



CAMPUS
DE EXCELENCIA
INTERNACIONAL



POLITÉCNICA

"Ingeniamos el futuro"

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Desarrollo de un sistema automático para el
acondicionamiento de nutrientes en aguas de riego

Autor: Fernando Martí de Mateo

Director: Jorge Dávila Muro

MADRID, ENERO 2019

AGRADECIMIENTOS

*xAHft1JaarmGx8WG1g4fR35al3OGjwveMiM+3Ua4GeuTTqDtVYkMwHlgf37sQOF4c
1NLk5Gh60/shu3A4gxuqIrM8j4qHG3xWY7a6DgSwOeEPidaGhV6PHjKPCYhoJOAn9
xt0Y4iKqAobSR4ZWmZouGFIkbjFvzaM/VAmFyUpaFwM8CBYREE8xQlAUnGB5cq/8
NNhqU5Dmzr0Rq4/7cJfiK5zXKOK9diOKyVr6VA6v3CPoNX4S6gD4GLrYxLqBgQYlW
+mViOwyJtUQXGoG42sy6CdBxu9rhsMjnmpjD4NiAZVVx3c3wqnPXE3hL4zEOk8fz0B
X9vGJflkHPxGFPIjOK+t958S6qZR0LYe6ZXZjHx7/5bX4HInsB5NfPs/qJBtZxSW9PQ6
6AwL89WDUcKzhze5x+z2UvAlzWKYCDwdy9QX2wMQNuCc84ich9IQ43sAwrvCh/EX
bdQqQAfyll16WRfVn0a7jV42Z1n7A60DYKO2ZZaD03O26iKy4gvSJLS82QHDq8TWri
zOLpL6wrukdGqBtvX03cEgGYBT+GWuM+E0bGXCiT19clxz5t+2I9uEMlvSWeU39R2
0UFWZ2EixxrDFktgdHshtzDSjnT5NOMq95scrk7Fe0gfOpHZUs5Q1IbPr7yVTDEzfAn
TY5zVgGT5uTQFSDGnj2Io5a6YV9nykpc7G9Bpj2Evang9oqbGw+VMmmXi1xRAZxqT
zRnhuDT+z3YbPbizijvxPvLoNe/OdMbdsIoKLYUt1aL4caK0K4rGq9SFyZkYoJgzLxFG/c
5+ohppdmd0YcWgmzfGGzHMYSn6NzlChbykc5Xe3+QCI/WMcAfeKIXaiLp7CetpEXh3
Ig8JYK40wbH3kCajG8uhv5d3TljsG7j75pspoI8jll005+00ECPi+AqO4sWJGmcjJR4fq
+tOPpcrwsqbmft4T4OrhOUMnFcG/DxlX2j6bSnbTgYs6aLIENa5INp/6idp4L2pnkXkZT
0pv6MWNfSQdh7OaivAzrrqW+khcTbCpWlU4q7DC01QiAQinudZYEy+j+GxBkpfss38
gxxHnLIC6eWpTwzNth9DT24DbLieTMIRYroLy/hzCghR3Vw/1Fddq/wgSYy/HU+SNuU
m5wzK/LqXaJI6BKUXo/6HM5W3Ppy8Gngp/VKDbjklno2+gVFK+c+ulDv2UmeQJaD
ceYdfx/kG3hTFE78iBa779Jhjd+MWBrrduMJWjdC0N2iCud8ArOMM7eYqqg6K0txtCf
EQ4YXKwDBnYvYZ9QbrmF1qDC0yKeArzL8uKQfUyskxgyIenkSrkh8ADkZj+qEY/YNa
1wpvbUj0xEL135cfpjASf8Me6TfWnG+kpEZKqlzq+oRDyOmaw8Ma26+Rjm7q0CWjb
CgbCk0BYHuls13RtKzR5RqStK8GuK6ayy5MrjvivU0yB2F3yCtfZAFO8U/HLAdjOm4/S
YJlYuk70rArIj3DtjMiG3z+6ED4ws99nf7to53S4QPULM1aHjCPrPctuAxtma4pDgmEjs
QCT7CGgYRNsBED1oaKojn+8WJp0mksoRy20oyGhC/WZnC5kSyb7hNJ18ZUsOiaoti7
i5oTt/a2KoMzv+0WlC3RhJRK94y5cUwJ9v4PZsiggehfpazWBBDP757EXG9XohMNHBS
16L14MGGWD/E+JFlzQafLNQDPXl+QQi/78oXjx0ozj3yuPC3ci8rC3CYaBAOU1sc0c
EUV1a2j+BV//jL3Tkmj91l5NB9/JKaCp37rk8j+kA6QAcsAxURCq+YCF8RfeS0qsj9gL
L+yiSG+jk8rZFaD4kBiicywtFVmTXdR8qS/ISnnpNfRFtM8eNk3ZSa2Yd/ZXGuJlh84gT
cJEt1vWuB0dzvGBTlHRI2fIOvIlgewURox55/v16anCIg3AsTBye8TUqYVVeBkaKvcG8r/
WjlxSYGBa/5ik0tjbG7forFteU/r+kp1XrLlqtwRqa8cZ4Fl6BhuECDSx7/tREhEeagSvQ
pXvMJEiKCF+DoUWmBKnyrumw4iEb8BYa35hcA8b+7b3v1q13drIzXCXOTXMUB7s
cBZePJ7q12ybBqUp5lJL7h4f58=*

(Mensaje cifrado con el algoritmo AES con una clave de 96 bits, y posteriormente codificado en Base64).

TABLA DE CONTENIDOS

1. INTRODUCCIÓN Y OBJETIVOS.....	I
2. DESARROLLO	III
2.1. MÓDULO 1: DE MICROCONTROLADORES, SENSORES Y SERVOMECANISMOS	iii
2.1.1. INTRODUCCIÓN	iii
2.1.2. DISPOSITIVOS Y TECNOLOGÍAS UTILIZADAS.....	iv
2.1.2.1. SENSORES, SERVOMECANISMOS Y MICROCONTROLADORES	iv
2.1.2.2. MARCO DE TRABAJO	vi
2.1.3. ARQUITECTURA DE RED	vi
2.1.3.1. DISEÑO Y TOMA DE DECISIONES.....	vi
2.1.3.2. DISEÑO DE LA INTERFAZ DE PROGRAMACIÓN DE APLICACIÓN DEL SERVIDOR.....	viii
2.1.3.3. SEGURIDAD	xiv
2.1.4. TIPOS DE MICROCONTROLADORES	xvi
2.1.4.1. MICROCONTROLADOR PRINCIPAL	xvi
2.1.4.1.1. DEFINICIÓN Y CONSIDERACIONES.....	xvi
2.1.4.1.2. DIAGRAMA DE FLUJO	xvii
2.1.4.2. MICROCONTROLADOR SECUNDARIO DE SENSORES BÁSICOS.....	xviii
2.1.4.2.1. DEFINICIÓN Y CONSIDERACIONES.....	xviii
2.1.4.2.2. DIAGRAMA DE FLUJO	xix
2.1.4.2.3. ESQUEMA ELECTRÓNICO.....	xx
2.1.4.3. MICROCONTROLADOR SECUNDARIO DE CONTROL DE RIEGO Y NEUTRALIZADOR DE PH xxi	
2.1.4.3.1. DEFINICIÓN Y CONSIDERACIONES.....	xxi
2.1.4.3.2. DIAGRAMA DE FLUJO	xxiv
2.1.4.3.3. ESQUEMA ELECTRÓNICO.....	xxv
2.2. MÓDULO 2: MIDDLEWARE.....	xxvi
2.2.1. INTRODUCCIÓN	xxvi
2.2.2. ARQUITECTURA DE RED	xxvii
2.2.3. DISEÑO DE LA INTERFAZ DE PROGRAMACIÓN DE APLICACIÓN DEL SERVIDOR	xxix
2.2.4. SEGURIDAD	xxxiv
2.2.5. DIAGRAMA DE FLUJO.....	xxxv

2.3.	MÓDULO 3: SISTEMA WEB BASADO EN UN ENTORNO <i>CLOUD</i>	xxxvi
2.3.1.	INTRODUCCIÓN	xxxvi
2.3.2.	MARCO DE TRABAJO Y TECNOLOGÍAS	xxxvi
2.3.3.	ARQUITECTURA WEB	xxxvii
2.3.4.	SISTEMA WEB	xxxix
2.3.4.1.	APLICACIÓN CLIENTE WEB	xxxix
2.3.4.1.1.	DEFINICIÓN Y CONSIDERACIONES	xxxix
2.3.4.1.2.	MÓDULO DE GESTIÓN DE USUARIOS	xxxix
2.3.4.1.3.	MÓDULO DE GESTIÓN DE CULTIVOS	xl
2.3.4.2.	SERVIDOR WEB	xli
2.3.4.2.1.	DEFINICIÓN Y CONSIDERACIONES	xli
2.3.4.2.2.	DISEÑO DE LA INTERFAZ DE PROGRAMACIÓN DE APLICACIONES DEL SERVIDOR	xlii
2.3.4.2.3.	SEGURIDAD	xlvi
2.3.4.3.	HOSTING	li
3.	RESULTADOS Y CONCLUSIONES	LII
4.	ANEXOS	LXIX
5.	BIBLIOGRAFÍA	LXX

TABLA DE ILUSTRACIONES

Ilustración 1. Arquitectura de red del sistema.	ii
Ilustración 2. Diseño de la arquitectura de red del módulo 1.	viii
Ilustración 3. Diagrama de flujo del microcontrolador principal.	xvii
Ilustración 4. Diagrama de flujo del microcontrolador secundario de sensores básicos.	xix
Ilustración 5. Esquema electrónico del microcontrolador secundario de sensores básicos.	xx
Ilustración 6. Diagrama de flujo del microcontrolador secundario de control de riego y neutralización de pH.	xxiv
Ilustración 7. Diagrama electrónico del microcontrolador secundario de control de riego y neutralización de pH.	xxv
Ilustración 8. Diseño de la arquitectura de red del módulo 2.	xxviii
Ilustración 9. Diagrama de flujo del middleware.	xxxv
Ilustración 10. Diseño de la arquitectura de red del módulo 3.	xxxviii
Ilustración 11. Microcontrolador principal y microcontrolador secundario de control de riego en una protoboard.	lii
Ilustración 12. Microcontrolador secundario de control de riego en el soporte construido.	liii
Ilustración 13. Vista general del soporte con las botellas, sensores, motores, pesos y electroválvulas.	liv
Ilustración 14. Vista frontal superior del soporte construido.	lv
Ilustración 15. Código fuente del microcontrolador principal, con 1353 líneas.	lvi
Ilustración 16. Código fuente del microcontrolador secundario de sensores básicos, con 348 líneas.	lvii
Ilustración 17. Código fuente del microcontrolador secundario de control de riego, con 582 líneas.	lviii
Ilustración 18. Carpeta de desarrollo del microcontrolador principal, usando Windows Subsystem for Linux.	lix
Ilustración 19. Ejemplo de comunicación automática entre los componentes del módulo 1.	lix
Ilustración 20. Microcontrolador del middleware conectado al módulo Sim900a mini, junto a una Raspberry Pi.	lx
Ilustración 21. Código fuente del middleware, con 1225 líneas.	lxi
Ilustración 22. Ejemplo de comunicación automática entre el módulo 1 y 2, cuando se envían peticiones TCP al módulo 2, estando conectado a la red local del módulo 2.	lxii
Ilustración 23. Formulario de inicio de sesión del módulo de gestión de usuarios de la aplicación web.	lxiii
Ilustración 24. Formulario de registro del módulo de gestión de usuarios de la aplicación web.	lxiv
Ilustración 25. Componente del módulo de gestión de cultivos de la aplicación web.	lxv
Ilustración 26. Envío de mensajes entre todas las partes al añadir un cultivo.	lxvi
Ilustración 27. Envío de mensajes entre todas las partes, al realizar la acción de regar.	lxvi

Ilustración 28. Token JWT en el almacenamiento local del navegador.	lxvii
Ilustración 29. Petición automática de mensajes de método "check_system", para actualizar los datos de monitorización del cultivo.	lxvii

RESUMEN

El objetivo de este proyecto consiste en diseñar y desarrollar un sistema automático, basado en los conceptos: el internet de las cosas y la inteligencia artificial; que permita monitorizar y auto mantener un cultivo en una maceta, de forma remota.

Para la monitorización y mantenimiento del cultivo, se utilizarán sensores varios, con los cuales se medirán las condiciones ambientales primarias condicionantes para el desarrollo del cultivo, y en función de estas mediciones, el sistema deberá realizar las funciones necesarias mediante microcontroladores y servomecanismos, con el objetivo de que el cultivo de la maceta pueda crecer y desarrollarse en unas condiciones favorables, con la mínima intervención posible de un humano.

El mencionado sistema, tendrá la posibilidad de ser monitorizado de manera remota. Esto será posible mediante la creación de una plataforma web en un entorno *cloud*, que incorporará un mecanismo de alta e inicio de usuarios, con el cual los usuarios podrán entrar en la plataforma, para poder gestionar y monitorizar sus cultivos e incluso, tendrán la posibilidad de ordenar ciertas acciones a sus cultivos, en remoto.

Adicionalmente, este sistema deberá contar con unos mínimos mecanismos de seguridad, con el objetivo de mantener la integridad, privacidad y autenticidad de la información.

SUMMARY

The aim of this project consists in design and develop an automatic system, based on the concepts: the internet of things and the artificial intelligence. Allowing to monitor and self-maintain a harvest in a plant pot, remotely.

For the monitoring and the maintenance of the harvest, several sensors will be used, to monitor the environmental conditions for the development of the harvest, and based on these measurements, the system must perform the necessary functions by the use of microcontrollers and servomechanisms, allowing the development of the harvest in favorable conditions, with the minimum human intervention possible.

The aforementioned system, will have the possibility of being monitored remotely. This will be possible through the creation of a web platform in a cloud environment, which will incorporate a registration and users initiation mechanism, where users can enter the platform to be able to manage and monitor their harvests. Also, they will have the possibility of ordering certain actions to their crops, remotely.

Additionally, this system must have minimum security mechanisms, with the objective of maintaining the integrity, privacy and authenticity of the information.

1.INTRODUCCIÓN Y OBJETIVOS

En este capítulo, se explicarán las diferentes fases del desarrollo del proyecto y sus motivos y necesidades. Finalmente, se mencionarán los objetivos principales que se han perseguido durante el transcurso del desarrollo del proyecto.

Para este proyecto, se ha de desarrollar un sistema autónomo con sensores, servomecanismos y microcontroladores, para monitorizar y permitir el desarrollo de un cultivo en una maceta. Adicionalmente, este sistema se podrá monitorizar y gestionar de manera remota. Por esto, se ha visto la necesidad de estructurar las fases del proyecto en tres:

- La primera parte, es la que se encarga de la monitorización y del mantenimiento del cultivo. En esta fase se explicará: qué sensores y servomecanismos se han utilizado y la función de cada uno, el microcontrolador que se utiliza para interactuar con los diferentes dispositivos y de qué manera, qué tipos de microcontroladores hay y cuál es la función de cada uno y finalmente, cómo se comunican entre sí todos los existentes microcontroladores.
- La segunda parte, surge como necesidad de que el sistema sea totalmente automático y que simultáneamente también pueda ser remoto. Esta fase consiste en un sistema intermediario, o *middleware* [1], que permita la comunicación entre los microcontroladores de la primera fase, con un servidor, correspondiente a la tercera fase; de esta manera, el sistema puede ser remoto. Para esta fase, se utiliza un microcontrolador conectado a un módulo, que proporciona acceso a la red GSM/GPRS [2] mediante una tarjeta SIM [3]. Así, el usuario no necesita realizar ningún desarrollo previo para que su cultivo pueda ser gestionado remotamente.
- Finalmente, la tercera parte explica el sistema remoto web, en donde: un usuario puede darse de alta y acceder al sistema, y monitorizar y gestionar sus cultivos. Para la realización de esta fase, se ha considerado montar un sistema basado en un entorno *cloud* [4], de manera que se pueda utilizar un navegador web convencional para el acceso. Para conseguir este sistema web, se ha tenido que diseñar y desarrollar una aplicación web usando Angular.js 7 [5] y un servidor con una interfaz de servicios, usando Node.js [6].

Estas tres partes forman en su conjunto una arquitectura de red [7], con diferentes tipos de redes y sistemas.

Para una mejor comprensión del sistema global, a continuación se muestra una ilustración, distinguiendo estas tres partes por colores, siendo el color verde el correspondiente a la primera parte, el color naranja el correspondiente a la segunda parte

y los colores azul y morado, los correspondientes a la aplicación cliente y el servidor de la tercera parte, respectivamente.

A estas tres fases descritas, nos referiremos a ellas en adelante, como módulo 1, módulo 2 y módulo 3, para las partes 1, 2 y 3, respectivamente.

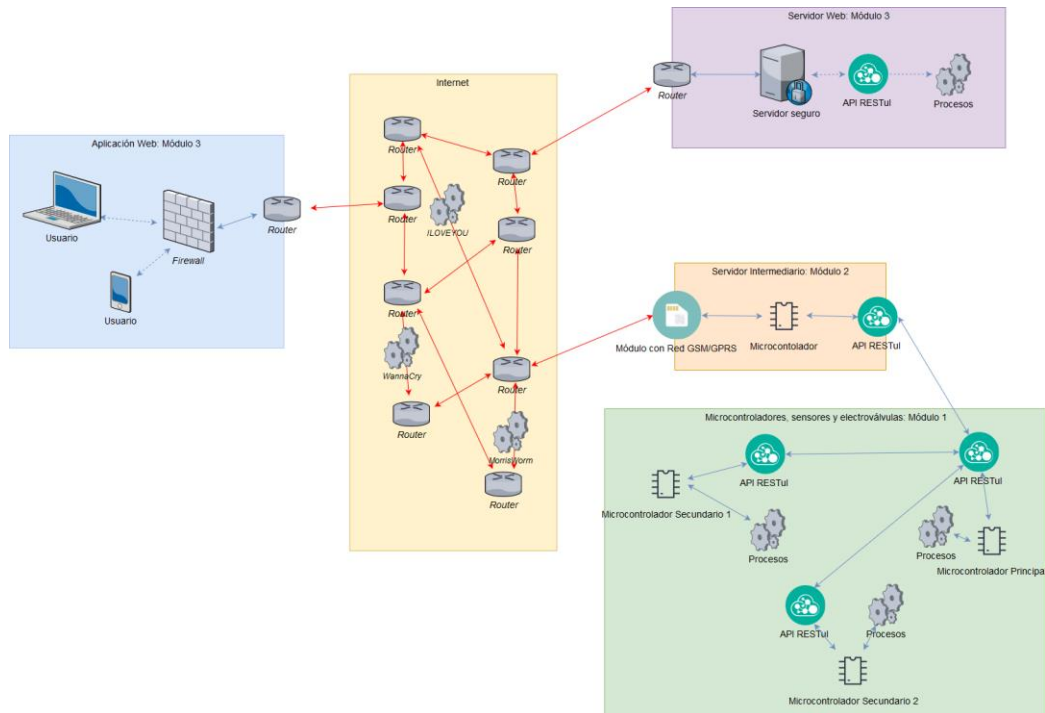


Ilustración 1. Arquitectura de red del sistema.

Esta ilustración, nos permite hacernos una idea general de la comunicación entre todas las partes existentes, que se detallará en profundidad en cada módulo.

Ya explicado el sistema a nivel general, a continuación se listarán los objetivos principales que se han pretendido conseguir con el desarrollo del proyecto:

- Aprender la informática implicada en una arquitectura de red a nivel global y la comprensión de sus partes.
- Aprender los problemas o limitaciones de usar una arquitectura de red, concretamente la seguridad informática, y aprender la informática implicada en mecanismos de seguridad de la información, en cuanto se refiere a la privacidad, autenticidad e integridad de los datos.
- Aprender las limitaciones computacionales en un microcontrolador, y la optimización de recursos que esto conlleva.
- Aprender nociones básicas de electrónica y su relación con la informática, y como la informática puede tener un impacto físico en el mundo.

2.DESARROLLO

2.1. MÓDULO 1: DE MICROCONTROLADORES, SENSORES Y SERVOMECANISMOS

2.1.1. INTRODUCCIÓN

Este módulo tiene como objetivo monitorizar las condiciones ambientales implicadas en el desarrollo de un cultivo, y según estas, realizar las acciones necesarias para que el cultivo pueda desarrollarse y crecer de la mejor manera, de forma autónoma. Las acciones desarrolladas en el proyecto son: regar y realizar una neutralización de pH [8] [9].

En este módulo, se pretende explicar a nivel detallado las decisiones de diseño durante el desarrollo del módulo 1. Para esto se han creado tres secciones:

- Inicialmente se describirá la elección de los sensores, servomecanismos y microcontroladores utilizados, así como una breve explicación del marco de trabajo y las tecnologías implicadas.
- En la sección posterior, se explicará el diseño de la arquitectura de red implicada en este módulo, y las decisiones de esta, que como un adelanto y por concienciación del lector: se utiliza un controlador principal, el cual se encarga de gestionar los datos de los sensores, enviados mediante la tecnología wifi [10], por dos microcontroladores secundarios. En diferentes apartados de esta misma sección, se especificará el diseño de los mensajes que se envían entre los microcontroladores y las vulnerabilidades y mecanismos de seguridad, relacionados con esta arquitectura.
- Finalmente, se usará otra sección en donde, por cada tipo de microcontrolador, se detallará: una definición de su objetivo y otras consideraciones, un diagrama de flujo de sus procesos, una especificación de la interfaz de aplicación de su servidor local y, en caso de ser un microcontrolador secundario: un esquema electrónico que detalle la conectividad entre este y sus sensores y servomecanismos asociados.

2.1.2. DISPOSITIVOS Y TECNOLOGÍAS UTILIZADAS

2.1.2.1. SENSORES, SERVOMECANISMOS Y MICROCONTROLADORES

La lista de sensores que se utiliza para la monitorización de las condiciones del cultivo es la siguiente:

- 1 x sensor de humedad FC-28 [11].
- 1 x sensor de lluvia HL-83 [12].
- 1 x sensor de temperatura y humedad DHT-22 [13].
- 1 x medidor de caudal de agua YF-S201 [14].
- 2 x sensor de peso HX-711 [15].
- 1 x fotorresistencia GL-5539 [16].
- 1 x sensor pH con sonda de electrodo pH [17].

Se han elegido estos sensores por su bajo precio: 0.99\$/unidad (sensor de humedad FC-28) [18], 1.55€/unidad (sensor de lluvia HL-83) [19], 3.01\$/unidad (sensor de temperatura y humedad DHT-22) [20], 3.81\$/unidad (medidor de caudal y agua YF-S201) [21], 3.12\$/unidad (sensor de peso HX-711) [22], 0.094\$/unidad (fotorresistencia GL-5539) [23] y 14.59\$/unidad (sensor y sonda de electrodo pH) [24].

El sensor de humedad FC-28 se utiliza para medir la humedad relativa de la tierra del cultivo, el sensor de lluvia HL-83, para comprobar si ha llovido; el sensor de temperatura y humedad DHT-22, para medir la temperatura y la humedad relativa del ambiente; el medidor de caudal YF-S201, para determinar con cuanta cantidad de líquido se riega; el sensor de peso HX-711, para medir el volumen de ácido o de base, necesario para realizar una neutralización de pH; la fotorresistencia GL-5539, para determinar la iluminación relativa del ambiente; la sonda de electrodo pH, para comprobar el nivel de pH del agua mediante el sensor pH.

Los servomecanismos se componen de electroválvulas y motores, se utilizan los siguientes:

- 1 x DC 12V 1/4'' marca DIGITEN.
- 2 x DC 12V 1/4'' marca U.S. Solid.

- 3 x motores DC 12V 1/4”.

Se utilizan dos tipos de electroválvulas, ya que en un inicio se compró la de marca DIGITEN (8.45€/unidad) [25], y no de muy buena calidad, se decidió probar con las de la marca U.S. Solid (13.79€/unidad) [26], de mejor calidad. Adicionalmente se utiliza un motor (8.90€/unidad) [27] con cada electroválvula, que facilita la transmisión del líquido a través de la electroválvula, ya que previamente sin el motor, el líquido no pasaba debido a la presión atmosférica.

El microcontrolador que se ha utilizado durante el desarrollo, ha sido el módulo de desarrollo NodeMCU ESP-32S [28]. Se ha escogido este microcontrolador por las características que proporciona para su bajo precio (4.97\$/unidad) [29]. Este microcontrolador es una versión superior del microcontrolador ESP-8266 [30], que con el módulo de desarrollo lo podemos encontrar por 2.78\$/unidad [31].

Las características técnicas del módulo de desarrollo del microcontrolador utilizado son las siguientes:

- Chip: Tensilica 32-bit Mono-/Dual-core CPU Xtensa LX6.
- Pines digitales I/O: 28.
- Pines analógicos de entrada (ADC): 8.
- Pines analógicos de salida (DAC): 2.
- UARTs: 3.
- SPIs: 2.
- I2Cs: 3.
- Memoria Flash: 4MB (ampliable).
- SRAM: 520KB.
- Frecuencia: 520 Mhz.
- Wi-Fi: IEEE 802.11 b/g/n/e/i.

Las características técnicas del chip ESP-8266 son:

- Pines digitales I/O: 17.
- Memoria Flash: 512KB (ampliable).
- SRAM: 32KB + 80KB.
- Frecuencia: 80 ~ 160 Mhz.
- Wi-Fi: IEEE 802.11 b/g/n.

Claramente, se ha decantado por utilizar el microcontrolador ESP-32, teniendo en cuenta el factor calidad/precio. Además, ya con el proyecto finalizado, si se escogiera el chip

ESP-8266 para el sistema, este se quedaría muy ajustado, concretamente por la capacidad de la SRAM.

2.1.2.2. MARCO DE TRABAJO

Para el desarrollo en los microcontroladores ESP-32, existen dos marcos de trabajo, desarrollados por Espressif Systems Pte. Ltd., disponibles: el oficial, denominado ESP/IDF [32], y otro que permite desarrollar usando el entorno de Arduino [33].

Para este proyecto, se ha decidido utilizar el marco de trabajo oficial, en primer lugar: porque al ser el oficial, está desarrollado exclusivamente para funcionar en el chip ESP-32, teniendo en cuenta todas sus funcionalidades, y no las funcionalidades de Arduino; en segundo lugar: de esta manera, desarrollar en este marco de trabajo permitiría utilizar en el sistema cualquier chip de la familia ESP, sin tener que modificar el desarrollo. Igualmente, cabe destacar que se pueden utilizar los dos entornos simultáneamente, pudiendo utilizar librerías escritas para Arduino [34].

Para el marco de trabajo ESP/IDF se utiliza el lenguaje C, con librerías externas de código abierto y licencia no comercial, incluidas en el entorno [35]. El entorno está alojado en GitHub, por lo que se usa el control de versiones git para su desarrollo. Aunque el entorno es reciente y aún está en desarrollo, tiene una comunidad bastante grande y volcada, con la que se puede compartir cualquier problema o mejora.

Con relación al proyecto, cabe destacar del desarrollo en el entorno ESP/IDF, que al utilizar la tecnología wifi para la comunicación entre los microcontroladores, el desarrollo ha tenido que estar basado en función de los posibles eventos wifi [36].

2.1.3. ARQUITECTURA DE RED

2.1.3.1. DISEÑO Y TOMA DE DECISIONES

En esta sección se pretende detallar la comunicación de los microcontroladores. Para ello se ha diseñado una arquitectura de red, en la que existen diferentes tipos de microcontroladores: un microcontrolador principal y dos microcontroladores secundarios. Cabe destacar el trabajo de fin de grado de Francisco Jesús Moral Parlón [37], que utiliza un diseño basado en microcontroladores maestro y esclavos.

Se ha decidido usar varios microcontroladores, meramente con fines académicos y didácticos, y entre estos microcontroladores, se diferencia entre: un microcontrolador principal y dos microcontroladores secundarios, como se ha comentado.

Para la comunicación entre los microcontroladores, se ha decidido utilizar la tecnología wifi, de igual manera: con fines didácticos; pudiéndose haber utilizado otra tecnología, como por ejemplo: bluetooth [38].

La razón por la que se utiliza un microcontrolador que sea el principal, es que al utilizar la tecnología wifi, la comunicación es asíncrona [39], por tanto este microcontrolador se encargará de recibir los mensajes de los otros microcontroladores, a medida que van enviando los datos. Por tanto, el microcontrolador principal se encargará de procesar los datos de los sensores leídos y enviados por los microcontroladores secundarios, de manera síncrona.

Se usan dos microcontroladores secundarios: uno se encarga de leer los sensores básicos, y el otro se encarga del control de riego y de la neutralización de pH, también se encarga de leer el pH y enviar el dato al microcontrolador principal, de igual manera que el microcontrolador de sensores básicos.

Para el envío de estos mensajes se usan *sockets* [40] mediante los protocolos TCP/IP [41]. El entorno de desarrollo proporciona librerías para poder enviar mensajes TCP/IP usando la tecnología wifi.

En esta arquitectura de red tenemos una red local, creada por el microcontrolador principal, en la que los dos microcontroladores secundarios son estaciones que se conectan a esta red local, mediante un SSID [42] y una contraseña predefinidos.

Resumidamente:

- El microcontrolador principal tras inicializar los servicios TCP/IP, creará un servidor DHCP [43] e inicializará el servicio wifi, con lo que creará un punto de acceso wifi e intentará acceder a la red wifi del módulo 2 (del cual se hablará en el módulo 2). Tras esto, se ejecutará un hilo que estará encargado de toda la conectividad wifi y sus eventos asociados. Paralelamente el proceso principal, iniciará un *socket* TCP en el puerto 40501, en donde se quedará escuchando llamadas entrantes. Adicionalmente, al detectar el microcontrolador principal que se han conectado a la red wifi los dos microcontroladores secundarios, se ejecutará en paralelo otro hilo que se encargará de analizar los datos recibidos de los sensores y cuando determine, enviará mediante un mensaje TCP la orden de regar o neutralizar el pH al microcontrolador secundario correspondiente, de manera autónoma.

- Los microcontroladores secundarios, de manera similar, tras inicializar el servicio TCP/IP y el servicio wifi con su hilo encargado de los eventos relacionados, se intentarán conectar a la red local del microcontrolador principal. y al conectarse, se ejecutará otro hilo encargado de: la lectura de los sensores y el envío de estos datos al microcontrolador principal. El proceso inicial, de manera concurrente, se encargará de iniciar un *socket* TCP en el puerto 40502, en donde se quedará a la espera de un mensaje TCP proveniente del microcontrolador principal.

En la sección 2.1.4. se proporciona un diagrama de flujo que muestra el funcionamiento detallado, por cada tipo de microcontrolador.

A continuación, se muestra la arquitectura de red de este módulo, mostrando las posibles fuentes y orígenes de mensajes, a nivel del protocolo TCP:

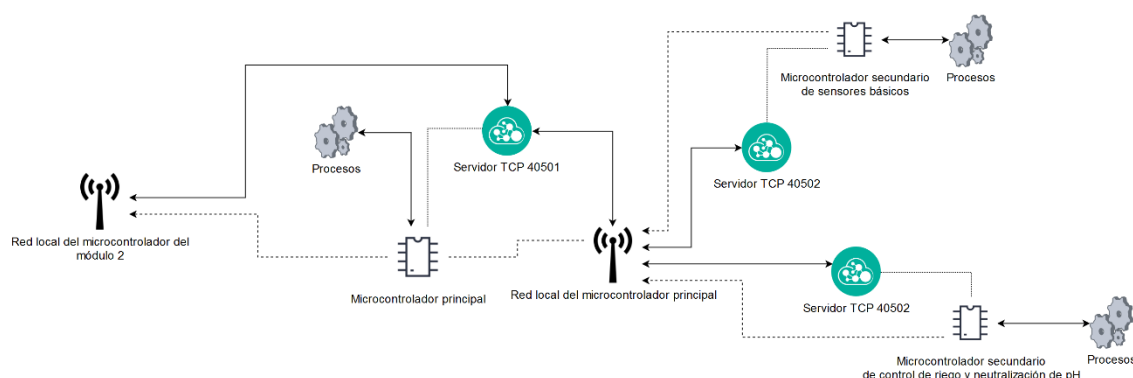


Ilustración 2. Diseño de la arquitectura de red del módulo 1.

Si la imagen no se aprecia, se puede ver la imagen original, disponible en el capítulo de Anexos.

2.1.3.2. DISEÑO DE LA INTERFAZ DE PROGRAMACIÓN DE APLICACIÓN DEL SERVIDOR

Para los mensajes TCP que se envían entre los microcontroladores, como se ha mencionado, se utiliza un servidor local. Para este servidor se ha definido una API [44], en la que para el diseño de los mensajes, se ha tenido como referencia los códigos de estado del protocolo HTTP [45] [46] y el formato JSON [47]. Dando lugar a los siguientes prototipos de mensaje, que por tipo de microcontrolador son:

- API del microcontrolador principal:

- Método “update-system”:

```
{
  "method": "update-system",
  "type": "basic_sensors",
  "properties":
  {
    "air_temperature": 25.2,
    "air_humidity": 50.2,
    "rain_sensor": 1024,
    "soil_humidity": 1024,
    "light_sensor": 325
  }
}
```

Este mensaje es enviado al microcontrolador principal, cuando un microcontrolador secundario ha leído sus sensores asociados. La propiedad “type” identifica el tipo de sensor secundario, “basic_sensors” corresponde al microcontrolador secundario de sensores básicos y “ph_controller” corresponde al microcontrolador secundario de control de riego y neutralizador del pH.

En caso de ser el microcontrolador secundario encargado del pH, la propiedad “properties” seguirá el siguiente patrón:

```
“properties”:
{
  “water_ph”: 7.01,
  “last_water”: true
}
```

- Método “get-system”:

```
{
  "method": "get-system"
}
```

Mensaje enviado por el microcontrolador del módulo 2. Este mensaje es enviado con el objetivo de obtener los valores de los sensores, que tiene almacenados el microcontrolador principal. La respuesta a este mensaje tiene el siguiente formato:

```
{
  "status": 200,
  "data":
  {
    "air_temperature": 25.2,
    "air_humidity": 50.2,
    "rain_sensor": 1024,
    "soil_humidity": 1024,
    "light_sensor": 325
    "water_ph": 7.2,
    "last_water": 4055568
  }
}
```

- Método “edit-system”:

```
{
  "method": "edit-system",
  "properties":
  {
```

```

        "objective_ph": 7.9,
        "ph_umbral": 0.1
    }
}

```

Mensaje enviado por el microcontrolador del módulo 2. Se envía este mensaje cuando se quiere modificar los parámetros del microcontrolador secundario neutralizador de pH. Este mensaje hace que el microcontrolador principal envíe el mensaje con método "update-ph" al microcontrolador secundario. Si el microcontrolador secundario modifica correctamente los parámetros indicados, responde mediante un mensaje al microcontrolador principal, con el siguiente formato:

```

{
    "status": 200
}

```

En donde 200 indica que se modificaron correctamente los parámetros. En este caso, el microcontrolador principal envía este mismo mensaje al microcontrolador del módulo 2.

- Método "send-action"

```

{
    "method": "send-action",
    "code": 0
}

```

Mensaje enviado por el microcontrolador del módulo 2. Este mensaje se envía cuando se quiere realizar una acción sobre el cultivo. Existen tres posibles acciones indicadas por el parámetro de la propiedad "code": cuando es un 0, enviará un mensaje al microcontrolador encargado del pH con la acción de regar; cuando es un 1, indicará la acción de neutralizar el pH y si es un 2, indicará realizar previamente una neutralización y posteriormente regar. Esta acción se indica mediante el mensaje con método "make-action".

- API del microcontrolador secundario de sensores básicos:

- Método “get-type”:

```
{
  "method": "get-type"
}
```

Mensaje enviado por el microcontrolador principal, cuando el microcontrolador secundario de sensores básico se conecta en la red wifi local. El microcontrolador principal necesita saber qué tipo de microcontrolador secundario se ha conectado a su red, por esto envía este mensaje.

El microcontrolador secundario le responde mediante un mensaje con el siguiente formato:

```
{
  "type": "basic_sensors"
}
```

Posteriormente, el microcontrolador principal le enviará un segundo mensaje con la clave para dar seguridad a las comunicaciones. El mensaje tendrá el siguiente formato:

```
{
  "secret": "opweRe498sd46a"
}
```

- API del microcontrolador secundario de control de riego y neutralización de pH:

- Método “get-type”:

```
{
  "method": "get-type"
}
```


Igual que el método “get-type” del microcontrolador secundario se sensores básicos. En este caso, el microcontrolador secundario le responde mediante el siguiente mensaje:

```
{  
    “type”: “ph_controller”,  
}
```

Acto seguido, el microcontrolador principal enviará un segundo mensaje indicando los parámetros por defecto que el microcontrolador secundario utilizará para realizar las acciones de riego o neutralización y la clave de seguridad:

```
{  
    “secret”: “opweRe498sd46a”,  
    “properties”:  
    {  
        “objective_ph”: 7.02,  
        “ph_umbral”: 0.1  
    }  
}
```

- Método “make-action”:

```
{  
    “method”: “make-action”,  
    “code”: 0  
}
```

Mensaje enviado por el microcontrolador principal, cuando se quiere realizar una acción en el cultivo. Como se ha comentado, la acción está indicada por la propiedad “code”, en donde 0 indica riego, 1 neutralización y 2 neutralización y riego.

El microcontrolador secundario, tras realizar la acción, enviará un mensaje al microcontrolador principal con método “update-system”.

- Método “update-ph”:

```
{
  "method": "update-ph",
  "properties":
  {
    "objective_ph": 7.02,
    "ph_umbral": 0.02
  }
}
```

Este mensaje es enviado por el microcontrolador principal, cuando se quieren modificar las propiedades que se utilizan en las acciones de riego o neutralización. El microcontrolador responde con el siguiente mensaje:

```
{
  "status": 200
}
```

Para los mensajes entre los microcontroladores, relacionados con el protocolo wifi o el protocolo IP, se encarga el propio entorno utilizado de gestionarlos, por tanto no es el objetivo de esta memoria describir estos tipos de mensajes [48] [49].

2.1.3.3. SEGURIDAD

Este módulo es considerado un módulo crítico del sistema, en donde una acción sobre el cultivo, puede influir en el desarrollo de este. Por tanto, es aquí donde la seguridad informática cobra un papel fundamental.

A continuación se listarán las vulnerabilidades encontradas inicialmente:

- Al enviarse los mensajes usando la tecnología wifi y los protocolos TCP/IP, no existe privacidad en el envío de los mensajes ya que al no estar cifrados, estos pueden ser interceptados y leídos [50].
- La red local se crea mediante seguridad WPA2 [51], actualmente este tipo de seguridad presenta vulnerabilidades, por lo que es fácilmente accesible sin necesidad de contraseña [52]; igualmente la contraseña está predefinida en el código, lo que hace al sistema más inseguro todavía.
- No está garantizado que los mensajes enviados provengan de una fuente fidedigna. Un sistema puede entrar en la red local y hacerse pasar por una fuente que no es, enviando mensajes al resto de sistemas de la red, provocando el impedimento del desarrollo del cultivo e incluso llegar a destruirlo.

Rápidamente se aprecia la gravedad de estas vulnerabilidades, en donde la solución sería: garantizar la veracidad de la fuente.

Por otra parte, el impedimento al acceso de la red local no está garantizado, pero el acceso se podría hacer más seguro de diversas maneras: filtrando el acceso a la red local mediante dirección MAC [53] o desarrollar una mejor seguridad para la creación de la red local, entre otros.

Y respecto a la privacidad, se podría ofrecer un protocolo de generación de claves compartidas usando claves públicas. De esta manera, los mensajes enviados estarían cifrados, lo que dificultaría la interceptación de estos [54].

Como esto es un proyecto amplio y general, y se dispone de tiempo limitado, la seguridad que se utilizará en este módulo se limitará a dar cierta veracidad a las fuentes de los mensajes de método “update-system”, enviados por los microcontroladores secundarios al microcontrolador principal, a modo de ejemplo del desarrollo.

Para esto, el microcontrolador principal al iniciar, generará una clave pseudoaleatoria (usando la recepción de señales de radio) [55], que enviará a los microcontroladores secundarios cuando estos se conecten a su red local, y estos utilizarán esta clave para firmar el mensaje y añadir una firma generada mediante la autenticación HMAC [56], usando la función SHA-256 [57]. Finalmente, las partes del mensaje se codificarán en Base64 [58] y separadas por un punto.

Por lo que los mensajes especificados anteriormente, seguirán el siguiente formato:

BASE64_PAYLOAD.BASE64_SIGNATURE

2.1.4. TIPOS DE MICROCONTROLADORES

2.1.4.1. MICROCONTROLADOR PRINCIPAL

2.1.4.1.1. DEFINICIÓN Y CONSIDERACIONES

Este microcontrolador es el encargado de procesar los datos de los sensores leídos y enviados por los microcontroladores secundarios, mediante mensajes TCP/IP, a través de la tecnología wifi. Para que esto sea posible, el microcontrolador principal crea una red local a la que se irán conectando los microcontroladores secundarios asociados. Adicionalmente, para permitir que el sistema sea remoto, el microcontrolador principal se conectará a la red local creada por el microcontrolador del módulo 2.

Este microcontrolador no tiene conectado ningún sensor, ni servomecanismo. Tan solo se limita a analizar los datos de los sensores y determinar qué acción disponible necesita el cultivo. Viene a ser el cerebro del cultivo.

En la siguiente sección, se muestra un diagrama de flujo con los diferentes procesos de este microcontrolador.

2.1.4.1.2. DIAGRAMA DE FLUJO

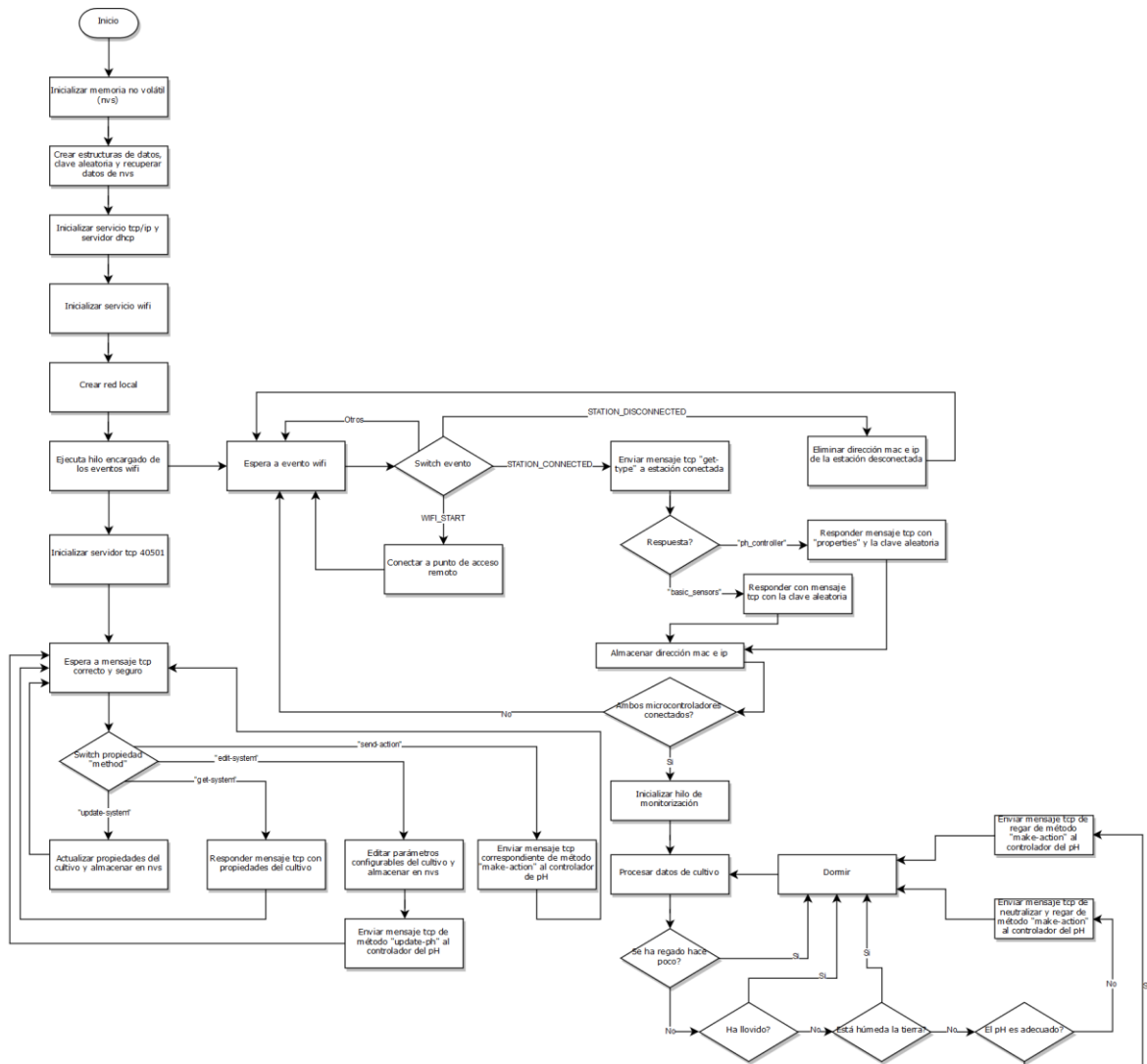


Ilustración 3. Diagrama de flujo del microcontrolador principal.

Si la imagen no se aprecia, se puede ver la imagen original, disponible en el capítulo de Anexos.

2.1.4.2. MICROCONTROLADOR SECUNDARIO DE SENSORES BÁSICOS

2.1.4.2.1. DEFINICIÓN Y CONSIDERACIONES

Este microcontrolador se encarga de leer los datos de los denominados: sensores básicos, y enviar estos datos al microcontrolador principal.

Los sensores básicos son considerados los siguientes:

- Sensor de humedad FC-28.
- Sensor de lluvia HL-83.
- Sensor de temperatura y humedad DHT-22.
- Fotorresistencia GL-5539.

El sensor de humedad FC-28, informa sobre la humedad relativa de la tierra. Este sensor utiliza una señal analógica y digital, que para tener una mayor precisión, se utiliza la señal analógica, conectada al pin ADC del canal 0 del módulo de desarrollo ESP-32. Para la obtención de este valor, se realizan 10 lecturas del valor, y se realiza la media. Los valores oscilan entre 0 y 1024, significando los valores cercanos a 0: mucha humedad en la tierra; y los valores cercanos a 1024: sequedad en la tierra. Este sensor se alimenta mediante los 5V DC [59] que proporciona el módulo de desarrollo ESP-32.

El sensor de lluvia HL-83, tiene un funcionamiento similar al del sensor FC-28. Nos permite saber si ha llovido recientemente o no. Se utiliza la señal analógica conectada al pin ADC del canal 3 del módulo de desarrollo. Da valores entre 0 y 1024 y se alimenta con 5V DC.

La fotorresistencia GL-5539, es una resistencia sensible a la luz, de manera que sus valores varían según la intensidad de la luz en el receptor. Esta resistencia es inversamente proporcional a la intensidad de la luz recibida. Para obtener un valor entre 0 y 1024, aproximadamente, se utiliza una resistencia de 22K en modo *pull-up* [60]. Mediante el pin ADC del canal 6, se realiza la lectura de 10 valores, y posteriormente se hace la media. Un valor cercano a 0 indica mayor luminosidad y un valor 1024, menor luminosidad. Es posible que se supere el valor de 1024 cuando hay muy poca luminosidad.

El sensor de temperatura y ambiente DHT-22, proporciona una temperatura absoluta del ambiente y una humedad relativa del mismo. El procesamiento del sensor DHT-22 se

realiza mediante una librería externa, mantenida por el usuario gosouth de GitHub [61]. Esta librería nos devuelve la temperatura y humedad finales. Se utiliza el pin I/O 16 del módulo de desarrollo para la lectura del sensor. La unidad de la temperatura es en grados centígrados y los de la humedad: un valor de 0 a 100, significando los valores próximos a 0, que el ambiente es muy húmedo, y los valores a 100, que el ambiente es menos húmedo.

En la siguiente sección se muestra un diagrama de flujo del programa de este microcontrolador, y en la sección posterior se muestra el esquema electrónico de las conexiones entre los sensores y el módulo de desarrollo.

2.1.4.2.2. DIAGRAMA DE FLUJO

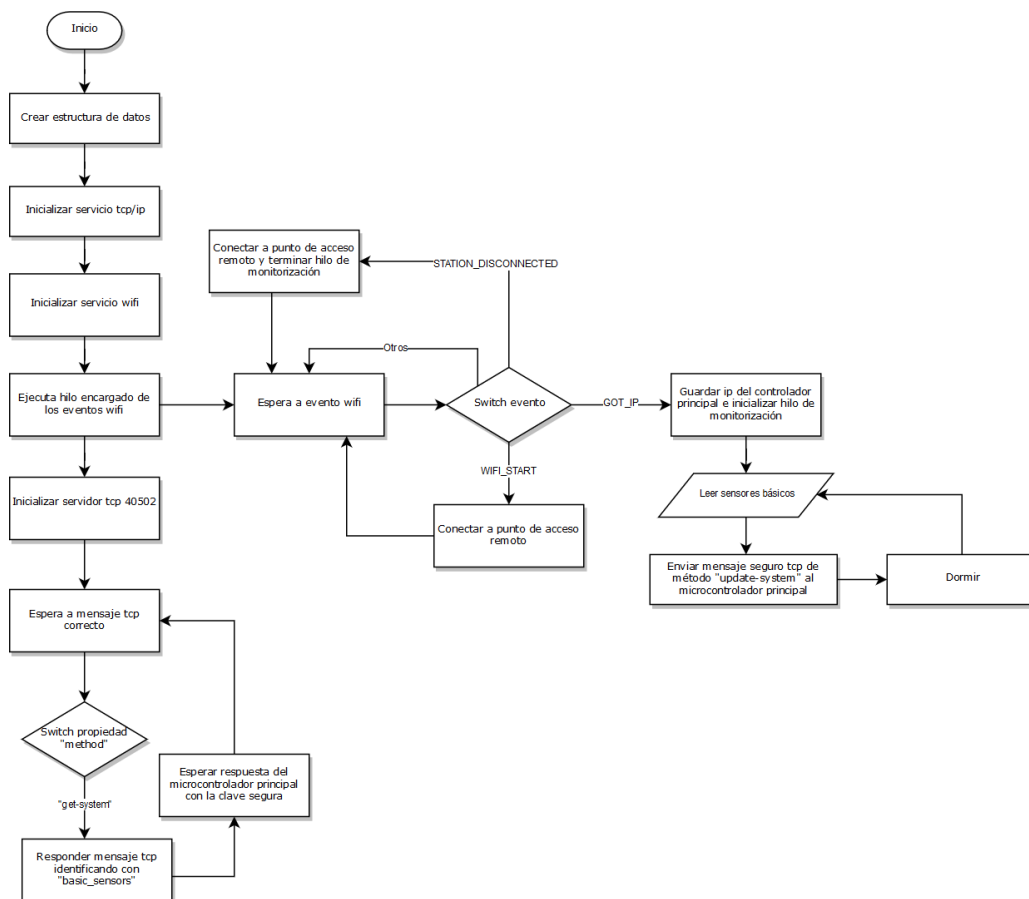


Ilustración 4. Diagrama de flujo del microcontrolador secundario de sensores básicos.

Si la imagen no se aprecia, se puede ver la imagen original, disponible en el capítulo de Anexos.

2.1.4.2.3. ESQUEMA ELECTRÓNICO

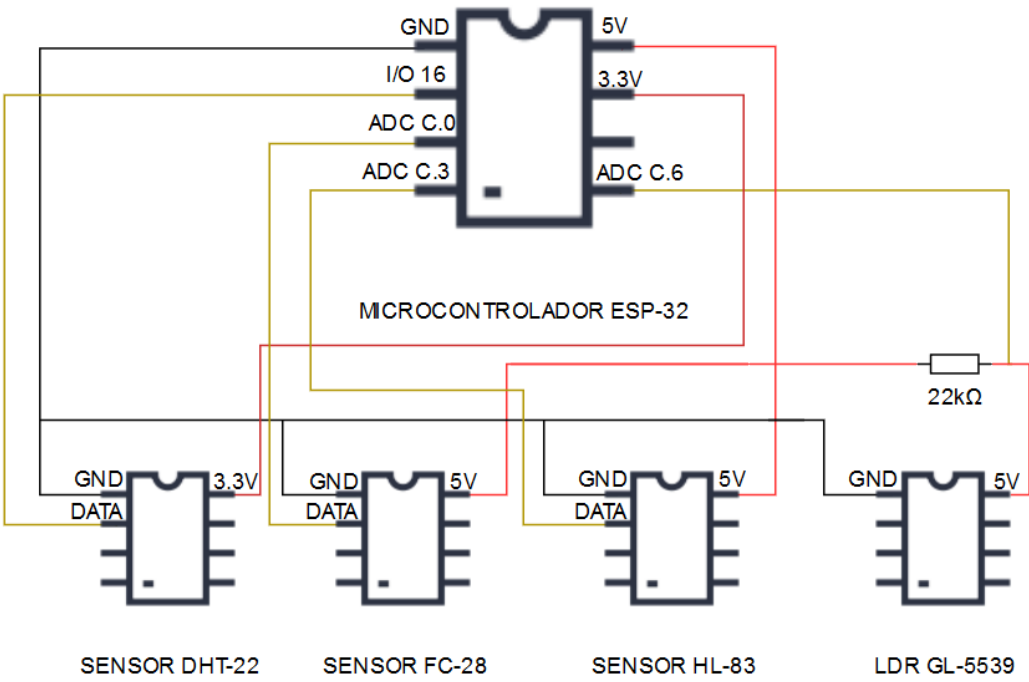


Ilustración 5. Esquema electrónico del microcontrolador secundario de sensores básicos.

2.1.4.3. MICROCONTROLADOR SECUNDARIO DE CONTROL DE RIEGO Y NEUTRALIZADOR DE PH

2.1.4.3.1. DEFINICIÓN Y CONSIDERACIONES

Este segundo microcontrolador secundario, se encarga de leer los datos proporcionados por el sensor de pH, que obtiene de la sonda de electrodo pH. Este sensor proporciona el valor del pH del agua que se utiliza para regar el cultivo. Este dato se envía al microcontrolador principal. Este microcontrolador se encargará de realizar las siguientes acciones sobre el cultivo: regar, neutralizar el pH del agua o neutralizar si es necesario y posteriormente regar. La acción de regar, provocará que se envíe un mensaje TCP con el método “update-system” al microcontrolador principal.

Para realizar estas acciones, el microcontrolador utiliza los siguientes sensores:

- Sensor de pH y sonda de electrodo pH.
- 2 x pesos HX-711.
- Sensor YF-S201.

El sensor de pH está conectado a la sonda de electrodo pH, y la sonda está introducida en una botella que contiene el agua con el que se pretende regar. De esta manera, se puede saber el pH en cada momento. Cabe destacar que el pH, depende de la temperatura del ambiente, por lo que el microcontrolador principal se encargará de ajustar el valor obtenido por este microcontrolador secundario [62]. El sensor de pH se alimenta con 5V DC y devuelve un valor analógico. Para determinar el pH con el valor analógico, se utiliza como referencia una librería externa para Arduino, mantenido por el usuario DFRobot de GitHub [63]. Para obtener el valor se utilizan dos pines ADC. En el desarrollo se utilizarán los pines ADC del canal 0 y 1 del módulo de desarrollo.

El sensor de peso HX-711, es también un sensor analógico. Se alimenta con 5V DC y utiliza dos pines, uno para obtener el valor de la presión superior y el otro para obtener el valor de la presión inferior, obteniendo el valor correspondiente al peso en gramos, colocado sobre un soporte. Como tenemos dos de estos sensores, uno para medir el volumen en la botella correspondiente a la concentración ácida, y el otro para la botella de la concentración base; utilizaremos 4 canales ADC: los pines ADC del canal 3 y 6 para

medir la botella de la concentración ácida y los pines ADC del canal 2 y 4 para la concentración base. Se utiliza una librería externa para Arduino, mantenida por el usuario bodge de GitHub [64].

El tercer sensor que se utiliza, el sensor YF-S201, es un sensor que mide la cantidad de caudal en L/minuto. Se alimenta con 5V DC. Sirve para saber cuántos con cuantos mililitros se está regando el cultivo. Este sensor funciona mediante pulsos producidos por el efecto Hall [65]. Esto significa que cada aproximadamente 2.25 mililitros se produce un cambio considerable del voltaje.

El módulo de desarrollo que tenemos, tiene un servicio de interrupciones, por tanto utilizaremos el pin I/O 4 para registrar las interrupciones que se reciban de este sensor y medir la cantidad de líquido [66]. Las interrupciones se activarán por flanco de subida [67], tras lo cual se activará la ISR [68] registrada. La ISR simplemente contará el número de interrupciones que se producen.

Aparte de los sensores, para poder realizar la neutralización y el riego, el microcontrolador utiliza los siguientes servomecanismos:

- 1 x DC 12V 1/4'' marca DIGITEN.
- 2 x DC 12V 1/4'' marca U.S. Solid.
- 3 x motores DC 12V 1/4''.

Cada electroválvula está conectada a un motor, lo que facilita el movimiento del líquido. Las electroválvulas y los motores funcionan con 12V DC. Para poder alimentarlos, se provee de un transformador de pared de 12V, que convierte la corriente alterna de 220V AC a los 12V DC necesarios para poder alimentar los dispositivos.

Cada circuito de cada par electroválvula y motor, está regido por un relé con optoacoplador de 3 canales, uno por cada circuito, siendo estos canales controlados por 3 pines I/O del módulo ESP-32. Concretamente los pines I/O 16, 17 y 18. De tal forma, que cuando se la señal de uno de estos pines es de un voltaje alto, el canal del relé se activa, activándose el circuito de 12V DC y accionando la electroválvula y motor conectados. Haciendo, de esta manera, pasar el líquido de un lugar a otro.

Para poder realizar la neutralización y el riego en el proyecto, se ha construido un soporte con maderas y botellas de agua pequeñas. A este soporte, se han conectado, mediante tornillos y silicona, los sensores, dispositivos y servomecanismos detallados en este microcontrolador, de la manera que se muestra en la sección 2.1.4.3.3.

Resumidamente, el funcionamiento de las acciones, a realizar por este microcontrolador, son:

- Acción de neutralizar:

El microcontrolador leerá el sensor de pH del agua y determinará si está dentro del rango del pH objetivos y umbral de pH, parámetros proporcionados inicialmente por el microcontrolador principal. Si el pH, está fuera del rango, el microcontrolador se dispondrá a neutralizar el pH del agua.

Para ello, se calculará la diferencia entre el pH objetivo y el pH actual del agua. Determinando de que manera se realiza la neutralización, o bien abriendo la electroválvula de la concentración ácida, o bien abriendo la electroválvula de la concentración base [69].

Determinado el tipo de neutralización, mediante los sensores HX-711 se obtendrá el peso de la botella de la concentración correspondiente, restando el peso de la botella vacía, medido previamente, se puede obtener el volumen de la concentración. Además, previamente el microcontrolador, tendrá definido la molaridad de las concentraciones. Obteniendo mediante la fórmula $\text{moles} = \text{Molaridad} / \text{Volumen}$, la cantidad de volumen del ácido o base, necesarios para neutralizar la reacción [70].

Pero en el desarrollo, se obtendrán los moles de concentración y en relación con el pH, se abrirá mas tiempo o menos tiempo la electroválvula.

Finalmente, se volverá a medir el pH del agua, y si está dentro del rango determinado por el pH objetivo y el umbral, se finalizará la acción, si no, se repetirá la neutralización, pero cada vez se abrirá menos tiempo la electroválvula, ya que el pH irá variando.

Si la acción consiste en neutralizar y regar, acto seguido a esta acción, se realizará la acción de regar. Si no, simplemente finalizará este proceso.

- Acción de regar:

Inicialmente, el microcontrolador secundario registra la ISR al pin I/O 4, para preparar la obtención de datos del sensor YF-S201, mediante el cambio de potencial en el pin I/O 4, interpretado por el programa, como interrupciones. Posteriormente, pone en modo bajo voltaje el pin I/O del canal del relé, correspondientes al circuito de la electroválvula y motor, que abren la botella del agua de riego.

Periódicamente, cada segundo, se desactivan las interrupciones del pin I/O 4. Para poder contar la cantidad de interrupciones durante el tiempo transcurrido, y determinar la cantidad de agua con la que se ha regado.

En el desarrollo, se ha puesto que si se ha regado con menos de 200 mililitro en total, el proceso comienza de nuevo, registrando la ISR al pin I/O 4 y calculando los mililitros hasta llegar a 200.

Finalmente, tras regar 200 mililitros, se envía un mensaje TCP de método “update-system” al microcontrolador principal.

Cabe destacar con relación al desarrollo, que se compró la sonda de electrodo de pH sin el sensor de pH, por esto, el desarrollo de una parte de la acción de neutralización está sin finalizar. Habiendo pseudocódigo con las instrucciones correspondientes.

2.1.4.3.2. DIAGRAMA DE FLUJO

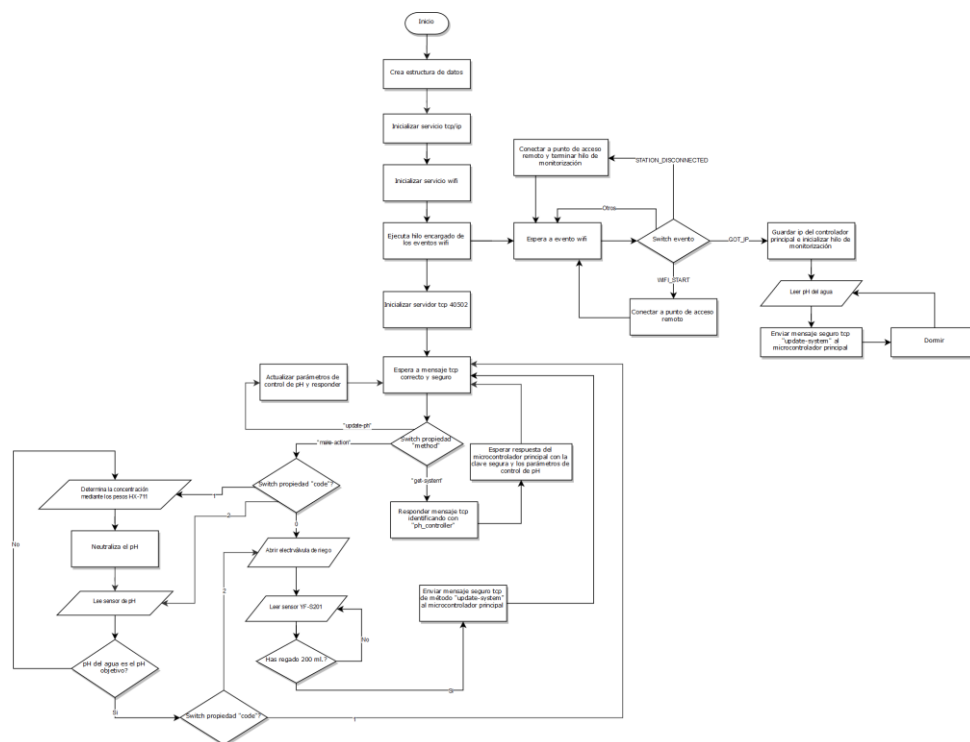


Ilustración 6. Diagrama de flujo del microcontrolador secundario de control de riego y neutralización de pH.

Se recomienda ampliar la ilustración para una mayor apreciación.

2.1.4.3.3. ESQUEMA ELECTRÓNICO

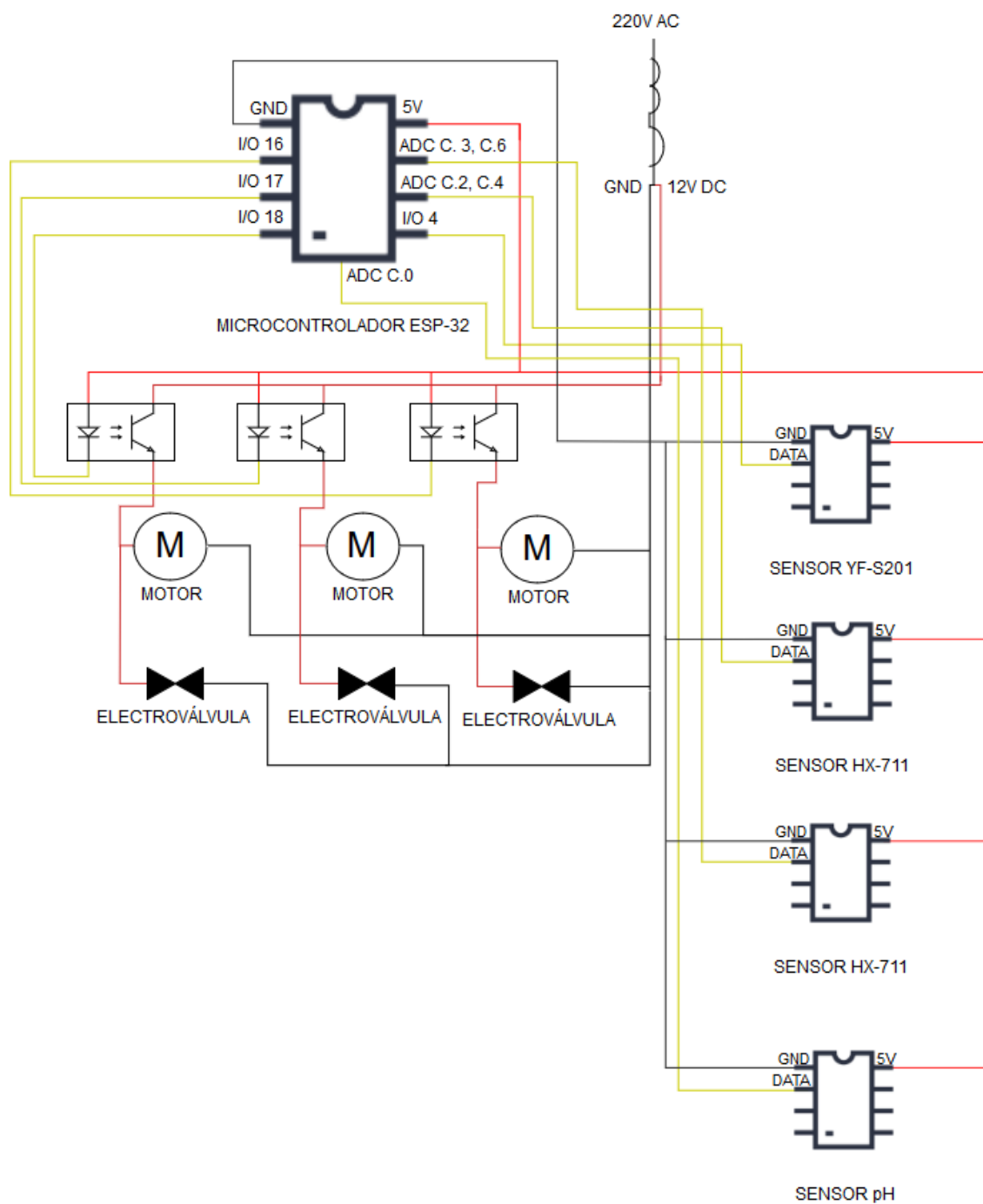


Ilustración 7. Diagrama electrónico del microcontrolador secundario de control de riego y neutralización de pH.

2.2. MÓDULO 2: *MIDDLEWARE*

2.2.1. INTRODUCCIÓN

Este módulo consiste en un *middleware*, que permite comunicar el módulo 1, que es el sistema autónomo que controla y gestiona el cultivo, con el servidor remoto del módulo 3.

En un inicio se tenía pensado utilizar una Raspberry Pi [71] para el dispositivo que se encargara de la comunicación, pero finalmente se decidió utilizar un microcontrolador ESP-32. Ya que, al haber trabajado con este microcontrolador inicialmente en el módulo 1, el desarrollo ha sido más rápido. Además, utilizar este microcontrolador hace que el sistema sea más económico para el usuario [72].

Para permitir la comunicación entre los microcontroladores de un sistema de cultivo y el servidor remoto, es necesario que el servidor remoto del módulo 3, pueda acceder al servidor del microcontrolador del módulo 2. Esto es posible de dos maneras: que se utilice un punto de acceso a internet o bien, que la red local que se crea en este módulo, esté dentro del alcance del equipo del servidor remoto.

En un inicio, se diseñó el sistema global teniendo en cuenta que este módulo funcionaría mediante un punto de acceso a internet. Para esto, se adquirió el módulo Sim900a mini v3.8.2 [73], que proporciona un acceso a la red GSM/GPRS mediante una tarjeta SIM [74]. Pero a última hora se ha descubierto que el módulo Sim900a no proporciona una IP pública, sino una IP privada compartida por otros mismos dispositivos, dentro de una red privada de la compañía [75]. Lo que impide la comunicación desde el servidor al microcontrolador, en cambio, enviar datos mediante el módulo GSM/GPRS al servidor si es posible, pero esto no es un objetivo perseguido, ya que se consumirían bastantes datos [76].

Aun así, el envío de datos desde el servidor al microcontrolador del módulo 2, sería posible si se utilizara un APN [77] privado con una IP pública y estática [75]. Pero debido a falta de tiempo, esto no ha sido posible desarrollarlo.

Por tanto, el *middleware* se ha desarrollado teniendo en cuenta que la conexión entre el módulo 2 y el módulo 3, será mediante la red local que crea el microcontrolador de este módulo. Igualmente, en esta memoria se muestra el uso del *middleware* tanto a través de internet, como por medio de una red local, porque al fin y al cabo, el funcionamiento es el mismo.

Para la explicación de las decisiones de este módulo, se ha decidido utilizar las siguientes secciones:

- Inicialmente, se explicará la arquitectura de red utilizada, así como la toma de decisiones en esta y los motivos asociados.
- Posteriormente, se especificará la interfaz de programación de aplicación del servidor utilizada, de manera similar al módulo 1.
- Se utilizará otra sección, en donde se hablará sobre la seguridad implicada en este módulo, así como las vulnerabilidades y posibles mecanismos de seguridad.
- Finalmente, se proporcionará un diagrama de flujo esquemático, que muestre los procesos de este *middleware*.

2.2.2. ARQUITECTURA DE RED

Este *middleware* se ha creado con el objetivo de permitir que un usuario pueda tener múltiples sistemas de cultivo del módulo 1, utilizando un solo punto de acceso a internet, o en este caso: una sola red local.

Este módulo podría ser omitido, si en cada controlador principal del módulo 1, se añadiera la API de este módulo y que cada microcontrolador principal, tuviera su punto de acceso a internet, o que la red local de todos los posibles microcontroladores principales del módulo 1, estuvieran al alcance del equipo del servidor del módulo 3. Lo que es altamente improbable, por esto surge la necesidad de este módulo.

El funcionamiento de este módulo es el siguiente: el microcontrolador, tras inicializar los servicios TCP/IP, crear un servidor DHCP e inicializar el servicio wifi, crea una red local y lanza un hilo encargado de los eventos wifi. Finalmente, el proceso principal crea un *socket* TCP en el puerto 40500, al que se queda escuchando a la espera de posibles mensajes TCP entrantes.

Como se comentó en el módulo 1, los microcontroladores principales existentes en el radio de conectividad de la red local, se irán conectando mediante un SSID y contraseña que tienen predefinidos.

El hecho de tener las redes predefinidas, permitiría la conexión entre el microcontrolador de este módulo y múltiples microcontroladores principales, cada uno con sus respectivos microcontroladores secundarios asociados; siempre que el SSID de las redes locales de los microcontroladores principales del módulo 1, sean diferentes.

No es objetivo de este proyecto pretender que los microcontroladores secundarios, sean intercambiables entre los posibles microcontroladores principales del módulo 1, que puedan existir dentro de una red local de módulo 2. Esto, más que una necesidad, vendría a ser una utilidad. Con redes abiertas se podría conseguir fácilmente, pero esto es una vulnerabilidad de seguridad [78].

A continuación, se muestra una ilustración de la conectividad entre el microcontrolador de este módulo y varios microcontroladores principales del módulo 1, con sus microcontroladores secundarios asociados. También se muestran las posibles formas de conectividad con el servidor del módulo 3, anteriormente mencionadas:

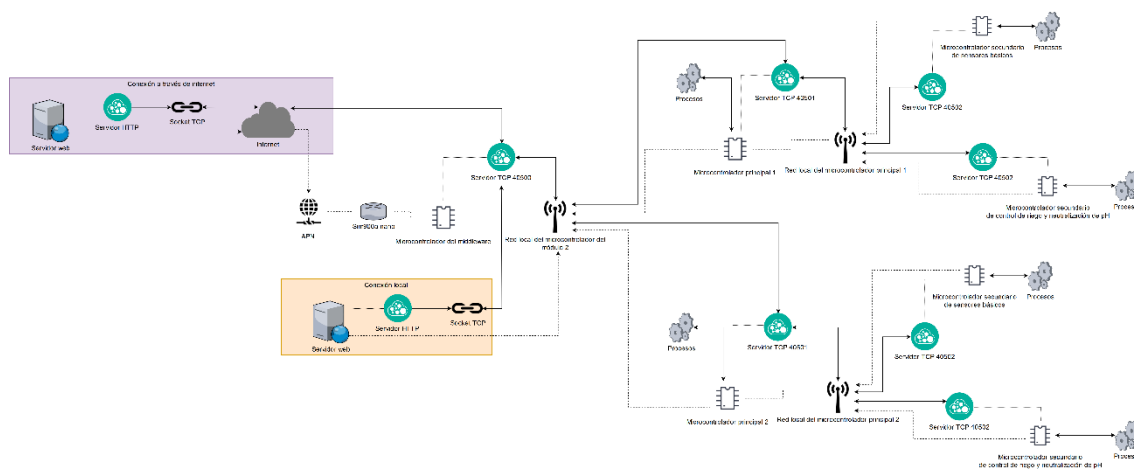


Ilustración 8. Diseño de la arquitectura de red del módulo 2.

Si la imagen no se aprecia, se puede ver la imagen original, disponible en el capítulo de Anexos.

2.2.3. DISEÑO DE LA INTERFAZ DE PROGRAMACIÓN DE APLICACIÓN DEL SERVIDOR

La API de este módulo, se ha diseñado considerando las funcionalidades que podrá realizar un usuario en remoto mediante el sistema web del módulo 3. Estas funcionalidades son las siguientes:

- Registrar un sistema de cultivo que funciona, con la opción de inicializar los parámetros configurables del microcontrolador secundario encargado del pH.
- Eliminar un cultivo registrado previamente.
- Editar los parámetros configurables del microcontrolador secundario encargado del pH, para un sistema de cultivo registrado previamente y que esté funcionando.
- Obtener los datos monitorizados por los sensores, de un cultivo registrado previamente y funcionando.
- Ordenar acciones en remoto al cultivo. Las acciones posibles son: neutralizar el pH y regar o solo neutralizar.

Estas funcionalidades, serán indicadas por el servidor del módulo 3, que mediante un *socket*, enviará mensajes TCP. Estos mensajes se han diseñado utilizando el formato JSON y teniendo en cuenta los estados de respuesta del protocolo HTTP, al igual que en el anterior módulo.

A continuación, se describirán los mensajes utilizados, relacionados con las funcionalidades anteriormente listadas:

- Registrar un cultivo:

```
{  
  "auth": BASE64_BASIC_AUTH_STRING,  
  "method": "add-system",  
  "properties": (opcional)  
}
```

```

        "objective_ph": 7.0,

        "ph_umbral": 0.2

    }

}

```

Al recibir este mensaje, el microcontrolador de este módulo examinará la lista de estaciones conectadas a la red local, y las que no hayan sido previamente registradas, se les enviará el mensaje TCP con método “get-system” (descrito en el módulo 1).

Si se recibe una respuesta que contenga la propiedad “status” con valor 200, examinará si el mensaje recibido inicial de método “add-system” contenía la propiedad “properties”. En caso afirmativo, envía el mensaje con método “edit-system” (descrito en el módulo 1).

Finalmente, el microcontrolador responderá mediante un mensaje JSON con la propiedad “status”, que contendrá alguno de los siguiente valores:

- 200: el sistema se registró correctamente, almacenando la dirección MAC del microcontrolador principal correspondiente.
- 409: se han registrado previamente 8 sistemas de cultivo, y no se pueden registrar más.
- 404: no se encuentra ningún sistema del módulo 1 funcionando correctamente.
- 400: el mensaje recibido no sigue el patrón definido.

- Eliminar un cultivo registrado:

```

{

    "auth": BASE64_BASIC_AUTH_STRING,

    "method": "delete-system",

    "id": 0

}

```

Tras este mensaje, el microcontrolador determinará si existe algún sistema del módulo 1, previamente registrado. En caso afirmativo, eliminará de la lista de registro, la dirección física del microcontrolador registrado, localizado en la posición indicada por la propiedad “id”.

Se responderá mediante uno de los siguientes valores en la propiedad “status”:

- 200: se eliminó del registro correctamente, la dirección MAC física correspondiente al microcontrolador registrado.
- 409: no existen microcontroladores principales previamente registrados.
- 404: no existe un microcontrolador registrado en la posición indicada por “id”.
- 400: el mensaje recibido no sigue el patrón definido.

- Editar las propiedades de un cultivo registrado:

```
{  
  
  "auth": BASE64_BASIC_AUTH_STRING,  
  
  "method": "edit-system",  
  
  "id": 0,  
  
  "properties":  
  {  
    "objective_ph": 7.0,  
    "ph_umbral": 0.2  
  }  
}
```

Este mensaje editará las propiedades de un cultivo registrado, indicado por la propiedad “id”, en caso de ser posible. Si es posible, se enviará el mensaje con método “edit-system” (descrito en el módulo 1), con las propiedades indicadas en la propiedad “properties”, al microcontrolador principal correspondiente.

Se responderá mediante uno de los siguientes valores, en la propiedad “status”:

- 200: se editaron correctamente las propiedades del sistema indicado.
- 409: no existen microcontroladores registrados.
- 404: el sistema indicado, no está conectado a la red local.
- 400: el mensaje recibido no sigue el patrón definido.
- 500: ocurrió un error en el microcontrolador del *middleware*.

- Obtener los datos del sistema indicado:

```
{
  "auth": BASE64_BASIC_AUTH_STRING,
  "method": "get-system",
  "id": 0
}
```

Este mensaje pedirá los últimos datos de los sensores, al sistema de cultivo registrado, indicado por la propiedad "id". Si es posible, se enviará el mensaje con método "get-system" (descrito en el módulo 1), al microcontrolador principal correspondiente.

Se responderá mediante uno de los valores siguientes, en la propiedad "status":

- 200: se recibieron correctamente los datos monitorizados del sistema indicado.
- 409: no existen microcontroladores registrados.
- 404: el sistema indicado, no está conectado a la red local.
- 400: el mensaje recibido no sigue el patrón definido.
- 500: ocurrió un error en el microcontrolador del *middleware*.

En caso de responder con un estado 200, el mensaje de respuesta al servidor, incluirá la propiedad "properties", con los últimos datos monitorizados del cultivo. El mensaje seguirá el siguiente patrón:

```
{
  "status": 200,
  "properties":
  {
    "air_temperature": 25.3,
    "air_humidity": 45.4,
    "rain_sensor": 505,
    "soil_humidity": 700,
    "light_sensor": 334,
```

```

        "ph_water": 7.0,
        "last_water": 450000
    }
}

```

- Enviar una acción al sistema de cultivo indicado:

```

{
    "auth": BASE64_BASIC_AUTH_STRING,
    "method": "send-action",
    "id": 0,
    "code": 0,
}

```

Este mensaje se envía, cuando se quiere realizar una acción sobre el sistema de cultivo indicado por la propiedad "id". La acción se indica mediante la propiedad "code", que tendrá los valores: 1 indicando neutralización, y 2 indicando neutralización y riego. Acto seguido, se enviará el mensaje con método "send-action" al microcontrolador principal, con la propiedad "code" recibida.

Se responderá mediante uno de los valores siguientes, en la propiedad "status":

- 200: se envió correctamente la acción a realizar.
- 409: no existen microcontroladores registrados.
- 404: el sistema indicado, no está conectado a la red local.
- 400: el mensaje recibido no sigue el patrón definido.
- 500: ocurrió un error en el microcontrolador del *middleware*.

2.2.4. SEGURIDAD

Al usarse un microcontrolador ESP-32, las vulnerabilidades de seguridad en este módulo, son las mismas vulnerabilidades que en el módulo 1. Por tanto, los mecanismos de seguridad son los mismos.

Aunque las vulnerabilidades sean las mismas, este módulo presenta una mayor amenaza que el módulo 1, ya que al estar conectado a un punto de acceso a internet, el atacante puede ser cualquiera. Para evitar esto, este módulo tendrá un usuario y contraseña predefinidos. Por tanto los mensajes enviados deberán una propiedad “auth”, en la que se indicarán el usuario y contraseña, separados por el carácter ‘:’ y codificado en Base64, finalmente precedido por la cadena de caracteres: “Basic ”, imitando la autenticación *basic* [79].

2.2.5. DIAGRAMA DE FLUJO

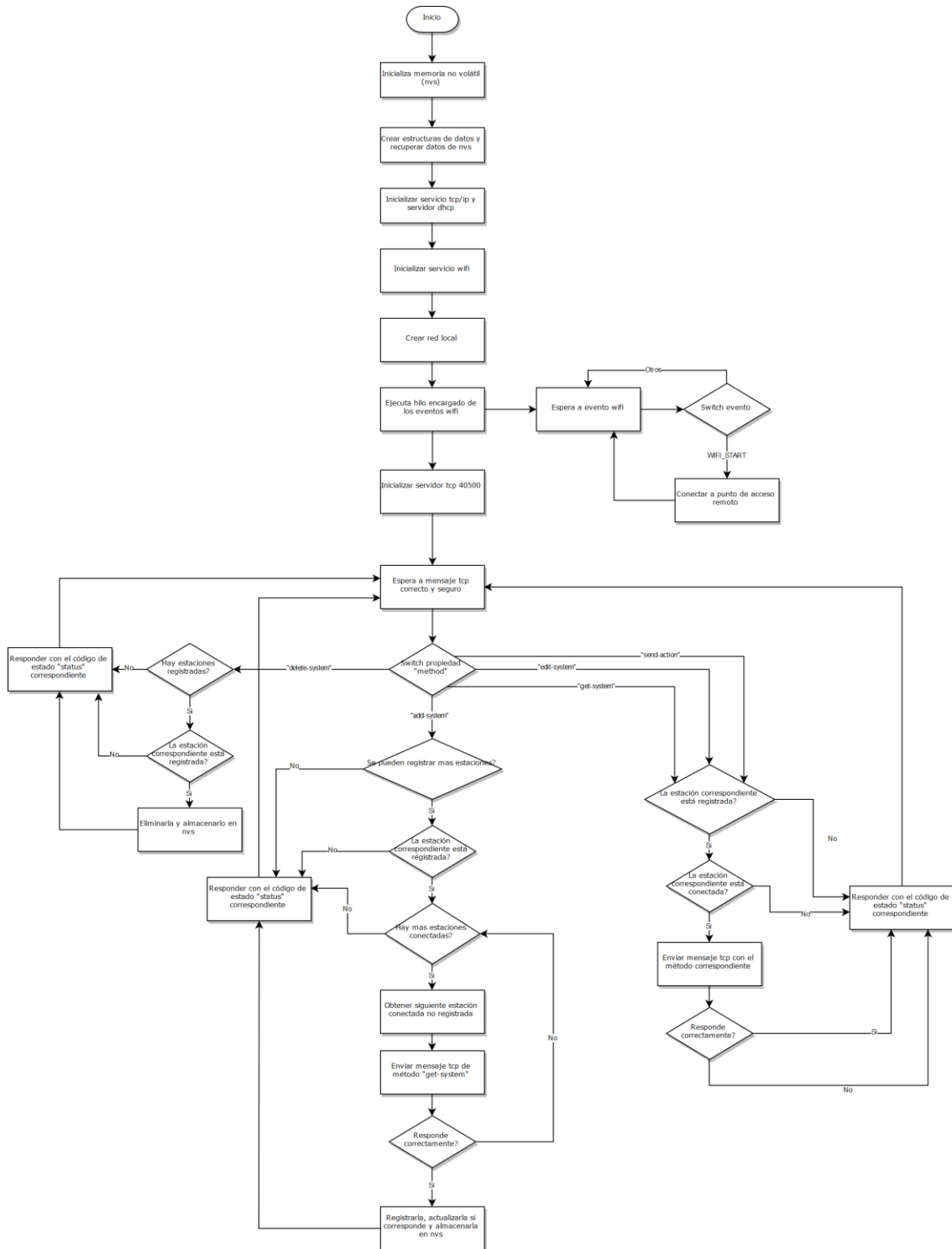


Ilustración 9. Diagrama de flujo del middleware.

2.3. MÓDULO 3: SISTEMA WEB BASADO EN UN ENTORNO *CLOUD*

2.3.1. INTRODUCCIÓN

En este módulo, se explicará el sistema web montado para permitir que múltiples usuarios puedan acceder desde un navegador web, donde poder registrarse y acceder a la plataforma desarrollada. Tras acceder, los usuarios podrán registrar sus cultivos existentes, eliminarlos, ver los datos de los sensores en continua actualización, editar parámetros esenciales del cultivo y ordenar acciones en remoto.

Las acciones que un usuario podrá ordenar a su cultivo son: neutralizar el pH del agua y regar o solo neutralizar el pH.

La estructura que se utilizará para explicar las partes referentes a este módulo, es la siguiente:

- Inicialmente, se definirán las tecnologías y lenguajes utilizados para montar las diferentes partes del sistema, y el motivo de la elección.
- Posteriormente, se comentará el diseño de la arquitectura web utilizada y las decisiones de diseño de esta.
- Finalmente, se comentará las diferentes partes de la arquitectura, que se divide en la aplicación cliente web, que a su vez se puede dividir en dos módulos: el módulo de gestión de usuarios y el módulo de gestión de cultivos, y el servidor web. En la sección del servidor web se definirá la API diseñada y los mecanismos de seguridad utilizados.

2.3.2. MARCO DE TRABAJO Y TECNOLOGÍAS

Para la aplicación web se utiliza el marco de trabajo Angular 7. Esta tecnología se desarrolló inicialmente por la empresa Google, pero actualmente es mantenido por un grupo externo a la empresa [80]. De código abierto y licencia no comercial [81].

Se ha decidido utilizar este marco de trabajo, que sigue el patrón de arquitectura Modelo-Vista-Controlador [82], ya que se tenía experiencia y soltura previa con la tecnología, lo que ha facilitado el desarrollo.

El patrón MVC, utiliza los lenguajes TypeScript para el controlador, CSS3 para la vista [83] y HTML5 para el modelo [84]. TypeScript es un lenguaje basado en JavaScript [85].

Esta aplicación utiliza una capa de servicios, por tanto necesita un servidor con una API para atender estos servicios, provenientes de las interacciones realizadas por los usuarios. A parte de un servidor que proporcione los recursos de la aplicación para poder utilizarla.

Por lo indicado anteriormente, se ha tenido como objetivo poder unificar la API con la suministración de recursos de la aplicación. Por lo que se ha decidido utilizar Node.js 10.15.0 junto al *middleware* Express [86] para Node.js, logrando el objetivo de usar un servidor para ambas tareas.

El desarrollo en Node.js y Express se realiza mediante el lenguaje JavaScript.

2.3.3. ARQUITECTURA WEB

Esta arquitectura web, está diseñada pensando en que la conexión entre el cliente, que sería la aplicación web, y el servidor, sea por medio de internet. Por tanto, para que esto sea posible, el servidor ha de estar alojado en un equipo con acceso a internet. Esto se puede conseguir fácilmente por medio de un *router* convencional. El acceso a internet deberá disponer de una IP pública, proporcionada por el ISP [87]. Además, en este caso, el *router* deberá redireccionar los datos entrantes por los puertos 80 y 443, al equipo que aloje el servidor. Es posible que haya que configurar los puertos del *firewall* [88] del equipo.

Se utilizan los puertos 80 y 443, ya que son los correspondientes a los protocolos HTTP y HTTPS [89]. Al atender las peticiones entrantes en estos puertos, el servidor procesará las peticiones y responderá por el mismo puerto. Las peticiones pueden ser recursos de la aplicación web o peticiones a la API.

El servidor utiliza un sistema de ficheros, para la consistencia de datos. Este sistema, permite una lectura y escritura más rápida y utiliza menos recursos computacionales, si se compara con los sistemas de base de datos. Además, es más sencillo de utilizar.

Una petición HTTP/S a la API, podrá conllevar la utilización del sistema de ficheros para almacenar datos de los usuarios. También podrá iniciar una conexión TCP con el microcontrolador del módulo 2 correspondiente.

Esta arquitectura permite tener múltiples clientes conectados de manera simultánea, y aunque el servidor almacenará las peticiones que irá recibiendo de los clientes, estas serán atendidas una a una por orden de llegada, por este motivo, el servidor se ha diseñado para que atienda las peticiones lo más rápido posible.

En la siguiente ilustración, se muestran los elementos relacionados con esta arquitectura, definidos anteriormente:

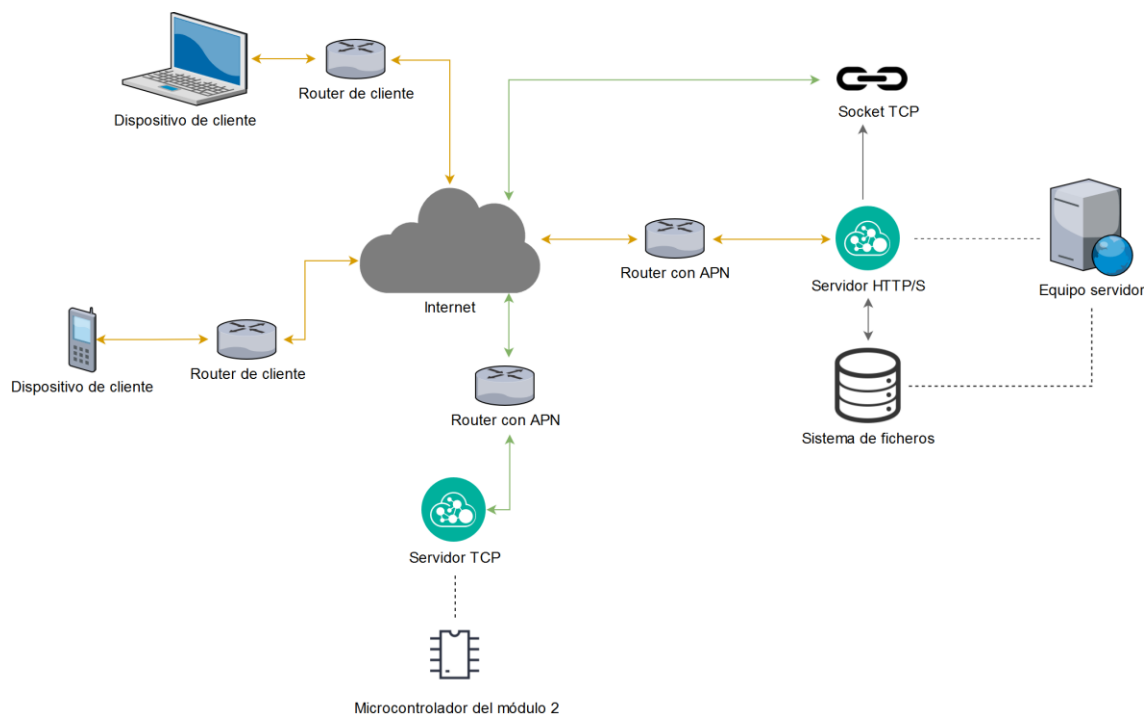


Ilustración 10. Diseño de la arquitectura de red del módulo 3.

2.3.4. SISTEMA WEB

2.3.4.1. APLICACIÓN CLIENTE WEB

2.3.4.1.1. DEFINICIÓN Y CONSIDERACIONES

La aplicación web, consiste en el cliente de la arquitectura. Un usuario, al introducir la IP pública, correspondiente al punto de acceso, proporcionado por el ISP, en un navegador convencional, provocará que el servidor envíe al cliente los recursos de la aplicación web, a través del protocolo HTTP/S.

La aplicación web está dividida en dos módulos: el módulo de gestión de usuarios y el módulo de gestión de cultivos. Inicialmente, el servidor enviará el módulo de gestión de usuarios, en donde el usuario se registrará o iniciará sesión, y tras la autenticación correcta por parte del servidor, enviará el módulo de gestión de cultivos.

Un módulo es un componente del marco de trabajo Angular, que se utiliza para organizar y agrupar los diferentes componentes y librerías de la aplicación. De esta manera, el servidor envía al cliente solo los recursos necesarios.

Cada módulo se compone de un componente. Estos componentes están diseñados de manera que sean *responsive*, es decir, se adaptan automáticamente a la anchura del navegador [90]. Lo que facilita al usuario la navegación y uso de los componentes, en un equipo convencional o en un equipo móvil.

En las siguientes secciones, se detallará el módulo de gestión de usuarios y el módulo de gestión de cultivos.

2.3.4.1.2. MÓDULO DE GESTIÓN DE USUARIOS

Este módulo contiene el componente que se encarga de la gestión de usuarios. Mediante este componente, el usuario se podrá registrar, indicando un identificador de usuario, email, contraseña y validación de contraseña; o bien, iniciar sesión en la plataforma, mediante un identificador de usuario y contraseña.

Se utilizan dos formularios: uno para el registro y otro para el inicio de sesión, con los campos identificados anteriormente [91]. En estos formularios se utiliza validación síncrona y asíncrona de los campos para comprobar los datos, antes de ser enviados al servidor [92].

La validación síncrona comprueba que los datos, cumplen las siguientes características: la contraseña contiene al menos 8 caracteres, existen datos introducidos en cada campo de los formularios y en el formulario de registro, que los campos de la contraseña y validación de contraseña son iguales.

La validación asíncrona realiza una petición al servidor, en donde comprueba el valor del campo del identificador de usuario, siendo válido cuando el identificador está almacenado en el sistema de ficheros del servidor, en el formulario de inicio de sesión, y también válido, cuando no existe el identificador del formulario de registro.

Al validarse los datos, y determinar que las validaciones son correctas, se enviará una petición POST al servidor con estos datos.

Se utiliza la autenticación *basic* para el envío de estos datos. Si el servidor responde con un código de estado 200, se guardará un token en el almacenamiento local del navegador, proporcionado por el servidor. Este token permitirá el acceso al módulo de gestión de cultivos, sin necesidad de recibir el módulo de gestión de usuarios, siempre y cuando el token sea válido (detallado en la sección 2.3.4.2.3.).

2.3.4.1.3. MÓDULO DE GESTIÓN DE CULTIVOS

Este segundo módulo contiene el componente de gestión de cultivos. Este componente permitirá al usuario gestionar sus cultivos, para ello, dispondrá de un formulario en donde podrá introducir las propiedades del *socket* (dirección IP y puerto TCP), que se utilizará para conectarse al microcontrolador del módulo 2. También se podrá introducir el usuario y contraseña del módulo 2 y se podrá indicar el tipo de cultivo.

Los campos del formulario del tipo de cultivo, indicarán los parámetros configurables iniciales del microcontrolador secundario de control de riego y neutralización de pH, que utiliza para realizar las acciones.

Los datos de este formulario, se envían al servidor, que abre una conexión TCP mediante estos datos. Si el módulo 2 responde correctamente al servidor, el servidor a su vez enviará una respuesta al cliente, que resultará en la adición de una tabla en el componente.

Esta tabla tiene los campos monitorizados del cultivo. Estos campos se actualizarán periódicamente, mediante mensajes HTTP entre el cliente y el servidor, que a su vez se obtendrán mediante mensajes TCP entre el servidor y el módulo 2. Finalmente, este último, utilizará también mensajes TCP con el microcontrolador principal que registró para obtener los datos.

Cada sistema de cultivo registrado, tendrá su tabla asociada a sus campos, siendo cada tabla independiente de las otras. Las tablas incorporan botones, mediante los cuales el usuario podrá ordenar acciones al cultivo, como regar y neutralizar; también podrá editar las propiedades configurables del cultivo o eliminar el cultivo.

Cabe destacar que la eliminación del cultivo, solo elimina la tabla relacionado con los datos y la referencia del módulo 2, continuando el sistema de cultivo del módulo 1, con sus procesos.

2.3.4.2. SERVIDOR WEB

2.3.4.2.1. DEFINICIÓN Y CONSIDERACIONES

El servidor web, es el encargado de dar respuesta a las peticiones de la aplicación cliente web. Estas peticiones pedirán recursos de la aplicación web o solicitarán el servicio de la API diseñada.

El servidor utiliza el puerto 80, correspondiente al protocolo HTTP, y el puerto 443, correspondiente al protocolo HTTPS, para atender las peticiones. Si recibe una petición por el puerto 80, envía un mensaje HTTP con el código de estado 301 [93], que provoca la redirección de esa misma petición, al puerto 443. De esta manera, se obliga a utilizar el protocolo HTTPS, asegurando la comunicación.

2.3.4.2.2. DISEÑO DE LA INTERFAZ DE PROGRAMACIÓN DE APLICACIONES DEL SERVIDOR

La API del servidor está diseñada teniendo como referencia el estilo de arquitectura de red REST, que persigue objetivos como: utilizar los métodos HTTP correctos según el caso, responder con códigos de estado determinados por cada situación, utilizar las cabeceras, *query* y cuerpo de las peticiones de una manera determinada, entre otros [94] [95].

Para los mensajes del cuerpo de las peticiones se utiliza el formato JSON, ya que Angular por defecto utiliza este formato de mensaje, lo que facilita el desarrollo.

Los métodos definidos de la API son los siguientes:

- POST /check_username:

Descripción: utilizada para la validación asíncrona del módulo de gestión de usuarios.

Cuerpo del mensaje:

```
{  
  
  "username": "admin"  
  
}
```

Códigos de estado de respuesta:

- 400: error en la petición.
- 500: error interno en el servidor.
- 401: el usuario no está registrado.
- 200: el usuario está registrado.

- POST /registration:

Descripción: utilizado para comprobar el formulario de registro del módulo de gestión de usuarios. Si el usuario no existe, quedará registrado en el sistema de ficheros.

Cuerpo del mensaje:

```
{
  "username": "admin",
  "password": "XxxADMINxxX",
  "email": "admin@gmail.com"
}
```

Códigos de estado de respuesta:

- 400: error en la petición.
- 500: error interno en el servidor.
- 403: usuario previamente registrado.
- 200: el usuario quedó registrado.

- GET /authentication:

Descripción: utilizado para comprobar el formulario de inicio de sesión del módulo de gestión de usuarios. Comprobará los datos en el sistema de ficheros. Se envía en una cabecera.

Cabeceras:

“Authorization”: “Basic BASE64_BASIC_AUTH_STRING”

Códigos de estado de respuesta:

- 400: error en la petición.
- 500: error interno en el servidor.
- 403: el usuario no tiene autorización para obtener el módulo de gestión de cultivos.
- 200: el usuario tiene autorización para obtener el módulo de gestión de cultivos.

- GET /authorization:

Descripción: utilizado para comprobar el token del cliente. El token se envía en una cabecera.

Cabeceras:

“Authorization”: “Bearer
BASE64_HEADER_STRING.BASE64_PAYLOAD_STRING.BASE64_SIGNATURE
_STRING”

Códigos de estado de respuesta:

- 400: error en la petición.
- 500: error interno en el servidor.
- 403: el token no es válido.
- 200: el token es válido y el usuario tiene autorización.

- GET /get_systems:

Descripción: utilizado para obtener los cultivos registrados por un usuario. En caso correcto, el servidor consulta los sistemas de cultivo almacenados en el sistema de ficheros y los envía en formato JSON en el cuerpo de la respuesta.

Cuerpo del mensaje:

```
{  
    "username": "admin"  
}
```

Códigos de estado de respuesta:

- 400: error en la petición.
- 500: error interno en el servidor.
- 403: el usuario no está registrado.
- 200: se envían los cultivos en el cuerpo de la respuesta.

- POST /check_system:

Descripción: utilizado para obtener los datos monitorizados del cultivo registrado, indicado por la propiedad “id” de un usuario. Se consulta el sistema de cultivo en el sistema de ficheros y si existe, se crea una conexión TCP con el módulo 2, al que enviará un mensaje TCP con el método “get-system”. Si todo es correcto, se envían al cliente los datos, recibidos por el módulo 2 mediante la conexión TCP creada, en el cuerpo del mensaje de la respuesta HTTP.

Cuerpo del mensaje:

```
{  
    "username": "admin",  
    "id": 0  
}
```

Códigos de estado de respuesta:

- 400: error en la petición.
 - 500: error interno en el servidor.
 - 403: el usuario no está registrado o el cultivo id no existe para ese usuario.
 - 404: punto de acceso del *middleware* no encontrado.
 - 500: hubo un error al obtener los datos del cultivo.
 - 401: credenciales del *middleware* erróneos.
 - 409: se accedió al *middleware*, pero no se encontró el sistema del módulo 1.
 - 410: se accedió al *middleware*, pero el sistema del módulo 1 no está conectado a la red del *middleware*.
 - 408: se accedió al *middleware* y al sistema de cultivo, pero tarda mucho en responder.
 - 200: se envían los datos del sistema de cultivo, recibidos por el *middleware*.
-
- POST /add_system:

Descripción: utilizado para registrar un cultivo existente, indicando unos parámetros. Al recibir este mensaje, el servidor creará una conexión TCP con los datos del mensaje. El mensaje enviado tiene el método "add-system". Si todo es correcto, se informa al cliente con el código descrito. La aplicación web creará una tabla para este cultivo, y enviará al servidor web peticiones POST HTTP /check_system periódicamente, para obtener y actualizar los datos monitorizados del sistema de cultivo.

Cuerpo del mensaje:

```
{  
    "ip": "127.0.0.1",  
    "port": 40500,  
    "username": "admin_MOD2",
```

```

    "password": "pass_MOD2",
    "harvest":
    {
        "objective_ph": 7.01,
        "ph_umbral": 0.02
    }
}

```

Códigos de estado de respuesta:

- 400: error en la petición.
- 505: error interno en el servidor.
- 404: punto de acceso del *middleware* no encontrado.
- 500: hubo un error al registrar el cultivo.
- 401: credenciales del *middleware* erróneos.
- 409: se accedió al *middleware*, pero existen demasiados sistemas de cultivo registrados.
- 410: se accedió al *middleware*, pero no se encontró ningún sistema del módulo 1 conectado.
- 200: se indica al cliente, que un sistema de cultivo se encontró y registró en el módulo 2.
- POST /set_systems:

Descripción: utilizado para almacenar en el sistema de ficheros los cultivos enviados por el usuario. Cuando la aplicación web va a finalizar o el usuario gestiona alguno de sus sistemas de cultivo, envía esta petición para que el servidor almacene el *array* de sistemas en el sistema de ficheros.

Cuerpo del mensaje:

```

{
    "username": "admin",
    "systems":
    [

```

JSON_SYSTEMS_ARRAY

```
]
}
```

Códigos de estado de respuesta:

- 400: error en la petición.
 - 505: error interno en el servidor.
 - 403: usuario no registrado.
 - 200: cultivos almacenados en el sistema de ficheros correctamente.
-
- POST /delete_system:

Descripción: utilizado para eliminar del sistema de ficheros, un cultivo registrado previamente por un usuario. El servidor enviará un mensaje TCP con el método “delete-system” al *middleware* almacenado en el sistema de ficheros.

Cuerpo del mensaje:

```
{
    "username": "admin",
    "id": 0
}
```

Códigos de estado de respuesta:

- 400: error en la petición.
- 505: error interno en el servidor.
- 403: el usuario no está registrado o el cultivo id no existe para ese usuario.
- 404: punto de acceso del *middleware* no encontrado.
- 500: hubo un error al eliminar del *middleware* el sistema de cultivo.
- 401: credenciales del *middleware* erróneos.
- 409: se accedió al *middleware*, pero no hay sistemas registrados con el id.
- 410: se accedió al *middleware*, pero no hay sistemas registrados.
- 200: el cultivo se eliminó del registro del *middleware*.

- POST /send_action:

Descripción: utilizado para ordenar una acción remota al sistema de cultivo indicado. Se enviará el mensaje TCP al *middleware*, almacenado en el sistema de ficheros, con el método “send-action”, indicando la acción correspondiente mediante la propiedad “code”.

Cuerpo del mensaje:

```
{
    "username": "admin",
    "id": 0,
    "code": 1
}
```

Códigos de estado de respuesta:

- 400: error en la petición.
- 505: error interno en el servidor.
- 403: el usuario no está registrado o el cultivo id no existe para ese usuario.
- 404: punto de acceso del *middleware* no encontrado.
- 500: hubo un error al enviar la acción al microcontrolador principal.
- 401: credenciales del *middleware* erróneas. 409: se accedió al *middleware*, pero no se encontró el sistema del módulo 1.
- 410: se accedió al *middleware*, pero el sistema del módulo 1 no está conectado a la red del *middleware*.
- 408: se accedió al *middleware* y al sistema de cultivo, pero tarda mucho en responder.
- 200: la acción se envió al microcontrolador secundario correspondiente.

2.3.4.2.3. SEGURIDAD

Como se ha ido viendo en las secciones de este módulo. Se utilizan varios mecanismos de seguridad para dar seguridad al sistema web. A continuación, se listarán los mecanismos utilizados y describirá el funcionamiento de cada uno:

- Protocolo HTTPS:

El servidor recibe peticiones HTTP y HTTPS, pero las peticiones HTTP las redirecciona al puerto 443. Por tanto, siempre se utiliza el protocolo HTTPS para la comunicación entre el cliente y el servidor.

El servidor utiliza en todo momento el protocolo HTTPS, que utiliza un certificado auto firmado, creado mediante la herramienta OpenSSL [96].

Este certificado usa el estándar X.509 [97] con claves RSA de 2048 bits (por curiosidad del lector: la NSA pagó 10\$ M. a RSA, para que el algoritmo no fuera completamente seguro [98]).

De esta manera, se provee al sistema web de cierta privacidad.

- Autenticación *basic*:

Esta autenticación no es segura, simplemente codifica en base 64 un mensaje, pero como se utiliza el protocolo HTTPS, la seguridad la proporciona el protocolo HTTPS.

Se utiliza este tipo de autenticación cuando un usuario intenta iniciar sesión. El cliente envía al servidor la petición POST /authentication, con la cabecera “Authentication”.

Esta cabecera contiene el usuario y contraseña, separados por el carácter ‘:’, y posteriormente codificado en base 64. Que estará precedido por la cadena de caracteres “Basic “.

El servidor procesa los datos, y si el usuario tiene permiso, el servidor le enviará un token.

Este token se define a continuación.

- Token JWT:

El token que crea el servidor, está basado en el estándar JWT, que corresponde al RFC 7519 [99]. Está en formato JSON y se divide en tres partes: cabecera, carga útil (del inglés, *payload*) y la firma.

El formato de la cabecera diseñada tiene el siguiente formato:

```
{  
  "algorithm": "sha256",
```

```
    "type": "jwt"
}
```

La carga útil, el siguiente:

```
{
    "username": "admin", (codificado en hexadecimal)
    "expiration_time": 4560,
    "refresh_time": 9456060,
    "token": BASE64_RANDOM_BYTES
}
```

Mediante este diseño, el servidor puede determinar si el token es válido. Para esto, la fecha indicada en la propiedad “refresh_time”, ha de ser menor a la actual; si es mayor, el token no es válido y el usuario tendrá que iniciar sesión de nuevo. Se usan 3 días como límite.

El campo “expiration_time”, indica al servidor que el token ha sido generado recientemente, si la fecha de este campo es menor a la actual, el servidor envía al cliente un nuevo token, de manera que no se tenga que registrar. Si es posterior, el token no se renueva y sigue teniendo autorización, pero el usuario no tendrá la necesidad de registrarse hasta que la fecha de “refresh_time” caduque. Para “expiration_time”, se usan 10 minutos.

Finalmente, se utiliza el algoritmo HMAC junto a la función indicada en el campo “algorithm” de la cabecera, SHA-256 en nuestro caso, y mediante una clave de 768 bits se generan los bytes correspondientes a la firma. El mensaje que utiliza el algoritmo HMAC para crear la firma, es la concatenación de: la cabecera codificada en base 64, el carácter ‘.’ y la carga útil codificada en base 64.

El token final tendrá el siguiente formato:

BASE64_HEADER.BASE64_PAYLOAD.BASE64_SIGANTURE

- Autenticación *bearer*:

Se utiliza la autenticación *bearer* [100], cuando el cliente ha recibido un token. Se enviará en la cabecera “Authentication”, la cadena de caracteres “Bearer ”, sucedido del token JWT, almacenado en almacenamiento local del navegador. De esta manera se puede asegurar el acceso a recursos del servidor.

El sistema desarrollado hace uso de esta autenticación al solicitar el módulo de gestión de cultivos, si tenemos un token.

En este desarrollo no se ha realizado, pero se debería incorporar también en los métodos de la API.

2.3.4.3. HOSTING

Para ejecutar este sistema web, se utiliza un equipo personal con un proceso Node.js que da servicio al servidor. Respecto al sistema de ficheros, el sistema operativo es Windows 10 [101].

El equipo está conectado a un *router* con un punto de acceso a internet. El ISP es Movistar, que nos ha dispuesto de una IP pública dinámica [102]. Por tanto, la IP actual es probable que sea modificada en cualquier momento.

Lo ideal para este sistema sería: obtener una IP privada contactando con el proveedor, utilizar un servicio de gestión de IP's dinámicas o alojar el sistema en un entorno virtual de pago como: el servicio EC2 de Amazon [103] o la empresa Raiola Networks, compañía de *hosting* con servidores en España [104].

3.RESULTADOS Y CONCLUSIONES

Esta memoria ha sido realizada de forma posterior al desarrollo del proyecto, por tanto todo lo que contiene, está basado en lo desarrollado.

Como se ha mencionado, se ha conseguido desarrollar el sistema automático general, como el intercambio de mensajes entre todas las partes, y a nivel específico la mayoría de las funcionalidades, presentes en la memoria. Con excepción de las siguientes funcionalidades:

- Acción de neutralizar. Concretamente la lectura del sensor de pH y los sensores de peso.
- Conexión entre el módulo 2 y 3, mediante internet.

Igualmente, hasta el día de la presentación de la memoria (en 10 días desde que se escribe esto). Se trabajará en estos dos puntos.

A continuación, se muestran una serie de ilustraciones del desarrollo de los diferentes módulos del proyecto:

- Módulo 1:

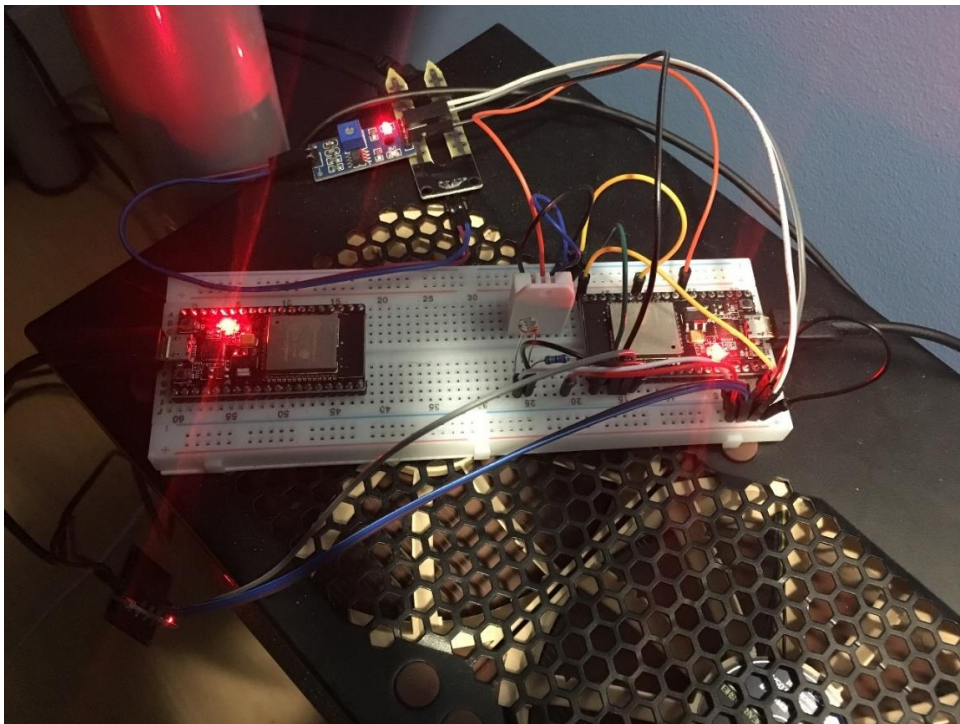


Ilustración 11. Microcontrolador principal y microcontrolador secundario de control de riego en una protoboard.

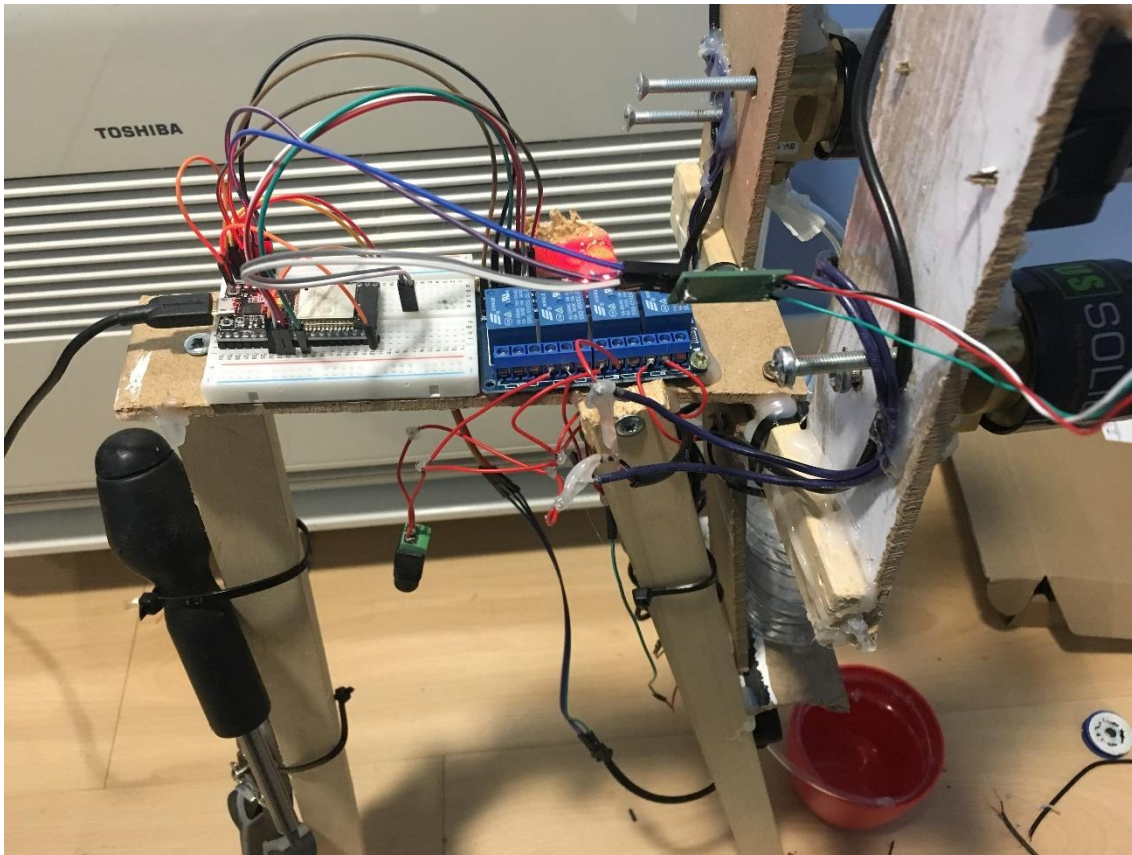


Ilustración 12. Microcontrolador secundario de control de riego en el soporte construido.



Ilustración 13. Vista general del soporte con las botellas, sensores, motores, pesos y electroválvulas.



Ilustración 14. Vista frontal superior del soporte construido.

```
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353

data[0] = '\0';
if (recv(slave_fd, data, 1024, 0) < 0)
{
    fprintf(stderr, "error: the recv call on the slave failed\n");
    close(slave_fd);
    close(client_fd);
    continue;
}

close(slave_fd);

json_data = cJSON_Parse(data);
if (cJSON_IsInvalid(json_data))
{
    fprintf(stderr, "error: the data is not in a json format\n");
    close(client_fd);
    continue;
}

if (!cJSON_IsObject(json_data))
{
    fprintf(stderr, "error: the json data is not of type object\n");
    close(client_fd);
    continue;
}

json_method = cJSON_GetObjectItemCaseSensitive(json_data, "status");
if (cJSON_IsInvalid(json_method))
{
    fprintf(stderr, "error: the json does not have a \"status\" property\n");
    close(client_fd);
    continue;
}

if (!cJSON_IsNumber(json_method))
{
    fprintf(stderr, "error: the json property \"status\" is not of type number\n");
    close(client_fd);
    continue;
}

int status = json_method->valueint;
if (status == 200)
{
    data[0] = '\0';
    strcpy(data, "{\"status\":200}");
    data[strlen(data)] = '\0';

    if (write(client_fd, data, strlen(data) + 1) == -1)
    {
        fprintf(stderr, "error: the write call on the client failed\n");
    }
}

close(client_fd);

return;
}
```

< source file length: 36.452 lines: 1.353 Ln: 1.025 Col: 44 Sel: 0 | 0 Windows (CR LF) UTF-8 IN

Ilustración 15. Código fuente del microcontrolador principal, con 1353 líneas.

```
287 }
288 data[0] = '\0';
289 if(read(client_fd, data, 1024) == -1)
290 {
291     fprintf(stderr, "error: the read call on the client failed\n");
292     close(client_fd);
293     continue;
294 }
295
296 fprintf(stdout, "DATA: %s\n", data);
297
298 json_data = cJSON_Parse(data);
299 if(cJSON_IsInvalid(json_data))
300 {
301     fprintf(stderr, "error: the data is not in a json format\n");
302     close(client_fd);
303     continue;
304 }
305
306 if(!cJSON_IsObject(json_data))
307 {
308     fprintf(stderr, "error: the json data is not of type object\n");
309     close(client_fd);
310     continue;
311 }
312
313 json_method = cJSON_GetObjectItemCaseSensitive(json_data, "method");
314 if (cJSON_IsInvalid(json_method))
315 {
316     fprintf(stderr, "error: the json does not have a \"method\" member\n");
317     close(client_fd);
318     continue;
319 }
320
321 if(!cJSON_IsString(json_method))
322 {
323     fprintf(stderr, "error: the json member \"method\" is not of type string\n");
324     close(client_fd);
325     continue;
326 }
327
328 if (strcmp(json_method->valstring, "get-type") == 0)
329 {
330     data[0] = '\0';
331     strcpy(data, "{\"status\":200,\"type\":\"\"");
332     strcat(data, type);
333     strcat(data, "\"");
334     data[strlen(data)] = '\0';
335
336     fprintf(stdout, "RESPONSE: %s\n", data);
337
338     if (write(client_fd, data, strlen(data) + 1) == -1)
339     {
340         fprintf(stderr, "error: the write call on the client failed\n");
341     }
342     close(client_fd);
343 }
344
345 return;
346 }
347
348 }
```

C source file length: 8.354 lines: 348 Ln: 153 Col: 51 Sel: 0 | 0 Unix (LF) UTF-8 IN

Ilustración 16. Código fuente del microcontrolador secundario de sensores básicos, con 348 líneas.


```
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?
master.c basic_sensors.c ph_controller.c
521 fprintf(stderr, "error: the json does not have a \"properties\" property\n");
522 close(client_fd);
523 continue;
524 }
525
526 if(!cJSON_IsObject(json_method))
527 {
528     fprintf(stderr, "error: the json data is not of type object\n");
529     close(client_fd);
530     continue;
531 }
532
533 json_data = cJSON_GetObjectItemCaseSensitive(json_method, "objective_ph");
534 if (cJSON_IsInvalid(json_data))
535 {
536     fprintf(stderr, "error: the json does not have a \"objective_ph\" property\n");
537 }
538 else
539 {
540     if(!cJSON_IsNumber(json_data))
541     {
542         fprintf(stderr, "error: the json property \"objective_ph\" is not of type number\n");
543     }
544     else
545     {
546         objective_ph = json_data->valuedouble;
547     }
548 }
549
550 json_data = cJSON_GetObjectItemCaseSensitive(json_method, "ph_umbral");
551 if (cJSON_IsInvalid(json_data))
552 {
553     fprintf(stderr, "error: the json does not have a \"ph_umbral\" property\n");
554 }
555 else
556 {
557     if(!cJSON_IsNumber(json_data))
558     {
559         fprintf(stderr, "error: the json property \"ph_umbral\" is not of type number\n");
560     }
561     else
562     {
563         ph_umbral = json_data->valuedouble;
564     }
565 }
566
567 data[] = '\0';
568 sprintf(data, "{\"status\":200}");
569 data[strlen(data)] = '\0';
570
571 fprintf(stdout, "RESPONSE: %s\n", data);
572
573 if (write(client_fd, data, strlen(data) + 1) == -1)
574 {
575     fprintf(stderr, "error: the write call on the client failed\n");
576 }
577 close(client_fd);
578
579 return;
580 }
581
582
```

C source file length: 13.385 lines: 582 Ln: 210 Col: 46 Sel: 21 | 1 Unix (LF) UTF-8 IN

Ilustración 17. Código fuente del microcontrolador secundario de control de riego, con 582 líneas.

```
if(!cJSON_IsObject(json_method))
{
}

fernando@DESKTOP-0EDSG51: ~/esp32/projects/aifarming/master
fernando@DESKTOP-0EDSG51:~/esp32$ cd esp-idf/
fernando@DESKTOP-0EDSG51:~/esp32/esp-idf$ ls
CMakeLists.txt  Kconfig  README.md  components  examples  requirements.txt
CONTRIBUTING.rst  LICENSE  add_path.sh  docs  idf.py  tools
fernando@DESKTOP-0EDSG51:~/esp32/esp-idf$ cd ..
fernando@DESKTOP-0EDSG51:~/esp32$ ls
arduino-esp32  esp-idf  esp-idf  xtensa-esp32-elf
fernando@DESKTOP-0EDSG51:~/esp32$ cd projects/
fernando@DESKTOP-0EDSG51:~/esp32/projects$ ls
aifarming  esp32-raw-data  esp32-sensor-hub
fernando@DESKTOP-0EDSG51:~/esp32/projects$ cd aifarming/
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming$ ls
ld: no input files
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming$ ls
Makefile  build  cmake  sdkconfig  sdkconfig.old
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming$ cd m
master/  middleware/
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming$ cd m
-bash: cd: m: No such file or directory
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming$ cd m
-bash: cd: m: No such file or directory
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming$ cd m
master/  middleware/
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming$ cd m
master/  middleware/
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming$ cd master/
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming/master$ ls
Makefile  build  cmake  sdkconfig  sdkconfig.old
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming/master$ ls main/
component.mk  master.c
fernando@DESKTOP-0EDSG51:~/esp32/projects/aifarming/master$

data[0] = '\0';
printf(data, "{\"status\":200}");
data[strlen(data)-1] = '\0';
```

Ilustración 18. Carpeta de desarrollo del microcontrolador principal, usando Windows Subsystem for Linux.

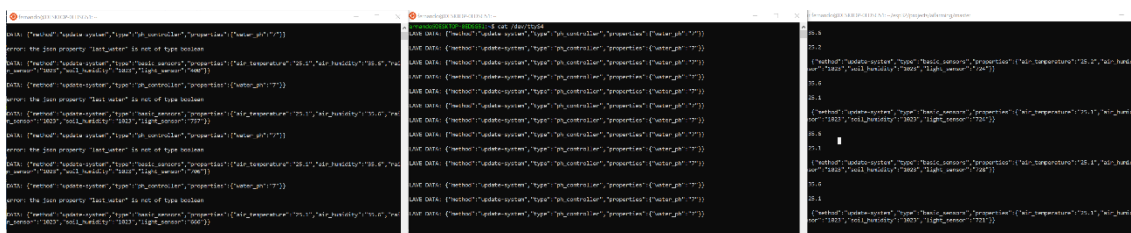


Ilustración 19. Ejemplo de comunicación automática entre los componentes del módulo 1.

Si la imagen no se aprecia, se puede ver la imagen original, disponible en el capítulo de Anexos.

- Módulo 2:

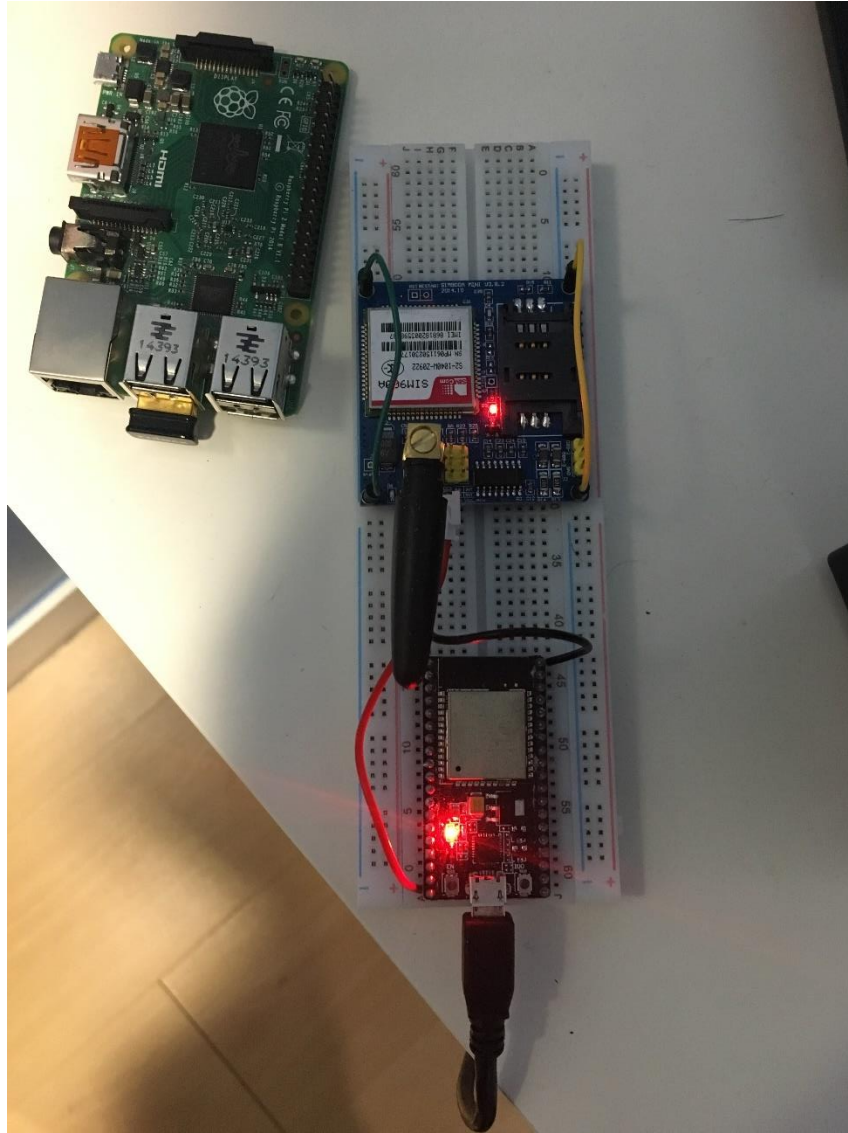


Ilustración 20. Microcontrolador del middleware conectado al módulo Sim900a mini, junto a una Raspberry Pi.


```
middleware.c
1164         fprintf(stderr, "error: the recv call on the slave failed\n");
1165         close(master_fd);
1166         break;
1167     }
1168     fprintf(stdout, "RESPONSE: %s\n", data);
1169     close(master_fd);
1170
1171     json_temp1 = cJSON_Parse(data);
1172     if (cJSON_IsInvalid(json_temp1))
1173     {
1174         fprintf(stderr, "error: the data is not in a json format\n");
1175         break;
1176     }
1177
1178     if (!cJSON_IsObject(json_temp1))
1179     {
1180         fprintf(stderr, "error: the json data is not of the type object\n");
1181         break;
1182     }
1183
1184     json_temp2 = cJSON_GetObjectItemCaseSensitive(json_temp1, "status");
1185     if (cJSON_IsInvalid(json_temp2))
1186     {
1187         fprintf(stderr, "error: the json does not have a \"status\" property\n");
1188         break;
1189     }
1190
1191     if (!cJSON_IsNumber(json_temp2))
1192     {
1193         fprintf(stderr, "error: the json property \"status\" is not of type number\n");
1194         break;
1195     }
1196
1197     if (json_temp2->valueint == 200)
1198     {
1199         status = 200;
1200     }
1201     break;
1202 }
1203
1204     }
1205 }
1206 break;
1207 }
1208 }
1209 }
1210
1211 data[0] = '\0';
1212 sprintf(data, "{\"status\":%d}", status);
1213 data[strlen(data)] = '\0';
1214
1215 if (write(client_fd, data, strlen(data)) == -1)
1216 {
1217     fprintf(stderr, "error: the write call on the client failed\n");
1218 }
1219
1220 close(client_fd);
1221 }
1222 return;
1223 }
1224 }
1225 }
```

C source file length: 33.816 lines: 1.225 Ln: 1 Col: 1 Sel: 0 | 0 Windows (CR LF) UTF-8 IN

Ilustración 21. Código fuente del middleware, con 1225 líneas.

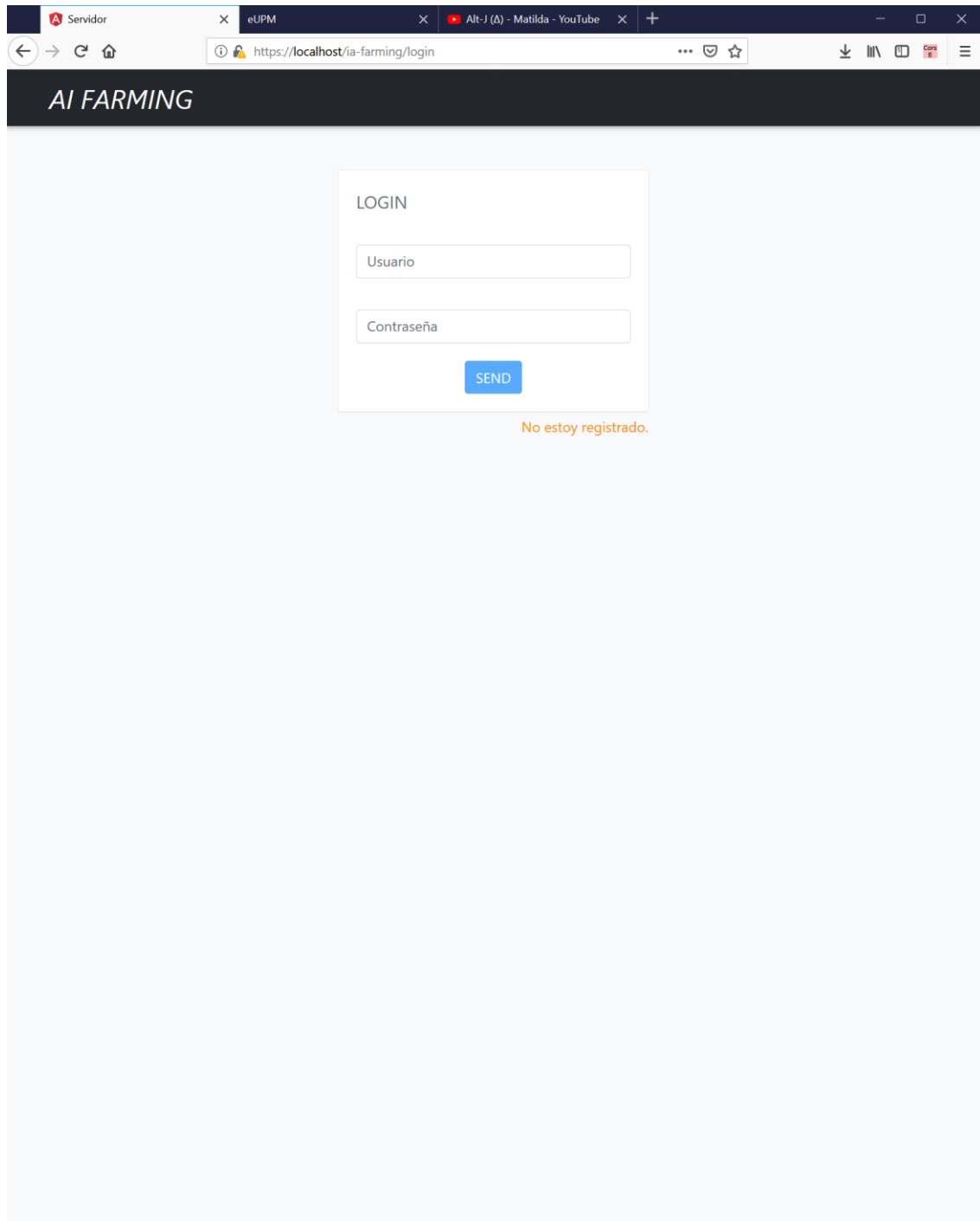
The image displays a multi-window environment illustrating the communication between a Node.js module (Module 1) and several ESP32 slave modules (Module 2) via TCP over a local network.

Windows and Content:

- Top Left (Terminal):** Shows the master module's communication logs. It includes a "SEND TO MASTER" command, a "RESPONSE FROM MASTER" with status 200 and sensor data (air_temperature, air_humidity, rain_sensor, water_ph, light_sensor, last_water), and a "STATION WRITE IN NWS" command. It also shows a "METHOD: send-action" and a "RESPONSE" with status 200.
- Top Right (Node.js command prompt):** Shows the server's log output. It displays "CONNECTED", "WRITE", "DATA" (status 200), and "CLOSED" for multiple connections. It also shows the server sending data back to the master with "SEND" and "WRITE" commands.
- Bottom Left (Terminal):** Shows the logs for three different slave modules (basic_sensors, basic_sensors, and ph_controller). Each log shows "SLAVE DATA" being received by the master, including sensor readings and status information.
- Bottom Right (Code Editor):** Displays the Node.js script for the TCP server. The script uses the 'net' module to create a socket, listen on port 40500, and handle incoming connections. It defines several message objects (message1 to message5) representing different commands and responses, and uses 'socket.write' to send them back to the clients.

Ilustración 22. Ejemplo de comunicación automática entre el módulo 1 y 2, cuando se envían peticiones TCP al módulo 2, estando conectado a la red local del módulo 2.

- Módulo 3:



The image shows a web browser window with the address bar displaying `https://localhost/ia-farming/login`. The browser tabs include 'Servidor', 'eUPM', and 'Alt-J (Δ) - Matilda - YouTube'. The page has a dark header with the text 'AI FARMING'. The main content area is light gray and contains a white login form titled 'LOGIN'. The form has two input fields: 'Usuario' and 'Contraseña', followed by a blue 'SEND' button. Below the form, there is a link that says 'No estoy registrado.' in orange text.

Ilustración 23. Formulario de inicio de sesión del módulo de gestión de usuarios de la aplicación web.

The image shows a web browser window with the address bar displaying `https://localhost/ia-farming/login`. The browser tabs include 'Servidor', 'eUPM', and 'Alt-J (A) - Matilda - YouTube'. The website has a dark header with the text 'AI FARMING'. The main content area is light gray and contains a white registration form titled 'REGISTRATION'. The form has four input fields: 'Usuario', 'Email', 'Contraseña', and 'Confirmar Contraseña'. Below these fields is a blue 'SEND' button. Underneath the button, the text 'Ya estoy registrado.' is displayed in orange. The browser's address bar also shows standard navigation icons (back, forward, refresh, home) and a menu icon on the right.

AI FARMING

REGISTRATION

Usuario

Email

Contraseña

Confirmar Contraseña

SEND

Ya estoy registrado.

Ilustración 24. Formulario de registro del módulo de gestión de usuarios de la aplicación web.

Servidor x eUPM x Alt-J (Δ) - Matilda - YouTube x +

https://localhost/ia-farming

AI FARMING

AÑADIR CULTIVO AUTÓNOMO ✕

INTRODUZCA LAS PROPIEDADES DEL SOCKET TCP DEL MIDDLEWARE

IP: Puerto:

Usuario: Contraseña:

SELECCIONE LAS CARACTERÍSTICAS DEL CULTIVO

Añada un nombre al cultivo:

Seleccione el tipo de cultivo:

✕

ACEPTAR

Ilustración 25. Componente del módulo de gestión de cultivos de la aplicación web.

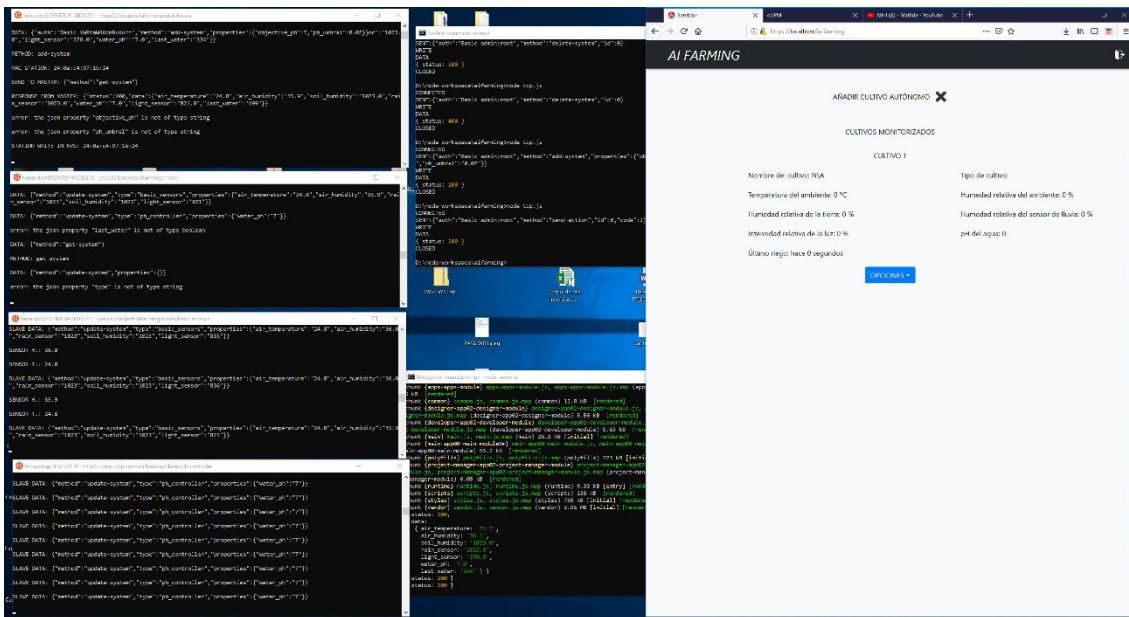


Ilustración 26. Envío de mensajes entre todas las partes al añadir un cultivo.

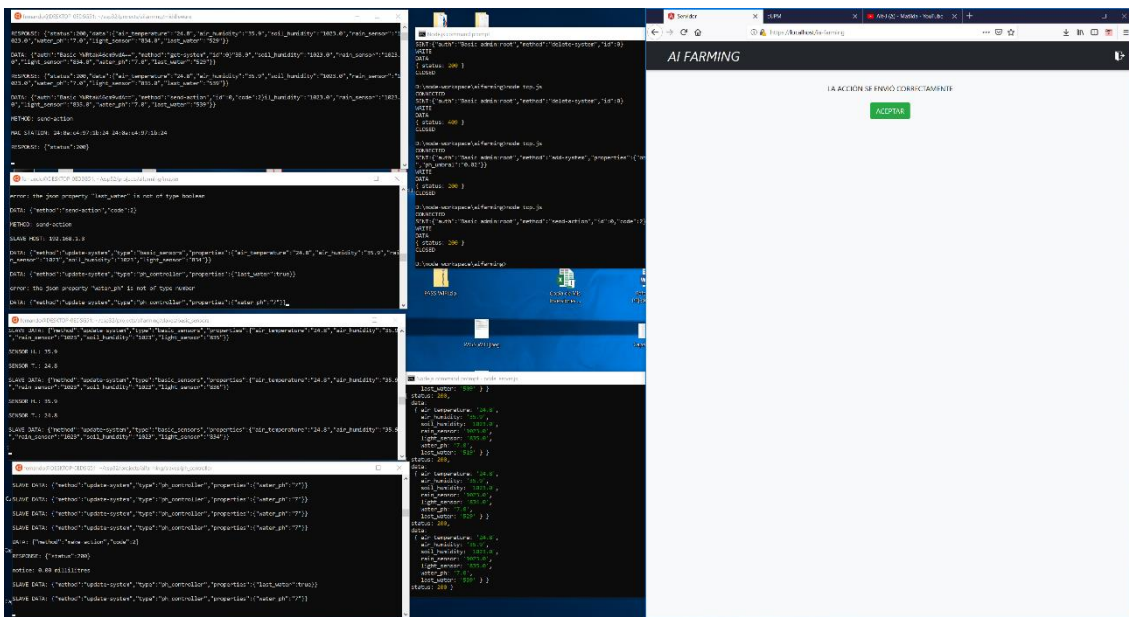


Ilustración 27. Envío de mensajes entre todas las partes, al realizar la acción de regar.



Ilustración 28. Token JWT en el almacenamiento local del navegador.

The screenshot shows the Chrome DevTools Network tab. It displays a list of network requests, all of which are 'check_system' requests. The table below summarizes the data shown in the screenshot.

Estado	Método	Archivo	Dominio	Causa	Tipo	Transferido	Tamaño	0 ms	10,24 s	20,48 s	30,72 s
200	POST	check_system	localhost	xhr	html	326 B	120 B	244 ms			
200	POST	check_system	localhost	xhr	html	329 B	123 B	214 ms			
200	POST	check_system	localhost	xhr	html	323 B	117 B		244 ms		
200	POST	check_system	localhost	xhr	html	326 B	120 B			223 ms	

Ilustración 29. Petición automática de mensajes de método "check_system", para actualizar los datos de monitorización del cultivo.

Los resultados del proyecto, como se ha podido comprobar mediante las fotos, han sido, en general, buenos. Aunque no me ha dado tiempo a optimizar y terminar todas las funcionalidades, este proyecto me ha gustado mucho, por lo que en mi tiempo libre me dedicaré a depurarlo y optimizarlo.

Cabe destacar que el microcontrolador ESP-32 está en desarrollo, y se han encontrado ciertos problemas de desarrollo, pero al final se ha encontrado la forma de ingeniarlas para arreglar estos problemas, cuya solución ha consistido en: almacenamiento en NVS y reiniciar.

La conclusión global que saco, con la realización de este proyecto:

“Actualmente se habla de una revolución industrial, revolución 4.0 la llaman. Esta revolución, es una revolución tecnológica, en la que la digitalización y la automatización de procesos industriales, van a estar a la orden del día. Por este motivo, considero que este proyecto ha sido realmente útil para la iniciación en las áreas del internet de las cosas y la inteligencia artificial, áreas de la informática, áreas de la electrónica, áreas de la digitalización, áreas de la globalización, áreas del mañana...”

Fernando Martí de Mateo

4.ANEXOS

Se ha creado un repositorio en GitHub, para alojar el código fuente de los diferentes módulos. El repositorio es: <https://github.com/MrFridunath/aifarming>

La estructura del repositorio es la siguiente:

- README.md: fichero de introducción al repositorio.
- LICENCIA: licencia.
- memoria.pdf: esta misma memoria.
- src: carpeta con el código fuente de los módulos.
 - mod1: código fuente del módulo 1.
 - master: código fuente del microcontrolador principal.
 - slaves:
 - basic_sensors: código fuente del microcontrolador secundario de sensores básicos.
 - ph_controller: código fuente del microcontrolador secundario de control de riego.
 - mod2: código fuente del módulo 2.
 - mod3: código fuente del módulo 3.
 - app: código fuente de la aplicación web.
 - server: código fuente del servidor web.
- img: todas las imágenes de esta memoria.
 - 1 intro: imágenes del capítulo de introducción.
 - 2.1 mod1: imágenes del módulo 1 del capítulo de desarrollo.
 - 2.2 mod2: imágenes del módulo 2 del capítulo de desarrollo.
 - 2.3 mod3: imágenes del módulo 3 del capítulo de desarrollo.
 - 3 res: imágenes del capítulo de resultados y conclusiones.

Licencia: queda prohibida la copia, la distribución y la comercialización del código y de esta memoria o partes de estos o cualquier otro recurso relacionado no mencionado; sin previa autorización expresa, clara, concisa, sin posibles ambigüedades, escrita a mano y firmada, por parte del autor: Fernando Martí de Mateo; la cual da derecho a la mera copia parcial de la memoria, sin un fin comercial y referenciando debidamente las partes utilizadas correspondientes.

5.BIBLIOGRAFÍA

- [1] «redhat,» [En línea]. Available: <https://www.redhat.com/en/topics/middleware/what-is-middleware>. [Último acceso: 13 Enero 2019].
- [2] M. Rouse, «SearchMobileComputing,» [En línea]. Available: <https://searchmobilecomputing.techtarget.com/definition/GPRS>. [Último acceso: 13 Enero 2019].
- [3] R. Margaret, «sSearchMobileComputing,» [En línea]. Available: <https://searchmobilecomputing.techtarget.com/definition/SIM-card>. [Último acceso: 13 Enero 2019].
- [4] IBM, «IBM,» [En línea]. Available: <https://www.ibm.com/cloud/learn/what-is-cloud-computing>. [Último acceso: 13 Enero 2019].
- [5] «Angular,» [En línea]. Available: <https://angular.io/>. [Último acceso: 13 Enero 2019].
- [6] «nodejs.org,» [En línea]. Available: <https://nodejs.org/en/about/>. [Último acceso: 13 Enero 2019].
- [7] DECSAI (Departamento de Ciencias de la Computación e I.A.), «Universidad de Granada,» [En línea]. Available: <https://elvex.ugr.es/decsai/internet/pdf/2%20Arquitecturas%20de%20red.pdf>. [Último acceso: 13 Enero 2019].
- [8] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/PH>. [Último acceso: 13 Enero 2019].
- [9] «CropNutrition,» [En línea]. Available: <https://www.cropnutrition.com/efu-soil-ph>. [Último acceso: 13 Enero 2019].
- [10] «Wikipedia,» [En línea]. Available: <https://es.wikipedia.org/wiki/Wifi>. [Último acceso: 13 Enero 2019].
- [11] «FEC,» [En línea]. Available: http://www.fecegypt.com/uploads/dataSheet/1480854383_water%20and%20soil.pdf. [Último acceso: 13 Enero 2019].

- [12] «botnroll.com,» [En línea]. Available: <https://www.botnroll.com/en/biometrics/2354-rain-sensor-hl-83-.html>. [Último acceso: 13 Enero 2019].
- [13] Aosong Electronics Corporation, «sparkfun,» [En línea]. Available: <https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>. [Último acceso: 13 Enero 2019].
- [14] YIFA the plastics Ltd, «HOBBY ELECTRONICS,» [En línea]. Available: <https://www.hobbytronics.co.uk/datasheets/sensors/YF-S201.pdf>. [Último acceso: 13 Enero 2019].
- [15] AVIA Semiconductor, «MOISER ELECTRONICS,» [En línea]. Available: https://www.mouser.com/ds/2/813/hx711_english-1022875.pdf. [Último acceso: 13 Enero 2019].
- [16] «KTH,» [En línea]. Available: <https://www.kth.se/social/files/54ef17dbf27654753f437c56/GL5537.pdf>. [Último acceso: 13 Enero 2019].
- [17] AtlasScientific, «AtlasScientific,» [En línea]. Available: https://www.atlas-scientific.com/_files/_datasheets/_probe/pH_probe.pdf. [Último acceso: 13 Enero 2019].
- [18] andwind, «eBay,» [En línea]. Available: <https://www.ebay.com/itm/Soil-Hygrometer-Detection-Module-Moisture-Sensor-FC-28-DC-3-3V-5V-Y5Z2/112535516172?epid=2193434611&hash=item1a33a3b40c:g:7rMAAOSwZEIzm2pd:rk:2:pf:0>. [Último acceso: 13 Enero 2019].
- [19] Provide The Best, «Amazon,» [En línea]. Available: https://www.amazon.es/Providethebest-Raindrops-Detection-Weather-Humidity/dp/B07K262NZH/ref=sr_1_4?ie=UTF8&qid=1547378724&sr=8-4&keywords=rain+sensor. [Último acceso: 13 Enero 2019].
- [20] good-module, «eBay,» [En línea]. Available: https://www.amazon.es/Providethebest-Raindrops-Detection-Weather-Humidity/dp/B07K262NZH/ref=sr_1_4?ie=UTF8&qid=1547378724&sr=8-4&keywords=rain+sensor. [Último acceso: 13 Enero 2019].
- [21] bifengsi-3, «eBay,» [En línea]. Available: <https://www.ebay.com/itm/YF-S201-1-30L-min-Water-Flow-Hall-Counter-Sensor-Water-control-Water-Flow-Rate/122990834756?hash=item1ca2d33444:rk:1:pf:0>. [Último acceso: 13 Enero 2019].

- [22] flyfun_diy, «eBay,» [En línea]. Available: <https://www.ebay.com/itm/1-4-10Pcs-1-2-3-5-10-20KG-Load-Cell-Weight-Sensor-HX711-AD-Weighing-Module/172830090237?hash=item283d79e3fd:rk:1:pf:0&var>. [Último acceso: 13 Enero 2019].
- [23] babaoshop, «eBay,» [En línea]. Available: <https://www.ebay.com/itm/20pcs-5539-GL5539-Photo-Light-Dependent-Resistor-Photoresistor-Brand-New/253975097115?hash=item3b22186f1b:g:9hIAAOxyuPtRANMp:rk:2:pf:0>. [Último acceso: 13 Enero 2019].
- [24] worldchips, «eBay,» [En línea]. Available: https://www.ebay.com/itm/Blue-Black-PH-Electrode-Probe-Liquid-PH-Value-Detection-Sensor-BNC-Module/322950904241?hash=item4b315fb9b1:m:mP5IAxz8ImPEda6SpfS_OFA:rk:6:pf:0. [Último acceso: 13 Enero 2019].
- [25] DIGITEN, «Amazon,» [En línea]. Available: https://www.amazon.es/gp/product/B016MP1HX0/ref=ppx_yo_dt_b_asin_title_o00_o00_s00?ie=UTF8&psc=1. [Último acceso: 13 Enero 2019].
- [26] Joyfray International, «Amazon,» [En línea]. Available: https://www.amazon.es/gp/product/B01NBUFFEG/ref=ppx_yo_dt_b_asin_title_o07_o00_s00?ie=UTF8&psc=1. [Último acceso: 13 Enero 2019].
- [27] solectroshop, «eBay,» [En línea]. Available: https://www.ebay.com/itm/Mini-Bomba-Sumergible-de-Agua-DC-12V-240L-h-3-6W-Arduino-Motor-Water-Pump-M0110/201969338740?ssPageName=STRK%3AMEBIDX%3AIT&_trksid=p2060353.m2749.l2649. [Último acceso: 13 Enero 2019].
- [28] NodeMcu, «Zerynth Docs,» [En línea]. Available: https://docs.zerynth.com/latest/official/board.zerynth.nodemcu_esp32/docs/index.html. [Último acceso: 13 Enero 2019].
- [29] bestpurchase2017, «eBay,» [En línea]. Available: https://www.ebay.com/itm/ESP-32-ESP32S-Development-Board-2-4GHz-WiFi-Bluetooth-Antenna-CP2102-Module-POP/264102703499?_trkparms=aid%3D555018%26algo%3DPL.SIM%26ao%3D1%26asc%3D20131003132420%26meid%3D8e0659d041354b8381d018c873598b0a%26pid%3D100005%26rk%3D1%26. [Último acceso: 13 Enero 2019].
- [30] Espressif Systems, «Electro Schematics,» [En línea]. Available: <https://www.electroschematics.com/wp-content/uploads/2015/02/esp8266-datasheet.pdf>. [Último acceso: 13 Enero 2019].

- [31] good-module, «eBay,» [En línea]. Available: <https://www.ebay.com/itm/ESP8266-ESP-12-WeMos-D1-Mini-WIFI-Dev-Kit-Development-Board-NodeMCU-Lua-GM/262457780584?hash=item3d1bb3e968:g:To8AAOSwepla-Pd4:rk:6:pf:0>. [Último acceso: 13 Enero 2019].
- [32] Espressif Systems, «GitHub,» [En línea]. Available: <https://github.com/espressif/esp-idf>. [Último acceso: 13 Enero 2019].
- [33] Espressif Systems, «GitHub,» [En línea]. Available: <https://github.com/espressif/arduino-esp32>. [Último acceso: 13 Enero 2019].
- [34] Espressif Systems, «GitHub,» [En línea]. Available: https://github.com/espressif/arduino-esp32/blob/master/docs/esp-idf_component.md. [Último acceso: 13 Enero 2019].
- [35] Espressif Systems, «Read the Docs,» [En línea]. Available: <https://docs.espressif.com/projects/esp-idf/en/stable/get-started/>. [Último acceso: 13 Enero 2019].
- [36] Espressif Systems, «Read the Docs,» [En línea]. Available: <https://docs.espressif.com/projects/esp-idf/en/stable/api-guides/wifi.html>. [Último acceso: 13 Enero 2019].
- [37] F. J. Moral Parlón, «Archivo Digital UPM,» [En línea]. Available: http://oa.upm.es/51599/1/TFG_FRANCISCO_JESUS_MORAL_PARLON.pdf. [Último acceso: 13 Enero 2019].
- [38] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/Bluetooth>. [Último acceso: 13 Enero 2019].
- [39] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Asynchronous_communication. [Último acceso: 13 Enero 2019].
- [40] Oracle, «Java Documentation,» [En línea]. Available: <https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>. [Último acceso: 13 Enero 2019].
- [41] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Internet_protocol_suite. [Último acceso: 13 Enero 2019].

- [42] «TechTerms,» [En línea]. Available: <https://techterms.com/definition/ssid>. [Último acceso: 13 Enero 2019].
- [43] Microsoft, «Microsoft,» [En línea]. Available: <https://docs.microsoft.com/en-us/windows-server/networking/technologies/dhcp/dhcp-top>. [Último acceso: 13 Enero 2019].
- [44] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Application_programming_interface. [Último acceso: 13 Enero 2019].
- [45] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Último acceso: 13 Enero 2019].
- [46] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes. [Último acceso: 13 Enero 2019].
- [47] D. Crockford, «Introducing JSON,» [En línea]. Available: <https://www.json.org/>. [Último acceso: 13 Enero 2019].
- [48] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/IEEE_802.11. [Último acceso: 13 Enero 2019].
- [49] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Internet_Protocol. [Último acceso: 13 Enero 2019].
- [50] «WIRESHARK,» [En línea]. Available: <https://www.wireshark.org/>. [Último acceso: 13 Enero 2019].
- [51] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/IEEE_802.11i-2004. [Último acceso: 13 Enero 2019].
- [52] KU Leuven, «Breaking WPA2,» [En línea]. Available: <https://www.krackattacks.com/>. [Último acceso: 13 Enero 2019].
- [53] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Direcci%C3%B3n_MAC. [Último acceso: 13 Enero 2019].
- [54] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Public-key_cryptography. [Último acceso: 13 Enero 2019].

- [55] Espressif Systems, «Read the Docs,» [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/system/system.html#_CPPv210esp_randomv. [Último acceso: 13 Enero 2019].
- [56] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/HMAC>. [Último acceso: 13 Enero 2019].
- [57] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/SHA-2>. [Último acceso: 13 Enero 2019].
- [58] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/Base64>. [Último acceso: 13 Enero 2019].
- [59] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Direct_current. [Último acceso: 13 Enero 2019].
- [60] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Pull-up_resistor. [Último acceso: 13 Enero 2019].
- [61] gosouth, «GitHub,» [En línea]. Available: <https://github.com/gosouth/DHT22>. [Último acceso: 13 Enero 2019].
- [62] «LibreTexts,» [En línea]. Available: [https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_\(Physical_and_Theoretical_Chemistry\)/Acids_and_Bases/Acids_and_Bases_in_Aqueous_Solutions/The_pH_Scale/Temperature_Dependence_of_the_pH_of_pure_W](https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Acids_and_Bases/Acids_and_Bases_in_Aqueous_Solutions/The_pH_Scale/Temperature_Dependence_of_the_pH_of_pure_W). [Último acceso: 13 Enero 2019].
- [63] DFRobot, «GitHub,» [En línea]. Available: https://github.com/DFRobot/DFRobot_PH. [Último acceso: 13 Enero 2019].
- [64] bogde, «GitHub,» [En línea]. Available: <https://github.com/bogde/HX711>. [Último acceso: 13 Enero 2019].
- [65] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Hall_effect. [Último acceso: 13 Enero 2019].
- [66] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/Interrupt>. [Último acceso: 13 Enero 2013].

- [67] E. Gómez, «RincónIngenieril,» 31 Enero 2017. [En línea]. Available: <https://www.rinconingenieril.es/flanco-subida-bajada/>. [Último acceso: 13 Enero 2019].
- [68] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Interrupt_handler. [Último acceso: 13 Enero 2019].
- [69] «Khan Academy,» [En línea]. Available: <https://es.khanacademy.org/science/biology/water-acids-and-bases/acids-bases-and-ph/a/acids-bases-ph-and-buffers>. [Último acceso: 13 Enero 2019].
- [70] T. Helmenstine, «ThoughtCo.,» 1 Enero 2018. [En línea]. Available: <https://www.thoughtco.com/neutralizing-a-base-with-acid-609579>. [Último acceso: 13 Enero 2019].
- [71] Raspberry Pi, «Raspberry pi,» [En línea]. Available: <https://www.raspberrypi.org/>. [Último acceso: 13 Enero 2019].
- [72] gowin_electronic, «eBay,» [En línea]. Available: <https://www.ebay.com/itm/Raspberry-pi-Zero-V1-3-W-Camera-Connector-Pi0-Version-1-3-W-board/312187932092?hash=item48afd9cdbc:m:mfYY9gp7t-7ytx4uaoioAWQ:rk:15:pf:0>. [Último acceso: 13 Enero 2019].
- [73] SIMCom, «SIMCOM.EE,» [En línea]. Available: <https://simcom.ee/modules/gsm-gprs/sim900/>. [Último acceso: 13 Enero 2019].
- [74] Happymacer, «instructables,» [En línea]. Available: <https://www.instructables.com/id/Using-the-Sim900sim900A-mini-module-with-Arduino-U/>. [Último acceso: 19 Enero 2019].
- [75] R. Pujar, «Embedded World,» 31 Diciembre 2015. [En línea]. Available: <https://www.raviyp.com/embedded/202-using-sim900-as-tcp-server-read-this-first>. [Último acceso: 13 Enero 2019].
- [76] Saddam, «CircuitDigest,» [En línea]. Available: <https://circuitdigest.com/microcontroller-projects/send-data-to-web-server-using-sim900a-arduino>. [Último acceso: 13 Enero 2019].
- [77] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Access_Point_Name. [Último acceso: 13 Enero 2019].

- [78] «Kaspersky lab,» [En línea]. Available: <https://www.kaspersky.com/resource-center/preemptive-safety/public-wifi-risks>. [Último acceso: 13 Enero 2019].
- [79] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Basic_access_authentication. [Último acceso: 13 Enero 2019].
- [80] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/AngularJS>. [Último acceso: 13 Enero 2019].
- [81] Angular, «GitHub,» [En línea]. Available: <https://github.com/angular/angular/blob/master/LICENSE>. [Último acceso: 13 Enero 2019].
- [82] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>. [Último acceso: 13 Enero 2019].
- [83] Mozilla, «MDN web docs,» [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3>. [Último acceso: 13 Enero 2019].
- [84] Mozilla, «MDN web docs,» [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5>. [Último acceso: 13 Enero 2019].
- [85] «TypeScript,» [En línea]. Available: <https://www.typescriptlang.org/>. [Último acceso: 13 Enero 2019].
- [86] «Express,» [En línea]. Available: <https://expressjs.com/>. [Último acceso: 13 Enero 2019].
- [87] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Internet_service_provider. [Último acceso: 13 Enero 2019].
- [88] «Wikipedia,» [En línea]. Available: [https://en.wikipedia.org/wiki/Firewall_\(computing\)](https://en.wikipedia.org/wiki/Firewall_(computing)). [Último acceso: 13 Enero 2019].
- [89] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/HTTPS>. [Último acceso: 13 Enero 2019].

- [90] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Responsive_web_design. [Último acceso: 13 Enero 2019].
- [91] «Angular,» [En línea]. Available: <https://angular.io/guide/forms-overview>. [Último acceso: 13 Enero 2019].
- [92] «Angular,» [En línea]. Available: <https://angular.io/guide/form-validation>. [Último acceso: 13 Enero 2019].
- [93] «MDN web docs,» [En línea]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/301>. [Último acceso: 13 Enero 2019].
- [94] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer. [Último acceso: 13 Enero 2019].
- [95] «UCI,» [En línea]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. [Último acceso: 13 Enero 2019].
- [96] «OpenSSL,» [En línea]. Available: <https://www.openssl.org/>. [Último acceso: 13 Enero 2019].
- [97] «Wikipedia,» [En línea]. Available: <https://en.wikipedia.org/wiki/X.509>. [Último acceso: 13 Enero 2019].
- [98] «THE VERGE,» [En línea]. Available: <https://www.theverge.com/2013/12/20/5231006/nsa-paid-10-million-for-a-back-door-into-rsa-encryption-according-to>. [Último acceso: 13 Enero 2019].
- [99] «JWT,» [En línea]. Available: <https://jwt.io/>. [Último acceso: 13 Enero 2019].
- [100] «Swagger,» [En línea]. Available: <https://swagger.io/docs/specification/authentication/bearer-authentication/>. [Último acceso: 13 Enero 2019].
- [101] «Microsoft,» [En línea]. Available: <https://docs.microsoft.com/es-es/windows/uwp/>. [Último acceso: 13 Enero 2019].

- [102] S. Arteaga, «Computer Hoy,» 31 Enero 2018. [En línea]. Available: <https://computerhoy.com/paso-a-paso/internet/como-comprobar-si-tenes-ip-dinamica-como-cambiarlo-78039>. [Último acceso: 13 Enero 2019].
- [103] Amazon, «aws,» [En línea]. Available: https://aws.amazon.com/es/ec2/?sc_channel=PS&sc_campaign=acquisition_ES&sc_publisher=google&sc_medium=ACQ-P%7CPS-GO%7CBrand%7CDesktop%7CSU%7CCompute%7CEC2%7CES%7CEN%7CText&sc_content=ec2_e&sc_detail=ec2%20amazon&sc_category=Compute&sc_segment=293645893172. [Último acceso: 13 Enero 13].
- [104] «Raiola Networks,» [En línea]. Available: <https://raiolanetworks.es/>. [Último acceso: 13 Enero 2019].

