Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingeniería de Sistemas Informáticos

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

Cabezal de Riego Inteligente

Autor: Luis Miguel Santurino Perales
Tutor: Alberto Cruz Ruiz
Codirector: Carlos Gilarranz Casado

Madrid, Julio, 2024

Este Trabajo de Fin de Grado ha sido depositado en la ETSI de Sistemas Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería de Computadores

Título: Cabezal de Riego Inteligente

Julio, 2024

Autor(a): Luis Miguel Santurino Perales
Tutor(a): Alberto Cruz Ruiz
Codirector(a): Carlos Gilarranz Casado

Departamento de Sistemas Informáticos
ETSI de Sistemas Informáticos
Universidad Politécnica de Madrid

Resumen

Este proyecto introduce nuevas herramientas del mundo del IoT (Internet de las Cosas) con el objetivo de mejorar la producción en los cultivos, gestionando de manera óptima el consumo de agua y la maquinaria que forma parte del sistema. El cabezal de riego inteligente funciona mediante goteo y está compuesto tanto por componentes analógicos como digitales que cumplen la misma función.

El sistema automatizará toda la actividad del riego, utilizando microcontroladores de bajo coste que controlen la actividad de los demás componentes, usando plataformas IoT que gestionen los valores recogidos por los dispositivos. Para llevar a cabo este cometido, se desarrollarán diversas funciones como la implementación de una red MESH que conecte múltiples dispositivos a Internet, la OTA para conseguir una actualización remota del firmware, y funcionalidades como el deep sleep para lograr un ahorro energético considerable una vez que el sistema deje de estar activo.

En este manuscrito se encontrarán múltiples capítulos, cada uno de los cuales tratará diferentes cuestiones que se han planteado a lo largo del desarrollo de este proyecto. El primer capítulo abordará el porqué de este proyecto, así como los objetivos que se plantean y se desean conseguir una vez se despliegue el sistema. A continuación, en el capítulo de Metodologías y Planificación, se definirán cada uno de los casos de uso y, por ende, los requisitos que se han identificado en el sistema. Además, se ilustrarán los tiempos necesarios para cada uno de los hitos y los costes, tanto materiales como humanos.

El capítulo del Estado del Arte hará un recorrido por las funcionalidades del sistema y las comparará con otras tecnologías disponibles actualmente en el mercado. Se expondrán estos productos para fortalecer, en contraposición, la visión del trabajo desarrollado, mostrando la relevancia de llevar a cabo este proyecto. El capítulo de Desarrollo será el más extenso y explicará detalladamente dos partes diferenciadas: el desarrollo del software y el desarrollo del hardware del sistema. En el primero, se hablará de todos los programas de software utilizados, explicando detalladamente su estructura interna y cómo se comunican entre ellos. Por otro lado, el desarrollo del hardware tratará principalmente los componentes que forman parte del sistema, así como el diseño y construcción de la PCB, explicando cada decisión tomada en relación con los componentes mencionados.

Además, se hablará de los consumos energéticos del sistema, de las pruebas realizadas y de las incidencias que se han presentado a lo largo del proceso. Una vez concluido este capítulo, en Resultados se mostrarán los objetivos alcanzados en este proyecto.

Por último, se resumirá en Conclusiones lo que se ha realizado durante todo este

tiempo y se presentarán algunas líneas futuras que se plantean en caso de querer continuar con ampliaciones o mejoras del sistema.

Abstract

This project introduces new tools from the world of IoT (Internet of Things) with the goal of improving crop production by optimally managing water consumption and the machinery that is part of the system. The smart irrigation head operates via drip irrigation and consists of both analog and digital components that serve the same function.

The system will automate all irrigation activities, using low-cost microcontrollers to control the activity of other components, leveraging IoT platforms to manage the values collected by the devices. To accomplish this, various functions will be developed, such as the implementation of a MESH network to connect multiple devices to the Internet, OTA for remote firmware updates, and deep sleep functionalities to achieve significant energy savings once the system is no longer active.

This manuscript will consist of multiple chapters, each addressing different issues that have arisen during the development of this project. The first chapter will discuss the rationale behind this project, as well as the objectives that are set and desired to be achieved once the system is deployed. Following this, the Methodologies and Planning chapter will define each use case and, consequently, the requirements identified in the system. Additionally, the timelines for each milestone and the costs, both material and human, will be illustrated.

The State of the Art chapter will review the system's functionalities and compare them with other technologies currently available in the market. These products will be presented to strengthen the vision of the developed work, demonstrating the relevance of undertaking this project. The Development chapter will be the most extensive and will detail two distinct parts: the software development and the hardware development of the system. The former will discuss all the software programs used, detailing their internal structure and how they communicate with each other. On the other hand, the hardware development will primarily address the components that are part of the system, as well as the design and construction of the PCB, explaining each decision made regarding the mentioned components.

Additionally, the manuscript will cover the system's energy consumption, the tests conducted, and the issues encountered throughout the process. Once this chapter is concluded, the Results chapter will present the objectives achieved in this project.

Finally, the Conclusions chapter will summarize what has been accomplished during this time and outline potential future directions for expanding or improving the system.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Contexto del Proyecto

En las últimas décadas, el cambio climático se ha convertido en una de las mayores amenazas para la humanidad [1]. A medida que las emisiones de gases de efecto invernadero continúan aumentando, el planeta se enfrenta a cambios ambientales sin precedentes que ponen en riesgo no solo la biodiversidad, sino también la estabilidad y la seguridad de las sociedades humanas.

El mayor factor de riesgo del cambio climático, relacionado con este trabajo es el impacto que tiene sobre la agricultura y la seguridad alimentaria [2]. Los patrones de precipitación están siendo alterados a nivel mundial, con algunas regiones experimentando aumentos en las lluvias y otras enfrentando sequías severas. [3]

Las sequías prolongadas pueden devastar cultivos y pastizales, reduciendo drásticamente la producción agrícola. Regiones como África subsahariana y partes de Asia ya están experimentando estos efectos [4], lo que agrava la inseguridad alimentaria en áreas que dependen de la agricultura de subsistencia. De manera similar, España ha enfrentado una situación crítica en 2023 [5], con sus embalses, que han alcanzado niveles históricamente bajos. Esta severa sequía en el país se tradujo en una crítica situación económica, con unas pérdidas que ascendieron a 5.550 millones de euros, dato que surge de un estudio de Aon sobre catástrofes naturales [6]. Esta escasez de agua no solo afecta al suministro de agua potable, sino que también tiene un impacto significativo en la agricultura y la producción de alimentos. Ambos escenarios subrayan la creciente vulnerabilidad de diversas regiones del mundo frente al cambio climático y la importancia de implementar estrategias efectivas para la gestión del agua y la seguridad alimentaria. Por otro lado, las lluvias excesivas y las inundaciones pueden destruir cultivos, erosionar suelos fértiles y causar pérdidas significativas de cosechas.

En relación con lo comentado, se encuentra el factor del aumento de temperaturas. Este incremento de las temperaturas globales influye en los rendimientos agrícolas de varias maneras como el estrés térmico [7] que causan en las plantas, reduciendo su crecimiento y productividad. En adición, se encuentra el fenómeno de la evapotranspiración [8] (demanda de agua que ejerce el clima sobre la planta). Al aumentar la temperatura, este proceso crece con ella, incrementando la cantidad de agua necesaria para los cultivos. En áreas donde el agua ya es un recurso limitado, esto

puede llevar a una competencia intensificada por el agua entre la agricultura, el uso doméstico y otras actividades.

Aunque se pueden encontrar varios problemas sumados a los anteriores como pudieran ser la fertilidad del suelo, incendios o las plagas y enfermedades [9], entre otros, los conflictos comentados en líneas anteriores son los más arraigados a las cuestiones que se plantean en este trabajo y a los que se intentarán poner remedio de una forma localizada a través del desarrollo de un cabezal de riego, intentando disminuir las consecuencias negativas que ejercen sobre la sociedad humana.

1.2. Objetivos del Proyecto

Poniendo en relación el problema real con este proyecto, lo que se pretende al realizar este trabajo es hacer hincapié en la resolución de forma eficiente del consumo de agua en los cultivos, siendo capaz el agricultor de gestionar de forma controlada el agua que se administrará en su cultivo. También se desea poder notificar con antelación posibles problemas relacionados con el deterioro y acumulación de impurezas que aparecerán en las tuberías o mangueras de las que consta el terreno. El objetivo es crear un sistema inteligente, que funcione de forma autónoma sin la necesidad de tener a una persona de forma presencial. El sistema podrá empezar y terminar de regar de forma remota sin necesidad de estar accionando un grifo cotidiano.

Un punto a favor es que las técnicas agrarias han ido evolucionando de manera positiva a lo largo de los siglos [10]. Los avances tecnológicos y científicos han sido trascendentales en esta actividad, elaborando nuevas máquinas e instrumentos que por consiguiente han creado nuevas formas de cultivo que obtienen mayor productividad del campo.

Siguiendo esta tendencia, con este trabajo se desea continuar evolucionando el sector primario o cualquier tipo de cultivo personal, dando nuevos mecanismos a la sociedad humana para poder gestionar y controlar de una forma más automatizada el campo, haciendo que el trabajo sea más sencillo para las personas. Por ello, se tiene en cuenta que el trabajo relacionado con el campo a día de hoy, una buena parte de este sigue llevándose acabo por personas de avanzada edad que posiblemente no hayan estado en contacto con tecnologías tan avanzadas por motivos evidentes.

La implementación de este sistema quiere conseguir la automatización del campo de una forma inteligente pero, siempre yendo de la mano con las técnicas ya existentes que aunque pueden ser rudimentarias, son necesarias que estén disponibles para que la brecha digital que pueda haber con estas personas sea lo más pequeña posible. Así el consumidor que pudiendo tener las dos técnicas presentes, le podrán servir para que pueda verificar la fiabilidad y el correcto funcionamiento del sistema al poder comparar las dos versiones. Estas similitudes serán explicadas de forma detallada en este apartado, unas líneas más abajo.

Por estos motivos principalmente, nace la necesidad e inquietud de realizar este trabajo final de grado. Aportando con las tecnologías IoT (Internet of Things) en el campo, que a primera vista parecen incompatibles, nuevas formas de gestión que hagan a los campos, a la sociedad humana y al Planeta en general, poder dar un paso más al frente contra la lucha del cambio climático y beneficiando de forma directa a toda persona que ponga en práctica estas medidas en sus cultivos.

Teniendo una visión genérica de lo que se pretende lograr con este trabajo, se han clasificado una serie de subojetivos claves a conseguir en el desarrollo de el cabezal de riego inteligente, de la siguiente manera :

1. **Cubrir grandes áreas de trabajo usando redes MESH**
   El cabezal de riego inteligente puede ser desplegado tanto en pequeñas como en extensas áreas. Esto se debe a la flexibilidad y sencillez de las tecnologías empleadas en este sistema. Dependiendo de la cantidad de área que se quiera cubrir, se pondrán un mayor o menor número de dispositivos en el sistema. Estos dispositivos que se mencionan y que serán utilizados, son los microcontroladores ESP32-S3 de la empresa Espressif Systems. Los micros son el cerebro del sistema, gestionando y organizando a todos los demás componentes que forman parte del conjunto. Si el área a cubrir fuese pequeña, puede llegar a implementarse un solo dispositivo que se encargue de todas las tareas necesarias para el buen funcionamiento del sistema, o bien, se puede hablar de varias decenas de dispositivos interconectados entre ellos de forma inalámbrica para poder aumentar el rango que pueden alcanzar por sí solos, llegando a distancias considerables. Esta red que es formada en el momento que se tiene más de un ESP32-S3 en el sistema, recibe el nombre de red mallada inalámbrica, en siglas MESH.

   El Cabezal de Riego Inteligente va a constar de 3 dispositivos ESP32-S3 formando una red MESH, donde se verán los 3 roles que puede tomar un dispositivo dentro de la red: nodo raíz, intermedio y hoja. Además, todos los nodos serán capaces de poder conectarse a Internet.

2. **Monitorización de todas las actividades del sistema utilizando aplicaciones IoT**
   Se desea poder llevar el control del sistema de forma remota utilizando distintos recursos software vinculados al IoT. Para ello, el Cabezal de Riego ha sido desarrollado a través de una plataforma IoT que contiene distintos programas que se relacionan entre sí. Uno de estos programas es Total Flow, donde se han creado todas las reglas necesarias para la comunicación.

   Complementario a esto, un bot de Telegram también será gestionado a partir de Total Flow. Desde este bot, el usuario podrá realizar distintas acciones como gestionar el estado de la electroválvula, ser alertado por algún tipo de problema en las presiones o ser avisado de la cantidad de agua suministrada al sistema.

   Total Flow va a ser la aplicación que determina cómo deben tratarse los datos que salen del dispositivo a Internet y también los datos que debe recibir el ESP32-S3 por parte de los usuarios. Además, se vincula una base de datos que recibe todos los datos recibidos por los sensores.

   Por otro lado, se creará un cliente de MQTT que funcione de una forma parecida a Whatsapp. Irán llegando notificaciones en forma de chat que irán informando de los datos que se envían y reciben.

   En adicción, como se ha comentado en líneas anteriores, una base de datos sencilla usando PostGreSQL se encargará de almacenar todos los datos de valores obtenidos por los sensores.

   Por último, se usará Grafana[11]. Software libre que permite la visualización y el

formato de datos métricos. Permite crear cuadros de mando y gráficos a partir de múltiples fuentes, incluidas bases de datos. Será una interfaz a la que el usuario tendrá acceso y desde ella podrá controlar los valores provenientes de la base de datos para así poder supervisar el sistema. Además, podrá encender y apagar la electroválvula de forma automática.

3. **Poder activar o desactivar el sistema sin necesidad de estar presencialmente en el campo**
   Aunque se ha comentado, destacar esta tarea pues aun pareciendo simple, es una de las de mayor peso para el usuario. Gracias a ella, el usuario podrá ahorrar tiempo en desplazamientos al campo simplemente haciendo un click desde el software de Grafana o Telegram que él mismo gestionará.

4. **Controlar pérdidas u obstrucciones en las tuberías**
   Otra de las tareas más importantes en el Cabezal, será poder saber en todo momento el estado en el que se encuentran las tuberías y mangueras de saturadas. El sistema constará de múltiples sensores de presión distribuidos de la siguiente forma: un sensor de presión antes y después de un filtro de impurezas que al comparar sus valores si superan cierto umbral, será síntoma de que la tubería está siendo obstruida por la acumulación de pequeños materiales que el agua lleva en ella. Estos sensores serán instalados con el microcontrolador raíz de la red MESH.

   Se añadirán otro par de sensores de presión también en el siguiente lugar: uno al inicio de la manguera y otro al final de esta. De forma idéntica al anterior par de sensores, se compararán sus presiones y de nuevo si superasen cierto umbral, sería síntoma en este caso de que habría pérdidas en algún lugar de la manguera. Pudiera deberse a algún tipo de rotura por la que estuviera escapando agua. Estos sensores serán instalados con los microcontroladores intermedio y hoja.

   De una forma u otra el usuario será avisado por Telegram para que tenga en cuenta la situación y que él ya realice las acciones pertinentes.

5. **Poder llevar un control absoluto de la cantidad de agua suministrada al sistema en todo momento**
   Para ello, se introduce un nuevo componente digital del que no se ha hablado aún: el caudalímetro. Este dispositivo será colocado a continuación del último sensor de presión del filtro.

   El caudalímetro irá informando de la cantidad de agua que se ha consumido a medida que el agua atraviese este. Su función es la misma que la de un contador habitual que se instala en las viviendas.

6. **Evitar la instalación de cables en el campo**
   Un punto fuerte de este proyecto es poder desplegar el sistema en el campo sin necesidad de tirar cables. Es decir, el Cabezal lo normal sería pensar que va a ser instalado dentro de una vivienda que estará provista de una instalación previa de cables. Sin embargo, los demás nodos de la red están pensados para poder ser instalados en distintos puntos del terreno, como podrían ser arquetas y no necesariamente se necesitarían utilizar cables para su instalación, pues esos microcontroladores pueden ser alimentados de diversas formas como baterías o pequeños paneles solares entre algunas de las ideas. Esto es posi-

ble debido a que estos microcontroladores no necesitan de grandes consumos energéticos para poder funcionar y tampoco lo necesitan los componentes que, principalmente son sensores de presión conectados a ellos.

Todo esto ofrece varios puntos positivos:

a) Menor impacto ambiental
Al no tener que excavar para colocar cables, se reduce la perturbación del suelo y la vegetación, preservando el ecosistema natural.

b) Ahorro de costos
La instalación de una red inalámbrica generalmente requiere menos materiales y mano de obra que la instalación de cables, lo que puede significar un ahorro significativo en costos.

c) Flexibilidad
Las redes inalámbricas son más fáciles de reconfigurar y expandir. Se pueden agregar o mover puntos de acceso sin necesidad de cambiar el cableado físico.

d) Menos mantenimiento
Los cables pueden dañarse por condiciones climáticas, animales o accidentes. Las redes inalámbricas, por otro lado, tienen menos componentes físicos que pueden fallar o requerir reparaciones.

e) Acceso en ubicaciones remotas
En áreas rurales donde la infraestructura de red es limitada, los dispositivos inalámbricos pueden conectarse a redes celulares o satelitales, proporcionando acceso a Internet en lugares donde sería impracticable instalar cableado.

f) Escalabilidad
La red inalámbrica puede escalar fácilmente agregando más nodos para ampliar la cobertura sin necesidad de grandes modificaciones en la infraestructura existente.

Estas ventajas hacen que las redes inalámbricas sean una opción atractiva para instalaciones en el campo, proporcionando conectividad eficiente y fiable sin las complicaciones y los costos asociados con el cableado físico.

7. **Actualización de versión del sistema por OTA (Over the Air)**
Teniendo bastante relación el apartado anterior con este: se desarrolla la actualización por OTA. Esto permite que si el sistema tuviera que tener algún tipo de actualización, no se necesite ir al lugar y tener que conectar un cable a los microcontroladores para poder compilar el código. El microcontrolador se encargará de actualizar su código directamente desde un Servidor en caso de que fuera necesaria una actualización.

8. **Construcción de una placa del sistema**
   Posiblemente sino es el que más, sea uno de los objetivos que más importancia tiene del proyecto. Se desarrollará una placa que gestione la actividad de todo el Cabezal. Esta PCB (Placa de Circuito Impreso) es construida con acetato de forma casera. En el apartado de desarrollo Hardware se explicará el proceso de fabricación.

   La placa estará formada por el ESP32-S3 que tiene la función de nodo raíz en la red MESH, junto a los dos sensores de presión que están situados alrededor del filtro, el caudalímetro y la electroválvula. De esta forma quedará contemplado todo el sistema que debería ir dentro de la vivienda, en una simple placa, evitando grandes instalaciones de cableados que ocupen un gran espacio.

9. **Compatibilidad de este sistema con técnicas habituales en el campo**
   Al comienzo de esta sección se explicó el deseo por modernizar ciertas actividades en el campo pero siempre respaldándose de mecanismos ya existentes para que, al final, cualquier usuario, pero principalmente los usuarios de avanzada edad, puedan respaldarse de los métodos que ya conocen para poder confiar en nuevas técnicas como las que se proponen en este documento. Por ello, cada sensor de presión digital que se ha citado, irá acompañado de un sensor de presión analógico, llamado manómetro, que sirva al usuario para ver los valores obtenidos en todo momento. Además, a parte del caudalímetro, justo delante de este, se encontrará un contador de agua, como los que se puede observar habitualmente en los hogares, para que de esta forma el usuario pueda saber la cantidad de agua que ha corrido por las tuberías.

   El propósito de tener los componentes analógicos en paralelo es dar confianza al usuario de los datos obtenidos con los componentes digitales una vez sean comparados.

El cabezal de riego inteligente constará de 3 microcontroladores que forman una red MESH entre ellos, enviando datos al exterior, informando del valor de los sensores de presión y del caudalímetro para que estos datos sean tratados por el software implementado para el sistema. Además el nodo raíz recibirá datos que dictaminan cómo quiere el usuario el estado de la electroválvula. Este nodo raíz formará parte de la placa junto los dos sensores de presión que rodean al filtro y el caudalímetro que irá informando del consumo de agua. Los otros dos nodos, que cada uno de ellos tendrá conectado un sensor de presión, simularán ser arquetas en el campo, para que se pueda probar la funcionalidad de la red mallada. Todas estas tareas del sistema, serán los resultados que el usuario podrá apreciar desde Grafana y Telegram.

Capítulo 2

Metodología y planificación

El objetivo de este apartado es numerar cada uno de los requisitos que forman parte del proyecto y dar una explicación detallada de ellos utilizando casos de uso extendidos que ilustrarán de forma precisa cada uno de los pasos por los que estarán sometidos el sistema y el usuario. Además se mostrará la gestión llevada con cada uno de los hitos a lo largo del tiempo usando diagramas de Gantt. En ellos se podrán observar los tiempos esperados y los reales en los que se han llevado a cabo cada uno de las tareas del sistema. Sumado a lo anterior, se darán detalles sobre los costes del proyecto.

2.1. Casos de uso

Se muestra cada una de las necesidades que tiene este sistema con el requisito que posteriormente está relacionado con este . Se definen estas tareas mediante el uso de casos de uso. Los casos de uso son una extensión del concepto de casos de uso en el análisis y diseño de sistemas, específicamente en el ámbito de la ingeniería de software. Se utilizan para capturar interacciones más detalladas y complejas entre usuarios (actores) y el sistema.

CU001 Habilitar/Deshabilitar el suministro de agua		
Descripción	El usuario podrá habilitar y deshabilitar el paso del agua por el sistema	
Comentario	Si el sistema, tenía la electroválvula abierta cuando se dio al botón de 'ON', realmente no surge ningún efecto. Idéntico si ocurre el contrario.	
Precondición	El sistema esté conectado a Internet	
Secuencia normal	Paso	Acción
	1	El usuario debe pulsar el botón 'ON' u 'OFF' en su interfaz
	2	El sistema manda la orden y la electroválvula pasa a estado abierto/cerrado
	3	El usuario recibe un mensaje notificando que la electroválvula ha sido abierta/cerrada
Postcondición	El sistema comienza/termina el suministro de caudal	
Requisito Funcional asociado	RF – 01 Suministro de agua al sistema	

Figura 2.1: Caso de uso número 1: Habilitar/Deshabilitar el suministro de agua.

CU002 Notificaciones de las presiones		
Descripción	Permitir al usuario recibir una notificación sobre las presiones cuando la diferencia entre ellas supere el umbral permitido para advertirle de posibles problemas	
Comentario	Esta notificación será acompañada del punto exacto donde se está produciendo el problema, pues se tiene dos puntos donde ver la diferencia de presiones: en el filtro y en la manguera	
Precondición	El sistema esté conectado a Internet	
Secuencia normal	Paso	Acción
	1	El sistema manda al usuario una notificación de advertencia sobre las presiones
Postcondición	-	
Requisito Funcional asociado	RF – 02 Notificación de alarma sobre las presiones	

Figura 2.2: Caso de uso número 2: Notificación con el valor de las presiones.

CU003 Notificación de cantidad de agua suministrada en el máximo establecido		
Descripción	Permitir al usuario recibir una notificación que lo advierta de que la cantidad de agua que fue establecida para suministrar sus cultivos ha sido alcanzada y por tanto aconsejarle el cierre del sistema	
Comentario	El sistema no dejará de funcionar hasta que el usuario decida cerrarlo.	
Precondición	El sistema esté conectado a Internet	
Secuencia normal	Paso	Acción
	1	El sistema manda al usuario una notificación de advertencia sobre la cantidad de agua suministrada
Postcondición	-	
Requisito Funcional asociado	RF – 03 Notificación de cantidad agua suministrada alcanzada	

Figura 2.3: Caso de uso número 3: Notificación con el valor del agua suministrada.

CU004 Interfaz para visualización de datos		
Descripción	Permitir al usuario poder visualizar todos los ratos relacionados con el sistema	
Comentario	Conexión a Internet	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El usuario podrá ver los valores relacionados con las presiones
	2	El usuario podrá ver el valor relacionado con el caudal suministrado
	3	El usuario podrá ver el valor relacionado con el estado de la electroválvula
Postcondición	-	
Requisito Funcional asociado	RF – 04 Interfaz de visualización de datos	

Figura 2.4: Caso de uso número 4: Interfaz para visualizar datos.

CU005 Actualización del software por OTA		
Descripción	Este caso de uso explica cómo debe actualizarse el sistema de forma inalámbrica vía OTA	
Comentario	-	
Precondición	El sistema esté conectado a Internet	
Secuencia normal	Paso	Acción
	1	El sistema debe comparar su versión con la del servidor para ver si debe actualizarse
	1.1	Si la versión es igual o inferior, no pasará nada
	2	El sistema solicita el binario al servidor
	3	El sistema se reinicia
	4	El sistema es actualizado con el nuevo firmware
Postcondición	El sistema se encontrará en la última versión desplegada	
Requisito Funcional asociado	RF – 05 Actualización del software por OTA	

Figura 2.5: Caso de uso número 5: Actualización por OTA.

CU006 Comunicación por red MESH		
Descripción	Este caso de uso debe permitir al sistema comunicarse sus nodos mediante una red mallada	
Comentario	-	
Precondición	Todos los dispositivos tengan conexión a internet	
Secuencia normal	Paso	Acción
	1	El nodo raíz se conecta a un punto de acceso
	2	El usuario raíz se convierte en un propio punto de acceso
	3	Los otros nodos, se conectan al punto de acceso del nodo raíz
Postcondición	-	
Requisito Funcional asociado	RNF – 01 Creación de red inalámbrica mallada	

Figura 2.6: Caso de uso número 6: Comunicación por MESH.

2.2. Requisitos del sistema

Los requisitos del sistema constituyen la piedra angular en el desarrollo de cualquier proyecto. Estos requisitos definen las capacidades, funciones y restricciones que debe cumplir el sistema, asegurando que cumpla con las expectativas y necesidades de los usuarios y otras partes interesadas.

Para analizar las funciones que debe cumplir el cabezal de riego inteligente, se han introducido tanto requisitos funcionales como no funcionales.

2.2.1. Requisitos funcionales

Estos requisitos describen las funciones específicas que el sistema debe realizar.

1. **RF-01: Suministro de agua al sistema**
   Es necesario tener mecanismos que de forma remota puedan permitir y cerrar el paso de agua por las tuberías de manera que el sistema comience a regar.Asociado a CU001.

2. **RF-02: Notificaciones de las presiones**
   El usuario debe poder recibir por medio de aplicaciones software, notificaciones que lo avisen de ciertos problemas que podrían estar ocurriendo en el interior de las tuberías pues, esta notificación se envía cuando la diferencia de valor entre dos sensores de presión, supera cierto umbral. Recordar, que estos mensajes se podrán referir en este trabajo a dos circunstancias: la obstrucción del filtro o bien algún tipo de rotura o pérdida en las mangueras. Asociado a CU002.

3. **RF-03: Notificaciones de la cantidad de agua suministrada, supera el máximo establecido**
   El usuario debe poder recibir por medio de aplicaciones software, notificaciones que lo avisen cuando la cantidad de litros suministrados al sistema, haya superado el número de litros establecidos como óptimos para el riego del cultivo que se tenga y dependiendo las condiciones atmosféricas. El sistema no terminará de funcionar, solamente informará. Asociado a CU003.

4. **RF-04: Interfaz para la visualización de datos**
   El usuario tendrá una interfaz desde la que podrá observar el valor de las presiones y de la cantidad de agua suministrada al sistema, así como el estado del sistema si estuviera con el paso de agua habilitado o deshabilitado. Asociado a CU004.

5. **RF-05: Actualización software por OTA**
   El sistema será capaz de actualizar su código de forma remota sin necesidad de cables ni personas de por medio. Cogerá información de un servidor y este se encargará de suministrarle lo que necesite en ese preciso momento. Asociado a CU005.

2.2.2. Requisitos no funcionales

Estos requisitos describen cómo debe funcionar el sistema.

1. **RNF-01: Creación de red inalámbrica mallada**
   El sistema estará conectado a Internet solamente por un único dispositivo que será el nodo raíz. Los demás dispositivos se conectarán a partir de este nodo raíz, ampliando el rango de cobertura a zonas donde no habría Internet. Asociado a CU006.
2. **RNF-02: Seguridad y fiabilidad**
   El sistema cifrará los datos entre los distintos nodos usando el protocolo TLS (habilitado en los microcontroladores).
3. **RNF-03: Eficiencia Energética**
   El sistema estará disponible la mayoría del tiempo pero no tiene sentido tenerlo en ejecución prácticamente en todo momento. Por este motivo, se usa el modo deep sleep, para reducir al mínimo su consumo de energía al apagar la mayoría de sus componentes y funciones, mientras mantiene solo lo esencial para un rápido despertar y reanudación de operaciones cuando sea necesario

2.3. Tiempos llevados a cabo para cada hito

En este apartado se va a detallar la distribución de tiempo llevada a cabo para cada uno de los hitos y sus actividades. Se utilizarán diagramas de Gantt para su explicación, en concreto dos. El primero de ellos muestra el tiempo esperado aproximado de cada tarea pensado al principio y el segundo muestra el tiempo real que se ha necesitado para cada una de ellas.

Primero se mostrarán las dos planificaciones y después se analizarán, mencionando las principales diferencias a lo largo del tiempo.

A continuación, la imagen mostrará el diagrama de Gantt relacionado con los tiempos ideales esperados en los que se pretendía cumplir el proyecto al principio de este.

En ella, se pueden ver 4 hitos principales: desarrollo de la OTA, desarrollo de la MESH, construcción de la PCB y acoplar todas las aplicaciones software con la base de datos al sistema.

Por otro lado, se encuentra, un hito opcional que se realizaría en caso de ir con tiempo suficiente, que es el denominado miscelánea.

Estos hitos comenzaron en los inicios de 2024 y se pretendían terminar a principios de Junio.

Aquí se ilustra el diagrama ideal del que se ha hablado:

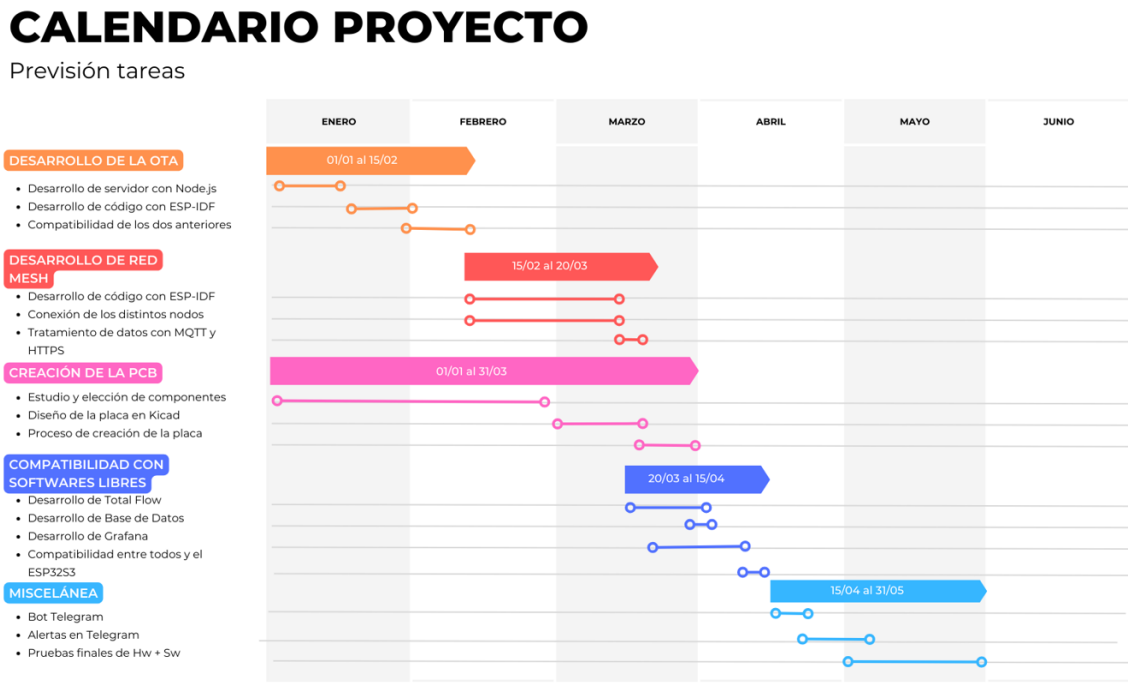

Figura 2.7: Diagrama de Gantt con los tiempos estimados de desarrollo.

Ahora, la imagen que se mostrará, ilustra los tiempos reales que ha necesitado cada una de las tareas a lo largo de estos meses.

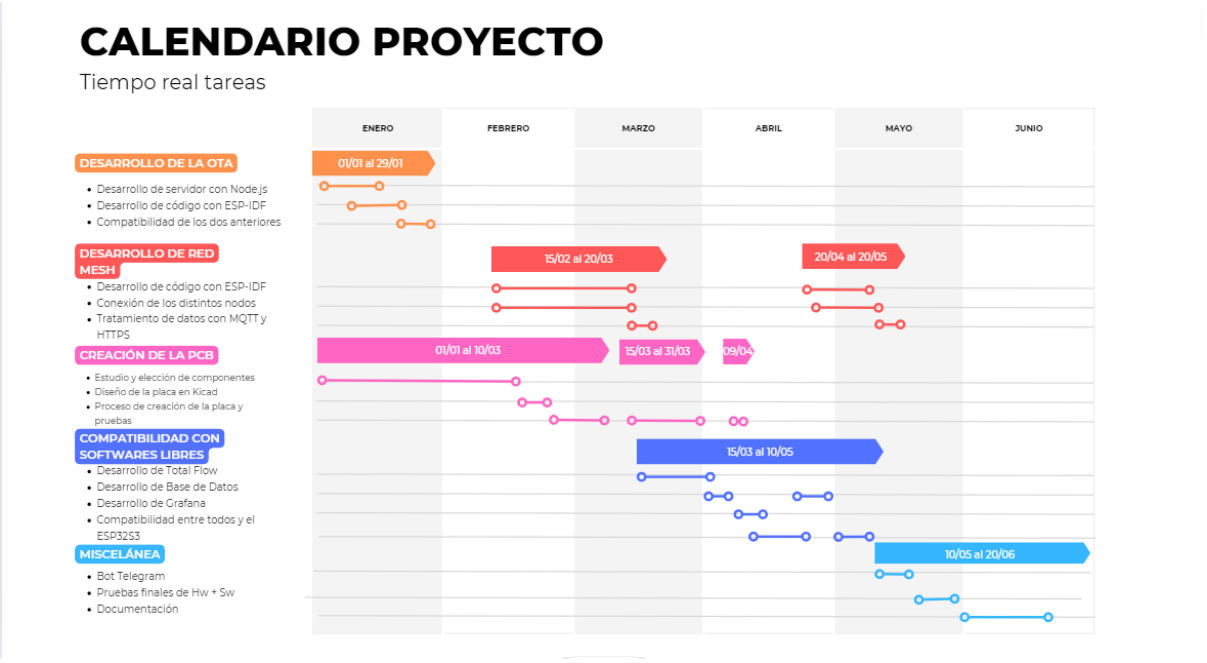

Figura 2.8: Diagrama de Gantt con los tiempos reales de desarrollo.

Como se puede observar, los hitos y actividades no han cambiado, los tiempos si. Las diferencias de lo esperado a lo real han sido principalmente por los siguientes motivos:

1. Desarrollo de la OTA
   Se esperaba terminar para mediados de febrero y se consiguió ahorrar más de dos semanas. Esto fue posible a que el servidor con Node.js fue fácil de desarrollar y probar con otros medios que no fueran el microcontrolador. Por tanto, se consiguió un tiempo adicional para poder desarrollar el código del ESP32-S3 que permitiera realizar actualizaciones por el aire.

2. Desarrollo de la MESH
   Es el hito que más complejidad y tiempo ha llevado del proyecto. En el tiempo real se pueden mostrar 3 importantes momentos. El primer intervalo lógicamente el más largo, sirvió para entender realmente qué era una red mallada, cómo funcionaba, para qué se quería introducir en este trabajo y cuáles iban a ser sus ventajas. Una vez entendido el concepto, se comenzó a barajar distintas opciones de redes malladas que ofrecía la empresa Espressif Systems con sus microcontroladores. Se probó primeramente con el protocolo ESP-NOW, pero su funcionamiento no se ajustaba a los requisitos de este sistema, pues no era reconfigurable, se necesitaba saber en todo momento dónde se destinaban los paquetes y no tenía capacidad de viajar a Internet. Después de ello, se comenzó a realizar avances con el protocolo ESP-MESH del microcontrolador. Y al finalizar la primera etapa, rondando el 10 de marzo de 2024, ya se era posible conectar nodo raíz con intermedios y hojas y verificar sus roles. Se comenzó a pensar cómo se quería mandar los paquetes desde los hijos al exterior. La principal idea era mandarlos al exterior pasando por el nodo raíz pero este simplemente redireccionándolo fuera sin necesidad de hacer más funciones que esa. El problema fue que el protocolo MESH Lite de Espressif Systems es bastante reciente y aun carece de ciertas facilidades y una de ellas fue esta. Dada su complejidad se decidió usar MESH-LITE. Igual que ESP-MESH pero más sencilla en cómputos. Con este sistema, todos los nodos están conectados a Internet y son capaces de enviar y recibir del exterior sin necesidad de tener que mandar sus paquetes a través de su nodo raíz. Esto fue el último intervalo que se puede ver, que duró poco pues al final se sabía ya de una forma muy especializada todo acerca de la MESH.

3. Desarrollo de la PCB
   Se divide en tres intervalos de tiempo principalmente. En el primero se estudió el sistema y lo que necesitaba. Por tanto se eligieron las piezas adecuadas viendo detalladamente sus propiedades. Una vez realizado este estudio y selección, se desarrolló el diseño de la PCB usando herramientas software. Llevó aproximadamente un mes este diseño pues aun había factores externos que iban cambiando y se tuvieron que tomar decisiones en ese tiempo que iban modificando el diseño. Tras el primer intervalo, el 15 de marzo se comenzó a elaborar la placa en el laboratorio de la Universidad. En este proceso se llevan a cabo unas cuantas fases que duraron unas 3-4 horas en total. Se tardó más de la cuenta porque la primera placa que se hizo fue un fracaso debido a que los químicos que se usan y que se explicarán más adelante, destruyeron la PCB. Se tuvo que realizar otra que ya fue satisfactoria y después de ello se empezó a soldar todos los componentes. El 9 de abril se comprobó que la placa funcionaba

correctamente y terminó el proceso de construcción.

4. Compatibilidad con softwares libres y BBDD con el ESP32
   Aproximadamente se ha tardado 2 meses en tener todo correctamente gestionado pero no ha supuesto un excesivo trabajo pues las herramientas que se han utilizado han sido fáciles de relacionar entre ellas en todo momento. Total Flow es el punto en común de todas. Cuando se realizaban en este software los avances, los demás podían avanzar en paralelo.

5. Miscelánea
   Finalmente, se dispuso de cierto tiempo libre con el que poder realizar el desarrollo de un bot de Telegram que se encargara de enviar notificaciones al usuario. En el diagrama anterior, se colocó por separado el bot y las alarmas, aquí se han fusionado pues al final las dos actividades eran la misma. También se han incluido aquí pruebas de Hardware con Software. Aproximadamente todo ha durado un mes y el diseño del bot ha sido fácil porque todas las reglas ya estaban creadas en Total Flow. Simplemente se ha necesitado vincular una aplicación con otra. También se han incluido aquí pruebas de Hardware con Software. Aunque se han ido realizando pruebas a lo largo de los meses con una cosa u otra, las pruebas más exhaustivas han sido en el último periodo.
   Por último se ha llevado a cabo la documentación del proyecto.

2.4. Tablas de costes

Costes			
Nombre	Cantidad	Precio (€/unidad)	Enlace a tienda
G3/4 Electroválvula solenoide latón	1	42,99	Amazon
G1/4 Pressure Sensor Transduce	4	14,87	Amazon
Manómetros de 4 bares	4	14,76	Automation24
Caudalímetro YF-S201	1	8,99	Amazon
Candeon Contador de Agua	1	35.09	Amazon
Cinta de Teflón x 4 Rollos	1	7,99	Amazon
ESP32-S3-DevKitC-1-N8R8	3	16,88	Mouser
Alimentador 24V 1A	1	9,50	Amazon
Regulador de tension L7812CV	1	0,47	Mouser
Regulador de tensión 7805	1	0,16	Gocar Electronic
Disipador radiador	2	0,60	Conectrol
Condensador 0,33 uF	2	0,20	Electropolis
Condensador 0,1 uF	2	0,13	Cetronic
Resistencias 1k07	3	0,1	Tecnoteca
Resistencias 1k74	3	0,1	Diotronic
Jumper Wire cables de Puente	1	5,99	Amazon
Tubo Termoretráctil	1	1,4	Cetronic
V-204 placa virgen fibra	1	3,21	Cetronic
Polietileno de baja densidad	1	16,65	Electropolis
Base Jack DC macho	1	0,35	Conectrol
Relé de 5V	1	1,69	Electronicaymas

Cuadro 2.1: Costes materiales.

En esta tabla se muestran cada uno de los materiales y componentes electrónicos que se han usado en el proyecto.

La suma de todos ellos hacen un total de 307,08 euros.

Por otro lado, el salario que tendrá el ingeniero, suponiendo una tarifa de 15 euros brutos por hora, trabajando solamente una persona desde enero hasta junio en los días laborales, se ha obtenido de la siguiente forma: Se han calculado los días laborales que ha habido en la Comunidad de Madrid del 1 de Enero a finales de Junio y salen un total de 130 días. Suponiendo una jornada laboral de 8 horas: 130 días cada uno con sus 8 horas hacen un total de 1040 horas trabajadas en el proyecto. Por tanto si la hora estaba pagada a 15 euros, la mano de obra es de 15.600 euros.

En resumen: Suponiendo una tarifa de mano de obra de 15 euros por hora y que solo ha trabajado una persona desde enero hasta junio en los días laborales, el salario total a pagarle sería de 15,600 euros. Este cálculo se basa en un total de 1040 horas de trabajo, considerando una jornada laboral de 8 horas al día y sumando los días laborales de cada mes durante este período. Sumado a lo anterior están los 307,08 euros de materiales.

La suma del salario del trabajador con la de los componentes en estos seis meses, hacen el total que habría que destinar al despliegue de este sistema, que es de 15.907,08 euros.

Capítulo 3

Estado del arte

Los cabezales de riego inteligentes comercializados, en general han emergido como una solución innovadora para optimizar el uso del agua, mejorar la productividad de los cultivos y reducir los costos operativos [12]. Estos dispositivos utilizan tecnologías avanzadas como sensores, controladores y algoritmos de inteligencia artificial para ajustar automáticamente el riego en función de las condiciones del suelo y el clima.

La implementación de tecnologías inteligentes en los sistemas de riego es fundamental para enfrentar los desafíos actuales, como el cambio climático y la necesidad de prácticas agrícolas más sostenibles. Los cabezales de riego inteligentes, al permitir un control preciso y automatizado del riego, representan un avance significativo en la agricultura de precisión. Este enfoque no solo contribuye a la conservación del agua, sino que también mejora la salud de los cultivos y aumenta la rentabilidad de las explotaciones agrícolas.

El objetivo de este apartado es revisar la literatura existente sobre los cabezales de riego inteligentes, analizando las principales investigaciones, desarrollos tecnológicos y aplicaciones prácticas y destacar las propias de este proyecto. Esta revisión permitirá identificar las tendencias actuales, las innovaciones más destacadas y las áreas que requieren mayor investigación.

A continuación, se revisan estudios de caso y proyectos implementados en diversas regiones donde se hace una comparativa con este proyecto mostrando, ventajas que pueden encontrarse en este sistema.

1. **Aqua Control C4099N**

   El Aqua Control C4099N [13] es un dispositivo innovador y ampliamente reconocido en el mercado, específicamente diseñado para automatizar y gestionar el riego en jardines y áreas verdes. Este programador de riego permite a los usuarios configurar horarios de riego automáticos, ajustando los días y horas en los que se desea que se active el riego. Esta funcionalidad es especialmente útil para garantizar que las plantas reciban el riego adecuado.

   Una de las principales ventajas de este programador es su capacidad para simplificar el mantenimiento de jardines y áreas verdes. Al automatizar el proceso de riego, se asegura que las plantas reciban la cantidad adecuada de agua en los momentos precisos, lo que contribuye a un crecimiento saludable y sostenido. Sin embargo, a pesar de sus numerosos beneficios, el Aqua Control C4099N

presenta algunas limitaciones importantes que deben considerarse para un uso más eficiente y responsable.

Una de las principales limitaciones del Aqua Control C4099N es que requiere una configuración manual por parte del usuario. Esto significa que, una vez programado, el dispositivo continuará operando según la configuración establecida hasta que se realice una nueva modificación manual. Esta característica puede ser problemática en situaciones donde se busca un consumo responsable del agua. Por ejemplo, en caso de cambios climáticos inesperados, como la llegada de lluvias, el dispositivo continuará con su programación de riego a menos que una persona lo ajuste manualmente. Esto puede resultar en un desperdicio significativo de agua y, además, podría ocasionar un riego excesivo, afectando negativamente a la salud de las plantas.

El hecho de que el Aqua Control C4099N no sea un dispositivo IoT (Internet de las Cosas) implica que no puede ser controlado o ajustado de forma remota. Esta falta de conectividad limita su funcionalidad y eficiencia, ya que siempre será necesario que una persona esté presente para realizar cualquier ajuste necesario. En un mundo donde la tecnología IoT está cada vez más integrada en nuestras vidas diarias, la capacidad de controlar dispositivos a distancia se ha vuelto una característica crucial para muchos usuarios.

En este trabajo, se propone una mejora significativa mediante la utilización de una electroválvula controlable de forma remota. Esta mejora permitiría evitar los problemas mencionados anteriormente, siempre y cuando el usuario haga un buen uso del sistema. El cabezal de riego rnteligente desarrollado en este proyecto aborda esta cuestión utilizando tecnologías IoT, lo que permite que una persona, desde el software de Grafana o a través de la aplicación de mensajería Telegram, pueda enviar órdenes de apagado o encendido al sistema. Esto no solo permite un ahorro significativo de agua, sino que también garantiza un mantenimiento óptimo de los cultivos, adaptándose de manera más eficiente a las condiciones cambiantes del entorno.

2. **Sistema de riego BRESSER Smart Garden Smart Home con ampliación de sensor del suelo**

   El BRESSER Smart Garden Smart Home [14] es una solución avanzada y completa para la gestión del riego en jardines, diseñada para integrar la tecnología inteligente con las necesidades de riego de plantas y césped. Este sistema se destaca por su capacidad para automatizar el riego, optimizando el uso del agua y asegurando que las plantas reciban el cuidado adecuado sin la necesidad de intervención constante del usuario.

   Una de las características más destacadas del BRESSER Smart Garden Smart Home es que se puede controlar a través de una aplicación móvil. Esto permite a los usuarios gestionar el riego de sus jardines desde cualquier lugar con acceso a internet. La aplicación proporciona una interfaz intuitiva desde la cual se pueden establecer horarios de riego, ajustar la duración y frecuencia del riego, y monitorear el estado del sistema en tiempo real. Esta funcionalidad es especialmente útil para aquellos que desean tener un control total sobre el riego de sus cultivos.

   El BRESSER Smart Garden Smart Home incorpora una serie de sensores que

monitorean las condiciones del suelo. Uno de estos sensores es el Sensor del Suelo BRESSER [15], que mide parámetros como la humedad del suelo.

Aunque el sistema BRESSER representa una solución tecnológica avanzada que combina la automatización y el monitoreo en tiempo real para ofrecer un riego eficiente, presenta algunas limitaciones en términos de cobertura. El sensor del suelo puede conectarse a una distancia máxima de 60 metros del temporizador, lo que puede restringir su uso en áreas más extensas. Esta limitación puede ser un desafío para aquellos que necesitan cubrir áreas grandes o jardines con una distribución compleja.

En comparación, el cabezal de riego inteligente desarrollado en este trabajo ofrece una solución más avanzada en términos de cobertura y flexibilidad. En condiciones favorables, las distancias entre nodos del sistema pueden llegar hasta 100 metros. Además, esos 60 metros representan la cobertura máxima del sistema BRESSER, ya que todos los dispositivos necesitan estar conectados al temporizador.

Una de las ventajas más significativas del cabezal de riego Inteligente es su estructura de red mallada. A diferencia de una red convencional donde todos los dispositivos dependen de una conexión directa al temporizador, una red mallada permite que los dispositivos se comuniquen entre sí, formando una red interconectada que puede extenderse mucho más allá del alcance individual de cada sensor o dispositivo. Gracias a esta configuración, las distancias que se pueden cubrir con el sistema de riego se amplían significativamente, pudiendo llegar incluso a kilómetros si fuera necesario. Esta capacidad de extensión y conectividad no solo mejora la cobertura y eficiencia del sistema de riego, sino que también aumenta su fiabilidad y flexibilidad.

La red mallada permite una gestión más precisa y adaptada a las necesidades específicas de cada cultivo, independientemente de su tamaño. Esto significa que, sin importar cuán grande o complejo sea el área a regar, el sistema puede ajustarse para proporcionar un riego óptimo en todas las zonas. Además, la estructura de red mallada mejora la resiliencia del sistema. Si un dispositivo falla o se desconecta, los otros dispositivos en la red pueden seguir comunicándose entre sí, asegurando que el sistema continúe funcionando de manera eficiente.

En conclusión, la red mallada del sistema en comparación con el modelo que se ha estudiado, ofrece una ventaja notable al permitir una cobertura mucho mayor y una mayor flexibilidad en la gestión del riego. Esta capacidad de extensión y conectividad mejora la eficiencia del riego, optimiza el uso del agua y garantiza un cuidado adecuado de las plantas, todo ello sin requerir una intervención constante del usuario. La integración de tecnología avanzada y la estructura de red mallada hacen de este sistema una solución superior, proporcionando beneficios significativos en términos de conveniencia, eficiencia y adaptabilidad.

3. **Repetidores de señal convencionales**

   Una red MESH ofrece numerosas ventajas significativas sobre un repetidor tradicional, destacándose en términos de cobertura, fiabilidad, escalabilidad y rendimiento general. La cobertura ampliada es una de las principales fortalezas de una red mallada, ya que múltiples nodos distribuidos se comunican entre sí para proporcionar una cobertura más amplia y uniforme. Cada nodo actúa

como un punto de acceso que puede recibir y retransmitir señales, permitiendo que la red se extienda más allá de las limitaciones de una única unidad central. En contraste, un repetidor simplemente amplifica la señal de un punto de acceso principal, pero su efectividad se reduce con la distancia y las obstrucciones físicas.

La fiabilidad y redundancia son otras áreas donde una red MESH supera a un repetidor. Los nodos en una red mallada crean múltiples rutas para los datos, lo que significa que si un nodo falla o se desconecta, los datos pueden ser redirigidos a través de otros nodos, mejorando significativamente la fiabilidad de la red. Por otro lado, un repetidor añade un solo punto de retransmisión, y si este falla, los dispositivos que dependen de él perderán la conexión, causando interrupciones en la red.

En cuanto a la escalabilidad, una red MESH es fácil de expandir añadiendo más nodos, permitiendo una expansión flexible y gradual de la red sin una pérdida significativa de rendimiento. Esto contrasta con los repetidores, donde añadir más unidades puede ser complejo y puede causar interferencias, ya que todos dependen del mismo canal de comunicación. Además, estas redes puede balancear las cargas de tráfico distribuyendo datos a través de múltiples rutas y nodos, evitando la congestión y manteniendo el rendimiento de la red. En cambio, un repetidor puede experimentar cuellos de botella cuando múltiples dispositivos están conectados a él, ya que toda la señal amplificada se canaliza a través de un único punto de acceso.

La facilidad de instalación y mantenimiento es otra ventaja de las estas redes. Los sistemas MESH modernos están diseñados para ser fáciles de instalar y gestionar a través de aplicaciones móviles o interfaces web intuitivas, con nodos que se configuran automáticamente y se ajustan dinámicamente para optimizar la red. En comparación, la configuración de repetidores puede ser más complicada y puede requerir ajustes manuales para evitar interferencias y optimizar el rendimiento.

Además, una red mallada proporciona una señal más consistente y estable en toda el área de cobertura, ya que los dispositivos se conectan al nodo más cercano y la red gestiona dinámicamente las conexiones para mantener la calidad de la señal. Por el contrario, la señal amplificada por un repetidor puede debilitarse con la distancia y las interferencias, resultando en zonas con señales inconsistentes o débiles.

En términos de seguridad, las MESH suelen incluir características de seguridad avanzadas, como encriptación de extremo a extremo, y son más fáciles de actualizar para mantener la seguridad. Un repetidor, sin embargo, puede ser más vulnerable a fallos de seguridad si no se configura correctamente, y la seguridad puede ser más difícil de gestionar a medida que se añaden más dispositivos.

En conclusión, las redes MESH ofrecen una solución más robusta, flexible y eficiente para extender la cobertura de la red en comparación con los repetidores tradicionales. Son especialmente útiles en entornos grandes y complejos donde la fiabilidad, la cobertura y el rendimiento son críticos. La capacidad de autoajuste y autorreparación de una red mallada la convierte en una opción superior para muchas aplicaciones de red moderna, tanto en hogares como en entornos

empresariales.

4. **Sensor control de riego smart Wi-Fi [16]**
   El sensor de control de riego smart Wi-Fi [16] monitorea y gestiona el sistema de riego de manera inteligente, ajustando automáticamente los tiempos de riego en función de las condiciones climáticas y la humedad del suelo. Esto permite optimizar el uso del agua, garantizando que las plantas reciban la cantidad adecuada sin desperdiciar recursos. Además, permite a los usuarios controlar y programar el riego desde cualquier lugar mediante una aplicación móvil.

   Actualmente, este dispositivo no tiene actualizaciones por OTA. Tener actualizaciones por OTA sería beneficioso porque permitiría mantener el sensor actualizado con las últimas mejoras de software, correcciones de errores y nuevas funcionalidades sin necesidad de intervención física. Esto no solo mejoraría la eficiencia y la seguridad del dispositivo, sino que también facilitaría la implementación de ajustes y mejoras continuas, optimizando aún más el rendimiento y la conveniencia para el usuario.

   La OTA ofrece múltiples ventajas adicionales. En primer lugar, mejora la seguridad del dispositivo, ya que las actualizaciones frecuentes pueden abordar vulnerabilidades y proteger el sistema contra amenazas emergentes. En segundo lugar, permite una mayor personalización y adaptabilidad, ya que los usuarios pueden recibir nuevas funciones y mejoras sin tener que reemplazar el hardware.

   Además, las actualizaciones por OTA reducen significativamente el tiempo de inactividad, ya que las actualizaciones se pueden realizar de manera remota y automática sin necesidad de desconectar el dispositivo ni de realizar complicadas intervenciones manuales.

   Otra ventaja es la facilidad de mantenimiento. Con las actualizaciones OTA, se puede ofrecer soporte continuo y mejorar la experiencia del usuario a lo largo del tiempo, lo que aumenta la satisfacción del consumidor y la vida útil del dispositivo. Esto también implica un menor costo total de propiedad, ya que el dispositivo puede evolucionar y mejorar con el tiempo sin incurrir en costos adicionales significativos para el usuario.

5. **Uso de microcontroladores con PCB**
   Este sistema dispone de unos pocos dispositivos y un microcontrolador que será actualizado por OTA. Como se buscaba en los objetivos del proyecto, el despliegue de este proyecto es limpio y con una infraestructura mínima, evitando de esta forma el cableado que otros sistemas de riego despliegan.. La PCB a desarrollar se encargará de contener cada uno de los cables y el microcontrolador se encontrará ahí alimentando directamente por la fuente de alimentación. Este sistema cabe perfectamente en una caja de 10x10cm sin necesidad de sacar cables al exterior, evitando ocupar espacios innecesarios, ahorrando costes y posibles averías al tener todo concentrado de una forma mayor y más pequeña.

Se va a analizar con detalle las tecnologías del sistema y sus efectos positivos.

1. **Redes MESH**
   ESP-WIFI-MESH [17], que es la que se utiliza en este trabajo, utiliza una topología de árbol bidireccional. Esta topología está formada por un nodo raíz que se encuentra en la capa o nivel 0. A partir de este, se van uniendo los demás en las siguientes capas. Esta topología en la MESH es capaz de que dispositivos que se encuentran fuera del rango de cobertura de un punto de acceso (router, módulo LTE, móviles con función de hotspot...), sean capaces de unirse mediante la conexión con otros dispositivos intermedios que si están en el rango de cobertura del punto de acceso. Es decir, nodos que se encuentren en niveles muy lejanos al 0, podrán conseguir cobertura uniéndose mediante niveles intermedios.

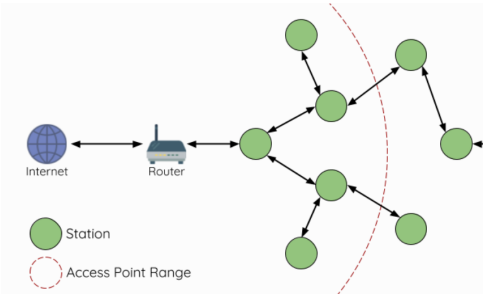

Figura 3.1: ESP-WIFI-MESH arquitectura de la red [17].

La infraestructura que sigue la MESH es bastante sencilla: generalmente, un solo dispositivo tomará el rol de nodo raíz. Es el encargado de conectarse al punto de acceso. A partir de él, se unen los demás dispositivos que ya no estarán conectados al punto de acceso original, sino que utilizarán al nodo raíz como punto de acceso. Estos dispositivos dependiendo el nivel de profundidad que tengan serán nodos intermedios que sirven de enlace en la conexión entre el nodo raíz y otros nodos que no están en el rango de cobertura del nodo raíz y que llamaremos nodos hoja. Los nodos hoja son los que se encuentran en el nivel más alto de profundidad (nodo raíz es el nivel 0). Es decir, no hay otros nodos que dependan de estos para unirse a la red, son los más lejanos.

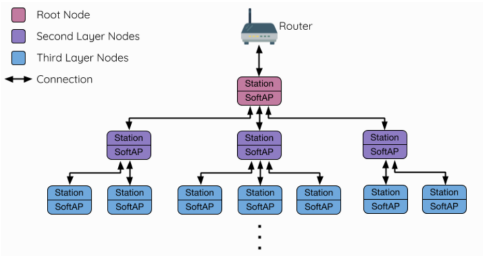

Figura 3.2: ESP-WIFI-MESH topología de árbol [17].

La que se utiliza en este trabajo, es una red de múltiples saltos (multi-hop), lo que significa que los nodos pueden transmitir paquetes a otros nodos en la red a través de uno o más saltos inalámbricos. Por lo tanto, los nodos en ESP-WIFI-MESH no solo transmiten sus propios paquetes, sino que simultáneamente sirven como repetidores para otros nodos. Siempre que exista un camino entre dos

nodos en la capa física (a través de uno o más saltos inalámbricos), cualquier par de nodos dentro de una red ESP-WIFI-MESH puede comunicarse.

El tamaño (número total de nodos) en una red ESP-WIFI-MESH depende del número máximo de capas permitidas en la red y del número máximo de conexiones descendentes que cada nodo puede tener. Ambas variables pueden configurarse para limitar el tamaño de la red.

Entrando en detalle en el rol específico de cada tipo de nodo se observa:

a) Nodo Raíz
El nodo raíz es el nodo superior en la red y sirve como la única interfaz entre la red ESP-WIFI-MESH y una red IP externa. El nodo raíz está conectado a un router Wi-Fi convencional y retransmite paquetes desde/hacia la red IP externa a los nodos dentro de la red ESP-WIFI-MESH. Solo puede haber un nodo raíz dentro de una red ESP-WIFI-MESH y la conexión ascendente del nodo raíz solo puede ser con el router

b) Nodo Hoja
Es un nodo que no tiene permitido tener nodos hijos (sin conexiones descendentes). Por lo tanto, solo puede transmitir o recibir sus propios paquetes, pero no puede reenviar los paquetes de otros. Si un nodo está situado en la capa máxima permitida de la red, se asignará como un nodo hoja. Esto impide que el nodo forme conexiones descendentes, asegurando que la red no añada una capa extra. Algunos nodos sin una interfaz softAP (solo estación) también se asignarán como nodos hoja debido a la necesidad de una interfaz softAP para cualquier conexión descendente.

c) Nodo Intermedio
Son aquellos conectados a la red que no son ni el nodo raíz ni un nodo hoja sino que son padres intermedios. Un intermedio debe tener una única conexión ascendente (un solo nodo padre), pero puede tener de cero a múltiples conexiones descendentes (de cero a múltiples nodos hijos). Por lo tanto, un nodo intermedio puede transmitir y recibir paquetes, pero también reenviar paquetes enviados desde sus conexiones ascendentes y descendentes.

d) Nodo Inactivo
Son aquellos que aún no se han unido a la red. Los nodos inactivos intentarán formar una conexión ascendente con un nodo padre intermedio o intentarán convertirse en el nodo raíz bajo las circunstancias correctas.

En la siguiente imagen se muestra el tipo de nodos disponibles en la red.

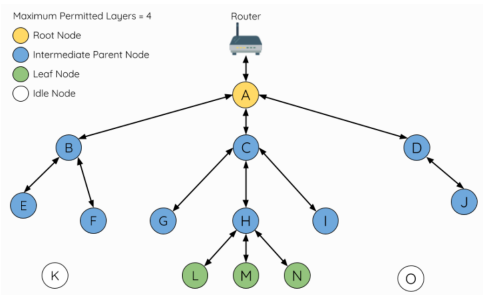

Figura 3.3: ESP-WIFI-MESH tipos de nodos. [17].

En la red MESH de Espressif Systems, el nodo raíz puede ser seleccionado automáticamente mediante un proceso de votación entre los distintos que formen la red. Quien tenga mejor RSSI (Received Signal Strength Indicator) con el router, se convertirá en nodo raíz y los demás se irán acoplando a la red con otros roles en función del RSSI. Otra forma de elección trata de que el programador directamente decida qué dispositivo quiere que actúe como raíz. Este es el caso utilizado en este sistema pues interesa que el ESP32-S3 que se conecte a la placa, actúe de raíz.

Otra de las ventajas de las redes malladas es que se puede autogestionar, es decir, si cualquier nodo por el motivo que sea, se desconecta de la red, los demás buscarán otro nodo para seguir con la conexión a la red.

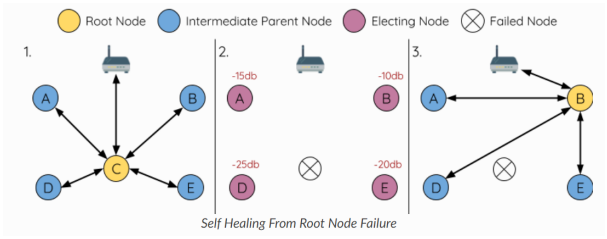

Figura 3.4: ESP-WIFI-MESH autogestionada. [17].

Las redes MESH pueden actuar en la transmisión de paquetes de dos formas diferenciadas. La primera sería cuando el único nodo que está conectado a Internet es el nodo raíz y por lo tanto, todos los demás nodos de la red, si quieren enviar un paquete al exterior, tendrán que mandárselo al nodo raíz para que este se encargue de redireccionarlo donde sea necesario. La segunda forma ocurre cuando todos los nodos de la red tienen acceso a Internet. De esta forma, no es necesario que los paquetes pasen a través del nodo raíz, todos podrán enviar directamente al exterior sus paquetes de forma independiente. Esta segunda forma, es la que se ha elegido en este sistema.

Enlazando lo anterior, cuando todos los nodos de la red MESH están conectados, se comportan como puntos de acceso y estaciones a la vez. Esto supone una gran ventaja porque significa que cualquier tipo de dispositivo con Wi-Fi es capaz de conectarse a cualquiera de estos nodos.

2. **Actualización por OTA**
   Los microcontroladores ESP32 cuentan con tecnologías Wi-Fi y Bluetooth integradas, lo que los convierte en una excelente opción para aplicaciones IoT debido a su versatilidad para crear sistemas completamente inalámbricos. Sin embargo, como sucede con cualquier microcontrolador, es necesario cargarle un programa para poder utilizarlo, y esta programación se realiza tradicionalmente mediante un cable. Esto significa que si en el futuro queremos actualizar la programación de ste sistema, se tendrá que estar físicamente cerca del dispositivo con un cable USB serial conectado a una computadora. Esto limita una de las principales ventajas del IoT, que es la capacidad de gestionar y usar el dispositivo de forma remota desde cualquier lugar.

   La funcionalidad OTA (Over-the-air) se refiere al proceso de actualizar firmware, software o configuraciones de un dispositivo de manera inalámbrica. Esta

tecnología es comúnmente utilizada en una variedad de dispositivos, incluyendo teléfonos móviles, routers, dispositivos IoT (Internet de las Cosas), y muchos otros dispositivos electrónicos que requieren actualizaciones periódicas.

Para poder agilizar la actualización de firmware en este sistema, se ha decidido implementar la funcionalidad de la OTA.

El sistema realizará peticiones periódicamente a un servidor creado con node.js donde se alojará el firmware. Hará una comparación de versiones. Si la que tiene el microcontrolador es inferior a la que recibe por parte del servidor, entenderá que es necesaria una actualización. En este momento hará una petición para obtener el nuevo código a actualizar.

OTA permite actualizaciones remotas de firmware, reduciendo costos y tiempo al evitar la necesidad de intervenciones físicas para mantener dispositivos actualizados, mejorando la eficiencia operativa y la experiencia del usuario.

3. **Accesibilidad a software libre IoT.**
Se comentó en el apartado de introducción que uno de los objetivos era poder gestionar toda la actividad del sistema desde aplicaciones IoT. Por ello, se utilizan distintas programas que además son software libre, por tanto cualquier persona tiene acceso a ellas y no necesitan pagar por sus servicios.

   Total.js Flow es el centro de todas ellas. Se encarga de gestionar todas las actividades del sistema de forma remota. Todas las peticiones que haya por parte del usuario o del sistema pasarán por aquí. Esto se debe a que aquí están creadas todas las reglas necesarias para la comunicación. Todas estas peticiones serán usando el protocolo HTTPS(Hypertext Transfer Protocol Secure) y MQTT(Message Queuing Telemetry Transport).

   En lo referido al protocolo HTTPS, se hacen peticiones GET y PUT para saber y cambiar el estado de la electroválvula si fuera necesario, de encendido a apagado y viceversa. También se utilizan para poder realizar actualizaciones vía OTA. Hay una regla creada para poder verificar la versión del sistema. Dependiendo del valor, el sistema interpretará si necesita una actualización o no.

   Por otro lado, el protocolo MQTT, uno de los más usados en IoT, servirá para poder publicar y suscribirse a distintos topics a los que se enviarán información de los sensores. El microcontrolador mandará los valores de sus sensores (publicar) a un topic creado en Total Flow y estará a la espera de recibir (suscribirse) por un topic, a qué estado debe poner la electroválvula.

   Sumado a lo anterior se dispone de un cliente de MQTT donde se puede publicar o suscribirse a un topic para recibir información en forma de chat.

   Complementario a lo anterior, un bot de Telegram es gestionado a partir de Total Flow. Desde este bot, el usuario podrá realizar distintas acciones como gestionar el estado de la electroválvula o ser alertado por algún tipo de evento.

   El usuario contará con una pequeña interfaz con dos botones para encender o apagar la electroválvula. Cuando toque uno de ellos, recibirá un mensaje por parte del bot, confirmando la acción que ha hecho.

   Además, el bot avisará al usuario en caso de que se produjera algún tipo de situación anómala en el sistema. Es decir, si los valores de las presiones en

el filtro o las mangueras, tuvieran una cierta diferencia, se sobreentiende que hay algún tipo de problema. Dependiendo el lugar donde ocurra, el sistema lo notificará para que el usuario sepa que debería actuar. Por último, el riego está diseñado para suministrar X cantidad de agua. En el momento que lo supere, se notificará al usuario para que tome acción.

A parte de tener reglas relacionadas con MQTT y HTTPS que van hacia Telegram o el propio servidor, Total Flow gestionará una base de datos con PostgreSQL. Cuando se envíen los datos por parte de los sensores por MQTT al servidor, serán redireccionados a la base de datos, donde se tiene una tabla con todos los valores de las presiones, del caudalímetro y de la electroválvula a lo largo del tiempo.

Esta base de datos será utilizada por Grafana. Aquí el usuario dispondrá de la interfaz para poder ver la actividad del sistema. Los datos se irán actualizando con los datos que van llegando a la base de datos. A parte de esto, en Grafana se podrá cerrar y abrir la electroválvula y de igual forma, se notificará por Telegram de esta acción.

Aquí se muestra una imagen de la plataforma que contiene todas las funcionalidades citadas:

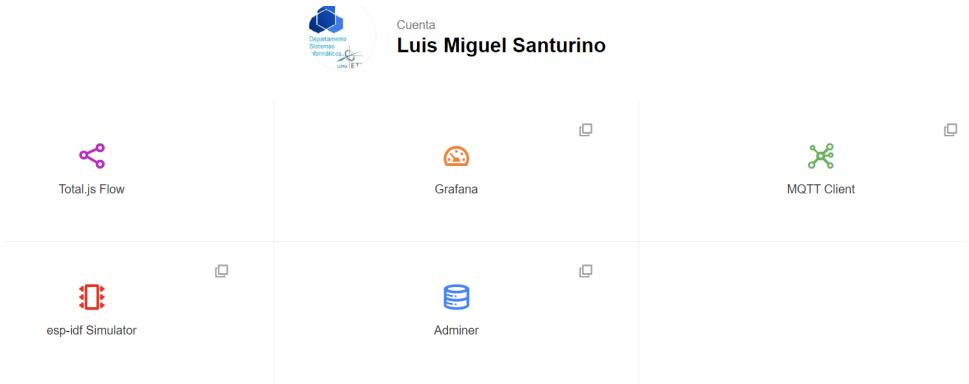

Figura 3.5: Plataforma IoT de desarrollo.

Se van a mostrar ahora ciertas ventajas que ofrecen estas aplicaciones en comparación con otras con funciones similares

a) Grafana vs Thingsboard
Grafana y ThingsBoard son dos herramientas populares para la visualización y gestión de datos en el ámbito de IoT y la monitorización de sistemas. Aunque ambos tienen sus propias fortalezas, Grafana se destaca en varias áreas clave que lo hacen superior en ciertos aspectos específicos.

Grafana es especialmente conocido por su capacidad de visualización avanzada. Ofrece una amplia variedad de opciones de visualización, como gráficos de líneas, barras, tablas y mapas de calor, que permiten a los usuarios crear paneles de control altamente personalizados y visualmente atractivos. En comparación, ThingsBoard también ofrece capacidades de visualización, pero la flexibilidad y la personalización de Grafana son generalmente consideradas más avanzadas.

La interfaz de usuario de Grafana es intuitiva y fácil de usar, lo que facilita la creación y gestión de dashboards incluso para aquellos sin experiencia técnica profunda. Mientras que ThingsBoard tiene una interfaz de usuario amigable, Grafana es a menudo preferido por su simplicidad y la eficiencia con la que los usuarios pueden diseñar sus paneles de control.

Una ventaja significativa de Grafana es su soporte para múltiples fuentes de datos. Es compatible con una amplia gama de bases de datos y servicios de monitorización como Prometheus, InfluxDB, Elasticsearch, MySQL y PostgreSQL, entre otros. Esta versatilidad permite a los usuarios integrar datos de diversas fuentes en un solo panel, proporcionando una visión completa y unificada del estado y rendimiento de sus sistemas. ThingsBoard, aunque compatible con varias fuentes de datos, no iguala la amplitud y profundidad de integraciones que Grafana ofrece.

Grafana también sobresale en la configuración de alertas. Los usuarios pueden establecer alertas personalizables basadas en umbrales y condiciones específicas, con la capacidad de enviar notificaciones a través de correo electrónico, Slack, PagerDuty, y otros servicios. Esta funcionalidad de alertas es altamente flexible y se puede ajustar para satisfacer necesidades específicas de monitorización. ThingsBoard ofrece funcionalidades de alertas, pero la flexibilidad y el nivel de personalización que Grafana proporciona son generalmente considerados más robustos.

La escalabilidad y el rendimiento son otras áreas donde Grafana se destaca. Está diseñado para manejar grandes volúmenes de datos de manera eficiente, lo que lo hace adecuado para entornos de gran escala donde el rendimiento es esencial. Aunque ThingsBoard también es escalable, Grafana es frecuentemente elegido por su capacidad para manejar cargas de trabajo intensivas de manera más eficaz.

Además, Grafana cuenta con una comunidad activa y una amplia gama de plugins que permiten extender su funcionalidad. Los usuarios pueden personalizar Grafana según sus necesidades específicas, añadiendo nuevas características y mejorando las capacidades existentes. ThingsBoard también tiene una comunidad y soporte de plugins, pero la cantidad y calidad de las extensiones disponibles para Grafana son generalmente superiores.

b) Total Flow vs Thingsboard
Total.js Flow es una herramienta de desarrollo visual basada en nodos, que permite a los usuarios diseñar flujos de datos y lógica de aplicación de una manera intuitiva y eficiente. En comparación con ThingsBoard, que también ofrece una interfaz de usuario amigable y capacidades de visualización, Total.js Flow destaca por su enfoque visual altamente intuitivo para la creación de aplicaciones IoT. Los usuarios pueden arrastrar y soltar nodos para construir flujos de datos complejos sin necesidad de escribir código, lo que facilita la rápida creación de prototipos y la implementación de soluciones IoT.

Una de las ventajas significativas de Total.js Flow es su flexibilidad en la integración de diversas fuentes de datos y servicios externos. La plataforma permite conectar fácilmente a diferentes APIs, bases de datos, y servicios

en la nube mediante nodos predefinidos, lo que simplifica enormemente el proceso de integración. Aunque ThingsBoard también soporta varias fuentes de datos, Total.js Flow ofrece una experiencia de integración más directa y adaptable gracias a su enfoque modular basado en nodos.

La personalización y extensibilidad son otras áreas donde Total.js Flow se destaca. Los usuarios pueden crear sus propios nodos personalizados para satisfacer necesidades específicas, lo que permite una gran adaptabilidad a diferentes casos de uso. Esto contrasta con ThingsBoard, que aunque es extensible, no ofrece el mismo nivel de facilidad y flexibilidad para la creación de componentes personalizados directamente en la interfaz de usuario.

Total.js Flow también es conocido por su rendimiento y capacidad de escalabilidad. Está diseñado para manejar grandes volúmenes de datos y flujos complejos de manera eficiente. La arquitectura de Total.js Flow permite que las aplicaciones se escalen fácilmente para satisfacer las demandas de grandes despliegues IoT. Aunque ThingsBoard es escalable y capaz de manejar grandes cantidades de datos, Total.js Flow es frecuentemente preferido en escenarios donde la capacidad de manejar flujos de datos complejos y personalizados es crucial.

En términos de comunidad y soporte, Total.js Flow cuenta con una comunidad activa y una amplia gama de recursos disponibles para ayudar a los desarrolladores. Los usuarios pueden acceder a documentación, foros y ejemplos para aprender a utilizar la plataforma de manera efectiva. Aunque ThingsBoard también tiene una comunidad y soporte sólido, Total.js Flow ofrece una experiencia más centrada en el desarrollador, con recursos específicamente diseñados para facilitar la creación y gestión de flujos de datos complejos.

Además, Total.js Flow ofrece una interfaz de usuario altamente personalizable, lo que permite a los desarrolladores ajustar la plataforma según sus necesidades y preferencias. Esta capacidad de personalización se extiende a la apariencia y funcionalidad de los paneles de control y las interfaces de usuario, proporcionando una mayor flexibilidad en comparación con ThingsBoard.

c) PostgreSQL vs MySQL
PostgreSQL presenta varias ventajas significativas en comparación con MySQL que lo hacen preferible en ciertos contextos. Una de las diferencias clave es su enfoque en el cumplimiento riguroso de los estándares SQL, lo que asegura que las aplicaciones desarrolladas en PostgreSQL sean más portables y menos propensas a problemas de interoperabilidad cuando se migra entre diferentes sistemas o se integran con otras tecnologías.

En términos de funcionalidades avanzadas, PostgreSQL soporta tipos de datos más complejos y específicos, como JSON, XML y tipos geométricos, de manera nativa. Esto simplifica el manejo de datos semi-estructurados y permite a los desarrolladores trabajar con datos en su forma más natural sin necesidad de conversiones o adaptaciones adicionales.

Otro aspecto destacado es la arquitectura MVCC (Multi-Version Concurrency Control) de PostgreSQL, que gestiona de manera eficiente la con-

currencia en bases de datos con múltiples usuarios realizando operaciones simultáneamente. Esta arquitectura minimiza los bloqueos y los conflictos, proporcionando un alto rendimiento incluso en entornos de alta concurrencia.

El optimizador de consultas de PostgreSQL es conocido por su capacidad para manejar consultas complejas de manera más eficiente que MySQL. Esto se traduce en tiempos de respuesta más rápidos y un mejor rendimiento general en operaciones de lectura y escritura, especialmente en bases de datos de gran tamaño y con esquemas complejos.

En cuanto a seguridad, PostgreSQL ofrece características avanzadas de autenticación, gestión de usuarios y control de acceso. Las opciones de cifrado y la capacidad de definir políticas de seguridad a nivel de columna permiten cumplir con estándares de seguridad exigentes, lo que lo convierte en una opción preferida para aplicaciones que manejan datos sensibles o críticos.

Capítulo 4

Desarrollo

Capítulo dedicado a describir el desarrollo del trabajo realizado. En él se van a explicar todos los pasos seguidos en la evolución de la parte hardware y software del sistema. Además se va a hablar de los consumos que ejerce cada componente y tecnología dentro del trabajo.

4.1. Desarrollo hardware

Esta sección mostrará todos los pasos de diseño hardware ejecutados para realizar la placa del sistema. También explicará con detalle cada componente utilizado y decisiones que se han tomado en consecuencia. Por último se mostrarán ciertas pruebas que se han hecho para comprobar el buen funcionamiento de ciertos componentes.

4.1.1. Requisitos hardware del sistema y elección de componentes

Comenzando por las decisiones tomadas respecto a las tensiones, se ha usado una fuente de alimentación de 24 voltios que es bastante común en el mundo de la agricultura usar voltajes como este. Además, para este sistema se ha considerado poder utilizar distintas electroválvulas. Se desea usar electroválvulas en el rango de 12 a 24V. Por estos motivos **la entrada es de 24 voltios con 1 amperio de corriente a través de una base jack DC macho** pues este sistema no va a demandar más allá de ese valor como se podrá observar en el apartado 4.3 de costes energéticos.

Aunque en el diseño electrónico de la placa que posteriormente se analizará no se tuvo en cuenta, se añadió al inicio de la PCB un **interruptor**, para poder encender y apagar el sistema de forma manual. El componente que se adquirió realmente era un switch de tres patillas, por tanto se cortó una de las que se encontraban en los extremos, para poder mantener en esa posición el circuito abierto.

Figura 4.1: Entrada jack DC macho para la alimentación.

Al tener una entrada de 24V, permite poder utilizar electroválvulas que se muevan entre los 12 y 24 V de tensión que necesitan para su alimentación. En este proyecto es de 12V. Debido a que se usa una electroválvula que necesita menor tensión que la que entra a la placa, se necesita reducir los 24V a 12V. Para ello se utiliza un **regulador de tensión L7812CV** que baja la tensión a 12V y permite una corriente a través de él que puede superar 1 A. Gracias a este componente, se puede alimentar adecuadamente a la electroválvula y al caudalímetro, pues también necesita 12V de alimentación. La tensión hasta el momento ha bajado de 24V a 12V, pero se siguen teniendo otros componentes como el relé, los sensores de presión y el microcontrolador que necesitan una alimentación inferior. Por tanto se coloca otro **regulador de tensión 7805** que hace que su tensión de salida sea de 5V. Esta tensión es la necesaria por todos los componentes anteriores para poder ser alimentados correctamente.

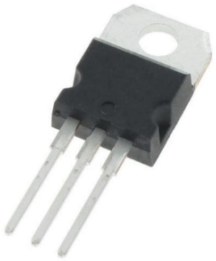

Figura 4.2: Regulador de tensión L7812CV.

Cada uno de estos reguladores está acompañado de dos condensadores dieléctricos, uno de entrada y otro de salida por varias razones importantes relacionadas con el rendimiento y la estabilidad del regulador. Aquí están las principales razones:

1. **Filtrado de Ruido (Desacoplamiento)**
   Condensador de entrada: Los reguladores de tensión lineales requieren un condensador de entrada para filtrar cualquier ruido de alta frecuencia presente en la fuente de alimentación de entrada. Esto ayuda a asegurar que la entrada al regulador sea lo más limpia posible, mejorando la estabilidad y el rendimiento del regulador.

   Condensador de salida: Un condensador de salida se utiliza para filtrar cualquier ruido o fluctuación de alta frecuencia que pueda generarse durante la regulación. Esto ayuda a mantener una salida de tensión más estable y sin ruido.

2. **Estabilidad del Regulador**
   Los reguladores de tensión pueden ser susceptibles a oscilaciones si no están adecuadamente estabilizados. Los condensadores ayudan a prevenir estas oscilaciones y asegurar que el regulador opere de manera estable bajo diferentes condiciones de carga.

3. **Respuesta Transitoria**
   Durante cambios rápidos en la carga (por ejemplo, cuando un dispositivo se enciende o apaga rápidamente), los condensadores ayudan a mantener la estabilidad de la tensión de salida. Sin los condensadores, el regulador puede no ser capaz de responder lo suficientemente rápido a estos cambios, resultando

en fluctuaciones de tensión.

4. **Reducción de Impedancia**
   Los condensadores ayudan a reducir la impedancia de la fuente de alimentación en las frecuencias altas. Esto es crucial porque el regulador puede tener dificultades para estabilizar una fuente de alta impedancia, lo que puede llevar a inestabilidad y ruido en la salida.

Para estos dos reguladores de tensión los valores de estos condensadores han sido iguales: en el de entrada de 0,33uF para filtrar el ruido de alta frecuencia y en el de salida de 0,1uF para mejorar la estabilidad y filtrar el ruido.

Es importante colocar los condensadores lo más cerca posible de los pines del regulador. Esto minimiza la inductancia y resistencia del trazado de la PCB que podría reducir la efectividad de los condensadores.

En resumen, los condensadores colocados cerca de los reguladores de tensión son esenciales para el filtrado de ruido, la estabilidad del regulador, la respuesta transitoria, y la reducción de impedancia. Estos componentes ayudan a asegurar que el regulador funcione de manera estable y eficiente bajo diversas condiciones operativas.

Figura 4.3: Condensador electrolítico de 0,1 uF.

Ya estabilizados los reguladores, una vez se empezaron las pruebas con la placa, se notó que ambos componentes estaban demasiado calientes. Para solucionar este problema, se instaló sobre ellos un **disipador de calor**. Estos componentes convierten la energía excedente en calor para mantener una salida de tensión constante. La diferencia entre la tensión de entrada y la tensión de salida, multiplicada por la corriente que pasa a través del regulador, se convierte en calor. Por ejemplo, si la entrada es de 20V y la salida es de 12V con una corriente de 1A, el regulador disipa (20V - 12V) * 1A = 8W como calor.

Los disipadores de calor están hechos de materiales como aluminio o cobre, que tienen alta conductividad térmica. Esto permite que el calor se transfiera rápidamente desde el regulador al disipador. Además estos componentes están diseñados con una especie de aletas para aumentar la superficie expuesta al aire. Esto mejora la transferencia de calor desde el disipador al aire circundante. Además, El calor se disipa al aire mediante convección, ya sea pasiva (movimiento natural del aire) o activa (uso de ventiladores para mejorar la circulación del aire).

Figura 4.4: Disipadores de calor.

Una vez reguladas todas las tensiones se han colocado distintos pines por la placa que constan de distintas tensiones y que mediante un **jumper** se elige la que se quiera utilizar. Esto se ha hecho de esta forma para no limitarse a una única forma de trabajo, dando flexibilidad para poder utilizar otros componentes que trabajen con otras tensiones. Es el ejemplo del relé. Se usa un relé de 5V pero se da la oportunidad de poder utilizar uno de 12V si se cambia de posición el jumper. Lo mismo ocurre con la electroválvula, consta de una pareja de pines donde puede elegir si alimentarse por 12 o 24V.

Figura 4.5: Jumper.

Otros **3 pines hembras** se encargan de poder conectar los 3 pines machos del relé. Uno para la alimentación que procede del jumper anteriormente comentado, el del medio para la tierra y el último que va hacia el microcontrolador para decidir si activar o desactivar la bobina. El otro lado del relé tendrá los cables pertenecientes a la electroválvula y el común irá a los 12 V de otros **3 pines** que se tienen para gestionar la electroválvula.

En adición a lo anterior, se colocan en total **44 pines hembra** (22 a cada lado) para incrustar el microcontrolador sobre ellos, evitando que se suelde contra la misma placa. De esta forma se evitan ciertos riesgos en caso de que el microcontrolador pereciese por culpa de algún tipo de corto en el sistema. De esta forma al estar por encima de estos pines, el microcontrolador tendrá que ser cambiado pero la placa no sufrirá daños adicionales.

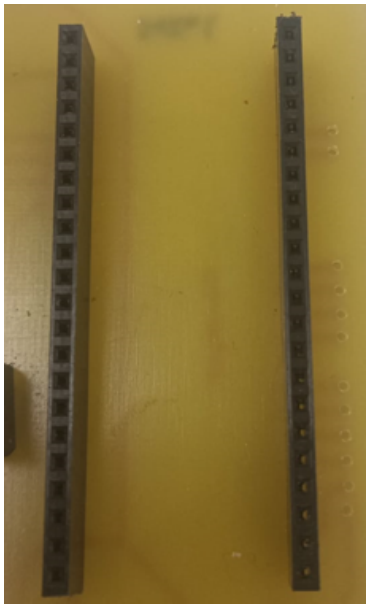

Figura 4.6: Pines donde va conectado el ESP32-S3.

Por otra parte, como dos de los sensores de presión y el caudalímetro van a ir conectados a la placa, se tienen 3 pines para cada dispositivo, es decir **9 pines** para gestionar estos dispositivos.

Asociados a cada uno de ellos, como los tres componentes tienen la misma tensión de salida que va de 4,5V a 0,5V que es superior a la que el microcontrolador acepta en sus pines (3,1V), se ha realizado un **divisor de tensión** en cada uno de ellos usando **resistencias de 1k07 y otra de 1k74 con 1 % de tolerancia** ambas. De esta forma se consigue que la entrada a los pines del microcontrolador sean en el rango de 0 a 3,1V que es lo que acepta como máximo.

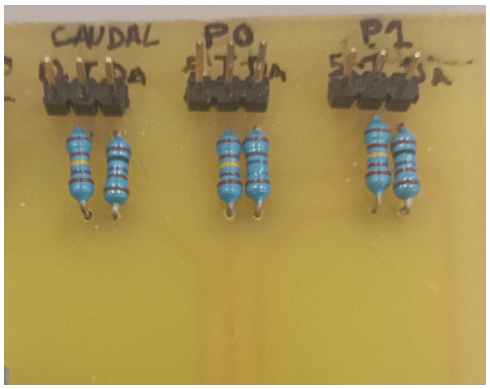

Figura 4.7: Divisores de tensión formados con las resistencias.

Por último, aunque no se trata de ningún componente, se ha tenido en cuenta posibles ampliaciones del sistema o bien algo que pudiera fallar, tuviera la oportunidad de trasladarse a otro pin del microcontrolador. Por ello, se hicieron pistas y se hicieron taladros en pines GPIOS (General Purposes IN/OUT) que no tuvieran ningún rol importante en la actividad interna del ESP32. Para poder saber cuáles usar y cuáles no, se utilizó el artículo de Luis LLamas [18].

Principalmente no pueden usarse los 8 pines relacionados con el bus de SPI. Se utilizan para comunicar el ESP32-S3 con la memoria flash externa, lo cual es crucial para operaciones de almacenamiento y recuperación de datos en aplicaciones embebidas. Estas son sus funciones:

1. SPICS0: chip Select para seleccionar la memoria flash SPI externa.
2. SPICS1: segundo Chip Select.
3. SPIHD: pin de Hold, utilizado para pausar las operaciones de la memoria flash.
4. SPIWP: pin de Write Protect utilizado para proteger la memoria flash contra escrituras no deseadas.
5. SPICLK: pin del reloj para la interfaz SPI.
6. SPIQ: pin de datos de entrada/salida.
7. SPID: pin de datos de entrada/salida.

Por otro lado están los dos pines para la conexión por USB

1. USB_D+: transporta la señal diferencial positiva.
2. USB_D-: transporta la señal diferencial negativa.

Los demás pines que no se pueden usar son por motivos de strapping. Es decir, tienen alguna función en el arranque del sistema para la correcta configuración del microcontrolador y no deben usarse para otro propósito.

En esta tabla se han recogido los pines en los que se han comentado los motivos por los que no deberían usarse y su porqué:

Pines que no deben usarse		
GPIO	Funciones	Motivo
0	RTC_GPIO0, GPIO0	STRAPPING Pulled-up
3	RTC_GPIO3, GPIO3, TOUCH3, ADC1_CH2	STRAPPING Floating
19	RTC_GPIO19, GPIO19, ADC2_CH8, USB_D-	USB_D-
20	RTC_GPIO20, GPIO20, ADC2_CH9, USB_D+	USB_D+
26	SPICS1, GPIO26	FLASH SPI
27	SPIHD, GPIO27	FLASH SPI
28	SPIWP, GPIO28	FLASH SPI
29	SPICS0, GPIO29	FLASH SPI
30	SPICLK, GPIO30	FLASH SPI
31	SPIQ, GPIO31	FLASH SPI
32	SPID, GPIO32	FLASH SPI
45	GPIO45	STRAPPING Pulled-down
46	GPIO46	STRAPPING Pulled-down
EN	CHIP_PU,RESET	STRAPPING Pulled-down

Cuadro 4.1: Pines no recomendados para su uso.

4.1.2. Diseño electrónico de la placa

Comentadas cada una las integraciones que tendrá la PCB, esta sección se encargará de explicar el diseño realizado a bajo nivel para elaborar la placa.

Comenzó usándose EasyEda [19]. Es una herramienta en línea que permite a los ingenieros, diseñadores y aficionados crear esquemas electrónicos y diseñar placas de circuito impreso (PCB). Pero, se descartó su desarrollo finalmente porque había máscaras de ciertos componentes que no eran posibles de encontrar.

De este modo, se optó por usar Kicad [20], que es un paquete de software libre para la automatización del diseño electrónico. Facilita el diseño de esquemáticos para circuitos electrónicos y su conversión a placas de circuitos impresos. Además, es un software con el que ya se estaba familiarizado pues se había utilizado en asignaturas de Ingeniería de Computadores para realizar el diseño de otras placas.

El diseño de placas electrónico consta de dos partes diferenciadas: el esquemático y el lógico.

4.1.2.1. Diseño esquemático con Kicad

Esta parte consta de los bloques a un mayor alto nivel que muestra la relación que debe haber entre los distintos componentes.

Se van a ir desglosando las distintas partes para ir comentándolas.

1. **Etapa de potencia**

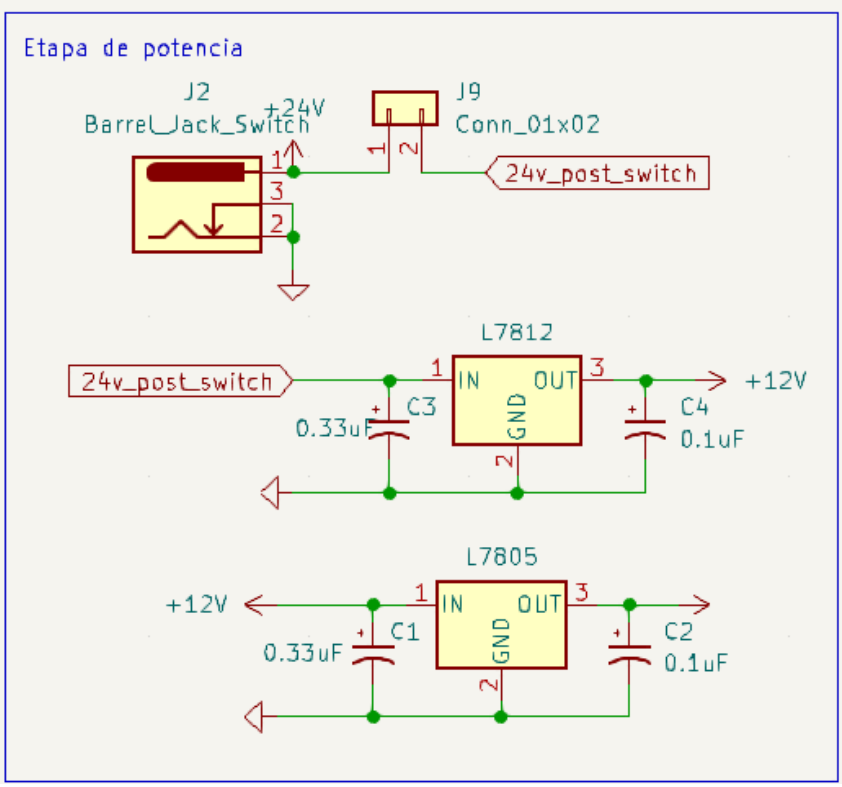

Figura 4.8: Etapa de potencia de la placa.

En la figura anterior se muestra la etapa de potencia de la PCB. Corresponde a la primera parte de la placa donde los reguladores de tensión entran en acción. La fuente de alimentación es la figura en la parte superior izquierda de la imagen. Se observa como tiene 3 pines, dos de ellos son tierras que realmente sirven para poder soldar el jack macho a la placa. El pin restante es donde se encuentra la tensión de 24V y que alimentará al circuito.

Kicad ofrece facilidades, una de ellas es el uso de etiquetas que relacionan un lugar con otro sin necesidad de conectar un cable ensuciando el diseño. Una de estas etiquetas es la llamada 24v_post_switch. Lo que quiere decir esta anotación es que la tensión que haya tras el interruptor, será la que haya de entrada en el siguiente bloque. Este bloque es el del regulador de tensión L7812CV.

En el bloque del regulador L7812CV, se tiene de entrada los 24V y son transformados en 12V de salida. Se muestran los dos condensadores que ya se comentó que servirían para regular la función del regulador.

De forma similar, en la parte inferior de la imagen se muestra ahora como entran 12V al regulador 7805 y aparecen los 5V de salida. Este regulador igual que el anterior, consta de sus dos condensadores.

En resumen, la etapa de potencia sirve para regular las tensiones y ya poder enviarlas a sus correspondientes dispositivos y alimentarlos.

2. **Pines con posible función en ampliaciones futuras**

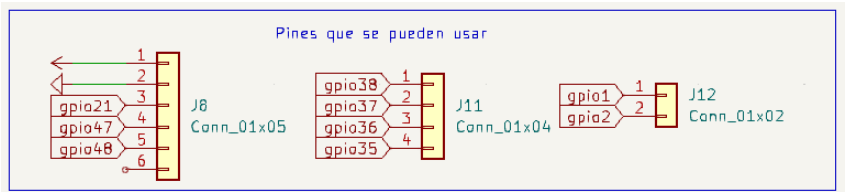

Figura 4.9: Pines con posible función en un futuro.

En apartados anteriores se comentó que había ciertos pines que no debían usarse pues tenían por defecto alguna función interna dentro del microcontrolador. Estos GPIOs son los que si pueden usarse pues no tienen ninguna función predeterminada. Se han diseñado para dejarlos como apoyo a posibles actualizaciones o roturas de otros GPIOs pero que en el sistema no tienen ninguna funcionalidad de momento.

Todos forman parte de pinouts que según el lugar donde se encuentren en el microcontrolador están unidos o no para evitar cruces de cables. Están referenciados con etiquetas para evitar lo que se dijo, dificultar el análisis del diseño.

3. **Sensores de presión**

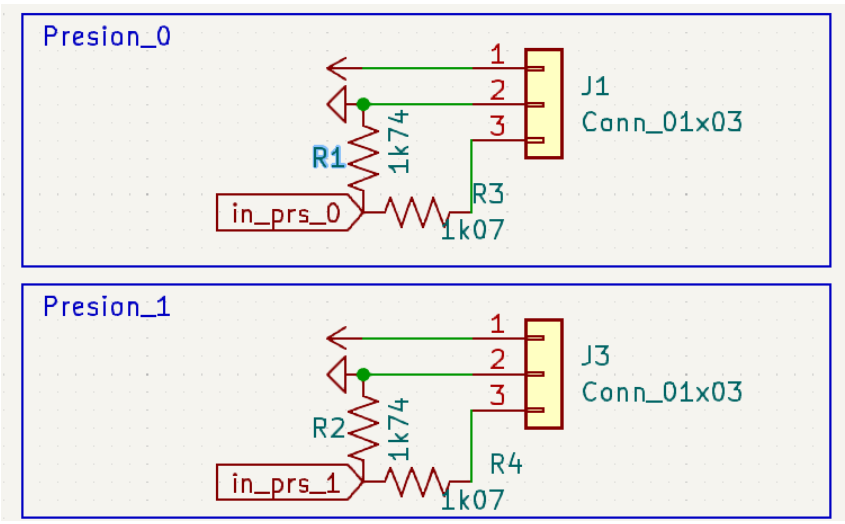

Figura 4.10: Bloques de los sensores de presión.

Estos son los bloques que contienen la forma en que los sensores de presión serán conectados a la placa, 3 pines para su alimentación a 5V, su tierra y el pin de datos que pasará a través del divisor de tensión formado por las resistencias y que irá la entrada del ADC del microcontrolador.

4. **Caudalímetro**

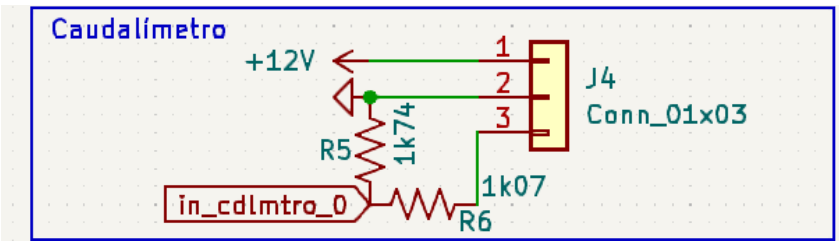

Figura 4.11: Bloque del caudalímetro.

El caudalímetro es exactamente igual que los bloques de los sensores de presión, lo único que es distinto en este componente es que necesita una alimentación de 12V.

Como se puede observar, los tres componentes tienen su pin de datos con una etiqueta que los identifica y que será enviada a su correspondiente pin en el microcontrolador.

5. **Electroválvula y relé**

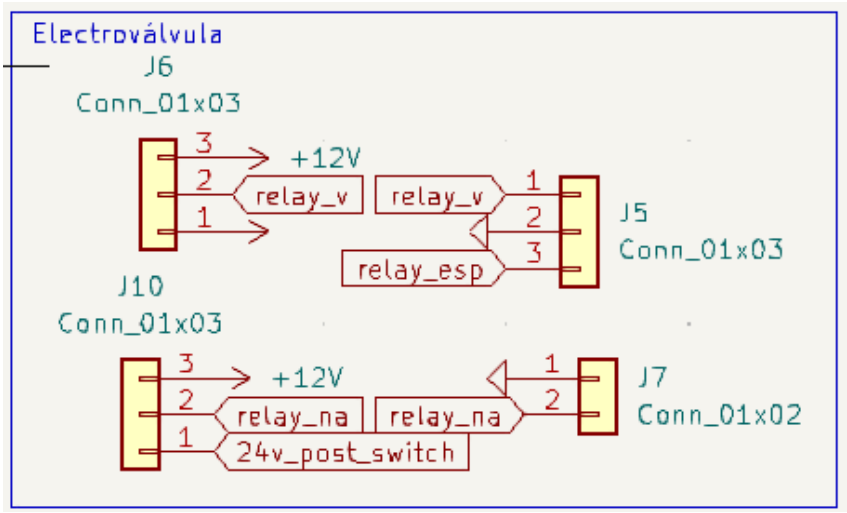

Figura 4.12: Bloque de la electroválvula y el relé.

Aquí se muestran primero los pines del relé. Los tres de la derecha corresponden a los pines hembra donde se meterá al relé y los de la izquierda servirán para poder elegir si queremos elegir una electroválvula de 12 o 5V para alimentar.

En la parte de abajo se muestran los pines pertenecientes a la electroválvula. La electroválvula necesita alimentación a 12V (la del proyecto), otro pin será el normalmente cerrado y el otro normalmente abierto.

Estos dos pinouts están relacionados entre ellos, pues la electroválvula necesita al relé para ser accionada. Entre los dos bloques habrá conexiones en común.

Por último se ilustrará el bloque del ESP32-S3. Este contiene todas las conexiones de los demás bloques referenciados mediante sus etiquetas.

Aparecerán muchos de los pines sin estar ocupados por etiquetas pues son GPIOS que no se utilizan en este proyecto.

6. **Microcontrolador ESP32-S3**

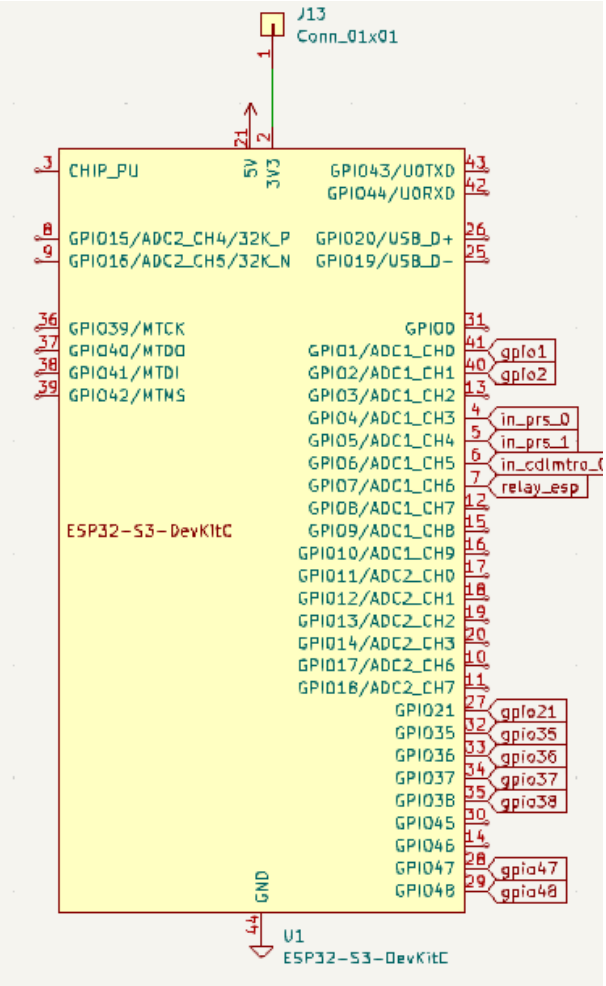

Figura 4.13: Bloque del ESP32-S3.

Como se puede observar, aparece el microcontrolador con todas las entradas que se han configurado en los otros bloques que se han estado explicando anteriormente junto las alimentaciones a 5V y 3,3V y Ground.

4.1.2.2. Diseño físico con Kicad

En esta sección se va a ilustrar y explicar el diseño físico realizado una vez el esquemático ha sido resuelto. Esta construcción comparada con la anterior está más cercana al bajo nivel. Aquí se usan las máscaras de cada uno de los componentes. Las máscaras son la apariencia que tiene las conexiones de los pines de los componentes de una forma más detallada. Es importante configurar bien este parámetro en cada elemento del sistema debido a que si se elige una máscara errónea puede obstaculizar el diseño físico de la PCB.

A continuación se muestra a gran escala el diseño físico de forma completa:

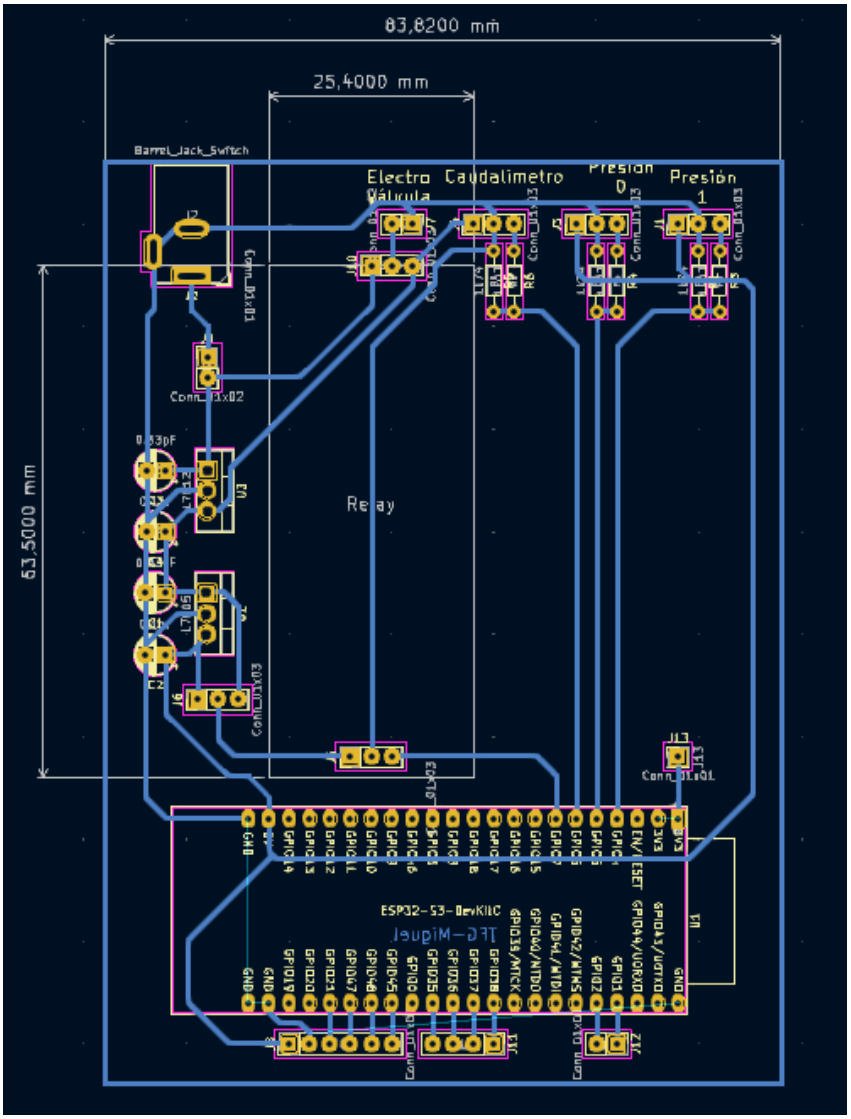

Figura 4.14: Diseño físico al completo del sistema.

A grandes rasgos es posible que no se entienda fácilmente el diseño. Por ello, se procede a explicar de una forma más detallada, ilustrando las distintas partes a las que se quiere hacer referencia.

Las líneas azules son las pistas de cobre que harán las comunicaciones entre los distintos pines. Tienen un ancho de 0,6mm. En las placas es posible grabar las pistas por su cara delantera o por su cara trasera. En este caso, se trata de la cara trasera. Si hubiera sido la cara frontal, las pistas serían de color rojo si se dejan los colores por defecto de Kicad. Este trabajo no necesitaba usar las dos caras porque se podía diseñar todo perfectamente en la trasera, reduciendo costes y trabajo. Se puede observar que ninguna de las pistas se cruza con otra si van a distintos destinos. Si ocurriera este suceso, la placa no estaría bien diseñada directamente.

Las líneas blancas son solo un recurso para tener información de las medidas de ciertos lugares de la placa. No serán grabadas en la placa pues Kicad ofrece algunas opciones para escribir sobre sus capas sin que estas sean parte de la placa a diseñar

en sí. En este caso se usa la opción User.Drawing.

Otro pequeño detalle que se puede observar es el siguiente: las pistas no tienen ángulos rectos. Evitar el diseño de ángulos rectos ayuda a que la corriente que pasa por las pistas pase de una manera más sencilla evitando que las pistas se debiliten.

A continuación, se van a distinguir dos partes principales en la placa y se explicarán de qué componentes se tratan.

1. **Etapa de potencia, relé y sensores**

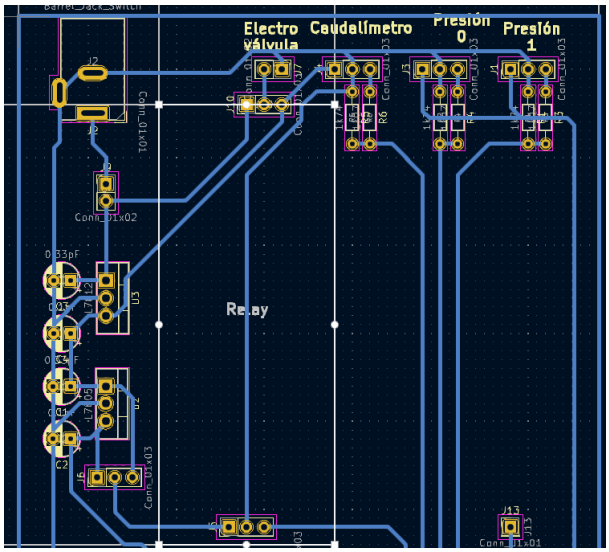

Figura 4.15: Diseño físico de la etapa de potencia, el relé y los sensores de la placa.

   En la parte izquierda de la imagen se tiene toda la etapa de potencia. El jack es el componente de la esquina superior izquierda. Se observa como uno de sus pines baja hacia el switch y los reguladores de tensión, mientras que los otros dos pines son tierras que servirán de unión para los demás componentes y de esa forma hacer tierra común entre todos.

   Al pasar el switch, se observan los dos reguladores de tensión, cada uno con sus dos condensadores. Una vez se ha pasado por estos componentes, la tensión dependiendo de si es 12V o 5V, va hacia distintos lugares. La parte inferior tiene todos los pines que se encargan de la alimentación del relé. En la parte derecha inferior se encuentra un pin que no se utiliza, pero que está por si se necesitara en alguna situación una alimentación de 3,3V por parte del microcontrolador.

   En la parte superior derecha se tienen todos los pines relacionados con la electroválvula, el caudalímetro y los sensores de presión. Estos tres últimos, acompañados de sus dos resistencias formando el ya mencionado, divisor de tensión.

2. **Microcontrolador ESP32-S3**

   Finalmente, se encuentra el microcontrolador con todas las entradas en la parte superior procedentes de los dispositivos del sistema, sumados a la alimentación de 5V y a la tierra. Todas las tierras de la placa van a parar a la del microcontrolador para que de esa forma sean comunes.

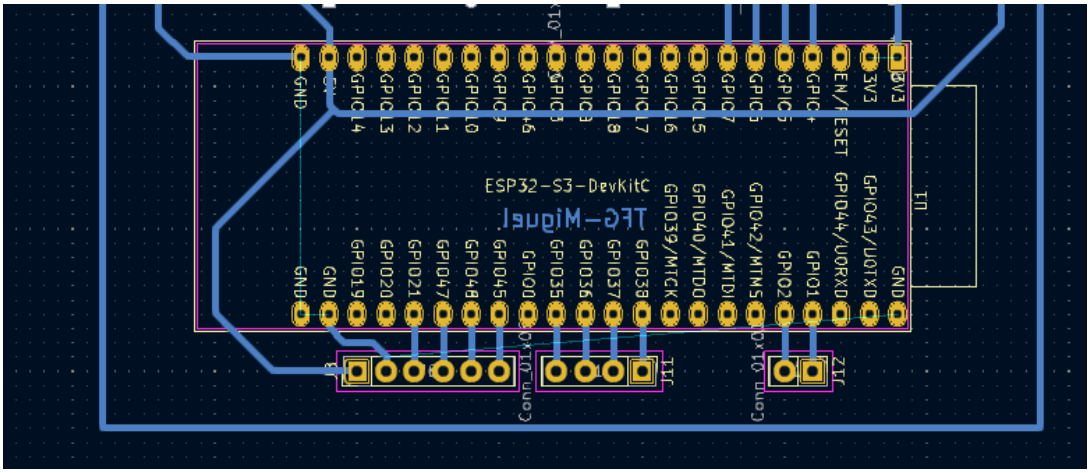

Figura 4.16: Diseño físico del microcontrolador.

Para finalizar, en la parte inferior se encuentran los pines disponibles para su uso que no tienen inicialmente una función en el Cabezal.

En esta imagen se muestra en 3 dimesiones como se verá la placa una vez se añadan los componentes.

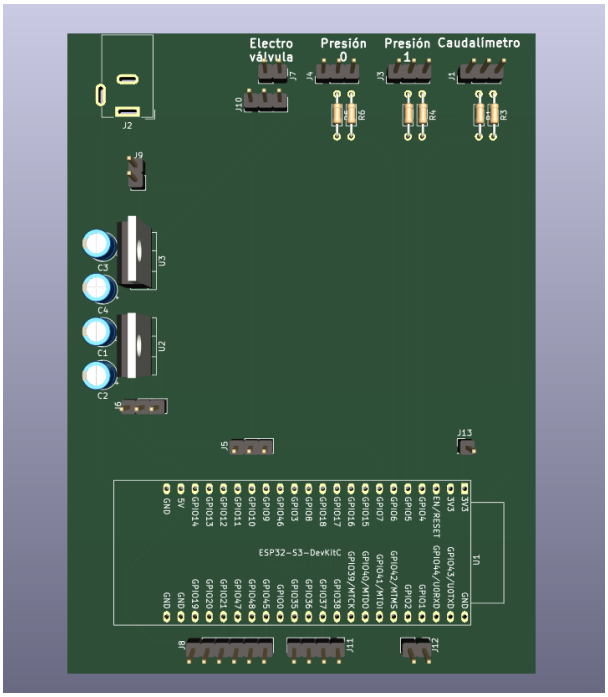

Figura 4.17: Diseño 3D de la placa.

4.1.3. Construcción de la placa

Resuelto el diseño electrónico, es el momento de construir la placa. Igual que antes, van a dividirse en puntos, los pasos seguidos para realizar la placa final.

1. **Impreso negativo** Para la construcción de la PCB es necesario disponer del circuito a realizar impreso en acetato.

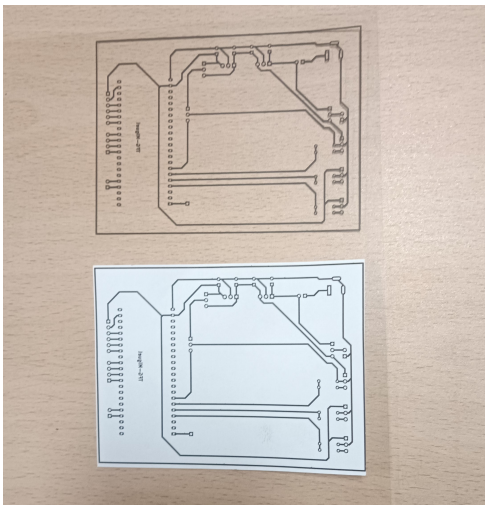

Figura 4.18: Circuito impreso en acetato.

Una vez se tiene el negativo, se quita la protección a la lámina que se va a utilizar para grabar el circuito.

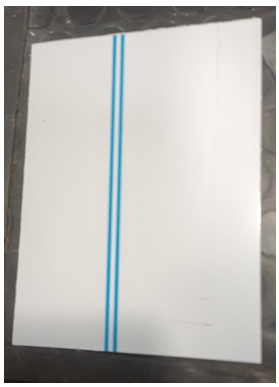

Figura 4.19: Lámina con protección.

Figura 4.20: Lámina sin protección.

Es recomendable una vez se ha quitado la protección, manipularla lo menos posible y hacer de forma rápida la insolación para evitar poder dañar las capas del material. En este momento se coloca el acetato sobre la lámina. Es recomendable dejar huecos para poder sujetar con celo el acetato.

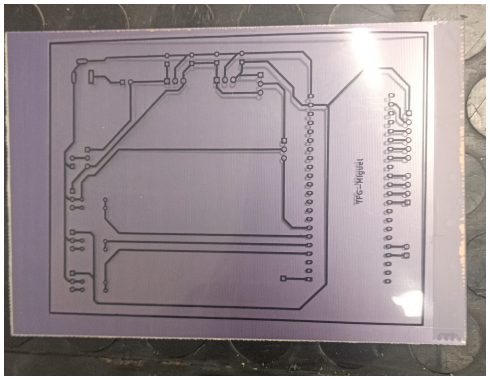

Figura 4.21: Impreso del circuito sobre el acetato.

2. **Insolación** En el proceso de insolación se somete a la placa a una emisión constante de luz ultravioleta. Se quita la protección de la placa con cuidado de que la luz no incida sobre ella. Se coloca cada una de las transparencias sobre la cara de la placa que le corresponda de tal manera que la resina reacciona sólo en las zonas donde no existe pista ya que la luz pasa a través del papel transparente.

   En estas imágenes se muestra lo que está ocurriendo:

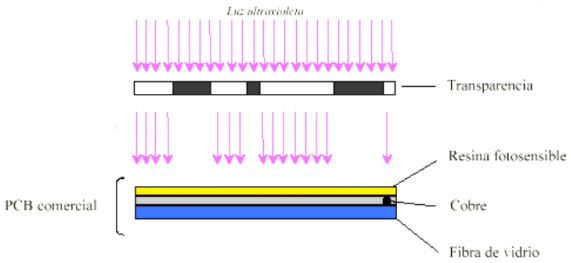

Figura 4.22: Emisión ultravioleta [21].

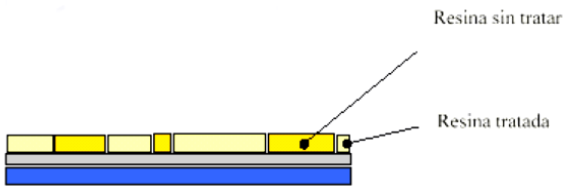

Figura 4.23: Final de la insolación [21].

Este proceso se lleva a cabo en la insoladora como se muestra en la imagen:

Figura 4.24: Proceso de insolación.

3. **Revelado** Consiste en eliminar de la placa la resina polimerizada. Para ello, se debe poner en contacto la placa con una sustancia química llamada revelador que en este caso ha sido hidróxido sódico. El hidróxido sódico elimina de la placa la resina tratada químicamente y deja al descubierto el cobre únicamente en aquellas zonas donde incidió la luz ultravioleta (donde no debe existir pista conductora)

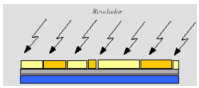

Figura 4.25: Revelador actuando sobre la placa [21].

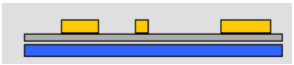

Figura 4.26: Resina eliminada de la placa [21].

Una vez se ha introducido la placa varias veces en este componente químico, se aclara con agua la PCB y se observan claramente las pistas.

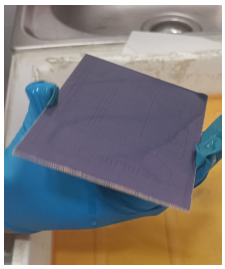

Figura 4.27: Placa sumergida en hidróxido sódico.

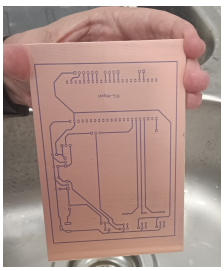

Figura 4.28: Placa tras el hidróxido sódico.

4. **Atacado** Lo que se ha obtenido hasta ahora es una placa donde las zonas en las que se desea que haya pista (negro) poseen el cobre protegido por la resina fotosensible. En aquellas regiones donde no se quiere que halla pista (blanco) el cobre está directamente expuesto al exterior. El objetivo del atacado es eliminar todo el cobre expuesto - no protegido por la resina -. El proceso se resuelve químicamente. Se ha utilizado una solución en agua de persulfato de amonio. Dicha solución, en contacto con el cobre reacciona con él de tal forma que éste se desprende depositándose en el fondo de la cubeta o recipiente que se utilice. Se podrá comprobar el desprendimiento pues el agua va tomando un color azulado.

   Esta imagen ilustra lo que ocurre en el atacado

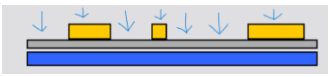

Figura 4.29: Comienzo del atacado.

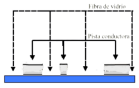

Figura 4.30: Fase de atacado [21].

A continuación se muestra el proceso de atacado y como el cobre sobrante va desapareciendo, cogiendo cada vez más y más un color amarillento la placa

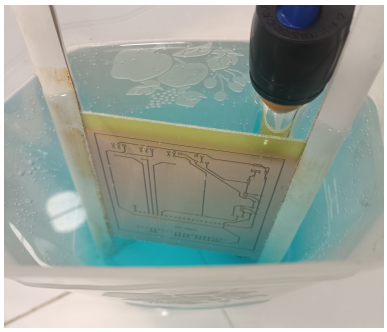

Figura 4.31: Placa al comienzo de su atacado.

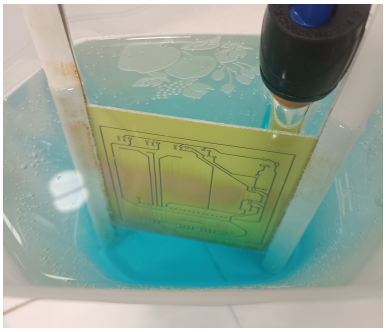

Figura 4.32: Placa terminando su atacado.

Una vez la placa acaba su fase de atacado se puede observar que ha perdido todo el cobre sobrante (se deben cortar las partes no necesarias) y que se ha quedado en perfecto estado para empezar a ser taladrada y soldada.

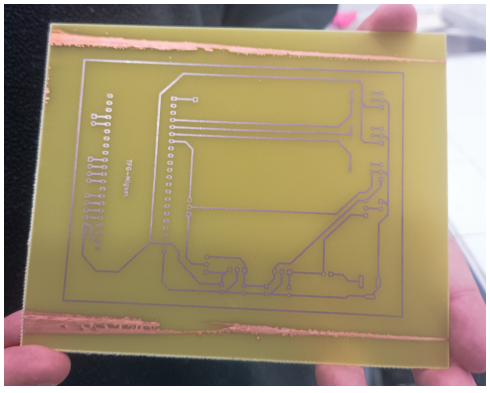

Figura 4.33: Proceso de atacado terminado.

5. **Corte y taladrado** En este momento la placa ya ha terminado todas las etapas en la que se necesitan tratar con químicos para eliminar ciertas capas. Ahora se cortan las partes que no interesan de la placa y se taladran cada uno de los pines con un calibre más pequeño o más grande, ajustándose al tamaño del pin del que se trate.

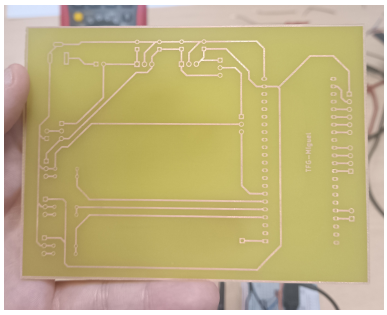

Figura 4.34: Placa una vez realizado los cortes de partes sobrantes.

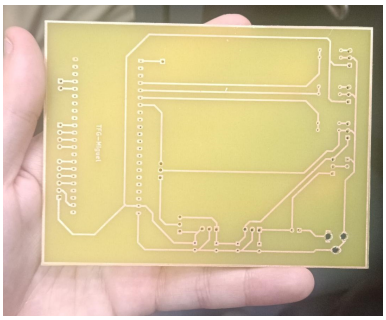

Figura 4.35: Placa taladrada.

6. **Continuidad** Se comprueba que todas las pistas tienen continuidad de extremo a extremo con el multímetro.

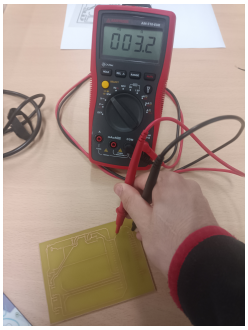

Figura 4.36: Comprobación de continuidad en las pistas.

7. **Soldar** Una vez se ha comprobado que todas las pistas están correctamente, se empiezan a soldar los componentes a la placa con la soldadora, utilizando estaño.

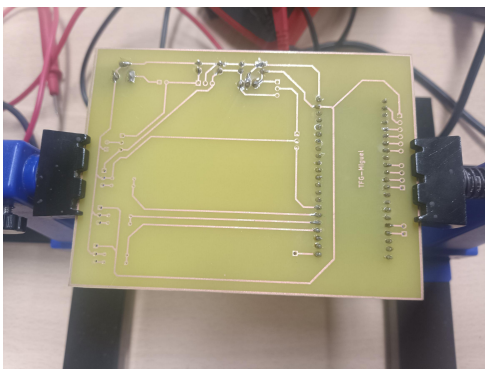

Figura 4.37: Proceso de soldado.

Finalmente, el proceso de la placa ha acabado. Este es el resultado de la construcción:

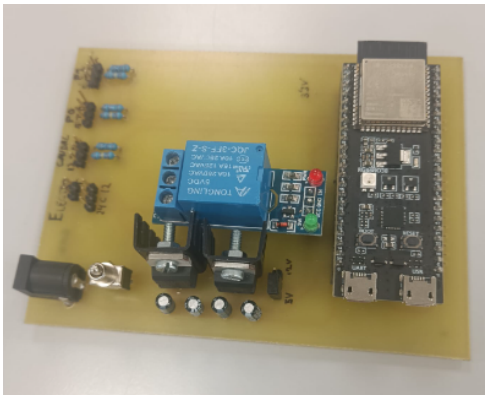

Figura 4.38: Placa final.

4.1.4. Componentes digitales y analógicos utilizados en el Cabezal

El objetivo de esta sección es explicar qué dispositivos se han utilizado, sus características y cuáles son sus funciones en el sistema.

El sistema está compuesto por una tubería de polietileno de baja densidad a la que se acoplan todos los componentes eléctricos y digitales. Es un material excelente para trabajar en riegos gracias a su resistencia a la humedad, aislamiento eléctrico y costes entre otras.

Figura 4.39: Tubería de polietileno de baja densidad [22].

Sobre esta tubería se apilan distintos componente. Los analógicos que servirán de refuerzo al sistema son los manómetros que miden hasta 4 bares y el contador de agua que va acumulando en su medida el agua que va circulando en litros.

Figura 4.40: Manómetro de 4 bares [23].

Figura 4.41: Contador de agua [24].

Por otro lado, están los dispositivos digitales que se unirán a la tubería y que se enumerarán a continuación:

1. **Electroválvula TORO EZ-FLO-EZP-02-54 [25]** Dispositivo electrónico que permite abrir y cerrar el cabezal permitiendo el flujo de agua. Para que la electroválvula funcione correctamente será necesario una fuente de alimentación de al menos 12V. La electroválvula cuenta con el relé de 5V el cuál se conectará al ESP32-s3 para que sea el propio ESP-32 quién abra y cierre la electroválvula a través del accionamiento de la bobina del relé.

Figura 4.42: Electroválvula del sistema [25].

2. **Sensor de presión - DC 5V G1 [26]** Dispositivo electrónico que es capaz de medir la presión del agua dentro del cabezal. En el cabezal de riego, al lado de cada sensor de presión digital, se tiene el manómetro comentado el cual es capaz de medir la presión en el rango de 0-4 bares. El sensor mide la presión del agua en el rango de 0-1.2MPa (12 bares) con un voltaje de salida de 0.5-4.5V, por tanto, se ha usado el conversor analógico digital para convertir ese voltaje en un valor con el que se pueda trabajar para poder realizar la conversión a bares. El sensor tiene 3 líneas de conexión. Una línea para datos, otra línea que se conecta a 5V y otra que se conecta a GND de la placa.

Figura 4.43: Sensor de presión del sistema [26].

3. **Caudalímetro-YF-S201[27]** El caudalímetro es un dispositivo electrónico que es capaz de medir el caudal que atraviesa el Cabezal. El caudalímetro utilizado tiene un rango de trabajo de 1-30L/min y una presión de trabajo <=1.75MPa. El caudalímetro cuenta con 3 líneas de conexión. Una línea para datos, otra línea que se conecta a 5V y otra que se conecta a GND de la placa. Su salida al igual que el sensor de presión, es de 0,5 a 4,5V

   Tanto los sensores de presión como el caudalímetro, tienen un divisor de presión para reducir la tensión de salida de su pin de datos a una tensión con la que se pueda trabajar adecuadamente en el ESP32-S3

Figura 4.44: Caudalímetro del sistema [27].

4.1.5. Pruebas con el conversor analógico digital

Para empezar, el ESP32-S3 tiene dos conversores analógicos digitales, el ADC1 y el ADC2. El 2 no funciona correctamente como se muestra en este texto de la página oficial de Espressif Systems : "Since the ADC2 module is also used by the Wi-Fi, reading operation of adc2_get_raw() may fail between esp_wifi_start() and esp_wifi_stop()"[28].

Por tanto se ha decidido usar el ADC1 solamente, calibrándolo mediante Curve Fitting de la librería adc_cali_scheme.h. Curve Fitting es un proceso de ajuste de una curva matemática a un conjunto de datos [29]. En el contexto de la calibración del ADC, esto implica ajustar una función a las lecturas del ADC para corregir errores sistemáticos y mejorar la precisión de las mediciones.

En adicción a lo anterior, se ha elegido la máxima atenuación que acepta el ESP32-S3 que es de 0V a 3,1V. De esta forma el ADC podrá medir valores que estén en el rango de esas tensiones.

Como los sensores de presión y el caudalímetro tienen una salida de 0,5V a 4,5V, se ha realizado el divisor de tensión para cada uno de ellos. Se han elegido los valores en las resistencias de 1k07 y 1k74 con un 1 % de tolerancia pues son las que más se aproximan en el mercado para poder conseguir transformar la tensión procedente de estos dispositivos suponiendo que es máxima, al máximo soportado por el ESP32-S3 que es de 3,1V.

Para comprobar el funcionamiento correcto del ADC con sus características y del divisor de tensión, se han realizado pruebas con la fuente de alimentación para poder

observar que exactamente, se obtienen los valores esperados por consola.

En esta imagen se muestra como con una entrada de 5V, el ADC muestra que obtiene aproximadamente 3,080V valor que se acerca al 3,1V que coge como tope. Esta diferencia es aceptable pues las resistencias tiene un 1 % de tolerancia y se entiende que siempre puede haber un poco de ruido en la señal.

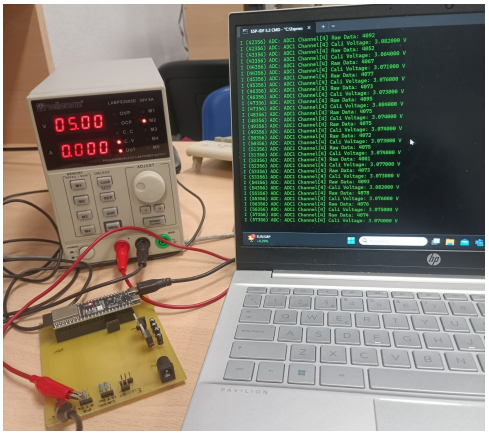

Figura 4.45: ADC con una entrada de 5V.

A continuación, bajando la tensión a la mitad, como es de esperar se obtiene la mitad en el ADC.

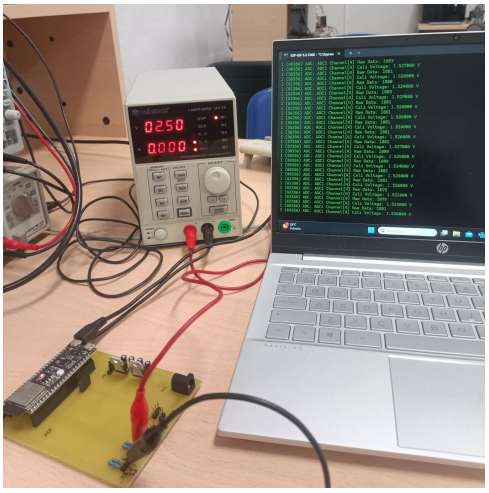

Figura 4.46: ADC con una entrada de 2,5V.

Por último, introduciendo 0V, la medida recogida por el ADC es de 0V.

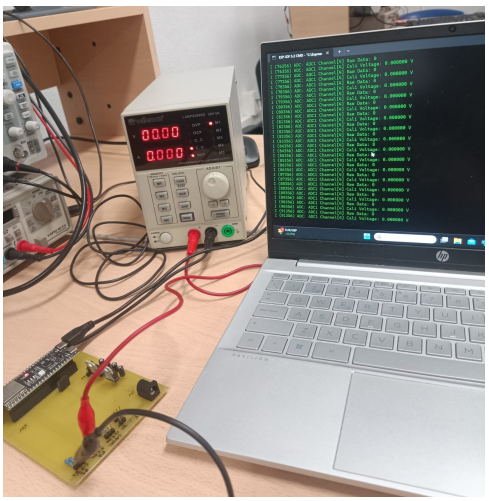

Figura 4.47: ADC con una entrada de 0V.

4.1.6. Incidencias hardware

1. Placa fallida en la fase de atacado.
   Se empezó a construir una primera placa que fracasó en el proceso de atacado. Las pistas fueron totalmente comidas por el persulfato de amonio y se tuvo que repetir la placa.

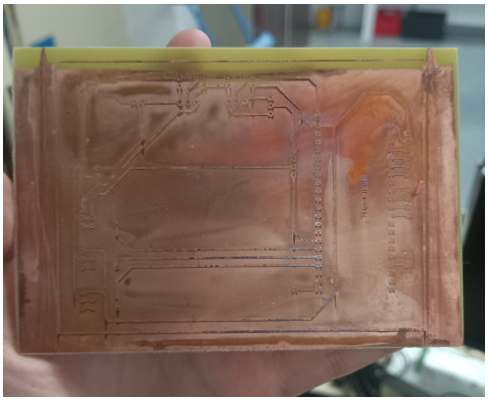

Figura 4.48: Placa fallida.

2. Introducción de disipadores.
   Como se comentó al comienzo del apartado de desarrollo hardware, los reguladores de tensión se sobrecalentaban y por este motivo se tuvieron que añadir disipadores de calor sobre ellos.

3. Electroválvula diferente a la esperada.
   La electroválvula que se pensaba tener para el proyecto, nunca llegó. Por tanto con la que se tiene ahora, de tres pines, cambia su forma de conexión a la placa. Por este motivo, los pines que se instalaron para gestionar los cables de

la electroválvula, han dejado de tener utilidad en el sistema de momento, si se usa la que está disponible actualmente.

4. Switch retirado.
   El switch que se instaló para abrir o cerrar el circuito justo después de la fuente de alimentación, no hacía buen contacto con la placa y en consecuencia la placa se encendía y apagaba de forma inesperada con simplemente rozar un poco la PCB. Debido a esto, se ha quitado el interruptor y se ha hecho un puente entre sus pines para que la corriente pase libremente.

4.2. Desarrollo software

4.2.1. Software libre utilizado

Para el desarrollo del código del microcontrolador se ha utilizado **Visual Studio Code.** En este entorno se puede descargar la extensión de **ESP-IDF 1.7.1** (Espressif IoT Development Framework) que es la herramienta que proporciona Espressif para desarrollar código en el chip ESP32 y todas las gamas derivadas de esta. Antes de

Figura 4.49: Logo de Visual Studio Code [30].

Figura 4.50: Logo de Espressif [31].

poder usar este entorno es necesario descargar varias programas para que funcione correctamente [32]:

1. Python 3.7 o superior.
2. Git.
3. Build Tools - Cmake y Ninja para compilar y montar el proyecto.
4. Microsoft C++ Build Tools build la aplicación en el ESP32.

Con estas herramientas se puede diseñar código en el ESP32-S3. Aquí se muestra una imagen de la página de Espressif [33] que ilustra para qué se necesita cada programa.

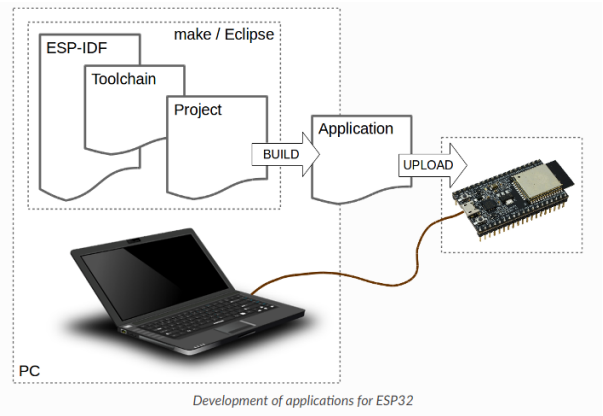

Figura 4.51: Herramientas necesarias para el desarrollo de código de ESP32-S3 [33].

Visual Studio Code da facilidad para programar código de forma sencilla, pudiendo ir a otras librerías rápidamente y dando atajos para el desarrollo. Sin embargo, se ha instalado **la versión 5.2 de la terminal de ESP-IDF.** No es necesaria teniendo la extensión en Visual Studio Code, sin embargo, por comodidad de los comandos se ejecuta con este CMD:

1. idf.py build se compila el código.
2. idf.py flash se flashea en la memoria del ESP32-S3.
3. idf.py build se puede ver la salida en la terminal de la ejecución del programa.
4. idf.py build flash monitor hace todo a la vez.

Para el desarrollo del servidor local, que alberga el binario para actualizar por OTA, se ha utilizado **node.js** y se han instalado los paquetes necesarios para poder albergar el servidor.

Figura 4.52: Software node.js [34].

Para la creación de reglas y comunicaciones del microcontrolador con los demás programas se ha usado **Total Flow.** En el interior de este, se encuentran varias partes diferenciadas: reglas REST API, MQTT, base de datos, bot de Telegram y Grafana.

1. **Reglas REST API**
   El microcontrolador es capaz de realizar una petición PUT al servidor mandando un json con los datos de los distintos dispositivos que conforman el sistema usando el protocolo HTTPS. Además, puede obtener el valor de los dispositivos de igual forma con una petición GET.

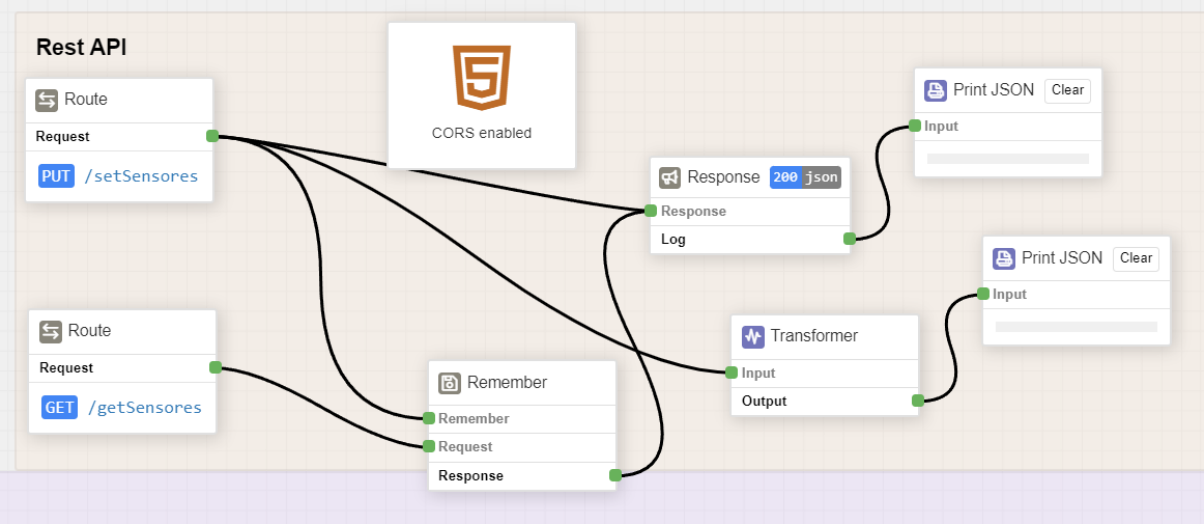

Figura 4.53: REST API reglas en Total Flow.

En esta imagen se ilustra que hay dos componentes llamados ROUTE que registran una ruta y luego envían datos de solicitud. El ESP32-S3 coge la ruta de estos componentes y se realiza la petición que corresponda. Aquí se muestra la función con la que haría el GET el microcontrolador:

```
void get_rest_sensors()
{
    esp_http_client_config_t config_get = {
        .url = "https://xx/xx/getSensores",
        .method = HTTP_METHOD_GET,
        .event_handler = client_event_get_handler
                };

    esp_http_client_handle_t client = esp_http_client_init(&config_get);
    esp_http_client_perform(client);
    esp_http_client_cleanup(client);
}
```

Se crea el struct config_get con la url, el tipo de petición y el manejador de eventos. Después se establece la conexión HTTPS y se recibe el dato. Finalmente se cierra la conexión entre servidor y cliente.

Sumados a los componentes ROUTE, se encuentra el response 200 (todo bien). Servirá para dar una respuesta al cliente por parte del servidor siempre que se haga una petición a este. El componente remember cuando recibe el output del método PUT, lo guarda y lo dispara cuando se hace un GET.

A través del método PUT se obtiene un json, con el transformer, se elige lo que se desea obtener o bien se hace un casting de los datos. Los print como es de esperar, sirve para verificar las salidas de los componentes que van hacia su entrada.

Por último CORS habilitado: El componente permite el uso compartido de recursos entre orígenes CORS, de modo que el navegador web podrá comunicarse directamente con la API REST. La funcionalidad sólo funcionará con un punto final Proxy definido para este flujo.

2. **Reglas para el accionamiento de la electroválvula.**
Estas reglas se encargan de pasar del estado ON a OFF o al revés de la electroválvula. Se comenzó usando el componente Trigger. De esta forma simplemente con darle un click desde Total Flow, se envía (publica) un true o un false hacia el topic "xx/xx/electrovalvula".

   Para que esto tenga efecto es necesario que se tenga un Broker MQTT habilitado que haga de intermediario entre el servidor y el microcontrolador. Por lo que se dispone de un componente que hace de Broker directamente. Su configuración es simplemente la url con el puerto que usa.

   Se trata de este componente:

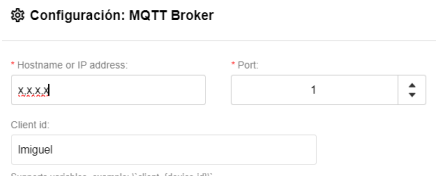

Figura 4.54: Configuración del componente MQTT Broker.

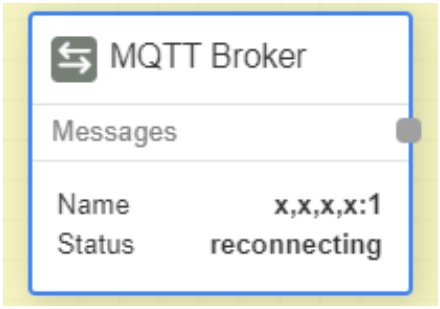

Figura 4.55: Componente MQTT Broker.

Una vez configurado el Broker y estando habilitado, es necesario tener un sitio donde publicar los mensajes. Por ello se tiene el componente Publish. Se debe configurar con la url del Broker.

Figura 4.56: Configuración del componente MQTT Publish.

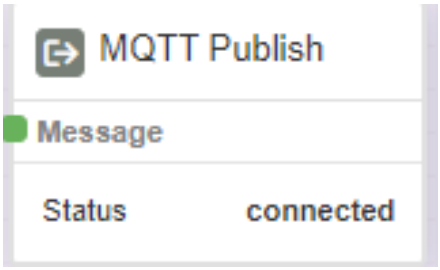

Figura 4.57: Componente MQTT Publish.

Una vez se tiene el Broker habilitado y el Publish conectado a este, se podrán publicar a los topics de los que se disponga. Se ha comentado que el topic encargado del comportamiento de la electroválvula es "xx/xx/electrovalvula.

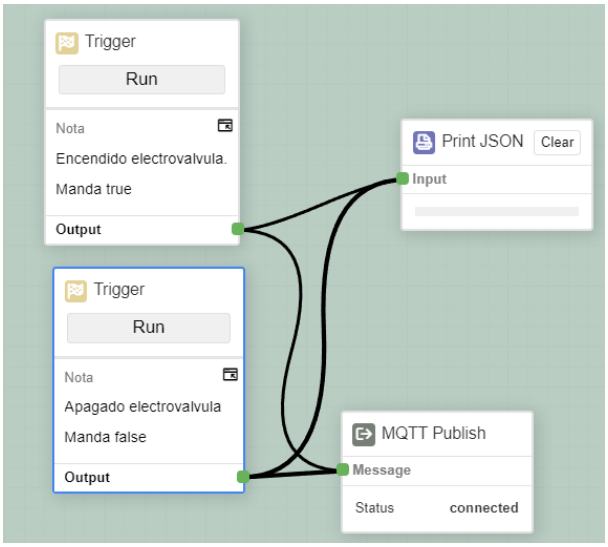

Figura 4.58: Reglas de la electroválvula con los Triggers en Total Flow.

Cuando se pulse al Run del Trigger, se enviará un true o un false al topic que recibirá la electroválvula y actuara según la orden.

Al ir avanzando en el proyecto, se consiguió una nueva funcionalidad. Poder enviar el true o false desde Grafana. Esto se hace a través de un PUT. Cuando se pulse un botón que se ha diseñado en Grafana que se verá más adelante, dependiendo de si es el de ON u OFF, se enviará un true o false. Para poder realizar esta función se usa un ROUTE con su correspondiente Response del servidor y su salida va hacia el Publish de forma similar a los Triggers.

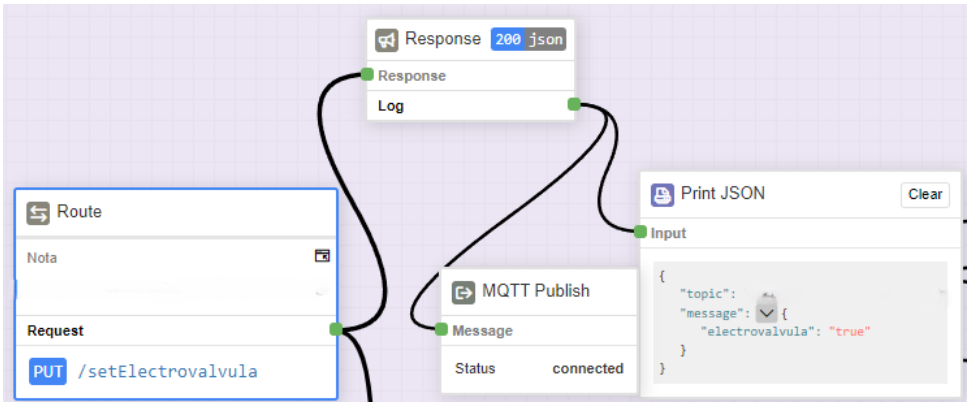

Figura 4.59: Regla de la electroválvula ejecutada desde Grafana en Total Flow.

3. Reglas para obtener valores de los sensores
   El microcontrolador al obtener los valores procedentes de los sensores, los publica en un topic. Para las pruebas se decidió hacer un topic para cada sensor y así ver claramente el valor de cada uno de ellos. En los resultados se verá que luego se unen todos los valores en un mismo topic. Para poder obtener los

valores que publica el ESP32-S3, hay que suscribirse a los topics que sean de interés. Aparece un nuevo elemento, el Subscribe.

Para su configuración se debe elegir el Broker que ya se tenía habilitado y el topic con el que trabajará.

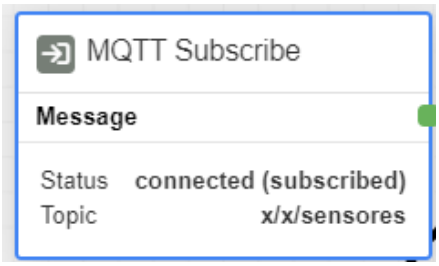

Figura 4.60: Configuración del componente MQTT Subscribe.

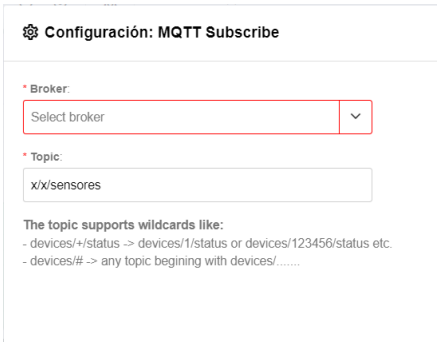

Figura 4.61: Componente MQTT Subscribe.

En la imagen que viene a continuación se ven tres Subscribe para poder tener el valor de cada uno de los sensores.

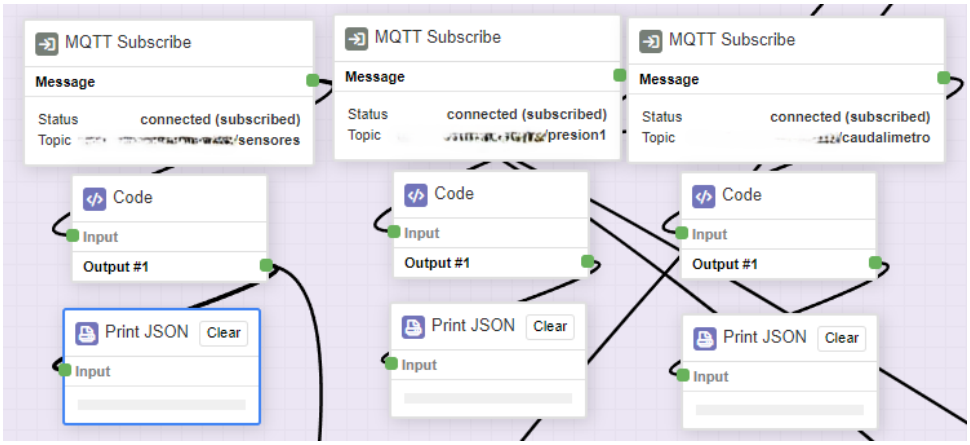

Figura 4.62: Suscripción a tres topics de los sensores.

4. Reglas para la base de datos de PostgreqSQL.
   Para la implementación de la base de datos es necesario tener un componente llamado QueryBuilder PG que es la API de Postgreq. Dentro de este componente se configura el número de conexiones simultáneas, la url de la base de datos y el nombre que tendrá.

Figura 4.63: Configuración del componente QueryBuilder PG.

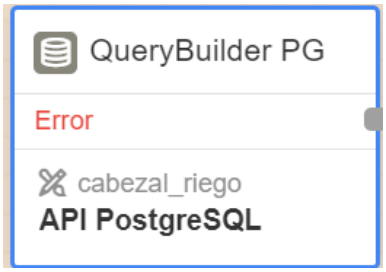

Figura 4.64: Componente QueryBuilder PG.

Una vez se ha establecido la base de datos que se va a usar, se añade un nuevo componente. Este se encargará de introducir con la consulta correspondiente, datos en la BBDD (INSERT INTO) . En este caso se encarga de meter en la tabla los valores de los sensores. Además, se harán consultas para obtener datos de la base de datos y tenerlos en distintos lugares (SELECT FROM). El nombre de este recurso es SQL Query. Su configuración es la consulta que se quiere realizar.

Por ejemplo, el SQL query que coge datos de la BBDD tiene:

```
SELECT LAST(measurement, time)
FROM device;
```

El SQL query que inserta datos en la BBDD:

```
INSERT INTO device (measurement)
VALUES ({data});
```

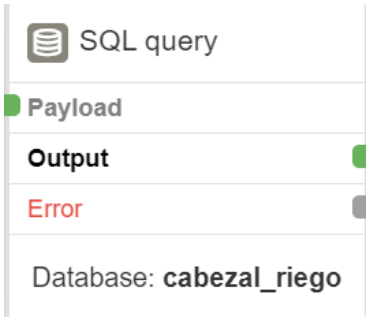

Figura 4.65: Componente SQL query.

Explicadas las funciones principales de los dos SQL queries que se van a usar, esta imagen muestra como se ha recibido en uno de ellos los valores de los sensores, procedentes de las reglas MQTT. Al pulsar el trigger, se recoge la última consulta recibida a la BBDD que corresponde a la que había sido introducida por el otro SQL query.

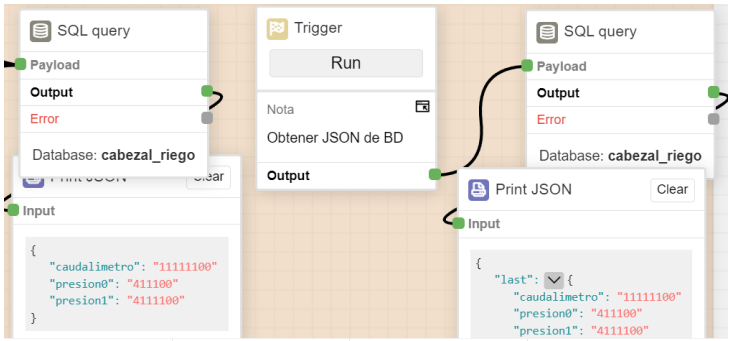

Figura 4.66: SQL queries funcionando.

La base de datos PostgreSQL es gestionada desde Adminer. Está formada por una tabla que contiene de atributos: el id único, la fecha de tipo timestamp y el valor de los sensores que se trata de un json. Aquí se muestra una imagen de la base de datos.

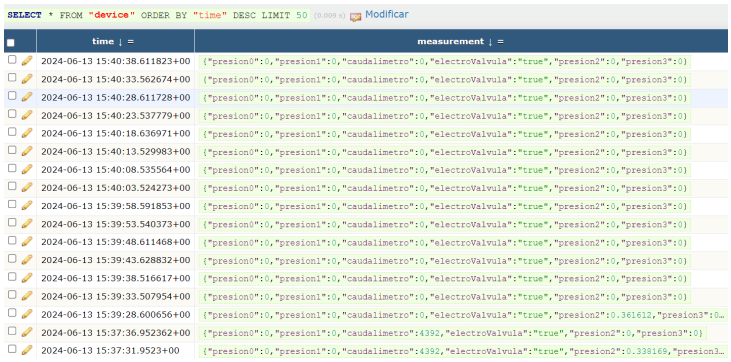

Figura 4.67: Registro en la base de datos.

5. Reglas de Telegram
   Se tienen dos componentes principales: el Receiver y el Sender. Cuando se recibe algo por parte de Telegram, se trata del Receiver. Por ejemplo: el accionamiento por comando de encender la electroválvula. Por otro lado se encuentra el Sender. Cuando haya un problema de presiones, de cantidad de agua o se haya encendido o apagado la electroválvula, se mandará una notificación al bot de Telegram. Estas acciones las lleva a cabo el Sender.

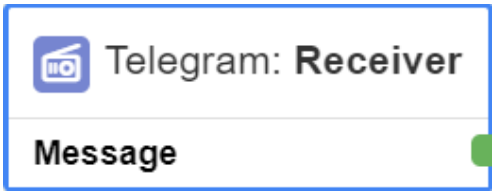

Figura 4.68: Componente Receiver.

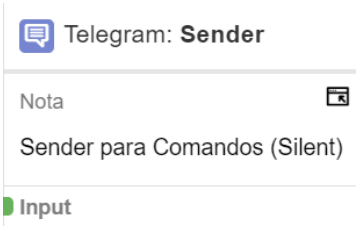

Figura 4.69: Componente Sender.

La configuración interna de cada uno de ellos es el Token del bot y el id del chat. Para tener estos datos ha sido necesario primero crear en Telegram un bot hablando con BotFather y después obtener el id del chat. Estos datos te los

ofrece fácilmente BotFather cuando estás creando tu bot.

El componente Receiver recibirá flujo cuando se mande algún tipo de comando desde el chat del bot. Se usa el comando switch para filtrar cada posible entrada del Receiver, es decir, dependiendo el comando que llegue del bot, habrá que hacer una acción u otra. El switch se encargará de mandar por un camino u otro a los comandos. Su funcionamiento interno es como el de un if-else.

El primer switch se encarga de redireccionar los comandos que empiecen por / al segundo switch. Este último, dependiendo del comando que se trate: /start, /encenderEv, /apagarEv o /help, mandará su salida a la entrada de un componente Code. En este componente básicamente se realiza código Javascript.

Aquí se muestra una imagen de lo comentado.

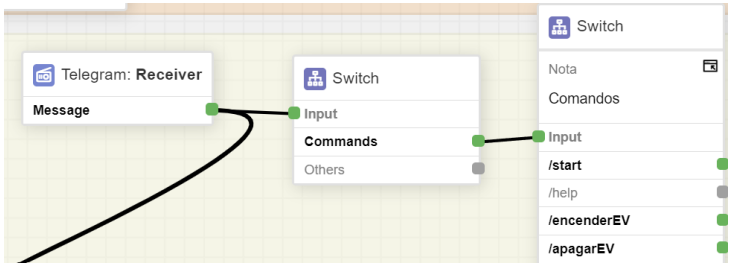

Figura 4.70: Filtrado de los comandos provenientes de Telegram.

Dentro de los Code se encuentra un método send que enviará notificaciones al bot de Telegram a través del Sender. En el caso de los comandos relacionados con la electroválvula, mandarán a su correspondiente topic, el dato de true o false por MQTT para que se accione.

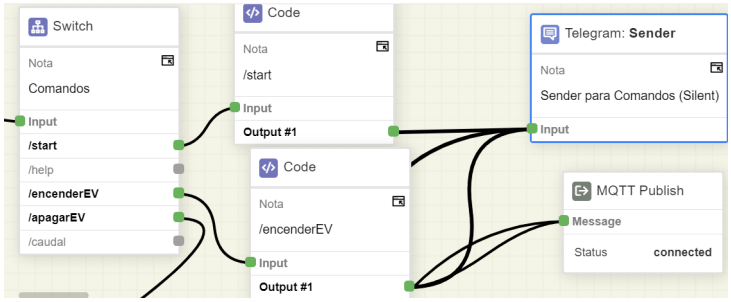

Figura 4.71: Envío de notificaciones a Telegram por Sender.

Como se comentó e ilustró en páginas anteriores, desde Grafana también se enciende y apaga la electroválvula. Lo hacía a partir de un método PUT donde su payload se dirige hacia otro de los Sender de Telegram para que el usuario también reciba un mensaje en caso de que se abra o cierre desde Grafana. A parte de esto, se tiene programado un Code que compare las presiones y el caudal suministrado para que se mande otra notificación al grupo.

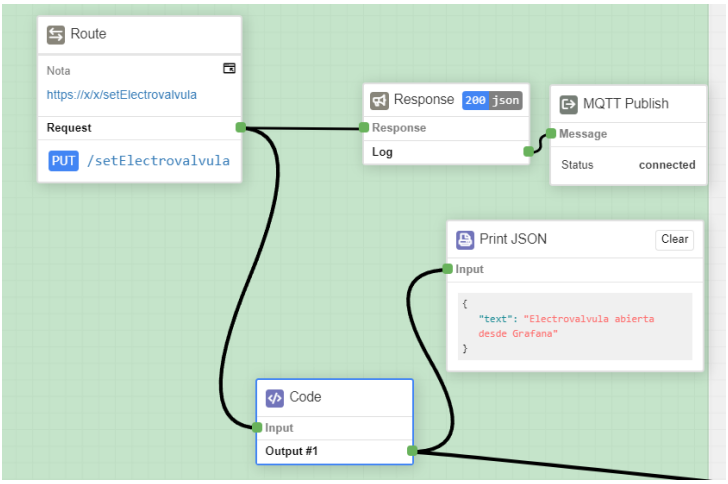

Figura 4.72: Envío de notificaciones a Telegram por Sender de la electroválvula desde Grafana.

El bot de Telegram se ve de la siguiente manera cuando se acciona la electroválvula desde Grafana o Telegram:

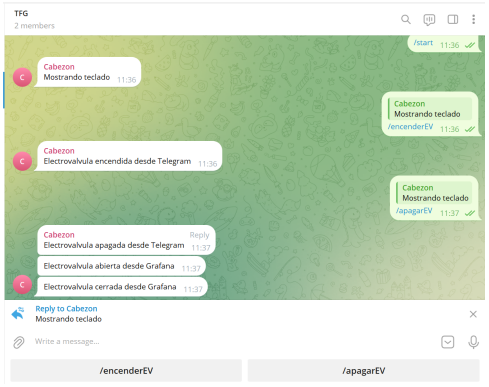

Figura 4.73: Chat de Telegram usando comandos de la electroválvula.

Si hay problemas con las presiones o el caudal se reciben estas notificaciones:

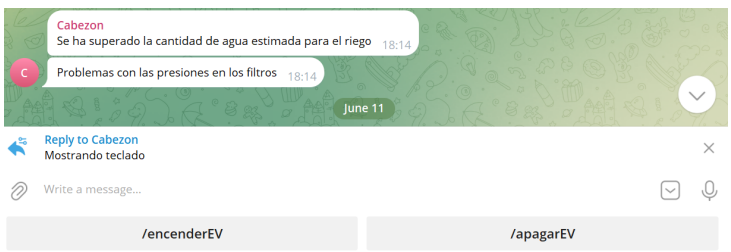

Figura 4.74: Chat de Telegram viendo notificaciones de los sensores.

Otro programa utilizado para ir viendo los datos que se iban transmitiendo con el protocolo MQTT ha sido **el cliente de MQTT de emqx** [35]. Con este software , se crea una conexión con el Broker y después se elige si se quiere publicar o estar suscrito a algún topic que esté disponible.

En esta imagen se muestra como al estar suscrito al topic xx/xx/get, se reciben mensajes como si fuera un chat parecido a Whatsapp.

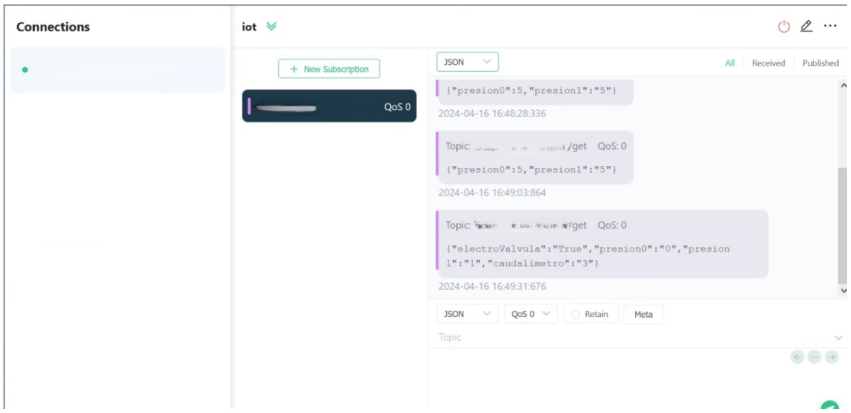

Figura 4.75: Cliente MQTT.

Por último, se ha utilizado **Grafana**. Desde aquí el usuario gestiona todo el sistema. Para ello se dispone de un dashboard. En este se pueden ver las presiones y una gráfica que compara los valores de las dos. Además se observa el cabezal entero con cada uno de los puntos de interés con la medida que ha recogido por última vez.

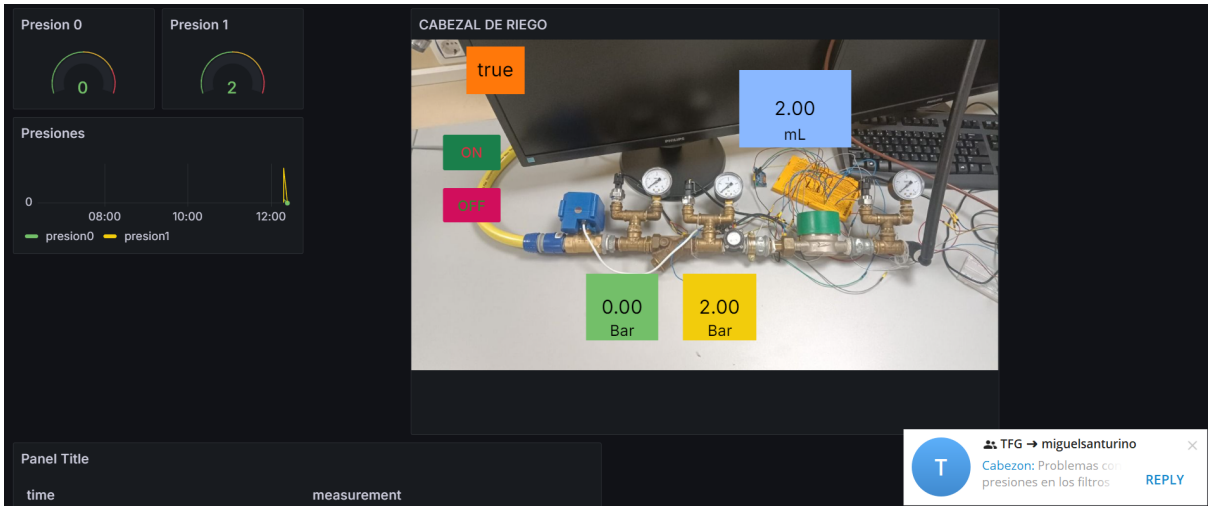

Figura 4.76: Dashboard de Grafana.

Como los valores de las presiones tienen un valor que supera el umbral que se ha programado, se muestra que llega un mensaje por Telegram avisando de la situación.

Los botones ON y OFF al pulsarlos enviarán en un json el true o false correspondiente a Total Flow y se accionará la electroválvula. Para conseguir esto se ha configurado en el botón el siguiente método.

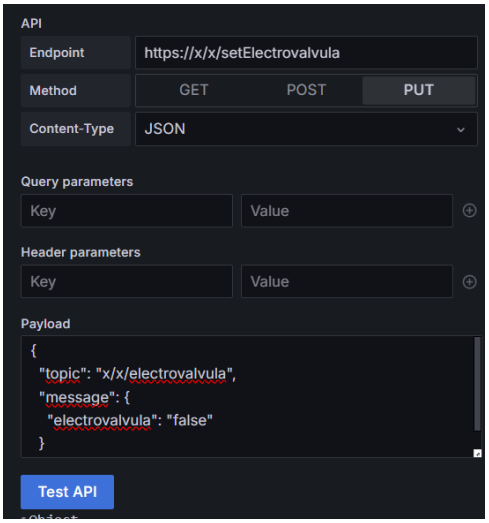

Figura 4.77: Configuración de los botones de Grafana.

4.2.2. OTA

El microcontrolador ESP32-S3 ha tenido que ser reconfigurado desde su menú entrando de forma manual en ciertos parámetros para poder desarrollar esta funcionalidad. Para ello, todo se hace desde la cmd versión 5.2 de ESP-IDF. Desde la ruta del proyecto se ejecuta el siguiente comando:

```
1 idf.py menuconfig
```

Tras este comando, se entra al framework de configuración de Espressif y se tiene un menú así:

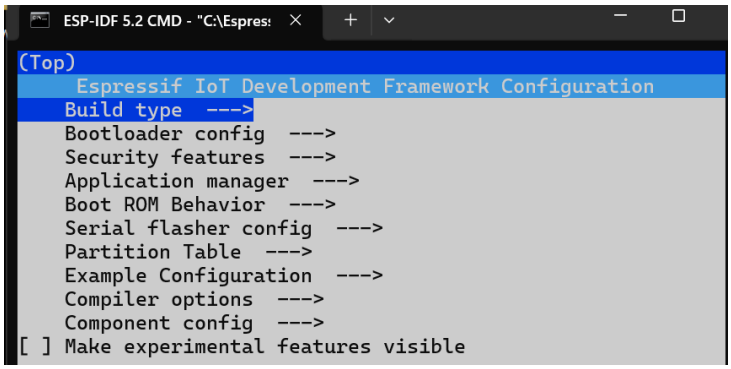

Figura 4.78: Framework de configuración de Espressif interno.

Lo primero que se hizo para permitir que el microcontrolador se comunicara con servidores HTTP fue habilitar HTTP para OTA. Se debe entrar a los siguientes apartados: **Component Config >ESP HTTPS OTA >Allow HTTP for OTA**

Por tratarse del protocolo HTTP, el microcontrolador no permite por defecto esta opción pues no es segura al no tratar de forma cifrada la información transmitida. Para la realización de pruebas de forma local, se ha creado una red aislada sin acceso a Internet, por tanto no supone un peligro habilitar esta opción y reduce considerable-

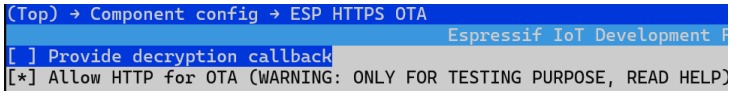

Figura 4.79: HTTP habilitado para OTA en ESP32-S3.

mente la complejidad pues no hay que tratar con certificados de seguridad entre el cliente y el servidor.

Otra configuración realizada al microcontrolador fue manipular las particiones. La partición por defecto del micro es **Single factory app, no OTA**. Inicialmente, para poder ver que funcionaba la OTA, se pasaba un binario donde su función era encender un led, de esa forma se veía rápidamente los resultados. Como es de esperar, el tamaño de este binario era mínimo, por lo tanto con elegir la opción **Factory app, two OTA definitions** en:

Partition Table >Partition Table (Custom partition table CSV) —>Factory app, two OTA definitions, era suficiente para que las actualizaciones se cumplieran de forma exitosa.

El ESP32-S3 trabaja con dos particiones OTA: OTA_0 y OTA_1. Permiten al microcontrolador manejar actualizaciones de firmware de manera segura y confiable. Este enfoque minimiza el riesgo de dejar el dispositivo inutilizable por una actualización fallida y asegura que siempre haya una versión funcional del firmware disponible. Los pasos que siguen para actualizar son los siguientes:

1. El firmware nuevo se descarga a la partición OTA que no está en uso (por ejemplo, si el firmware actual está en OTA_0, el nuevo firmware se descarga en OTA_1).
2. Una vez que la descarga y verificación del nuevo firmware se completan con éxito, el bootloader del ESP32-S3 actualiza los punteros o configuraciones para que en el próximo reinicio, el sistema arranque desde la partición OTA_1.
3. El sistema reinicia y arranca desde la nueva partición con el firmware actualizado.
4. Si el nuevo firmware arranca correctamente y pasa todas las pruebas iniciales, la partición OTA que contenía el firmware anterior puede ser marcada como disponible para la próxima actualización.

A la hora de introducir el código del proyecto de forma completa, el tamaño del binario que se recoge por OTA, había aumentado de forma considerable, por tanto, fue necesario manipular las particiones de forma manual para hacer que las dos particiones OTA que tiene el micro, pudieran tener un tamaño suficiente para albergar el tamaño del código. Se ha creado un archivo de tipo .csv (comma-separated value) nombrado partitions.csv que se encuentra dentro de la carpeta del proyecto y que contiene estos cambios. Así que la partición adecuada a elegir es:

Partition Table >Partition Table (Custom partition table CSV) —>Custom partition table CSV.

El contenido de las particiones del sistema se puede observar en esta imagen del archivo partitions.csv

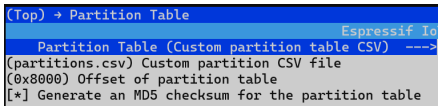

Figura 4.80: Partitions.csv elegida como partición customizada.

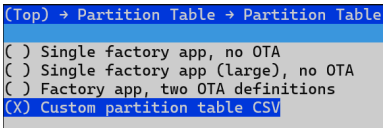

Figura 4.81: Elección customizada de la tabla CSV.

```
# Name,   Type, SubType,  Offset,   Size,
nvs,      data, nvs,      0x9000,   16k
otadata,  data, ota,      0xd000,   8k
phy_init, data, phy,      0xf000,   4k
ota_0,    app,  ota_0,    0x10000,  8M
ota_1,    app,  ota_1,    ,         8M
coredump, data, coredump, ,         64K
reserved, data, 0xfe,     ,         128K
```

Figura 4.82: Fichero partitions.csv del proyecto.

Una vez realizados los pasos anteriores, fue necesario una última configuración: aumentar el tamaño de memoria flash pues la que se tiene por defecto se había quedado pequeña tras todos estos cambios. Para ello la ruta es:

Serial flasher config >Flash size >32MB

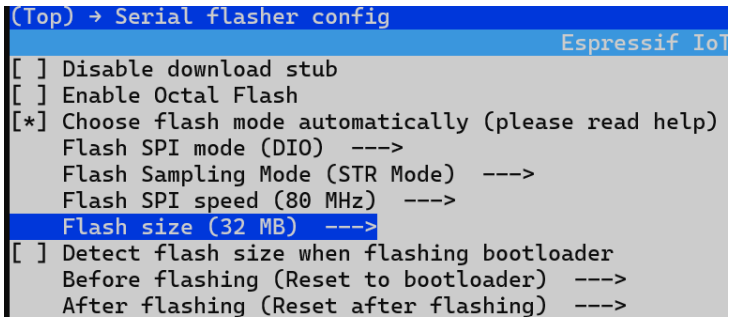

Figura 4.83: Configuración del tamaño de la flash en ESP32-S3.

En este momento, la OTA está perfectamente configurada para llevarse a cabo, ahora se explicará cómo es el código de esta funcionalidad de una forma generalizada y sus conexiones con los servidores.

La OTA es una tarea periódica que se ejecutará cada cierto tiempo en el sistema. Lo primero que hará es verificar que la versión de firmware que el microcontrolador posee, es superior o igual a la que se aloja en un json perteneciente al servidor de Total Flow. En caso de que esta versión sea inferior en el ESP32-S3, se ejecutará la OTA, obteniendo previamente el binario del servidor local.

Los datos que se tratan, son del tipo json. Gracias a la libreria cJson.h su implementación es bastante sencilla.

Como se ha comentado, lo primero que hace el sistema, es preguntar si está desfasado en la versión de firmware. Para ello hace una petición GET al servidor IoT de la siguiente forma: **https://xx/xx/getVersionOta**

En esta imagen se muestra de qué forma se gestiona la versión en el servidor de Total

Flow.

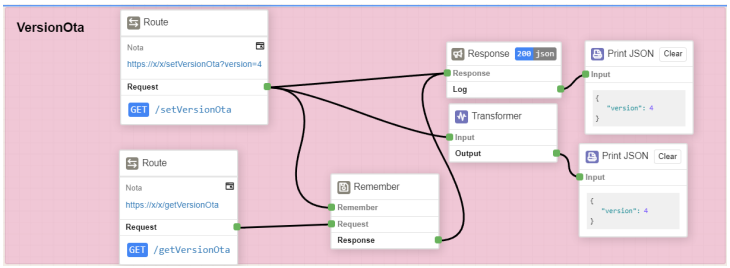

Figura 4.84: Tratamiento de la versión del sistema desde Total Flow.

Desde el servidor se ajustará la versión del firmware, que se guardará en el componente remember. En el momento que el microcontrolador realice la petición, el remember saltará y el servidor responderá al microcontrolador con **"version": 4**, en este caso. El ESP32-S3 una vez tiene este valor, hará las comparaciones pertinentes para tomar la decisión de ejecutar una actualización o seguir igual y volver periódicamente a hacer esta petición.

Suponiendo que la versión obtenida por el micro en la petición es superior a la que contiene en su programa actual, el ESP32 comienza el proceso de inicialización de la OTA.

En este momento, se realiza otra petición pero esta vez al servidor de Node.js desarrollado de forma personal.

Este servidor es sencillo en cuanto complejidad. Principalmente tiene el archivo index.js que es quien corre el servidor desde el puerto 80 (defecto para HTTP). En este archivo se desarrollan las peticiones necesarias para poder contener el binario del firmware a actualizar y que el microcontrolador pueda obtenerlo. Por otro lado, tiene el archivo firmware.json. Este contiene un json con la ruta hacia el binario que debe descargar el micro. Y como se acaba de decir, contiene el binario. Todos estos archivos se encuentran en el mismo directorio en el que está el index.js.

Cuando se realizan peticiones a este servidor, se pueden ir recibiendo estos eventos:

```
PS C:\Users\soyxu\Desktop\ota\servidor> node index.js
Servidor http corriendo en el puerto 80
Se sirvió el documento JSON con éxito
Archivo descargado
```

Figura 4.85: Eventos en el servidor de node.js.

OTA no depende de más elementos del proyecto. Los dos servidores que entran en juego tienen su propia función y la OTA no necesita realizar más peticiones para recoger información de otro tipo.

4.2.3. MESH

Inicialmente se empezó a desarrollar ESP-WIFI-MESH como se comentó en puntos anteriores. Sin embargo, la forma en que se debía comunicar con el exterior de la red, hizo que se tomara la decisión de cambiar a ESP-MESH-Lite. Esto se debe a

que para ESP-WIFI-MESH la comunicación con el exterior debe ser a través del nodo raíz de forma bidireccional. En ESP-WIFI-MESH todos los nodos que no son el raíz, no tienen la posibilidad de ir directamente a Internet, deben ir a través del raíz. En cambio, con ESP-MESH-LITE se puede ir a Internet con cualquier de los nodos pertenecientes a la red.

ESP-MESH-Lite es una aplicación de red Wi-Fi de IoT-Bridge, basada en el modo SoftAP + Station, un conjunto de soluciones Mesh construidas sobre el protocolo Wi-Fi. ESP-MESH-LITE permite que numerosos dispositivos distribuidos en una gran área física (tanto en interiores como en exteriores) estén interconectados bajo una sola WLAN [36].

Realmente, ESP-MESH-Lite está contenido en ESP-MESH pero con funcionalidades más simples. Por ello en el estado de arte se habló de forma genérica del protocolo del microcontrolador, pues ESP-MESH-Lite sigue exactamente esas reglas de funcionamiento.

Aquí se recogen las principales diferencias que están recogidas de la página oficial de Espressif [36]:

1. La mayor diferencia entre ESP-MESH-LITE y ESP-MESH es que ESP-MESH-LITE permite que los sub-dispositivos en la red accedan de forma independiente a la red externa, y la información de transmisión es independiente del nodo padre, lo que reduce enormemente la dificultad de desarrollar la capa de aplicación.

2. ESP-MESH-LITE necesita menos espacio de memoria que ESP-MESH pero en esta última, la reconfiguración de los nodos es relativamente mejor.

3. La forma de elegir al nodo padre automáticamente es distinta:

    a) ESP-MESH: Cuando todos los dispositivos están inactivos después de encenderse, el nodo raíz se selecciona según el RSSI (Indicador de Intensidad de Señal Recibida), y luego el nodo raíz se conecta al router.

    b) ESP-MESH-LITE: Por defecto, el dispositivo que se enciende primero después de la provisión se selecciona como el nodo raíz. Si más de un dispositivo se enciende al mismo tiempo, después de que todos los dispositivos se conectan al router, todos los dispositivos comienzan a transmitir el RSSI del router, y el dispositivo con el RSSI más fuerte se selecciona como el nodo raíz. El resto de los nodos se desconectan del router y vuelven a escanear para encontrar el nuevo nodo padre.

4. Solo el nodo raíz en ESP-MESH habilita la pila LWIP, y todos los nodos hijos necesitan ser reenviados a través del nodo raíz si desean comunicarse con la red externa.

En líneas anteriores, se decía que solo el nodo raíz en ESP-MESH habilita la pila LWIP, en ESP-MESH-Lite cada uno la habilita y puede ser tratado como un dispositivo directamente conectado al router que puede independientemente invocar interfaces de red como Socket, MQTT, HTTP, etc.. en la capa de aplicación. [36].

La conexión del nodo raíz es mediante Wi-Fi de 2,4 GHZ a una red que se ha denominado **TFG_WIFI**, usando de punto de acceso un dispositivo móvil.

La imagen que se va a mostrar a continuación ilustra como es la transmisión de datos de varios nodos conectados recibiendo y emitiendo información de forma bidireccional en ESP-MESH-Lite. Todos los datos transmitidos están cifrados con WPA2-PSK [37].

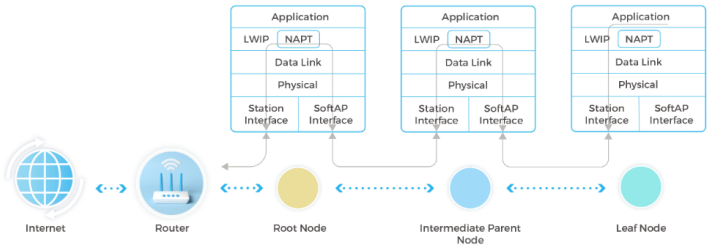

Figura 4.86: Transmisión de datos bidireccional en Lite [36].

La siguiente tabla muestra ciertos datos de interés sobre características de las redes Lite que todas tienen en común:

Función	Milisegundos
Tiempo de provisionamiento a la red	<60000
Delay entre saltos	8-12
Reparación de la red tras desconexión de root	<50000
Reparación de la red tras desconexión de hijo	<45000

Cuadro 4.2: Características comunes en ESP-MESH-Lite.

Para comenzar el desarrollo de ESP-MESH-LITE fue necesario hacer varias integraciones: Se descargó del repositorio oficial de Espressif en github [36], la librería de ESP-MESH-Lite. Esta tecnología a día de hoy está desplegada pero sigue teniendo integraciones constantemente por parte de los desarrolladores, por tanto no se encuentra como una librería por defecto en la carpeta de Espressif.

Su integración vino acompañada de un ejemplo práctico denominado **local_control**. Sirvió de plantilla para ir añadiendo o quitando código según interesaba.

Una vez se encontraba instalado todo lo referido a ESP-MESH-Lite, fue necesario añadir otra dependencia para poder formar el bridge de la red para poder viajar a Internet. Se añadió la dependencia necesaria llamada iot_bridge [38]

```
1 idf.py add-dependency "espressif/iot_bridge^0.11.7"
```

El comienzo de las pruebas fue para ver cómo iba la topología, separando de forma intencionada los nodos para que de esa forma cogieran el rol que interesaba. Todas estas pruebas han sido realizadas con 3 nodos para poder usar los 3 roles que tiene un dispositivo dentro de la red de forma operativa (sin contar los nodos ociosos).

Se disponía inicialmente de dos dispositivos ESP32 y un único ESP32-S3. La conexión entre nodos siempre era la misma: ESP32-S3 era el raíz y los ESP32 se quería

tener uno como intermedio y otro como hoja. El problema es que siempre se conectaban los dos ESP32 como hijos. La diferencia en las tecnologías que se usan entre los modelos respecto a temas Wi-Fi o señal, no parecían que fueran el problema. Por ello se decidió disponer de 3 ESP32-S3 y hacer las pruebas con el modelo más avanzado y que se quería usar en el proyecto.

En este momento, haciendo las pruebas con más distancia, al aire libre y dispositivos del mismo tipo, se consiguió lo que se esperaba. Un nodo padre, un nodo intermedio y un nodo hijo. Esta imagen muestra la ejecución desde el punto de vista del nodo raíz. Se puede observar como en su tabla de rutas se encuentran dos entradas en el instante 845118ms y 848548ms, cada una con la dirección MAC de los otros dos dispositivos.

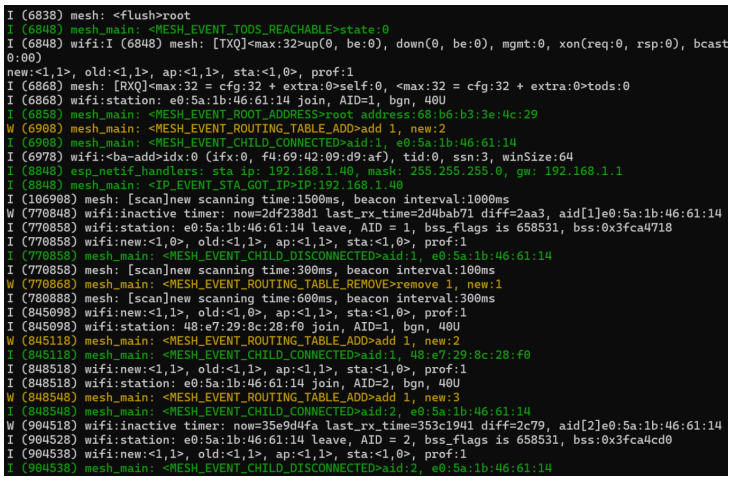

Figura 4.87: Nodo raíz con las direcciones MAC de los dispositivos.

En el proyecto, solo se usan tres dispositivos para ver el funcionamiento de una red MESH de forma reducida. Sin embargo podrían estar en la misma red, decenas de estos dispositivos. Si se habla de microcontroladores ESP32-S3, el fabricante, informa de que la distancia entre dos de ellos puede ser de 100 metros en unas condiciones ideales (prácticamente imposible) [17]. Pero, si realmente fuera así, unir 10 dispositivos ESP32-S3 cada uno a 100metros, harían posible cubrir 1km. En el ejemplo que Espressif ofrece de local_control, pone a prueba a 50 dispositivos.

Otra de las pruebas que se han realizado, es verificar que una vez la red MESH se ha levantado, desde cualquier otro dispositivo, se puede observar que aparece una nueva Wi-Fi que se ha denominado **TFG_MESH** y a la que se puede acceder. Esta red es la que se crea con el bridge y a la que se van a unir cada uno de los nodos de MESH. Es necesario que todos los dispositivos tengan la misma configuración de la red mallada para que se puedan unir sin problema. Es decir, deben conocer todos el SSID y su contraseña.

Dicho lo anterior, como el nodo raíz se conecta al punto de acceso del dispositivo móvil, desde otro dispositivo como un ordenador, se puede apreciar que aparece el punto de acceso a TFG_MESH y unirse a él como se puede apreciar en esta imagen:

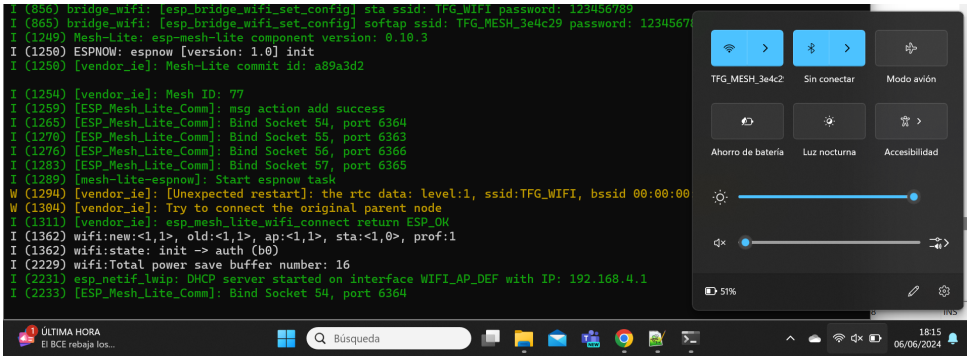

Figura 4.88: Ordenador conectándose a la red MESH.

Entrando en detalle en la imagen anterior: la terminal pertenece a un nodo que se ha conectado al punto de acceso del móvil (TFG_WIFI) y que ha creado la red MESH (TFG_MESH) como se puede observar en el momento 863ms. Al final de la imagen se ve que obtiene una IP de parte del servidor DHCP. A la derecha se ilustra que el ordenador ha podido unirse a la red TFG_MESH.

Esto es uno de los puntos claves dentro de las redes MESH como estas. Permite que dispositivos de cualquier tipo puedan tener acceso a la red, brindando de cobertura y de poder conseguir rangos de trabajo mucho más lejanos que otros sistemas que se limitan al rango de su punto de acceso.

Comentar que todas las credenciales, tanto de la red TFG_WIFI como de la red TFG_MESH están ocultas y se han definido en el archivo sdkconfig que todo proyecto contiene. Este archivo almacena todas las configuraciones que se editan en el menuconfig y en el momento de compilación, el micro cogerá este archivo para saber exactamente cómo debe ser configurado el ESP32-S3.

Si se produce un cambio de modelo de microcontrolador a la hora de compilación, se debe ejecutar el siguiente comando en caso de estar usando un ESP32-S3:

```
idf.py set-target "esp32s3"
```

Este cambio de modelo hace que el fichero sdkconfig se modifique y se quede por defecto. Todos los cambios realizados anteriormente deben ser repetidos de nuevo para poder ser guardados.

4.2.4. Programación de los dispositivos digitales

1. **Electroválvula**
   Se ha destinado el GPIO7 para el accionamiento de este componente. Lo único que hay que configurar es el pin 7 como output e inicializarlo en nivel alto, de esta forma, se mantendrá cerrada. Este motivo de iniciar a nivel alto como cerrado es debido a la forma en que se conecta al relé. Se ha decidido que a nivel alto sea cerrada y por tanto a nivel bajo estará abierta. Aquí se muestra la configuración de la electroválvula:

```
gpio_set_direction(7,GPIO_MODE_OUTPUT)
gpio_set_level(7,1)
```

Como el sistema está conectado al servidor IoT mediante MQTT y está suscrito a un topic que siempre que el usuario desde Telegram o Grafana presione el ON o el OFF, se publicará en dicho topic. Por consiguiente, el microcontrolador recibirá el dato y dependiendo de si se trata de un 'true' o un 'false', abrirá o cerrará la electroválvula.

Como se comentó en el caso de uso número 1, si la electroválvula se encuentra abierta y el usuario presiona la tecla de abrir, no va a ocurrir ningún tipo de acción en el sistema. De forma análoga si el sistema está cerrado y se da al botón de apagar desde algún programa.

2. **Sensor de presión**
   Para el sensor de presión, se ha dejado los pines números 5 y 4, que corresponden al ADC1_CHANNEL4 y ADC1_CHANNEL3 respectivamente. La configuración de cada uno de ellos como se explicó en el apartado de pruebas con el ADC, ha sido usando Curve Fitting y la opción one_shot del ADC. Aquí se tiene la configuración de uno de ellos:

```
1     adc_oneshot_unit_handle_t adc1_handle = NULL;
2     adc_oneshot_unit_init_cfg_t init_config1 = { .unit_id = ADC_UNIT_1, };
3     ESP_ERROR_CHECK(adc_oneshot_new_unit(&init_config1, &adc1_handle));
4     adc_oneshot_chan_cfg_t config = {
5         .bitwidth = ADC_BITWIDTH_12,
6         .atten = ADC_ATTEN_DB_12, // 0 mV ~ 3100 mV
7                 };
8         ESP_ERROR_CHECK(adc_oneshot_config_channel(adc1_handle, ADC_CHANNEL_4, &config))
              ;
9     do_calibration1_chan4 = adc1_calibration_init(ADC_UNIT_1, ADC_CHANNEL_4,
          ADC_ATTEN_DB_12, &adc1_cali_chan4_handle);
```

Se explicó en el apartado de dispositivos hardware utilizados, el sensor de presión es capaz de medir de 0 a 1,2MPa. Su equivalente en bares es de 0 a 12. En este trabajo se va a trabajar con bares. Además, la salida de los sensores es de 0,5 a 4,5V y las entradas de los ADC leen hasta los 3,1V como máximo. Por este motivo se hizo los divisores de tensión, aunque realmente el divisor de tensión está hecho en un rango de 0 a 5V, siguiendo el estándar de las familias TTL que llegan hasta los 5V.

Teniendo en cuenta el rango de medición del sensor y el rango de tensiones de salida que tiene este y el que acepta el ADC del ESP32-S3, se ha hallado una fórmula matemática que transforme la tensión que entra en el ADC en su equivalente en bares. Es decir, si el ADC recibe a su entrada 3,1V, eso serán 12 bares. Esta formula depende del offset de cada sensor: el desplazamiento ha sido hallado conectando el sensor y viendo que medida registraba el ADC con un número aproximado de 100 pruebas para intentar tener un error mínimo. Este valor se compara el real que se obtiene con el multímetro en cada uno de los sensores.

Con todo esto, cada sensor tendrá su fórmula con un número de offset distinto dependiendo la precisión del sensor, el divisor de tensión y el ADC. Aquí se muestra una tabla de cada sensor con su medida a través del ADC, del multímetro y la diferencia entre los dos valores.

Sensor	Offset (mV)	Multímetro (mV)	Diferencia
Presión 0	264	286	22
Presión 1	288	304	16
Presión 2	281	298	17
Presión 3	280	298	18

Cuadro 4.3: Offset obtenido en cada sensor.

Sabiendo ya que es el offset se han hallado estas ecuaciones:

```
1 medidaAdc=min(max(offset,tension), 2790 + (offset - 310)) - offset;
2 presion0=(float)medidaAdc*(12/(2790-offset)
```

En la primera, se consigue la tensión real que se ha obtenido del ADC teniendo en cuenta los desplazamientos y siempre se asegura que de 0 al offset, sea 0 y por encima de la cota superior, siempre de esa cota superior que serían los 3,1V En la segunda se transforma la medida para que la tensión obtenida se transforme en bares.

3. **Caudalímetro**
   El sensor internamente tiene un rotor cuyas paletas tiene un imán, la cámara en donde se encuentra el rotor es totalmente aislado evitando fugas de agua. Externamente a la cámara tiene un sensor de efecto Hall que detecta el campo magnético del imán de las paletas y con esto el movimiento del rotor, el sensor de efecto Hall envía los pulsos por uno de los cables del sensor, los pulsos deberán ser convertidos posteriormente a flujo [39].

   Se ha usado el GPIO 6 para el caudalímetro. Su configuración es la siguiente:

```
1 gpio_config_t interruptConfig =
2         {
3         .pin_bit_mask = (1ULL<<PIN_NUM6_CAUDALIMETRO),
4         .mode = GPIO_MODE_INPUT,            //entrada
5         .intr_type = GPIO_INTR_POSEDGE  // interrupcion en flanco de subida
6     };
7     ESP_ERROR_CHECK(gpio_config(&interruptConfig)); //configuracion de la interrupt
8     ESP_ERROR_CHECK(gpio_install_isr_service(0));
9         gpio_isr_handler_add(PIN_NUM6_CAUDALIMETRO, pulse,(void *)PIN_NUM6_CAUDALIMETRO)
              ;
```

   En esta configuración se contempla que haya una interrupción en flanco de subida desde el GPIO 6 que será de entrada al microcontrolador, y un manejador hacia la función pulse que es la encargada de ir aumentando el agua que dirá el consumo que se lleva de agua.

   Se usa esta formula: Flujo del agua en L/min = Pulsaciones del sensor (Hz) / 7.5.

   Siendo 7,5 el factor de conversión K que nos da el fabricante para este sensor de 1/2" de diámetro [27]. De esta forma se halla la cantidad de agua promedio que pasa por minuto a través del caudalímetro. La variable que recoge la cantidad

de agua acumulada es de tipo volatile long, esto quiere decir que en tiempo de ejecución su valor va a ir cambiando.

Los valores obtenidos por los sensores de presión y el caudalímetro serán enviados por MQTT al servidor IoT y serán introducidos en la base de datos. Además, dependiendo el valor de estos sensores, puede que el usuario reciba algún tipo de notificación a Telegram si hay algún problema.

4. **Deep Sleep**
   Los sistemas IoT que recogen medidas de sensores, como es el caso de este proyecto, no tienen sentido que recojan datos cada pocos segundos debido a varias razones. Primero, hay que considerar la utilidad real de los datos. En el cabezal de riego, por ejemplo, se recogerán valores principalmente cuando el sistema esté en funcionamiento y activamente regando. Durante los periodos en los que no se está realizando el riego, las lecturas de los sensores no aportan ningún valor relevante a la operativa del sistema y simplemente contribuyen a un aumento innecesario del consumo energético.

   Implementar modos de bajo consumo, como el deep sleep, en microcontroladores como el ESP32-S3, es una estrategia eficaz para gestionar y optimizar el uso de la energía. El deep sleep permite que el microcontrolador entre en un estado de hibernación, donde solo se consumen unos pocos microamperios de corriente. Esto es esencial en aplicaciones donde el dispositivo necesita funcionar durante largos periodos con baterías o fuentes de energía limitadas. Además, durante el deep sleep, el microcontrolador puede ser despertado por eventos específicos, como una interrupción externa o un temporizador interno, asegurando que el sistema pueda volver a un estado operativo completo cuando sea necesario.

   Gracias a estos modos de ahorro de energía, se reduce drásticamente el consumo del sistema mientras se mantiene la capacidad de respuesta inmediata. El ESP32-S3, en particular, ofrece varias opciones para la gestión de energía, permitiendo al desarrollador elegir el balance óptimo entre consumo y funcionalidad activa. Estos modos incluyen el light sleep, donde algunas partes del sistema permanecen activas para una recuperación más rápida, y el deep sleep, donde casi todo el sistema se apaga para un consumo mínimo.

   En el caso del deep sleep del ESP32-S3, los consumos y las funcionalidades activas o desactivadas pueden variar. En este estado, el reloj principal se detiene, la memoria RAM puede ser parcialmente o completamente apagada, y solo un pequeño circuito permanece activo para escuchar eventos de despertar. Esta estrategia es particularmente útil en aplicaciones de monitoreo ambiental, agricultura de precisión y otros sistemas IoT donde la actividad continua no es necesaria, pero la capacidad de recopilar datos periódicos y operar durante largos periodos sin recargar es crítica.

   En la imagen siguiente, recogida de la asignatura de Sistemas Basados en Computadores, se muestra un resumen detallado de los consumos energéticos y las funcionalidades que se mantienen activas o desactivadas en los distintos modos de operación del ESP32-S3.

Máximo consumo	Active	Todos los componentes activos siempre, incluidos bluetooth y wifi	90-260mA
	Modem-sleep	Wifi y bluetooth desactivados. Además, se puede conseguir reducir el consumo, reduciendo la velocidad del procesador (máxima, media y baja)	Máxima: 20mA
			Media: 2-10mA
			Baja: 3mA
	Light-sleep	Wifi y bluetooth desactivados. Procesador principal (ESP32 CPU) dormido y despierta ante eventos externos (timer o botonazo). Al despertar, el procesador mantiene contenido de la memoria.	0.8mA
	Deep-sleep	Wifi, bluetooth, procesador principal (ESP32 CPU), y la mayoría de los periféricos desactivados. El co-procesador sigue activo. Atiende a eventos externos. Si se desactiva el co-procesador se reduce aun más	0.15mA
			10μA
Mínimo consumo	Hibernation	Todo desactivado excepto el RTC timer. Se puede reactivar por timer. En este caso, no se preservan datos al volver de la hibernación.	5μA

Figura 4.89: Modos de funcionamiento y consumos del ESP32-S3 [40].

El modo deep sleep comienza una vez se llama a esta función:

```
1 esp_deep_sleep_start();
```

El sistema está configurado para que se levante una vez se ha comenzado el deep sleep según el número de milisegundos que se pongan en la siguiente función. En este caso son 20 segundos.

```
1 esp_sleep_enable_timer_wakeup(20 * 1000000);
```

A parte de la forma estándar para despertar al sistema, se ha desarrollado una función donde se configura un GPIO touch para que cuando note el contacto de un dedo humano, el sistema se reactive. Los pines GPIO touch son capaces de detectar cambios en la capacitancia, lo cual se produce cuando un objeto conductor, como un dedo humano, se acerca o toca el pin. Esto convierte al ESP32-S3 en un controlador táctil capacitivo, permitiendo la implementación de interfaces táctiles sin necesidad de componentes adicionales. El sensor táctil funciona midiendo el tiempo que tarda un pin en cargarse a un cierto nivel de voltaje. La presencia de un objeto conductor cercano cambia la capacitancia, alterando el tiempo de carga. El microcontrolador detecta estos cambios y los interpreta como eventos táctiles. La función desarrollada es la siguiente:

```
1  void parametros_dee_sle()
2  {
3      gpio_config_t io_conf = {};
4      io_conf.intr_type=GPIO_INTR_POSEDGE;
5      io_conf.pin_bit_mask = (1ULL << 14); // gpio 14
6      io_conf.mode = GPIO_MODE_INPUT;
7      io_conf.pull_up_en = GPIO_PULLUP_DISABLE;
8      io_conf.pull_down_en = GPIO_PULLDOWN_DISABLE;
9      gpio_config(&io_conf);
10
11          // Configurar el pin para despertar del deep sleepen nivel alto
12     esp_sleep_enable_ext0_wakeup(14, 1);
13 }
```

El deep sleep será tratado por parte del usuario desde el bot de Telegram. Tendrá un comando que al ejecutarse, enviará una orden a Total Flow la cual avisará al sistema de que debe entrar en estado de deep sleep.

4.3. Costes energéticos del sistema

En este apartado se ilustrarán los costes energéticos asociados que tiene el cabezal de riego inteligente, teniendo de referencia siempre el caso donde se encuentra solamente el ESP32-S3 consumiendo en el sistema.

Para realizar estas pruebas se ha soldado a la entrada del jack macho dc de 24 voltios y corriente de 1 amperio, su hembra con un cable para la tensión y otro para cerrar el circuito a tierra; se ha utilizado una fuente de alimentación del laboratorio ajustando los valores a 24V y 1 A.

En esta primera ilustración, se muestra que el sistema si no tiene ningún tipo de dispositivo conectado, su consumo es de 0 watios pues no circula nada de corriente por su interior.

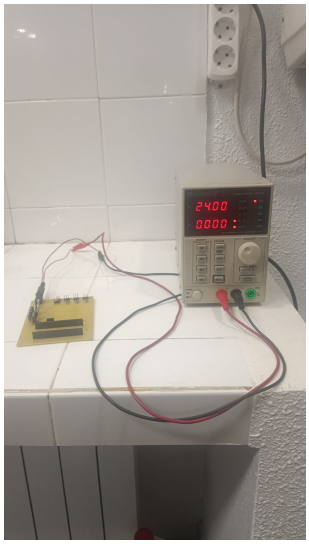

Figura 4.90: Consumo de la placa del sistema sin componentes conectados.

Ahora se conecta el ESP32-S3 sin tecnologías activadas; su código en este momento es un bucle infinito sin ningún tipo de función. En un bucle while(1) vacío, el ESP32-S3 seguirá consumiendo energía en modo activo mientras ejecuta el bucle, pero consumirá menos energía que si estuviera realizando tareas más intensivas en CPU o comunicaciones.

Se observa una corriente de 0.054 amperios por lo tanto el consumo será aproximadamente de 1,296 watios teniendo el microcontrolador enchufado.

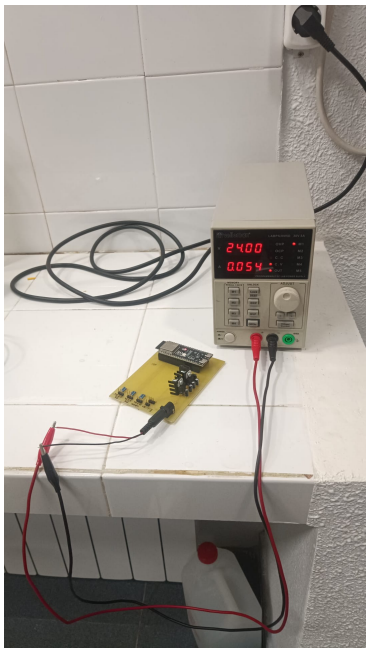

Figura 4.91: Consumo de la placa del sistema con el microcontrolador conectado.

A continuación, se añade al ESP32-S3 todas las funcionalidades que va a requerir para este sistema.

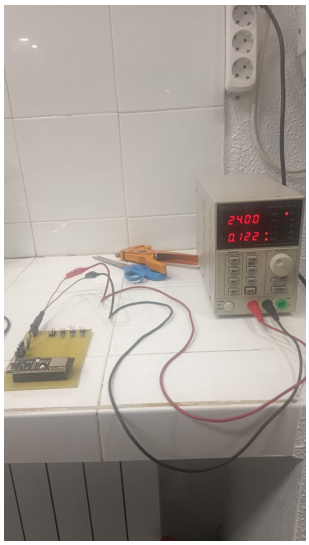

Figura 4.92: Consumo de la placa del sistema con el microcontrolador y sus funciones programadas.

Es de esperar, que el consumo aumente pues está conectado a Internet por WIFI y a la red MESH con otro nodo que se encontraba en otro lugar del laboratorio. Además tiene el conversor analógico digital configurado y los protocolos MQTT y HTTPS corriendo. La corriente aumenta a 0,122 amperios que se traducen en un consumo de 2,928 watios.

En este momento, se empiezan a introducir componentes dentro del sistema manteniendo las funcionalidades anteriores en el ESP32-S3. Primero, se conecta el relé y se activa su bobina para ver el consumo del sistema:

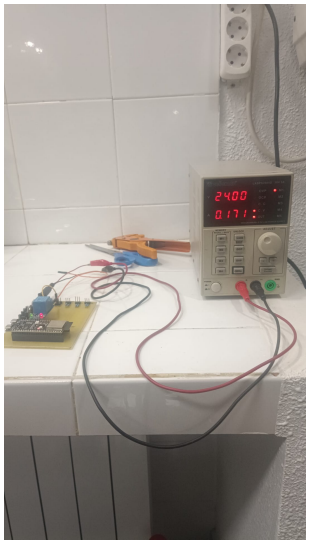

Figura 4.93: Consumo de la placa del sistema con el relé activado.

Se llega a los 0,171 amperios cuando la bobina está activa, dando lugar a 4,14 watios de consumo.

Tras haber probado el relé, se introduce el componente del que será encargado de activar: la electroválvula.

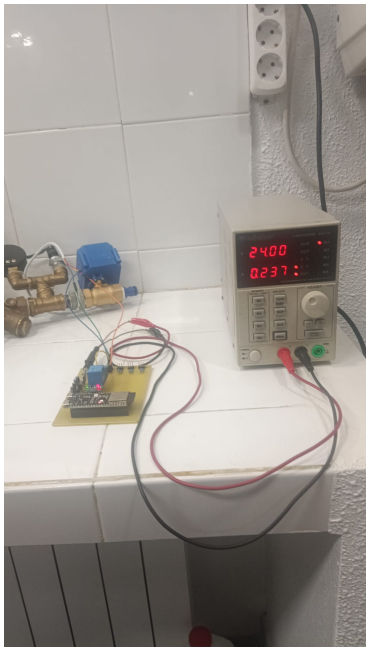

Figura 4.94: Consumo de la placa del sistema con la electroválvula abriéndose.

Esa corriente es justo en el momento que la electroválvula se está abriendo. Se nota un aumento considerable. Los 0,237 amperios en términos de consumo se traducen en 5,688 watios.

Ahora se desactivan el relé y la electroválvula para ver cuál es el consumo de los sensores de presión.

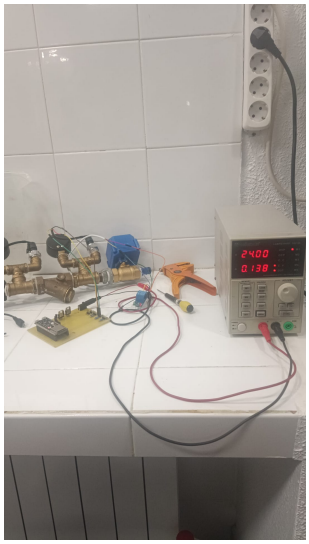

Figura 4.95: Consumo de la placa del sistema con el sensor de presión.

Se llega a los 0,138 amperios, por tanto el consumo es de 3.312 watios. La prueba con el caudalímetro fue prácticamente igual por tanto se ha omitido su imagen y explicación.

Otra función que se ha desarrollado en el proyecto para lograr un gran ahorro energético, como se comentó es el deep sleep. Gracias a esta función, la placa solo tiene una corriente de 0,009 amperios, lo que se traduce en 0,216 watios.

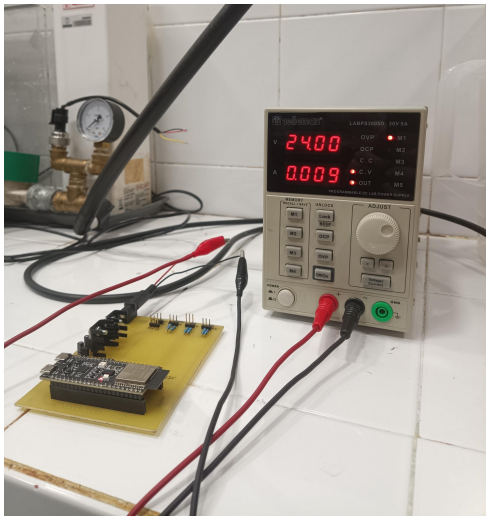

Figura 4.96: Consumo de la placa cuando está en modo Deep Sleep.

Finalmente, se ilustra una imagen de todo el sistema que iría conectado a la placa con un consumo medio de 6,192 watios aproximadamente.

Figura 4.97: Consumo de la placa del sistema completo.

En esta tabla se muestran de forma resumida los resultados obtenidos anteriormente:

Función	Watios	Amperaje a 24V
ESP32S3 en modo Deep sleep	0,216	0,009
ESP32S3 sin funciones	1,296	0,054
ESP32S3 con funciones	2,928	0,122
ESP32S3 con funciones + sensor de presión	3,312	0,138
ESP32S3 con funciones + caudalímetro	3,312	0,138
ESP32S3 con funciones + relé	4,14	0,171
ESP32S3 con funciones + relé + electroválvula	5,688	0,237
Todo el sistema funcionando	6,192	0,258

Cuadro 4.4: Consumos energéticos del sistema.

Capítulo 5

Resultados

El fin de este capítulo es dar una explicación a cada uno de los resultados finales de cada una de las funcionalidades implementadas a lo largo del proyecto.

1. **Monitorización de las funcionalidades del sistema mediante aplicaciones IoT.**
   Se ha obtenido un gran resultado en el desarrollo de las aplicaciones IoT, consiguiendo una intercomunicación entre las distintas partes de manera exitosa. Total Flow como se comentó, ha sido el centro de comunicaciones entre los distintos componentes IoT del sistema. Gracias a su configuración ha sido posible relacionar la base de datos de PostgreSQL que se encarga de almacenar los datos de los sensores, con Grafana; donde se realizan las consultas pertinentes para poder obtener los valores de cada uno de los sensores que se mencionaban. Además, se han creado un gran conjunto de reglas que gobiernan el correcto funcionamiento del bot de Telegram de una manera eficiente y bastante sencilla. Sin olvidar mencionar, todo lo relacionado con las reglas MQTT y HTTPS creadas para gestionar el envío y recepción de datos del ESP32-S3 sobre sus dispositivos digitales conectados o bien para poder llevar un control de cuándo debería ejecutarse la OTA dependiendo de su versión.

   Se va a mostrar como ha quedado finalmente la distribución de las normas y reglas de Total Flow justificando el porqué de cada bloque.

   La REST API no ha sufrido apenas cambios en comparación con la versión que se enseñó durante el desarrollo. Siguen manteniéndose las dos mismas peticiones que se tenían. GET para que el microcontrolador pueda recoger los datos de los sensores mediante HTTPS y un PUT para que el microcontrolador pueda envíar datos al servidor. Todos los elementos complementarios, son utilizados para poder llevar el control de flujo y transformar los datos que proceden del microcontrolador en la forma que interesa para su manejo.

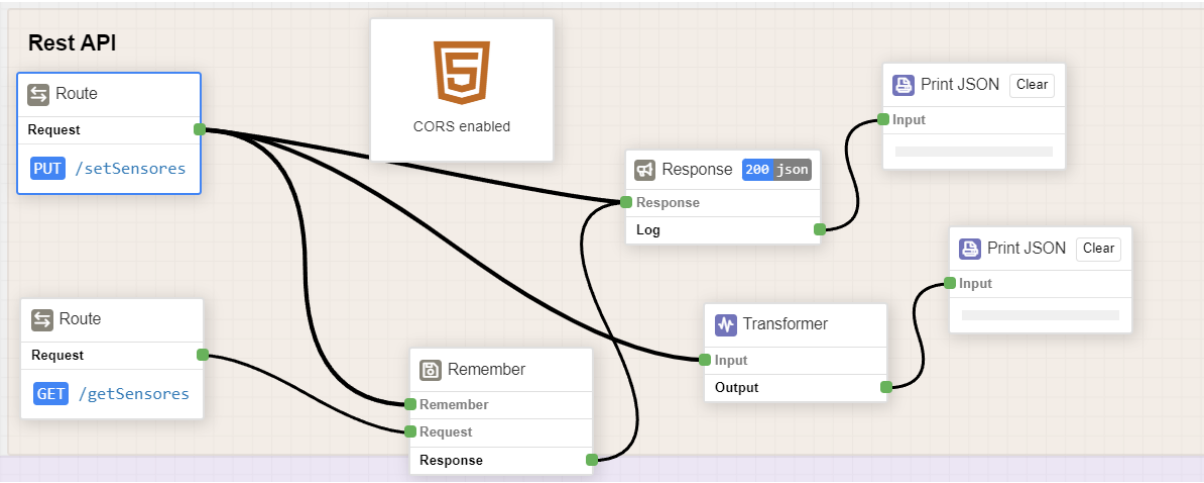

Figura 5.1: Reglas API REST de Total Flow.

Otro bloque que no ha sufrido ninguna modificación, ha sido el encargado de suministrar al microcontrolador la versión actualizada del firmware por si hubiera que disparar la actualización por OTA. Su funcionamiento quedó desplegado en pleno desarrollo y ha sido eficiente su configuración hasta el final del proyecto. Desde el servidor se modifica la versión cuando interesa la actualización por OTA y el ESP32-S3 se encarga de obtener con una petición GET esa versión. El micro tiene una tarea periódica que se ejecuta dos veces por día mínimo para verificar el estado de la OTA.

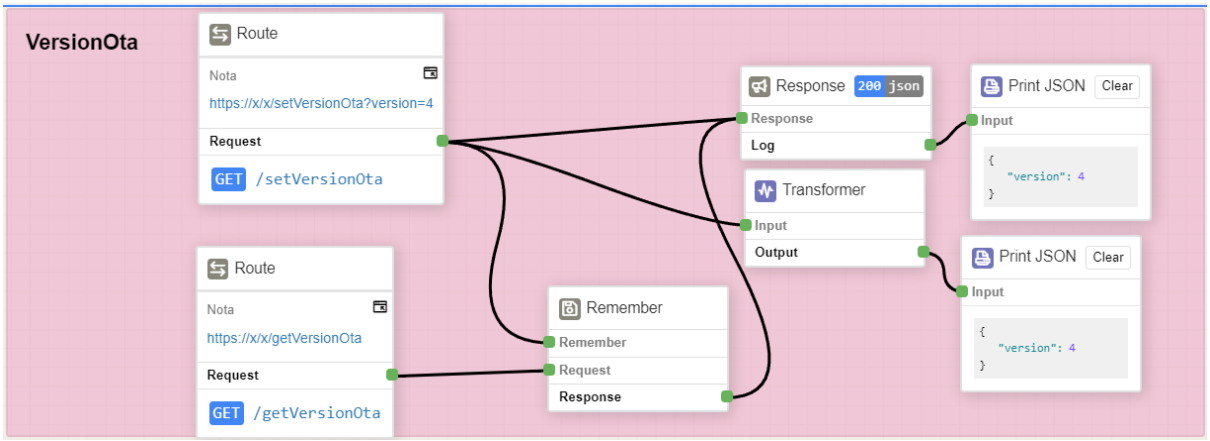

Figura 5.2: Reglas versión OTA de Total Flow.

Dos bloques nuevos que se han elaborado para el final del proyecto, han sido por un lado: el que se encarga de notificar cuando hay un problema de las presiones en el filtro y el caudalímetro, y el encargado de las notificaciones por problemas de presiones en las mangueras. Se han organizado de esta forma pues el caudalímetro y las presiones del filtro proceden del mismo ESP32-S3 (el de la placa), en cambio, las presiones de las mangueras proceden cada una de un microcontrolador distinto, por tanto su desarrollo ha sido ligeramente más complejo.

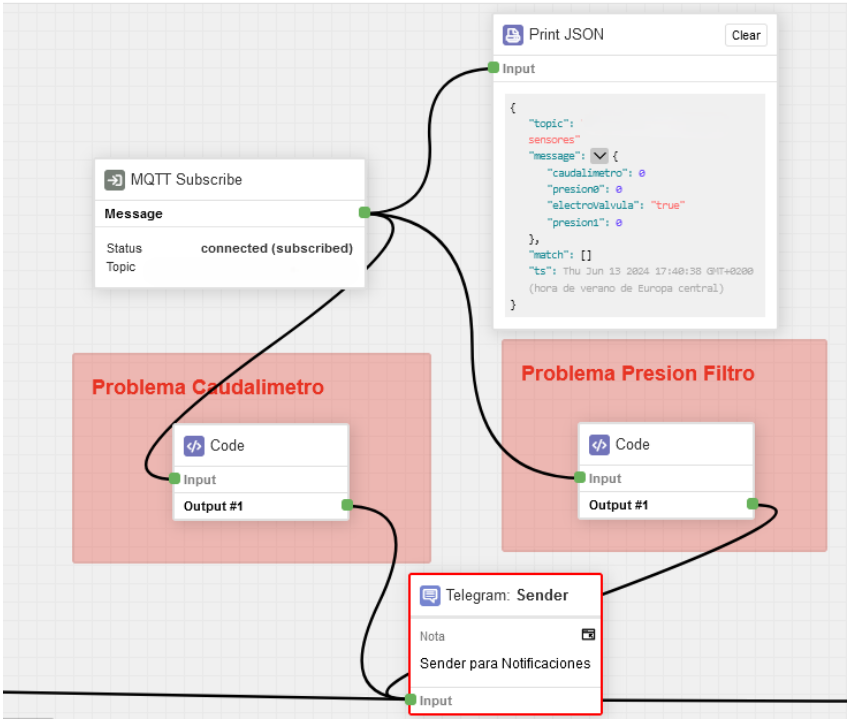

Figura 5.3: Reglas para el caudalímetro y filtro del sistema.

En la imagen superior se muestra el topic suscrito a sensores, que proviene del microcontrolador de la placa. Ambos van hacia los componentes Code, donde programados en javascript, se han puesto las condiciones que harán que se envíe o no una notificación a Telegram mediante el Sender que aparece en la parte inferior. La diferencia que deben tener las presiones debe ser superior a 0,6 bares para que se envíe la notificación. El caudalímetro una vez supera los 3 litros, envía el mensaje a Telegram.

En la imagen inferior se va a mostrar el bloque encargado de gestionar el valor de las presiones pero ahora de las mangueras:

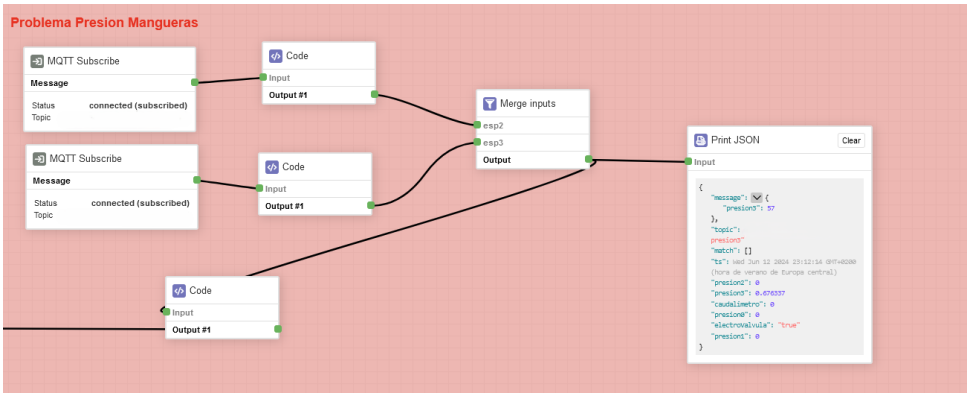

Figura 5.4: Reglas para las mangueras del sistema.

Este bloque contiene dos suscripciones. Cada una proviene de uno de los nodos que fue conectado en el sistema sin estar empotrado en la placa. Cada uno envía el valor de su sensor de presión y en el componente Code se filtra la información que viene y se queda con el payload que interesa. Es decir, el valor de cada presión. El componente Merge Inputs funciona como una especie de barrera. Hasta que no lleguen los dos valores de cada uno de las suscripciones, no seguirá avanzando el output. De esta forma se garantiza que se recoge el último dato suministrado por los sensores. Una vez salen del Merge, se hace la comparación de las presiones en el último Code. Si es superior a los 0,6 bares, se mandará la notificación por Telegram.

A parte de tener topics distribuidos por los distintos bloques, se tiene uno genérico para MQTT como se muestra en esta imagen:

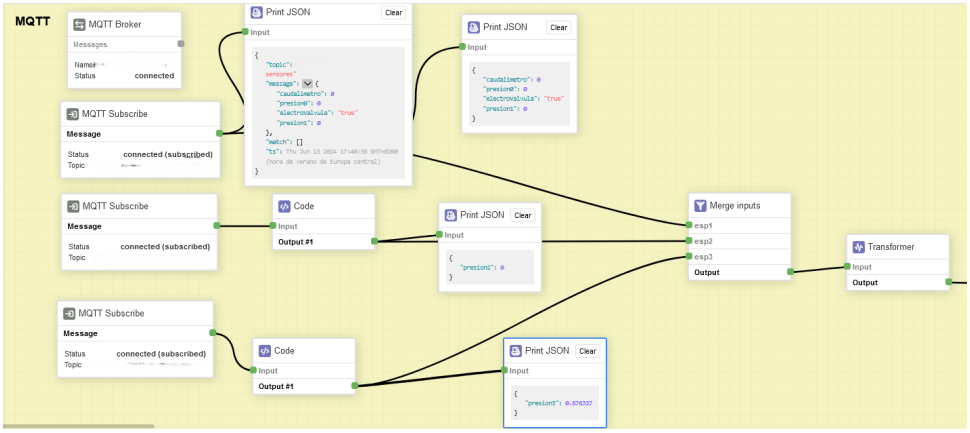

Figura 5.5: Reglas para MQTT en Total Flow.

Se muestra el Broker en la parte superior izquierda del sistema, que sirve para cada uno de los Subscribe y Publish dentro del servidor Total Flow. Como se puede apreciar, se encuentran 3 Subscribe. Cada una proviene de cada uno de los nodos del sistema. Tras llegar el dato en los componentes Code, como viene siendo de forma genérica, se transforma el json recibido en la carga útil que se necesita. Tras este paso todas se dirigen al Merge Input. Como se ha comentado, hasta que no lleguen las 3 suscripciones, no se avanzará al siguiente paso. Todo esto va hacia un transformer que sirve para eliminar campos del json que no dan información relevante. Estos campos son 'ts', 'match' y 'topic'. Se puede observar en el print de la imagen en la parte superior izquierda, que contiene estos campos y no ofrecen ningún valor relevante para ser tratado. Los datos cuando salen de este componente, se dirigen hacia el bloque de la base de datos para ser introducidos. A continuación se explicará dicho bloque.

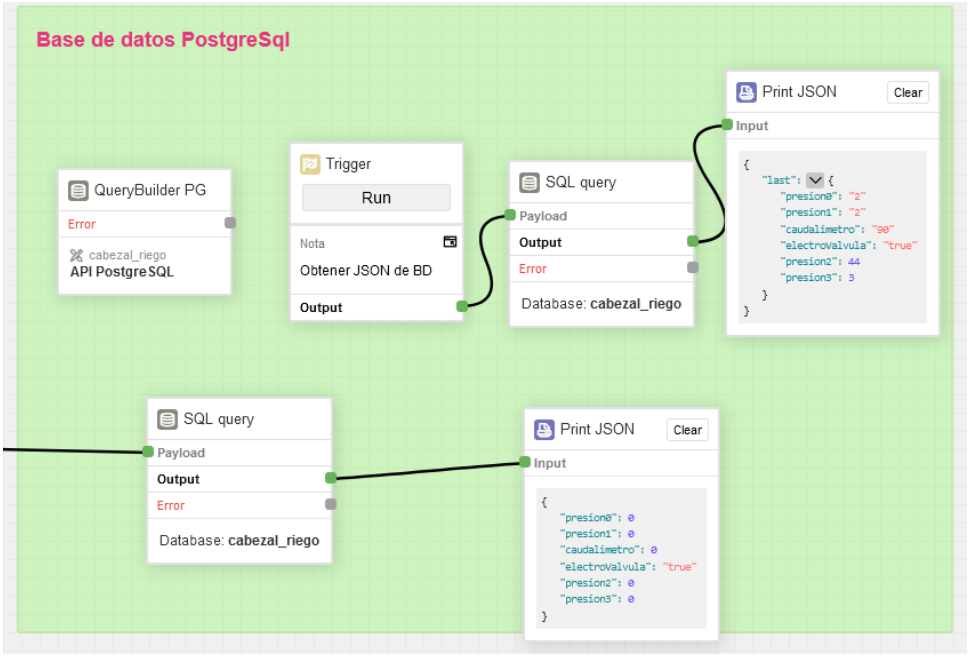

Figura 5.6: Reglas para postgreSQL en Total Flow.

El bloque de la base de datos de PostgreSQL es bastante sencillo. Por un lado se tiene el QueryBuilder PG que se encarga de mantener toda la configuración con la base de datos que se tiene en Adminer. Por otro lado, se tiene un trigger por si se quisiera recibir alguna query de la BBDD para ver los últimos valores ingresados. En la parte inferior, como se puede observar, viene la conexión de la izquierda del bloque de MQTT. Todos los datos pasan por la SQL query y son insertado mediante un INSERT INTO en la tabla de la base de datos.

Por último, se tiene el bloque de Telegram. El receiver se encarga de mandar todo texto hacia el switch. Además, se manda la información que se ha recibido por parte del bot hacia el print. El switch filtrará la información de tal forma que si lo que se ha enviado por parte del usuario, comienza con un '/', le permitirá avanzar hacia el siguiente switch. De cualquier otra forma, será descartado pues ese mensaje carece de funcionalidad dentro del sistema. Una vez se encuentra en el segundo switch, dependiendo del comando que haya llegado, se ejecutará uno de los 4 Code que se puede observar. Cada uno de ellos envía una notificación a Telegram, por este motivo todos se dirigen hacia el Sender de Telegram. Los comandos de abrir y cerrar electroválvula se envían al Publish pues irán a parar al microcontrolador y este se encargará de tomar una decisión dependiendo de lo que haya recibido.

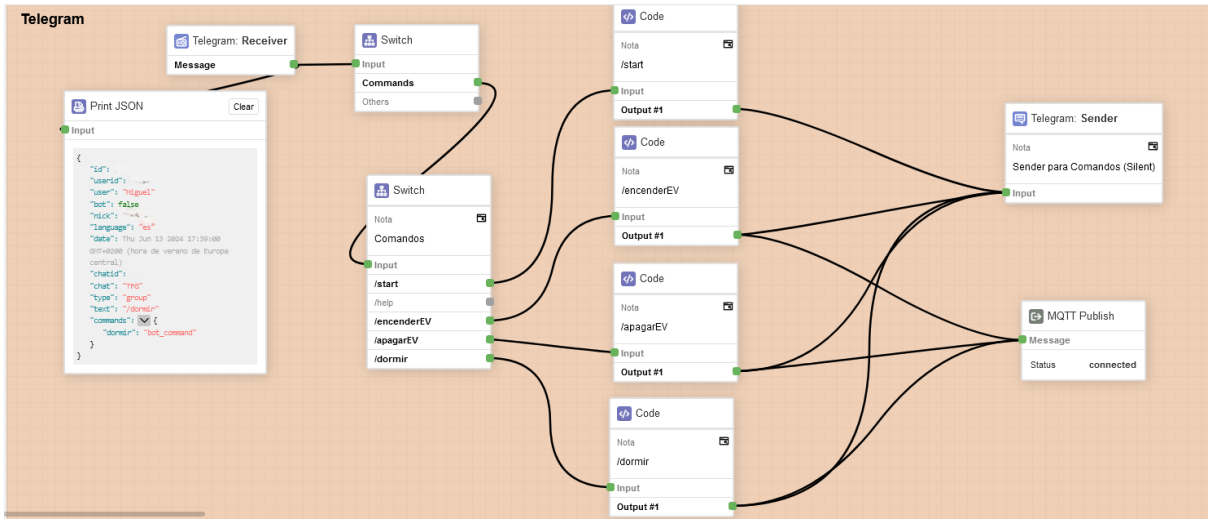

Figura 5.7: Reglas para Telegram en Total Flow.

2. **Grafana**
   Ha sido posible elaborar un dashboard que cumpla con las requisitos del usuario. En él se pueden observar todos los datos de las presiones emparejadas y del caudalímetro. Acompañando a sus valores, aparece una gráfica que va mostrando el histórico de los valores de cada medida. Por último, en el centro se tiene una imagen del sistema con los valores de cada uno de los sensores y dos botones para poder encender y apagar la electroválvula. Esto último acompañado de un true o un false dependiendo del estado en el que se encuentre la electroválvula en ese momento. Para ello se realiza una consulta a la BBDD que informe del último estado de la electroválvula. Todo lo demás sigue igual que en la sección de desarrollo.

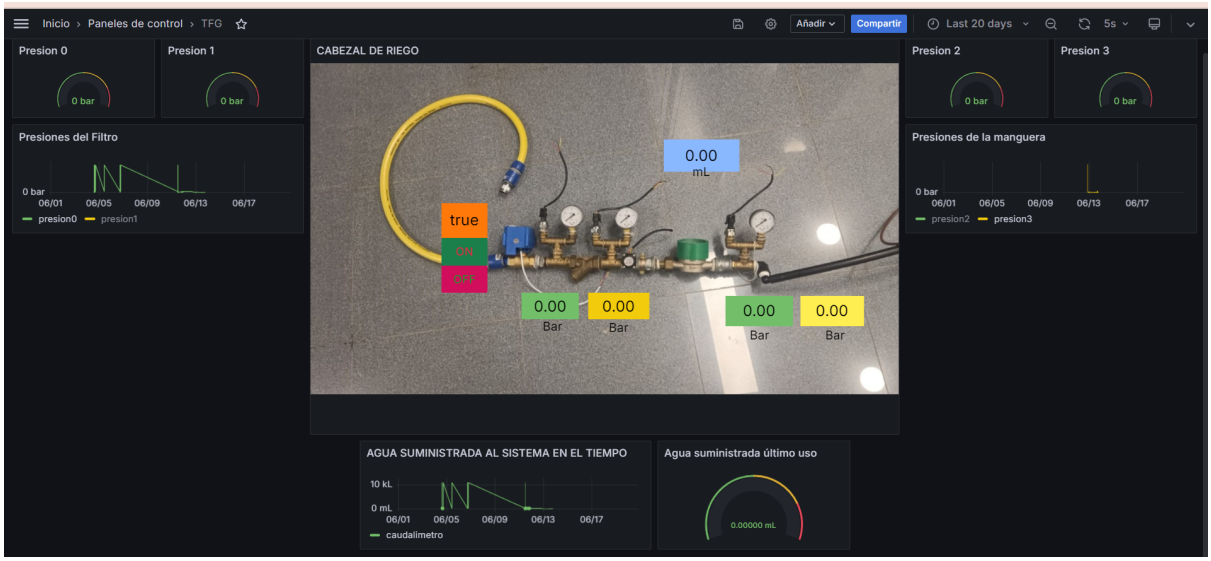

Figura 5.8: Dashboard del usuario en Grafana.

3. **Base de datos de PostgreSQL**
   La base de datos cumple con su cometido. Es extremadamente sencilla, simplemente recoge datos en una tabla llamada device y se realizan consultas desde Grafana para obtener lo que interese de esa tabla.

time ↓ =	measurement ↓ =
2024-06-13 15:40:38.611823+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:40:33.562674+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:40:28.611728+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:40:23.537779+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:40:18.636971+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:40:13.529983+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:40:08.535564+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:40:03.524273+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:39:58.591853+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:39:53.540373+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:39:48.611468+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:39:43.628832+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:39:38.516617+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:39:33.507954+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0,"presion3":0}
2024-06-13 15:39:28.600656+00	{"presion0":0,"presion1":0,"caudalimetro":0,"electroValvula":"true","presion2":0.361612,"presion3":0…
2024-06-13 15:37:36.952362+00	{"presion0":0,"presion1":0,"caudalimetro":4392,"electroValvula":"true","presion2":0,"presion3":0}

Figura 5.9: Registros en la base de datos.

4. **Telegram**
   El bot de Telegram funciona correctamente. Se puede abrir y cerrar la electroválvula desde su interfaz. A parte, se reciben perfectamente las notificaciones que alarman de un problema de las presiones o de haber llegado al máximo recomendado de regado. En última estancia, se consiguió desarrollar otro comando para dormir al sistema y que entre en modo de Deep Sleep.

Figura 5.10: Chat con el bot de Telegram.

5. **Red MESH**
   La red formada por los 3 nodos a partir del protocolo ESP-MESH-Lite, se ha desarrollado satisfactoriamente. Funciona según lo esperado, los tres nodos son capaces de conectarse a internet a través de solo uno de ellos. Haciendo que las distancias entre los componentes puedan ser considerables. Además, se puede unir a la red cualquier otro dispositivo. Motivo por el cual las MESH de este tipo son muy poderosas, pues consiguen habilitar la cobertura a aparatos en sitios que posiblemente no se pudiera. En adicción, la red es reajustable. Si uno de los nodos desaparece de la red, los demás se auto reconfigurarán de la mejor forma posible. Sumado a esto, cuando la red se levanta, inicialmente será root el que se conecte primero. Después, el que tenga mejor RSSI con el punto de acceso, se convertirá en root y los demás se reajustarán.

6. **OTA**
   La OTA funciona correctamente poniendo en ejecución a dos servidores distintos. Se consigue actualizar el sistema en segundos y se evitan pérdidas de tiempo humano y material. Para las pruebas que se realizaron en el sistema, se configuró inicialmente sin posibilidad de encender o apagar la electroválvula. El código actualizado permitía esta función.

7. **Deep Sleep**
   Se ha conseguido gestionar este modo en el microcontrolador, logrando grandes ahorros en consumos energéticos. Ha sido un acierto implementar esta función pues el sistema pasa de una media de 8 watios a un número cercano a los 0 watios. Teniendo en cuenta que el sistema de riego lo normal sería tenerlo una o dos horas al día funcionando y dejarlo sin operar las demás, es un gran ahorro.

8. **PCB del sistema**
   La PCB que se diseñó y construyó, funciona correctamente con los componentes añadidos. Con ella se consigue ahorrar en espacio, complejidad y costes asociados.

Capítulo 6

Conclusiones

El trabajo ha finalizado de manera satisfactoria, habiendo superado cada uno de los hitos que se plantearon al principio del proyecto. El cabezal de riego ha cumplido con las expectativas con las que se iniciaron, además de cumplir con los objetivos de la agenda 2030 relacionándose con los Objetivos de Desarrollo Sostenible (ODS) [41], específicamente con el ODS 6 (Agua limpia y saneamiento) y el ODS 12 (Producción y consumo responsables). Este proyecto contribuye directamente a la gestión eficiente de los recursos hídricos (ODS 6) al reducir el uso excesivo y el desperdicio de agua en la agricultura. Además, fomenta prácticas de producción más sostenibles (ODS 12) al optimizar los procesos agrícolas y alentar el uso racional de los recursos naturales. Como resultado de todo el trabajo realizado, se ha obtenido un sistema completo y consolidado, capaz de gestionar de forma óptima el agua en los cultivos, quitando trabajo a las personas gracias a la automatización de todas las funciones del sistema.

Líneas futuras:
Se es consciente de que varios puntos del sistema pueden mejorar. Una de estas mejoras podría ser una base de datos más robusta. La base de datos que forma parte del proyecto, cumple con su función, pues la información que guarda y la forma en que lo hace, son suficientes para el buen funcionamiento del sistema. En cambio, podría tratarse de una base de datos más compleja, formada por más de una tabla en la que se relacionasen mediante los IDs de los distintos microcontroladores. Otra función que podría ser ampliada sería el dashsboard de Grafana. Podrían añadirse nuevos paneles para obtener nuevas mediciones o ver la información de otra manera que tal vez sea más estética para el usuario. Por último, uno de los puntos desarrollados a los que mayor partido se podría sacar es a la red MESH. Al final este proyecto ha servido para ilustrar el funcionamiento de un cabezal reducido. Sin embargo, sería interesante poder desplegar este sistema de una manera más amplia, utilizando docenas de nodos, cubriendo una gran zona de cultivo.

Por otro lado, se podría en un futuro, ampliar las funcionalidades que tiene el sistema. Las principales ideas serían la introducción un módulo 4G para no tener que depender de un punto de acceso de otros dispositivos. Con este módulo, que no deja de ser una tarjeta SIM que dará conexión a los microcontroladores, se conseguiría una estabilidad en la red, minimizando costes en la red y simplificando el sistema. Por otro lado, tras el desarrollo del proyecto, aunque la placa elaborada ha cubierto las funciones del sistema, el diseño y construcción de una nueva PCB de una forma más profesional, podría tener mejores rendimientos para el sistema. En ella, se

introduciría el módulo 4G y se crearían los módulos para el USB y la UART del microcontrolador, de esta forma se evitaría meter la placa de desarrollo y solo se usaría el chip. Así la placa tendría un menor tamaño y mayor eficiencia.

Bibliografía

[1] "Cambio climático, la mayor amenaza para la humanidad." https://www.telefonica.com/es/sala-comunicacion/blog/cambio-climatico-amenaza-humanidad/.

[2] "El cambio climÁtico y la agricultura: Como nos afecta." https://www.utw.es/el-cambio-climatico-y-la-agricultura-como-nos-afecta/#:~:text=El%20cambio%20clim%C3%A1tico%20y%20la%20agricultura%20est%C3%A1n%20estrechamente%20relacionados%20ya,de%20temperatura%20y%20las%20sequ%C3%ADas.

[3] "Cambio climático y patrones de precipitación." https://adaptecca.es/recursos/buscador/cambio-climatico-y-patrones-de-precipitacion-efecto-sobre-

[4] "Sequía en África oriental y asia." https://www.oxfam.org/es/sequia-en-africa-oriental-si-no-llueve-pronto-no-vamos-sobrevivir.

[5] "La sequía de 2023 generó perdidas de 5.500 millones de euros." https://www.iagua.es/noticias/redaccion-iagua/sequia-2023-genero-unas-perdidas-economicas-espana-5500-millones-segun-aon/.

[6] "La sequía se agrava en españa con los embalses en niveles agónicos." https://climatica.coop/cinco-noticias-6-octubre-2023/.

[7] "Estrés térmico plantas – guía para el agricultor." https://nevianfertilizantes.com/blog/estres-termico-plantas/.

[8] "Wikipedia evapotranspiración." https://es.wikipedia.org/wiki/Evapotranspiraci%C3%B3n.

[9] "El suelo, el gran perjudicado." https://www.ashestolife.es/el-suelo-el-gran-perjudicado-tras-los-incendios/.

[10] "Los 10 avances tecnológicos más importantes en la industria agrícola." https://bloglatam.jacto.com/avances-tecnologicos-en-la-en-la-agricultura/.

[11] "Grafana." https://grafana.com/.

[12] "Riego inteligente: ¿la revolución del futuro?." https://www.agroptima.com/es/blog/riego-inteligente//.

[13] "Aqua control c4099n programador de riego para jardín." https://www.amazon.es/AQUA-CONTROL-Programador-Riego-C4099N/dp/B0166L5CGS/ref=asc_df_B0166L5CGS/?tag=googshopes-21&linkCode=df0&hvadid=

301426493190&hvpos=&hvnetw=g&hvrand=6566445217780297590&hvpone=
&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1005493&
hvtargid=pla-350013145425&mcid=91ad5a2cbeeb3fa2af247a610938af92&
th=1.

[14] "Sistema de riego bresser smart garden smart home." https://www.optical-systems.es/Sistema-de-riego-BRESSER-Smart-Garden-Smart-Home/7510100.

[15] "Sensor del suelo bresser para el sistema de riego 7510100 / 7510200 smart garden smart home." https://www.bresser-iberia.es/Sensor-del-suelo-BRESSER-para-el-sistema-de-riego-7510100-7510200-Smart-Gar
7910102.

[16] "Sensor control de riego smart wifi." https://https://www.leroymerlin.es/productos/sensor-control-de-riego-smart-wifi-90212853.html.

[17] "Esp-wifi-mesh." https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/esp-wifi-mesh.html.

[18] "Luis llamas: pines del esp32-s3, 2023." https://www.luisllamas.es/que-pines-puedo-usar-esp32-s3/.

[19] "Easyeda." https://easyeda.com/es.

[20] "Kicad." https://www.kicad.org/.

[21] "Andrés roldán, construcción de una pcb." https://assets.aon.com/-/media/files/aon/reports/2024/climate-and-catastrophe-insights-report.pdf.

[22] "Tubería de polietileno de baja densidad." https://cablematic.com/es/productos/tuberia-de-polietileno-de-baja-densidad-20-mm-25-metros-JA228/.

[23] "Manómetro de 4 bares." https://www.ma-gaco.es/MANOMETRO-GLICERINA-63mm-ROSCA-SUPERIOR-1/4-4-Bar.

[24] "Contador de agua." https://www.genebre.es/271-contadores-de-agua.

[25] "Electroválvula toro ez-flo-ezp-02-54." https://www.amazon.es/gp/product/B01MS7H1ZP/?th=1.

[26] "Sensor de presión - dc 5v g1." https://www.amazon.es/gp/product/B098R8VRRY.

[27] "Caudalímetro - yf-s201." https://www.amazon.es/gp/product/B07MY24YFX.

[28] "Espressif, analog to digital converter (adc)." https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32s3/api-reference/peripherals/adc.html.

[29] "Introduction to curve fitting." https://www.baeldung.com/cs/curve-fitting#:~:text=Curve%20fitting%20is%20the%20process,remove%20noise%20from%20a%20function.

[30] "Visual studio code." https://code.visualstudio.com/.

[31] "Espressif systems: Esp32-s3." https://www.espressif.com/en/products/socs/esp32-s3.

[32] "Guía esp-idf." https://moodle.upm.es/titulaciones/oficiales/pluginfile.php/11353021/mod_resource/content/17/Guia_ESP-IDF-v2.pdf.

[33] "Guía esp-idf." https://docs.espressif.com/projects/esp-idf/en/v3.2.2/get-started/index.html.

[34] "Node.js." https://nodejs.org/en.

[35] "Mqtt client emqx." http://mqtt-client.emqx.com/#/recent_connections.

[36] "Difference between esp-mesh-lite and esp-mesh." https://github.com/espressif/esp-mesh-lite/blob/master/components/mesh_lite/User_Guide.md#difference-between-esp-mesh-lite-and-esp-mesh.

[37] "Esp-wifi-mesh cifrado." https://github.com/espressif/esp-mesh-lite/blob/master/components/mesh_lite/User_Guide.md#further-notes.

[38] "Esp-iot-bridge component." https://components.espressif.com/components/espressif/iot_bridge.

[39] "Tutorial sensor de flujo de agua." https://naylampmechatronics.com/blog/47_tutorial-sensor-de-flujo-de-agua.html.

[40] "Modos de funcionamiento y consumos." https://moodle.upm.es/titulaciones/oficiales/pluginfile.php/11353024/mod_resource/content/14/Gestio%CC%81n%20de%20consumo%20en%20ESP32%20v4.pdf.

[41] "Objetivos de desarrollo sostenible." https://www.educo.org/blog/que-son-los-17-objetivos-de-desarrollo-sostenible?utm_source=google&utm_medium=cpc&utm_campaign=educo_brand_dsa&utm_term=kw&utm_content=text&tc_alt=64115&n_o_pst=n_o_pst&n_okw=___c_76410967186&gad_source=1&gclid=CjwKCAjwydSzBhBOEiwAj0XN4NrsHhojEFMx1YeCR0iZ46hcas9VAY2HRByBpNoOXntQbWliFeuDBwEl.