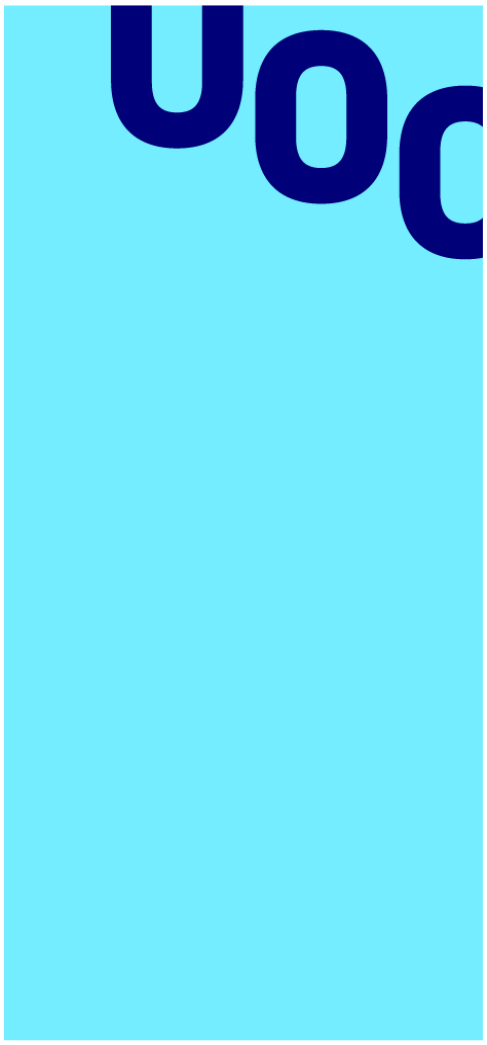
Aplicación móvil y diseño para la gestión de un sistema de riego residencial



Universitat Oberta de Catalunya

María Campillo Sánchez

Máster en Ingeniería de Telecomunicaciones
Smart Cities

Tutor/a de TF
Rubén Molina Casasnovas
Profesor/a responsable de la asignatura
Carlos Monzo Sánchez

10 Junio 2024

Ficha del Trabajo Final

Título del trabajo:	Aplicación móvil y diseño para la gestión de un sistema de riego residencial
Nombre del autor/a:	María Campillo Sánchez
Nombre del Tutor/a de TF:	Rubén Molina Casasnovas
Nombre del/de la PRA:	Carlos Monzo Sánchez
Fecha de entrega:	06/2024
Titulación o programa:	Grado en Ingeniería de Tecnologías y servicios de Telecomunicación
Área del Trabajo Final:	Smart Cities
Idioma del trabajo:	Castellano
Palabras clave	Riego, aplicación, sensores

Resumen del Trabajo

El objetivo de este proyecto es el desarrollo y diseño de una aplicación móvil, destinada a centralizar los parámetros para el óptimo cuidado de las plantas en entornos residenciales.

De esta manera en el proyecto se creará un prototipo que permitirá tomar los valores provenientes de los sensores integrados en las plantas, su análisis y en consecuencia de dichos valores obtenidos actuar o notificar al usuario dependiendo del resultado obtenido.

La aplicación estará diseñada para contener información fundamental sobre las plantas, tales como requerimientos de humedad y luminosidad, y posibilitará la activación del riego automático a través de una conexión WiFi, en caso de ser necesario.

Por todo esto el proyecto implementará un sistema de sensores que registran los valores de las condiciones ambientales, que compartirá la información con un sistema de control donde se analizaran y guardaran los datos obtenidos, conectado mediante una red de comunicaciones inalámbrica que interactuara

con la aplicación móvil desde la cual el usuario tendrá acceso a todas las funcionalidades del sistema.

Con un enfoque integral en la eficiencia y comodidad del usuario, este proyecto se propone ofrecer una solución tecnológica avanzada para el cuidado de las plantas en el hogar, optimizando el proceso de riego y garantizando un ambiente propicio para su desarrollo saludable.

Abstract

The objective of this project is the development and design of a mobile application, aimed at centralising the parameters for the optimal care of plants in residential environments.

In this way, the project will create a prototype that will allow taking the values coming from the sensors integrated in the plants, analysing them, and consequently acting or notifying the user depending on the obtained result.

The application will be designed to contain fundamental information about the plants, such as humidity and luminosity requirements, and if necessary, it will enable the activation of automatic irrigation through a WiFi connection.

For all this, the project will implement a system of sensors that record the values of the environmental conditions, which will share the information with a control system where the data obtained will be analysed and stored, connected via a wireless communications network that will interact with the mobile application from which the user will have access to all the functionalities of the system.

With an integral focus on efficiency and user comfort, this project aims to offer an advanced technological solution for the care of plants in the home, optimising the irrigation process and guaranteeing an environment favourable to their healthy development.

Índice

Contenido

Lista de Figuras

1. Introducción

En esta parte de la memoria se expondrán los motivos por los cuales se ha realizado este proyecto junto con la planificación de este. Se establecerán los objetivos principales, analizando los aspectos de sostenibilidad, diversidad y ética social junto con un índice de la memoria.

1.1. Contexto y justificación del Trabajo

A pesar de que existen algunos tipos de plantas que son muy fáciles de cuidar, otras requieren condiciones específicas para que su desarrollo sea óptimo. En ambos casos, al automatizar las mediciones de los parámetros ambientales y poder almacenarlas en una base de datos accesible desde una aplicación móvil facilita considerablemente las tareas relacionadas con el cuidado de las plantas.

El objetivo principal es que, dependiendo del tipo de planta registrada en la aplicación, ésta sea capaz de ajustar sus parámetros permitidos y alertar cuando sea necesario realizar modificaciones en la rutina de riego. En situaciones extremas, como sequías o exceso de humedad, la aplicación podría activar el riego automático sin necesidad de intervención por parte del usuario. Se planea analizar diversas magnitudes, como temperatura, humedad ambiental, humedad del suelo, luminosidad y nivel de agua.

Estos valores serán obtenidos mediante sensores instalados cerca o dentro de las plantas, los cuales se conectarán a un microcontrolador para analizar los datos y ejecutar las acciones correspondientes. La gestión del sistema se realizará a través de una conexión inalámbrica.

Este proyecto busca permitir que mediante este diseño se pueda regar las plantas de manera automatizada mediante la aplicación, ya que si por ejemplo el usuario se va de vacaciones durante un periodo de tiempo podrá tener la opción de regar sus plantas de acuerdo con los parámetros que cada planta específica tenga, ya que con otros productos ofrecidos en el mercado no se da la opción de observar cada magnitud importante para el correcto cuidado de la planta o son de un alto coste, por lo que al tratarse de un producto de uso individual se ha intentado usar componentes asequibles que no incrementan significativamente el coste del producto final.

1.2. Objetivos del Trabajo

El objetivo principal es el diseño e implementación de un sistema de riego automatizado según las condiciones ambientales recogidas mediante los sensores, mediante el uso de una aplicación móvil desde la que se gestionará el riego óptimo para las plantas.

Para cumplir el objetivo se seguirán los siguientes pasos:

- Diseño e implementación de una aplicación móvil que permita centralizar y gestionar los parámetros ambientales de manera automatizada para el riego óptimo para las plantas.

- Se desarrollará un prototipo que pueda recopilar datos provenientes de sensores instalados en las plantas, analizarlos y tomar acciones automáticas o generar alertas en función de los valores obtenidos.

- Establecer una comunicación inalámbrica entre los sensores, el microcontrolador y la aplicación móvil.

- Proporcionar al usuario acceso remoto a través de la aplicación móvil para el control de los parámetros del sistema de riego.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Este proyecto tiene un impacto directo en la sostenibilidad, dado que permite gestionar de manera eficiente el consumo de agua invertido en el riego de las plantas en el hogar, además ayudando al correcto cuidado de las plantas se puede aumentar el tiempo de vida de estas.

- Impacto en sostenibilidad:

Cuando usamos sensores para medir los valores ambientales, se reduce de manera significativa el desperdicio de agua, ya que se sabe exactamente cuando es necesario regarlas evitando un uso excesivo de agua o que debido a variaciones temporales sea necesario regarlas antes de lo planeado consiguiendo que la planta no se seque en caso de extremo calor. Esto contribuye a la conservación de recursos naturales y promueve un riego más sostenible y responsable.

- Impacto ético-social:

Un gran aporte de este proyecto en el ámbito ético-social es que mediante el acceso ofrecido con la aplicación móvil se les permite a personas con movilidad reducida u horarios ocupados cuidar las plantas de su hogar de manera eficiente, sin tener que hacer un esfuerzo físico o tener que estar en casa en ese momento. Además, con las alertas en tiempo real se facilita el realizar esta tarea.

- Impacto en diversidad:

Con el diseño de la aplicación móvil se busca mantener una interfaz fácil de usar y que sea muy intuitiva para que independientemente de lo familiarizado que esté el usuario con el uso de sistemas automatizados no le sea difícil hacer uso de este dispositivo. Además, en el diseño del prototipo se ha intentado mantener un coste bajo para que el rango de personas que pueden permitirse un producto con estas características sea lo más amplio posible, intentando poder ofrecer una igualdad en los servicios para toda la población.

1.4. Enfoque y método seguido

Actualmente existen sistemas que ofrecen un riego automatizado, pero suelen estar enfocados a mayor escala y especializados en un único tipo de planta o bajo unas condiciones ambientales pre controladas, por lo que se suelen basar en un riego temporizador, por lo que el enfoque de este trabajo será crear un sistema que en base a las condiciones ambientales en tiempo real se permita decidir la decisión a tomar de manera óptima adaptándose a un riego a pequeña escala como es el del hogar. Además, los otros sistemas ofrecidos en el mercado tienen un coste mayor, mientras que en este caso se ha intentado usar componentes de bajo coste para que sea más accesible.

La metodología de este trabajo será realizar un análisis de los mejores componentes para el diseño del prototipo, elaborando el plan para el diseño del sistema completo que incluye el mecanismo de riego, los sensores o la aplicación móvil, integrándolo todo mediante un microcontrolador para la gestión de los datos obtenidos gracias a los sensores. Se realizarán pruebas para optimizar el prototipo y asegurar que los datos ambientales son leídos y almacenados correctamente para después comprobar que desde la aplicación móvil se pueden leer esos datos y solicitar la activación del sistema de riego.

1.5. Planificación del trabajo

La planificación del proyecto se ha basado en las PECs, creando hitos en un diagrama de Gantt para cada una de las entregas de las PECs. Se dividirá en cinco fases:

- Fase 1: Definición

Propuesta de título, resumen, palabras clave, motivación, impactos de sostenibilidad, ético-social y diversidad, enfoque y método seguido, planificación del proyecto y un índice preliminar de la memoria.

- Fase 2: Estado del arte

Repaso de los conocimientos existentes sobre el riego automatizado y su control mediante aplicaciones móviles junto con la justificación de los componentes elegidos para la realización del proyecto.

- Fase 3: Diseño e implementación

Desarrollo e implementación del prototipo y diseño de la aplicación móvil junto con la realización de las pruebas necesarias para comprobar la funcionalidad del proyecto.

- Fase 4: Memoria

Desarrollo completo de las características del proyecto junto con un análisis de los resultados obtenidos en la fase 3, finalizando con las conclusiones.

- Fase 5: Presentación y defensa

Elaboración de la presentación, explicando el proyecto junto con la resolución de las cuestiones por parte del tribunal.

	Task Name	Duration	Start	Finish
1	**Organización proyecto**	**4,11 mons**	**Wed 28/02/24**	**Thu 20/06/24**
2	**Fase 1: Definición**	**1,83 wks**	**Wed 28/02/24**	**Sun 10/03/24**
3	Titulo	1 day	Wed 28/02/24	Thu 29/02/24
4	Resumen	3 days	Thu 29/02/24	Tue 05/03/24
5	Palabras clave	1 day	Tue 05/03/24	Wed 06/03/24
6	Motivacion	1 day	Wed 06/03/24	Thu 07/03/24
7	Impactos	1,13 days	Thu 07/03/24	Thu 07/03/24
8	Enfoque	1 day	Thu 07/03/24	Fri 08/03/24
9	Planificación	1,13 days	Fri 08/03/24	Fri 08/03/24
10	Índice	2,13 days	Fri 08/03/24	Sat 09/03/24
11	Entrega PEC1	0 days	Sun 10/03/24	Sun 10/03/24
12	**Fase 2: Estado del arte**	**2,23 wks**	**Mon 11/03/24**	**Sun 24/03/24**
13	Sistemas similares	4,13 days	Mon 11/03/24	Thu 14/03/24
14	Magnitudes	3 days	Thu 14/03/24	Tue 19/03/24
15	Estudio componentes	3 days	Tue 19/03/24	Fri 22/03/24
16	Comunicación	2,13 days	Fri 22/03/24	Sat 23/03/24
17	Entrega PEC2	0 days	Sun 24/03/24	Sun 24/03/24
18	**Fase 3: Diseño e implementación**	**2,01 mons**	**Mon 25/03/24**	**Fri 17/05/24**
19	Diseño Hardware	10 days	Mon 25/03/24	Mon 08/04/24
20	Diseño Software	15 days	Mon 08/04/24	Mon 29/04/24
21	Comunicación	10 days	Mon 29/04/24	Mon 13/05/24
22	Pruebas	5,13 days	Mon 13/05/24	Fri 17/05/24
23	Entrega PEC3	0 days	Fri 17/05/24	Fri 17/05/24
24	**Fase 4: Memoria**	**3,43 wks**	**Sat 18/05/24**	**Mon 10/06/24**
25	Reunir documentación	2 days	Sat 18/05/24	Tue 21/05/24
26	Formato	2 days	Tue 21/05/24	Thu 23/05/24
27	Resultados	5 days	Thu 23/05/24	Thu 30/05/24
28	Conclusiones	7 days	Thu 30/05/24	Mon 10/06/24
29	Entrega PEC4	0 days	Mon 10/06/24	Mon 10/06/24
30	**Fase 5: Presentación**	**1,63 wks**	**Tue 11/06/24**	**Thu 20/06/24**
31	PowerPoint	5 days	Tue 11/06/24	Tue 18/06/24
32	Video	2 days	Tue 18/06/24	Thu 20/06/24
33	Entrega PEC5	0 days	Thu 20/06/24	Thu 20/06/24

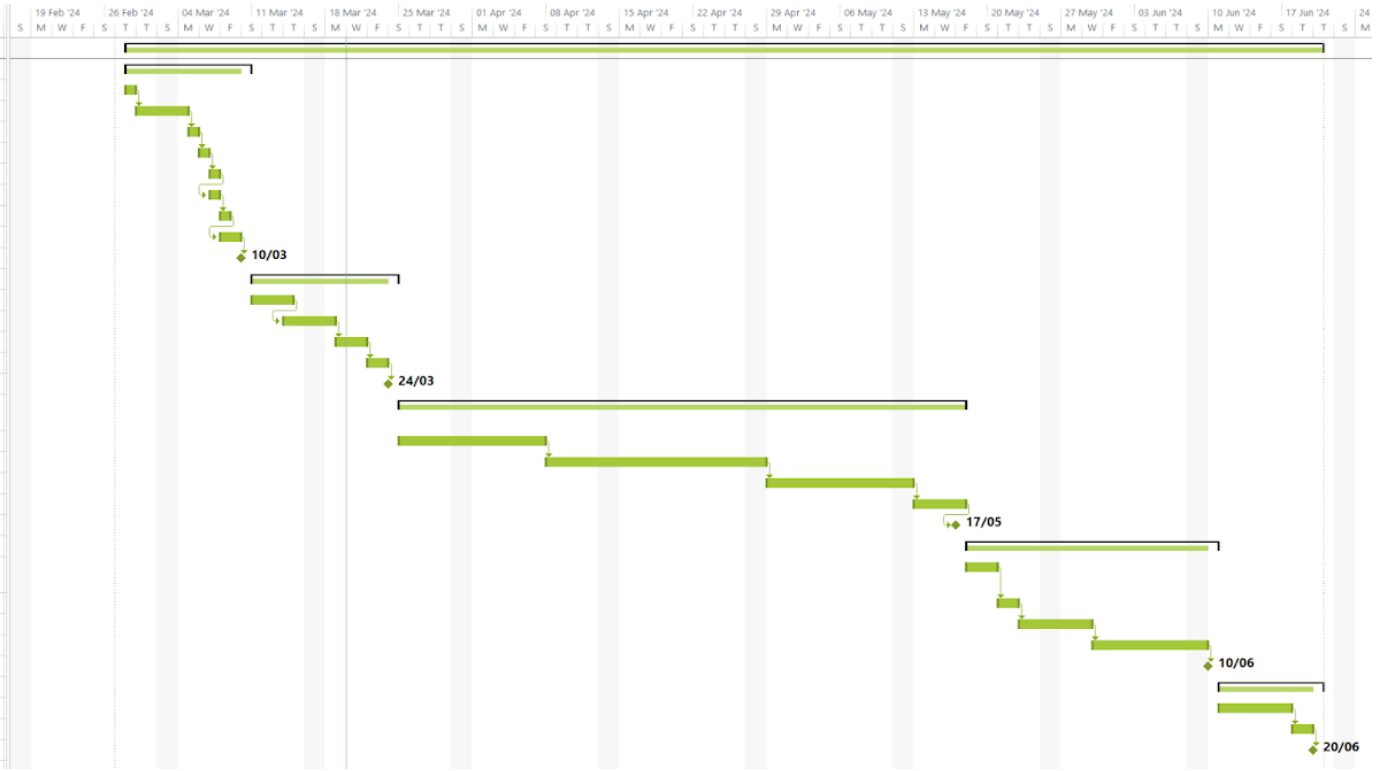


Figure 1. Diagrama de Gantt

1.6. Breve sumario de productos obtenidos

Como resultado de este proyecto, se pretende obtener una aplicación móvil para el cuidado de las plantas en el hogar, fácil de usar e intuitiva desde la que se podrá gestionar el riego automatizado a la vez que se monitorizan los valores ambientales a tiempo real, recibiendo alertas en caso de ser necesario. Esta aplicación estará conectada a un prototipo de sistema de riego automatizado que serán los que recaben los datos obtenidos mediante los sensores y almacenarlos en una base de datos para que se permita la comunicación entre ambas partes y de esta forma garantizar el correcto cuidado de las plantas del hogar.

1.7. Breve descripción de otros capítulos de la memoria

Tras el primer capítulo de introducción, en el capítulo 2 se establece el estado del arte, donde se explica el objetivo del proyecto, analizando proyectos previos con un tema relacionado al proyecto a desarrollar. También se expondrán las especificaciones del proyecto, analizando los componentes que mejor se adapten al diseño Hardware del proyecto, se desarrolla el diseño Software, diseñando la aplicación móvil y la base de datos.

El capítulo 3 es donde se describe el proceso para crear el prototipo referenciando al capítulo anterior, explicando el interconexionado entre las distintas partes del proyecto, mostrando la visión conjunta de todo el proyecto y se incluirá el presupuesto para los componentes adquiridos.

En el capítulo 4 estarán incluidas las conclusiones a las que se llegan tras analizar los resultados obtenidos en el capítulo anterior y posibles variaciones o ampliaciones a las que se puede llegar si se continuara el proyecto en un futuro.

El capítulo 5 incluye el glosario con los términos de la memoria

En el capítulo 6 se indican las fuentes bibliográficas usadas en el proyecto.

En el capítulo 7 estarán los anexos donde se mostrará la documentación adicional necesaria recopilada a lo largo de la realización del proyecto.

2. Materiales y métodos

A continuación, analizaremos las diferentes tecnologías existentes dentro del panorama actual, centrándonos en tecnologías que hagan uso de Arduino para controlar los sensores y funcionen mediante una aplicación móvil para simplificar y mejorar la eficiencia del riego automatizado actualmente.

2.1. Estado del arte

Gracias a los avances tecnológicos logrados en los últimos años cada vez es más común y sencillo automatizar ciertas tareas, como puede ser el cuidar las plantas del hogar de manera automatizada. Gracias a la automatización de esta tarea el usuario es capaz de realizar un cuidado más preciso de las plantas, ya que se tiene información en tiempo real de magnitudes clave para el cuidado de estas como puede ser la temperatura o la humedad [1]. Además reduce el tiempo que el usuario debe invertir en regar las plantas, o incluso en caso de no encontrarse en el hogar, poder realizar el cuidado de las plantas a distancia, por ello se ha realizado una investigación sobre los proyectos, las tecnologías y los productos existentes relacionados con el proyecto llevado a cabo, donde se analizaran los sistemas de riego automatizado, aplicaciones móviles preexistentes dedicadas al cuidado de las plantas y los sensores más usados para este propósito.

2.1.1. Sistemas de riego

Con los avances tecnológicos que se han producido durante los últimos años, cada vez existen un mayor rango de opciones en cuanto a sistemas de riego automatizados, dependiendo de las necesidades de cada usuario. Las más comunes que podemos encontrar en el mercado son:

- Temporizadores de riego programables: Son los sistemas más básicos que podemos encontrar en el mercado, permiten al usuario activar el riego en periodos específicos junto con la duración de la apertura de las válvulas de riego y su frecuencia.

- Sistemas de riego por goteo: Estos sistemas permiten distribuir el riego de manera uniforme entre todas las zonas de la plantación a través de tubos con orificios, este sistema se puede automatizar si se le incorpora el temporizador de riego programable [2].

- Sistema de riego por aspersión: Este sistema se ajusta menos a las necesidades de nuestro proyecto debido a que al funcionar mediante aspersores se enfoca a superficies de mayor magnitud como jardines.

- Sistema de riego basados en sensores: En este proyecto usaremos como una de las bases este sistema de riego, que utiliza sensores de humedad del suelo, humedad ambiental, temperatura, luminiscencia entre otros [3] y mediante los datos recopilados se ajustarán automáticamente los patrones de riego.

- Sistemas de riego controlados mediante aplicaciones móviles: Este sistema sería la otra base de este proyecto, estos sistemas permiten desde una aplicación móvil controlar la apertura de las válvulas de riego, permitiendo al usuario regar las plantas incluso si no se encuentra en su hogar [4].

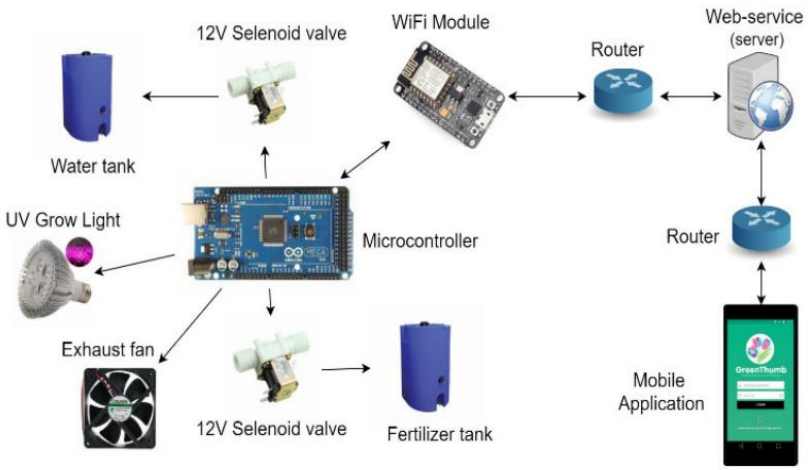


Figure 2. Componentes proyecto descrito en [4]

2.1.2. Magnitudes y sensores

Este tipo de sistemas automatizados dedicados al riego de plantas mediante una aplicación móvil se basan en el uso de un conjunto de sensores que ofrecen las magnitudes necesarias para el cuidado óptimo de las plantas permitiendo cuidarlas de manera eficiente y de manera más cómoda para el usuario, se tratan de sensores de humedad, de temperatura o del nivel de agua, entre otros, estos sensores tal y como se explica en [5] estos sensores se conectaran a un Arduino mediante el cual se permitirá observar en tiempo real los datos proporcionados por los sensores. Además de observar las magnitudes de los sensores, mediante Arduino se puede lograr obtener la funcionalidad de activar automáticamente la bomba de agua para regar las plantas cuando se cumplan los requisitos previamente indicados, como podría ser cuando la humedad del suelo desciende por debajo de un umbral determinado.

Como se ha indicado, el conocimiento preciso de las magnitudes ambientales es necesario para el correcto cuidado de las plantas y su apropiado crecimiento, específicamente es necesario conocer magnitudes como la temperatura, humedad, humedad del suelo y luminiscencia, para así poder tener un entorno óptimo para las plantas en el hogar.

- Temperatura: interviene en el crecimiento de la planta y mantener a una planta en rangos de temperatura opuestos a los recomendados puede provocar que la planta se seque, se congele o que adquiera enfermedades debido a la temperatura errónea. Por ello se usan sensores de temperatura de alta precisión mediante los cuales se permite saber en tiempo real las condiciones térmicas en las que se encuentra la planta.

- Humedad: esta magnitud afecta a la absorción del agua y la transpiración de las plantas, ya que dependiendo del tipo de planta que se cultive se necesitará una mayor humedad ambiental o se le deberá proporcionar un lugar en el hogar más seco para lograr un nivel de humedad menor.

- Humedad del suelo: la humedad del suelo es un factor crítico, ya que es el indicador de si las raíces de las plantas están recibiendo suficiente agua, por ello es clave monitorear continuamente esta magnitud teniendo en cuenta que dependiendo del tipo de planta necesitarán una mayor humedad o una menor, también es importante que el sensor de humedad sea colocado correctamente porque si se coloca únicamente de manera superficial sería posible que la cantidad de agua durante el riego no sea la suficiente pero el sensor detecte como que ya ha alcanzado el nivel óptimo de acuerdo con la característica específica de la planta en cuestión.

- Luminiscencia: las plantas realizan la fotosíntesis, por lo que es necesario que la planta reciba luz durante el día, pero dependiendo del tipo de planta la intensidad necesaria variará, incluso dependiendo del día la planta podría necesitar luz o no, por ello este sensor se encarga de indicar el nivel de luz disponible para de esta forma poder optimizar la ubicación de las plantas.

2.1.3. Proyectos e investigaciones previas

Ya existen estudios previos como el de [6] donde se analiza la forma de controlar el proceso de riego mediante sensores inalámbricos mediante los cuales los agricultores pueden gestionar a distancia el terreno [7], además haciendo uso de aplicaciones móviles los agricultores son capaces de recibir notificaciones sobre los valores a tiempo real obtenidos en los sensores, como puede ser la humedad o la temperatura a la que se encuentra el sustrato [8].

Haciendo uso de esos datos, si se almacenan en la nube, se pueden analizar esos datos para que los agricultores puedan extrapolar esos datos y tomar decisiones acordes a ellos o prever qué puede ocurrir en el futuro teniendo en cuenta los datos recopilados prediciendo su rendimiento. Mediante estos avances tecnológicos los agricultores son capaces de

optimizar el uso del agua, lo cual es clave en zonas con limitadas reservas de agua, reduciendo el desperdicio de los recursos disponibles [9].

Estos proyectos previos, aunque destinados a sistemas más extensos, se pueden extrapolar para tener un sistema automatizado para el riego de las plantas en el hogar, ya que, ya que, si se compara con el sistema de riego automatizado destinado a la agricultura, encontramos las mismas tecnologías y la misma red de sensores solo que en menor medida para así poder obtener un sistema de riego inteligente y eficiente para el hogar.

Artículos como [10] o [11] muestran la importancia de hacer un uso del riego automático mediante el uso de aplicaciones móviles, que se puede extrapolar al hogar, ya que ofrece un análisis de datos en tiempo real para las necesidades específicas de cada tipo de planta que el usuario tenga en el hogar, además al monitorizar los sistemas de riego, se asegura que las plantas siempre reciban la cantidad adecuada de agua en el momento necesario también en plantas en interiores.

En el artículo [12] se centraron en el análisis de datos basado en la nube teniendo un almacenamiento de los datos a largo plazo, para tener un modelo predictivo mediante un análisis de tendencias de forma que el usuario pueda tomar las decisiones con toda la información posible.

2.1.4. Soluciones comerciales

Durante el desarrollo del proyecto será importante tener en cuenta las aplicaciones e investigaciones previas relacionadas con el sistema automatizado de riego para el hogar ya que nos permitirá obtener información valiosa para lograr un mayor entendimiento del campo y así ofrecer al usuario un proyecto innovador.

Existen programas que permiten almacenar y analizar los datos obtenidos en la nube para de esta manera tener a largo plazo una recopilación de los datos, lo que permitirá poder adelantarse a las necesidades de las plantas y tomar decisiones informadas teniendo toda la información posible sobre las características de la planta y su entorno.

En el mercado existen diversas aplicaciones móviles que ofrecen una variedad de herramientas y recursos para el cuidado de las plantas en el hogar, algunas ofrecen información sobre la planta, ayudando a identificarla o permite el control de esta, algunos ejemplos serían:

- Blossom Pflanzer: esta aplicación ofrece información detallada de la planta, proporcionando también detalles sobre la frecuencia necesaria de riego o los requisitos de luz de cada tipo de planta, además ofrece una alarma para recordarte que debes regar las plantas

- Plant parent: mediante esta aplicación se pueden tomar fotografías de las plantas para que la aplicación reconozca la planta y permita emitir un diagnóstico sobre la planta indicando si tiene deficiencias, necesita más luz, menos agua, etc. Esta aplicación móvil además de indicar un diagnóstico de la planta te permite crear un plan de riego para que la planta esté lo más sana posible o indica al usuario el mejor lugar del hogar en el que colocar la planta

- Planta: esta aplicación móvil tras tomar una foto identifica el tipo de planta indicando el mejor momento para regar la planta, la luminosidad que necesita entre otros detalles.

En conclusión, las aplicaciones móviles principales que podemos encontrar en el mercado permiten diagnosticar enfermedades de las plantas mediante reconocimiento de imágenes, crear recordatorios para realizar el riego de las plantas u ofrecen consejos e información sobre las necesidades de la planta, como pueden ser la cantidad de luz que necesitan o cuando es mejor regarlas.

Tras analizar las opciones disponibles en el mercado actual encontramos la empresa Gardena, que cuenta con un Smart System que cubre los requisitos del proyecto, pero que en este caso está compuesto de múltiples productos individuales lo que hace que el coste del producto final sea elevado, ya que consta de un sensor que incorpora el análisis de humedad, temperatura y luz, un producto que controla el flujo de agua y por ultimo otro producto que es el encargado de realizar la conexión entre ambos y permitir que el usuario controle el riego desde su móvil.

Figure 3. Sistema de riego Gardena

Se han abordado los puntos clave que muestran la importancia del uso de sistemas de riego automatizados, mostrando que es una solución conveniente para garantizar la calidad del cuidado de las plantas y que mediante la automatización de los sistemas de riego en las

plantas para uso en el hogar se permite optimizar el uso del agua, facilitando al usuario el mantener el jardín o huerto urbano.
Este proyecto pretende investigar sobre este campo para lograr un enfoque en el cuidado de las plantas mediante la tecnología de automatización, usando técnicas de análisis de datos, investigando los sensores más apropiados para este diseño y controlar de manera óptima el riego de las plantas del hogar adaptando las investigaciones previas a un entorno y condiciones especifico dentro del hogar.

2.2. Especificaciones del sistema

El proyecto debe cumplir ciertas características y especificaciones para que funcione correctamente, es por esto por lo que en este apartado se comentarán cuáles son las especificaciones para lograr cumplir con el objetivo del proyecto.

Como se ha comentado en los apartados anteriores, se trata de un proyecto cuya finalidad es lograr un sistema de riego automatizado para el hogar mediante una aplicación móvil, que funcionará en función de las características específicas de la planta a regar y de las condiciones climáticas en tiempo real para de esta manera lograr un riego óptimo.
La primera parte del proyecto se centra en la recopilación de las magnitudes ambientales, esto se logra mediante sensores, que serán los encargados de obtener los valores en tiempo real de la temperatura, la humedad, la humedad del suelo y la luminosidad. Estos valores a continuación serán almacenados y analizados para determinar si es necesario realizar alguna acción o si sus parámetros siguen dentro de los límites óptimos de la planta.

Dichos parámetros óptimos deberán ser indicados por parte del usuario en la aplicación móvil para que se puedan comparar con los valores obtenidos por los sensores ambientales. Además de los sensores mencionados anteriormente también se contará con un sensor que medirá el nivel del agua para poder informar al usuario si es necesario añadir más agua al compartimento encargado de almacenarla hasta que sea necesario su uso tras la activación del riego. Los sensores están conectados a un Arduino mediante el cual se leerán todos sus valores y en el caso de que los parámetros estén fuera de los límites, como puede ser que la tierra no está lo suficientemente húmeda se avisará al usuario de que es necesario regar, o si la planta está recibiendo un exceso de luz o por el contrario no la suficiente cantidad el usuario también será informado de que necesita mover la planta a otra posición más favorable para el correcto desarrollo de esta. En este proyecto también se contempla la posibilidad de que la humedad del suelo descienda de forma drástica y que el usuario no pueda acceder a su riego por lo que existe una condición que si es extremadamente baja su humedad se activa automáticamente sin necesidad de recibir una orden por parte del usuario el riego de manera automática.
Al Arduino también se encontrará conectado un relé que será el que recibirá la orden de activar el riego, ya que este estará conectado a una bomba de agua junto con un motor que serán los que permitirán el riego de las plantas cuando sea necesario.

Desde Arduino se podrán visualizar estos datos recabados pero el objetivo es que el usuario pueda tener acceso a ellos y actuar en consecuencia desde su móvil haciendo uso de una aplicación móvil, por esto, se conectara el Arduino con una base de datos de Firebase que actuará de nexo entre el Arduino y la aplicación móvil, en la base de datos se almacenarán

los valores obtenidos por los sensores, junto con los valores óptimos para el cuidado de las plantas proporcionados por el usuario desde la aplicación y además será el encargado de guardar los datos de los usuarios registrados en la aplicación móvil para de esta forma poder restringir el acceso, permitiendo únicamente al usuario acceder a los parámetros recogidos en su hogar.

La aplicación móvil deberá contar con diferentes secciones que permitirá añadir nuevas plantas con los parámetros específicos de cada una, eliminarla en caso de ser necesario, otra sección estará destinada a la visualización de los valores de los sensores en tiempo real, además se podrá activar el riego automático mediante un botón desde la aplicación móvil. Uno de los objetivos del proyecto es que el usuario pueda recibir notificaciones en la aplicación relevantes sobre el cuidado de las plantas en su hogar, para el diseño de la aplicación móvil se ha usado Android Studio, mediante el cual se programará la aplicación conforme a todas las especificaciones mencionadas anteriormente en este apartado de tal forma que todas queden cubiertas.

En conclusión, este proyecto se centrará en el uso de sensores específicos para el análisis de las condiciones climáticas, conseguir una comunicación bidireccional entre Arduino, Firebase y Firebase con la aplicación móvil para que el usuario sea capaz de controlar de manera remota el riego desde la aplicación móvil además de poder visualizar los datos obtenidos por los sensores en tiempo real. También se deberán implementar medidas de seguridad para que los datos de los usuarios se encuentren protegidos.

2.3. Diseño Hardware

Para la realización del proyecto han sido necesarios múltiples componentes eléctricos, por lo que serán descritos a continuación indicando sus características y cómo se han interconectado para lograr el objetivo del proyecto.

2.3.1. Arquitectura

Para el diseño hardware del sistema serán necesarios tal y como se ha indicado en capítulos anteriores, diferentes tipos de componentes más específicamente serán necesarios distintos tipos de sensores, un microcontrolador y un actuador de riego. A continuación, se muestra un diagrama con la interconexión entre estos componentes para obtener una visión del diseño hardware desarrollado.

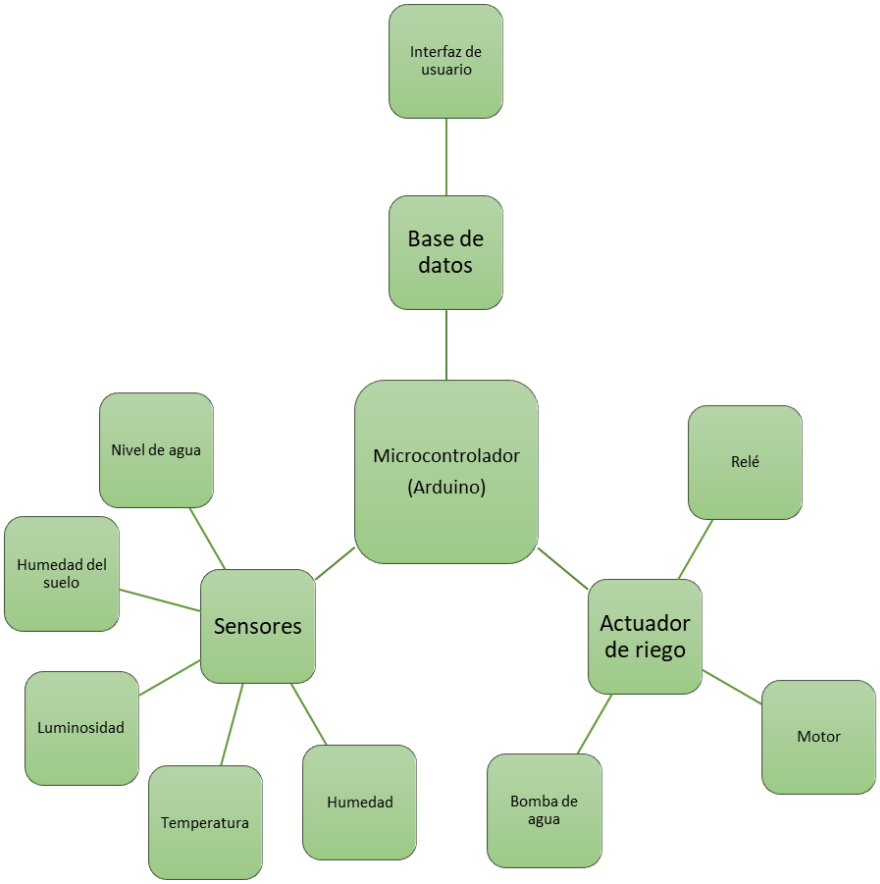


Figure 4. Diagrama organización arquitectura hardware

En la figura anterior se muestra las conexiones que existen entre los distintos grupos del proyecto:

- Microcontrolador: se encuentra localizado en el centro del diagrama ya que es el que conecta los distintos componentes hardware, se encarga de captar la información proporcionada por los sensores, activar si es necesario el actuador de riego y enviar la información recabada a la base de datos para que almacene, distribuya y analice dichos datos.

- Sensores: proporcionan los valores de las condiciones ambientales, se hará uso de sensores de temperatura, nivel de agua, humedad de suelo y luminosidad.

- Actuador de riego: está formado por un relé, un motor y una bomba de agua, será el encargado de realizar el riego activando el relé permitiendo que circule la corriente en el motor cuando se reciba la orden por parte del microcontrolador.

- Base de datos: Almacena la información recopilada por el microcontrolador y la que recibe por parte del usuario.

- Interfaz de usuario: el cliente se comunicará con el microcontrolador a través de órdenes que serán analizadas por la base de datos.

2.3.2. Especificaciones de los componentes

A continuación, se describirán las especificaciones de cada componente que se ha decidido usar en el desarrollo del producto, la elección puede haber sido debida a sus características, precio o compatibilidad con el resto de los componentes.

2.3.2.1. Arduino UNO R4 WiFi

El Arduino va a ser el encargado de interactuar con los distintos componentes que van a formar parte del proyecto, por eso tiene que ser capaz de recibir la información, transmitirla y gestionar la activación de los componentes cuando sea necesario. Los Arduino cuentan con interfaces de entrada (analógicas y digitales), un microcontrolador, que procesa las señales y controla mediante interfaces de salida.

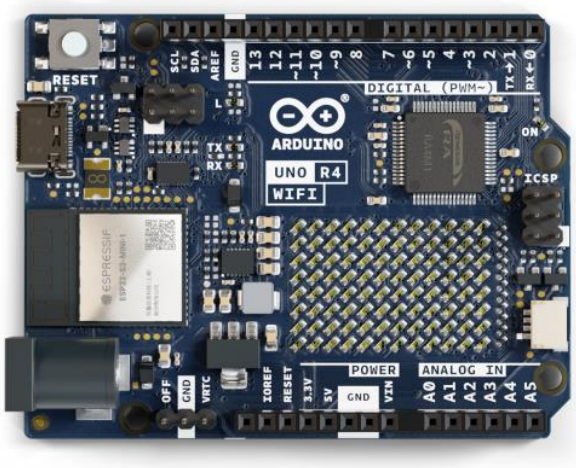


Figure 5. Arduino UNO R4 WiFi

Se ha elegido el Arduino UNO R4 WiFi [13], porque tiene múltiples puertos, tanto digitales como analógicos, además cuenta con un módulo ESP32-S3 WiFi, lo que permite realizar el proyecto mediante una conexión wifi sin la necesidad de usar un componente externo como podría ser un ESP8266.
Cuenta con una memoria flash de 256 kB siendo 8 veces mayor a la versión anterior, 32 kB SRAM y 8 kB de EEPROM, cuenta con un microcontrolador RA4M1 cuyo voltaje de operación es de 5 V, mientras que el ESP32-S3 funciona a 3.3V. Tiene 14 puertos digitales,

6 puertos analógicos, pines dedicados a conexiones I2C, SPI y UART. La velocidad de reloj es de 48 MHz, siendo 3 veces mayor que la de sus predecesores. Además de la conexión WiFi cuenta con bluetooth y una matriz de 12x8 LEDs.
El módulo ESP32-S3 cuenta con una antena para poder trabajar en la banda de 2.4 GHz.
Como se puede ver en la siguiente figura, ciertos pines pueden ser usados para múltiples funciones, los principales pines que se van a usar durante este proyecto serán los siguientes:

- +5V
- GND
- Pines analógicos: A0, A1, A2, A3, A4, A5
- Pines digitales: D0, D1, D2, D3, D4, D5, D6, D7, D8, D9

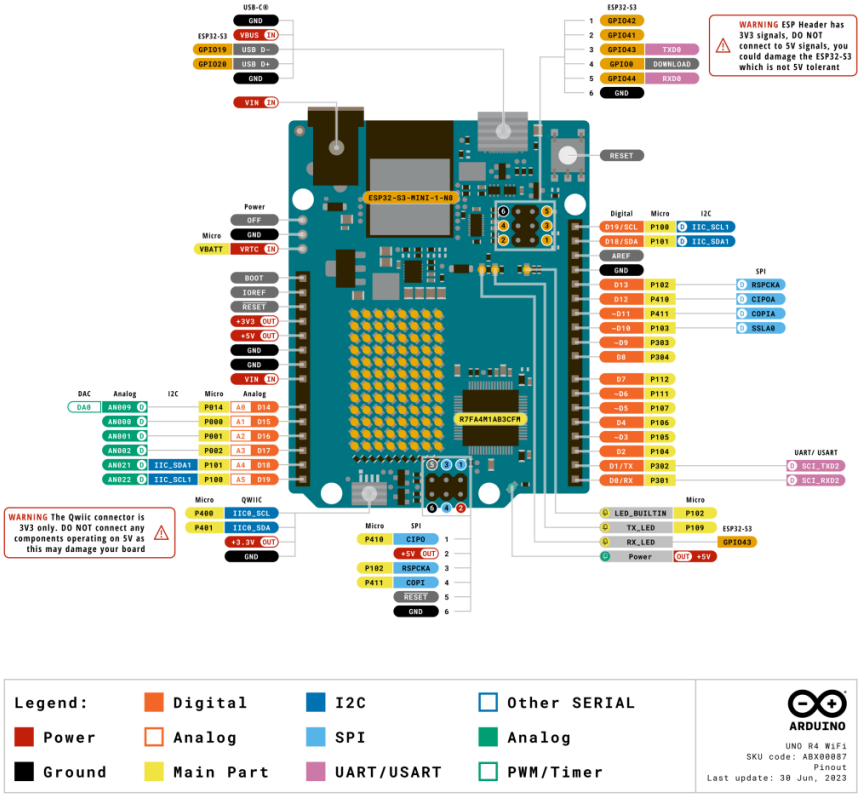


Figure 6. Pines Arduino UNO R4 WiFi

2.3.2.2. Sensor de temperatura y humedad DHT22

Se trata de un sensor de temperatura y humedad del aire de bajo coste, existe una versión más económica que sería el DHT11, pero se decidió hacer uso del DHT22 [14] dado que los rangos de algunas de sus características encajaban en mejor medida con las necesidades del proyecto.

Figure 7. Sensor de temperatura y humedad AZDelivery DHT22

El rango de temperaturas es ideal para este proyecto, ya que abarca desde -40°C hasta 80°C y es necesario dado que la localización del prototipo se encuentra en un área de bajas temperaturas, por lo que si se hubiera elegido un sensor con características más limitadas no lograría su objetivo. Opera a 3-5V por lo que encaja con los valores de Arduino. Cuenta con 4 pines:

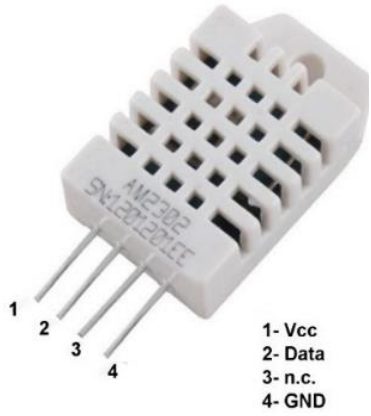


Figure 8. Pines sensor de temperatura y humedad AZDelivery DHT22

2.3.2.3. Sensor de humedad del suelo LM393

Para medir la humedad del suelo se ha elegido el sensor Hailege LM393 [15], el módulo permite un voltaje de entrada de 3.3V-5V por lo que es ideal para ser usado junto con un Arduino, cuenta con un limitador ajustable mediante un potenciómetro, el sensor tiene dos entradas, una marcada con" +" y otra con "-", compara ambos valores calculando la diferencia de voltaje entre ambos y de acuerdo con ese valor trasmite una salida acorde a ese valor.

Figure 9. Sensor de humedad de suelo Hailege LM393

Como se puede ver en la siguiente figura el sensor cuenta con 6 pines:

- VCC
- GND
- DO: Salida digital
- AO: Salida analógica
- +: entrada positiva
- -: entrada negativa

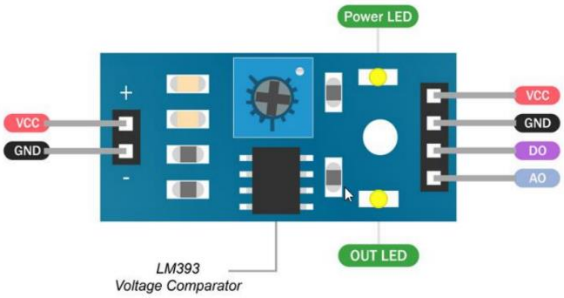


Figure 10. Pines sensor de humedad de suelo Hailege LM393

2.3.2.4. Sensor de luminosidad KY-018

El sensor de luminosidad permitirá medir la cantidad de luz ambiental, KY-018 [16] permite un voltaje de operación de 3.3V-5V, lo que permite conectarlo directamente al Arduino. Tiene una salida analógica, cuanto mayor sea la cantidad de luz recibida, menor resistencia presentará.

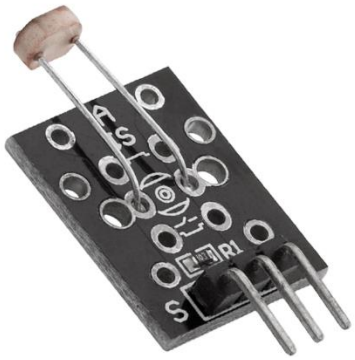


Figure 11. Sensor de luminosidad AZDelivery KY-018

Cuenta con 3 pines como se muestra en la siguiente figura:

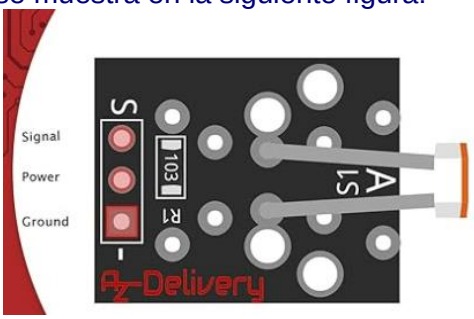


Figure 12. Pines sensor de luminosidad AZDelivery KY-018

2.3.2.5. Sensor del nivel de agua

El sensor del nivel de agua [17] fue diseñado específicamente para proyectos con Arduino, trabaja en un rango de 3-V y menos de 20mA, se trata de un sensor analógico y permite un rango de humedad del 10%-90%, cuenta con 3 pines:

- S: señal analógica
- +: VCC
- -: GND

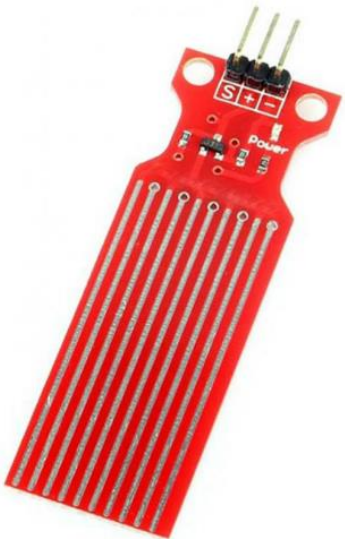


Figure 13 . Sensor de nivel de agua

2.3.2.6. Relé

Mediante el relé [18] se controla el motor que hará que la bomba de agua funcione y se realice el riego. Tiene un voltaje de entrada de 5V que estará conectada de manera independiente para que su alto consumo no afecte al funcionamiento del resto del conjunto de sensores.

Figure 14. Relé

Cuenta con 3 terminales y 3 pines:

- NC: Normalmente cerrado
- NO: Normalmente abierto
- C: Terminal común
- Ground
- 5V
- Señal

Figure 15. Pines Relé

2.4. Diseño Software

Tras hablar del diseño Hardware, ahora nos enfocaremos en el diseño Software del sistema de riego automatizado, donde se incluye la interfaz de usuario, la comunicación de la base de datos tanto desde la aplicación móvil como desde Arduino y la lógica de control utilizada. Nos centraremos en la creación de la aplicación móvil, la conexión con la base de datos, además de la forma de cómo interconectar las distintas partes del proyecto mediante WiFi y la programación usada en el Arduino para realizar las operaciones necesarias en el sistema de riego y en la lectura de los sensores. A pesar de que se trata de un proyecto interconexionado primero se deben definir por separado cada parte del proyecto para de esta manera poder implementar más tarde el sistema de manera conjunta.

2.4.1. Programación del Arduino

El microcontrolador Arduino necesita un código mediante el cual se indicará la manera de tomar los valores ofrecidos por los sensores y activar las acciones que permitan controlar los actuadores conectados al Arduino, además mediante este código se indicará la configuración de cada pin conectado al Arduino ya sea de entrada o salida.
El Arduino actúa como pieza central de la estructura, ya que coordinará los valores obtenidos por los sensores con la información que reciba de la aplicación móvil a través de la base de datos y activará o desactivará el riego tras que todos estos parámetros sean analizados.

Para implementarlo se usa IDE, tratándose del entorno desarrollado por Arduino para programar el Arduino en uso. Para este proyecto comenzaremos con la estructura básica que luego se irá ampliando con nuevas funciones para lograr todos los requerimientos que hemos especificado. La estructura básica serán las variables definidas, seguidas de “void

setup()" que será la función encargada de realizar las configuraciones iniciales cuando se enciende o restablece el Arduino y la última parte de la estructura básica es "void loop()" la cual se ejecutará indefinidamente hasta que el Arduino se apague o se reinicie pasando en este último caso de nuevo a la función "void setup()".

Los pines de entrada/salida se definirán al inicio permitiendo así la conexión a los sensores o al relé a continuación en el "void setup()", tal y como se muestra en la siguiente figura:

dht22.ino ReadMe.adoc

```
#include <DHT.h>
#include <Wire.h>
#include <BH1750.h>
#include <WiFi.h>

int SENSOR = 2;
int temp, humedad;
DHT dht (SENSOR, DHT22);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
 humedad = dht.readHumidity();
 temp = dht.readTemperature();
if (isnan(humedad) || isnan(temp)) {
Serial.println("Error al leer el sensor DHT!");
return;
}

 Serial.print("Temperatura: ");
 Serial.print(temp);
 Serial.print("ºC Humedad: ");
 Serial.print(humedad);
 Serial.println("%");
 delay(2000);
 }

```

Figure 16. Código en IDE con funciones básicas para Arduino

Tal y como se ve en la figura anterior una vez establecido el pin del sensor se pasa a la función "void loop()", donde entraremos en el bucle para leer los valores del sensor de manera indefinida, en este caso recibiendo los valores de humedad y temperatura ambiente. En caso de que el sensor sufra algún problema y no se reciba la información esperada se recibirá un mensaje de aviso y si todo es correcto se imprimirán los valores para que el usuario tenga acceso a ellos.

Se ha decidido que la comunicación serie se establezca con "Serial.begin(9600)", "Serial" hace referencia a que el puerto de comunicación que se está usando es la serie, mientras que "begin(9600)" indica la velocidad de transmisión de los datos, en este caso 9600 bits por segundo.

Estas son las funciones básicas con las que se inicia el proyecto, pero una vez generada la base de datos se podrán recibir datos desde la aplicación móvil, como puede ser la humedad necesaria para el correcto cuidado de la planta, por ello se establece constantes con la información recibida como se muestra en la siguiente figura:

```
200  void obtenerConfiguracionDesdeFirebase() {
201      // Obtener configuraciones desde Firebase
202
203    Firebase.RTDB.getInt(&fbdo, direccion + "/configuracion/humedadSueloMinima");
204    if (fbdo.dataType() == "int") {
205      humedadSueloMinima = fbdo.intData();
206    }
```

Figure 17. Código en IDE para leer datos desde Firebase

Tras haber guardado los datos recibidos desde la aplicación móvil se podrán comparar con los recibidos por los sensores y de esta manera evaluar la acción necesaria, que puede variar desde mandar un mensaje a la aplicación móvil a activar/desactivar el riego. La activación automática de riego no debería ser activada por el Arduino sin la acción previa del usuario a no ser que se cumplan unas condiciones ambientales extremas como son las definidas a continuación, mediante las cuales se permitirá la activación de emergencia.

```
if (humedad_suelo < humedadSueloMinima ) {
Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones", " Nivel de humedad del suelo extremadamente bajo! Activar riego automáticamente.");
abrir_valvula();
}
```

Figure 18. Código en IDE para activación de riego automático

Conforme se va dando forma al proyecto se necesitarán una mayor cantidad de funciones además de las dos básicas explicadas anteriormente, las funciones completas se pueden encontrar en el anexo A, pero algunas de ellas son la función de abrir la válvula de riego mediante la activación del relé que permanecerá activa por un periodo definido de tiempo antes de volver a desactivarse.

```
void abrir_valvula () {
    digitalWrite(bomba_PIN, HIGH);
    delay(10000);
    digitalWrite(bomba_PIN, LOW);
}
```

Figure 19. Código en IDE para apertura de la válvula

También se deben tener en cuenta las librerías, ya que sin ellas no se podrá acceder a las funciones predefinidas de estas, se incluyen en el código del proyecto mediante “#include”, cómo puede ser el caso de la librería “#include <DHT.h>” que permite realizar las funciones necesarias para acceder a los datos del sensor DHT22 que hemos mostrado anteriormente.

Al inicio del proyecto se decidió que la conexión se debería hacer mediante Wifi, por este motivo se decidió adquirir un Arduino que tuviera integrado un módulo wifi, dado que de esta manera contamos con la posibilidad de conectarnos a redes inalámbricas y comunicarnos con los servicios necesarios sin que sea necesario el uso de un módulo intermedio como podría ser un módulo ESP8266. La librería necesaria para permitir su uso es “#include WiFi.h”. Si tuviéramos que usar un módulo externo para poder conectarnos a la base de datos la probabilidad de pérdidas de datos se incrementaron, por lo que de esta manera nos aseguramos de que los datos recibidos por los sensores en el Arduino sean transmitidos directamente a la base de datos además de reducir el coste ya que eliminamos uno de los

módulos. Como ya se explicó en las especificaciones del Arduino el módulo WIFI integrado es un ESP32-S3, mediante el cual se podrá verificar los datos de Wifi antes de proceder al envío y recepción de información con la base de datos. Antes de comenzar la transmisión de datos se deben declarar las credenciales SSID y la contraseña de la red WiFi, para a continuación tal y como se muestra en la figura inicializar la conexión en “void setup()” indicando la dirección IP correspondiente.

```
24  // Insert your network credentials
25  #define WIFI_SSID "Pinguine zu verk
26  #define WIFI_PASSWORD "Proteincooki
27
```

Figure 20. Código en IDE para conexión WiFi

```
56  void setup(){
57    Serial.begin(9600);
58    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
59    Serial.print("Connecting to Wi-Fi");
60    while (WiFi.status() != WL_CONNECTED){
61      Serial.print(".");
62      delay(300);
63    }
64    Serial.println();
65    Serial.print("Connected with IP: ");
66    Serial.println(WiFi.localIP());
67    Serial.println();
```

Figure 21. Código en IDE para conexión WiFi

Además, para poder acceder a la base de datos y poder enviar los valores obtenidos mediante los sensores es necesario definir los parámetros que permite identificar la base de datos en Firebase.

```
28  // Insert Firebase project API Key
29  #define API_KEY "AIzaSyAMHhj6nOhwPjadh2c2wrsl8_wDgo
30
31  // Insert RTDB URLefine the RTDB URL */
32  #define DATABASE_URL "https://riego-automatico-44be
33
34  //Define Firebase Data object
35  FirebaseData fbdo;
36
37  FirebaseAuth auth;
38  FirebaseConfig config;
```

Figure 22. Código en IDE para inicialización de la base de datos

2.4.2. Base de datos

La base de datos va a ser una parte clave del proyecto, ya que se encargará de conectarse con la aplicación para almacenar los datos que se proporcionen desde esta y al mismo tiempo compartir los valores que almacene por parte del Arduino, de la misma forma se encarga de autentificar a los usuarios y de denegar su acceso si no es el usuario correcto. Se ha elegido Firebase dado que nos permite gestionar los datos en tiempo real actualizando automáticamente y en cada ciclo del bucle que se produce desde el Arduino, ya que para este proyecto no es necesario almacenar un largo registro de datos sino permitir la mayor eficiencia y fiabilidad a la hora de intercambiar los cambios producidos entre Arduino y la aplicación móvil.

Figure 23. configuración proyecto en Firebase

Cuando se crea el proyecto en Firebase además de darle nombre al proyecto se deben establecer reglas donde se establecen los permisos para leer y escribir, inicialmente con fines de prueba durante el desarrollo del proyecto se establece como “true” para ambos, pero a lo largo del proyecto cuando se creen los requisitos y los usuarios correspondientes esas reglas deberán ser modificadas.

Figure 24. Reglas para los permisos de la base de datos

En la página principal de Realtime Database se puede ver la identificación del proyecto de Firebase junto con las secciones que se pueden crear, ya sea directamente desde Firebase

o como se muestra, la sección “sensor” que se crea tras almacenar los datos de los sensores desde Arduino o “configuración” que proviene de los datos proporcionados por el usuario desde la aplicación móvil. También se pueden crear secciones para los distintos usuarios, en este ejemplo mostrado a continuación por ser la primera versión solo cuenta con un usuario que podría ser ampliado en un futuro durante la realización del proyecto en caso de ser necesario.

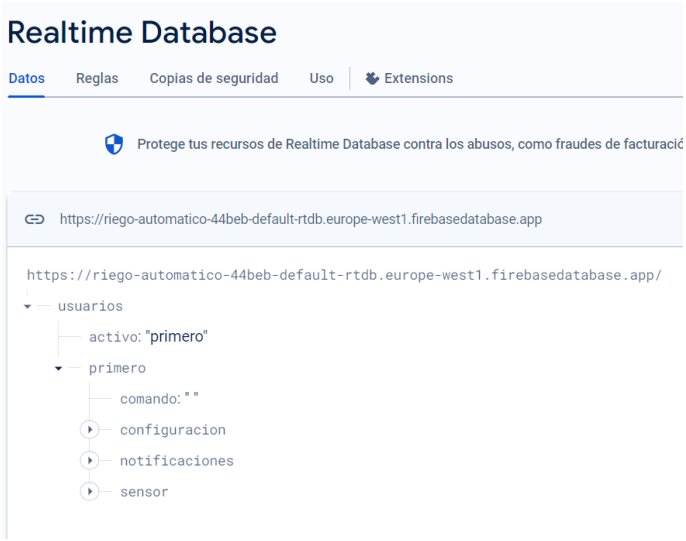


Figure 25. Vista general de la base de datos Realtime Database

2.4.3. Aplicación móvil

Para el diseño de la aplicación móvil se ha usado Ionic Framework usando Angular, la ventaja de usar Ionic es que permite mediante un mismo código crear la aplicación móvil para las principales plataformas, como Android o iOS.
La aplicación diseñada en Ionic se encargará de leer los datos proporcionados por la base de datos en Firebase para poder leerlos desde la app, añadir información a la base de datos, con la posibilidad de modificarlo más tarde en caso de ser necesario y activar el riego automático desde la aplicación móvil. La aplicación está diseñada para que antes de poder acceder al interior de la aplicación el usuario debe estar registrado e identificarse, para de esta manera poder ubicarse en la base de datos con la correcta información.
El proyecto en Ionic tendrá múltiples páginas dedicadas a cada ventana que esté visible en la aplicación móvil.

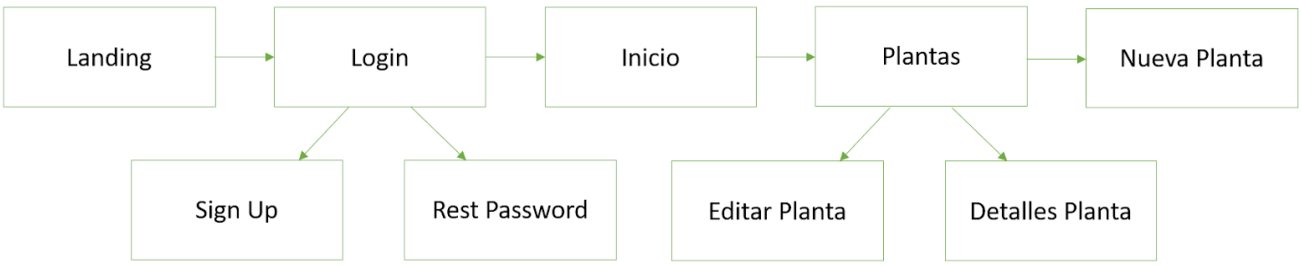


Figure 26. Organización de la aplicación móvil

La primera ventana que verá el usuario cuando abra la aplicación será “landing”, desde aquí se muestra el nombre de la aplicación y un botón para avanzar a la pantalla de “login”.

Desde Ionic tendremos diferentes partes del código, cada una tiene una finalidad, por ejemplo, tendremos dentro de “landing” un .html que se encargará de hacer visible las variables definidas en el código principal o de crear botones, tablas, seleccionar la imagen de fondo que deseamos, etc. En este caso para la ventana “landing” hemos puesto una imagen de fondo, junto con un título y un botón que nos lleva a la siguiente ventana.

```
<ion-content >
  <div class="container">

    <ion-img src="assets/riego.jpg"/>
    <ion-title>Riego <p>automatico</p> </ion-title>
    <ion-fab  vertical="bottom" horizontal="end" style="margin-bottom:
3px;" >
    <ion-fab-button [routerLink]="['/login']"  >
      <ion-icon name="chevron-forward"></ion-icon>
```

Figure 27. Código langing.html en Ionic

También tendremos un .scss desde el cual se definirá el tamaño, los márgenes que deseamos para cada parte de la ventana junto con los colores o sombras.

```scss
.container {
    width: 100%;
    height: 100vh;
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
    align-items: center;
  }
  ion-title {
    margin-top: -1200px;
    font-size: 38px;
    font-weight: bold;
    color: #333;
    text-align: center;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
  }
ion-img {
    width: 100%;
    height: 100%;
    object-fit: cover;
  }
```

Figure 28. Código langing.scss en Ionic

En este caso como es simplemente una ventana que nos lleva a la siguiente antes incluso de que el usuario se registre en la aplicación el .ts está vacío, pero en las siguientes ventanas se mostrará el contenido de este.

```typescript
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-landing',
  templateUrl: './landing.page.html',
  styleUrls: ['./landing.page.scss'],
})
export class LandingPage implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Figure 29. Código langing.ts en Ionic

Finalmente, esta primera ventana queda de la siguiente manera:

Figure 30. Ventana landing generada en Ionic

Tras pulsar el botón el usuario llegara a la siguiente ventana “login”

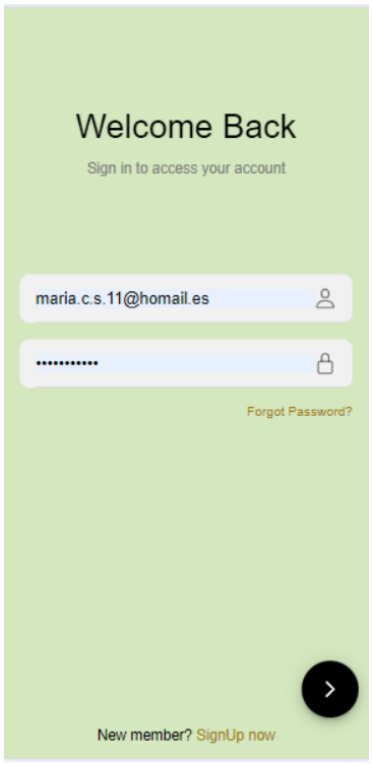


Figure 31. Ventana login generada en Ionic

Esta ventana es más compleja que la anterior, ya que contiene texto que te lleva a otras ventanas además del botón del que hablamos anteriormente. Esto se logra desde .html indicando el link hacia la nueva ventana.

```
      <div class="forgotpassword">
        <a [routerLink]="['/reset-password']" style="color:
#9f7d0e">Forgot Password?</a>
      </div>
      <div style="padding-top: -26px;padding-bottom: 16px;"
*ngIf="this.loginForm.controls?.['password'].touched &&
this.loginForm.controls?.['password'].invalid">
        <ion-text >Password is required.</ion-text>
    </div>
```

Figure 32. Código login.html en Ionic

En “login” necesitaremos una myor cantidad de código en .ts dado que necesitamos acceder a la base de datos para buscar si el usuario que se está introduciendo es valido, ademas antes de realizar ese paso se debe comprobar que los datos introducidos son válidos, por ejemplo, que el patrón de caracteres introducidos en el apartado email se corresponda con la forma general de un email, que sería texto@texto.texto

```
ngOnInit() {
  this.loginForm = this.formBuilder.group({
    email: [
      '',
      [
        Validators.required,
        Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+.[a-z]{2,3}$'),
      ],
    ],
    password: ['', [

Validators.pattern('(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[$@$!%*?&])[A-Z
a-z\d$@$!%*?&].{8,}'),
        Validators.required,
    ]
    ],
  });
}
```

Figure 33. Código login.ts en Ionic

Se comprobará en Firebase Authentification que los datos introducidos existen ya como usuario y en ese caso nos llevará a la siguiente ventana que será la de “inicio”.

```
  async login() {
    const loading = await this.loadingController.create();
    await loading.present();
    if (this.loginForm.valid) {

      const userCred = await
this.authService.loginUser(this.loginForm.value.email,
this.loginForm.value.password).catch((err) => {
        this.presentToast(err)
        console.log(err);
        loading.dismiss();
      })

      if (userCred) {
        loading.dismiss();
        const userId = userCred.user?.uid;

if(userId){this.data.database.ref(`usuarios/activo`).set(userId);}
        this.router.navigate(
          ['/inicio'])
      }
    } else {
      return console.log('Please provide all the required values!');
    }

  }
  get errorControl() {
    return this.loginForm?.controls;
  }
```

Figure 34. Código login.ts en Ionic

En la ventana “signup” tenemos una distribución similar a la que encontramos en “login”, dos recuadros donde incluir el email correspondiente y la contraseña deseada, donde se ponen ciertas condiciones para garantizar la seguridad de la cuenta como se muestra en la siguiente imagen. En caso de que se haya accedido a esta ventana por error existe la posibilidad de volver a la ventana “login” y en caso contrario de que se desee crear la cuenta se dispone de un botón para incluir los datos en la base de datos.

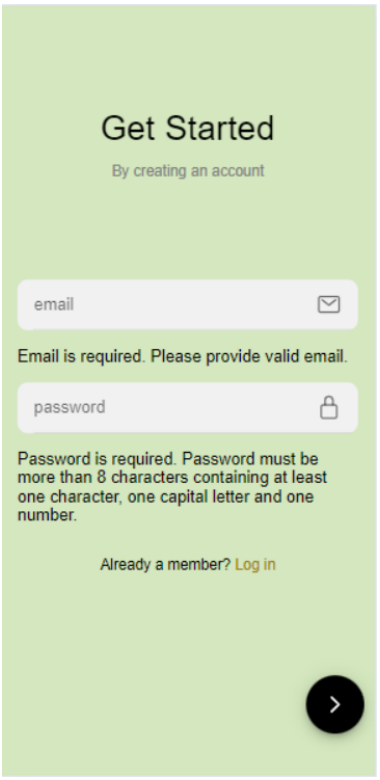


Figure 35. Ventana sign in generada en Ionic

En caso de que los datos introducidos sean válidos, se almacenarán en Firebase Authentification donde se creará una identificación para el usuario mediante la cual se podrá acceder a sus datos. Tras este proceso se irá a la ventana “inicio” donde se tendrá acceso a los valores de los sensores y se podrán incluir las plantas del usuario.

```
async signUP(){
  const loading = await this.loadingController.create();
  await loading.present();
  if (this.signForm.valid) {

    const user = await
this.authService.registerUser(this.signForm.value.email,
this.signForm.value.password).catch((err) => {
      this.presentToast(err)
      console.log(err);
      loading.dismiss();
    })

    if (user) {
      loading.dismiss();
      const userId = user.user?.uid;

if(userId){this.data.database.ref(`usuarios/activo`).set(userId);}
      this.router.navigate(['/inicio'])
    }
  } else {
    return console.log('Please provide all the required values!');
  }
}

async presentToast(message: undefined) {
  console.log(message);

  const toast = await this.toastController.create({
    message: message,
    duration: 1500,
    position: 'top',
  });

  await toast.present();
}
```

Figure 36. Código singin.ts en Ionic

En caso de que el usuario ya tenga una cuenta, pero se haya olvidado de la contraseña, accediendo a Firebase se comprobará si el email proporcionado por el usuario efectivamente está ya registrado y en caso correcto se enviará a dicha dirección un email solicitando el cambio de contraseña, tras enviar el email, se volverá a la ventana “login” para que el usuario pueda ingresar la nueva contraseña.

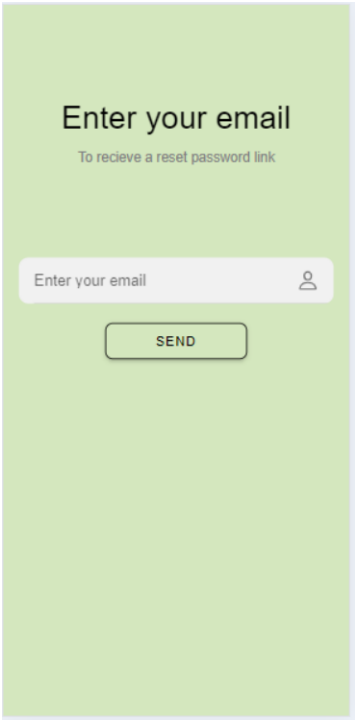


Figure 37. Ventana reset password generada en Ionic

```
reset(){
  this.authService.resetPassword(this.email).then( () =>{
    console.log('sent'); //show confirmation dialog
    this.presentToast()
  })
}
async presentToast() {
  const toast = await this.toastController.create({
    message: 'Your reset password link has been sent on your email',
    duration: 2000, // Duration in milliseconds
    position: 'bottom' // Position of the toast (top, bottom, middle)
  });

  toast.present();
  toast.onDidDismiss().then(()=>{
    this.router.navigate(['/login']);
  })
 }
}
```

Figure 38. Código reset-password.ts en Ionic

La ventana “inicio” sería la ventana central de la aplicación, desde esta ventana se podrán observar los valores de los sensores en tiempo real, se verán las notificaciones que se definieron en el código creado en Arduino, además de permitir cerrar sesión al usuario, activar el riego desde la aplicación y acceder a la ventana “plantas”, donde se encontrarán los datos de las plantas que posee el usuario.

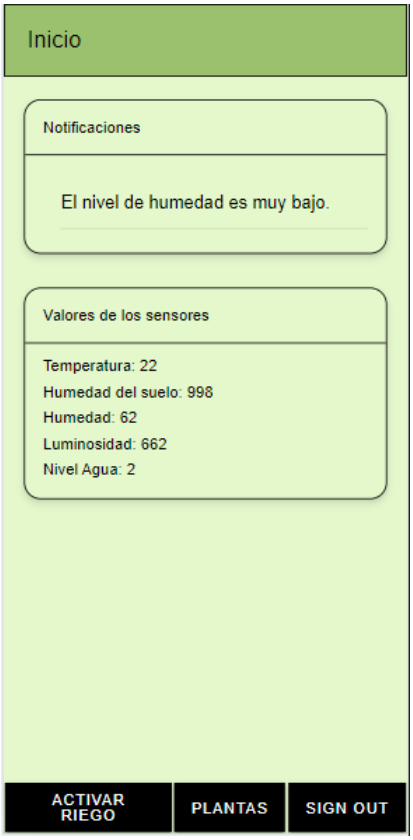


Figure 39. Ventana inicio generada en Ionic

Lo primero que se hace en esta ventana es guardar los datos del usuario tales como email y nombre usuario para poder acceder a la dirección correcta en la base de datos, para a continuación acceder con esos datos a las funciones “obtenerValoresSensores” y “ver Notificaciones”

```
ngOnInit(): void {


  this.authService.getProfile().then(user => {
    this.email = user?.email;
    this.nombre = user?.uid;
    console.log(user?.email);
    this.obtenerValoresSensores();
    this.verNotificaciones();
  }).catch(error => {
    console.error('Error getting user profile:', error);
  });


}
```

Figure 40. Código inicio.ts en Ionic

Cuando el usuario pulse el botón activar riego, lo que provocará es que se escriba en la base de datos “activar” para que tal y como se ha diseñado el código en Arduino este sepa que debe activar la bomba de agua.

```
 activarRiego(){

this.data.database.ref(`usuarios/`+this.nombre+'/comando').set("activar")
.then(() => {
     console.log('Riego activado.');
   }).catch((error) => {
     console.error('Error al activar el riego: ', error);
   });

  }
```

Figure 41. Código inicio.ts en Ionic

Para obtener los valores de los sensores se deberá acceder a la información almacenada en la base de datos, haciendo uso de “valueChanges()” para de esta manera poder actualizar los valores que aparecen en la aplicación cada vez que el valor del sensor se modifica en la base de datos.

```
obtenerValoresSensores(){

this.temperatura=this.data.object('usuarios/'+this.nombre+'/sensor/temper
atura').valueChanges().subscribe((valor)=>{
   console.log("Temperatura:", valor);
   this.temperatura = valor;
 });
```

Figure 42. Código inicio.ts en Ionic

Recibir las notificaciones funciona de la misma manera que cuando se obtienen los valores de los sensores con una modificación y se trata de que en este caso se permite al usuario borrar la notificación una vez que la ha leído, de manera que se elimina también de la base de datos.

```
 verNotificaciones() {
   this.data.object('usuarios/' + this.nombre +
'/notificaciones').valueChanges().subscribe((notificaciones) => {
     if (notificaciones) {
       this.notificaciones = Object.keys(notificaciones).map(key => ({
         id: key,
         contenido: notificaciones[key]
       }));
     }
   });
 }

 delete(id: String){

   this.data.database.ref('usuarios/'+this.nombre+ '/notificaciones/' +
id).remove()
    .then(() => {
       console.log('Notificacion eliminada correctamente.');
     this.router.navigate(['/inicio'])
    })
    .catch((error) => {
      console.error('Error al eliminar la notificacion: ', error);
    });

 }

}
```

Figure 43. Código inicio.ts en Ionic

Cuando el usuario pulsa en el botón “plantas” accede a la siguiente ventana “plantas”, donde puede encontrar un botón para añadir otra planta, volver a la ventana “inicio” y el listado de las plantas que ya ha registrado en la aplicación.

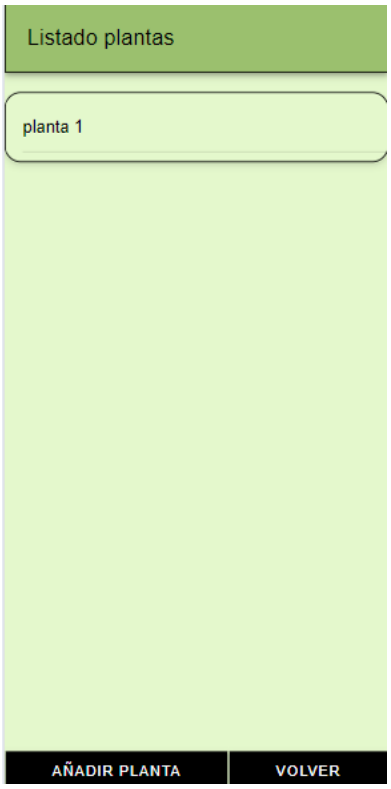


Figure 44. Ventana plantas generada en Ionic

Cuando se pulsa en “añadir plantas” se redirige a la ventana “nueva planta” al igual que cuando se pulsa “volver” se abre la ventana “inicio”.
El listado de plantas permite desplazar la planta mostrando a la izquierda un botón “details” que te lleva a la ventana “detalles planta” o si se desliza a la derecha se ve “edit” y “delete”, mediante “edit” se redirige a la ventana “editar planta” y si se pulsa “delete” no solo se borra de la aplicación, sino que también de la base de datos de la misma manera que se ha realizado con las notificaciones.

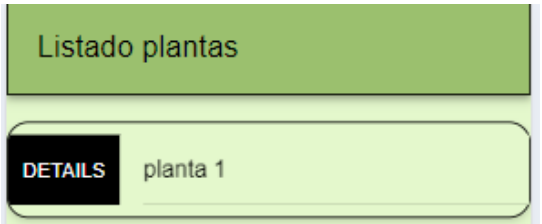


Figure 45. Ventana plantas generada en Ionic

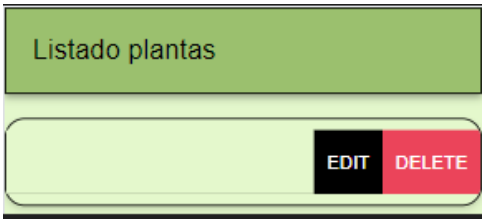


Figure 46. Ventana plantas generada en Ionic

Cuando el usuario accede a la ventana “nueva planta” debe incluir los valores de la planta que serán almacenados en la base de datos bajo el nombre que se haya usado al crearla, en caso de pulsar el botón “cancelar” no se guardaran los datos que el usuario haya escrito, cuando se pulse cualquiera de los dos botones existentes se regresará a la ventana ”plantas”.

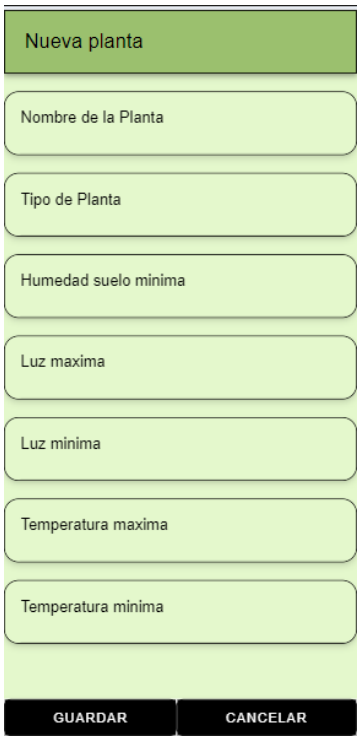


Figure 47. Ventana nueva planta generada en Ionic

Cuando el usuario accede a “detalles planta” el código definido busca en la base de datos el nombre de la planta seleccionada y muestra la información almacenada sobre esta, esto se consigue accediendo al parámetro seleccionado mediante .subscribe()

```
ngOnInit() {

  this.authService.getProfile().then(user => {
    this.email = user?.email;
    this.nombre = user?.uid;
    console.log(user?.email);
    this.route.params.subscribe(params => {
      this.nombrePlanta = params['nombrePlanta'];
      this.obtenerDetallesPlanta(this.nombrePlanta);
    });
  }).catch(error => {
    console.error('Error getting user profile:', error);
  });

}

obtenerDetallesPlanta(nombrePlanta: string) {

his.data.object('usuarios/'+this.nombre+'/configuracion/'+nombrePlanta).
alueChanges().subscribe(valor=>{
      this.detallesPlanta = valor;
    console.log(valor);

  });
}
volver(){
  this.router.navigate(['/plantas'])

}
```

Figure 48. Código nueva-planta.ts en Ionic

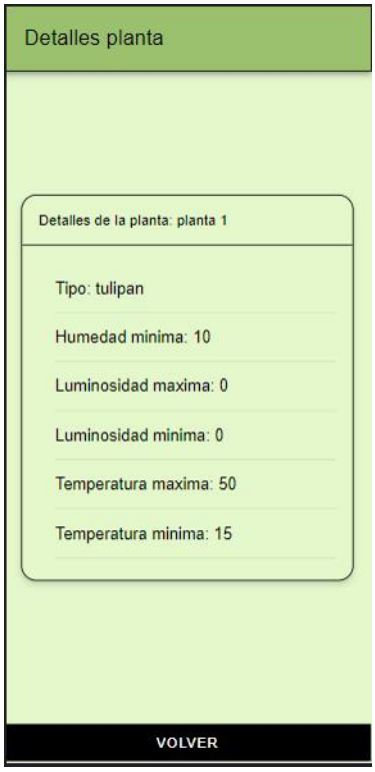


Figure 49. Ventana detalles planta generada en Ionic

Cuando se accede a “editar planta” se realiza el mismo procedimiento que cuando queríamos ver los detalles de la planta para obtener los valores almacenados, solo que en adición a ese paso ahora se muestran en un editable para de esta manera poder actualizar los parámetros necesarios, esto se consigue mediante .update().

```
guardarPlanta(){


this.data.database.ref(`usuarios/`+this.nombre+'/configuracion/'+this.nom
brePlanta).update({
     tipo: this.tipoPlanta,
     humedad: this.humedad,
     luzMax: this.luzMax,
     tempMax: this.tempMax,
     luzMin: this.luzMin,
     tempMin: this.tempMin,
   }).then(() => {
     console.log('Planta guardada exitosamente.');

     if (this.nombrePlanta !== this.nombrePlantaOriginal) {
       console.log('Nuevo nombre de la planta:', this.nombrePlanta);

this.data.database.ref(`usuarios/`+this.nombre+'/configuracion/'+this.nom
brePlantaOriginal).remove();
     }
     this.router.navigate(['/plantas', { nombrePlanta: this.nombrePlanta
}])
   }).catch((error) => {
     console.error('Error al guardar la planta: ', error);
   });

 }

 cancelar(){
   this.router.navigate(['/plantas'])
 }
```

Figure 50. Código editar-planta.ts en Ionic

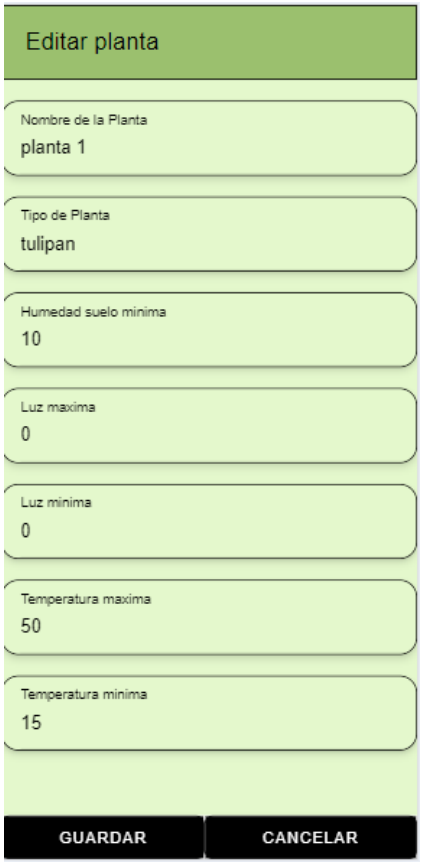


Figure 51. Ventana editar planta generada en Ionic

2.5. Presupuesto

A continuación, una vez que se han analizado y especificado todos los componentes que se necesitan para la realización del prototipo del sistema de riego automático se indican los costes que tienen estos componentes.

Componente	Precio
Arduino UNO R4 WiFi	29,75
Jumper Wire	7
Relé	1,4
LM393	1,4
Sensor nivel agua	1,4
Breadboard	3
DHT22	6,33
KY-018	2
Bomba de agua	7,69
Total	59,97

3. Resultados

Tras ver de manera individual cada parte que forma parte del proyecto ahora debemos analizar cómo realizar la integración de las distintas partes en una estructura completa mediante la cual se permite acceso unas a otras para de esta manera poder analizar, actualizar y crear parámetros para el funcionamiento del proyecto.

3.1. Estructura completa

A continuación, se mostrará en detalle las conexiones que existen entre las tres partes clave del proyecto, comenzando por la conexión entre los distintos sensores y válvulas con el Arduino, continuando con la conexión entre el Arduino y la base de datos y terminando con la conexión entre la base de datos y la aplicación móvil.

3.1.1. Conexión entre sensores y Arduino

Comenzaremos mostrando las conexiones independientes de cada sensor mediante la cual se conseguirá obtener la información necesaria para el funcionamiento del proyecto, los esquemáticos se han realizado con el programa fritzing [19].

3.1.1.1. DHT22

Podemos observar las conexiones entre el sensor DHT22 y el Arduino, donde se conectan los pines de VCC y GND a los del Arduino y la salida digital del sensor al pin 2 del Arduino.

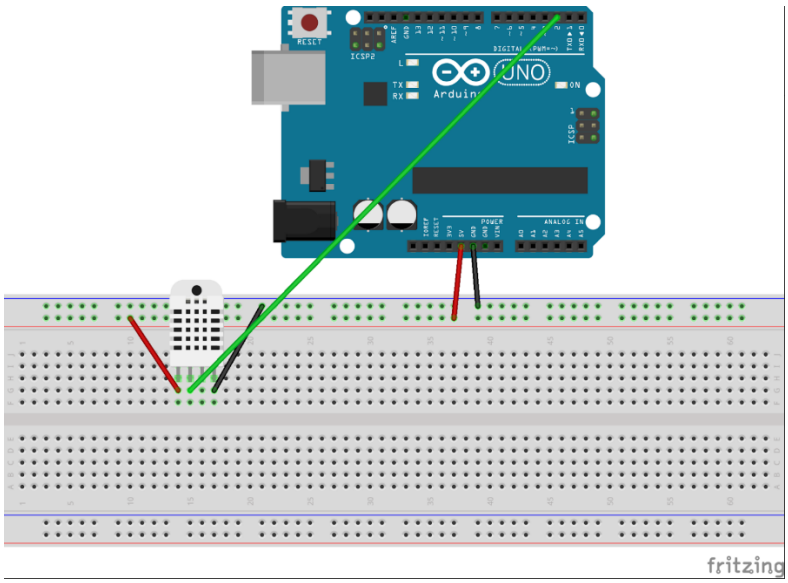


Figure 52. Esquema conexión DHT22 con Arduino

3.1.1.2. LM393

Podemos observar las conexiones entre el sensor LM393 y el Arduino, donde se conectan los pines de VCC y GND a los del Arduino y la salida analógica del sensor al pin A0 del Arduino.

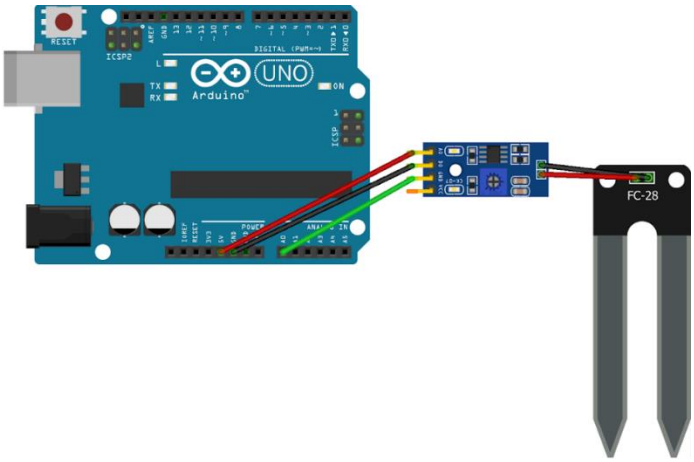


Figure 53. Esquema conexión LM393 con Arduino

3.1.1.3. KY-018

Podemos observar las conexiones entre el sensor KY-018 y el Arduino, donde se conectan los pines de VCC y GND a los del Arduino y la salida analógica del sensor al pin A2 del Arduino.

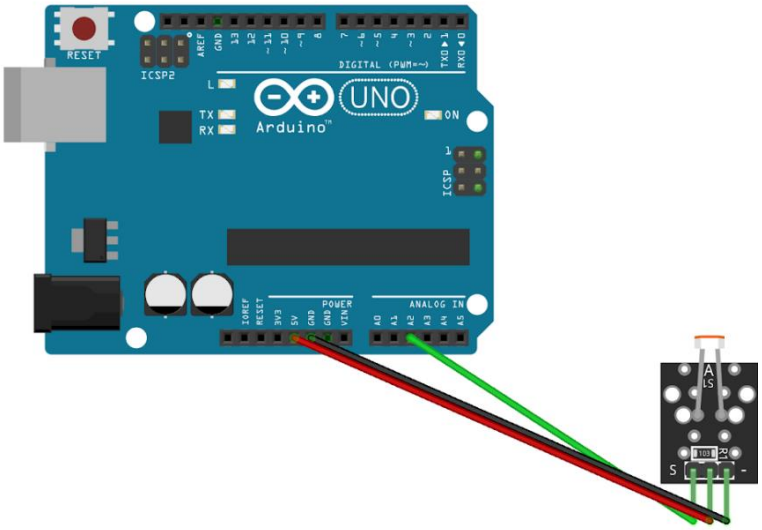


Figure 54. Esquema conexión KY-018 con Arduino

3.1.1.4. Sensor nivel del agua

Podemos observar las conexiones entre el sensor que mide el nivel del agua y el Arduino, donde se conectan los pines de VCC y GND a los del Arduino y la salida analógica del sensor al pin A1 del Arduino.

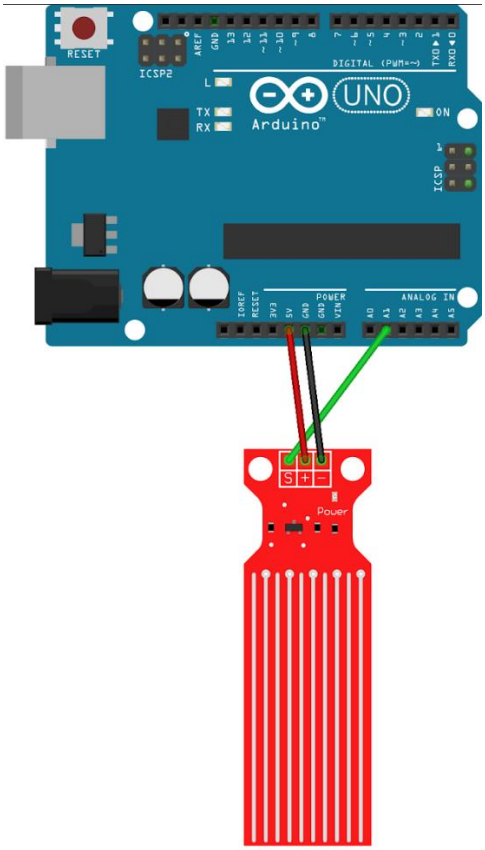


Figure 55. Esquema conexión sensor nivel del agua con Arduino

3.1.1.5. Relé

Podemos observar las conexiones entre el relé y el Arduino, donde se conectan los pines de VCC y GND a los del Arduino y la salida digital del sensor al pin 13 del Arduino, además el relé se conectara a una batería externa para poder poner en funcionamiento el motor mediante el cual al activarse cuando el relé lo permita se iniciará el riego, se debe contar con una batería externa ya que el consumo de potencia podría ser demasiado alto y provocar fallos si se usa la misma alimentación que para el Arduino.

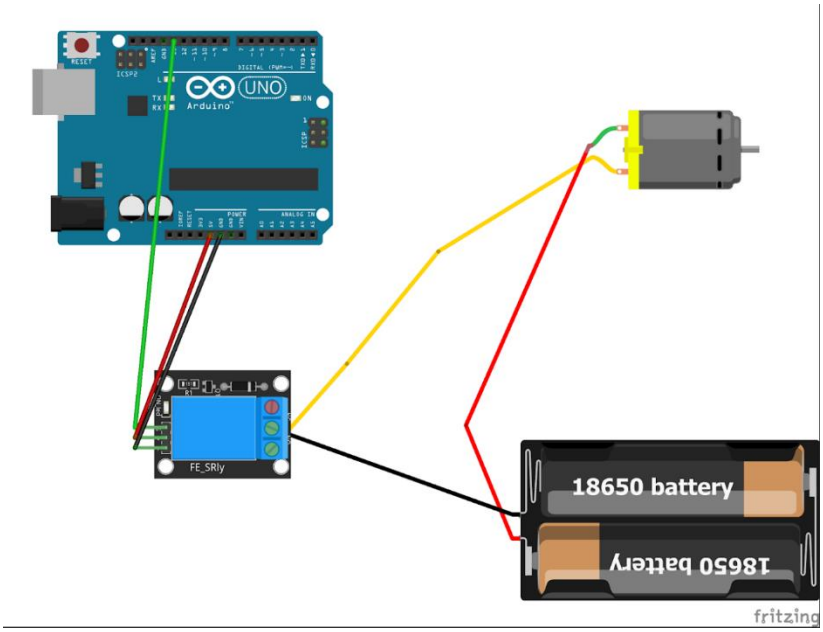


Figure 56. Esquema conexión relé con Arduino

3.1.1.6. Sistema completo

Tras comprobar el correcto funcionamiento de cada sensor de forma independiente se agrupan todos los sensores para poder llevar a cabo el prototipo del proyecto, a continuación, se muestra el esquema completo de las conexiones.

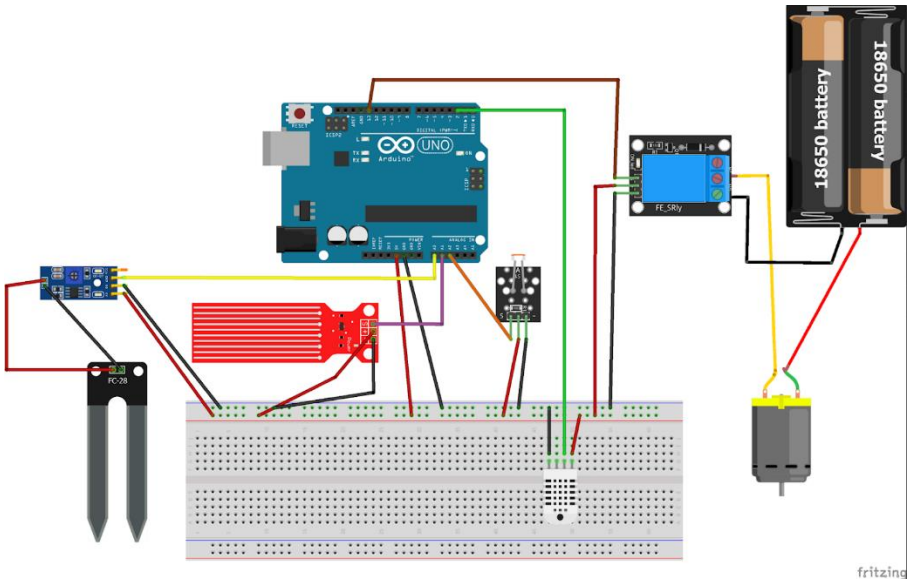


Figure 57. Esquema completo Arduino

3.1.2. Conexión entre Arduino y base de datos

Para realizar la conexión entre el Arduino y la base de datos Firebase es necesario tener la API-KEY y la dirección de la base de datos como se muestra a continuación.

```
// Insert Firebase project API Key
#define API_KEY "AIzaSyAMHhj6nOhwPjadh2c2wrsl8_wDgo1ic"

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://riego-automatico-44beb-default-rtdb.europe-west1.firebasedatabase.app/"
```

Figure 58. Código direccion Firebase en Arduino

Una vez establecida la conexión existirán dos fases principales, una de ellas será el obtener la información ya existente en la base de datos, para de esa manera poder obtener la configuración necesaria de cada planta, eso se conseguirá solicitando a Firebase los datos que se encuentran en una dirección concreta, como se le va a permitir al usuario asignar más de una planta y cada planta tendrá unas necesidades distintas respecto a su cuidado, se debe iterar por /configuracion para obtener un listado de todas las plantas y de esta manera poder analizarlas de manera independiente:

```
void obtenerConfiguracionDesdeFirebase() {
    if (Firebase.RTDB.getString(&fbdo, direccion +"/configuracion")) {
        if (fbdo.dataType() == "json") {
            FirebaseJson &json = fbdo.jsonObject();
            for (size_t i = 0; i < json.iteratorBegin(); i++) {
                int type;
                String key, value;
                json.iteratorGet(i, type, key, value);

                if (type == FirebaseJson::JSON_OBJECT && value.indexOf("nombrePlanta") != -1) {
                    String plantName = key.substring(key.lastIndexOf('/') + 1);
                    nombresPlantas.push_back(plantName);
                }
            }
    for (size_t i = 0; i < nombresPlantas.size(); ++i) {

        Serial.println(nombresPlantas[i]);
        Firebase.RTDB.getInt(&fbdo, direccion + "/configuracion/"+nombresPlantas[i]+"/humedad");
  if (fbdo.dataType() == "int") {
    humedadSueloMinima = fbdo.intData();
  }
  Firebase.RTDB.getInt(&fbdo, direccion + "/configuracion/"+nombresPlantas[i]+"/luzMin");
  if (fbdo.dataType() == "int") {
    luxMinimo = fbdo.intData();
  }
  Firebase.RTDB.getInt(&fbdo, direccion + "/configuracion/"+nombresPlantas[i]+"/luzMax");
  if (fbdo.dataType() == "int") {
    luxMaximo = fbdo.intData();
  }
  Firebase.RTDB.getInt(&fbdo, direccion + "/configuracion/"+nombresPlantas[i]+"/tempMin");
  if (fbdo.dataType() == "int") {
    tempMinima = fbdo.intData();
  }

  Firebase.RTDB.getInt(&fbdo, direccion + "/configuracion/"+nombresPlantas[i]+"/tempMax");
  if (fbdo.dataType() == "int") {
    tempMaxima = fbdo.intData();
  }
    }
        } else {
            Serial.println("Error: Data is not a JSON object");
        }
    } else {
        Serial.println("Error: Failed to read data from Firebase");
```

Figure 59. Código obtención datos de Firebase en Arduino

La otra fase es la de enviar información a Firebase para que la almacene para su uso en la aplicación, esto se consigue enviando a una dirección concreta el valor deseado:

```
123   // Write an Int number on the database path test/int
124   if (Firebase.RTDB.setInt(&fbdo, direccion + "/sensor/temperatura", temp)){
125     Serial.println("PASSED");
126
127   }
128   else {
129     Serial.println("FAILED Temperatura");
130     Serial.println("REASON: " + fbdo.errorReason());
131   }
```

Figure 60. Código envio de datos a Firebase en Arduino

Además de esto, se desea que se puedan recibir notificaciones, para ello primero se deberán almacenar en la base de datos, eso se consigue enviando a la base de datos en caso de que se cumplan los requisitos establecidos el aviso correspondiente mediante una orden "push", uno de estos requisitos puede ser que el nivel de humedad que recibe el Arduino desde el sensor sea menor que la humedad mínima que hemos recibido previamente desde la aplicación móvil a través de la base de datos.

```
void notificaciones(){
  if (nivel_agua < nivelAguaMinimo ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones", "El nivel de agua es muy bajo. Pon agua en el recipiente");
  }
  for (size_t i = 0; i < nombresPlantas.size(); ++i) {
 if (humedad_suelo < humedadSueloMinima + 100 ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+ ": El nivel de humedad es muy bajo.");
  }
  if (humedad_suelo < humedadSueloMinima ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ": Nivel de humedad del suelo extremadamente bajo! Activar riego automáticamente.");
  abrir_valvula();
  }

  if (temp < tempMinima ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ": Temperatura muy baja.");
  }
    if (temp > tempMaxima ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ": Temperatura muy alta.");
  }
  if (lux < luxMinimo && esDeDia()) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ": Necesita mas luz.");
  }
    if (lux > luxMaximo && esDeDia()) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ": Necesita menos luz.");
  }
  }
}
```

Figure 61. Código notificaciones en Arduino

Como los sensores estarán mandando información constantemente al Arduino y desde la aplicación móvil se pueden actualizar ciertos datos, es importante que se comprueben los datos recibidos de forma periódica, para que en caso de que los valores de los sensores varían, se modifiquen también en la base de datos al igual que si se recibe una orden desde la aplicación móvil, desde la base de datos lo podamos leer.

```
184     // Verificar y actualizar configuración periódicamente
185   if (millis() - lastConfigCheckTime >= CONFIG_CHECK_INTERVAL) {
186     obtenerConfiguracionDesdeFirebase();
187     notificaciones();
188     lastConfigCheckTime = millis(); // Actualizar tiempo de la última verificación
189
190   }
191
192 }
```

Figure 62. Código actualizacion periodica en Arduino

3.1.3. Conexión entre base de datos y aplicación móvil

Para realizar la conexión entre la base de datos y la aplicación móvil primero se debe habilitar en el proyecto de Firebase la autentificación mediante correo electrónico y contraseña.

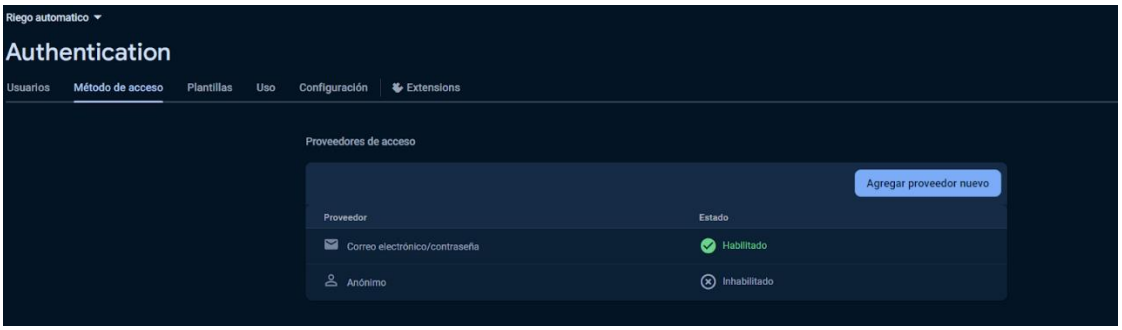


Figure 63. Autenticación en Firebase

Tal y como se ha explicado previamente el email y la contraseña se almacenan en la base de datos, en este caso estamos trabajando con un email ejemplo:

Figure 64. Ventana login generada en Ionic

A continuación, se muestra como efectivamente esa información está almacenada y además podemos ver que se crea un nombre de usuario que será lo que se usará para identificar en la base de datos la información de ese usuario específico.

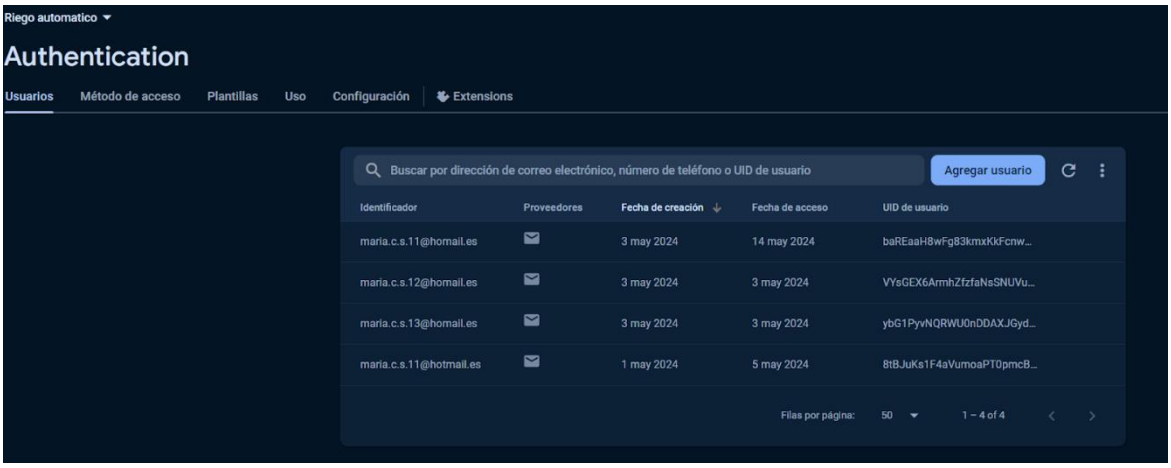


Figure 65. Autenticación de usuarios en Firebase

Dentro de Realtime Database se puede observar cómo dentro del nombre de usuario encontramos la configuración con cada planta creada, las notificaciones y los valores obtenidos en ese momento por los sensores.

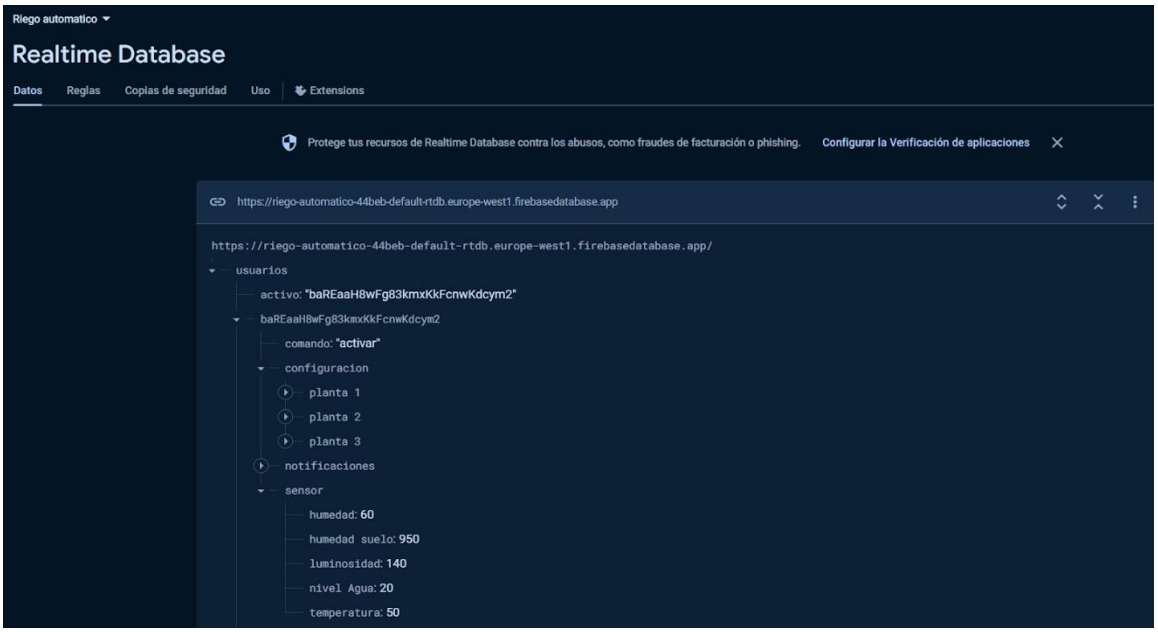


Figure 66. Realtime Database Firebase

Pero para que la conexión se pueda establecer se debe poner toda la información del proyecto de Firebase en .environments.ts

```
export const environment = {
  production: false,
  firebaseConfig : {
    apiKey: "AIzaSyAMHhj6nOhwPjadh2c2wrsl8_wDgo1i
    authDomain: "riego-automatico-44beb.firebaseapp.com",
    databaseURL: "https://riego-automatico-44beb-default-rtdb.europe-west1.firebasedatabase.app",
    projectId: "riego-automatico-44beb",
    storageBucket: "riego-automatico-44beb.appspot.com",
    messagingSenderId: "1034550936598",
    appId: "1:1034550936598:web:ac7e2cd7d833ce47d1a137",
    measurementId: "G-NC8DG24G0P"
  }
};
```

Figure 67. Código environments.ts en Ionic

3.2. Prototipo

Para el diseño final donde se integran los sensores conectados al Arduino, a la base de datos y esta a la aplicación móvil además se le ha añadido al diseño una maqueta de una casa impresa en 3D como se puede observar en la siguiente figura donde se incluirán todos los componentes físicos del proyecto para que esta manera se puedan colocar al lado de las plantas desde las que tomara las medidas.

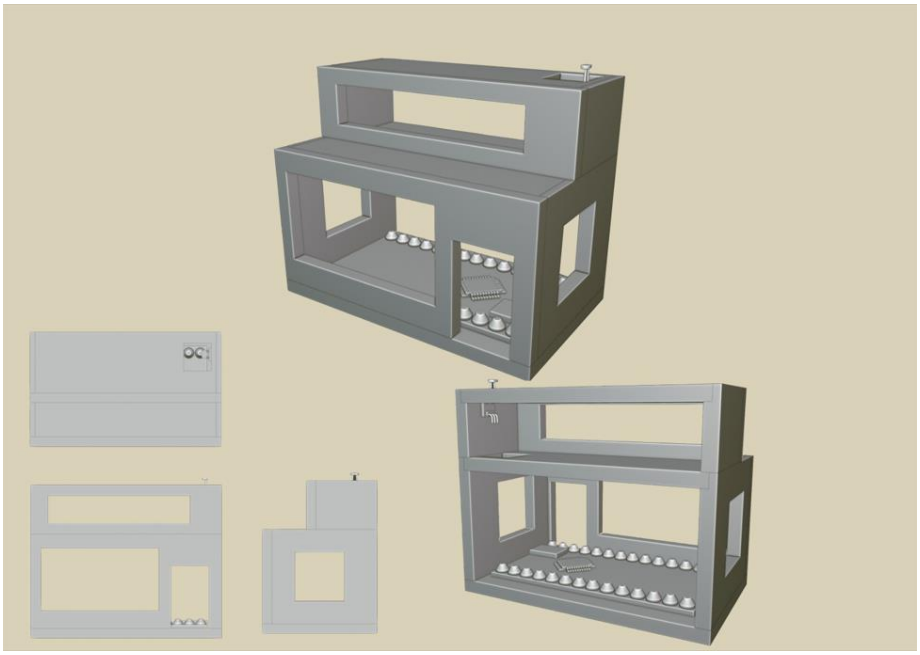


Figure 68. Diseño maqueta casa 3D

El conjunto del Arduino con los sensores y la bomba de agua queda de la siguiente manera:

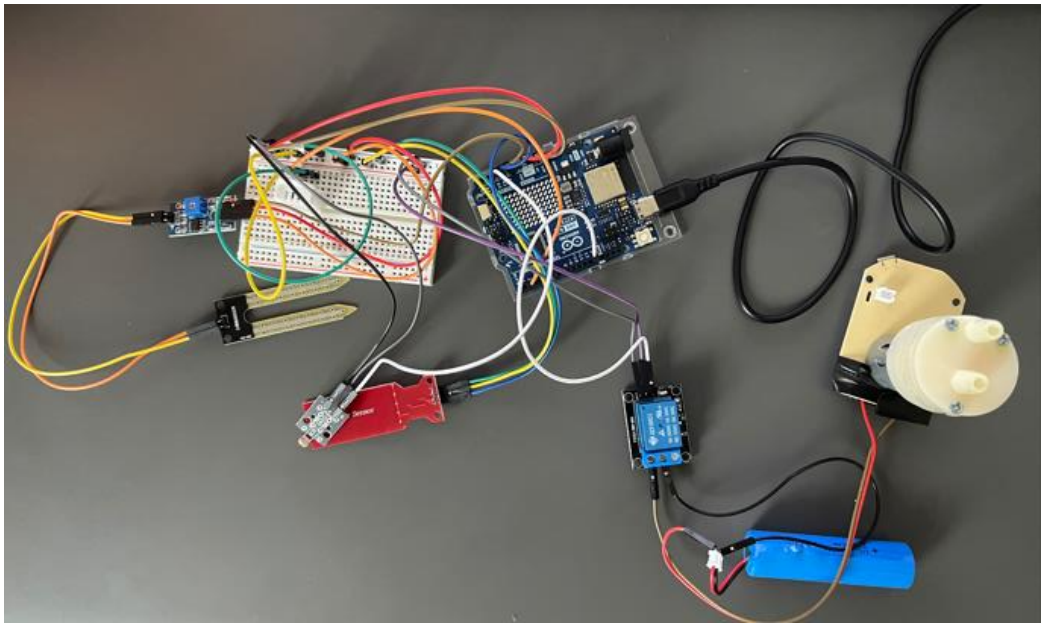


Figure 69. Conjunto Arduino, sensores y bomba de agua

Tras insertarlo en la maqueta el resultado final sería:

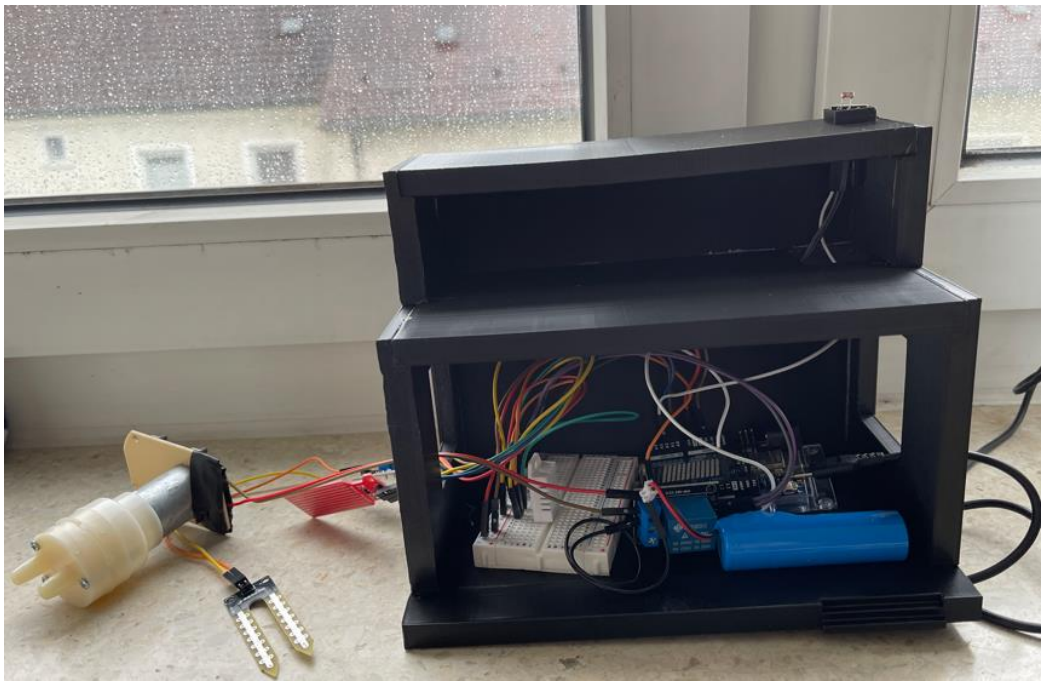


Figure 70. Arduino dentro de maqueta 3D

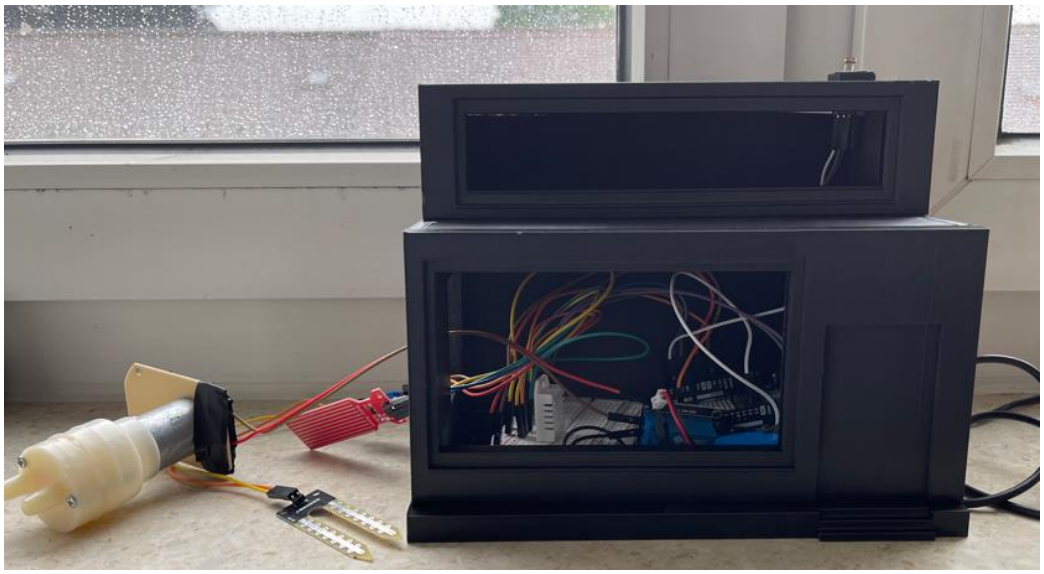


Figure 71. Arduino dentro de maqueta 3D con parte frontal incluida

Como se ha incluido un sensor de luminosidad era importante que estuviera fuera de la maqueta, por ello se diseñó con una chimenea por la que pudiera sobresalir.

Figure 72. Sensor de luminosidad en maqueta 3D

Una vez que toda la parte física esta completada se puede comprobar que cuando los sensores indica que están fuera de los rangos establecidos en la aplicación móvil se verán reflejadas las notificaciones informando al usuario al igual que en la base de datos.

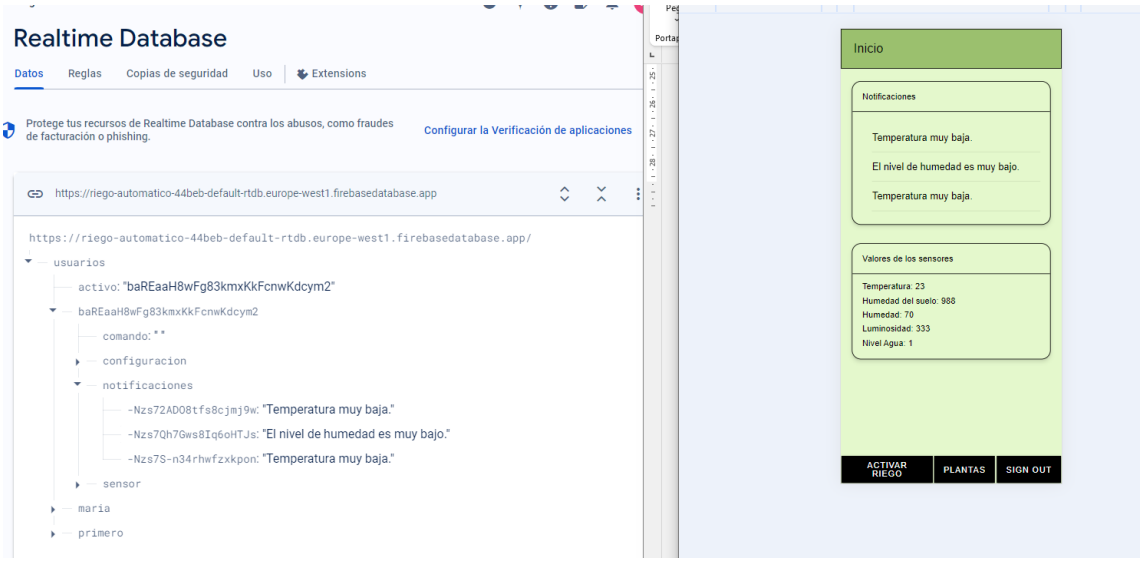


Figure 73. Notificaciones base de datos y aplicación móvil

Cuando en la aplicación móvil se pulsa el botón “activar riego” se observa como en la base de datos se cambia “comando” que antes estaba vacío a “activar” que indicara al Arduino que debe activar la válvula de agua.

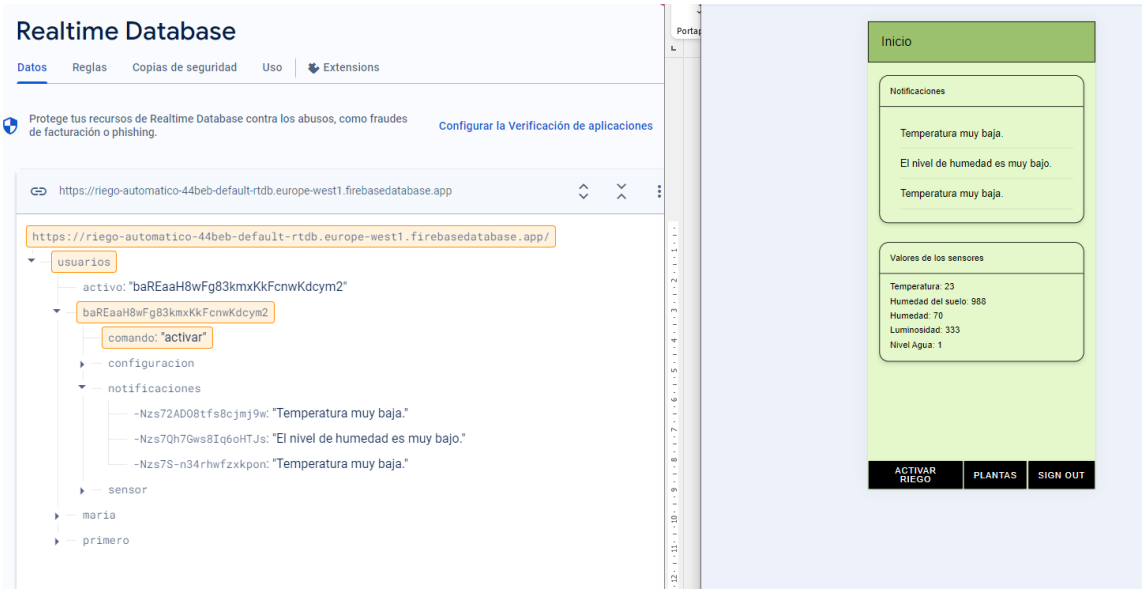


Figure 74. Activación riego en base de datos y aplicación móvil

Tras el tiempo establecido “comando” volverá a estar vacío y se observa que los datos insertados en la aplicación móvil, en este caso sobre “planta 1” se ven reflejados en la base de datos.

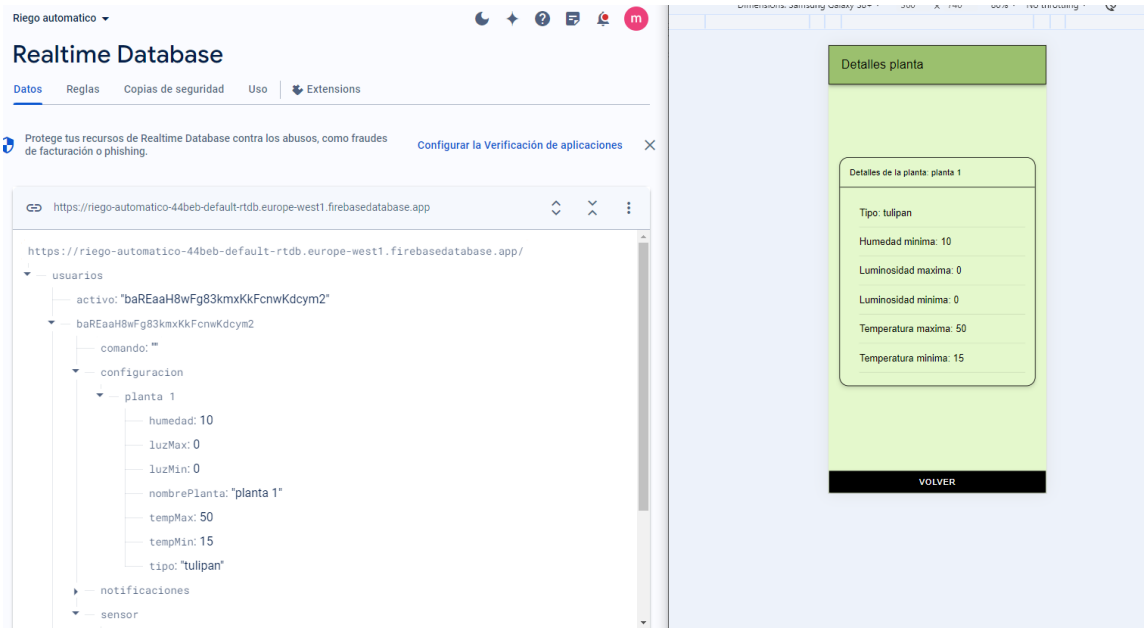


Figure 75. Datos plantas base de datos y aplicación móvil

Al igual que se verán los datos recogidos por los sensores tanto en la base de datos como en la aplicación móvil.

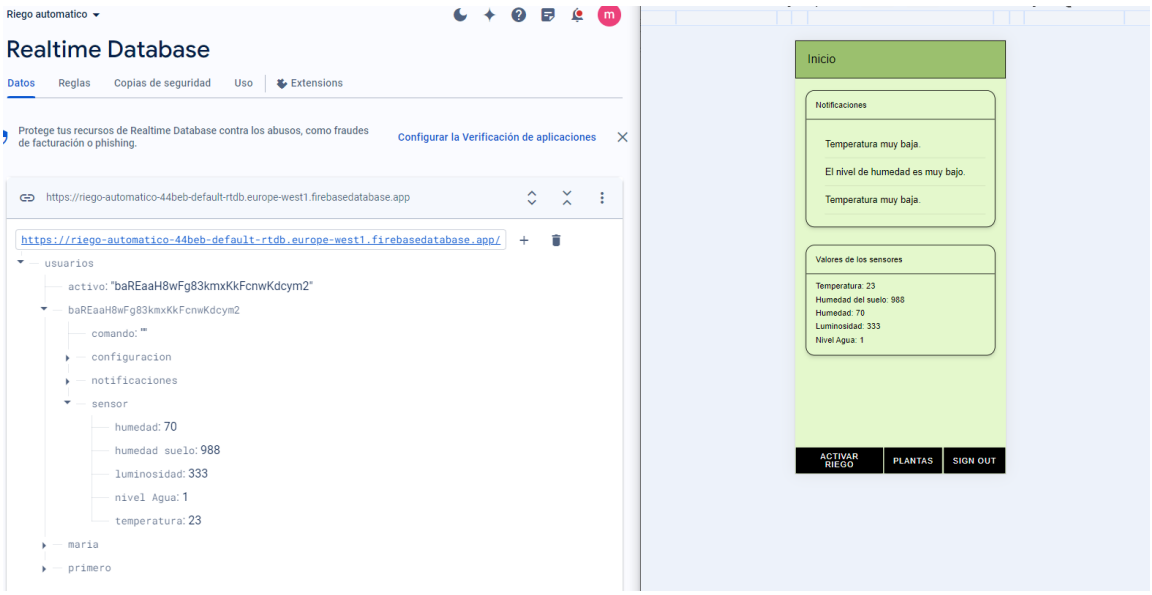


Figure 76. Datos sensores base de datos y aplicación móvil

4. Conclusiones

En este proyecto se ha diseñado e implementado un sistema de riego automático mediante un conjunto de sensores que registraban las condiciones ambientales junto con un Arduino que analizaba dichos datos y determinaba si era necesario regar las plantas o si la luz era excesiva o en caso contrario insuficiente al igual que indica si es necesario reponer el agua usada para realizar el riego.

Toda esta información es enviada y almacenada en una base de datos que está conectada a una aplicación móvil mediante la cual el usuario puede activar el riego, recibir las notificaciones, ver los datos recogidos por los sensores y crear múltiples plantas con la información específica de cada una de ellas para que se pueda comparar de manera independiente cada una de ellas con los valores de los sensores, ya que las necesidades de humedad o luz pueden ser diferente dependiendo del tipo de planta que el usuario haya incorporado en la aplicación móvil.

Las simulaciones realizadas con las diferentes partes del proyecto funcionando de manera simultánea muestra que se actualizan los datos de manera satisfactoria tanto en la base de datos como en la aplicación móvil en función de las modificaciones que se realicen en la aplicación o el cambio de los valores recogidos por los sensores. Es por esto por lo que este proyecto se podrá emplear de manera satisfactoria para regar de manera automática las plantas del hogar.

Además, este sistema permite optimizar el uso del agua para regar las plantas, ya que solo se activará en caso de que sea necesario y también logrará que las plantas tengan una mayor calidad y menos probabilidades de que se sequen o estén débiles, ya que se individualiza para cada planta según sus condiciones ideales.

Al ofrecer al usuario notificaciones, permite que el usuario siempre este informado de las condiciones climáticas que sufren sus plantas permitiendo de esta manera que intervenga si es necesario.

En resumen, este proyecto permite combinar la tecnología actual con la jardinería para que se mejore la experiencia del usuario y se realicen las tareas de jardinería de manera más eficiente e independiente.

5. Trabajos futuros

En cuanto a mejoras o incorporaciones que se podría realizar en un futuro en función de este proyecto seria la posibilidad de tener una mayor zona de riego, esto se podría conseguir incorporando un mayor número de relés que se podría separar dependiendo de las necesidades de riego del tipo de plantas que tenga el usuario. Aumentando el número de relés se podría optimizar el riego para distintas zonas o tipos de plantas, para que, de esta manera reciban la cantidad adecuada de agua acorde con sus requerimientos.

Debido a la limitación de tiempo las notificaciones que recibe el usuario desde la base de datos se han creado de manera que aparecen dentro de la propia aplicación móvil, pero una mejora podría ser hacer uso de las notificaciones push que hacen que el usuario, aunque no tenga abierta la aplicación pueda ver en su pantalla de inicio la notificación mediante un mensaje emergente.

Como se trataba de un prototipo solo se ha hecho uso de un sensor de cada tipo, pero en el futuro se podrían incorporar un mayor número de sensores de humedad para que cada planta cuente con el propio y esta variable se almacenaría junto con la información individual de cada planta seleccionada en la base de datos. De esta manera la monitorización seria más precisa para cada una de las plantas, facilitando el control de sus necesidades de manera más individualizada.

Por último, debido a que el Arduino usado es relativamente nuevo aun no cuenta con todas las librerías como otros modelos más antiguos, por lo que aún no es posible usar la librería “wifimanager” para pedir directamente al usuario que indique sus datos wifi. Sin embargo, es algo que se podría implementar conforme las librerías se actualicen, esto facilitará la conectividad del dispositivo del usuario con la red wifi y el Arduino.

6. Glosario

Angular: Framewrok Javacript de desarrollo de aplicaciones
Arduino: Plataforma para diseno de prototipo electronico
GND: Ground (tierra, 0V)
HTML: hypertext markup language
IDE: Integrated Development Environment
Ionic: Herramienta para el desarrollo de aplicaciones hibridas
JSON: JavaScript Object Notation
RX: Recepcion
TX: Transmision

7. Bibliografía

1. Bustos Lafuente, A. (2019). "Diseño y control domótico de un sistema de riego automático para un huerto urbano en el hogar." Universidad Oberta de Catalunya.

2. Ravi Prakash, B. et al. (2020). “Super Smart Irrigation System using Internet of Things “

3. Sushanth, G. et al. (2018). “IOT Based Smart Agriculture System”

4. Irawan, B. et al. (2017). “Controlling and monitoring ornamental plants care remotely using android application.”

5. Dhawale. (2019). "Review on IoT Based Smart Agriculture System."

6. Qiu, G.-h., et al. (2020). "Research on the intelligent agricultural closed-loop system under the Internet of Things Architecture."

7. Castillo Melgar, C. (2021). “Diseño de un sistema de riego automatizado para cultivos de ciclo corto con Arduino. Estudio de caso pimiento.”

8. Gutierrez Jagüey, J. et al. (2015). “Smartphone Irrigation Sensor”

9. Li. (2020). "Research on Precision Planting Management System Based Agricultural Data"

10. Işık, M. F., et al. (2015). "Precision Irrigation System (PIS) Using Sensor Network Technology Integrated with IOS/Android Application."

11. Guerrero-Ulloa, G., et al. (2023). "Internet of Things (IoT)-based indoor plant care system."

12. Kim, T., et al. (2013). "An Approach for a Self-Growing Agricultural Knowledge Cloud in Smart Agriculture."

13. Arduino® UNO R4 WiFi datasheet. Arduino (2024) [online] Disponible: ABX00087-datasheet.pdf (arduino.cc)

14. DHT22 Datasheet. (2021) [online] Disponible: Digital+humidity+and+temperature+sensor+AM2302.pdf (adafruit.com)

15. Soil Moisture Sensor Datasheet. (2020) [online] Disponible: Soil Moisture Sensor Module Pinout, Features, Specs & Circuit (components101.com)

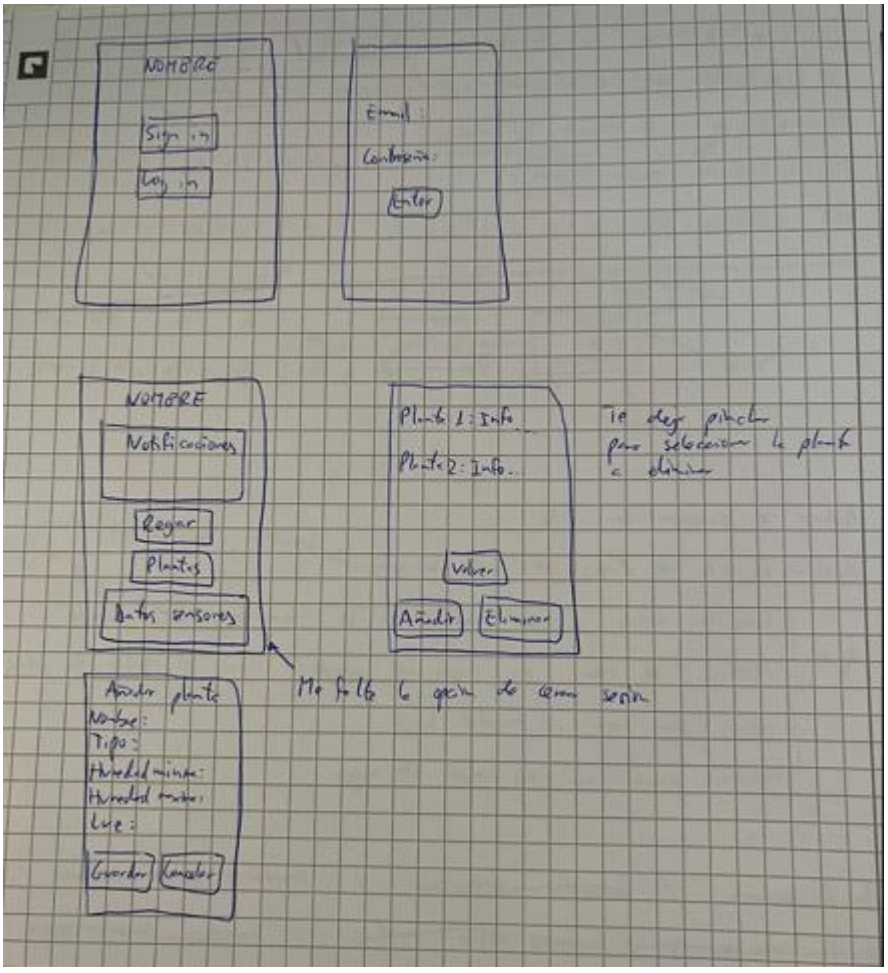
16. KY-018 Sensor Datasheet. (2017) [online] Disponible: KY-018-Joy-IT.pdf (datasheethub.com)

17. Water Level Sensor Datasheet. (2023) [online] Disponible: Arduino Water Level Sensor - Datasheet Hub

18. Rele Datasheet. (2021) [online] Disponible:COM-KY019RM_DATASHEET_2021-04-06.pdf (cdn-reichelt.de)
19. Fritzing programa [online] Disponible: Welcome to Fritzing

8. Anexos

8.1. Boceto aplicación móvil



8.2. Código Ionic “landing”

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-landing',
  templateUrl: './landing.page.html',
  styleUrls: ['./landing.page.scss'],
})
export class LandingPage implements OnInit {
```

```
  constructor() { }
  ngOnInit() {
  }
}
```

8.3. Código Ionic “login”

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AuthServiceService } from 'src/app/auth-service.service';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import { AlertController, LoadingController } from '@ionic/angular';
import { ToastController } from '@ionic/angular';
import { AngularFireDatabase} from '@angular/fire/compat/database';

@Component({
  selector: 'app-login',
  templateUrl: './login.page.html',
  styleUrls: ['./login.page.scss'],
})
export class LoginPage implements OnInit {
  loginForm: FormGroup;

  constructor(private data:AngularFireDatabase,private toastController: ToastController,
private alertController: AlertController, private loadingController: LoadingController, private
authService: AuthServiceService, private router: Router, public formBuilder: FormBuilder) {
}
  ngOnInit() {
    this.loginForm = this.formBuilder.group({
      email: [
        '',
        [
          Validators.required,
          Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+.[a-z]{2,3}$'),
        ],
      ],
      password: ['', [
        Validators.pattern('(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[$@$!%*?&])[A-Za-
z\d$@$!%*?&].{8,}'),
        Validators.required,
      ]
      ],
    });
```

```
  }
  async login() {
    const loading = await this.loadingController.create();
    await loading.present();
    if (this.loginForm.valid) {

      const userCred = await this.authService.loginUser(this.loginForm.value.email,
this.loginForm.value.password).catch((err) => {
        this.presentToast(err)
        console.log(err);
        loading.dismiss();
      })

      if (userCred) {
        loading.dismiss();
        const userId = userCred.user?.uid;
        if(userId){this.data.database.ref(`usuarios/activo`).set(userId);}
        this.router.navigate(
          ['/inicio'])
      }
    } else {
      return console.log('Please provide all the required values!');
    }

  }
  get errorControl() {
    return this.loginForm?.controls;
  }

  async presentToast(message: undefined) {
    console.log(message);

    const toast = await this.toastController.create({
      message: message,
      duration: 1500,
      position: 'top',
    });

    await toast.present();
  }
}
```

8.4. Código Ionic “inicio”

```
import { Component, OnInit } from '@angular/core';
import { AuthServiceService } from 'src/app/auth-service.service';
import { Router,ActivatedRoute  } from '@angular/router';
import { User } from 'firebase/auth';
import { AngularFireDatabase} from '@angular/fire/compat/database';

@Component({
  selector: 'app-inicio',
  templateUrl: './inicio.page.html',
  styleUrls: ['./inicio.page.scss'],
})
export class InicioPage implements OnInit{
  email :any;
  nombre:any;
  temperatura: any;
  humedad: any;
  humedadSuelo: any;
  luminosidad: any;
  nivelAgua: any;
  notificaciones: any[] =[];
  nombreNotificaciones: any;

  constructor(private data:AngularFireDatabase,private
authService:AuthServiceService,private router: Router,private route: ActivatedRoute) {

  }
  ngOnInit(): void {

    this.authService.getProfile().then(user => {
      this.email = user?.email;
      this.nombre = user?.uid;
      console.log(user?.email);
      this.obtenerValoresSensores();
      this.verNotificaciones();
    }).catch(error => {
      console.error('Error getting user profile:', error);
    });
  }
  activarRiego(){
    this.data.database.ref(`usuarios/`+this.nombre+'/comando').set("activar").then(() => {
      console.log('Riego activado.');
```

```
    }).catch((error) => {
      console.error('Error al activar el riego: ', error);
    });

  }
 signOut(){
  this.authService.signOut().then(() =>{
    this.router.navigate(['/landing'])
  })
 }
 irAPlantas(){

      this.router.navigate(['/plantas'])

 }
 obtenerValoresSensores(){

this.temperatura=this.data.object('usuarios/'+this.nombre+'/sensor/temperatura').valueChan
ges().subscribe((valor)=>{
    console.log("Temperatura:", valor);
    this.temperatura = valor;
  });

this.humedad=this.data.object('usuarios/'+this.nombre+'/sensor/humedad').valueChanges().
subscribe((valor)=>{
    console.log("humedad:", valor);
    this.humedad = valor;
  });

  this.humedadSuelo=this.data.object('usuarios/'+this.nombre+'/sensor/humedad
suelo').valueChanges().subscribe((valor)=>{
    console.log("humedadSuelo:", valor);
    this.humedadSuelo = valor;
  });

this.luminosidad=this.data.object('usuarios/'+this.nombre+'/sensor/luminosidad').valueChan
ges().subscribe((valor)=>{
    console.log("luminosidad:", valor);
    this.luminosidad = valor;
  });

  this.nivelAgua=this.data.object('usuarios/'+this.nombre+'/sensor/nivel
Agua').valueChanges().subscribe((valor)=>{
    console.log("nivelAgua:", valor);
```

```
    this.nivelAgua = valor;
  });
  }

  verNotificaciones() {
    this.data.object('usuarios/' + this.nombre +
'/notificaciones').valueChanges().subscribe((notificaciones) => {
      if (notificaciones) {
        this.notificaciones = Object.keys(notificaciones).map(key => ({
          id: key,
          contenido: notificaciones[key]
        }));
      }
    });
  }

  delete(id: String){

    this.data.database.ref('usuarios/'+this.nombre+ '/notificaciones/' + id).remove()
    .then(() => {
        console.log('Notificacion eliminada correctamente.');
      this.router.navigate(['/inicio'])
    })
    .catch((error) => {
      console.error('Error al eliminar la notificacion: ', error);
    });

  }

}
```

8.5. Código Ionic “reset password”

```
import { Component, OnInit } from '@angular/core';
import { AuthServiceService } from 'src/app/auth-service.service';
import { ToastController } from '@ionic/angular';
import { Router } from '@angular/router';

@Component({
  selector: 'app-reset-password',
  templateUrl: './reset-password.page.html',
  styleUrls: ['./reset-password.page.scss'],
})
```

```
export class ResetPasswordPage implements OnInit {
email:any
  constructor(private authService:AuthServiceService,private toastController:
ToastController,private router: Router) { }

  ngOnInit() {
  }

  reset(){
    this.authService.resetPassword(this.email).then( () =>{
      console.log('sent'); //show confirmation dialog
      this.presentToast()
    })
  }
  async presentToast() {
    const toast = await this.toastController.create({
      message: 'Your reset password link has been sent on your email',
      duration: 2000, // Duration in milliseconds
      position: 'bottom' // Position of the toast (top, bottom, middle)
    });

    toast.present();
    toast.onDidDismiss().then(()=>{
      this.router.navigate(['/login']);
    })
  }
}
```

8.6. Código Ionic “signup”

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AuthServiceService } from 'src/app/auth-service.service';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import { AlertController, LoadingController } from '@ionic/angular';
import { ToastController } from '@ionic/angular';
import { AngularFireDatabase} from '@angular/fire/compat/database';

@Component({
  selector: 'app-signup',
  templateUrl: './signup.page.html',
  styleUrls: ['./signup.page.scss'],
})
```

```
export class SignupPage implements OnInit {
  signForm: FormGroup;

  constructor(private data:AngularFireDatabase, private toastController:
ToastController,private loadingController: LoadingController,private
authService:AuthServiceService,private router: Router, public formBuilder: FormBuilder) {

  }

  ngOnInit() {
    this.signForm = this.formBuilder.group({
      email: [
        '',
        [
          Validators.required,
          Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+.[a-z]{2,3}$'),
        ],
      ],
      password: ['', [
        Validators.pattern('(?=.*[a-z])(?=.*[A-Z])(?=.*[0-8])(?=.*[$@$!%*?&])[A-Za-
z\d$@$!%*?&].{8,}'),
        Validators.required,
      ],
    ],
    });
  }
  get errorControl() {
    return this.signForm?.controls;
  }

  async signUP(){
    const loading = await this.loadingController.create();
    await loading.present();
    if (this.signForm.valid) {

      const user = await this.authService.registerUser(this.signForm.value.email,
this.signForm.value.password).catch((err) => {
        this.presentToast(err)
        console.log(err);
        loading.dismiss();
      })

      if (user) {
        loading.dismiss();
```

```
        const userId = user.user?.uid;
        if(userId){this.data.database.ref(`usuarios/activo`).set(userId);}
        this.router.navigate(['/inicio'])
      }
    } else {
      return console.log('Please provide all the required values!');
    }
  }

  async presentToast(message: undefined) {
    console.log(message);

    const toast = await this.toastController.create({
      message: message,
      duration: 1500,
      position: 'top',
    });

    await toast.present();
  }
}
```

8.7. Código Ionic “plantas”

```
import { Component, OnInit } from '@angular/core';
import { AuthServiceService } from 'src/app/auth-service.service';
import { Router,ActivatedRoute  } from '@angular/router';
import { User } from 'firebase/auth';
import { AngularFireDatabase} from '@angular/fire/compat/database';
import { BehaviorSubject, Observable } from 'rxjs';

@Component({
  selector: 'app-plantas',
  templateUrl: './plantas.page.html',
  styleUrls: ['./plantas.page.scss'],
})
export class PlantasPage implements OnInit {

  plantas: BehaviorSubject<string[]> = new BehaviorSubject<string[]>([]);
  email:any;
  nombre:any;
  constructor(private data:AngularFireDatabase,private
authService:AuthServiceService,private router: Router,private route: ActivatedRoute) { }
```

```
  ngOnInit() {

    this.route.params.subscribe(params => {
      const nombrePlanta = params['nombrePlanta'];
      console.log(nombrePlanta);
      if (nombrePlanta) {
        this.plantas.next([...this.plantas.value, nombrePlanta]);
      }
    });
    this.authService.getProfile().then(user => {
      this.email = user?.email;
      this.nombre = user?.uid;
      console.log(user?.email);
      this.obtenerPlantas();
    }).catch(error => {
      console.error('Error getting user profile:', error);
    });
  }
  volverAInicio(){
    this.router.navigate(['/inicio'])
  }
  AnadirPlanta(){
    this.router.navigate(['/nueva-planta'])
  }
  edit(nombrePlanta: string){
    this.router.navigate(['/editar-planta', { nombrePlanta: nombrePlanta }]);
    console.log(nombrePlanta);
  }
  delete(nombrePlanta: string){

    this.data.database.ref('usuarios/'+this.nombre+ '/configuracion/' +
nombrePlanta).remove()
    .then(() => {
       console.log('Planta eliminada correctamente.');
      this.router.navigate(['/plantas'])
    })
    .catch((error) => {
      console.error('Error al eliminar la planta: ', error);
    });

  }
  details(nombrePlanta: string){
    this.router.navigate(['/detalles-planta', { nombrePlanta: nombrePlanta }]);
```

```
    console.log(nombrePlanta);

  }

  obtenerPlantas(){


this.data.list(`usuarios/${this.nombre}/configuracion`).snapshotChanges().subscribe(snapsh
ot => {
      const nombresPlantas = snapshot.map(item => item.key);
this.plantas.next(nombresPlantas);
    });

  }
}
```

8.8. Código Ionic “nueva planta”

```
import { Component, OnInit } from '@angular/core';
import { AuthServiceService } from 'src/app/auth-service.service';
import { Router,ActivatedRoute  } from '@angular/router';
import { User } from 'firebase/auth';
import { AngularFireDatabase} from '@angular/fire/compat/database';

@Component({
  selector: 'app-nueva-planta',
  templateUrl: './nueva-planta.page.html',
  styleUrls: ['./nueva-planta.page.scss'],
})
export class NuevaPlantaPage implements OnInit {
  email:any;
  nombre:any;
  nombrePlanta :any;
  tipoPlanta:any;
  humedad:any;
  luzMax:any;
  luzMin:any;
  tempMax:any;
  tempMin:any;
  constructor(private data:AngularFireDatabase,private
authService:AuthServiceService,private router: Router,private route: ActivatedRoute) { }

  ngOnInit() {
```

```
  this.authService.getProfile().then(user => {
    this.email = user?.email;
    this.nombre = user?.uid;
    console.log(user?.email);
  }).catch(error => {
    console.error('Error getting user profile:', error);
  });
 }

 guardarPlanta(){
  this.data.database.ref(`usuarios/`+this.nombre+'/configuracion/'+this.nombrePlanta).set({
    tipo: this.tipoPlanta,
    humedad: this.humedad,
    luzMax: this.luzMax,
    tempMax: this.tempMax,
    luzMin: this.luzMin,
    tempMin: this.tempMin,
  }).then(() => {
    console.log('Planta guardada exitosamente.');
    this.router.navigate(['/plantas', { nombrePlanta: this.nombrePlanta }])
  }).catch((error) => {
    console.error('Error al guardar la planta: ', error);
  });

 }

 cancelar(){
  this.router.navigate(['/plantas'])
 }
}
```

8.9. Código Ionic “editar planta”

```
import { Component, OnInit } from '@angular/core';
import { AuthServiceService } from 'src/app/auth-service.service';
import { Router,ActivatedRoute  } from '@angular/router';
import { User } from 'firebase/auth';
import { AngularFireDatabase} from '@angular/fire/compat/database';

@Component({
  selector: 'app-editar-planta',
  templateUrl: './editar-planta.page.html',
  styleUrls: ['./editar-planta.page.scss'],
```

```
})
export class EditarPlantaPage implements OnInit {
  email:any;
  nombre:any;
  nombrePlanta :any;
  tipoPlanta:any;
  humedad:any;
  luzMax:any;
  luzMin:any;
  tempMax:any;
  tempMin:any;
  detallesPlanta: any;
  nombrePlantaOriginal:any;
  constructor(private data:AngularFireDatabase,private
authService:AuthServiceService,private router: Router,private route: ActivatedRoute) { }

  ngOnInit() {
    this.authService.getProfile().then(user => {
      this.email = user?.email;
      this.nombre = user?.uid;
      console.log(user?.email);
      this.route.params.subscribe(params => {
        this.nombrePlanta = params['nombrePlanta'];
        this.nombrePlantaOriginal = this.nombrePlanta;
        this.obtenerDetallesPlanta(this.nombrePlanta);
      });
    }).catch(error => {
      console.error('Error getting user profile:', error);
    });
  }
  guardarPlanta(){

this.data.database.ref(`usuarios/`+this.nombre+'/configuracion/'+this.nombrePlanta).update
({
      tipo: this.tipoPlanta,
      humedad: this.humedad,
      luzMax: this.luzMax,
      tempMax: this.tempMax,
      luzMin: this.luzMin,
      tempMin: this.tempMin,
    }).then(() => {
      console.log('Planta guardada exitosamente.');

      if (this.nombrePlanta !== this.nombrePlantaOriginal) {
```

```
        console.log('Nuevo nombre de la planta:', this.nombrePlanta);

this.data.database.ref(`usuarios/`+this.nombre+'/configuracion/'+this.nombrePlantaOriginal)
.remove();
      }
      this.router.navigate(['/plantas', { nombrePlanta: this.nombrePlanta }])
    }).catch((error) => {
      console.error('Error al guardar la planta: ', error);
    });

  }

  cancelar(){
    this.router.navigate(['/plantas'])
  }

  obtenerDetallesPlanta(nombrePlanta: string) {

this.data.object('usuarios/'+this.nombre+'/configuracion/'+nombrePlanta).valueChanges().s
ubscribe(valor=>{
      this.tipoPlanta = valor['tipo'];
      this.humedad = valor['humedad'];
      this.luzMax = valor['luzMax'];
      this.tempMax = valor['tempMax'];
      this.luzMin = valor['luzMin'];
      this.tempMin = valor['tempMin'];
      console.log(valor);
    });
}
}
```

8.10. Código Ionic “detalles planta”

```
import { Component, OnInit } from '@angular/core';
import { AuthServiceService } from 'src/app/auth-service.service';
import { Router,ActivatedRoute  } from '@angular/router';
import { User } from 'firebase/auth';
import { AngularFireDatabase} from '@angular/fire/compat/database';

@Component({
  selector: 'app-detalles-planta',
  templateUrl: './detalles-planta.page.html',
  styleUrls: ['./detalles-planta.page.scss'],
```

```
})
export class DetallesPlantaPage implements OnInit {

  detallesPlanta: any;
 nombre:any;
 email:any;
 nombrePlanta: string;
  constructor(private data:AngularFireDatabase,private
authService:AuthServiceService,private router: Router,private route: ActivatedRoute) { }

  ngOnInit() {

    this.authService.getProfile().then(user => {
      this.email = user?.email;
      this.nombre = user?.uid;
      console.log(user?.email);
      this.route.params.subscribe(params => {
        this.nombrePlanta = params['nombrePlanta'];
        this.obtenerDetallesPlanta(this.nombrePlanta);
      });
    }).catch(error => {
      console.error('Error getting user profile:', error);
    });
  }
  obtenerDetallesPlanta(nombrePlanta: string) {

this.data.object('usuarios/'+this.nombre+'/configuracion/'+nombrePlanta).valueChanges().s
ubscribe(valor=>{
       this.detallesPlanta = valor;
      console.log(valor);

    });
  }
  volver(){
    this.router.navigate(['/plantas'])

  }
}
```

8.11. Código Arduino

```
#include <Arduino.h>
#if defined(ESP32) || defined(ARDUINO_RASPBERRY_PI_PICO_W)
#include <WiFi.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#elif __has_include(<WiFiNINA.h>)
#include <WiFiNINA.h>
#elif __has_include(<WiFi101.h>)
#include <WiFi101.h>
#elif __has_include(<WiFiS3.h>)
#include <WiFiS3.h>
#endif

#include <Firebase_ESP_Client.h>
#include <DHT.h>
//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "Pinguine zu verkaufen"
#define WIFI_PASSWORD "Proteincookiebananenmilch42069"
#define USER_EMAIL "maria.c.s.11@hotmail.es"
#define USER_PASSWORD "Maria1998@1"
// Insert Firebase project API Key
#define API_KEY "AIzaSyAMHhj6nOhwPjadh2c2wrsl8_wDgo1icX8"

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://riego-automatico-44beb-default-rtdb.europe-
west1.firebasedatabase.app/"

//Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;
unsigned long lastConfigCheckTime = 0;
const unsigned long CONFIG_CHECK_INTERVAL = 6000;
bool signupOK = false;
```

```
int SENSOR = 2;
int humedad_suelo = A0; //mido la humedad al analogico 0
int nivel_agua = A1;
int luz = A2;
int temp, humedad, humedad_suelo_Value, AguaValue, lux;
DHT dht (SENSOR, DHT22);
int bomba_PIN = 13; //Pin 13 para la bomba
int humedadSueloMinima, luxMinimo,luxMaximo, tempMinima, tempMaxima;//poner en
firebase como configuracion: igual que comando: activar riego
const int LUX_UMBRAL = 500; // Define el umbral de luminosidad para determinar si es de
día o de noche
const int HORA_AMANECER = 6; // Hora de amanecer
const int HORA_ATARDECER = 18; // Hora de atardecer
int nivelAguaMinimo = 50;
String limpiar ="";
String direccion = "usuarios/";
std::vector<String> nombresPlantas;
void setup(){
  Serial.begin(9600);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  /* Assign the api key (required) */
  config.api_key = API_KEY;
  //auth.user.email = USER_EMAIL;
  //auth.user.password = USER_PASSWORD;
  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;

  /* Sign up */
  if (Firebase.signUp(&config, &auth, "", "")){
    Serial.println("ok");
    signupOK = true;
  }
```

```
  /* Assign the callback function for the long running token generation task */
  config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  if (Firebase.RTDB.getString(&fbdo, "usuarios/activo")) {
       direccion = "usuarios/" + fbdo.stringData();

  }
  dht.begin();
  pinMode(humedad_suelo, INPUT);
  pinMode(nivel_agua, INPUT);
  pinMode(luz, INPUT);

  obtenerConfiguracionDesdeFirebase();
  notificaciones();
}

void loop(){

   delay(2000);

   humedad = dht.readHumidity();
   temp = dht.readTemperature();
   humedad_suelo_Value = analogRead(humedad_suelo); //take a sample
   AguaValue = analogRead(nivel_agua); //take a sample
   lux = analogRead(luz);

   Serial.print("Temperatura: ");
   Serial.println(temp);
   Serial.print("Humedad: ");
   Serial.println(humedad);
   Serial.print("Humedad suelo: ");
   Serial.println(humedad_suelo_Value);
   Serial.print("Nivel Agua: ");
   Serial.println(AguaValue);
   Serial.print("Luminosidad: ");
   Serial.println(lux);
   if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 15000 ||
sendDataPrevMillis == 0)){
     sendDataPrevMillis = millis();

   // Write an Int number on the database path test/int
```

```
  if (Firebase.RTDB.setInt(&fbdo, direccion + "/sensor/temperatura", temp)){
    Serial.println("PASSED");

  }
  else {
    Serial.println("FAILED Temperatura");
    Serial.println("REASON: " + fbdo.errorReason());
  }


  // Write an Int number on the database path test/int
  if (Firebase.RTDB.setInt(&fbdo, direccion + "/sensor/humedad", humedad)){
    Serial.println("PASSED");

  }
  else {
    Serial.println("FAILED Humedad");
    Serial.println("REASON: " + fbdo.errorReason());
  }

  // Write an Int number on the database path test/int
  if (Firebase.RTDB.setInt(&fbdo, direccion + "/sensor/humedad suelo",
humedad_suelo_Value)){
    Serial.println("PASSED");

  }
  else {
    Serial.println("FAILED Humedad suelo");
    Serial.println("REASON: " + fbdo.errorReason());
  }

  // Write an Int number on the database path test/int
  if (Firebase.RTDB.setInt(&fbdo, direccion + "/sensor/nivel Agua", AguaValue)){
    Serial.println("PASSED");

  }
  else {
    Serial.println("FAILED Nivel Agua");
    Serial.println("REASON: " + fbdo.errorReason());
  }

  //Write an Int number on the database path test/int
  if (Firebase.RTDB.setInt(&fbdo, direccion + "/sensor/luminosidad", lux)){
    Serial.println("PASSED");
```

```
  }
   else {
     Serial.println("FAILED Luminosidad");
     Serial.println("REASON: " + fbdo.errorReason());
   }
}
  if (Firebase.RTDB.getString(&fbdo, direccion + "/comando")) {
       String comando = fbdo.stringData();
       if (comando == "activar") {
          // Ejecutar acción correspondiente en Arduino
          abrir_valvula();
          Firebase.RTDB.setString(&fbdo, direccion + "/comando", " ");
       }

   }

     // Verificar y actualizar configuración periódicamente
  if (millis() - lastConfigCheckTime >= CONFIG_CHECK_INTERVAL) {
   obtenerConfiguracionDesdeFirebase();
   notificaciones();
   lastConfigCheckTime = millis(); // Actualizar tiempo de la última verificación

  }

}

void abrir_valvula () {
   digitalWrite(bomba_PIN, HIGH);
   delay(10000);
   digitalWrite(bomba_PIN, LOW);
}

void obtenerConfiguracionDesdeFirebase() {
    if (Firebase.RTDB.getString(&fbdo, direccion +"/configuracion")) {
       if (fbdo.dataType() == "json") {
          FirebaseJson &json = fbdo.jsonObject();
          for (size_t i = 0; i < json.iteratorBegin(); i++) {
             int type;
             String key, value;
             json.iteratorGet(i, type, key, value);

             if (type == FirebaseJson::JSON_OBJECT && value.indexOf("nombrePlanta") != -
1) {
```

```
                String plantName = key.substring(key.lastIndexOf('/') + 1);
                nombresPlantas.push_back(plantName);
            }
        }
    for (size_t i = 0; i < nombresPlantas.size(); ++i) {

        Serial.println(nombresPlantas[i]);
        Firebase.RTDB.getInt(&fbdo, direccion +
"/configuracion/"+nombresPlantas[i]+"/humedad");
  if (fbdo.dataType() == "int") {
    humedadSueloMinima = fbdo.intData();
  }
  Firebase.RTDB.getInt(&fbdo, direccion + "/configuracion/"+nombresPlantas[i]+"/luzMin");
  if (fbdo.dataType() == "int") {
    luxMinimo = fbdo.intData();
  }
  Firebase.RTDB.getInt(&fbdo, direccion + "/configuracion/"+nombresPlantas[i]+"/luzMax");
  if (fbdo.dataType() == "int") {
    luxMaximo = fbdo.intData();
  }
  Firebase.RTDB.getInt(&fbdo, direccion +
"/configuracion/"+nombresPlantas[i]+"/tempMin");
  if (fbdo.dataType() == "int") {
    tempMinima = fbdo.intData();
  }

  Firebase.RTDB.getInt(&fbdo, direccion +
"/configuracion/"+nombresPlantas[i]+"/tempMax");
  if (fbdo.dataType() == "int") {
    tempMaxima = fbdo.intData();
  }
     }
        } else {
            Serial.println("Error: Data is not a JSON object");
        }
    } else {
        Serial.println("Error: Failed to read data from Firebase");
    }
}

void notificaciones(){
  if (nivel_agua < nivelAguaMinimo ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones", "El nivel de agua es muy
bajo. Pon agua en el recipiente");
```

```
  }
  for (size_t i = 0; i < nombresPlantas.size(); ++i) {
 if (humedad_suelo < humedadSueloMinima + 100 ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+ ": El
nivel de humedad es muy bajo.");
  }
  if (humedad_suelo < humedadSueloMinima ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ":
Nivel de humedad del suelo extremadamente bajo! Activar riego automáticamente.");
  abrir_valvula();
  }
  if (temp < tempMinima ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ":
Temperatura muy baja.");
  }
    if (temp > tempMaxima ) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ":
Temperatura muy alta.");
  }
  if (lux < luxMinimo && esDeDia()) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ":
Necesita mas luz.");
  }
    if (lux > luxMaximo && esDeDia()) {
  Firebase.RTDB.pushString(&fbdo, direccion + "/notificaciones",nombresPlantas[i]+  ":
Necesita menos luz.");
  }
  }
}
bool esDeDia() {
  unsigned long tiempo_actual = millis(); // Obtiene el tiempo actual en milisegundos
  unsigned long tiempo_transcurrido = tiempo_actual / 1000; // Convierte a segundos
  // Convierte el tiempo transcurrido en horas
  int hora_actual = tiempo_transcurrido / 3600;
    // Compara la hora actual con el rango de horas de día
 if (hora_actual >= HORA_AMANECER && hora_actual <= HORA_ATARDECER) {
  //   Si es de día, verifica la luminosidad
    if (lux > LUX_UMBRAL) {
      return true; // Es de día
    }
 }

  return false; // Es de noche
}
```