye el sensor y el receptor, esto permite que mediante la plataforma se pueda consultar datos procedentes del sensor Fliwer.

Figura 8 - Sistema de riego Fliwer Sense&Water

2.1 Crítica al estado del arte

En primer lugar, se requiere un sistema de riego inteligente que decida regar cuando las plantas lo necesiten, no se requiere un sistema que riegue cuando una programación configurada lo ordene, como es el caso del programador JARDIBRIC.
En segundo lugar, se requiere un sistema que se base en sensores de humedad de suelo y no se base en pluviómetros o datos meteorológicos. Esto se debe a que se requiere un riego uniforme y preciso, por lo que con la medida de un pluviómetro o datos meteorológicos es posible que algunas zonas del jardín vertical no hayan sido humedecidas suficiente o uniformemente. Esto puede ocurrir por diversas causas, ya sea por la ubicación de la zona, por el viento, porque una zona reciba más sol o simplemente las plantas situadas bajo del jardín no reciban el agua necesaria.
En tercer lugar, se requiere una página web o una aplicación móvil para poder visualizar y controlar remotamente, es decir, a distancia el estado del jardín vertical. Por lo que, aunque el programador SensoTimer ST6 Duo eco!ogic cumpla con los anteriores requisitos, no dispone de una página web o una aplicación móvil.
Finalmente, el sistema de riego encontrado que cumple con la mayoría de los requisitos es Fliwer Sense&Water sin embargo, aparte de que el precio es desorbitado, el sensor de suelo utilizado no valdría para jardines verticales debido que es un tamaño muy grande, y es necesario un sensor de humedad suelo mucho más pequeño y menos pesado.
En conclusión, como se puede observar, ninguno de los sistemas de riego existentes está diseñados para jardines verticales ni se ciñen a los requisitos especificados.

2.2 Propuesta

La propuesta que se presenta trata de un sistema de riego inteligente para el cuidado de jardines verticales. El sistema de riego desarrollado en este proyecto se muestra en la Figura 9 y dispone de cuatro sensores de humedad de suelo para poder obtener la humedad de cada zona, un sensor de temperatura externa, cuatro válvulas de agua para poder regar cada zona por separado. El usuario puede observar la temperatura externa y la humedad de suelo de cada zona, así como controlar manualmente el funcionamiento de las válvulas desde cualquier lugar y en tiempo real desde una página web, ya que el sistema diseñado envía los datos de los sensores y el estado de las válvulas mediante Wifi a la nube. A diferencia de los sistemas de riego expuestos anteriormente, el sistema desarrollado en este proyecto está destinado para el cuidado de jardines verticales ya que el sensor de humedad de suelo utilizado es un sensor preciso y con la medida idónea para su ubicación en los bolsillos de un jardín vertical. Además, el riego se hace de manera inteligente, basándose en la humedad de suelo que necesita cada planta y se puede controlar todo este sistema remotamente desde una página web, donde se pueden especificar los puntos de consigna a partir de los cuales se deben activar o desactivar las válvulas de agua. Cabe destacar que el prototipo desarrollado solo se muestran tres zonas pero el sistema se ha desarrollado para poder disponer de cuatro zonas.

Figura 9 - Sistema de riego SMARTICAL

3. Análisis del problema

En este apartado se va a exponer primeramente los requisitos que se deben tener en cuenta para el desarrollo del producto, a continuación, la identificación y el análisis de las posibles soluciones, la solución propuesta y finalmente el presupuesto calculado.

3.1 Especificación de requisitos

Se requiere un sistema inteligente que active el riego de cada zona en función de la humedad de suelo necesaria para cada planta. Dispone de una aplicación web donde se pueden fijar los puntos de consigna a partir de los cuales se deben activar las válvulas de agua de cada zona, ya que cada zona puede necesitar una humedad de suelo diferente.

Los requisitos que se deben cumplir son los especificados a continuación:

- Las plantas deben ser regadas según la humedad de suelo que precise cada una.
- El usuario debe poder visualizar la temperatura ambiente que hay en su jardín remotamente.
- El usuario debe poder visualizar la humedad de suelo de cada zona remotamente.
- El usuario debe poder visualizar el estado de las válvulas de agua de cada zona remotamente.
- El riego debe poder ser manipulado manualmente, por lo que el usuario debe poder actuar el estado de las válvulas de agua, abriéndolas o cerrándolas, de cada zona remotamente.
- El usuario debe poder configurar los rangos de humedad de suelo de cada zona remotamente.

Cabe destacar que en este proyecto el jardín vertical se divide en zonas para permitir así una gestión de riego especializada por cada zona ya que en cada zona se riega dependiendo de los puntos de consigna de humedad de suelo especificados. Esto permite tener diferentes especies de plantas ya que cada especie necesita una humedad de suelo específica la cual depende de las condiciones, el tipo de suelo, su estructura, su composición, pero sobre todo del ADN de la planta.

3.2 Identificación y análisis de soluciones posibles

Se desea crear una placa de circuito impreso, por lo que los componentes seleccionados deben serán soldados a la placa de circuito impreso o conectados mediante bornas que dispondrá dicha placa. Cabe destacar que, como se ha comentado en el apartado “Estructura del documento”, se adjunta en el anexo un documento llamado “Características técnicas de dispositivos” especificando las características técnicas, las ventajas y desventajas de cada dispositivo que se expone a continuación y el motivo de elección de dicho dispositivo.

3.2.1 Sensores

Un sensor es un dispositivo que detecta cambios en su entorno y envía la información a otros dispositivos electrónicos. Concretamente capta magnitudes físicas, como por ejemplo, la temperatura u otras alteraciones de su entorno y las transformar en magnitudes eléctricas.

3.2.1.1 Sensor de humedad del suelo

Para el cumplimiento del requisito en que el usuario debe poder visualizar la humedad de suelo de cada zona remotamente y el requisito en el que las plantas deben ser regadas según sus necesidades, es necesario medir la humedad de suelo en cada zona, así como, regar dicha zona. Para la obtención del valor de humedad de suelo se utiliza un sensor que indica la humedad de la tierra en la que está ubicado.

I. Descripción

Un sensor de humedad del suelo se emplea para medir el contenido de humedad del suelo, es decir, la cantidad de agua por volumen de tierra que hay en un terreno. También conocido mediante las siglas VWC (*Volumetric Water Content*) y concretamente mide la relación entre el volumen de agua y el volumen del suelo.
El sensor SoilWatch 10 [6] es un sensor de humedad del suelo que permite medir el contenido relativo de agua en el suelo. Este sensor mide los niveles de humedad del suelo mediante detección capacitiva, no la mide como otros sensores de humedad, como es el caso de los sensores resistivos. En comparación con otros sensores de

suelo, las lecturas de la humedad no están variando constantemente ya que este sensor proporciona las mismas lecturas independientemente de la tensión de alimentación.

Figura 10 - Sensor de humedad de suelo Soilwatch 10

Gracias a su diseño, tiene una larga duración dado que es resistente al agua y a la intemperie. Se puede enterrar en el suelo durante un período prolongado sin efectos adversos sobre la precisión ya que no hay electrodos expuestos que se corroan en pocas semanas.

3.2.1.2 Sensor de temperatura

Para el cumplimiento del requisito en el que el usuario debe poder visualizar la temperatura ambiente que hay en su jardín de forma remota se va a utilizar un sensor de temperatura NTC.

I. Descripción

Un sensor de temperatura es un dispositivo que está midiendo constantemente la temperatura, la cual es una señal física, dicha señal física es transformada en una señal eléctrica y ésta es procesada por un equipo eléctrico o electrónico. Existen muchísimos tipos de sensores para medir la temperatura, en este proyecto se utiliza un termistor. Un termistor es un sensor de temperatura por resistencia, que varía su valor con la temperatura. Existen dos tipos de termistores, los termistores NTC *(Negative Termperature Coefficiente*) que varían su valor con un coeficiente negativo, es decir, si aumenta la temperatura, disminuye el valor de la resistencia y los termistores PTC (*Positive Termperature Coefficiente*), los cuales actúan al contrario

que los NTC, ya que varían su valor con un coeficiente positivo, es decir, si aumenta la temperatura, aumenta el valor de la resistencia.

Figura 11 - Sensor de temperatura 103AT-11

El sensor de temperatura 103AT-11 [7] es un termistor NTC de alta precisión con salida analógica que está encapsulado por un plástico impermeable. Dicho plástico asegura el aislamiento del cable y del sensor con un grado de protección IP67, por lo que tiene una protección completa contra el polvo y se puede realizar una inmersión completa en agua.

3.2.2 Actuadores

Un actuador es un capaz de transformar una señal eléctrica en una variable física como puede ser el sonido, por ejemplo. En este proyecto se utiliza un actuador, que es la válvula de agua. Antes de comenzar con la explicación de la válvula de agua, es necesario mencionar cómo se va a realizar la activación y la desactivación de dicho actuador.

Relés

Para la gestión de las válvulas de agua son necesarios tantos relés como válvulas se dispongan. Dichos relés son necesarios debido a que la tensión de salida del módulo Arduino, posteriormente explicado, es de 5 VDC, y las válvulas de agua necesitan 24VAC para su activación. Por ese motivo se utilizan relés, los cuales son manejados

desde la programación en Arduino y permiten gestionar la alimentación de dichas válvulas. Posteriormente en el apartado Diseño detallado se explicará detalladamente la conexión de los relés y las válvulas de agua.

Los relés utilizados son relés PCN-105D3MHZ [8], que como se observa en la Figura 12, es un tipo de relé de potencia compacto y delgado para PCB con bobina de 5VDC. Se ha elegido este tipo de relé por su tamaño, porque se pueden soldar en la PCB y porque se puede alimentar con la tensión de salida de 5VDC del Arduino.

Figura 12 - Relé PCN-105D3MHZ

3.2.2.1 Válvula de agua

Para el cumplimiento de los requisitos relacionados con riego como son que las plantas deben ser regadas según sus necesidades, concretamente, según la humedad de suelo y que el riego debe poder ser manipulado manualmente, por lo que el usuario debe poder actuar el estado de las válvulas de agua, abriéndolas o cerrándolas, de cada zona remotamente, son necesarias válvulas de agua que permiten dejar pasar el agua, es decir, regar, o por el contrario, no dejar pasar el agua, cerrando la válvula de agua, pudiendo así finalizar el riego.

I. Descripción

La válvula de agua es un instrumento que permite controlar el flujo de agua que pasa a través de ella. Por otro, existen válvulas solenoides de agua, que son válvulas de agua automáticas, es decir, que no es necesario que sean manipuladas manualmente. Las válvulas solenoides para agua son controladas mediante una corriente eléctrica con el uso de un solenoide, que es un dispositivo que permite crear un campo magnético, y

gracias a ello es posible mover un émbolo y actuar sobre la válvula de agua abriéndola o cerrándola.

La válvula de agua que se utiliza en este proyecto es la que se puede observar en la Figura 13, es una válvula de agua Serie Mini RPE de la marca Euro-Rain [9], dicha válvula es una válvula que no necesita flujo regulado y permite o detiene el paso del flujo de agua dependiendo de si se le aplica energía o no que produzca el comentado campo magnético.

Figura 13 - Válvula de agua Serie Mini RPE

3.2.3 Ordenador del sistema

Tanto los sensores de humedad del suelo y de temperatura como las válvulas de agua deben ser consultados y controlados. Por ello son conectados a un ordenador que recibe la lectura de dichos sensores, los procesa y dependiendo de los valores, actúa de una manera u otra sobre las válvulas de agua. En otras palabras, el ordenador es el dispositivo en el cual se ejecuta la aplicación desarrollada para el control de todos los dispositivos.

En este proyecto la parte del "ordenador del sistema" está compuesto por dos dispositivos, por una placa con microcontrolador y por un módulo wifi que va a permitir actuar con los dispositivos remotamente. Gracias a esta combinación, en el caso en el que el módulo wifi falle, el programa sigue funcionando porque está ejecutándose en el microcontrolador, sin embargo, no podrá programarse a distancia.

3.2.3.1 Wemos D1 mini pro

Para el cumplimiento de los requisitos en los que se debe visualizar la temperatura externa, la humedad de suelo de cada zona y controlar remotamente, es decir, a distancia, el estado del jardín es necesario que el sistema se conecte a internet. Para poder realizar la conexión a internet se utiliza una placa Wifi que permite enviar y recibir datos de internet.

I. Descripción

Wemos D1 mini Pro [10] es una placa Wifi basada en el ESP-8266EX de tamaño muy pequeño, con una memoria flas de 16MB y con 11 pines digitales. Además, dispone de un conversor CP2104 de USB a Serial. Dicha placa dispone de una antena de cerámica, la cual proporciona una buena calidad de recepción y envío de datos, además dispone de un conector para poner una antena externa, lo cual permite un mayor alcance.

Figura 14 - Wemos D1 mini Pro

Para controlar dicho hardware, se puede realizar con una herramienta de programación de código abierto que dispone Arduino, Arduino IDE, que posteriormente se explicará en detalle.

3.2.3.2 Arduino Nano

Para poder interactuar con los sensores analógicos se utiliza una placa que lee el valor de dichos sensores, procesa dichos valores y realiza acciones acordes con los valores, como por ejemplo actuar sobre una válvula de agua situada en una zona específica.

I. Descripción

Arduino [11] es, en lo que al hardware respecta, una placa pequeña compuesta, entre otros componentes, por un microcontrolador ATMega328. Arduino Nano puede ser alimentado a través de una conexión Mini-B USB, una fuente de alimentación no regulada de 6-20V (pin 30) o una fuente de alimentación regulada de 5V (pin 27).

Respecto a los pines, contiene 30 pines, de los cuales, 8 pines analógicos, 14 pines digitales, 2 pines de reinicio y 6 pines de potencia. Los pines analógicos y digitales deben configurarse como entradas o salidas.

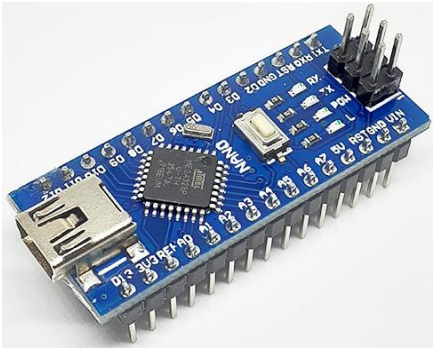


Figura 15 - Arduino Nano

Para controlar dicho hardware, Arduino tiene una herramienta de programación de código abierto, es decir, tiene su propio software llamado Arduino IDE (*Integrated Development Environment*) el cual es un entorno de desarrollo integrado que funciona *online* y *offline*. Dicho IDE es fácil de utilizar tanto para escribir código como para volcar dicho código sobre el microprocesador. Además, es multiplataforma y el lenguaje de programación de Arduino está basado en C++.

Figura 16 - Arduino IDE

3.3 Solución propuesta

Una vez presentada la solución *hardware* escogida, se procede a comentar la solución *software*. Como se ha comentado anteriormente, el producto presentado en esta memoria es un sistema de riego inteligente para jardines verticales. Con el término inteligente se hace referencia a un sistema que, dada una información, toma decisiones por sí solo basándose en unos parámetros. Con el fin de desarrollar un sistema inteligente de riego, como se ha comentado anteriormente, se ha diseñado un programador que recibe datos de los sensores, controla los actuadores y en dicho programador se ha desarrollado de una programación la cual hace posible el monitoreo, el control y permite al usuario personalizar la configuración a su jardín vertical. El sistema consta de sensores de humedad de suelo, un sensor de temperatura y válvulas de agua. Las válvulas de agua pueden ser controladas por el usuario tanto en modo manual, ya que el usuario puede activar y desactivar la válvula de agua de cada zona cuando lo desee, y en modo inteligente, dado que el usuario puede establecer el rango de humedad de suelo entre el que debería fluir la humedad de suelo de una zona en concreto. Por ejemplo, si en la zona 1 se ha plantado un tipo de planta que necesita una humedad de suelo entre un 50% y un 60%, se puede establecer como rango mínimo 50% y como rango máximo 60%, y de esta forma cuando la humedad de suelo sea inferior a 50%, se activará la válvula de agua regando así las plantas de dicha zona, hasta que se alcance una humedad de suelo de un 60%, donde se cerrará la válvula de agua.

Cabe destacar que, en el modo inteligente, se ha definido una restricción para que el usuario no pueda establecer humedades de suelo incoherentes. Se ha establecido que el nivel mínimo de humedad de suelo no pueda ser mayor o igual al nivel máximo de humedad de suelo especificado.

Tanto la visualización de las humedades de suelo de cada zona y la temperatura externa, así como el control de las válvulas de agua ya sea en modo manual o modo inteligente, se puede realizar mediante una página web en la cual el usuario deberá iniciar sesión, y posteriormente podrá visualizar el estado de su jardín vertical y configurarlo como desee. Además, podrá modificar la contraseña de acceso, así como cerrar sesión.

3.4 Presupuesto

En este apartado se muestra la valoración económica del producto desarrollado. En la Tabla 1 se muestran los gastos en hardware y los gastos en mano de obra. Cabe destacar que la unidad monetaria que se utiliza es el Euro (€) y que la unidad de tiempo del desarrollo del proyecto que se utiliza es la hora (h.).

	Coste unitario	**Cantidad**	**Total**
Sensor SoilWatch 10 (5m.) [38]	28,72	4	114,88
Sensor de temperatura 103AT-11 [39]	2,99	1	2,99
Relé PCN-105D3MHZ [40]	2,46	4	9,84
Limitador 5V [41]	3,38	1	3,38
Válvula de agua [9]	11	4	44
Wemos D1 mini pro [42]	7,04	1	7,04
Arduino Nano [43]	4,06	1	4,06
Manufacturación PCB [34]	-	5	20
Servidor AWS	0,019/hora	720h(1mes)	13,68
Mano de obra	40	5	200
	SUBTOTAL		416,39
	I.V.A	21%	
		TOTAL	503,8

Tabla 1 - Presupuesto

Como se puede observar en la Tabla 1, se ha destinado un presupuesto para el área Investigación y Desarrollo (I+D) dado que se han invertido 580 horas en investigar, diseñar, desarrollar y realizar pruebas para obtener dicho producto. Cabe destacar que en I+D no se tienen en cuenta las horas dedicadas al montaje hardware debido a que el montaje del sistema en un jardín vertical se cobrará por unidad. Por lo que el coste que conlleva las horas invertidas en I+D es de 23.200 €. Suponiendo que se realicen una venta mínima de 200 sistemas de riego inteligente, el coste de I+D se dividiría entre la cantidad de sistemas que se pretenden vender, por lo que el coste de I+D por cada sistema sería de 116 €.

Finalmente, el coste de adquirir dicho sistema e implantarlo en un jardín vertical constaría de tres bloques, por una parte, se encuentra el coste de I+D, que son 116 €, por otra parte, se encuentra el montaje del sistema en el jardín vertical, y por último el coste del material hardware comentado, por lo que el coste final aproximado sería de 532 € euros sin I.V.A, y con I.V.A constaría 643 €.

4 Diseño de la solución

4.1 Arquitectura del Sistema

En este apartado se expone la arquitectura del sistema tanto en el ámbito *software* como en el ámbito *hardware*.

4.1.1 Arquitectura software

En primer lugar, hay que comentar que en los apartados en los que se comentan aspectos del *software* desarrollado en este proyecto se estructuran en cuatro subapartados para su mejor comprensión debido a que el sistema desarrollado consta de cuatro partes claramente diferenciadas que se comunican entre sí para formar el sistema. Los cuatro subapartados comentados son los siguientes: el programador, la página embebida, la página web y el *backend*. Cabe destacar que el programador desarrollado está compuesto por el módulo wifi Wemos y Arduino, por lo que en este subapartado se comenta aspectos del *software* tanto para el Arduino como para el Wemos.
Respecto a la arquitectura *software* de este proyecto, a continuación, se va a explicar de qué trata cada subapartado y posteriormente se comentará la relación entre cada subsistema explicando así la arquitectura del *software* desarrollada.

4.1.1.1 Programador

En este subapartado se comentan las diferentes responsabilidades entre los diferentes módulos y cómo se comunican entre ellos para realizar diferentes tareas.

Arduino

El módulo Arduino es el que más pines digitales tiene de entradas y salidas y el único que tiene entradas analógicas de 3.3v o 5v. Por esas razones, la placa Arduino es la encargada de abrir y cerrar las válvulas de agua y de recoger toda la información de los sensores. Arduino tiene la suficiente memoria ROM para poder guardar las diferentes configuraciones de riego dependiendo de los parámetros de humedad del suelo.

Arduino está en un estado pasivo esperando recibir peticiones del módulo Wemos para enviar la información de los relés y sensores, recibir nueva configuración de riego dependiendo de la humedad del suelo o peticiones de abrir y cerrar directamente los relés.

Además, este módulo una vez configuradas las zonas de riego dependiendo de la humedad del suelo para regar, funciona por si solo sin depender del módulo Wemos para funcionar, por lo que, si por alguna razón el módulo Wemos dejara de funcionar, Arduino seguiría funcionando sin poder configurar o modificar los parámetros de riego.

Wemos

El módulo Wemos es el “cerebro” del programador con acceso a internet para poder controlar el programador, para poder acceder a él necesita actuar como un pequeño servidor web capaz de recibir peticiones HTTP, para poder controlarlo mediante una web embebida directamente sin necesidad de internet, esta página embebida, también llamada página de configuración, se utiliza para poder configurar el programador ya sea para conectarlo una red wifi o para registrar el programador dentro del sistema y poder vincular dicho programador a un usuario.

Wemos es el encargado de comunicarse con Arduino para que tenga la información suficiente para poder funcionar, como son los parámetros de humedad del suelo para activar/desactivar automáticamente una válvula de agua.

Además, Wemos es el encargado de suscribirse al servidor *backend* mediante *Websockets* para recibir o enviar peticiones a la página web principal del usuario mediante la cual podrá controlar el programador.

4.1.1.2 Página embebida

Por un parte, se encuentra la aplicación web necesaria para poder conectar el programador a la red wifi de la nueva propiedad donde se va a ubicar, ya sea una casa, un local o espacio abierto que reciba señal de wifi, así como iniciar sesión un usuario ya registrado.

Sin esta página web embebida no se podría configurar el programador de ningún modo para poder conectarlo a las redes wifi. Esta página es la conexión directa con el

usuario sin necesidad de internet, si hubiera cualquier problema con la configuración del programador se podría acceder directamente a él mediante la página embebida.

4.1.1.3 Página web

Por otro lado, se encuentra la aplicación web donde el usuario va a poder iniciar sesión, ver el estado de las zonas del jardín, como por ejemplo la humedad de suelo de cada zona, y modificar dicho estado, como puede ser el caso de activar una válvula de agua de una zona.

Está página web estaría disponible desde cualquier lugar, gracias a su ubicación en la nube y en cualquier momento, por lo que se podría controlar el programador remotamente.

4.1.1.4 Backend

El *backend* es el programa que está ejecutándose en un servidor en la nube que se encarga de almacenar la información en la base de datos y de proveer dicha información tanto al programador como a la página web en tiempo real mediante *Websockets*.

Arquitectura general del sistema software

Por lo que la relación entre los subsistemas comentados sería la que se muestra en la Figura 17 y se explica a continuación:

El *backend* está conectado tanto con la página web como al programador, y como se ha comentado anteriormente, en el programador hay embebida una página web la cual también utiliza la información que recibe el programador del *backend*.
El programador se convierte en un servidor devolviendo la página web embebida, por lo que desde el navegador que muestra dicha página web embebida, se puede realizar peticiones al servidor del programador mediante *Websockets* y peticiones HTTP.
Por otra parte se encuentra el *backend*, el cual interactúa con el programador, teniendo en cuenta que el programador actúa como servidor, e interactúa con la página web. Se puede considerar como un intermediario que transmite los datos desde la página web con la que interactúa el usuario hasta el programador que gestiona el sistema de riego y viceversa, de forma que si el usuario modifica desde la

página web el estado de una válvula, dicha petición se transmite al *backend*, el *backend* modifica el estado de dicha válvula y el programador recibe ese cambio, por lo que actúa sobre la válvula de agua. Gracias a esta arquitectura el usuario puede visualizar y modificar información de su programador desde cualquier parte ya que la información se encuentra en la nube en todo momento. Cabe destacar que en el caso en el que fallase el *backend* o la conexión del programador a internet, el programador seguiría funcionando de forma normal pero no se podría actuar a través de internet sobre las válvulas de agua, como se ha comentado anteriormente.

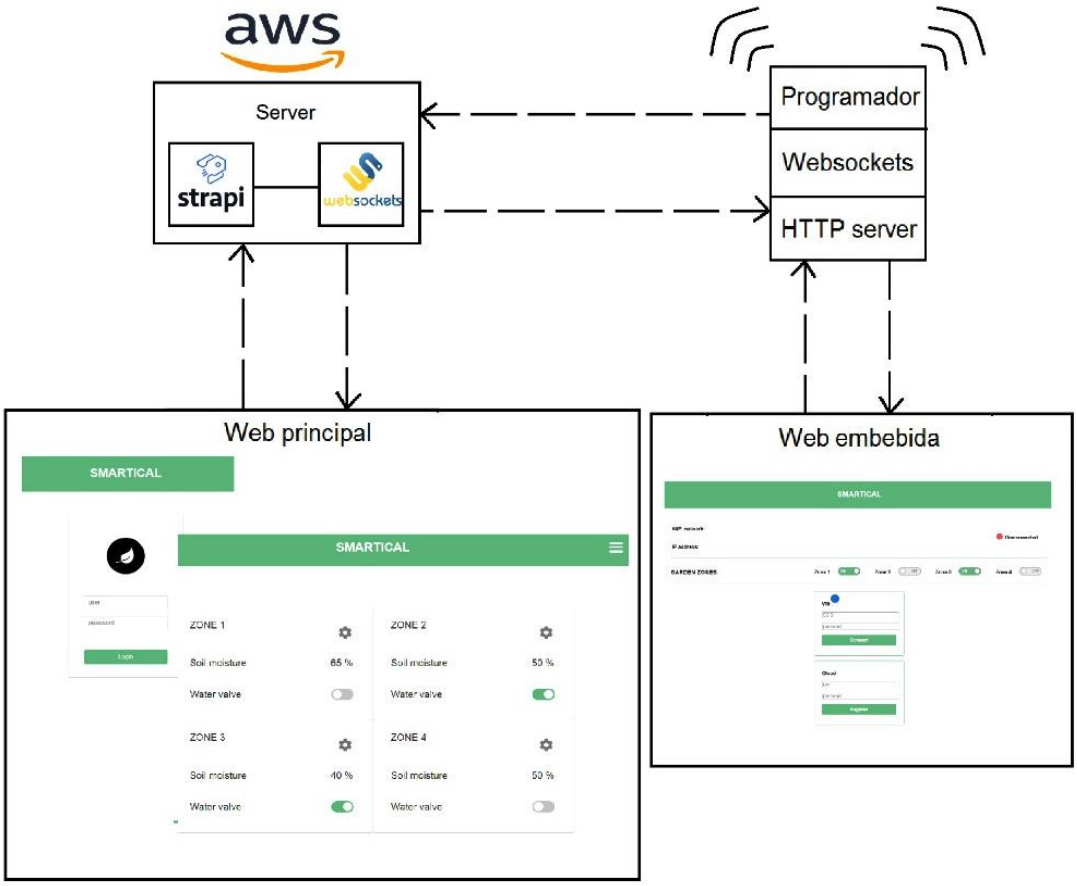


Figura 17 - Diagrama arquitectura general del sistema software

4.1.2 Arquitectura hardware

En cuanto a la arquitectura hardware, la cual se puede observar en la Figura 18, está compuesta por un sensor de temperatura, por sensores de humedad de suelo, válvulas de agua y el sistema central, también llamado programador.

Figura 18 - Arquitectura hardware

4.2 Diseño detallado

En esta sección se expone el diseño detallado de la parte *software*, es decir, los flujos de ejecución del programa y diagramas de secuencia UML, y respecto a la parte de hardware, se explica el diseño de la placa en EAGLE y la conexión general de cada componente electrónico utilizado con el PCB creado.

4.2.1 Diagramas de flujo

A continuación, se muestra el diagrama de flujo de la Figura 19 de la activación/desactivación de la válvula de agua de una zona dependiendo de la humedad de suelo de la que disponga dicha zona. Gracias a los diagramas de flujo se puede observar, gráficamente, el flujo de trabajo que se realiza cuando se produce un cambio en la humedad de suelo de cada zona. Como se ha comentado anteriormente, el usuario especifica dos puntos de consigna para cada zona, que es el valor de humedad de suelo que debe tener una zona como mínimo para que se active la válvula de agua o como máximo para que se desactive la válvula de agua. Hay que recordar que el usuario puede controlar las válvulas de agua en todo momento, así como modificar los puntos de consigna. Cabe destacar que la abreviatura “V.A” hace referencia a Válvula de Agua.

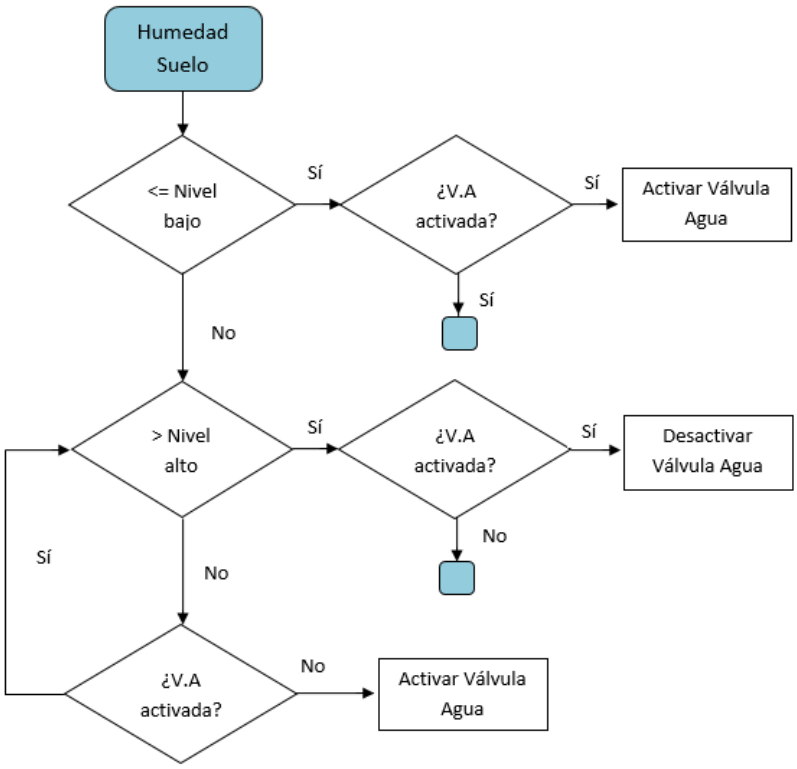


Figura 19 - Diagrama de flujo válvula de agua

4.2.2 Diagramas de secuencia UML

Un diagrama de secuencia es un tipo de diagrama que permite modelar la interacción existente entre los objetos en un sistema. En este apartado se ha van a explicar las llamadas realizadas entre los diferentes objetos que intervienen en este proyecto, como son, el cliente, el servidor, el programador, la web y el *router* wifi.

I. Conexión a la nube

El usuario introduce la dirección IP 192.168.4.1 y el servidor le devuelve el HTML al navegador mostrando éste la página web principal donde el usuario debe iniciar sesión. Tras introducir el usuario y la contraseña, el servidor le manda dicha información al programador, y éste le devuelve el *token* el cual se almacena en el servidor. Finalmente, el servidor inicia la conexión con *Websockets* enviándole el *token* para autenticarlo e identificar el usuario para devolverle la información del programador correspondiente.

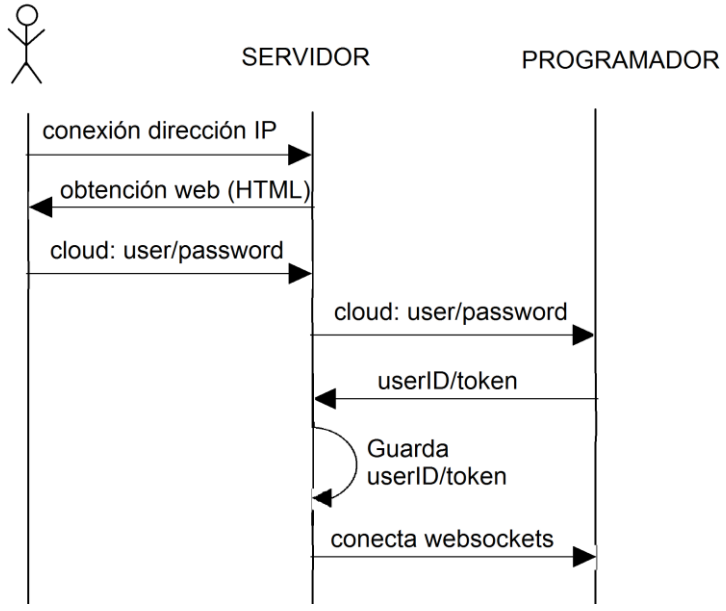


Figura 20 - Diagrama de secuencia conexión a la nube

II. Conexión a la web

La interacción del usuario con el programador se realiza mediante la página web y el servidor. En primer lugar, el usuario introduce la URL en el navegador, y el usuario debe introducir su correo electrónico y contraseña, si es correcto, se le muestra la información del programador con el que está vinculado. Posteriormente si el usuario desea realizar alguna acción como, por ejemplo, cambiar el estado de un relé, lo realiza mediante la página web, ésta le envía la orden mediante *Websockets* al servidor, y el servidor se la envía al programador, el cual finalmente realiza la acción de apagar dicho relé.

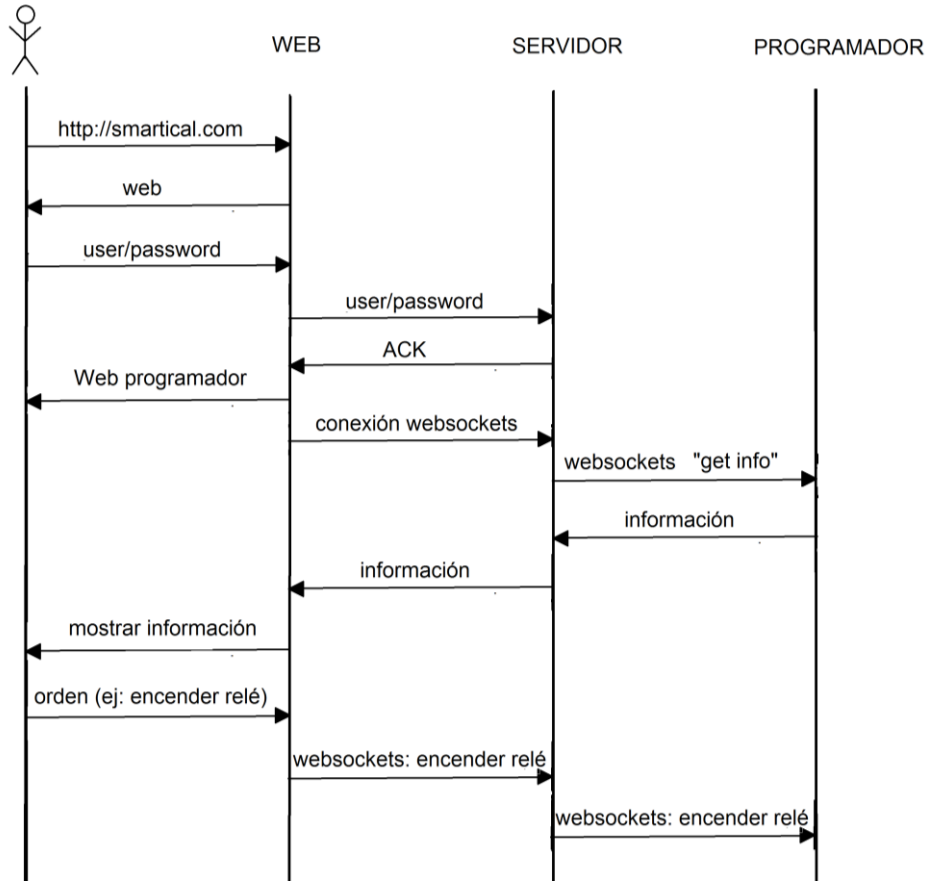


Figura 21 - Diagrama de secuencia conexión a la web

III. Conexión del programador al wifi

Para que el programador pueda conectarse a una red Wifi se debe de hacer la configuración previa especificada en la Figura 22. En primer lugar, se conecta el programador a la luz, y como no se ha configurado previamente, el programador actúa como un punto de acceso ya que no conoce ninguna red Wifi y crea la suya propia con el nombre del programador. De forma que el usuario se debe conectar a la red wifi del programador. Una vez conectado el programador siempre tiene la misma IP que es 192.168.4.1, por lo que se debe introducir en el navegador como URL dicha dirección IP, y aparece la página web de configuración. En dicha página web se puede introducir la nueva red Wifi y la contraseña. Una vez el programador tiene conexión a internet se destruye el punto de acceso previamente creado y el programador ya puede enviar y recibir datos del servidor, por lo que el usuario ya puede acceder a la página web principal que muestra la información de su programador.

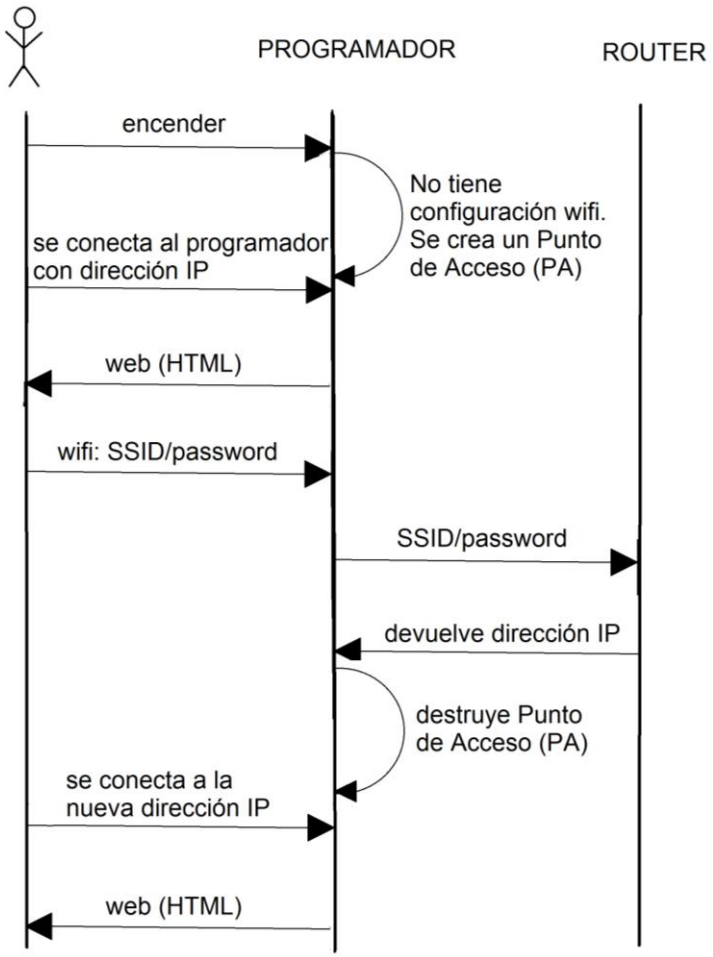


Figura 22 - Diagrama de secuencia conexión del programador a una red wifi

4.2.3 Explicación del diseño del PCB

En este apartado se van a exponer algunos puntos del diseño del PCB. El sistema inteligente de riego está compuesto por sensores de humedad de suelo, un sensor de temperatura y válvulas de agua. Para conectar todos estos dispositivos es necesario una placa, como es el Arduino, que pueda leer datos de los sensores, analizarlos y actuar sobre las válvulas correspondientes.

Debido a que las válvulas de agua utilizadas en el mercado para jardines verticales trabajan a 24 VAC, se utiliza un transformador que pasa de 220 VAC a 24 VAC, para alimentar las válvulas de agua. Por otro lado, los dispositivos utilizados en el proyecto tienen una tensión de alimentación de 5 VDC, como es el caso de los sensores de humedad y temperatura, la placa Arduino, la placa Wifi Wemos y los relés que controlan el funcionamiento de las válvulas de agua. Por lo que es necesario transformar 24 VAC a 5 VDC.

Transformación de 24 VAC a 24 VDC [12]

En primer lugar, se debe transformar 24 VAC a 24 VDC con un puente de diodos o también llamado puente rectificador. Un puente rectificador está compuesto por 4 diodos rectificadores. Un diodo es un componente electrónico que permite el flujo de electricidad en un solo sentido, y en el caso en el que se trate de hacer en sentido contrario, la corriente no fluye. Los diodos rectificadores son diodos utilizados para rectificar el voltaje de corriente alterna a una corriente continua. El puente rectificador de onda completa es un circuito que conduce la corriente con una polaridad directa y no conduce la corriente en la polaridad inversa.

En la Figura 23 a la parte izquierda se muestra la onda de la corriente alterna y la parte derecha se muestra la onda de la corriente continua, que es la que se desea obtener.

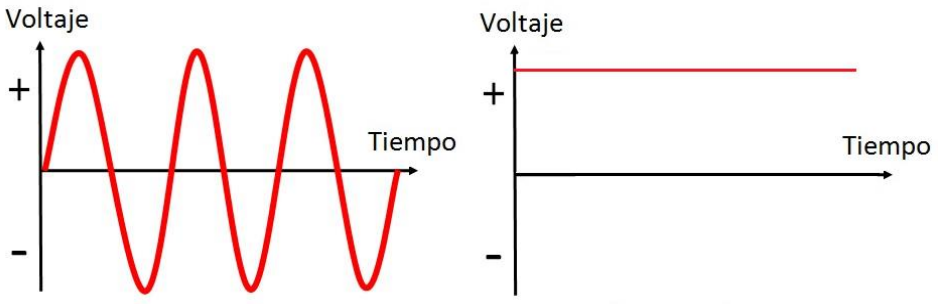


Figura 23 - Gráficas antes y después de la transformación de 24 VAC a 24 VDC

Con el puente de diodos, durante el semiciclo positivo, como puede observarse en la Figura 24, los diodos D1 y D3 conducen la corriente, por lo que en la resistencia de la carga se recibe un semiciclo positivo, lo que se observa en la imagen como la onda verde.

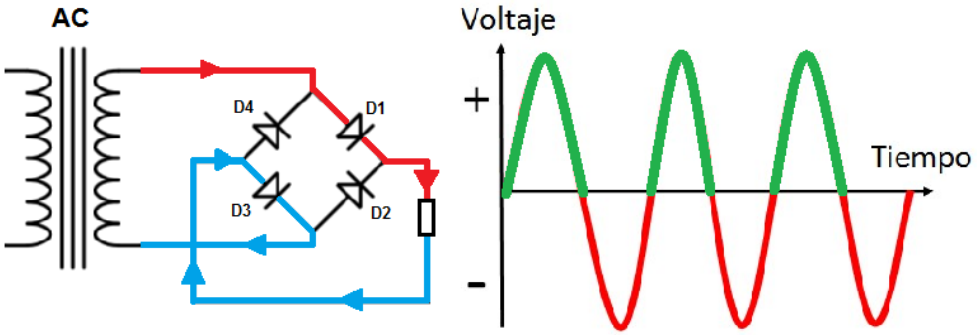


Figura 24 - Gráfica transformación de VAC a VDC semiciclo positivo

Mientras que, en el semiciclo negativo, como puede observarse en la Figura 25, los diodos D2 y D4 conducen la corriente, por lo que en la resistencia de la carga se recibe de nuevo un semiciclo positivo, lo que se observa en la imagen como la onda verde.

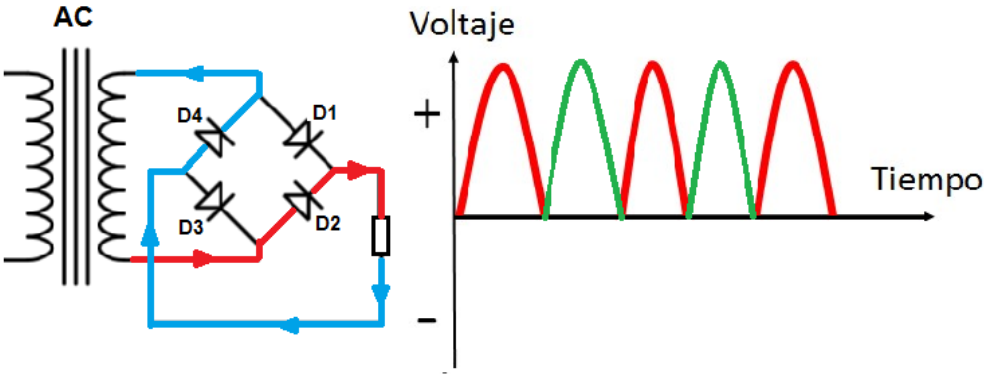


Figura 25 - Gráfica transformación de VAC a VDC semiciclo negativo

De esta manera la resistencia de la carga siempre recibe valores positivos, como se muestra en la Figura 26, independientemente de que el semiciclo sea positivo o negativo.

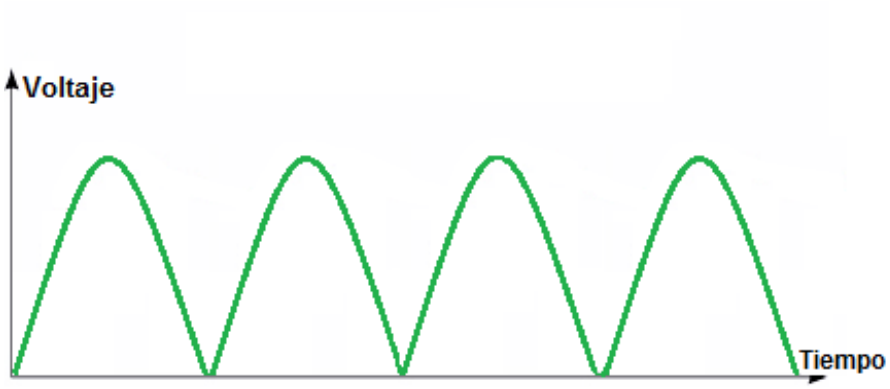


Figura 26 - Gráfica de voltaje resultante tras el puente de diodos

Sin embargo, en la onda resultante se observa el efecto rizado en el que hay una variación de voltaje, por lo que se necesita un condensador que filtre dicha ondulación y, en el momento de bajada de voltaje, produzca una realimentación. Como se puede observar en la Figura 27, finalmente el circuito necesario está compuesto por un puente rectificador de diodos, concretamente son cuatro diodos rectificadores, y uno o más condensadores.

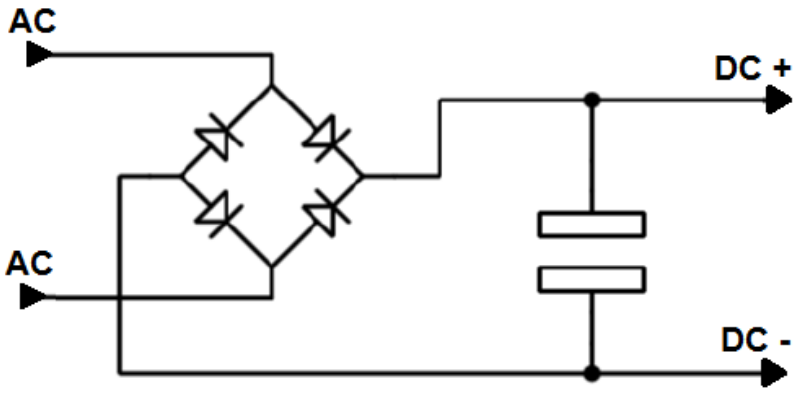


Figura 27 - Circuito transformación VAC a VDC

En este circuito, al circular la corriente alterna por el puente de diodos, como se ha explicado anteriormente, dos diodos se encargan de transformar los ciclos negativos en ciclos positivos y otros dos diodos se encargan de mantener los ciclos positivos. Posteriormente el condensador se encarga de mantener la tensión en el momento en el que la fuente de corriente alterna cambia su polaridad.

Concretamente, los condensadores se utilizan para evitar, dentro de su capacidad, que el voltaje cambie de valor, es decir, durante el periodo de carga, almacena energía y durante el periodo de descarga proporciona esa misma energía. Por lo que con el puente de diodos se logra rectificar la señal y dicha señal es filtrada con un condensador, de esta manera se consigue convertir la corriente alterna en continua, obteniendo una onda filtrada como la que se muestra en la Figura 28.

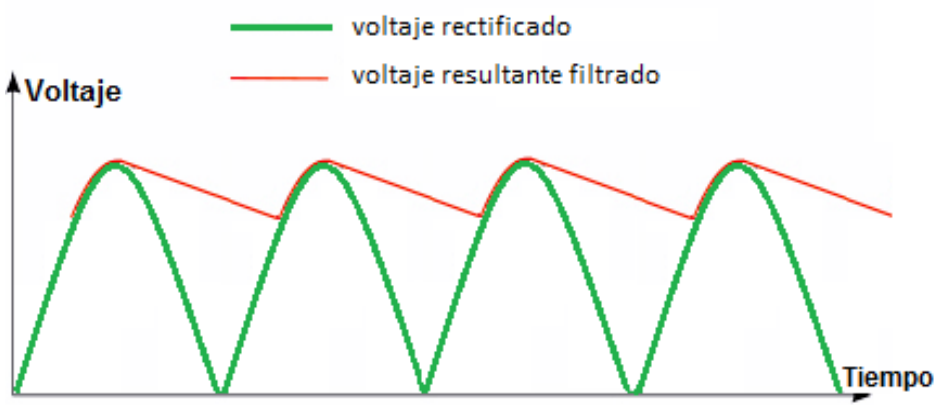


Figura 28 - Gráfica de voltaje resultante tras el filtrado

Transformación de 24 VDC a 5 VDC [12]

Una vez se obtienen 24 VDC, se debe transformar a 5 VDC. Esto se puede realizar con un circuito como el que se puede observar en la Figura 29, compuesto por los siguientes componentes:

- Un limitador o regulador de tensión LM2576 a 5 V
- Un diodo Schottky que puede conmutar a alta velocidad y permite rectificar señales de alta frecuencia
- Un inductor de 150uH que almacena energía a través de campos magnéticos y produce energía a otro elemento
- Un condensador de 1000uF para estabilizar la señal

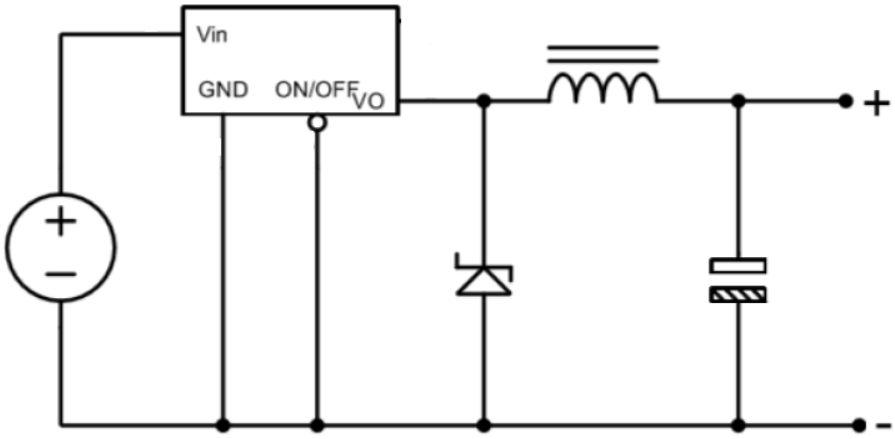


Figura 29 - Circuito de transformación 24 VDC a 5 VDC

Una vez explicada la parte de transformación de 24VAC a 5VDC, se procede a explicar la alimentación a las válvulas de agua a través de los relés.

Un relé es un dispositivo electromagnético compuesto por una bobina que cuando circula corriente por ella, se crea un campo magnético que produce que se cierre el contacto. De esta forma se deja o no pasar la corriente, dependiendo de si el contacto está cerrado o abierto. Por lo que el relé permite controlar grandes cargas con una pequeña señal sin que se caliente, por lo que no es necesario un disipador.

Las válvulas son accionadas mediante relés. Cuando se quiera abrir una válvula, el contacto del relé correspondiente se cerrará, permitiendo la circulación de la corriente, 24VAC, y activando la válvula de agua. Como puede observarse en la Figura 30, se debe incorporar un diodo en paralelo con la bobina, llamado diodo antiparalelo, que permite la descarga de campo magnético de la bobina a través de él, protegiendo el resto del circuito contra sobretensiones.

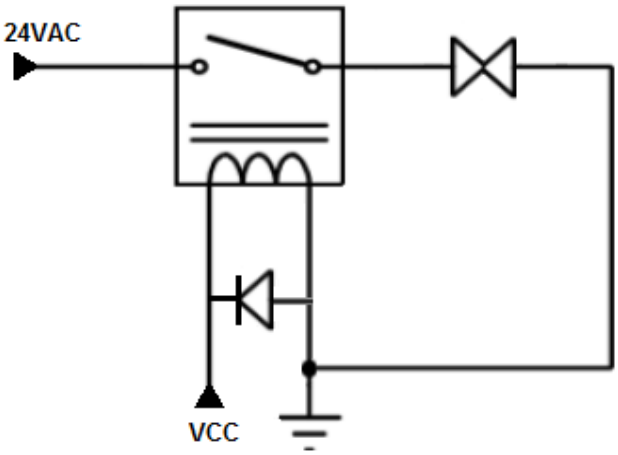


Figura 30 - Circuito de alimentación a las válvulas de agua mediante relés

4.2.4 Diseño de la placa PCB en EAGLE

Una vez se han realizado todas las conexiones de los dispositivos entre sí y funcionan correctamente, se procede a diseñar la placa PCB en EAGLE [13]. El diseño de la placa PCB en Eagle consta de dos fases; la creación del esquemático y la colocación de cada componente electrónico en la placa, llamado emplazamiento o *placement*.

4.2.4.1 Creación del esquemático

La creación del esquemático se realiza en la ventana de EAGLE llamada “Schematic”, en esta ventana se puede seleccionar los componentes electrónicos que se desean poner y crear conexiones lógicas entre dichos componentes. Cabe destacar que se deben importar librerías de algunos componentes que se desean incorporar como es el caso de las placas Arduino Nano y Wemos D1 mini pro, así como varios componentes como es el caso del limitador LM2576, los relés, etc.

El esquemático de este proyecto consta de varias áreas;

- El área de las bornas de potencia donde irán conectados los cables de potencia de los sensores de humedad de suelo y de las válvulas.
- El área en la que se encuentra todo lo relacionado con la fuente de alimentación y su transformación a 5VDC.
- El área de los relés.
- El área de las placas Wemos y Arduino.

- El área del sensor de temperatura.
- El área de los leds.
- El área en que se juntan las entradas y salidas no utilizadas de la placa Wemos, por una parte, y de la placa Arduino por otra parte.

Bornas de tornillos

El circuito deberá incorporar una serie de bornas de tornillos a los que irán conectados los cables de potencia de los sensores de humedad de suelo y de las válvulas. En la Figura 31 se muestra cómo se conectarán los cables a las bornas en el caso en el que se usasen válvulas de 24VAC.

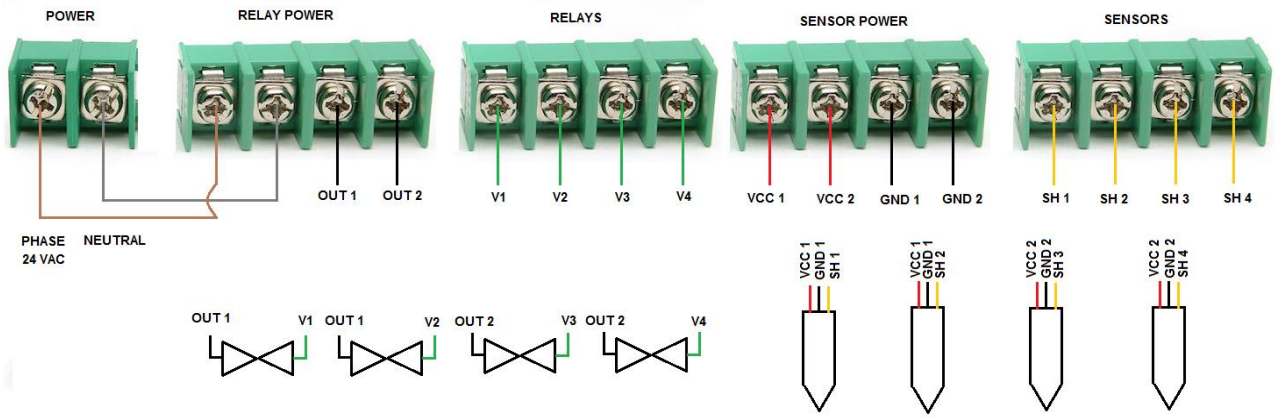


Figura 31 - Conexiones de bornas de tornillo con válvulas de 24VAC

Sin embargo, como se ha comentado anteriormente, el circuito se ha diseñado de forma que puedan utilizarse válvulas de voltaje diferente a 24VAC. Por lo que la conexión de válvulas de diferente voltaje a 24VAC sería la que se muestra en la Figura 32.

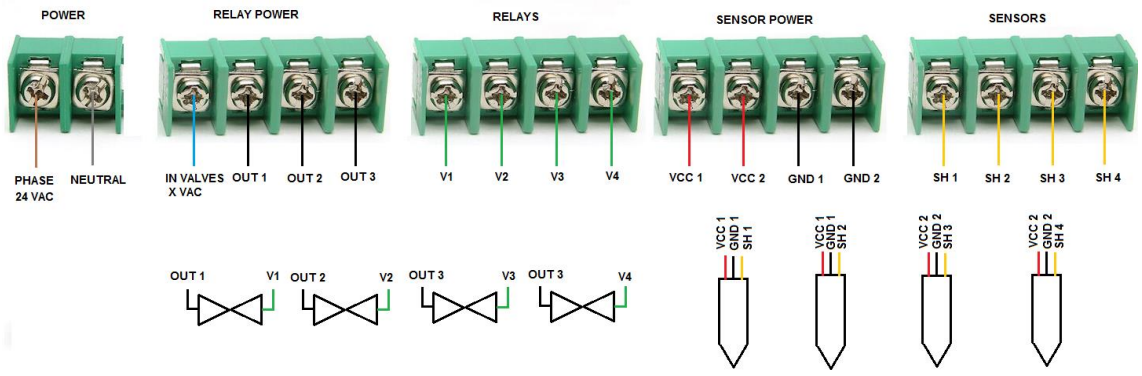


Figura 32 - Conexiones de bornas de tornillo con válvulas de voltaje diferente a 24VAC

Mientras que en la Figura 33 se muestra cómo se representa dicha conexión en EAGLE. En EAGLE, estos conectores se representan como agujeros, en los cuales, una vez fabricada la placa, se soldarán las bornas. Como puede observarse la conexión entre componentes se ha realizado mediante etiquetado para que el esquemático sea más claro.

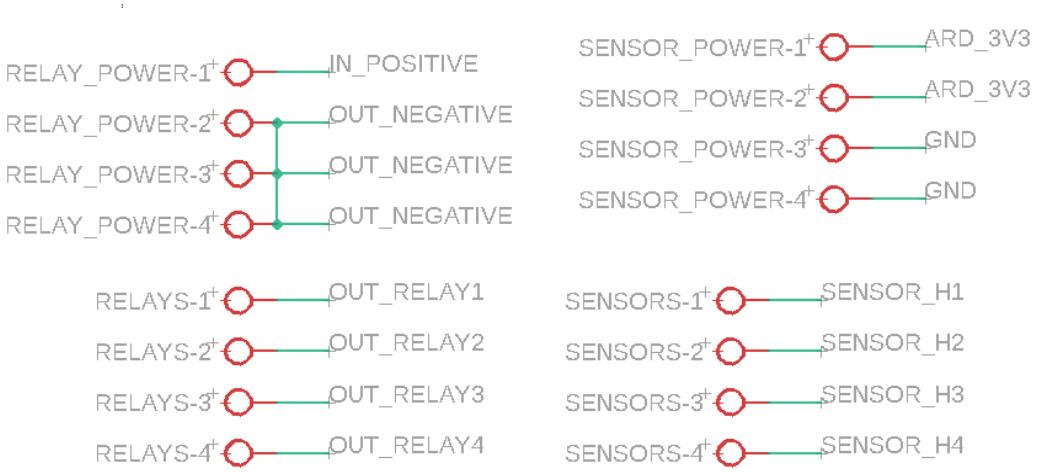


Figura 33 - Conexiones de bornas de tornillo en EAGLE

Fuente alimentación y transformación

Como se ha comentado anteriormente, en primer lugar, se transforma la corriente alterna a corriente continua mediante un puente de diodos o rectificador y un condensador para filtrar la señal, en este caso se han puesto dos condensadores de 1000uF en vez de uno de 2000uF porque el condensador de 2000uF es demasiado grande. Como se observa en el esquemático de la Figura 34, el puente de diodos está conectado, por una parte, a la línea de corriente (fase) y por una parte a la línea neutral (neutro).

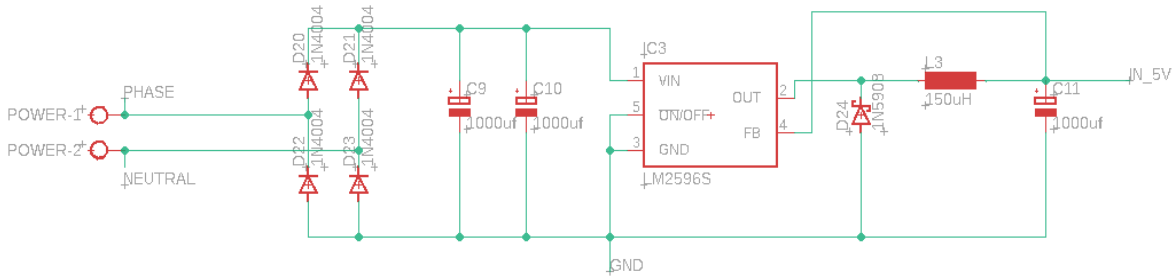


Figura 34 - Diseño circuito de fuente alimentación y transformación en EAGLE

Después de la etapa rectificadora, se procede a convertir 24 VDC a 5VDC, por lo que los condensadores se conectan al limitador de 5 voltios y éste a un diodo, un inductor

y un condensador, explicados previamente. De esta forma se obtienen 5VDC, que servirá de alimentación para los demás componentes.

Relés

Los relés y las conexiones de los relés se representan en EAGLE como se observa en la Figura 35. El pin 4 (COM) de los relés son conectados al terminal donde se conecta la línea positiva de la válvula de agua, mientras que en el pin 1 (COIL_1) se conecta la entrada del relé, mediante la cual el Arduino manda abrir o cerrar relé. En el pin 3 (NO) está conectado a la fase (24VAC) y el pin 2 (COIL_2) está conectado a GND (neutro).

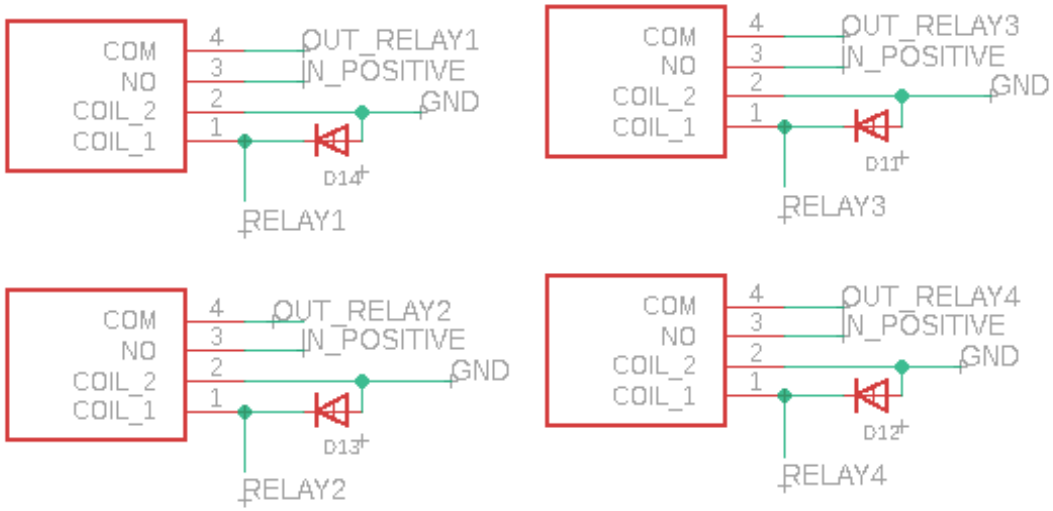


Figura 35 - Circuito de conexiones de relés en EAGLE

Conexión Wemos y Arduino

En la Figura 36 se muestra la representación de la placa Arduino Nano en EAGLE y se han etiquetado todas las conexiones con los demás dispositivos.

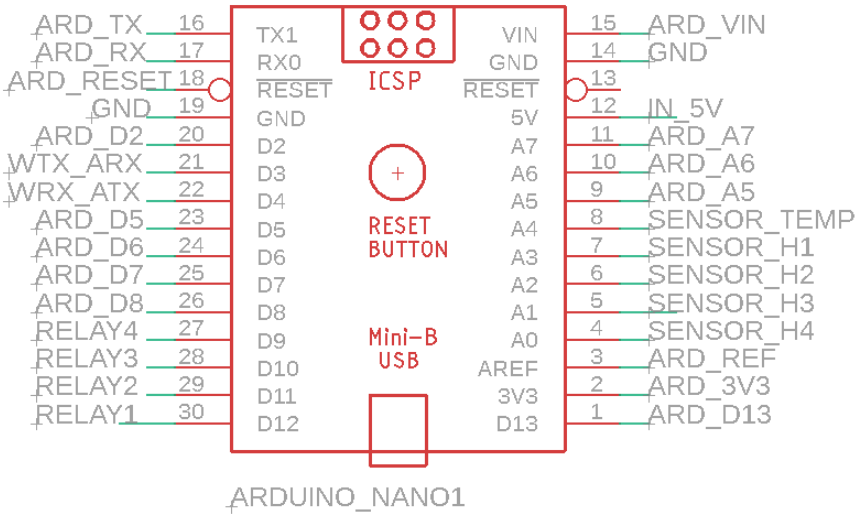


Figura 36 - Circuito de conexión de Arduino en EAGLE

Las entradas o salidas que no han sido utilizadas se han unido en un conector hembra, representado en la Figura 37, por si en un futuro se desean utilizar.

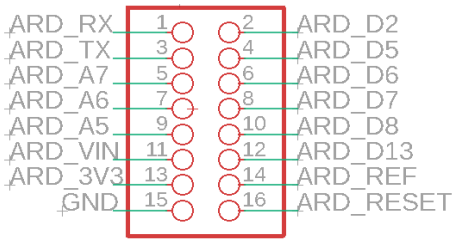


Figura 37 - Circuito de conexión de entradas no utilizadas de Arduino en EAGLE

Mientras que en la Figura 38 se muestra la representación de la placa Wemos D1 Mini Pro en EAGLE y se han etiquetado todas las conexiones con los demás dispositivos.

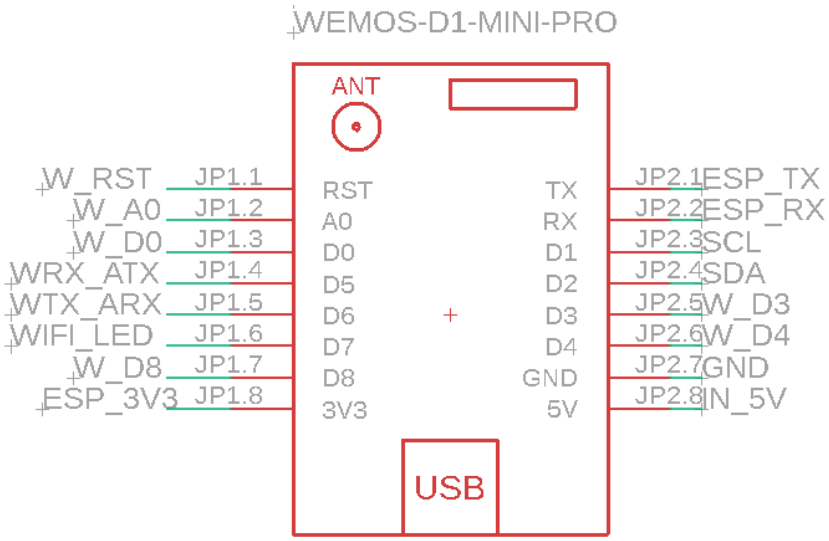


Figura 38 - Circuito de conexión de Wemos en EAGLE

Al igual que se ha realizado con las entradas o salidas que no han sido utilizadas en Arduino, se ha realizado lo mismo con las entradas o salidas que no se han utilizado, se han unido en un conector hembra, representado en la Figura 39, por si en un futuro se desean utilizar.

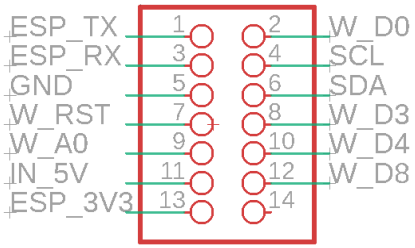


Figura 39 - Circuito de conexión de entradas no utilizadas de Wemos en EAGLE

Sensor de temperatura

Para la conexión del sensor de temperatura externa, como se puede observar en la Figura 40, se ha representado mediante una borna de tonillo de 2 vías. A demás se ha añadido una resistencia de 10K ya que el sensor de temperatura seleccionado la requiere.

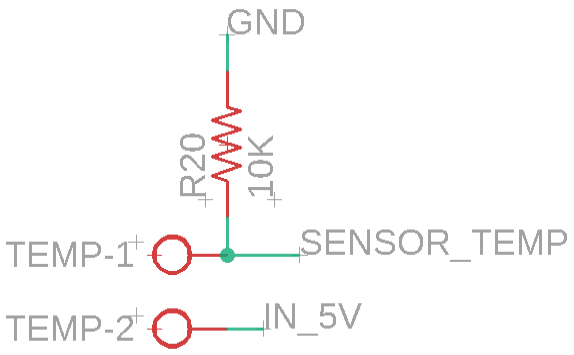


Figura 40 - Circuito de conexión sensor de temperatura en EAGLE

LEDs

Se ha decidido incorporar cinco leds en la placa como indicadores. Como se observa en la Figura 41, uno de los leds es para indicar cuando el wifi está conectado o desconectado, mientras que los cuatro leds restantes es para indicar si las válvulas están abiertas o cerradas, por lo que cada led está conectado a una válvula.

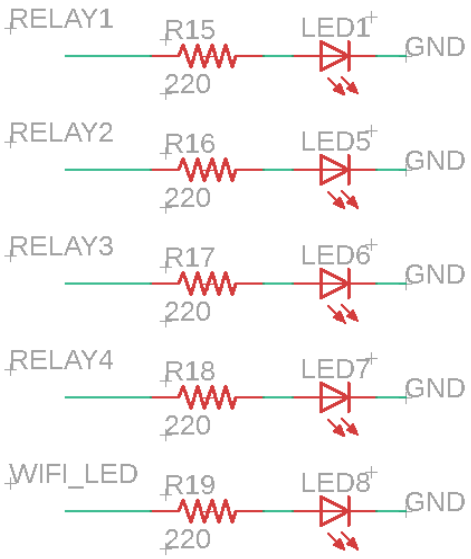


Figura 41 - Circuito de conexión de LEDs en EAGLE

4.2.4.2 Creación de la placa de circuito impreso

Una vez realizado el esquemático, se procede a realizar el *Layout*. Para ello en la ventana “Schematic”, se debe pulsar a un botón llamado “Generate/Switch to board” que se encuentra en la parte superior a la izquierda, como puede observarse en la Figura 42.

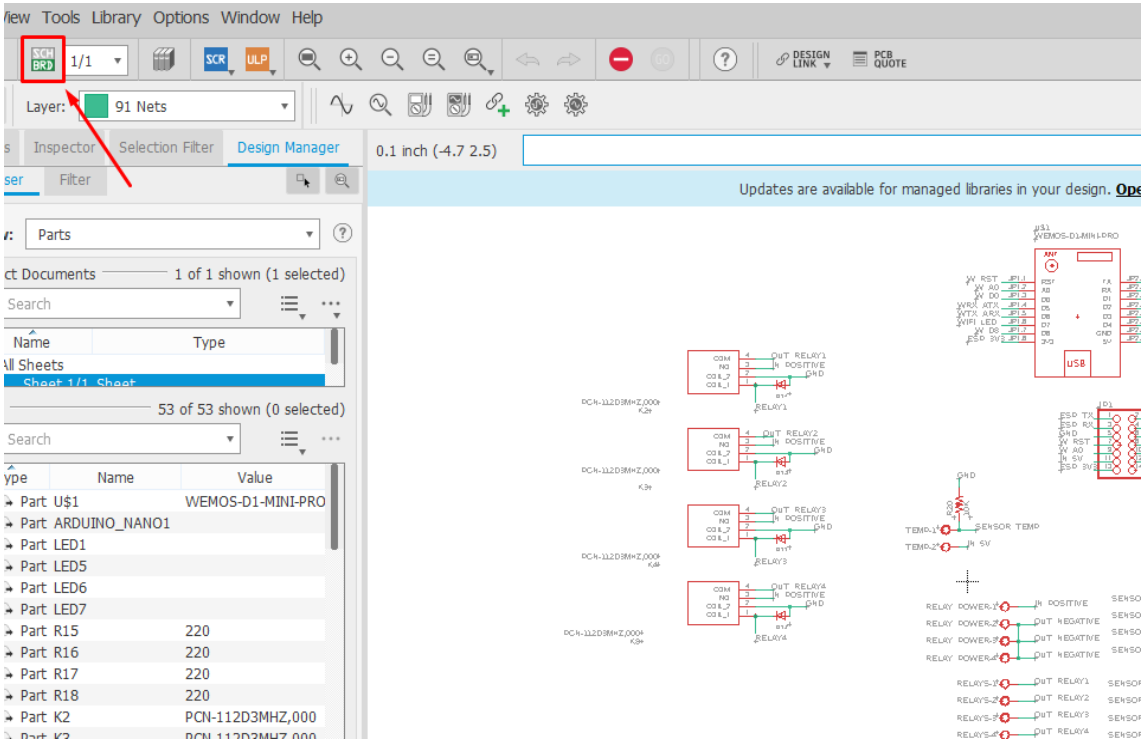


Figura 42 - Creación *layout* en EAGLE

Tras pulsar el botón “Generate/Switch to board” aparece una nueva ventana como se muestra en la Figura 43. A la izquierda se pueden observar los componentes del esquemático conectados mientras que a la derecha aparece un rectángulo que representa la placa. En primer lugar, se debe dimensionar la placa con las medidas deseadas, en este caso se va a realizar lo más pequeña posible, pero teniendo en cuenta que deben caber todos los componentes dentro.

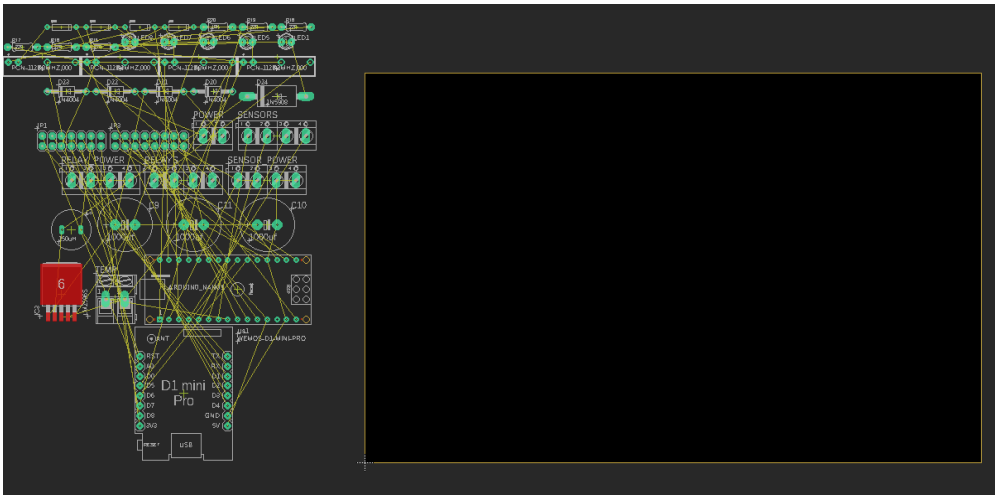


Figura 43 - Dimensionar placa en EAGLE antes de colocación componentes

Posteriormente se procede a colocar todos los componentes dentro del rectángulo. Para la colocación de los componentes se tiene en cuenta la proximidad entre los componentes conectados ya que la fase de enrutamiento será más sencilla si se realiza un buen *placement*. Cabe destacar que la colocación de los componentes que se van a manipular usualmente debe ir en los bordes de la placa, como es el caso de las bornas de tornillo. El posicionamiento final de todos los componentes se puede observar en la Figura 44.

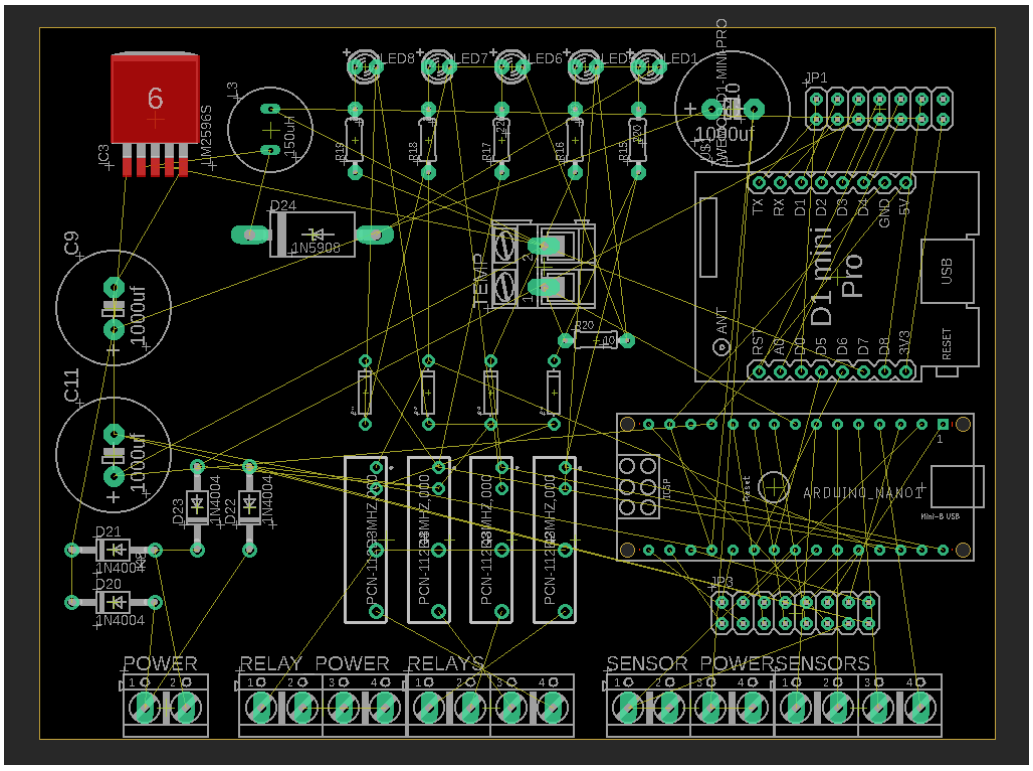


Figura 44 - Colocación componentes en la placa en EAGLE

A continuación, se realiza el enrutado, para ello se utiliza el icono mostrado en la Figura 45, que permite realizar el enrutado manualmente. El enrutado manual permite, entre otras cosas, seleccionar la capa donde se va a realizar la ruta, el ancho de la pista o el ángulo del cambio de dirección de la pista, el cual es aconsejable que no sea de 90º.

Figura 45 - Botón de enrutado en EAGLE

En primer lugar, se ha comenzado a realizar el enrutamiento de las pistas por las que pasan 24VAC. Como se muestra en la Figura 46, se han utilizado líneas más gruesas para representar las pistas con mayor voltaje o densidad de corriente.

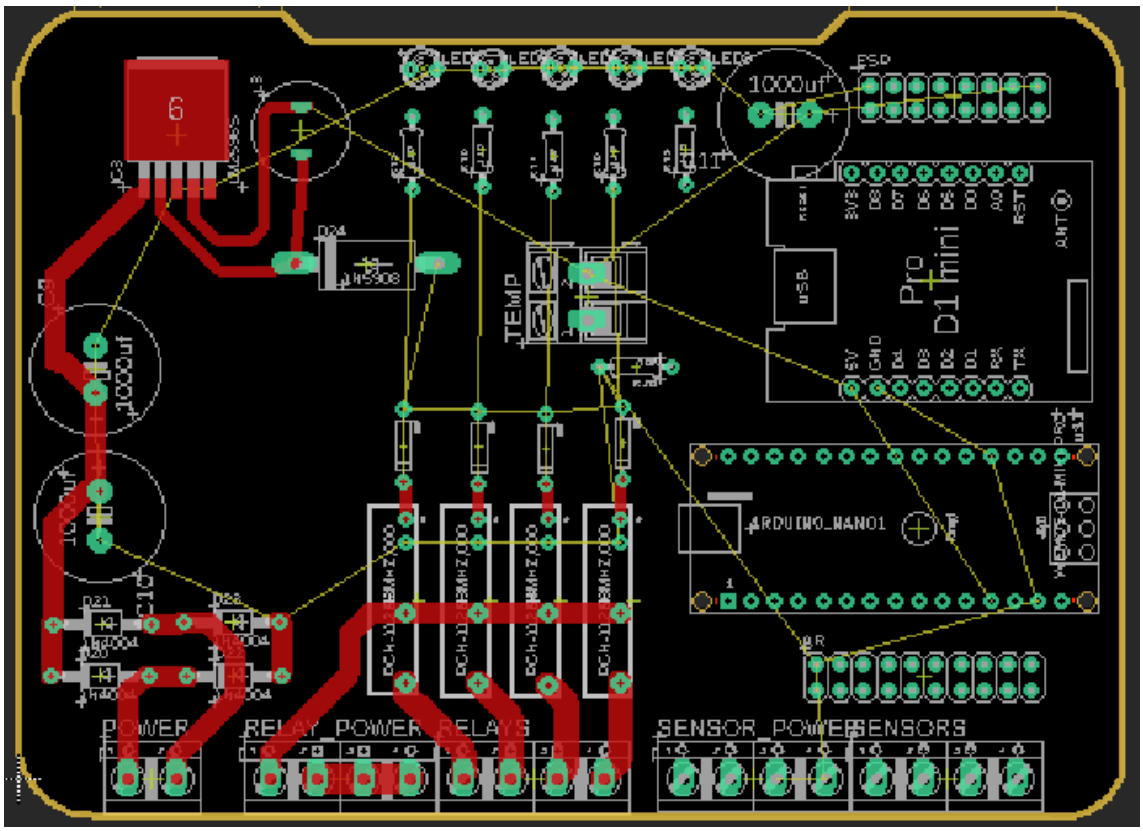


Figura 46 - Enrutado pistas 24VAC en EAGLE

Posteriormente, se realizan las demás conexiones, las pistas de color rojo representan las pistas que van en la capa “Top”, la cara delantera de la placa, y las pistan en color

azul son las pistas pertenecientes a la capa “Bottom”, que es la cara trasera de la placa. También se puede observar cómo, en algunas ocasiones, se han tenido que realizar “puentes” para realizar una conexión desde un pin de un componente a otro. Estos puentes consisten en cambiar a otra cara la pista cuando no sea posible continuar por la misma capa, y posteriormente cambiar a la capa inicial.

Como se observa en la Figura 47, también se han realizado los agujeros para atornillar la placa en las esquinas y se ha intentado dar un poco más de forma a la placa mediante el redondeo de las esquinas y el surco en la zona de los leds, estos dos últimos cambios son meramente estéticos.

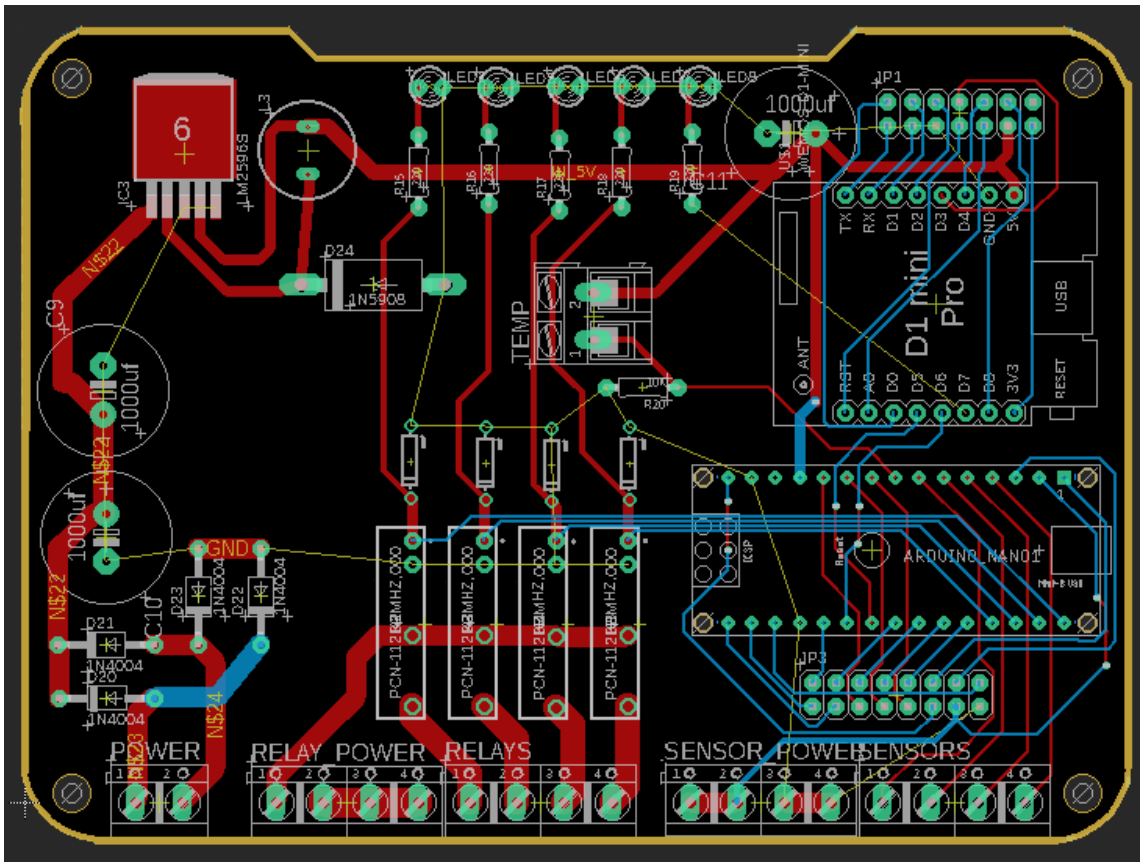


Figura 47 - Enrutado entre dispositivos en EAGLE

Tras realizar el enrutado se debe crear un plano de masa. El plano de masa es un plano de metal que está conectado a tierra y es recomendado realizarlo para no tener que diseñar rutas largas para conectar cada pin GND de cada componente a tierra y para aislar varias pistas entre sí.

En este proyecto se crean planos de masa en las capas utilizadas, en este caso en la capa “Bottom”, como se observa en las figuras 48 y 49 y en la capa “Top”, como se aprecia en las Figura 50. Cabe destacar que el plano de masa no debe estar en

contacto con el área donde hay alta tensión ya que son diferentes masas con diferentes tensiones, y podrían producirse cortocircuitos.

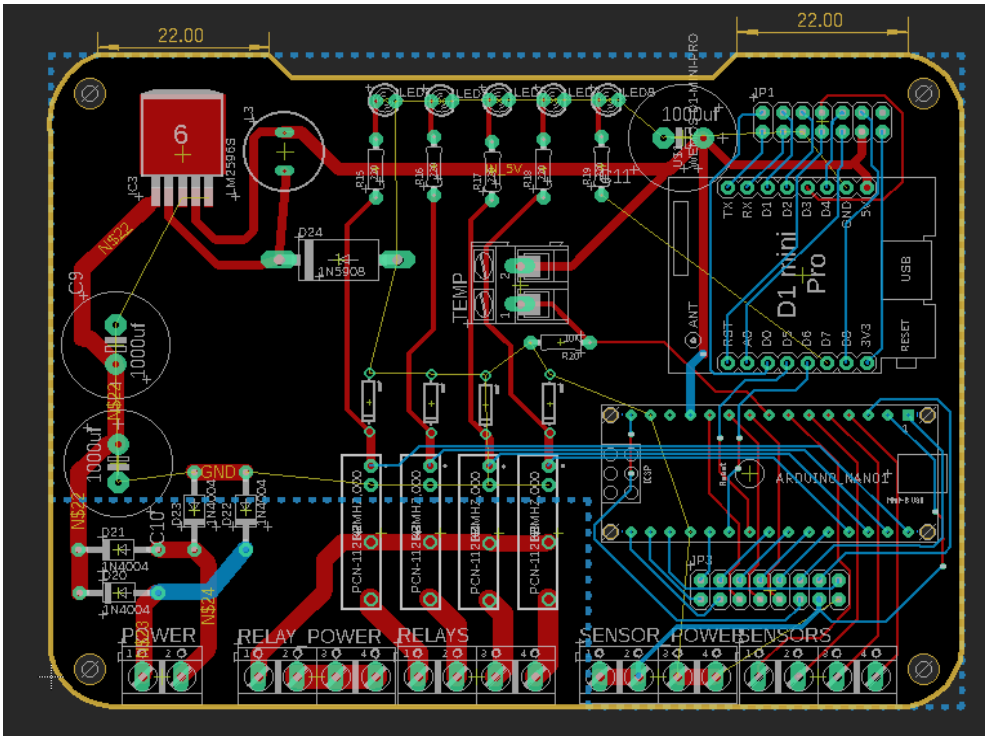


Figura 48 - Plano de masa capa Botton (I)

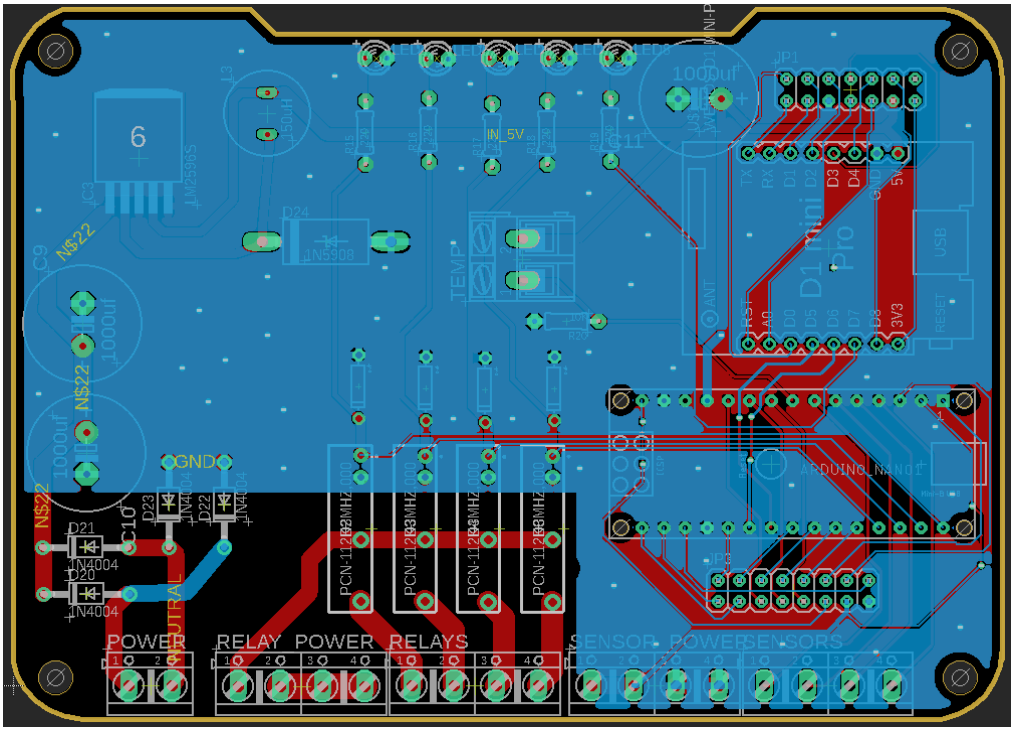


Figura 49 - Plano de masa capa Botton (II)

Además, también se deben realizar vías pasantes que son una especie de pequeños agujeros los cuales permiten conectar capas con el fin de unir áreas de cobre más grandes en diferentes capas, lo que mejora la disipación térmica y crea bucles de retorno más cortos.

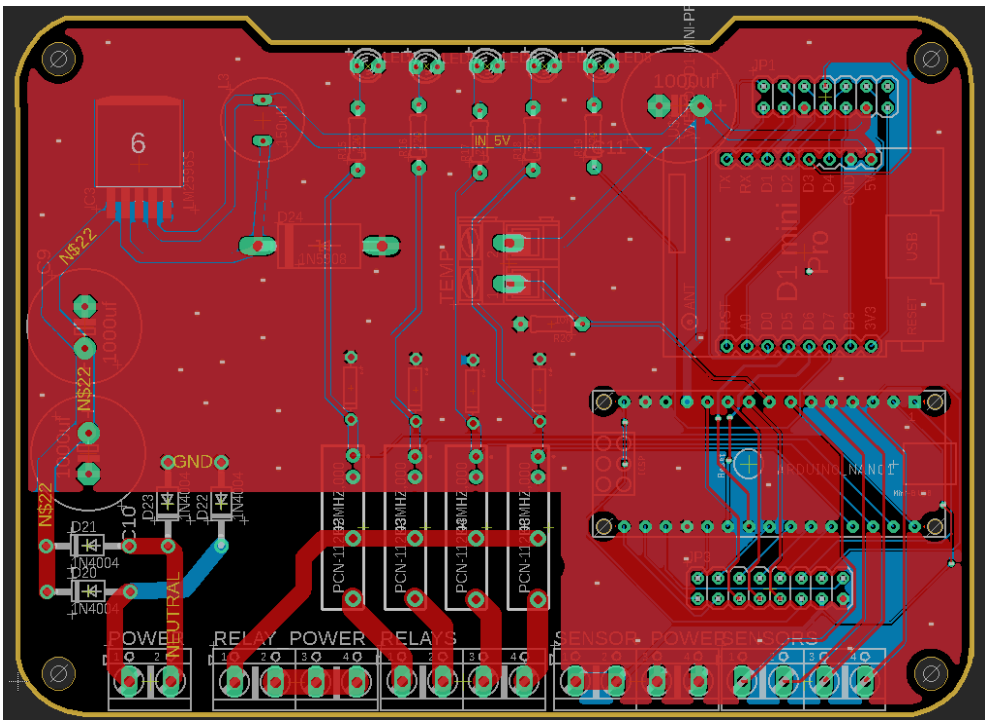


Figura 50 - Plano masa capa top y vías pasantes

En la pestaña “manufacturing”, como se observa en la Figura 51, se visualiza cuál sería el aspecto de la placa de circuito impreso en 3D.

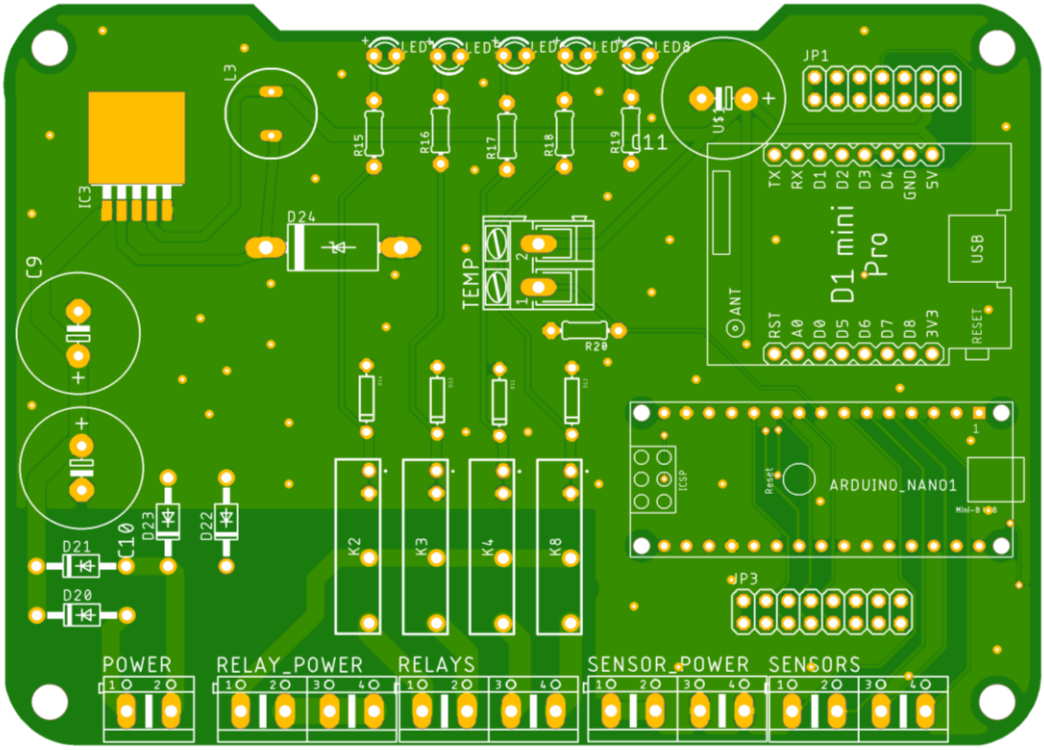


Figura 51 - Placa circuito impreso 3D en *manufacturing*

Por último, mediante el editor se producen varios archivos, entre ellos, los archivos GERBER. Para generar dichos archivos se hace pulsando el botón “CAM Processor” y aparece una ventana, como la que se observa en la Figura 52 y finalmente se pulsa el botón “Process job”.

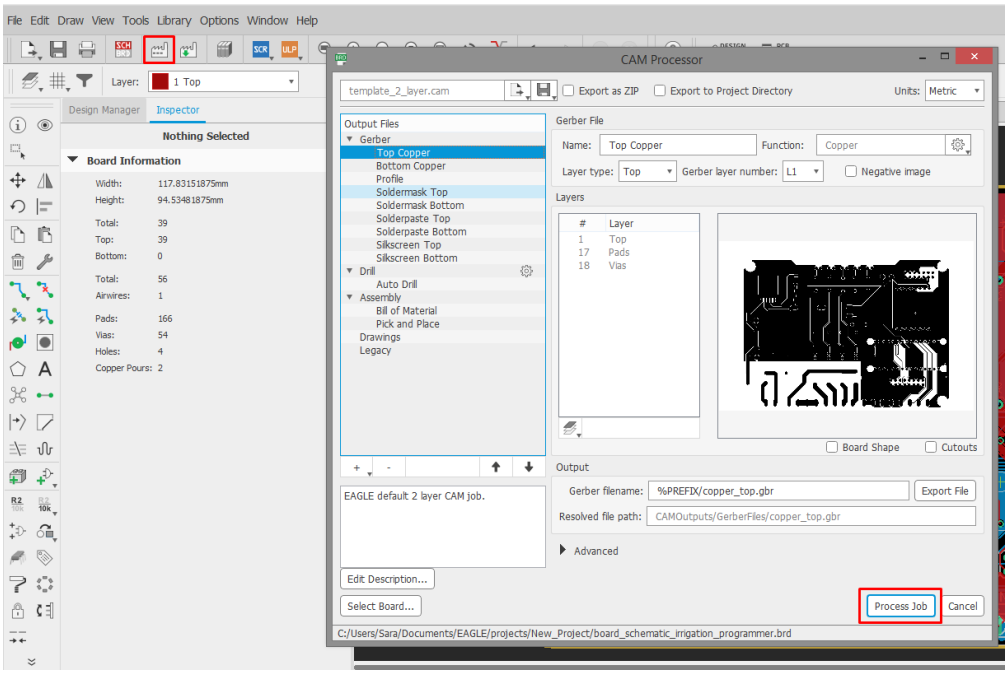


Figura 52 - Generación archivos GERBER en EAGLE

Una vez generados dichos archivos *gerber*, se envían a un fabricante de PCBs y de esta manera se consigue el circuito impreso diseñado manualmente. En este caso la página web que se ha utilizado para la fabricación de la tarjeta es JLCPCB [34], ya que permite imprimir 5 placas por un precio de 6 euros, pero a este precio se le suman 14 euros de envío. Como se observa en la Figura 53 y 54, primero se adjuntan los ficheros *gerber* generados y posteriormente se selecciona la configuración por defecto.

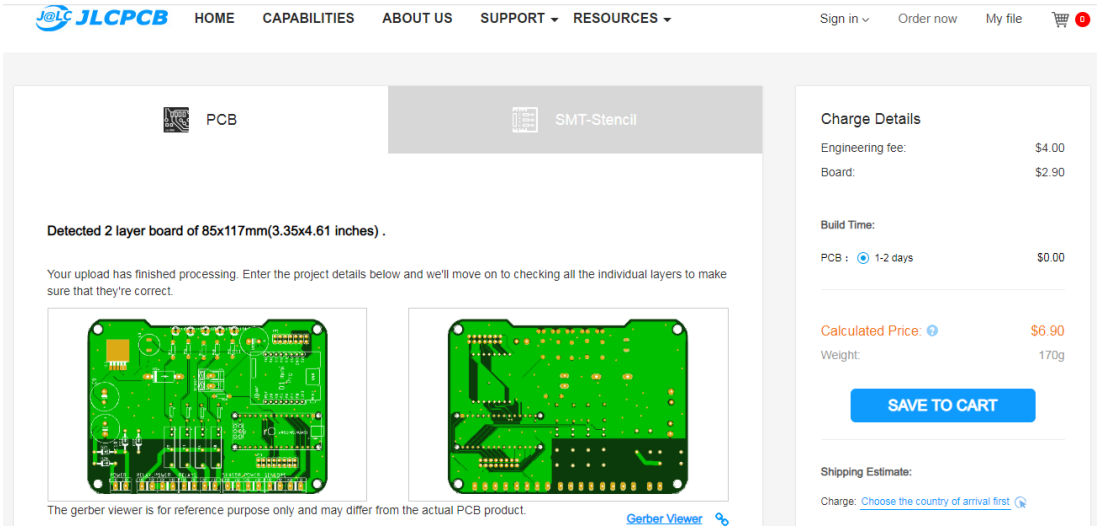


Figura 53 - Impresión circuito en JLCPCB : Adjuntar archivos GERBER

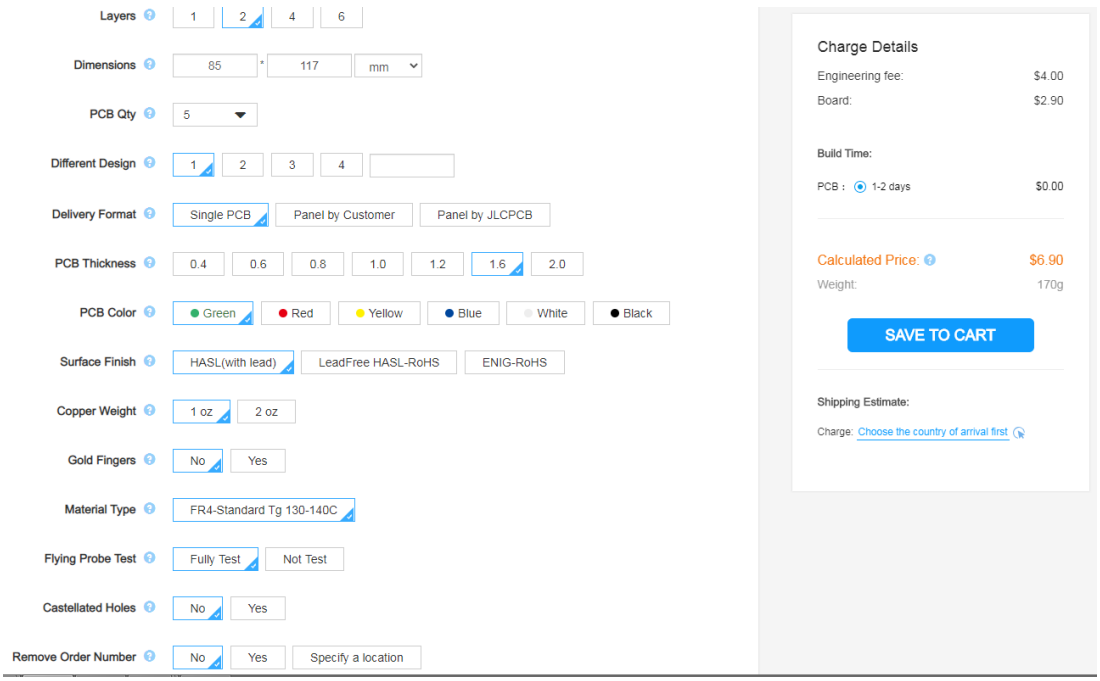


Figura 54 - Impresión circuito en JLCPCB : Configuración

4.2.5 Conexión general del hardware

En este apartado se muestra la conexión de los sensores, de los actuadores y de los módulos Arduino y Wemos. Como se puede observar en la Figura 55, en la parte superior se muestra la conexión de los sensores con Arduino, en la parte media la conexión de Arduino y Wemos y, por último, en la parte inferior se muestra la conexión de las válvulas de agua con Arduino.

- Sensores de temperatura y humedad a GND, VCC y entrada de datos correspondiente de Arduino.
- La salida de las válvulas de agua se conecta a Arduino mediante relés y la placa de relés a los pines VCC, GND, a la entrada de datos correspondiente de Arduino y a la entrada de potencia de 24VAC.
- Wemos se conecta con Arduino para realizar la comunicación serial. Un pin receptor de Arduino se conecta con un pin transmisor de Wemos, mientras que un pin receptor de Wemos se conecta a un pin transmisor de Arduino, y por último se conecta GND de Wemos a GND de Arduino.

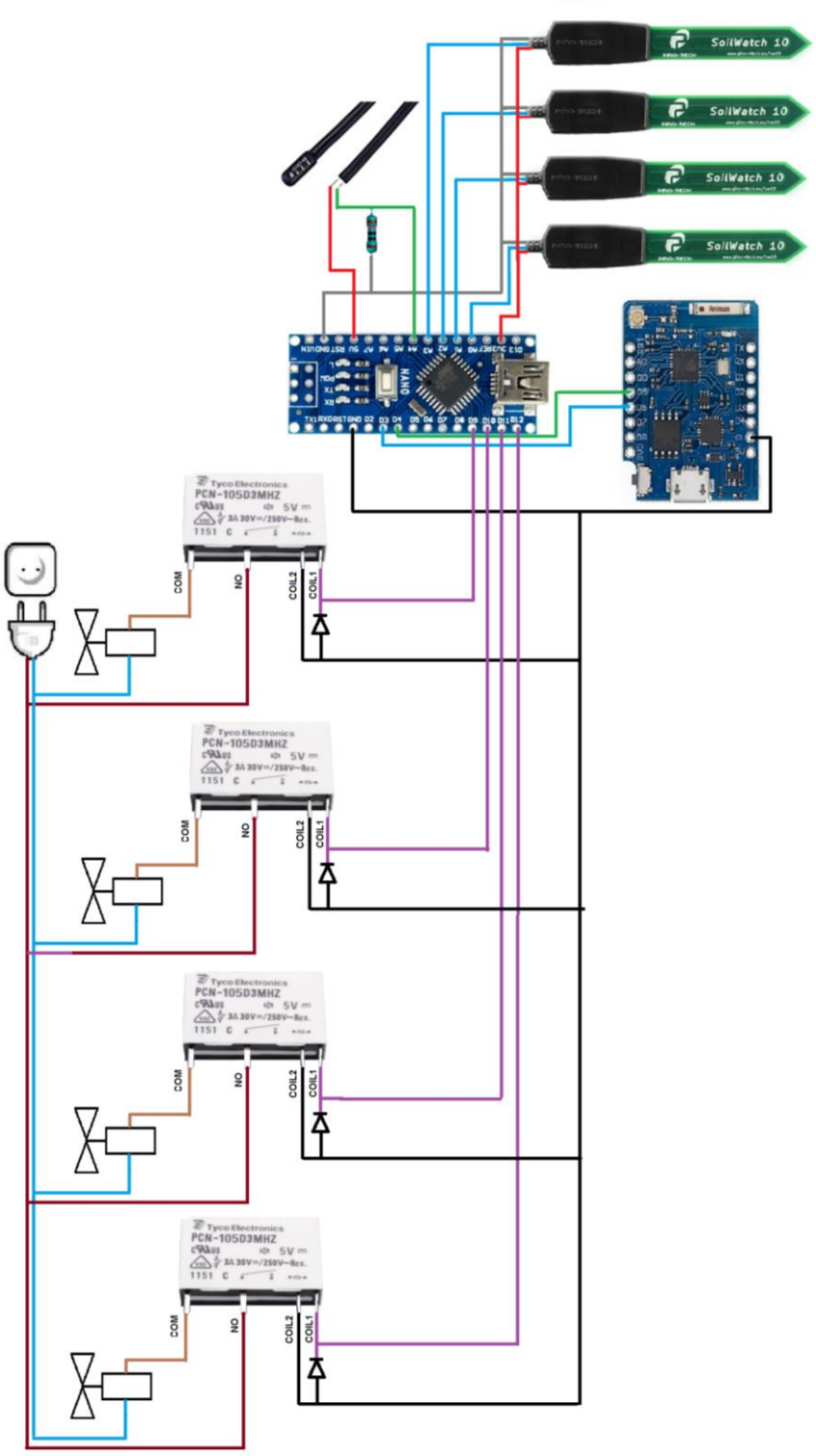


Figura 55 - Esquema conexión general de hardware

En conclusión, los sensores se conectan directamente a la placa Arduino mientras que cada válvula de agua debe conectarse a su relé correspondiente, y cada relé se conecta al pin correspondiente del Arduino. En el apartado de Desarrollo hardware se expondrá el conexionado detallado de cada dispositivo.

4.3 Tecnología utilizada

En este apartado se comentan las diferentes tecnologías y herramientas software que se han utilizado en el proyecto para la implementación de la aplicación.

4.3.1 Tecnologías de implementación

En cuanto a las tecnologías de implementación, se han utilizado múltiples tecnologías las cuales se han conectado entre sí para poder realizar tareas complejas de forma sencilla. A continuación, se presentan las tecnologías más significativas usadas en este proyecto y cuál ha sido su uso.

4.3.1.1 WebSocket y HTTP

WebSocket [14] es un protocolo de comunicación que proporciona un canal del comunicación bidireccional y *full-duplex* sobre un socket TCP. Cabe recordar que f*ull-duplex* hace referencia a un sistema en el que se puede enviar y recibir información de forma simultánea. En este proyecto se utiliza este protocolo para que los datos se transmitan en tiempo real y de esta forma el usuario pueda observar un flujo de datos fluido.

La parte de *Websockets* dentro de este proyecto, es la parte en la que se necesita acciones instantáneas en el programador, cómo activar/desactivar una válvula de agua mediante la página web, cambiar los parámetros de humedad del suelo para la activación de una zona y para poder recibir en tiempo real, cada cierto periodo de tiempo, los datos de humedad del suelo o temperatura.

Se ha de mencionar que dentro de la familia de *Websockets,* como se comenta en el apartado de librerías, se ha utilizado SocketIO, una librería para Arduino que permite una comunicación en tiempo real, bidireccional y basado en eventos.

Para el resto de las llamadas no descritas anteriormente, se utiliza HTTP [15], este protocolo de comunicación se usa para todas las llamadas que necesitan una respuesta, como por ejemplo, registrar un usuario en el programador ya que es necesario recibir el *token* para poder hacer llamadas autorizadas o preguntar por el estado del programador, para saber si está registrado en la nube.

4.3.1.2 Programador

Para programar los módulos de los que dispone el programador, es decir, Arduino Nano y Wemos, se ha utilizado el lenguaje de programación C++, el cual es un lenguaje orientado a objetos, de alto nivel, portable, fuertemente tipado, está basado en clases, su sintaxis es heredada del lenguaje C y es eficiente con el hardware ya que es un lenguaje compilado.

Arduino y C++

Arduino es conocido por ser una librería escrita en C++ para poder programar microchips fácilmente. Por ello, se ha utilizado toda la tecnología relacionada con Arduino ya sea utilizando hardware compatible con Arduino, sensores o librerías sobre Arduino para poder hacer tareas específicas como un pequeño servidor web en Wemos o poder almacenar información, con formato clave y valor, en la ROM de forma sencilla.

Para programar un microchip son necesarios lenguajes que sean capaces de ejecutarse en entornos de alto rendimiento, por lo tanto, se utilizan lenguajes como C o C++ para poder programar los módulos, en este caso, como se ha utilizado Arduino, el lenguaje de programación utilizado es C++.

4.3.1.3 Página embebida

Preact

Preact [16] es un *framework* en JavaScript para el desarrollo de aplicaciones web, la parte de front-end, que a diferencia de React, ocupa alrededor de 3kb y además es más rápido. Cabe destacar que Preact no es una versión diferente de React, sino es una librería que puede ofrecer soluciones parecidas a necesidades que se pueden necesitar cubrir con React. Preact se asemeja en React en varios conceptos, como la sintaxis, ambos se basan en el paradigma de componentes y trabajan con Virtual DOM. En este proyecto se ha decidido utilizar este *framework* porque se necesitaba crear una página web pero que ocupase poco espacio ya que está embebida en el Wemos y porque anteriormente ya se había utilizado React para crear páginas web, por lo que Preact era un *framework* que se ajustaba totalmente a las necesidades.

Figura 56 - Logo Preact

4.3.1.4 Página web

ReactJS

React [17] es una biblioteca de JavaScript de código abierto utilizada para crear interfaces de usuario. Esta biblioteca permite construir aplicaciones que usan datos que cambian todo el tiempo ya que permite que las vistas se asocien con los datos, de modo que, si cambian los datos, también cambian las vistas. Está basado en programación orientada a componentes, donde cada componente es una pieza de código reutilizable y combinando varios componentes, se crean componentes más grandes hasta obtener una web completa. En este proyecto se ha utilizado para crear la página web porque ya se había trabajado con esta biblioteca previamente, porque es uno de los *frameworks* más utilizados en la actualidad y porque es una biblioteca que facilita mucho el desarrollo de una interfaz gráfica.

Figura 57 - Logo React

4.3.1.5 Backend

Strapi

Strapi [18] es un *CMS (Content Management System) Headless open source*, lo que quiere decir que es un sistema de gestión de contenido y con *Headless* que en español significa “sin cabeza” hace referencia a que este sistema es solo para la parte *backend.* Ya que la parte *frontend* es considerado “la cabeza” de la aplicación, que es la página web, mientras que la parte del *backend* es considerado como “el cuerpo” de la aplicación, que es el repositorio del contenido. En Strapi se realiza la comunicación

entre la parte *backend* y *frontend* mediante una *API (Application Programming Interface)* que se puede personalizar. En este proyecto se ha utilizado Strapi para realizar la parte del *backend* de forma sencilla y rápida.

Figura 58 - Logo Strapi

4.3.2 Herramienta de desarrollo

4.3.2.1 Desarrollo de código de programación

Visual studio code

Para el desarrollo del proyecto completo se ha utilizado VSCode [19] que es un editor de código fuente que incluye Git integrado, además de soportes de integración y gran cantidad de extensiones.

Figura 59 - Logo Visual Studio Code

PlatformIO

PlatformIO [20] es un entorno de desarrollo pensado para aplicaciones IoT que se instala como un *plugin* en VisualSC. Entre sus características se destacan el manejo centralizado de las dependencias y la capacidad de migrar los proyectos, ya que empaqueta todo el código C++ para insertarlo en el Arduino y proporciona todas las cabeceras necesarias para que funcione el proyecto en el Wemos. En este proyecto se utiliza para la gestión de librerías ya que en el fichero “platform.ini” se especifica las placas que se van a utilizar, así como la configuración del proyecto. PlatformIO se encarga de descargar todas las herramientas necesarias y las instala automáticamente.

Figura 60 - Logo PlatformIO

Arduino IDE

Mientras que para el desarrollo previo de pequeños programas/pruebas para obtener datos de los sensores y controlar los actuadores, se ha utilizado Arduino IDE [21] que es una aplicación multiplataforma utilizada para escribir y cargar programas en placas compatibles con Arduino.

Figura 61 - Logo Arduino IDE

4.3.2.2 Desarrollo de PCB

EAGLE

Para el desarrollo de la placa de circuito impreso se ha utilizado EAGLE *(Easily Applicable Graphical Layout Editor)* [13], una aplicación de automatización de diseño electrónico que permite diseñar diagramas de circuitos, PCBs. EAGLE permite conectar diagramas esquemáticos, enrutamiento automático, colocación de componentes, entre otras cosas.

Figura 62 - Logo EAGLE

Los PCBs diseñados mediante EAGLE se pueden enviar a manufacturar gracias a que EAGLE guarda los archivos de diseño de Gerber, y estos archivos son los necesarios para poder manufacturar el circuito impreso.

4.3.3 Librerías

A continuación, se va a mencionar las librerías más relevantes que se han utilizado en el desarrollo del proyecto.

En cuanto al programador, para la comunicación serial entre Arduino y Wemos se ha utilizado la librería "SoftwareSerial" [22] tanto en Arduino como en Wemos. Para la programación en Arduino se ha utilizado la librería Arduino[21] y para Wemos se han utilizado las siguientes librerías:

- "arduinoJSON" [23] se utiliza para el manejo de los objetos JSON, como por ejemplo, para pasar de JSON a Objetos y viceversa.
- "justwifi" [24] es una librería para ESP8266 que gestiona la conexión Wifi.
- "ESPAsyncWebServer" [25] y "ESPAsyncTCP" [26] se utilizan para crear el servidor en ESP8266.
- "Embedis" [27] sirve para para almacenar y recuperar datos de pines, sensores, interfaces y otros dispositivos de E/S.
- "Eeprom_rotate" [28] para guardar cosas en la EPROM fácilmente, para guardar claves valor, ente otras cosas.

Por otra parte, para el desarrollo de la página embebida se ha utilizado el *framework* Preact [16] mientras que para la página web se ha utilizado React [17]. En la parte del *backend* se ha utilizado Strapi [18] y SocketIO [29].

Por último, cabe destacar que también se han utilizado librerías de los componentes utilizados en el desarrollo del circuito en EAGLE.

5 Desarrollo de la solución propuesta

5.1 Desarrollo software

En este apartado se expone el desarrollo de la solución propuesta tanto en el ámbito software, es decir, el desarrollo de los subsistemas desarrollados, como en el ámbito del hardware, donde se muestra detalladamente como se han interconectado todos los elementos electrónicos; los sensores, actuadores y programador. Cómo se ha realizado la soldadura de todos los componentes electrónicos anteriormente comentados en el PCB y como se han realizado el montaje de la maqueta del jardín vertical todos los dispositivos.

5.1.1 Programador

Para el desarrollo del código del programador se ha utilizado Arduino, y para el desarrollo del módulo Wemos se ha basado en el código de ESPURNA [31] que es un *firmware* para cualquier tipo de *hardware* IoT, el cual se configura para poder actuar sin depender del *firmware* del fabricante, y poder configurar otros protocolos como, por ejemplo, MQTT. Gracias a este repositorio se ha sabido cómo desarrollar muchas de las partes que se utilizan, como es el caso de crear un punto de acceso, descubrir librerías Arduino para hacer acciones específicas, guardar información en EPROM, hacer una web embebida, y mandar información por *Websockets* en Arduino.

I. Programación

Comunicación serial entre Arduino y Wemos

En cuanto a la comunicación entre Arduino y Wemos, se ha realizado mediante el puerto serie. Para transmitir la información deseada, como es, por ejemplo, el estado de las válvulas de agua se ha creado un protocolo de comunicación que trata de un array de 18 valores donde cada valor representa determinada información dependiendo de su posición:

- El primer valor del array `(dataSensores[0])` hace referencia al identificador de la zona.

- Los cuatro siguientes valores (`dataSensores[1-4]`) representan el valor de humedad de suelo que tiene cada zona.
- El siguiente valor (`dataSensores[5]`) es la temperatura externa.
- Los cuatro siguientes valores (`dataSensores[6-9]`) hacen referencia al estado de los relés de las 4 zonas.
- Los cuatro siguientes valores (`dataSensores[10-13]`) representan el valor mínimo de humedad de suelo que se ha especificado en cada zona, mientras que los cuatro siguientes valores(`dataSensores[14-17]`) hacen referencia al valor máximo de humedad de suelo definido en cada zona.

```
dataSensores[0] = 1;//id
dataSensores[1] = getHumidity(0);//sensor1
dataSensores[2] = getHumidity(1);//sensor2
dataSensores[3] = getHumidity(2);//sensor3
dataSensores[4] = getHumidity(3);//sensor4
dataSensores[5] = getTemperature();//temp
dataSensores[6] = getRelayState(0);//relay1
dataSensores[7] = getRelayState(1);//relay2
dataSensores[8] = getRelayState(2);//relay3
dataSensores[9] = getRelayState(3);//relay4
dataSensores[10] = 15;//min humidity
dataSensores[11] = 20;//min humidity2
dataSensores[12] = 25;//min humidity3
dataSensores[13] = 70;//min humidity4
dataSensores[14] = 99;//max humidity1
dataSensores[15] = 60;//max humidity2
dataSensores[16] = 50;//max humidity3
dataSensores[17] = 80;//max humidity4
```

Cambio estado de relés

El estado de los relés de cada zona cambia dependiendo de la humedad de suelo de dicha zona. Este control se realiza mediante el método *checkHumidity.*

```
void checkHumidity(uint8_t id, uint8_t currentHumidity) {
   uint8_t min = configurations[id][0];
   uint8_t max = configurations[id][1];
   uint8_t relayState = getRelayState(id);

   if (currentHumidity < min && relayState == 0){
       changeRelayState(id, true);
   } else if(currentHumidity >= max && relayState == 1){
       changeRelayState(id, false);
   }
}
```

Este método es llamado cada cierto periodo de tiempo para comprobar si los niveles de humedad de suelo se encuentran entre los rangos especificados por el usuario. Dicho método recibe dos argumentos, el identificador de la zona (id*)* que podría ser entre 0 y 3 y la humedad actual del suelo (*currentHumidity*), con dicha humedad actual se comprueba si es menor que la humedad permitida y el estado del relé, si es así, se enciende la válvula de agua de esa zona. De lo contrario, si la humedad actual es mayor o igual a la del punto de consigna y el relé está activado, entonces apaga la válvula de agua.

Lectura de humedad

```
void sensorsLoop() {
  static unsigned long next = millis();
  if (next < millis()) {
    next += 5000; //every 5 seconds

    for(uint8_t id = 0; id < BUILT_IN_SENSORS; id++) {
      int moistureValue = analogRead(sensors[id]);
      int mappedValue = map(moistureValue, minADC, maxADC, 0, 100);
      sensorValues[id] = mappedValue;
    }
   _temperature = _getNTCTemperature();
  }
}
uint8_t getHumidity(uint8_t id) {
  return sensorValues[id];
}
```

Al tener cuatro sensores de humedad es necesario repetir el mismo proceso utilizando un pin diferente de lectura. Por ello se utiliza una función de Arduino disponible llamada “map”, la cual devuelve el valor que necesitas en el rango de valores correspondiente dependiendo del voltaje de entrada representado en bits (1024). En este caso es necesario leer entre 0% de humedad y 100%, dependiendo del voltaje de entrada en bits entre 0 y 600 ya que los sensores de humedad utilizados operan mejor a 3.3v, lo que representa 600, si fuera 5v sería 1024.

Lectura de temperatura

```
uint8_t getTemperature() {
    return _temperature;
}

uint8_t _getNTCTemperature() {
  float vo = analogRead(PIN_TEMPERATURE);
  float resistorfixed = 10000;
  float R2 = resistorfixed * (1023.0 / vo - 1.0);

  float logR2 = log(R2);
  float T = (1.0 / (C1 + C2*logR2 + C3*logR2*logR2*logR2));
  float temp = T - 273.15;

  return temp;
}
```

El sensor de temperatura utilizado, al ser NTC, se debe utilizar una fórmula para obtener la temperatura en grados centígrados, por lo que para programar dicha fórmula se ha basado en un código ya realizado [32]

II. Websockets

Una vez el usuario es vinculado al programador y el programador es creado en el *backend*, el *backend* dispone de toda la información necesaria para iniciar una llamada por *Websockets* directamente a internet y tener acceso a el programador desde cualquier parte.

El programador necesita realizar dos tareas principales con *Websockets*, esperar eventos de usuario para realizar las acciones correspondientes y enviar cada cierto periodo de tiempo información de los sensores.

SocketIO tiene tres métodos utilizados importantes:

1. "`socket.begin`" para poner la URL y puerto que se necesita conectar.
2. "`socket.loop`" el cual se tiene que llamar dentro del método "`loop`" de Arduino.
3. "`socket.onEvent`" para registrar la función *callback* que será llamada cuando la conexión reciba información.

En la función de inicio de SocketIO llamada "`cloudSetup`" si el usuario ya ha sido registrado, el programador dispondrá de la información necesaria para conectarse, entonces se llama a "`socket.onEvent`" y a "`socket.begin`" para iniciar la conexión.

```
void cloudSetup() {
    if(isRegister()) {
        socket.onEvent(socketIOEvent);
        socket.begin(BACKEND_URL, BACKEND_SOCKET_PORT, String("/socket.io
/?a=" + getDeviceId() + "&c=device").c_str());
    }
}
```

Por otra parte, en el método "`cloudLoop`", el cual es llamado periódicamente, llama a la función "`socket.loop`" para que se mantenga la conexión abierta. Como se puede observar en el código de a continuación, solo se llama a "`socket.loop`" si el programador está conectado a la red wifi y está registrado en el *backend*.

```
void cloudLoop() {
    if(wifiConnected() && isRegister()) {
        socket.loop();
    }
}
```

Cuando Wemos recibe la información de los sensores y de las zonas por Serial entonces llama al método "`emit`", el cual llama al método "`socket.sendEVENT`" con la información del evento y su contenido. Como se observa se debe formatear en el formato que entiende el protocolo SocketIO, con un *array* con el primer parámetro con el *topic* y segundo argumento con el mensaje.

```
void emit(String event, String payload) {
    String msg = String("42[\"");
    msg += event;
    msg += "\"";
    msg += ",";
```

```
    msg += payload;
    msg += "]";
    socket.sendEVENT(msg);
}
```

Cuando un usuario realiza una acción, ésta es recibida por la función *callback*, la cual recoge el mensaje, lo transforma de texto un objeto JSON y dependiendo del mensaje se hace una acción cómo por ejemplo, encender o apagar un relé.

```
void socketIOEvent(socketIOmessageType_t type, uint8_t * payload, size_t
length) {
    switch(type) {
        case sIOtype_DISCONNECT:
            DEBUG_MSG_P(PSTR("[SocketIO] Disconnected!\n"));
            isReadyToSend = false;
            break;
        case sIOtype_CONNECT:
            isReadyToSend = true;
            break;
        case sIOtype_EVENT:
        {
            DynamicJsonDocument doc(1024);
            DeserializationError error =
                  deserializeJson(doc, payload, length);
            if(error) {return;}
            String eventName = doc[0];
            String message = doc[1];
            if(eventName == "action") {
                DynamicJsonDocument root(1024);
                auto error = deserializeJson(root, message.c_str());
                if (error) {return;}
                const char *action = root["action"];
                if (action) {
                    onAction(action, root.as<JsonObject>());
                }
        }
     break;
    }
}
```

Todos los mensajes tienen el atributo “action” con el nombre de la acción a realizar e “id” para saber a qué zona es dirigida la acción.

```
void onAction(const char *action, JsonObject payload) {
    if (strcmp(action, "zone") == 0) {
        uint8_t id= payload["id"];
        uint8_t value = payload["state"];

        String output;
        serializeJson(payload, output);
        sendRelayCommand(id, value);
    }
}
```

Un ejemplo de un mensaje de acción para encender o apagar una zona es el siguiente:

{ “action”: “zone”, “id”: 1, “state”: 0 }

5.1.2 Página embebida

III. Vistas

La página de configuración embebida solamente se compone de una vista como la que se observa en la Figura 63. Dicha página consta de cuatro secciones, en la parte superior se encuentra el estado del programador, donde se muestra el nombre de la red Wifi, la dirección IP y en la parte derecha se muestra si el programador está conectado a una red wifi. A continuación, se encuentra la sección de las zonas del jardín, donde se puede actuar sobre las válvulas de agua de cada zona. Posteriormente se muestra una sección de registro de Wifi donde se debe introducir el SSID y la contraseña del wifi a la que se desea conectar el programador. Finalmente, en la parte inferior se encuentra el inicia de sesión del usuario.

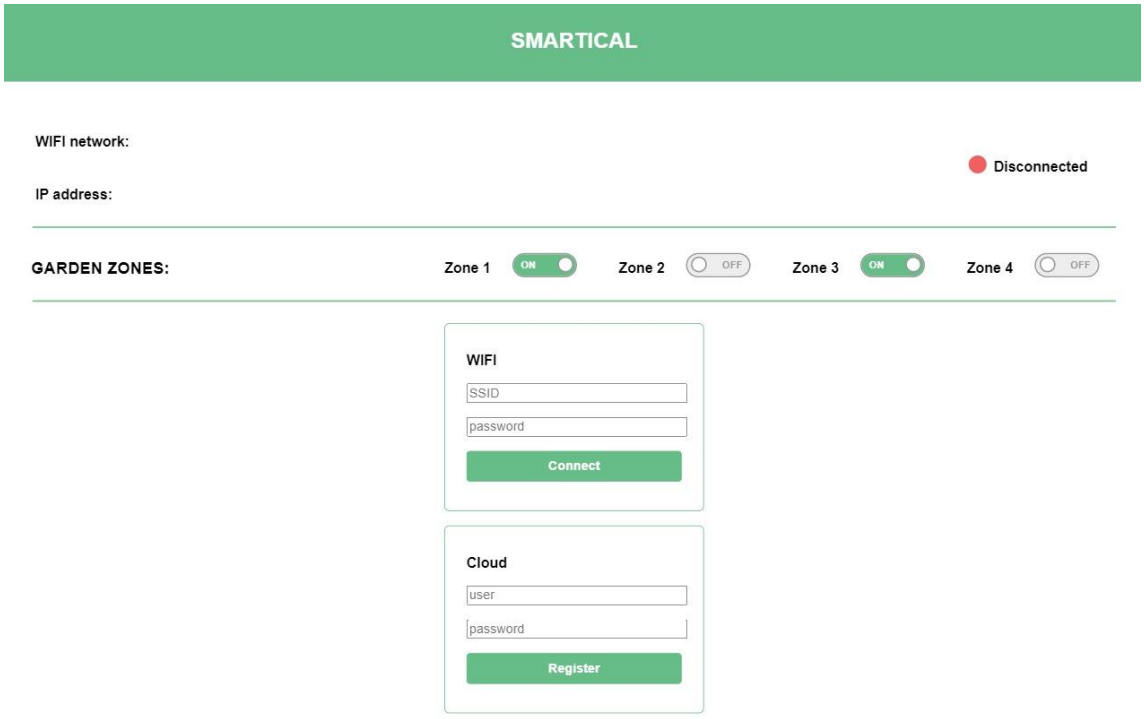


Figura 63 - Vista página web embebida

IV. Programación

En el módulo WIFI Wemos es necesario mostrar una página web para poder configurar el programador cuando no tiene conexión a internet y poder modificar el estado de las válvulas de agua. Conectarse directamente al programador es fundamental ya que, en un primer momento, el programador no sabe conectarse a la red WIFI, por lo tanto, se necesita una forma de poder acceder a él sin necesidad de ninguna conexión a internet.

Para poder acceder a la página web, la primera vez que se enciende el programador, éste crea un punto de acceso para poder conectarse a su red local y poder mostrar la página web, la cual se encuentra en la URL compuesta por la dirección IP 192.168.4.1. La página web que se muestra en un navegador web se tiene que comunicar con el programador para poder mandarle acciones y que éste actúe sobre esas acciones, como, por ejemplo, conectarse a una red WIFI, enviar la información del programador al servidor *backend*, etc. Una vez introducida la red wifi y la contraseña, se reinicia el programador y aparece, como se muestra en la Figura 64, el nombre de la red wifi y la dirección IP.

Figura 64 - Vista página web embebida: programador conectado a red wifi

Para comunicarse con el módulo WIFI, éste, crea una API REST para poder comunicarse con él a través de HTTP.

La página web ha de estar embebida dentro del código C++ de Arduino por lo que es necesario poder generar un único fichero con HTML, CSS, Javascript y añadirlo en el

código para cuando se llame a el módulo WIFI mediante HTTP, pueda devolver ese fichero HTML.

Para desarrollar la página web se ha usado Preact, que es una versión de React que ocupa poco espacio, ya que el módulo Wemos sólo tiene 4 MB de memoria, por lo que si se utiliza un *framework* debe ser tamaño pequeño para poder añadir más código. A continuación, se va a explicar la configuración para hacer funcionar con el módulo Wifi, ya que el código de la página web es un código realizado con el *framework* React. La configuración está basada en el código de "configduino" [30] en el que crea una página web embebida con Preact para usarlo con ESP8266.

Una vez terminada la página web utilizando el *framework* Preact, es necesario convertir todos los diferentes ficheros Javascript y CSS a un solo fichero para, posteriormente, convertirlo en un gran String y añadirlo como parte del código C++ como una variable. Para ello se ha utilizado "webpack" que es un configurador de Javascript que se puede configurar según las necesidades, en este caso se crea un fichero HTML (index.html) a partir de todos los ficheros Javascript y CSS.

Todo esto se realiza añadiendo en un fichero llamado "webpack.config.babel.js" unos *plugins* como los que se muestran en el siguiente código. Por lo que este *plugin* se configura indicándole en "`inlineSource`" el tipo de ficheros que tiene que convertir, en este caso, son los ficheros CSS y JavaScript.

```
plugins: [
  new ExtractTextPlugin({
    filename: "style.css",
    allChunks: true,
    disable: ENV !== "production"
  }),
  new HtmlWebpackPlugin({
    template: "./index.ejs",
    minify: { collapseWhitespace: true },
    inlineSource: "(.js|.css)$"
  }),
  new HtmlWebpackInlineSourcePlugin(HtmlWebpackPlugin),
  new webpack.DefinePlugin({
    'process.env.production': ENV === "production",
    'process.env.BUILD': IS_BUILD
  })
],
```

Una vez añadidos estos *plugins* de webpack, solo falta especificar que se utilice webpack en vez de utilizar directamente Preact. Por lo que, cuando se ejecuta "`npm`

`run build`” para generar la página web, se debe añadir en el fichero “package.json”, la siguiente línea de build de webpack.

- `"build": "webpack -p --progress"`

Por lo que cuando se realice “`npm run build`” para construir la web, se genera un único fichero “index.html” en el que se encuentra todo el HTML, CSS y JS generado a partir del *framework* de Preact, para que ocupe menos, y ese fichero se comprime en “gzip”. Posteriormente, se debe introducir este código dentro de Arduino, y para ello se ha utilizado un *script* basado en ESPURNA que realiza este mismo proceso. En el código que se muestra a continuación, se crea un fichero C++ con una variable PROGMEM con el código HTML en binario.

```
const data = fs.readFileSync(source);

wstream.write("#define index_html_gz_len " + data.length + "\n");
wstream.write("const uint8_t index_html_gz[] PROGMEM = {");

for (i = 0; i < data.length; i++) {
  if (i % 1000 == 0) wstream.write("\n");
  wstream.write("0x" + ("00" + data[i].toString(16)).slice(-2));
  if (i < data.length - 1) wstream.write(",");
}
```

La librería que se utiliza para devolver el fichero HTML tiene soporte para PROGMEM, por lo que cuando se llame a el PATH principal se devuelve el HTML completo, como se muestra en el código.

```
#define index_html_gz_len 9939
const uint8_t index_html_gz[] PROGMEM = {
0x1f,0x8b,0x08,0x08,0x10,0xe2,0xd3,0x5e,0x00,0x03,0x69,0x6e,0x64,0x65,…
};
```

Adicionalmente se debe informar al navegador que ese HTML está comprimido con “gzip” para que lo pueda descomprimir, lo cual se realiza añadiendo una cabecera.

```
AsyncWebServerResponse *response = request->beginResponse_P(200,
"text/html",
index_html_gz, index_html_gz_len);
response->addHeader("Content-Encoding", "gzip");
```

De esta forma, se genera un fichero HTML, se convierte en una variable PROGMEM y se devuelve cuando se llama el PATH principal (/) del servicio web del programador.

5.1.3 Página web

I. Vistas

La página web realizada consta de varias vistas, en primer lugar, se encuentra la vista de *login*, la cual se observa en la Figura 65, donde el usuario debe iniciar sesión con un correo electrónico y una contraseña.

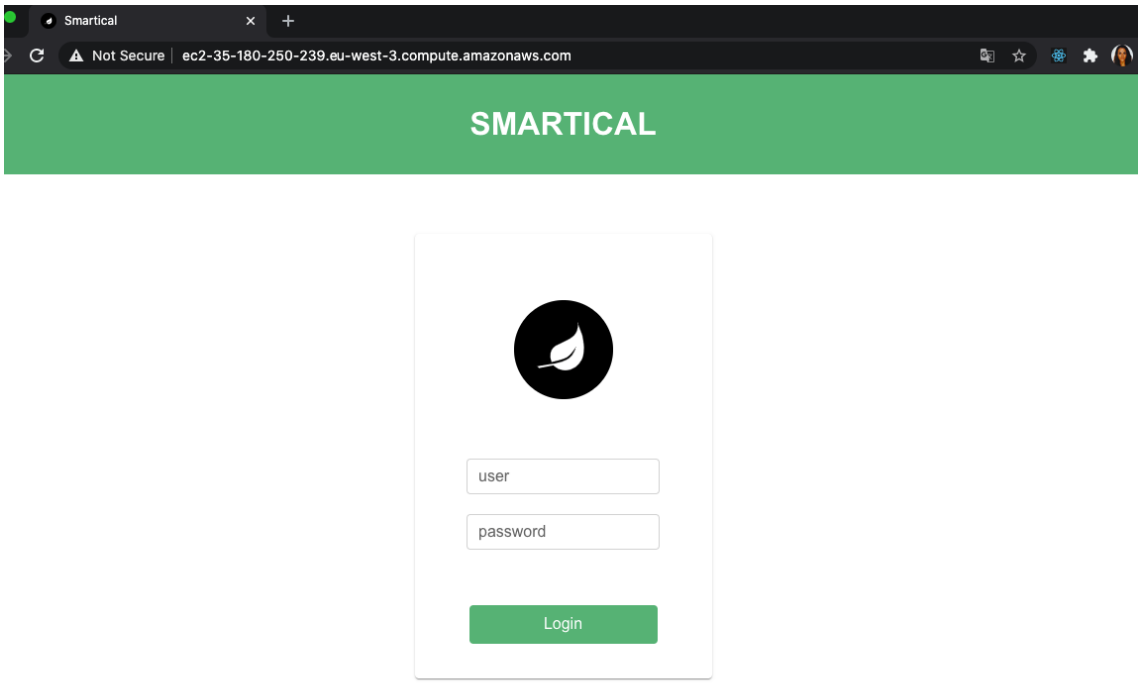


Figura 65 - Vista inicio de sesión página web

Una vez el usuario inicia sesión, aparece una ventana, como la de la Figura 66, con toda la información de su programador, es decir, el estado de su jardín vertical.

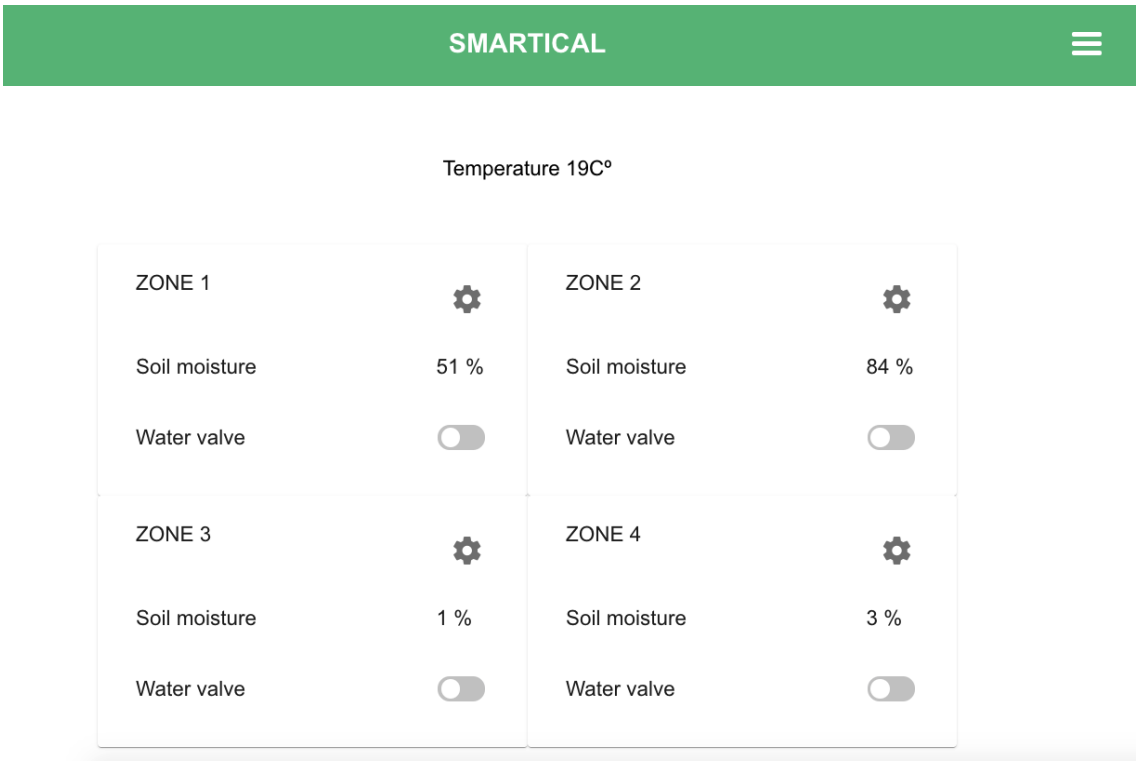


Figura 66 - Vista principal página web

El usuario puede modificar los puntos de consigna de activación y desactivación de cada zona pulsando en el icono de configuración de la zona correspondiente, y de esta manera aparecerá una ventana como la que se muestra en la Figura 67, donde debe especificar el rango mínimo y máximo de humedad de suelo que se desea para esa zona. De forma que si la humedad de suelo está por debajo del mínimo se activará la válvula de agua de esa zona y si la humedad de suelo de dicha zona alcanza el máximo especificado, se cerrará la válvula de agua.

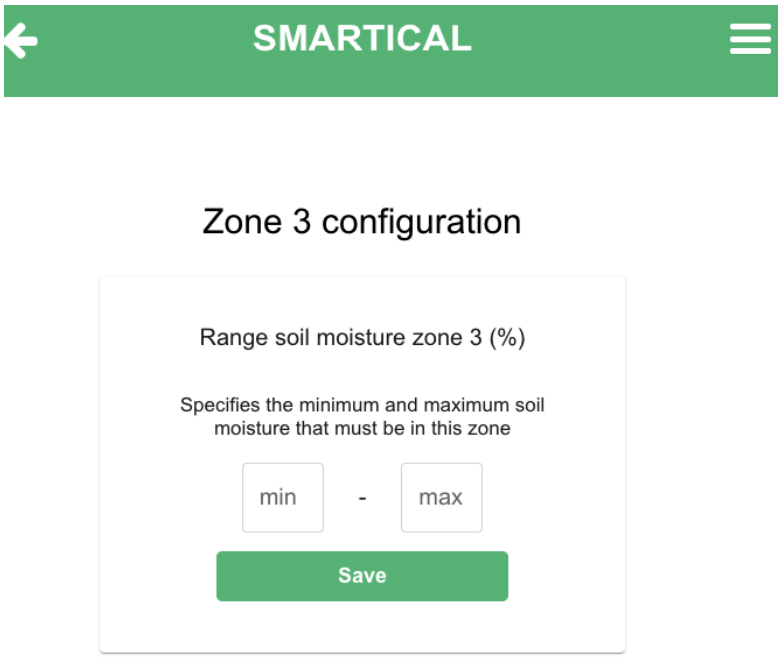


Figura 67 - Vista configuración de zona página web

Adicionalmente, el usuario podrá cambiar de contraseña y cerrar sesión en configuración, pulsando en el icono de menú de la ventana principal, situado arriba a la derecha, de manera que aparecerá una ventana como la que se muestra en la Figura 68 donde el usuario deberá introducir la contraseña actual y la nueva contraseña.

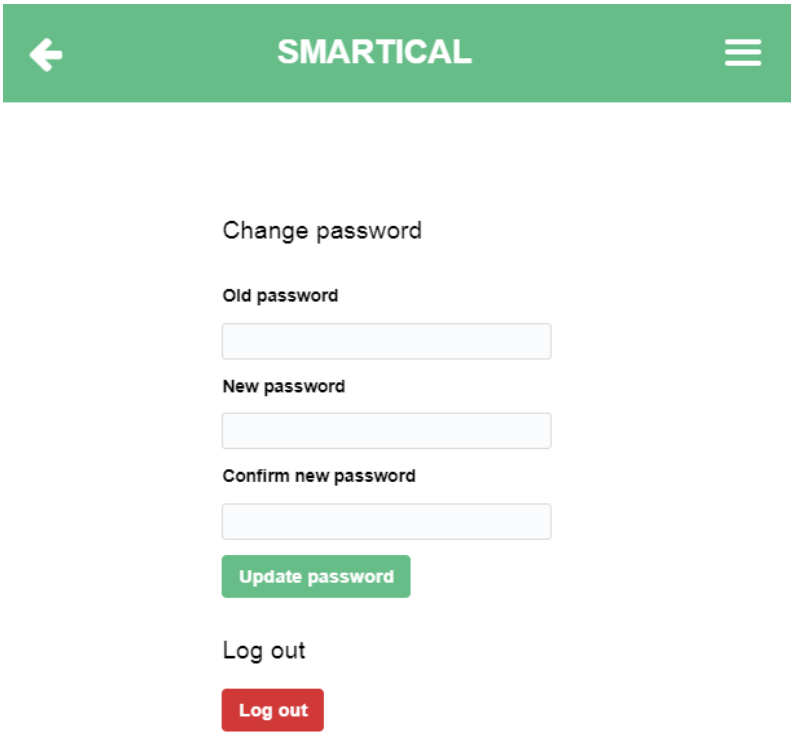


Figura 68 - Vista cambio de contraseña página web

II. Programación

La página web para controlar el programador está desarrollada en React y es la encargada de realizar el inicio de sesión en el sistema con el usuario y contraseña y de mostrar la información del programador que tiene el usuario instalado.

Al realizar la página embebida en Preact que es un *framework* simplificado de React, se pueden reutilizar los mismos componentes, cómo botones, campos de texto, etc. Se ha elegido React, porque es el *framework* de programación Javascript con el que se había trabajado anteriormente.

Inicio sesión

El componente Login que se muestra en el código se encarga de recoger el usuario y contraseña y guardarla en el estado local del componente para cuando pulsen el botón de "*login*", hacer una llamada a el servidor con la información de usuario y contraseña,

```
function Login(props) {
  const [user, setUserInput] = useState('');
  const [password, setPasswordInput] = useState('');
  const [isError, setIsError] = useState(false);
  const { history } = props;

  function handleButton() {
    if (user.length !== 0 && password.length >= 6) {
      postUserLogin(user, password)
        .then(response => {
          store.dispatch(saveUser(response.jwt,
response.user.programmers[0]));
          setIsError(false);
          history.push('/main');
        })
        .catch(error => setIsError(true));
    } else {
      setIsError(true);
    }
  }

  return (
    <div className="app-login">
      <Card className="card-login">
        <CardContent className="card-content-login">
          <img src={logohoja} alt="Logo" className="logo" />
          <input
            className="user-login"
            value={user}
```

```
            onChange={e => setUserInput(e.target.value)}
            placeholder="user"
          />
          <input
            className="password-login"
            value={password}
            onChange={e => setPasswordInput(e.target.value)}
            placeholder="password"
          />
          <p className="error">
            {' '}
            {isError ? 'User or password incorrect' : ''}
          </p>
          <button className="button" onClick={handleButton}>
            Login
          </button>
        </CardContent>
      </Card>
    </div>
  );
}

export default withRouter(Login);
```

si la respuesta del servidor devuelve "*status 200*" con la información del usuario, se pasa a el componente de la página principal, de lo contrario, se muestra un mensaje de error como el que aparece en la Figura 69.

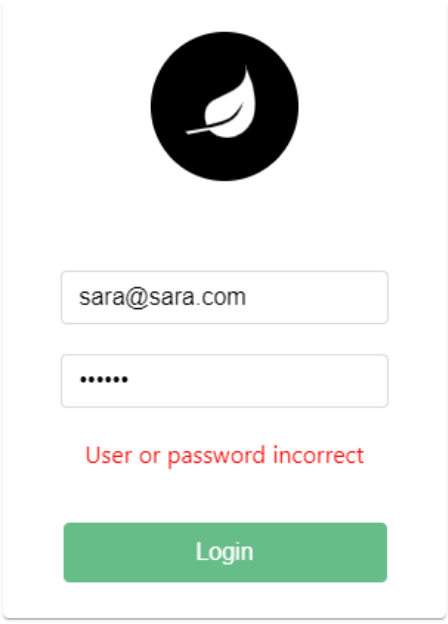


Figura 69 - Mensaje de usuario o contraseña incorrecta en inicio de sesión

Navegación entre vistas

Para poder navegar entre las diferentes páginas que tiene la aplicación, se ha hecho uso de React Router que es una librería en React que permite mostrar componentes dependiendo de la URL.

En el código que se muestra a continuación se observa el componente principal de la aplicación y dependiendo del PATH de la URL muestra un componente u otro.

```
const render = () => {
  ReactDOM.render(
    <React.StrictMode>
      <Router>
        <Header />
        <Route exact path="/">
          <Login />
        </Route>
        <Route path="/main">
          <Main />
        </Route>
        <Route path="/settings">
          <UserConfiguration />
        </Route>
        <Route exact path="/zone/:zoneName">
          <ZoneConfiguration />
        </Route>
      </Router>
    </React.StrictMode>,
    document.getElementById('root'),
  );
};
```

De esta forma se pueden registrar las rutas de la aplicación con sus componentes, por ejemplo */main*, muestra el componente principal llamado *<Main/>.*

III. Arquitectura

La arquitectura utilizada para la web ha sido Redux, que es de las arquitecturas más comúnmente utilizada, en el que los componentes llaman a Acciones que son lanzadas a los Reducers y estos modifican un estado global de la aplicación en el que está estructurada la información que se muestra en la web.

A continuación, se presenta el caso de uso de iniciar sesión en la web mediante Redux.

Actions

Las acciones son llamadas desde los componentes y contienen

- un *payload* que es la información necesaria para llevar a cabo esa acción y
- un tipo que es el identificador de la acción,

en este caso la acción es guardar la información de un usuario, como es el *token* para poder realizar futuras llamadas y el programador de dicho usuario.

El identificador de la acción es SAVE_USER y como *payload* son el *token* y el programador.

```
export const saveUser = (token, programmer) => {
  return {
    type: 'SAVE_USER',
    token,
    programmer,
  };
};
```

Reducers

Una vez llamada a la acción, ésta pasa por el Reducer el cual es el encargado de modificar el estado actual de la aplicación por uno nuevo, el estado de la aplicación contiene toda la información necesaria para mostrarla en la web. En este caso si se necesita mostrar el programador, se debe tener en el estado de la aplicación la información del programador.

Los Reducers tienen un estado inicial de la aplicación del cual se parte para empezar las modificaciones sobre ese estado, en este caso, al recibir un *token* y un programador, se añade dicho *token* y programador a la información que había previamente en el estado.

```
const reducer = (state = initialState, action) => {
  switch (action.type) {
    case 'SAVE_USER':
      return {
        ...state,
        token: action.token,
        programmer: action.programmer,
      };
    default:
      return state;
  }
};
```

Si hubiese más acciones, se deberían añadir en los diferentes casos del *switch* y hacer las respectivas modificaciones. Además de guardar el usuario, se disponen de acciones de cambiar los estados de los relés en la pantalla, y cambiar los rangos de humedad para un programador.

Store

Finalmente explicar la relación entre la acción y el Reducer, cómo se pasa de llamar una acción a que la reciba el Reducer. Esto se realiza mediante la librería Redux, en la que para llamar una acción es necesario llamar a la función "`dispatch`" y para registrar los Reducers es necesario crear el Store donde se almacena el estado y con la que se lanzan las acciones.

```
import { createStore } from "redux";
const store = createStoret(reducer);
export default store;
```

Y para lanzar una acción, se debe llamar al Store y llamar a su función "`dispatch`".

```
store.dispatch(saveUser(response.token, response.user.programmer);
```

De esta manera se lanza la acción de guardar información de usuario y el Store llamará al Reducer, el Reducer cambiará el estado, y el Store guardará ese nuevo estado y se mostrará en React.

IV. Despliegue

Para publicar la página web se ha utilizado se ha utilizado el servicio de EC2 de AWS (Amazon Web Service) [33] para alojar la página web.

En primer lugar, se debe crear un servidor en AWS, para ello, una vez registrado en AWS, se selecciona EC2, como se muestra en la Figura 70, el cual es un ordenador en la nube (una máquina virtual), para poder conectarte mediante SSH e iniciar el servidor.

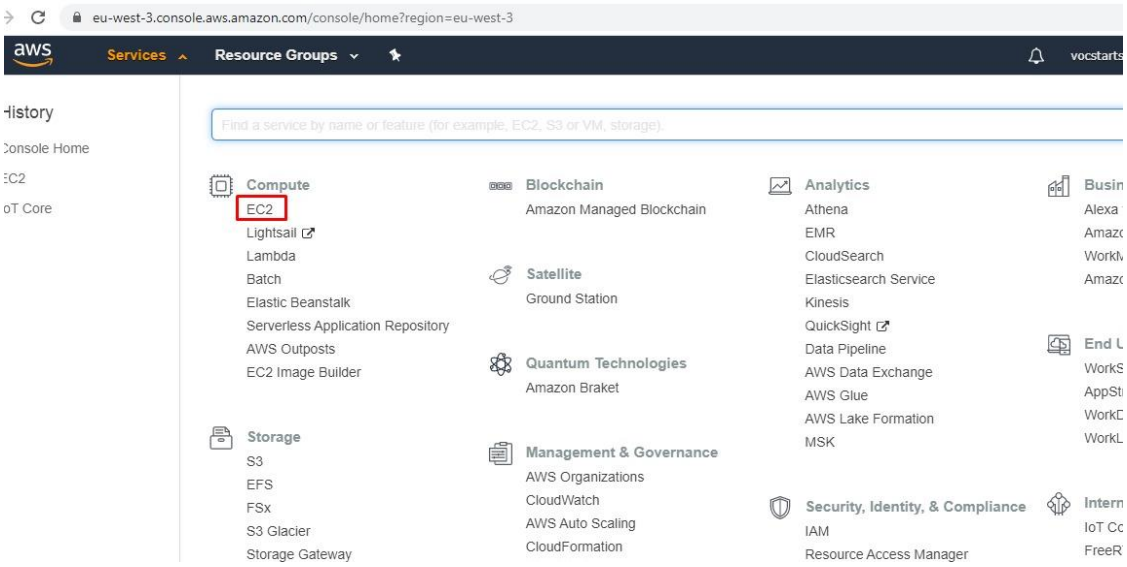


Figura 70 - Selección EC2 como servidor AWS

Dentro de EC2, en las instancias se crea una nueva instancia mediante el botón “Launch instance”, como se observa en la Figura 71.

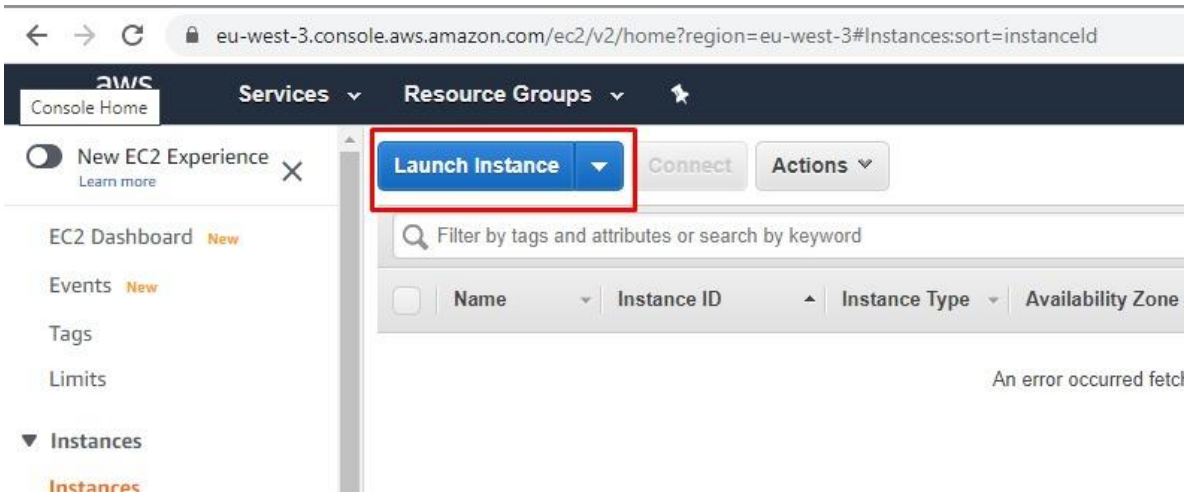


Figura 71 - Creación nueva instancia en AWS

Posteriormente se selecciona el tipo de sistema operativo, Figura 72, y especificaciones que se desean, Figura 73, en este caso se selecciona Amazon Linux porque es la opción más económica y en cuanto a especificaciones, se selecciona el tipo *small* porque es suficiente para este proyecto.

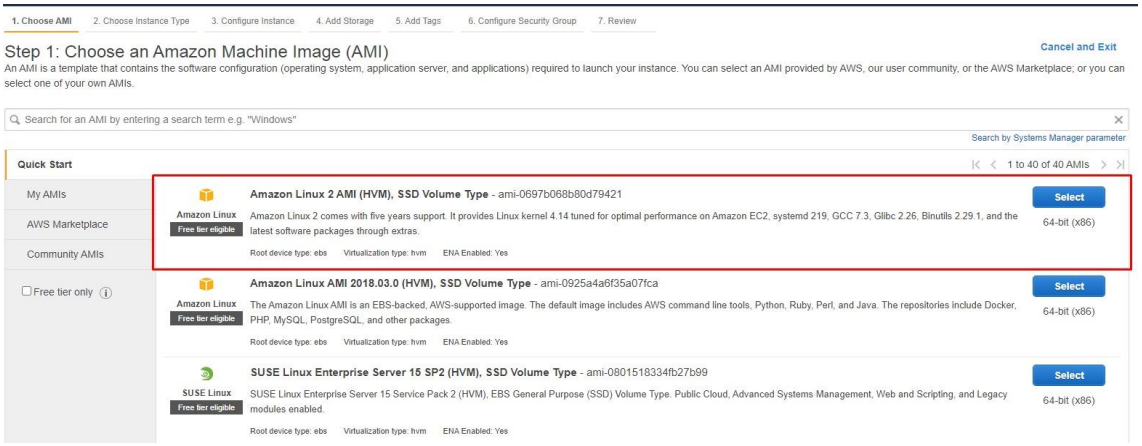


Figura 72 - Selección Amazon Linux como AMI en AWS

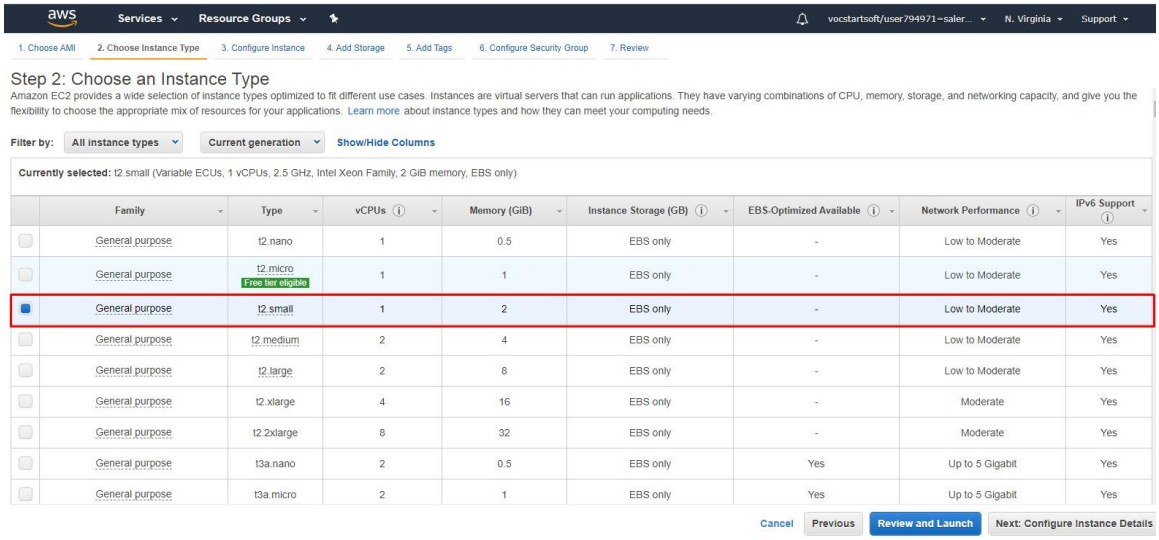


Figura 73 - Selección instancia tipo small en AWS

Una vez seleccionado el tipo de instancia, en los pasos 3, 4 y 5 se pulsa el botón “Next” ya que no se debe realizar ninguna modificación ni selección. En el paso 6, “Configure Security Group”, como se muestra en la Figura 74, se añade una regla para poder habilitar el puerto 80, para poder acceder a la instancia de AWS y habilitar todas las llamadas TCP por dicho puerto.

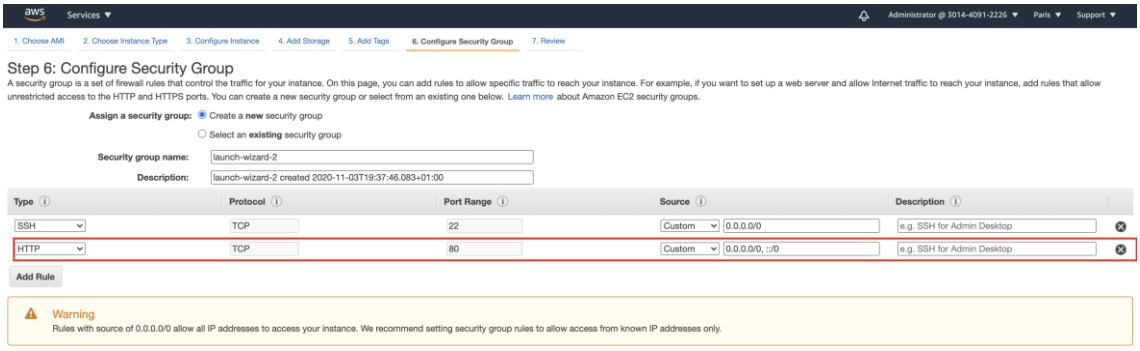


Figura 74 - Habilitar puerto HTTP en la instancia creada en AWS

Y pulsando el botón “Review and Launch”, se crea la instancia. Una vez creada la instancia, se puede instalar los programas necesarios, en este caso se desea conectar, mediante el botón “Connect” y aparecen las diferentes opciones que se muestran en la Figura 75 para poder conectarse.

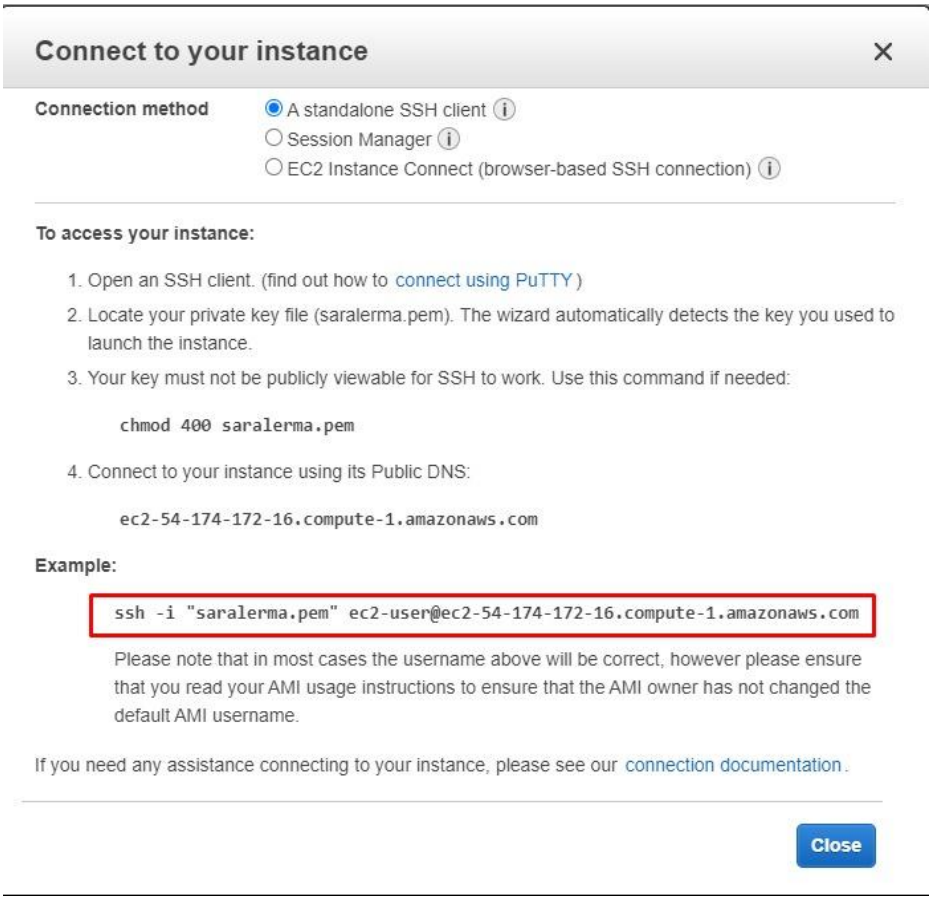


Figura 75 - Instrucciones conexión a instancia creada en AWS

Desde un terminal, como se expone en la Figura 76, se introduce el comando especificado y se inicia una conexión *ssh*.

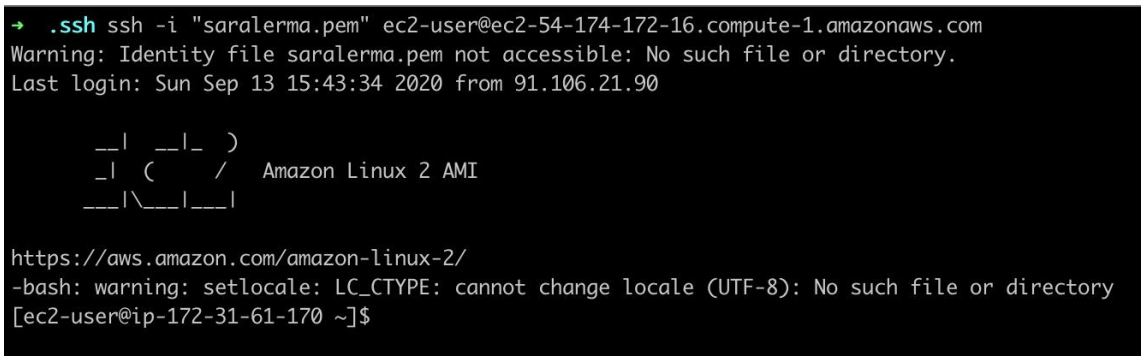


Figura 76 - Conexión a servidor AWS desde la terminal

Para introducir el proyecto dentro del servidor, como se ha utilizado Git como control de versiones de código, se debe instalar Git y clonar el proyecto como se especifica en los siguientes comandos.

- `sudo yum install git -y`
- `git clone https://github.com/SaraLerma/programmer.git`

Posteriormente, se instalan las dependencias como se muestra en la Figura 77, con el comando `npm install` dentro de la carpeta “frontend”, donde se aloja el código relacionado con la página web.

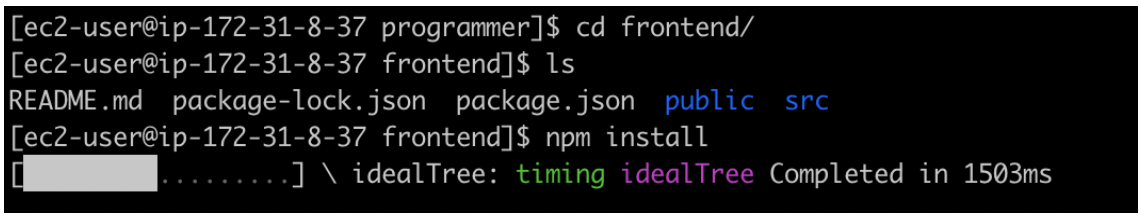


Figura 77 - Instalación dependencias en el servidor con npm install

Una vez instaladas las dependencias, se necesita ejecutar el siguiente comando

- `npm build`

Al ejecutar dicho comando se transforma todo el código de la página web react en ficheros HTML, CSS y Javascript en el directorio “build” para poder ser servidos por el servidor mediante el fichero “index.html”.

Es necesario un servidor web, llamado nginx, para poder acceder a dichos ficheros HTML, CSS, Javascript de forma externa al servidor mediante una dirección IP y el puerto 80. Para la instalación de nginx se debe ejecutar el siguiente comando:

- `amazon-linux-extras install nginx1.12`

Una vez instalado *nginx*, se debe configurar para habilitar el puerto 80 mediante el fichero “nginx.conf” que se muestra a continuación. En la configuración se inicia un servidor web en el puerto 80 y se define donde se encuentra el directorio “root” al que se debe dirigir cuando se accede a la localización raíz “/”.

```
server {
  listen 80;
  location / {
    root   /usr/share/nginx/html;
    index  index.html index.htm;
  }
}
```

El fichero “nginx.conf” debe estar ubicado en el directorio `/etc/nginx/conf.d/`

- `sudo cp nginx.conf /etc/nginx/conf.d/`

Una vez *nginx* está configurado se debe copiar el contenido del directorio “build” en el que se encuentra la página web en el directorio “root” que se ha configurado en *nginx*.

- `sudo cp -r build/* /usr/share/nginx/html`

Por último, se debe inciar el servidor web *nginx*, para servir la página web, con el siguiente comando.

- `sudo nginx -g 'daemon off;'`

Tras iniciar el servidor se introduce la URL en el navegador y se puede observar la página web realizada como se muestra en la Figura 78.

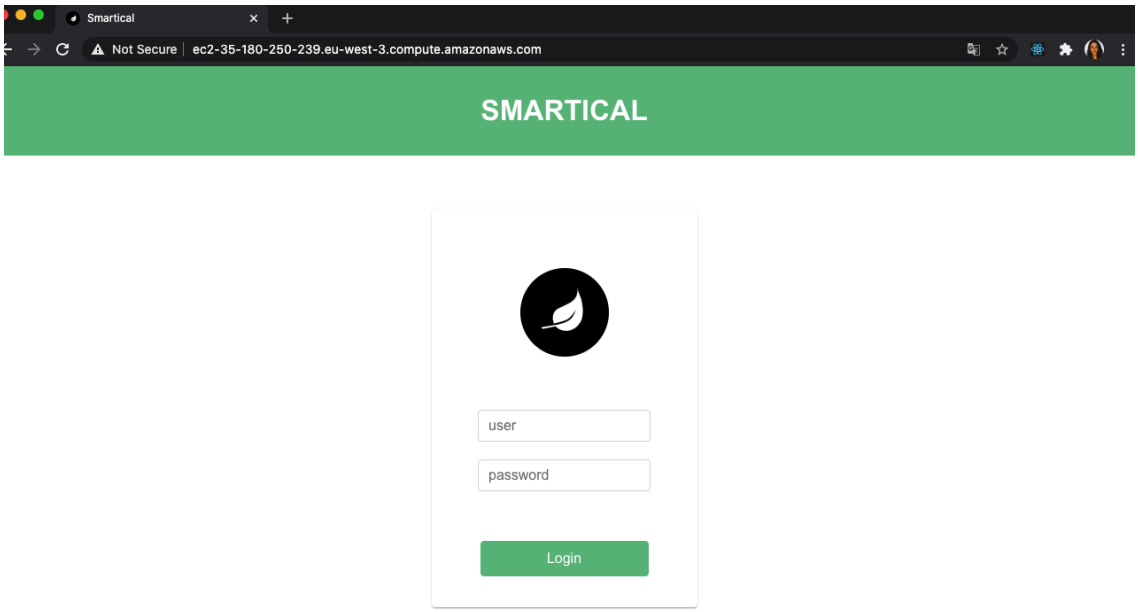


Figura 78 - Página web desplegada con AWS

5.1.4 Backend

Para la realización de la aplicación *backend* que se encarga de toda la gestión de datos en la nube y almacenamiento de datos en una base de datos, se ha utilizado Strapi, que crea automáticamente una API REST de las tablas de la base de datos, las cuales se crean mediante una página web para administradores de la plataforma Strapi. Está realizada en Node.js y se utiliza y configura cómo cualquier aplicación Node con Javascript. La diferencia es que el código necesario para la generación de código mediante una página web ya está realizado, cómo es la gestión de usuarios, el inicio sesión, registro, manejo de *token* de autenticación, etc. Además, Strapi permite extender todo su comportamiento mediante código, y poder añadir *WebSockets*.

I. Strapi

Para instalar Strapi y añadirlo al proyecto, se realiza mediante la línea de comandos con el siguiente comando:

- `npx create-strapi-app backend --quickstart`

Con esta línea de comandos se descarga el proyecto Strapi y se modifica el nombre a “backend”. Una vez el proceso termina, se configura automáticamente con una base de datos embebida llamada SQLite, que es la base de datos utilizada en este proyecto y está alojada en el servidor, en el mismo lugar que Strapi. En la figura 79 se muestra la estructura de Strapi generado.

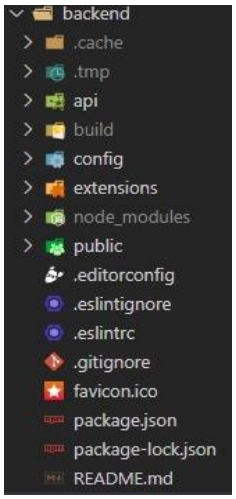


Figura 79 - Estructura de proyecto Strapi

Para iniciar Strapi en modo desarrollo se utiliza el siguiente comando:

- `npm run develop`

Dicho comando inicia Strapi en modo desarrollo y muestra en "`localhost:1337/admin`" la página web de administrador, dejando modificar las tablas de la base de datos y generando nuevo código.

La primera vez que se inicia, como se muestra en la Figura 80, Strapi pregunta por añadir un usuario Administrador.

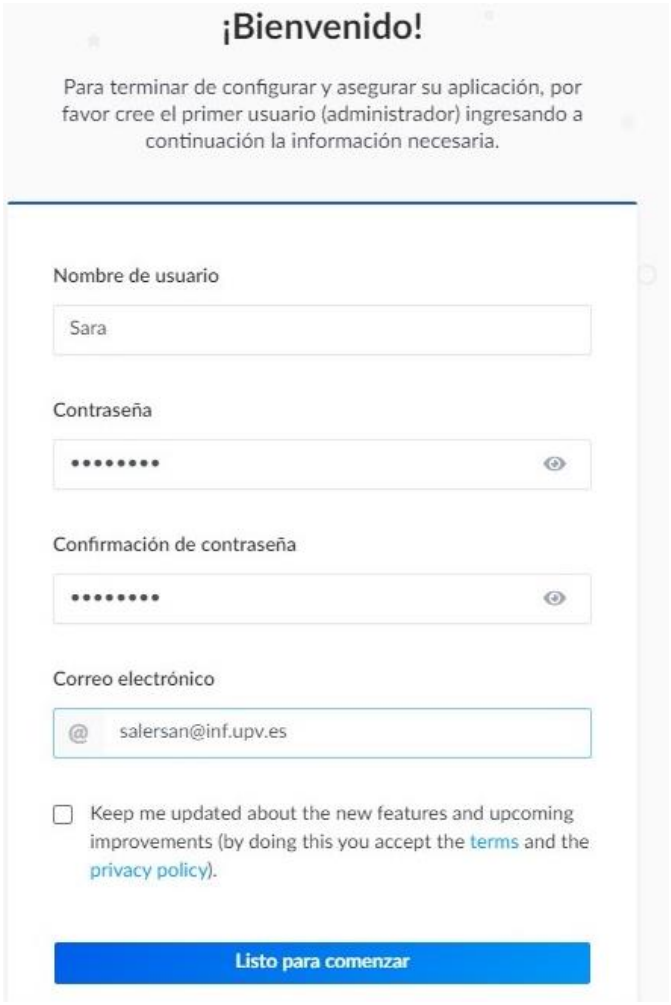


Figura 80 - Registro de administrador en Strapi

Tras haberlo creado, se puede acceder a la página de administración de Strapi. Una vez dentro, como se explica en la Figura 81, se pulsa en la pestaña "Content-Types Builder", la cual sólo aparece en modo desarrollador y se crean las tablas de la base de datos, la tabla usuario ya está creada.

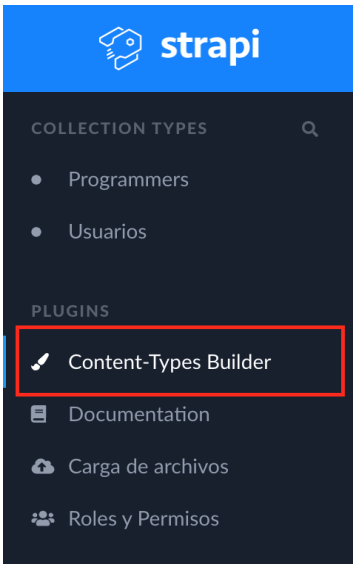


Figura 81 - Creación de tablas en Strapi

Se crea una única tabla llamada “Programmer” la cual contiene toda la información necesaria para mostrar el programador en la web, tener la información de los estados de las zonas y relacionarla con un usuario registrado.

De esta forma al crear la tabla “Programmer”, automáticamente se genera todo el código necesario para poder acceder a él mediante peticiones HTTP con una API REST. Como se muestra en la Figura 82, Strapi tiene un apartado de documentación en el que muestra una página web Swagger en la que se puede ver todos los *endpoints* que tiene el *backend* Strapi. Cómo se puede observar en la documentación, se disponen de todas las peticiones HTTP (GET, PUT, POST, etc.)

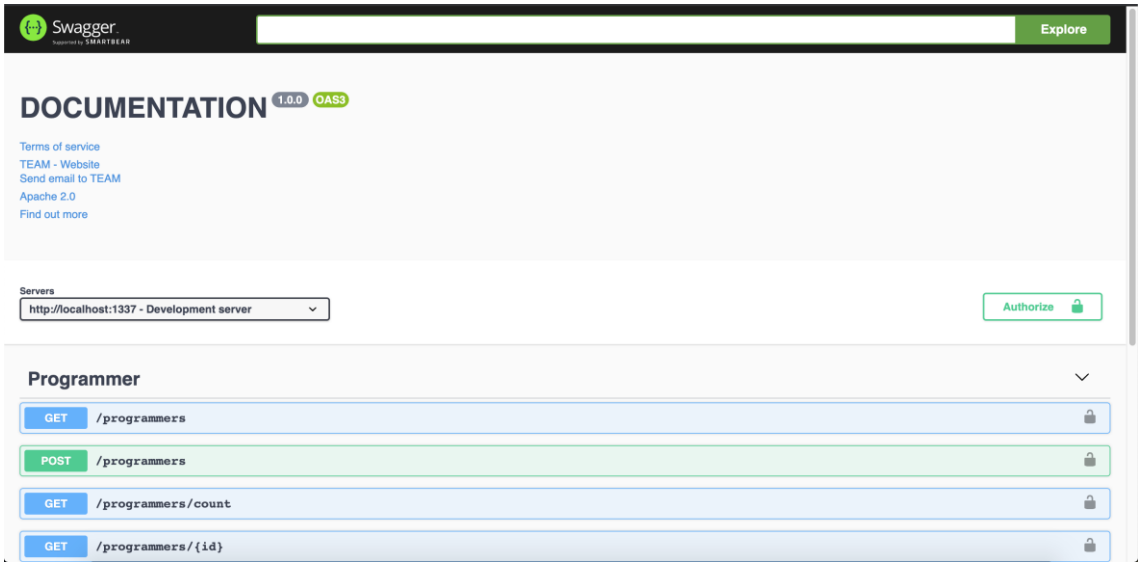


Figura 82 - Página web Swagger orecida por Strapi

Gracias a Strapi se puede realizar todo el desarrollo completo del servidor que llamará la web de una manera fácil y rápida.

II. WebSockets

La parte más complicada es la realización de conexiones mediante *Websockets* para poder actualizar en tiempo real los estados del programador. Anteriormente se ha utilizado *Websockets* para la comunicación en tiempo real entre la página web embebida y el módulo WiFi Wemos, por lo que se utilizan de la misma manera para integrarlo con Strapi.

Para poder integrar *Websockets* con Strapi, hay un fichero llamado “bootstrap.js” en “config/functions”, en el cual Strapi proporciona para poder iniciar recursos o añadir capacidades a el servidor NodeJS de Strapi, en este caso se va a añadir *Websockets* al servidor NodeJS HTTP.

Como se observa en el código de a continuación, el cual se ha basado en el código que especifica la librería SocketIO [36], con “`socket.on`” se pueden definir los canales de comunicación que se utilizan como *topics*. En este caso, cuando un cliente de SocketIO se conecte al servidor, será recibido en el método “`io.on(“connection”)”`. Seguidamente de las definiciones de los *topics* que puede recibir, como “`disconnect`” para notificar de la desconexión del cliente y “`data`” cuando el cliente envía datos al servidor.

De esta manera, se puede añadir a “`strapi.server`” un punto de acceso para los *Websockets*.

```
module.exports = async () => {
    const io = require('socket.io')(strapi.server);
    let devices = [];
    let users = [];
    // listen for user connection
    io.on('connection', function(socket) {
        socket.on('disconnect', () => {
          console.log(“Client disconnect”);
        });
        socket.on('data', async (data) => {
          console.log(“Client sends: “+ data);
        });

    strapi.io = io; // register socket io inside strapi main object to
use it globally anywhere
    console.log("Registered");
  };
}
```

El servidor de *Websockets* espera dos tipos de clientes, Usuario y Dispositivo, éste último es el programador.

- Conexión tipo **Usuario** se refiere a todas aquellas conexiones de la página web necesarias para accionar las válvulas y ver la información desde la web.
- Conexión tipo **Dispositivo** se refiere a todas aquellas conexiones que las realiza el programador directamente al servidor cuando tiene conexión a internet, éste es el encargado de enviar la información de los sensores.

Para saber diferenciar las conexiones entre los diferentes tipos se ha utilizado “query params” para añadir información relevante dependiendo de si es un usuario o un dispositivo. Por ejemplo, si es una conexión tipo Usuario, es necesario conocer el *token* del usuario para saber si tiene autorización para dicha información. Mientras que, por ejemplo, si es una conexión tipo Dispositivo, es necesario conocer el ID del programador para poder relacionarlo con el usuario. Por lo tanto, si una conexión envía un *token* es un Usuario y si envía un ID es un dispositivo, todas estas comprobaciones se realizan a nivel de autenticación en SocketIO al inicio de la conexión HTTP.

De esta manera cuando se inicia una nueva petición se puede obtener la información de la conexión para saber si se debe continuar con la conexión o cerrar la conexión ya que no está autorizado.

```
io.use(async (sock, next) =>{
      if(isDevice(sock.handshake)) {
        validDevice(sock, next);
      } else {
        validUser(sock, next);
      }
    })
```

Cuando se recibe una nueva petición, para saber si es una conexión de tipo Dispositivo, se comprueba que está enviando el ID del programador en la base de datos. Se ha puesto el nombre “deviceId” a la variable de “query params” para el ID del dispositivo.

```
const isDevice = (handshake) => (
  handshake.query && handshake.query.deviceId
);
```

Para validar finalmente un Dispositivo y ver que es válido, se recoge ese ID del programador y se comprueba que efectivamente se encuentre en el sistema.

```
const validDevice = async(sock, next) => {
  try {
    if(sock.handshake.query && sock.handshake.query.deviceId) {
      const result = await strapi.query('programmer').findOne({ id: sock.
handshake.query.a});
      if(result !== null) {
        sock.isDevice = true;
        sock.deviceId = sock.handshake.query.deviceId;
        next();
      } else {
        next(new Error('Device Authentication error'));
      }
    }
  }catch(err) {
    next(new Error('Device Authentication error'));
  }
};
```

Para validar un Usuario y comprobar que es válido, se ha basado en el código que aporta Strapi para validación JWT [37]. Se coge el *token, y* dentro de la información del *token* está el ID del usuario y se comprueba que dicho usuario existe en el sistema.

```
const validUser = async (sock, next) => {
  if(sock.handshake.query && sock.handshake.query.deviceId && sock.handsh
ake.query.token) {
    try {
      const { id, isAdmin = false } = await strapi.plugins[
        'users-permissions'
      ].services.jwt.verify(sock.handshake.query.token);
      sock.isDevice = false;
      sock.userId = id;
      sock.deviceId = sock.handshake.query.deviceId;
      next();
    } catch (err) {
      next(err);
    }
  } else {
    next(new Error('Authentication error'));
  }
};
```

En ambos casos si todo es correcto, es necesario llamar a la función "`next`" de SocketIO para continuar con la conexión, de lo contrario se llama a "`next`" con un error.

Una vez comprobado si es una conexión tipo Usuario o Dispositivo se realizan diferentes acciones.

Una conexión tipo Usuario:

- recibe información de su dispositivo conectado
- realiza acciones de encendido y apagado de la válvula de agua una zona en el *topic* "`action`"
- envía nueva información con la humedad mínima y máxima en el *topic* "`action`"

```
socket.on('disconnect', () => {
          users.forEach((user, i) => {
            if(user.userId === socket.userId) {
              users.splice(i, 1);
            }
          });
```

```
        });
        socket.on('action', action => {
          devices.filter(device => device.deviceId === socket.deviceId)
                 .forEach(device => {
                    device.emit('action', action);
                 });
        });

        const connDevs = devices.filter(device => device.deviceId === soc
ket.deviceId);
        users.push(socket);
      }
```

Mientras que una conexión tipo Dispositivo:

- envía los datos de los sensores cada cierto tiempo en el topic “`data`”
- se la envía a su usuario conectado
- espera eventos que pueda realizar el usuario en el topic “`data`”

```
if(socket.isDevice) {
        socket.on('disconnect', () => {
          users.filter(user => user.deviceId === socket.deviceId)
               .forEach(user => {
                 user.emit('ddisconnect', "");
               });
        });

        socket.on('data', async (data) => {
          users.filter(user => user.deviceId === socket.deviceId)
               .forEach(user => {
                 user.emit('data', data);
               });
               if(!includesDevice(devices, socket.deviceId)) {
                connect();
              }
        });
        users.filter(user => user.deviceId === socket.deviceId)
               .forEach(user => {
                 user.emit('dconnect', "");
               });
```

III. Servidor

En la parte de *backend*, al igual que en la parte del *frontend*, se ha utilizado el servicio de EC2 de AWS. Se ha realizado el mismo procedimiento que se ha explicado en el apartado de Despliegue de la página web, tanto en la creación de una instancia nueva como en la conexión a dicha instancia.

Una vez conectados a la nueva instancia y realizados los pasos de instalación de Git explicados anteriormente en la parte *frontend*, se debe clonar el repositorio de Github.

- `git clone https://github.com/SaraLerma/programmer.git`

Posteriormente, se instalan las dependencias como se muestra en la Figura 83, con el comando *npm install* dentro de la carpeta *backend*, donde se aloja el código relacionado con el servidor.

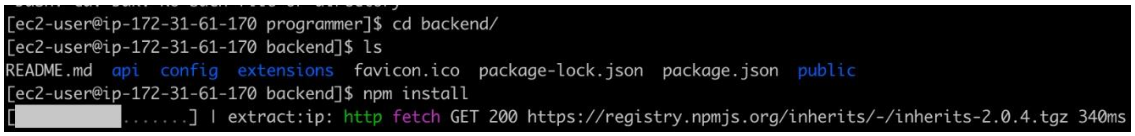


Figura 83 - Instalación dependencias con npm install

Una vez instalada las dependencias, se puede ejecutar el servidor con el siguiente comando

- `npm run start`

Al conectarse con SSH si se utiliza el comando para ejecutar el servidor, dicho proceso dependería de la conexión SSH, por lo que si se desconectase del servidor también terminaría el proceso del programa, por lo que es necesario ejecutar el servidor en segundo plano. Para ello se utiliza un gestor de procesos de Node llamado PM2 [35], que es el recomendado por Strapi.

Por lo que se instala pm2 usando `npm`:

- `npm install pm2@latest -g`

Una vez instalado, es necesario dejar ejecutándose el servidor llamado Strapi en modo producción. De esta manera, como se muestra en la Figura 84, se inicia el servidor en segundo plano.

```
[ec2-user@ip-172-31-50-182 backend]$ NODE_ENV=production pm2 start --name strapi npm -- start
```

Figura 84 - Inicio del servidor en segundo plano

Tras iniciar el servidor se puede introducir la URL creada automáticamente con la instancia EC2 en el puerto 80 y aparecerá, como se observa en la Figura 85, Strapi funcionando.

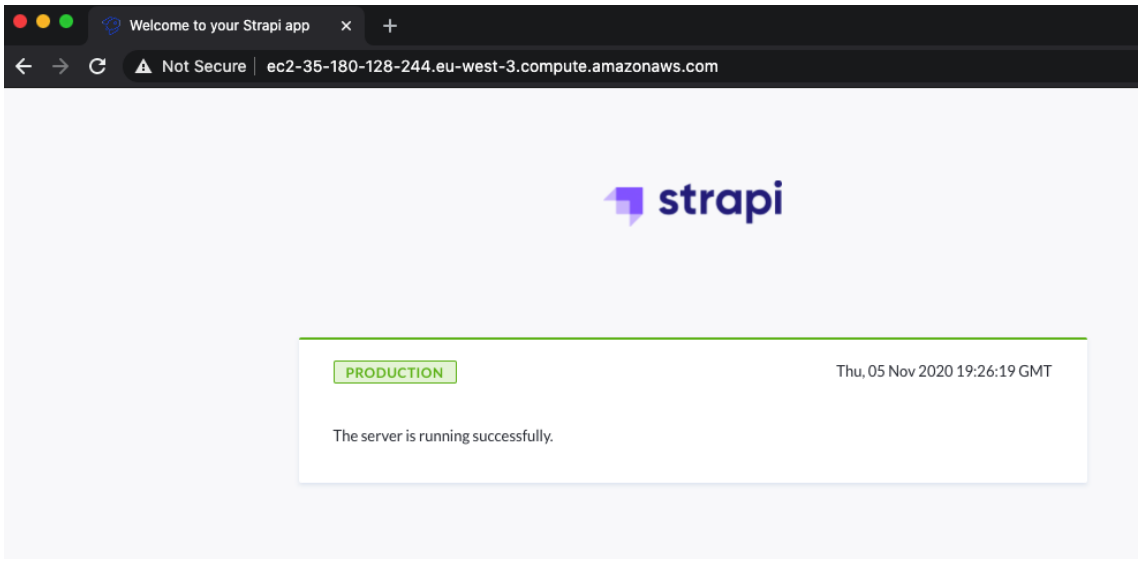


Figura 85 - Página cargada con URL de instancia y puerto Strapi

La primera vez que se inicia el servidor es necesario acceder a “/admin”, como se muestra en la Figura 86, para registrar el usuario Administrador. Toda la información se guarda en la base de datos embebida SQLite en el propio servidor.

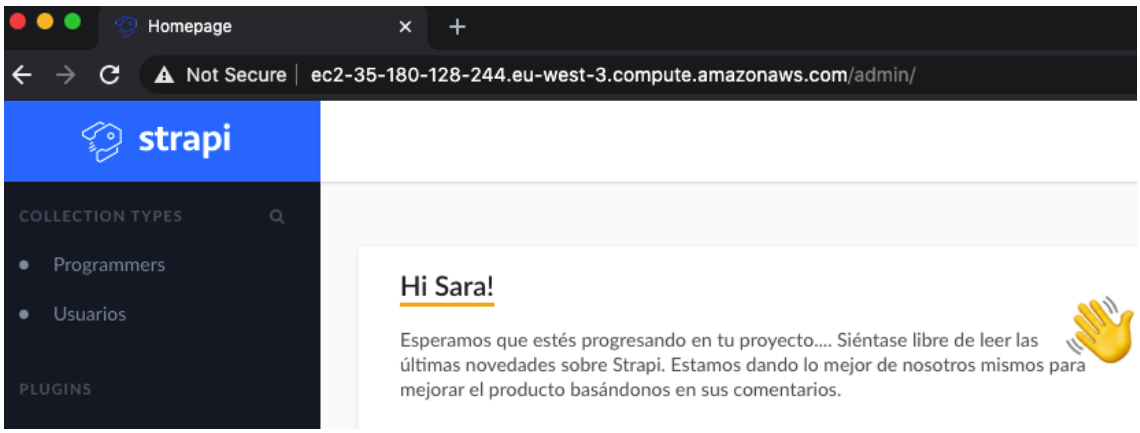


Figura 86 - Registro usuario administrador

5.2 Desarrollo hardware

En este apartado se muestra detalladamente la interconexión de los elementos electrónicos, así como el montaje de todo el conexionado sobre la maqueta de jardín vertical.

5.2.1 Conexionado hardware

A continuación, se explica cómo se ha realizado la conexión de los componentes electrónicos y dispositivos que se han utilizado en el proyecto.

5.2.1.1 Entradas

Conexión sensor temperatura

El sensor de temperatura dispone de dos cables, uno de ellos debe ir conectado a un pin de 5VCC del Arduino, como puede observarse en la Figura 87, que se encuentra representado por el cable rojo, mientras que el otro cable del sensor debe conectarse a una patilla de la resistencia, y ésta a una entrada analógica del Arduino, esta conexión está representada mediante el cable azul. Por último, la otra patilla de la resistencia debe conectarse con el pin GND de Arduino, representado con el cable negro.

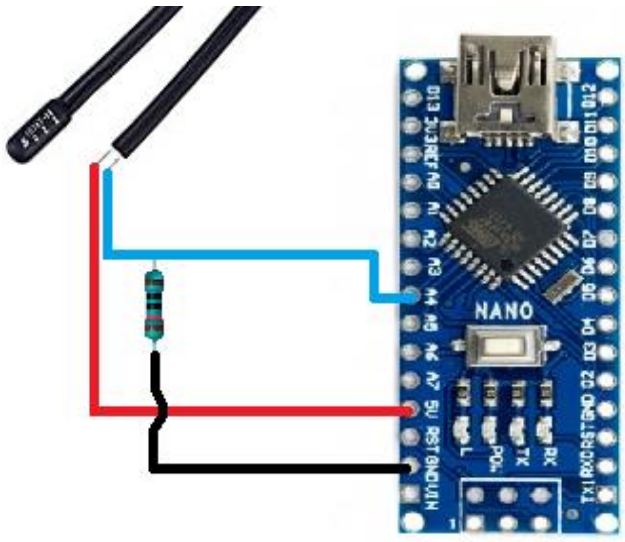


Figura 87 - Conexión sensor de temperatura con Arduino

Conexión sensor humedad de suelo

Respecto al sensor de humedad de suelo SoilWatch, dispone de 3 cables: VCC, GND y data. Como se comentó anteriormente, el sensor SoilWatch necesita una alimentación de entre 3.1 y 5.0V, por lo que, como se muestra en la Figura 88, se conecta el cable marrón VCC con el pin de 3.3V del Arduino. El cable verde GND se conecta a la potencia negativa, por lo que se conecta a uno de los pines de tierra del Arduino, denominado como *Ground*. Por último, el cable blanco Output, en la Figura 90 representado en color azul, es el cable por el que se transmite los datos de los cuales posteriormente se obtiene la humedad del suelo, por lo que se conecta a uno de los pines de entradas analógicas del Arduino.

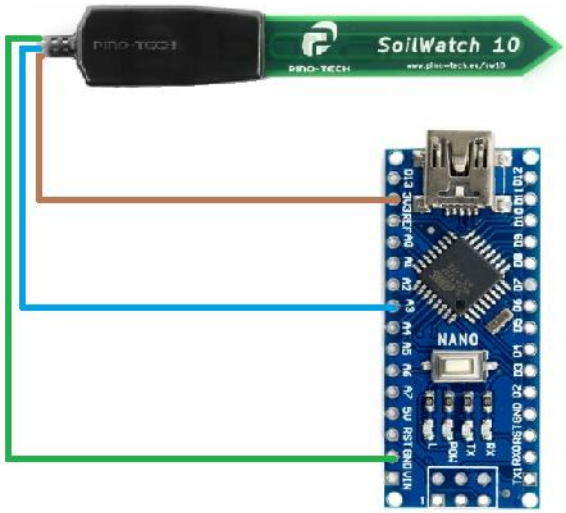


Figura 88 - Conexión sensor de humedad de suelo con Arduino

5.2.1.2 Salidas

En este apartado se expone la conexión del módulo Arduino con la válvula de agua mediante los relés. Cabe destacar previamente al desarrollo del conexionado de la válvula de agua, el funcionamiento del relé utilizado en el proyecto así como sus contactos. El relé utilizado está compuesto por una bobina la cual deja pasar la corriente o no a las válvulas de agua en función si el relé recibe una pequeña corriente, en este caso de 5 VDC, del Arduino. En cuanto a los contactos, el relé dispone de cuatro contactos, como se puede observar en la Figura 89, COM, COIL_1, NO Y COIL_2. A continuación se explica su conexión con el resto de los componentes.

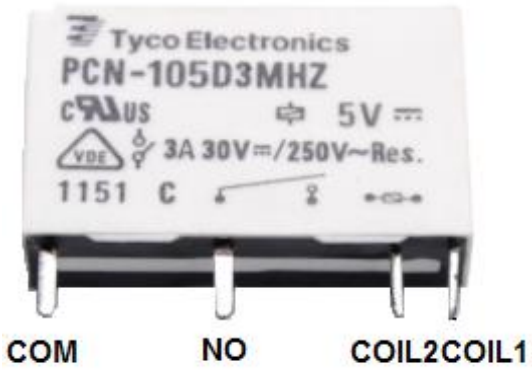


Figura 89 - Información pastillas relé

Conexión válvula de agua

La conexión de la válvula de agua se puede desglosar en dos partes, una parte donde se explica el conexionado de la válvula con la placa de relés y otra parte en la que se presenta el conexionado del relé con el módulo Arduino. Respecto a la conexión de la válvula de agua con el relé, la válvula de agua dispone de dos terminales, como se observa en la Figura 90, uno de los terminales se conecta al neutro del enchufe, representado en la figura por el cable azul, y el otro terminal se conecta al contacto COM del relé que permite el paso de la corriente a la válvula de agua, en la figura representa el cable marrón. La fase del enchufe se conecta con el NO del relé. En cuanto a la conexión del relé con el Arduino, el pin COIL1 se destina a la entrada de alimentación del relé y se conecta a una salida digital del Arduino de 5V y el pin COIL2, que representa GND, se conecta con un pin GND del Arduino. De esta forma al activar la salida digital del Arduino, se excitará la bobina haciendo cambiar de estado el contactor. Se ha puesto un diodo antiparalelo a la bobina para la descarga de ésta.

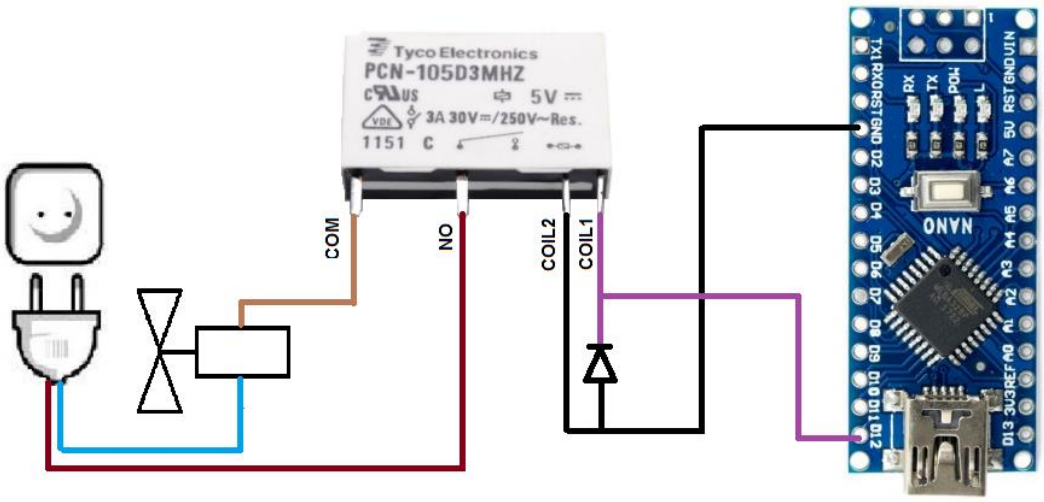


Figura 90 - Conexión válvula de agua con relé y relé con Arduino

5.2.1.3 Internet

Conexión Arduino y Wemos

Las placas Arduino nano y Wemos D1 mini pro intercambian información mediante el protocolo de comunicación Serial. En la Figura 91 se muestra la conexión necesaria entre ambas placas, el cable verde representa la conexión en la que el pin digital 4 de Arduino se comporta como transmisor y el pin digital 5 de Wemos se comporta como receptor, mientras el cable azul representa la conexión en la que el pin digital 3 de Arduino se comporta como receptor y el pin digital 6 de Wemos se comporta como transmisor. Por último, el GND debe ser común, por lo que, como se observa con el cable negro, se debe conectar el GND del Wemos con el GND del Arduino.

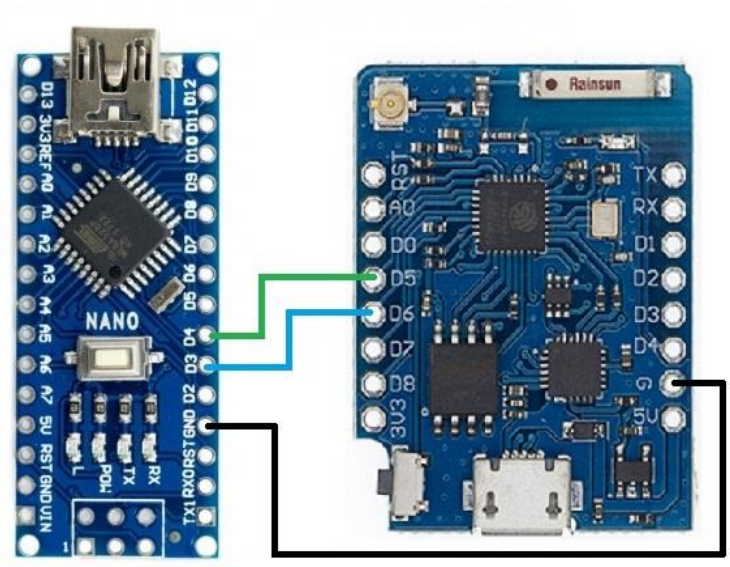


Figura 91 - Conexión Wemos con Arduino

5.2.2 Soldadura

En este apartado se explica cómo se han soldado todos los componentes electrónicos anteriormente comentados en el PCB y como se han interconectado todos los elementos electrónicos; los sensores, actuadores y programador.

Una vez recibido el circuito impreso, el cual se muestra en la Figura 92,

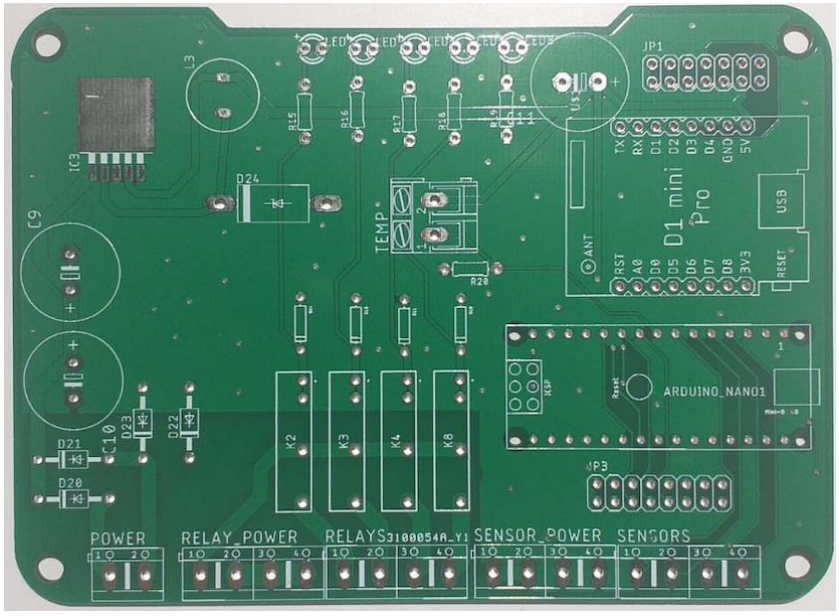


Figura 92 - Circuito impreso sin los componentes soldados

se procede a soldar cada componente en el sitio que se había especificado mediante el diseño del circuito y finalmente se obtiene la placa soldada como se muestra en la Figura 93.

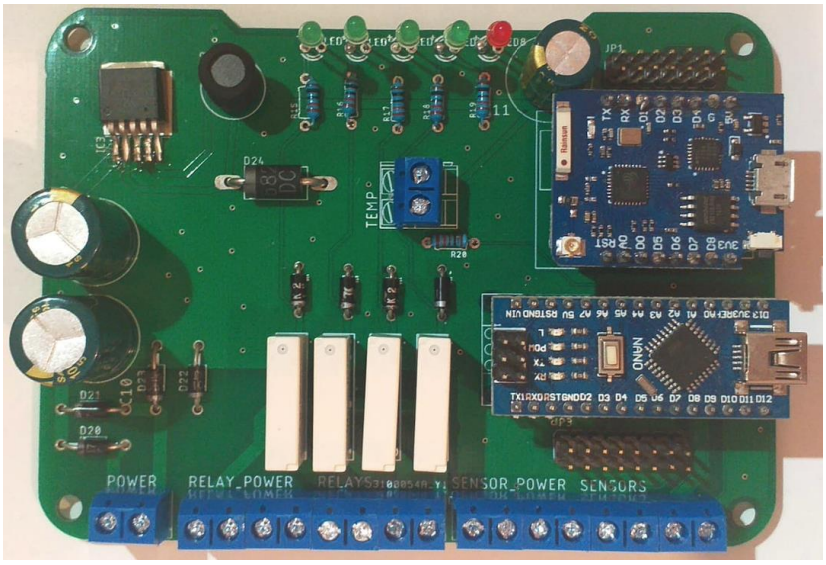


Figura 93 - Circuito impreso con los componentes soldados

5.2.3 Montaje hardware

En primer lugar, para realizar la maqueta se ha realizado un soporte de madera. En dicho soporte de madera se colocan los bolsillos del jardín vertical como se muestra en la Figura 94.

Figura 94 - Soporte con bolsillos

Posteriormente se colocan las válvulas de agua conectadas con los tubos de riego por donde pasará el agua en el soporte de madera como se muestra en la Figura 95.

Figura 95 - Soporte con válvulas de agua y tubos de riego

En la parte superior, como se observa en la Figura 96, se coloca el programador para poder ver el estado de los leds, que representan la activación y la desactivación de las válvulas de agua. Y se realiza la conexión como se especificó en el apartado de Conexionado hardware. Para esta maqueta se han colocado tres válvulas de agua, una por cada fila del jardín vertical, pero el sistema creado se puede utilizar hasta cuatro válvulas de agua.

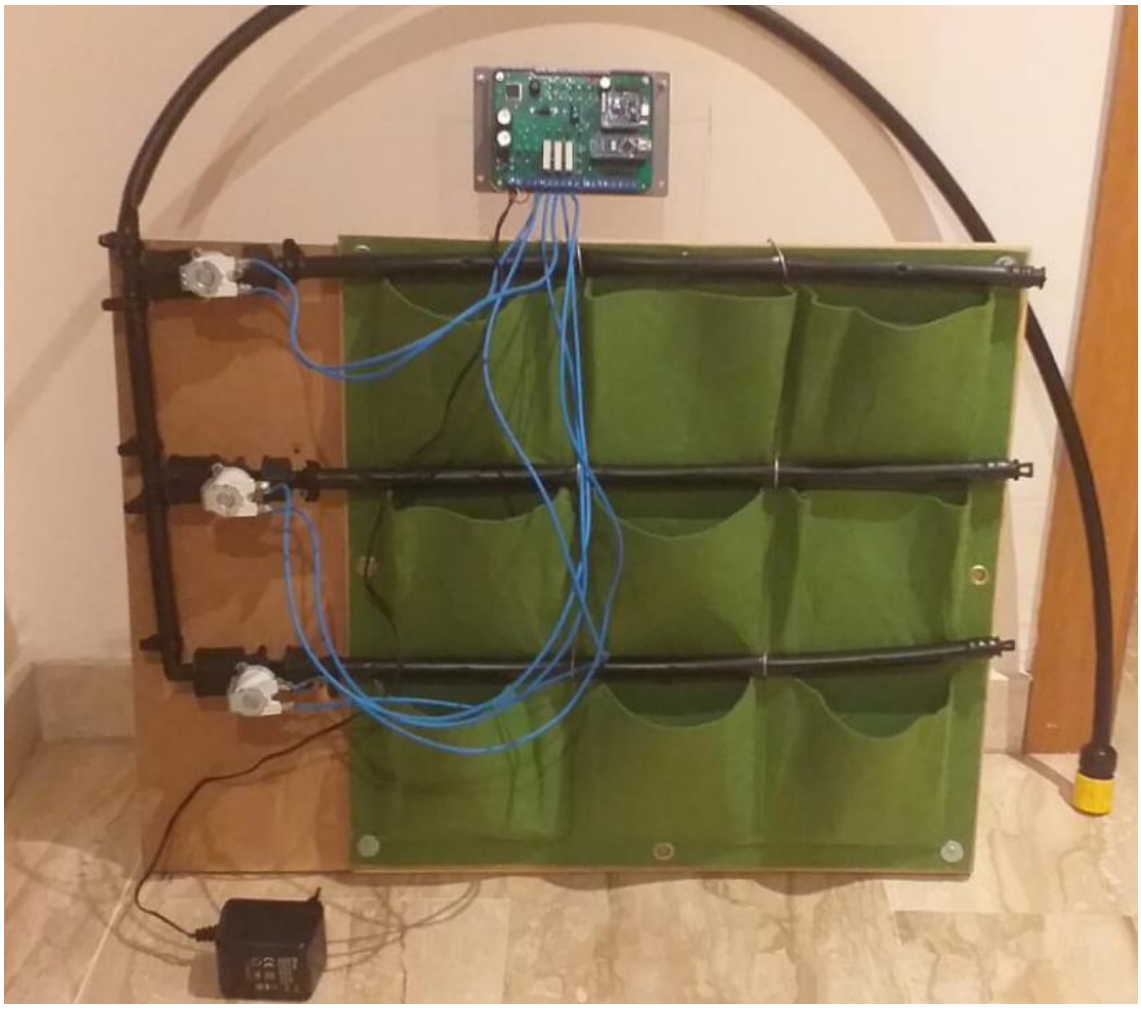


Figura 96 - Soporte con programador conectado a las válvulas de agua

A continuación, se conectan los sensores a las bornas correspondientes del programador. Y finalmente, se coloca el abono junto con las plantas y se introducen los sensores de humedad de suelo en cada zona.

Figura 97 - Montaje hardware final

6 Implantación

En este apartado se van a explicar las configuraciones que se deben realizar previamente al uso del sistema inteligente de riego. Para realizar la puesta en marcha de dicho sistema se deben realizar por una parte una configuración software, como es la configuración del wifi y el registro del nuevo usuario en la base de datos. Por otra parte, se debe realizar la configuración previa respecto al hardware, como es la conexión entre los dispositivos electrónicos y la colocación correcta de los mismos en el jardín vertical.

6.1 Configuración software

6.1.1 Configuración del wifi

Para conectar el sistema a una red wifi del lugar donde se sitúa el jardín vertical, es necesario disponer de un dispositivo que tenga señal wifi ya que el módulo Wemos inicialmente si no tiene ninguna red wifi a la que conectarse, actúa como un punto de acceso generando su propia red Wifi. Por lo que en primer lugar se debe encender el programador, y éste crea un punto de acceso de forma que el usuario debe conectarse a la red ofrecida por el programador. Una vez conectado a la red del programador, el usuario debe introducir como URL en el navegador la dirección IP 192.168.4.1 y aparece una página web como la que se muestra en la Figura 98 donde se puede introducir la red Wifi y su contraseña.

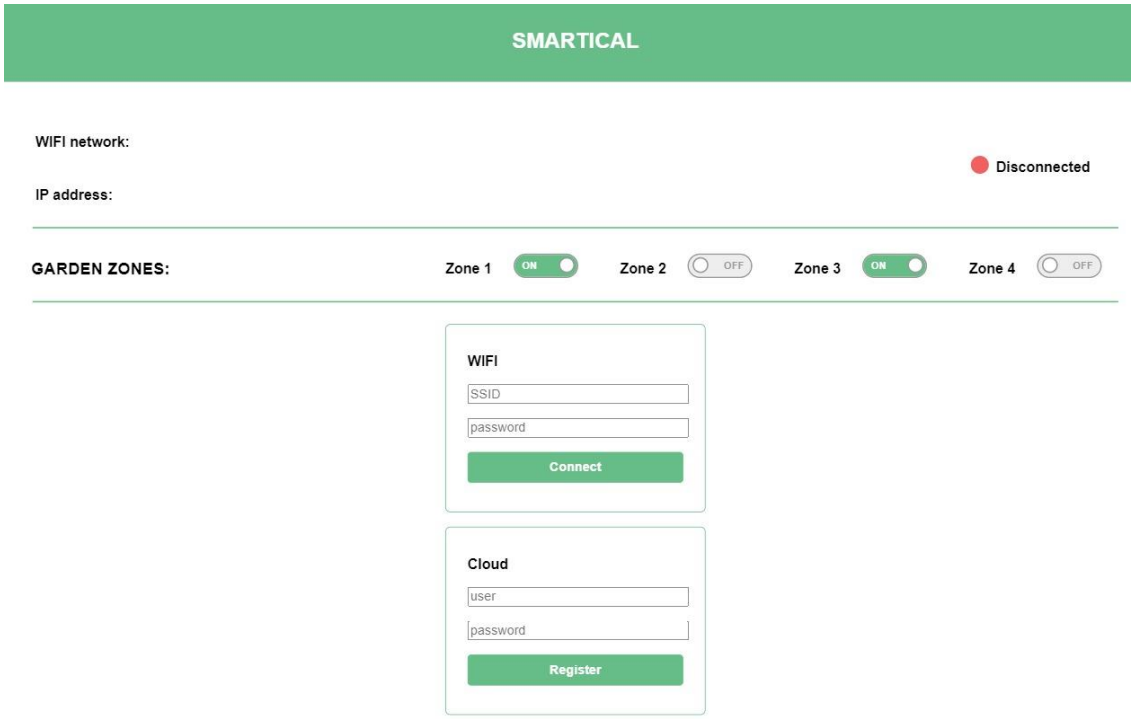


Figura 98 - Página web embebida

Tras conectarse a la nueva red wifi, el punto de acceso se destruirá y el usuario podrá iniciar sesión en la página web principal accediendo a la página *http://ec2-35-180-250-239.eu-west-3.compute.amazonaws.com/* y ver la información de su programador. De esta forma se consigue que el programador se conecte a la red Wifi del lugar donde se sitúa el jardín vertical.

6.1.2 Registro nuevo usuario en Strapi

Para que el usuario pueda acceder a la información de su programador debe iniciar sesión en la página web. Para ello el administrador le debe registrar previamente en Strapi en la tabla de usuarios, por lo que en la pestaña Usuarios, como se especifica en la Figura 99, se pulsa el botón “Agregar nuevo user”.

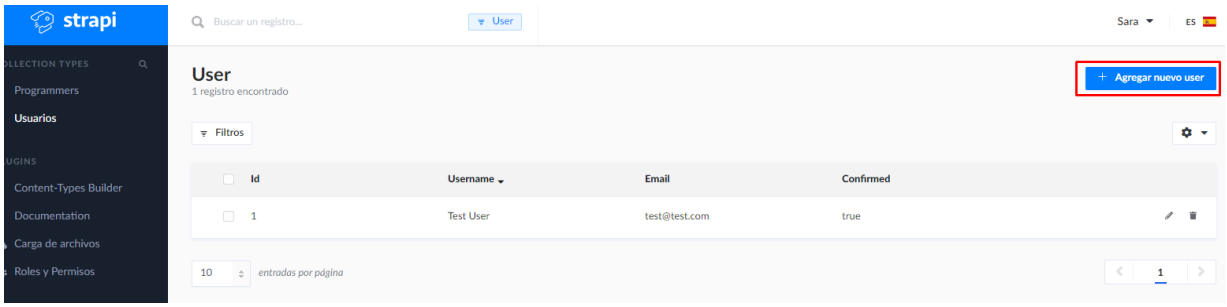


Figura 99 - Agregar nuevo usuario Strapi (I)

A continuación, se muestra un formulario, como el de la Figura 100, en el que se debe especificar el nuevo nombre de usuario, el correo electrónico, la contraseña y los permisos. Y se pulsa el botón “Guardar”.

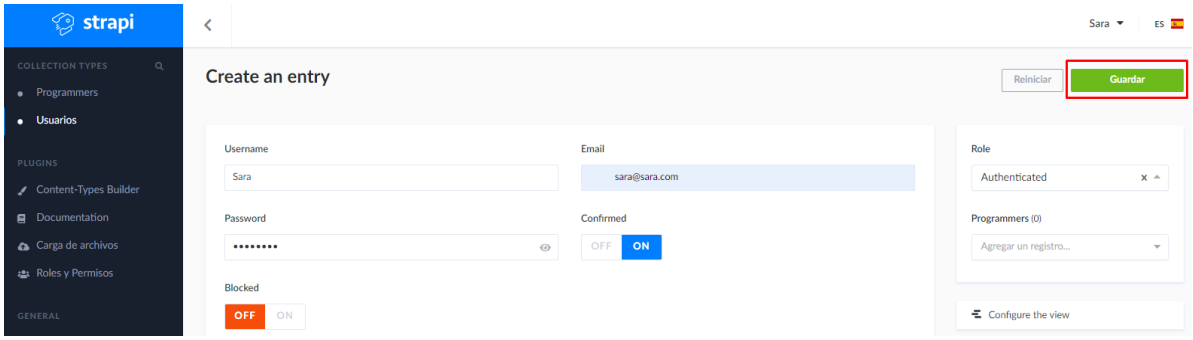


Figura 100 - Agregar nuevo usuario Strapi (II)

Finalmente, el nuevo usuario aparece en la tabla de usuarios como se muestra en la Figura 101 y se vincula a un programador.

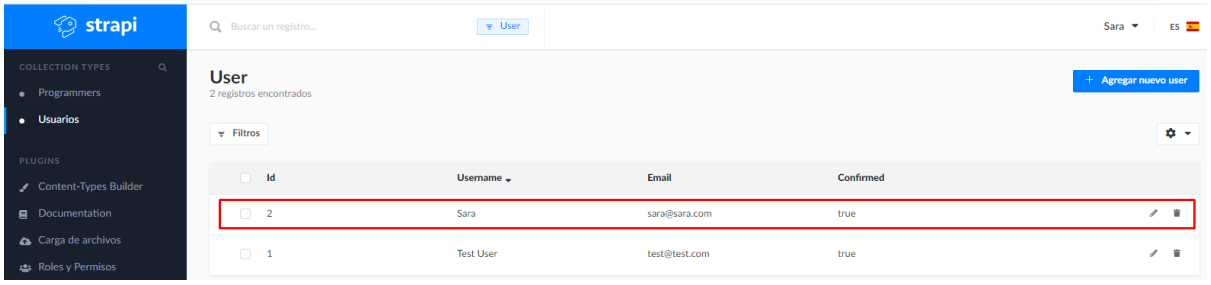


Figura 101 - Agregar nuevo usuario Strapi (III)

6.2 Configuración hardware

En este apartado se muestran una serie de pasos necesarios que se deben realizar para que sea posible el funcionamiento correcto del sistema desarrollado.

6.2.1 Conexión agua

Para poder realizar el riego es necesario realizar la conexión de una toma de agua a las válvulas de agua, como se observa en la Figura 102. Para ello se debe conectar una manguera de la red de agua de la propiedad a la entrada de agua del jardín vertical.

Figura 102 - Conexión agua

6.2.2 Conexión luz

Para que el hardware funcione es necesario conectar a la luz el programador, concretamente, el cable que alimenta a todos los aparatos electrónicos del jardín vertical, constituido por un cable de alimentación y un enchufe macho. Por lo que se necesita una toma de corriente que no diste mucho de la ubicación del programador o que se conecte un alargador, como se observa en la Figura 103, de forma que pueda haber una distancia mayor desde la toma de luz hasta la ubicación del programador. Para ello se debe de suministrar al programador energía eléctrica, 220V CA.

Figura 103 - Conexión luz

7 Conclusiones

Este proyecto tenía como objetivo principal desarrollar un sistema de riego inteligente para jardines verticales que permita la optimización del agua mediante un sistema de irrigación eficiente, así como mejorar la calidad de vida de las plantas ubicadas en el jardín vertical. Por lo que, tanto el objetivo principal como los objetivos específicos que eran informar de la humedad de suelo de cada zona y controlar el riego de cada zona mediante una página web desde cualquier lugar se han cumplido.

Durante el desarrollo han surgido varios inconvenientes, el principal inconveniente fue la imposibilidad de comprar componentes electrónicos presencialmente debido al coronavirus y comprándolos mediante internet tardaban entre 1 o 2 meses. Otro inconveniente fue cuando estuve haciendo pruebas para convertir 24VAC a 5VDC utilizando un limitador lineal de 5 voltios, pero se sobrecalentaba y dejaba de funcionar, por lo que buscando por internet averigüé que se calentaba porque recibía más amperaje del que necesitaba, por lo que había que utilizar un limitador no líneal DC/DC que son más eficientes y tiene mejores prestaciones.

Respecto a los errores cometidos en el proyecto, el principal error cometido fue pensar en un inicio que iba a ser más sencillo y que podría compaginarlo perfectamente con una jornada laboral completa, pero fue duro y “gracias” al confinamiento tuve más tiempo para adelantar y centrarme en el proyecto.

Pese a los problemas y errores cometidos, ha sido una experiencia muy enriquecedora que me ha servido en el ámbito personal como en el ámbito profesional. Respecto al ámbito personal, ha habido muchos momentos frustrantes y finalmente he sabido llevar la situación hacia adelante, he aprendido a gestionar el tiempo utilizando una herramienta para definir tareas subtareas, priorizando las tareas a realizar y dedicándole el tiempo adecuado a cada tarea. Mientras que en el ámbito profesional este proyecto me ha ayudado mucho en mi formación en el campo de electrónica, en el uso de la plataforma Strapi ya que nunca la había utilizado y he observado que con ella se pueden realizar fácilmente una amplia gama de proyectos. Respecto al *software* este proyecto me ha servido a poner en práctica la programación con Websockets, HTTP, programación con *hardware* y la realización de un servicio web alojado en AWS desarrollado con JavaScript. Además, se ha creado una aplicación

web con React y una aplicación web embebida creando un pequeño servidor en C++ para poder acceder al Wemos y configurarlo.

7.1 Relación del trabajo desarrollado con los estudios cursados

En el máster se han dado diversas asignaturas las cuales no están relacionadas directamente con el desarrollo de este proyecto, pero han aportado habilidades y destrezas sobre algunos campos utilizados en este proyecto, como es el caso de la asignatura de “Gestión de proyectos” en la que se ha enseñado diferentes metodologías así como a planificar las tareas dentro de un proyecto y para ello se ha utilizado la herramienta ClubHouse [1]. En cuanto a las asignaturas que han aportado conocimientos directos para el desarrollo de este proyecto son las asignaturas en las que se ha enseñado *websockets* y HTTP, como es el caso de “servicios y aplicaciones distribuidas” y de “redes y seguridad,” entre otras. Sin embargo, la asignatura que tiene relación con la parte hardware del proyecto es “sistemas informáticos para el control de instalaciones y procesos” dado que en esta asignatura se ha explicado desde el principio hasta el final el diseño y la manufacturación del PCB, ya que se ha diseñado desde cero un PCB, se ha realizado el enrutado y se ha explicado cómo se debe enviar a manufacturar, y muchas más cosas, utilizando la herramienta EAGLE. Respecto a la parte del *backend* del proyecto, se ha utilizado AWS dado que en la asignatura “inteligencia ambiental” se ha realizado un proyecto en que se trabajaba con AWS. Con este proyecto se ha logado un objetivo personal que era aprovechar lo máximo posible los conocimientos adquiridos en el máster e integrar todos ellos en un mismo proyecto.

8 Trabajos futuros

Con la realización del presente proyecto se han contemplado muchas ampliaciones así como mejoras, las cuales se podrían haber realizado si se dispusiera de más tiempo y presupuesto. Las ampliaciones que se podría realizar en este proyecto serían las siguientes:

- La incorporación de placas solares que captaran energía de la luz del sol durante el día de manera que se produjera electricidad con la que se alimentaría el programador así como los actuadores.
- El desarrollo de una aplicación móvil en la que se muestre el mismo contenido que la aplicación web desarrollada, de forma que el usuario no tuviese que buscar la página web.
- La incorporación de un sistema de filtrado de agua para reutilizarla.
- Incluir más funcionalidades en la aplicación, así como la posibilidad de especificar una programación de riego a unas horas determinadas, proporcionando al usuario la posibilidad de optar por un modo programado, en vez de modo manual o inteligente. Otra posible funcionalidad podría ser mostrar históricos semanales o mensuales de la humedad de suelo de la zona y la temperatura, aunque fuese meramente informativo.
- La incorporación de un caudalímetro para disponer de la medida de agua utilizada y comprobar el ahorro de agua.

9 Referencias

[1] Clubhouse – herramienta de gestión de proyectos. Disponible en: https://clubhouse.io/

[2] JARDIBIC- sistema de riego. Disponible en: https://www.leroymerlin.es/fp/14096901/programador-jardibric-de-1-vias

[3] Hunter Eco-Logic – sistema de riego. Disponible en: https://www.hunterindustries.com/es/product/programadores/eco-logic

[4] SensoTimer ST6 Duo eco!ogic - sistema de riego. Disponible en: https://www.kaercher.com/es/home-garden/sistemas-de-riego/sistemas-de-riego-automatico/sensotimer-st6-duo-eco-ogic-26452140.html

[5] Fliwer Sense&Water - sistema de riego. Disponible en: https://viagua.es/material-riego/fliwer-sensewater-wifi/

[6] Sensor humedad de suelo SoilWatch. Disponible en: https://pino-tech.eu/wp-content/uploads/2017/08/SoilWatch10.pdf

[7] Sensor de temperatura 103AT-11. Disponible en: https://www.mouser.es/ProductDetail/Semitec/103AT-11?qs=wgO0AD0o1vvXePzqWokBCw==

[8] Relés PCN-105D3MHZ. Disponible en: https://es.farnell.com/te-connectivity/pcn-105d3mhz/rel-spst-no-250vac-30vdc-3a/dp/4444930 http://www.farnell.com/datasheets/1863034.pdf?_ga=2.106002755.1955401885.1585076889-1260896642.1583328412

[9] Válvula de agua Euro Rain. Disponible en: http://www.euro-rain.es/images/stories/eurorain/pdf/rpe_electrovalvula_serie_mini.pdf

[10] Wemos D1 Mini Pro. Disponible en: https://www.electrohobby.es/wifi/348-wemos-d1-mini-pro-16m-0606110073714.html

[11] Arduino nano. Disponible en: https://descubrearduino.com/arduino-nano-pinout/

[12] (12/11/2017) Carlos Alan González Cortez. Aprende electrónica desde nivel básico hasta microcontroladores.

[13] EAGLE. Disponible en: https://www.autodesk.com/products/eagle/overview

[14] Websockets. Disponible en:

https://en.wikipedia.org/wiki/WebSocket#:~:text=WebSocket%20is%20a%20computer%20communications,being%20standardized%20by%20the%20W3C
[15] HTTP. Disponible en: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[16] Preact. Disponible en: https://preactjs.com/
[17] React. Disponible en: https://reactjs.org/
[18] Strapi. Disponible en: https://strapi.io/
[19] Visual Studio Code. Disponible en: https://code.visualstudio.com/
[20] PlatformIO. Disponible en: https://platformio.org/
[21] Arduino IDE. Disponible en: https://www.arduino.cc/en/Main/software
[22] Librería SoftwareSerial. Disponible en: https://www.arduino.cc/en/Reference/softwareSerial
[23] Librería ArduinoJSON. Disponible en: https://github.com/bblanchon/ArduinoJson
[24] Librería JustWifi. Disponible en: https://github.com/xoseperez/justwifi
[25] Librería ESPAsyncWebServer. Disponible en: https://github.com/me-no-dev/ESPAsyncWebServer
[26] Librería ESPAsyncTCP. Disponible en: https://github.com/me-no-dev/ESPAsyncTCP
[27] Librería Embedis. Disponible en: https://github.com/thingSoC/embedis
[28] Librería EEPROM_rotate. Disponible en: https://github.com/xoseperez/eeprom_rotate#0.9.2
[29] Librería SocketIO. Disponible en: https://socket.io/
[30] ConfigDuino - Ejemplo de código de como crear una página web con Preact. Disponible en: https://github.com/madpilot/configduino-sample
[31] Espurna – firmware universal para ESP8266. Disponible en: https://github.com/xoseperez/espurna
[32] Fórmula para calcular la temperatura con un sensor NTC. Disponible en: http://sin.lyceeleyguescouffignal.fr/make-an-arduino-temperature-sensor-thermistor-tutorial
[33] AWS - plataforma de computación en la nube. Disponible en: https://aws.amazon.com/es/
[34] JLCPCB – página web para la impresión de PCBs. Disponible en: https://jlcpcb.com/
[35] Gestión procesos PM2. Disponible en: https://strapi.io/documentation/v3.x/guides/process-manager.html
[36] Autenticación SocketIO. Disponible en: https://socket.io/docs/migrating-from-0-9/#Socket-io-uses-middleware-now
[37] Autenticación JWT con Strapi. Disponible en:

https://strapi.io/documentation/v3.x/guides/jwt-validation.html#customize-the-jwt-validation-function

PRESUPUESTO

[38] Sensor SoilWatch. Disponible en: https://pino-tech.eu/soilwatch10/

[39] Sensor temperatura NTC. Disponible en: *https://www.mouser.es/ProductDetail/Semitec/103AT-11?qs=wgO0AD0o1vvXePzqWokBCw==*

[40] *Relé PCN-105D3MHZ*. Disponible en: *https://es.farnell.com/te-connectivity/pcn-105d3mhz/rel-spst-no-250vac-30vdc-3a/dp/4444930?gclid=EAIaIQobChMIgs_0xcfe6QIVB_hRCh0OvQAvEAAYASAAEgKO9PD_BwE&mckv=scJWtDYQh_dc|pcrid|68846789817|kword|pcn%20105d3mhz|match|p|plid||slid||product||pgrid|11749273137|ptaid|kwd-11451281051|&CMP=KNC-GES-GEN-SKU-MDC*

[41] Limitador 5V. Disponible en: https://es.aliexpress.com/item/32949236530.html?spm=a2g0s.9042311.0.0.274263c0zjfPae

[42] Wemos. Disponible en: https://www.amazon.es/Conector-externa-NodeMCU-ESP-8266EX-desarrollo/dp/B07SQL5C4C?th=1

[43] Arduino nano. Disponible en: https://es.aliexpress.com/item/32993781660.html

Anexos

Manual de usuario

SMARTICAL

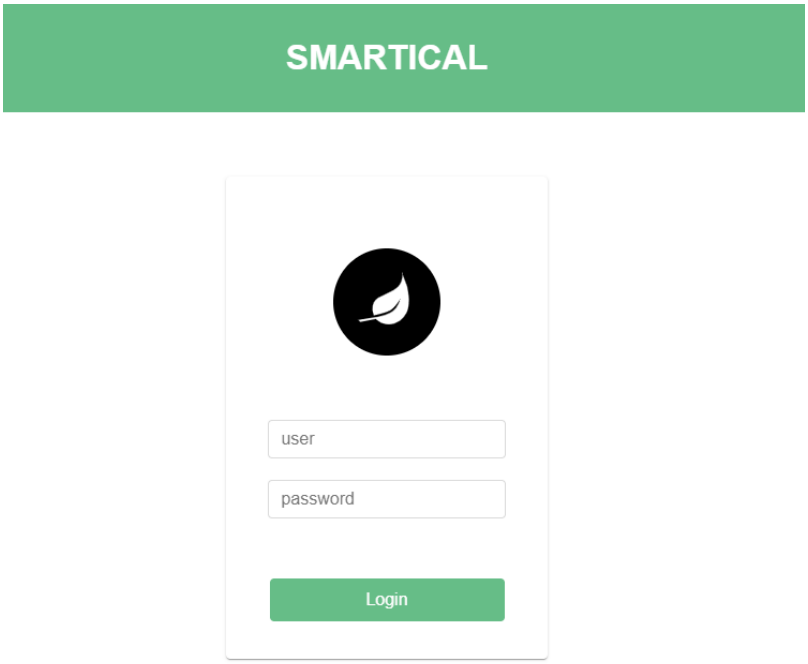
Sistema de riego inteligente para jardines verticales

Tabla de Contenidos

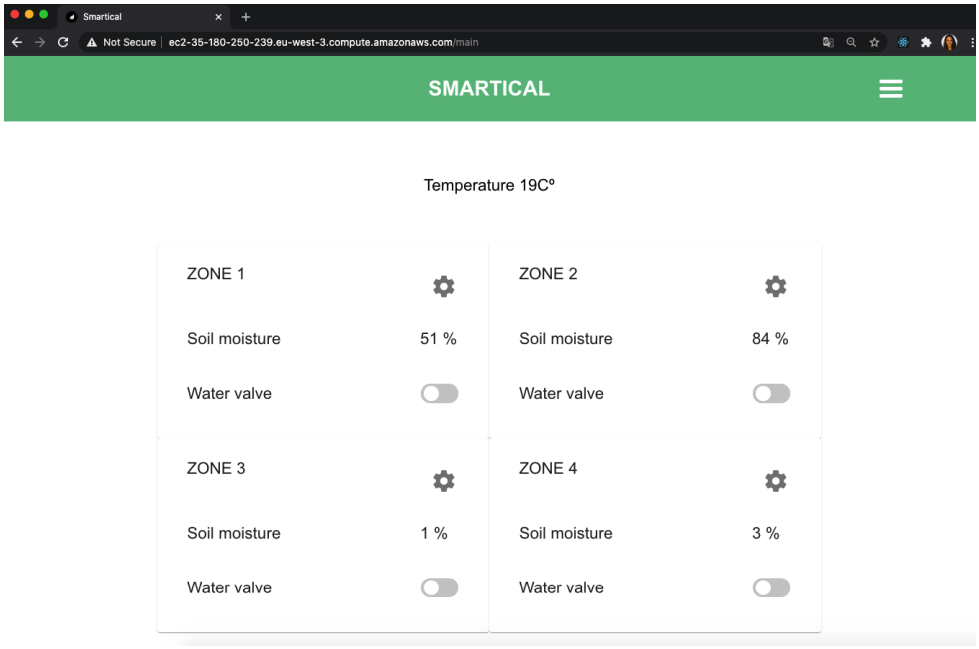
1. Control de jardín vertical

En las siguientes imágenes se muestra la página web que permite el control y la configuración del jardín vertical. A continuación, se explica detalladamente todos los apartados y el funcionamiento de la aplicación.

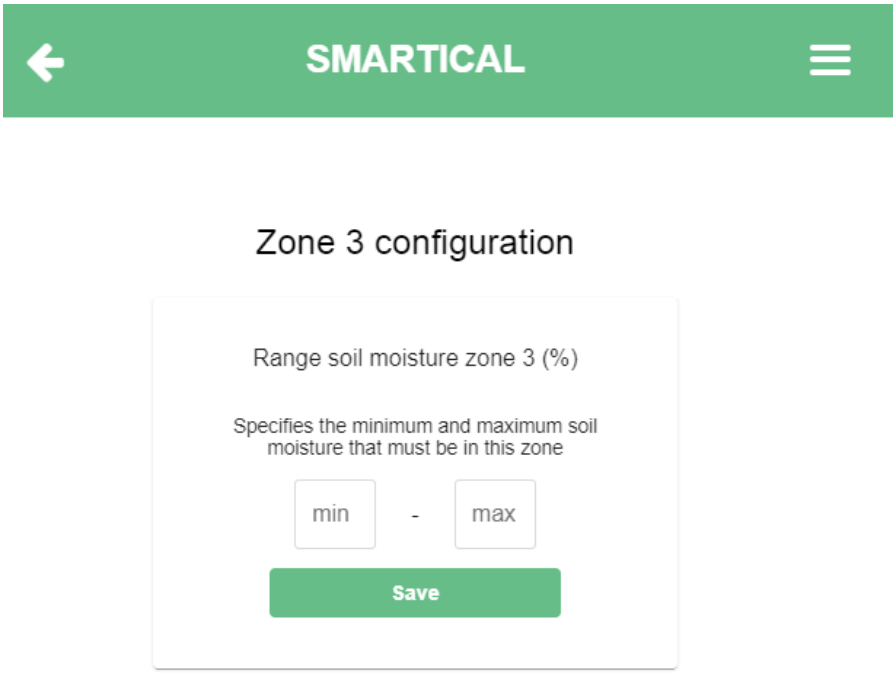
El usuario debe iniciar sesión con un usuario y una contraseña que el administrador le proporcionará.



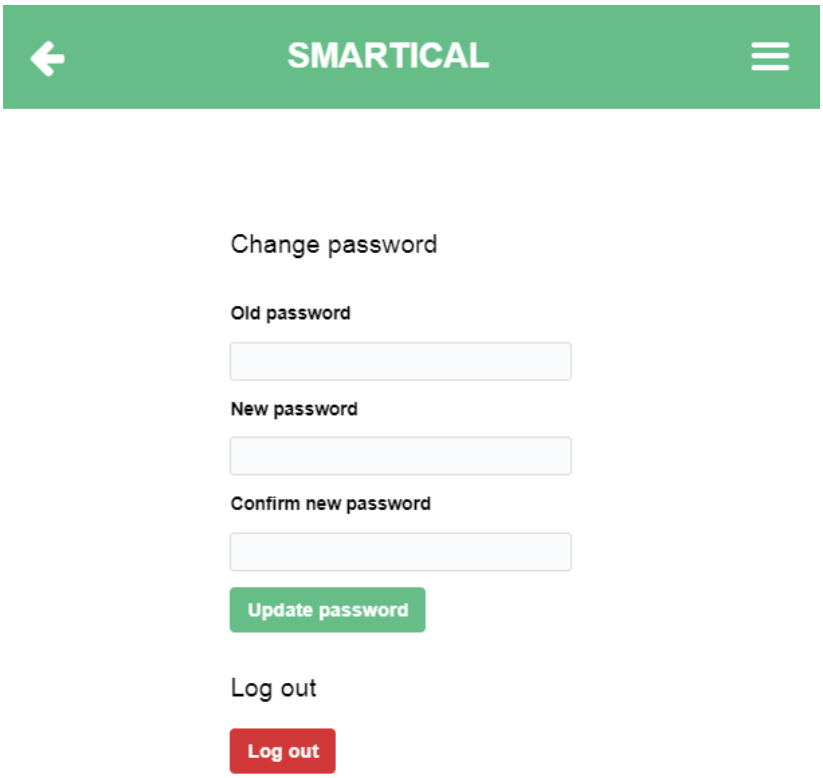
Una vez el usuario inicia sesión, le aparecerá una ventana con toda la información de su programador, es decir, el estado de su jardín vertical.



El usuario puede modificar los puntos de consigna de activación y desactivación de cada zona pulsando en el icono de configuración de la zona correspondiente, y de esta manera aparecerá una ventana como la que se muestra en la siguiente imagen donde debe especificar el rango mínimo y máximo de humedad de suelo que se desea para esa zona. De forma que si la humedad de suelo está por debajo del mínimo se activará la válvula de agua de esa zona y si la humedad de suelo de dicha zona alcanza el máximo especificado, se cerrará la válvula de agua.



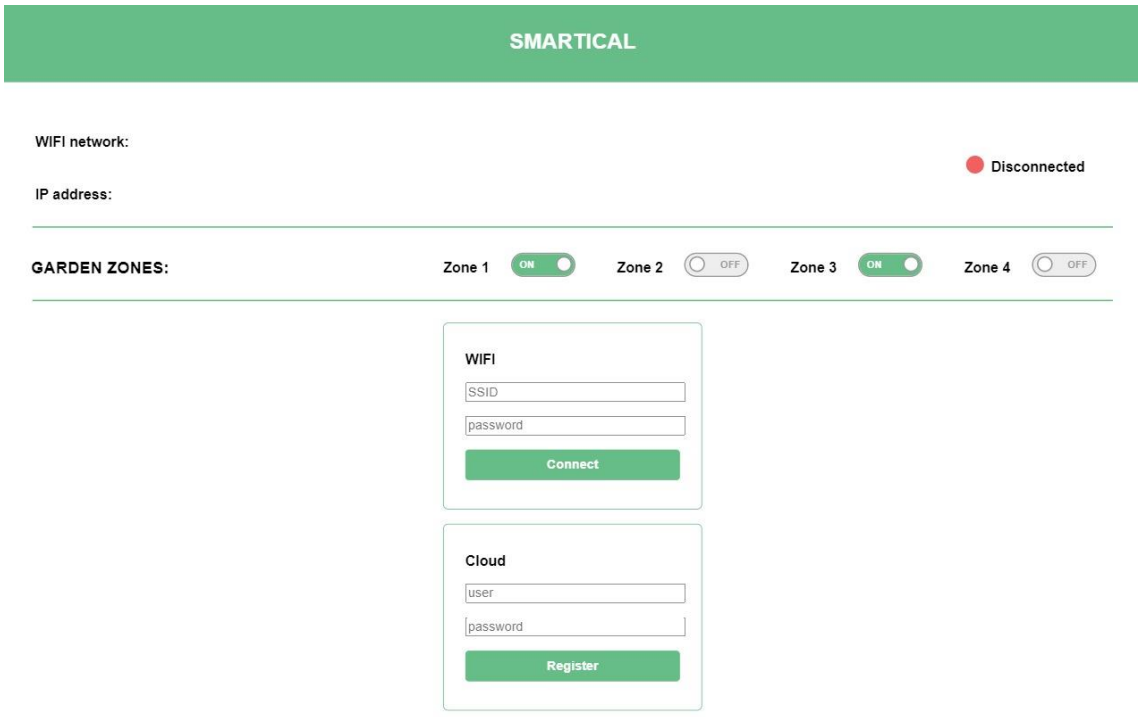
Adicionalmente el usuario podrá cambiar de contraseña y cerrar sesión en configuración, pulsando en el icono de menú de la ventana principal, situado arriba a la derecha, de manera que aparecerá una ventana como la que se muestra en la siguiente imagen donde el usuario deberá poner la contraseña actual y la nueva contraseña.



2. Conexión Wifi

Para realizar la configuración inicial, hay que acceder a la página de configuración del programador. Dicha página consta de cuatro secciones, en la parte superior se encuentra el estado del programador, donde se muestra el nombre de la red Wifi, la dirección IP y en la parte derecha se muestra si está conectado o no al wifi. A continuación, se encuentra la sección de las zonas del jardín, donde se puede actuar sobre las válvulas de agua de cada zona. Posteriormente se muestra una sección de registro de Wifi donde se debe introducir el SSID y la contraseña del wifi a la que se desea conectar el programador. Finalmente, en la parte inferior se encuentra el inicia de sesión del usuario.

La primera vez que se enciende el programador, se debe conectar el programador a una red wifi, y para ello se debe acceder a la página de configuración introduciendo como URL la dirección IP **192.168.4.1**. Una vez aparece la página de configuración, hay un apartado en el que se debe introducir la red wifi y la contraseña, una vez introducida, se reinicia el programador y aparece, el nombre de la red wifi y la dirección IP.

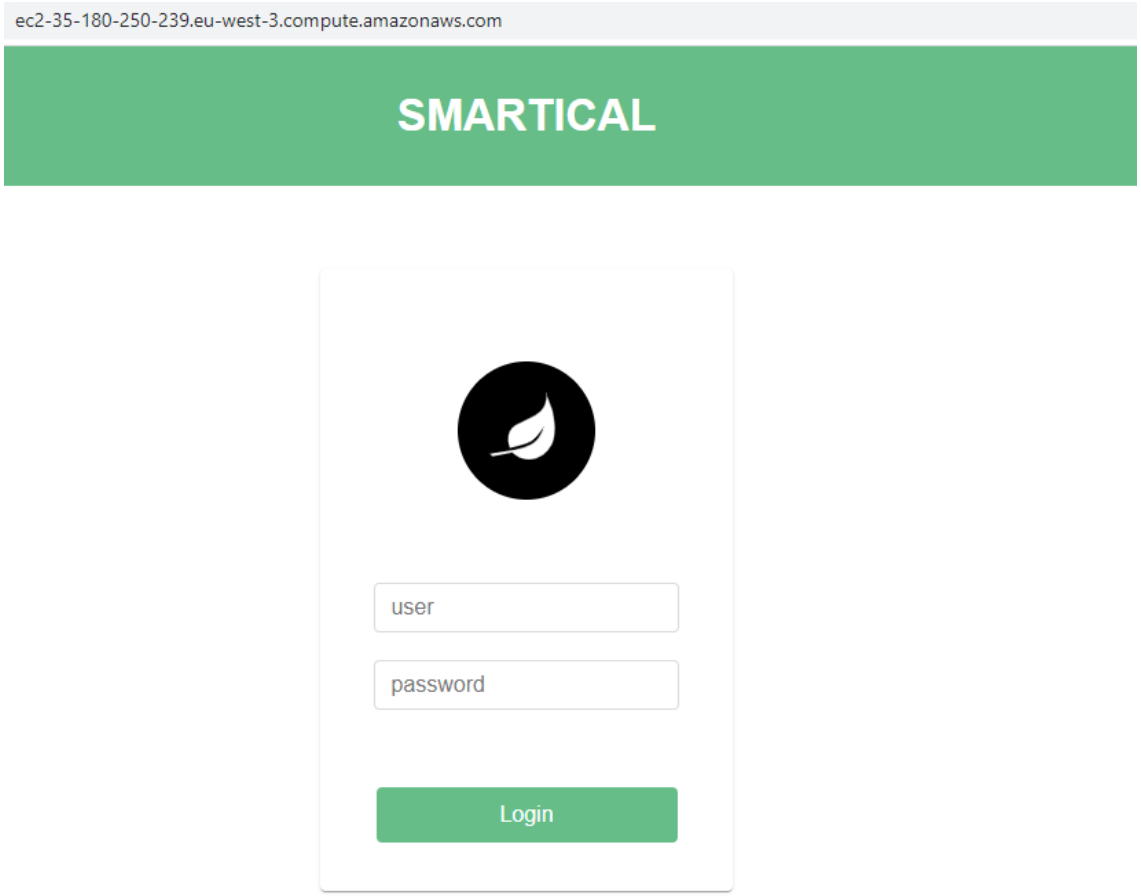


3. Conexión a la página web

Para poder visualizar la información de su jardín vertical debe introducir la URL:

http://ec2-35-180-250-239.eu-west-3.compute.amazonaws.com

Y le aparecerá una página de inicio de sesión donde deberá introducir su usuario y su contraseña y tras ello le aparecerá toda la información de su jardín vertical.



4. Conexión con agua

Para realizar la conexión de agua potable con el sistema de riego inteligente se debe conectar una toma de agua o manguera de la red de agua de su propiedad a la entrada de agua del sistema de riego.

5. Conexión con luz

Para que el sistema de riego inteligente funcione correctamente se le debe de suministrar energía eléctrica, 220V CA, para ello conecte el enchufe que se encuentra conectado al programador a una toma de electricidad de su propiedad.

Características técnicas de dispositivos

En este anexo se exponen las características técnicas, las ventajas y desventajas y el motivo de elección de cada dispositivo que se ha seleccionado para realizar este proyecto.

1. Sensor de humedad del suelo

I. Características técnicas

- Tensión de alimentación: 3.1 - 5.0V (máximo absoluto 5.5V)
- Corriente de alimentación máxima: ~ 15mA
- Salida: 0 - 3.0V (aproximadamente)
- Temperatura de funcionamiento: 2ºC a 45ºC
- Dimensiones: 170 mm * 25 mm * 15mm
- Longitud del cable: 1.5 m

II. Ventajas y desventajas

Ventajas

- Proporciona lecturas precisas.
- Fácil de utilizar y conectar con Arduino.
- Impermeable y resistente a la intemperie.
- Larga durabilidad.

Desventajas

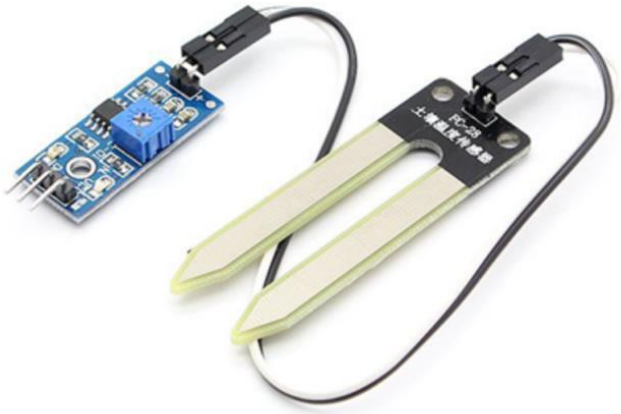
- Precio elevado, aproximadamente 20 euros.
- No capta temperaturas menores de 0ºC.

III. Motivo de elección

Durante el proceso de selección del sensor se han considerado las siguientes características; su tamaño, su impermeabilidad, su durabilidad, su robustez y que sus lecturas sean precisas. La alternativa a la solución propuesta es la siguiente;

Higrómetro de suelo FC-28

Es un sensor económico que mide la humedad de la tierra calculando la conductividad de la tierra. Este sensor consiste en dos placas separadas a una distancia específica y cubiertas por un material conductor de manera que en caso de existir humedad, entre dichas placas, se detectará por un circuito de control. Dicho circuito de control se encarga de transformar la conductividad en un valor analógico o en un valor digital si supera cierto umbral de humedad, el cual se puede fijar mediante un potenciómetro.
Por lo que debido a que el agua es un gran conductor, cuanto más húmeda esté la tierra, más conductividad habrá. El principal motivo por el que no se ha elegido este sensor es porque muestra problemas de corrosión.



2. Sensor de temperatura

I. Características técnicas

- Precisión: ± 0.3°C
- Temperatura de funcionamiento: - 50ºC a 110ºC
- Tolerancia: 1%
- Tiempo de respuesta aproximado: de 10 a 75 segundos en el aire
- Dimensiones: 6mm * 5mm * 60 cm
- Resistencia: 10 kOhms

II. Ventajas y desventajas

Ventajas

- Cable de 128 cm de largo
- Preciso
- No requiere ajuste entre el sensor y el circuito de control

Desventajas

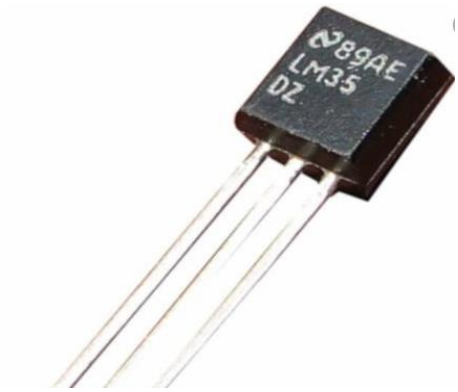
- Podría tener un tiempo de respuesta menor, pero para este proyecto no es necesario.

III. Motivo de elección

Durante el proceso de selección del sensor se han considerado las siguientes características; su tamaño, su impermeabilidad, su durabilidad, su robustez y que sus lecturas sean precisas. La alternativa a la solución propuesta es la siguiente;

LM35 DZ

LM35 DZ es un sensor de temperatura con salida analógica el cual proporciona una tensión de salida de 10mV por cada grado Celsius detectado, es proporcional a la temperatura del dispositivo, por ejemplo, si la tensión de salida es 250, entonces la temperatura es de 25 grados centígrados.



Realmente como sensor de temperatura se podría haber utilizado otros sensores como es el caso del sensor LM35, pero finalmente se ha optado por el sensor 103AT-11 por su cable de gran longitud y por su recubrimiento impermeable.

3. Válvula de agua solenoide

I. Características técnicas

- Potencia: Solenoide
- Presión de trabajo de 0,2 a 10 Atm
- Temperatura máxima fluido de 25ºC
- Tensión de alimentación: 24VAC
- Número de vías: 2

- Potencia: 3,7W
- Intensidad arranque: 313 mA
- Intensidad funcionamiento: 224 mA

II. Ventajas y desventajas

Ventajas

- Impermeable de acuerdo con norma C.E.I. I.P. 55 standard.
- Pequeño tamaño.
- Económica en comparación con otro tipo de válvulas.

Desventajas

- Imprescindible que se le aplique el voltaje adecuado dado que si se aplica menor voltaje del requerido no funcionará mientras que si se aplica mayor voltaje del requerido generará mucho calor y desgastará el solenoide.

III. Motivo de elección

Durante el proceso de selección de la válvula de agua se han considerado las siguientes especificaciones; su tamaño, su precio, su impermeabilidad, su durabilidad, su robustez y que sea solenoide. La alternativa a la solución propuesta es la siguiente;

Válvula de globo

Una válvula de globo es un tipo de válvula con la que se puede regular la cantidad de flujo que pasa por ella y en ella se realiza un cierre hermético. Los motivos por los que no se ha elegido este tipo de válvula es porque no se necesita regular el flujo de agua, la válvula de globo se utiliza para caudales mayores que el caudal del riego en jardines verticales y son menos económicas.

4. Wemos D1 mini pro

I. Características técnicas

- Microcontrolador: ESP-8266EX
- Tensión de Operación: 3.3V
- Tensión de alimentación: 5V
- Pines E/S Digitales: 11 (incluyendo RX y TX)

- Entradas Analógicas: 1
- Memoria Flash: 16 MB
- Frecuencia de reloj: 80MHz/160MHz
- Dimensiones: 34.2mm x 25.6mm

II. Ventajas y desventajas

Ventajas

- Conector de antena externa.
- Antena de cerámica integrada.
- Compatible con Arduino, programación con el IDE de Arduino, MicroPython, NodeMCU

Desventajas

- No dispone de más de 1 entrada analógica.
- Dispone de pocos pines E/S frente otras placas, pero son suficientes para este proyecto.

III. Motivo de elección

Durante el proceso de selección del módulo Wifi se han considerado las siguientes características; su tamaño, que se pueda soldar o usar en una PCB, que disponga de una antena externa para un mayor alcance y que disponga de dos pines para la comunicación con la placa Arduino, uno de transmisión y otro de recepción. La alternativa a la solución propuesta es la siguiente;

Nodemcu v3

NodeMCU es una pequeña placa Wifi de desarrollo basada en el ESP8266, que al igual que Wemos D1 mini pro, se puede programar mediante el IDE de Arduino. Dicha placa dispone de 4MB de memoria Flash, dispone de 10 pines digitales y se puede utilizar en una PCB. El principal motivo por el que no se ha elegido esta placa es porque no dispone de conector para incorporar una antena externa.

5. Arduino nano

I. Características técnicas

- Microcontrolador: ATmega328
- Tensión de Operación (nivel lógico): 5 V
- Tensión de alimentación (recomendado): 7-12 V
- Tensión de alimentación (límites): 6-20 V
- Pines E/S Digitales: 14 (de los cuales 6 proveen de salida PWM)
- Entradas Analógicas: 8
- Corriente máxima por cada PIN de E/S: 40 mA
- Memoria Flash: 32 KB de los cuales 2KB son usados por el *bootloader*
- SRAM: 2 KB
- EEPROM: 1 KB
- Frecuencia de reloj: 16 MHz
- Dimensiones: 18.5mm x 43.2mm

II. Ventajas y desventajas

Ventajas

- Pequeño tamaño
- Posee entradas analógicas, necesarias para los sensores de humedad.

Desventajas

- No viene con una toma de corriente continua, lo que significa que no se puede suministrar una fuente de alimentación externa a través de una batería.

III. Motivo de elección

Durante el proceso de selección del microcontrolador se han considerado las siguientes características; su tamaño, que se pueda soldar o usar en una PCB y que disponga de entradas analógicas. La alternativa a la solución propuesta es la siguiente;

Raspberry Pi zero

Es un ordenador de tamaño pequeño que dispone de un procesador, 40 pines, RAM y almacenamiento a través de una tarjeta SD. De los 40 pines de los que dispone la placa, 28 son GPIO (General Purpose Input/Output) y 12 para la alimentación. Cabe destacar que no dispone de pines analógicos, por lo que para conectar los sensores analógicos se debería utilizar un convertidor externo a la Raspberry Pi. El principal motivo por el que no se ha elegido este ordenador es porque no dispone de entradas analógicas.