DISEÑO DE UN SISTEMA AUTOMATIZADO DE RECICLAJE, TRATAMIENTO Y REUTILIZACION DE AGUAS LLUVIAS Y GRISES PARA UNA VIVIENDA

DIANNETH CASTRILLON BENITEZ
WILLIAM ALBERTO SUAZA LOSADA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
NEIVA
2016

DISEÑO DE UN SISTEMA AUTOMATIZADO DE RECICLAJE, TRATAMIENTO Y REUTILIZACION DE AGUAS LLUVIAS Y GRISES PARA UNA VIVIENDA.

DIANNETH CASTRILLON BENITEZ
WILLIAM ALBERTO SUAZA LOSADA

PROYECTO DE GRADO PRESENTADO PARA OPTAR
AL TÍTULO DE INGENIERO ELECTRÓNICO

DIRECTOR:
ING. EDILBERTO POLANIA PUENTES
INGENIERO ELECTRÓNICO.
DOCENTE UNIVERSIDAD SURCOLOMBIANA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
INGENIERÍA ELECTRÓNICA
NEIVA
2018

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del primer jurado

_____

Firma del segundo jurado

Neiva, Abril de 2018

AGRADECIMIENTOS

Primero a Dios por brindarnos la fuerza necesaria para poder culminar los estudios luego de estos largos años de esfuerzo y sacrificio y a su vez estar bendiciéndonos en todo lo emprendido en el trasegar de nuestras vidas.

A nuestros padres por su apoyo incondicional porque a pesar de tantos obstáculos siempre confiaron en nosotros. Y a todas las personas que de una u otra manera participaron activa y pasivamente en la realización del presente trabajo.

Al director Ing. Edilberto Polania Puentes; por su compromiso y buena gestión en las asesorías y sugerencias realizadas al presente trabajo de grado, a los diferentes docentes de la carrera, por brindarnos su apoyo desinteresado en los momentos de la Academia y por estar siempre allí con una voz de aliento para poder continuar en este tesoro que es el estudio.

TABLA DE CONTENIDO

LISTA DE TABLAS 7

LISTA DE FIGURAS ……..…….....8

RESUMEN 11

ABSTRACT……..……13

1. INTRODUCCIÓN 165

2. MARCO TEORICO 176

2.1. Antecedentes..... 176

2.2. Marco Legal 187

3. MARCO CONCEPTUAL. 198

3.1. Filtros de Carbón Activado:..... 198

3.2. Electroválvulas 209

3.3. Electrobombas 222

3.4. Microcontrolador ESP32 22

3.5. Interruptores de nivel 254

3.6. Lámparas Ultravioleta…… ……26

3.7. Balastro Electrónico 298

3.8. Sensor de ultrasonidos..... 309

3.8.1. Sensor SFR05.....30

4. DESCRIPCION DEL SISTEMA..... 34

4.1. Montaje y Explicacion del Sistema.…….……34

4.2. Desarrollo y explicacion del circuito ……..…….42

4.2.1. Fuente de Poder de 5V ……..……42

4.2.2. Etapa de Control……..……..…..……45

4.2.3. Codigo Fuente ESP32-WROOM-32..…….... ……..……45

4.2.4. Etapa de Aislamiento entre etapa de Control y de Potencia……....51

4.2.5. Etapa de Potencia…….……..…53

4.2.6. Diagrama Completo del Circuito…….……..…..56

5. SISTEMA DE BASE DE DATOS Y APLICACIÓN MOVIL “AQUA VIVA” 598
5.1 Firebase……..……...……58
5.2. Estructura de la base de datos….. ……..……59
5.3. Herramientas de desarrollo de la aplicación móvil…… ……..……66
5.4. Aplicación móvil…….……67
5.4.1 Páginas de la Aplicación móvil…….……68
5.4.2 Código fuente de la aplicación AQUA VIVA ….……73
6. COSTOS DEL SISTEMA..... 743
7. CONCLUSIONES Y RECOMENDACIONES. 744
8. REFERENCIAS BIBLIOGRAFICAS 755
9. ANEXOS..... 776
9.1 ANEXO 1. Código fuente ESP32-WROOM-32…… ……76
9.2 ANEXO 2.Código fuente de la aplicación AQUA VIVA……...116

LISTA DE TABLAS

TABLA 1. Hoja de datos de la Electroválvula……22

TABLA 2. Parámetros de Lámparas Ultravioleta……..29

TABLA 3. Costos del Sistema……74

LISTA DE FIGURAS

FIGURA 1. Filtro de Carbón Activo y de membranas……...20

FIGURA 2. Electroválvula……..21

FIGURA 3. Electroválvula……..21

FIGURA 4. Electrobomba……..23

FIGURA 5. Diagrama de bloque del ESP32- DEVKITC……...24

FIGURA 5.1 Diagrama de Pines del ESP32- DEVKITC…….……....25

FIGURA 6. Interruptor de Nivel ……26

FIGURA 7. Lámpara Ultravioleta……..28

FIGURA 8. Balastro Electrónico……..…30

FIGURA 9. Sensor por Ultrasonidos……31

FIGURA 10. Sensor SFR05……32

FIGURA 11. Diagrama de conexión del SFR05 en modo 1……..33

FIGURA 12. Diagrama de tiempos del SFR05 en modo 1…….33

FIGURA 13. Diagrama de conexión del SFR05 en modo 2……...34

FIGURA 14. Diagrama de tiempos del SFR05 en modo 2…….35

FIGURA 15. Capa inferior del filtro de gravilla…….36

FIGURA 16. Capa intermedia del filtro de gravilla……...37

FIGURA 17. Capa superior de filtro de gravilla……....37

FIGURA 18. Tanque 0 del sistema……38

FIGURA 19. Tanque 1 y 2 del sistema……..…38

FIGURA 20. Vista interior del tanque 2……39

FIGURA 21. Tanque 3 del sistema……...41

FIGURA 22. Sensor SFR05 en el tanque 3……..…42

FIGURA 23. Sistema completo……..43

FIGURA 24. Fuente de poder de 5v……..43

FIGURA 25. Diagrama ESP32……46

FIGURA 26. Señal de red ESP32……..47

FIGURA 27. Página Web1 del ESP32……..48

FIGURA 28. Página Web2 del ESP32……..48

FIGURA 29. Conexión de pulsadores al ESP32……..49

FIGURA 30. Conexión de Interruptores al ESP32……..…49

FIGURA 31. Convertidor de nivel Lógico……..50

FIGURA 32. Conexión del convertidor de nivel Lógico al ESP32……50

FIGURA 33. Diagrama del SRF05……51

FIGURA 34. Conexión de Leds al ESP32……52

FIGURA 35. Encapsulado MOC 3041……53

FIGURA 36. Diagrama MOC 3041……53

FIGURA 37. Conexión del MOC 3041 al ESP32……53

FIGURA 38. Conexión del BT139 al MOC 3041……..54

FIGURA 39. Diagrama de Conexión de lámparas UV al Balastro……57

FIGURA 40. Diagrama del Circuito Electrónico Completo……58

FIGURA 41. Base de datos REALTIME DATABASE – FIREBASE……60

FIGURA 42. Nodo “clave-sistemas”……...61

FIGURA 43. Nodo “sistemas-usuarios”……62

FIGURA 44. Nodo data del sistema……..…63

FIGURA 45. Nodo “T3”……64

FIGURA 46. Nodo “config”……..65

FIGURA 47. Nodo “data”……67

FIGURA 48. Icono y splash de aplicación móvil…….68

FIGURA 49. Página de registro de usuarios……....69

FIGURA 50. Página de inicio de sesión……70

FIGURA 51. Página para registrar sistemas……70

FIGURA 52. Menú lateral sin sistema seleccionado……...71

FIGURA 53. Página inicial del sistema……..72

FIGURA 54. Menú lateral con sistema seleccionado……..72

FIGURA 55. Páginas de los tanques…….73

RESUMEN

El presente proyecto de investigación permite utilizar a los autores del mismo los conocimientos adquiridos durante la carrera de Ingeniería Electrónica, así como los vistos en la materia de Instrumentación al controlar el nivel de tanques, electroválvulas y otros dispositivos electrónicos.

Se propone un sistema para dar un mejor uso al agua en viviendas, reciclando aquellas aguas que han sido menos contaminadas y aplicarles un tratamiento adecuado para dejarlas en un estado óptimo de reutilización en algunas actividades de los hogares.

Aunque la superficie del planeta tierra está cubierta en un 70% de agua aproximadamente, menos del 1% de esa cantidad es potable; es por esto que en algunos países del mundo la escasez de este recurso es evidente, y aunque Colombia sea uno de los pocos países privilegiados al disponer de una gran cantidad de agua dulce y potable, todos sus ciudadanos han atravesado en algún momento por la indeseable experiencia de no disponer de agua potable para satisfacer sus necesidades básicas, algunos por horas, otros por días y no pocos por semanas y meses enteros.

Existen muchas razones que han ocasionado la escasez y hasta la ausencia del vital líquido en todas las regiones del país, los fenómenos naturales, el mal estado de las redes de alcantarillado, la ausencia de estas y las pocas plantas de tratamiento de agua son solo unas de las muchas razones.

Los gobiernos han emitido leyes y desarrollado campañas de concientización y educación tratando de fomentar el buen uso del agua, pero estas medidas no son suficientes, por estas razones se hace necesario desarrollar sistemas que permitan dar un mejor uso a este recurso.

Los grandes avances tecnológicos han obligado a los sistemas a ser lo más autónomos posibles para evitar que el ser humano gaste tiempo y dinero interviniendo en dichos sistemas, por tal motivo es necesario automatizar el sistema de reciclaje, tratamiento y reutilización del agua que se propone en el presente proyecto.

Desarrollar sistemas automatizados que protejan el medio ambiente y que beneficien a la sociedad en general es responsabilidad de los ingenieros electrónicos, por esta razón, con los conocimientos de ingeniería, el grupo investigador trata de aportar una solución al problema del mal uso del agua en las casas, beneficiando no solo el medio ambiente sino también la economía de los hogares.

ABSTRACT

This research project can use the authors of the knowledge acquired during the Electronics Engineering degree and those seen in the field of instrumentation to monitor the level of tanks, valves and other electronic devices.

It propose a system to make better use of water in homes, recycling water which have been less contaminated and apply appropriate treatment to leave them in an optimal state of reuse in some household activities

Although the surface of planet earth is covered by 70% water, less than 1% of that is drinkable, which is why in some countries of the world, water scarcity is evident, although Colombia is one of the few countries privileged to have a lot of fresh, drinkable water, all its citizens have experienced at some point by the undesirable experience of not having drinking water to meet their basic needs, some for hours, others for days and not a few of weeks and months.

There are many reasons that have caused the shortage, or absence of the vital liquid in all regions of the country, natural phenomena, the poor state of sewerage networks, the absence of these and the few water treatment plants are just a of many reasons.

Governments have enacted laws and developed education and awareness campaigns trying to promote the proper use of water, but these measures are not enough, for these reasons it is necessary to develop systems to make better use of this resource.

Major technological advances have forced the systems to be as autonomous as possible to prevent human waste time and money taking part in

such systems, for this reason it is necessary to automate the recycling systems, water treatment and reuse is proposed in this project.

Develop automated systems that protect the environment and benefit society at large is responsible for electronic engineers, for this reason, engineering knowledge, the research seeks to provide a solution to the problem of misuse of water in the houses, benefiting not only the environment but also the economy of households.

1. INTRODUCCIÓN

El papel que juega el agua en la vida del ser humano es muy importante, la necesita para su alimentación, su aseo, su consumo, la agricultura y un sinnúmero de características que la hacen ser un recurso vital para la preservación de las especies dentro del planeta.

En la actualidad; el continuo cambio de la naturaleza transmisible en los fenómenos de cambio climático como es el “niño” que trae como consecuencia un desabastecimiento del recurso hídrico pone de manifiesto una seria problemática. Por otro lado; se presenta un crecimiento en la población mundial que está ocasionando una reducción considerable de los bosques naturales dado la necesidad de ampliar las áreas para ser ocupadas en viviendas o terrenos agrícolas perjudicando a un más esta situación.

Las aguas grises que resultan de ser usadas en lavanderos, baños, etc. se pueden emplear nuevamente en los baños y zonas de lavado. El agua lluvia al igual que las grises pueden emplearse en el uso doméstico llegando esta última a sustituir el agua potable.

De allí se expondrá en el presente trabajo un planteamiento de solución a l problema del agua; partiendo de la reutilización de las aguas lluvias y grises, las cuales se presentan con niveles de contaminación menor lo que permite a través de procesos de filtración reducir estos niveles para su posterior uso.

Esta alternativa es económica parte del diseño y creación de un prototipo automatizado para ser implementado en el sistema de reciclaje de aguas grises dentro de una vivienda.

2. MARCO TEORICO

2.1. Antecedentes

El agua se considera un recurso natural de vital importancia desde siempre para todo ser vivo, por esta razón, el ser humano siempre ha tratado de ubicar sus ciudades cerca de este preciado líquido. En la antigüedad cuando las ciudades crecían demasiado, se implementaban pozos para depositar agua y poderla utilizar posteriormente. Poco a poco el mundo científico descubrió que diversas enfermedades que padecían las comunidades se debían principalmente era a la contaminación del agua. A raíz de esto se empezaron a desarrollar formas de descontaminar el agua antes de beberla.

El ser humano ha desarrollado grandes plantas de tratamiento para el agua dulce, pero solo hasta hace pocos años, luego de observar la escasez de este líquido es que ha nacido la necesidad de tratar las aguas residuales y reutilizarlas.

Son muchas las personas y gobiernos que trabajan en proyectos de plantas de tratamiento de aguas residuales, incluyendo las grandes industrias como la petrolera. Hoy en día se habla de las casas sostenibles o auto sostenibles que buscan satisfacer las necesidades básicas de sus habitantes, utilizando la menor cantidad de recursos posibles. Dentro de los diversos sistemas que poseen estas casas se encuentran aquellos que reutilizan el agua y aprovechan las aguas lluvias.

Los países europeos se han interesado demasiado en los sistemas de reutilización del agua, debido a la escasez del líquido en sus países. España ha desarrollado leyes que obligan, desde el año 2005, a los constructores de viviendas a implementar sistemas de recolección de aguas lluvias en sus nuevas edificaciones.

En Colombia es poca la investigación que se ha llevado a cabo acerca de los sistemas de recolección de aguas lluvias y reutilización de aguas residuales. Quizás una de las razones sea la información que se ha dado del supuesto equivocado de que el territorio colombiano posee una enorme riqueza natural prácticamente inagotable. Es así como poco a poco el gobierno trata de incentivar y promover los proyectos que buscan proteger el medio ambiente.

En algunos proyectos de vivienda de interés social en Colombia se están implementando sistemas de energía alternativa como es el caso las del departamento de la Guajira, los cuales que usan energía eólica, pero solo hasta esta segunda mitad del año el gobierno ha expresado la necesidad de que en estas casas se implementen sistemas de recolección de aguas lluvias y reutilización de las aguas residuales.

Marco Legal

Según la Ley 373 de 1997; en su Art. 5. Reúso Obligatorio del Agua. Las aguas utilizadas, sean éstas de origen superficial, subterráneo o lluvias, en cualquier actividad que genere afluentes líquidos, deberán ser reutilizadas en actividades primarias y secundarias cuando el proyecto técnico y económico así lo ameriten y aconsejen según el análisis socio económico y las normas de calidad ambiental. El Ministerio del Medio Ambiente y el Ministerio de Desarrollo Económico reglamentarán en un plazo máximo de (6) seis meses, contados a partir de la vigencia de la presente ley, los casos y los tipos de proyectos en los que se deberá reutilizar el agua[1].

[1]Consulta de internet. Disponible en: https://www.minambiente.gov.co/images/normativa/leyes/1997/ley_0373_1997.pdf Fecha de Consulta: 14 de Abril de 2016

3. MARCO CONCEPTUAL.

3.1. Filtros de Carbón Activado:

Son sistemas de purificación de agua comunes en casas y edificios y se utilizan para filtrar contaminantes como el cloro, disolventes orgánicos, herbicidas, pesticidas y radón del agua. Los filtros los utilizan con frecuencia personas que son conscientes de la salud y que desean evitar que las partículas granuladas u olores desagradables y sabores queden en el agua. Se debe aclarar sin embargo, que los filtros de carbón activado no quitan las bacterias, virus u hongos, ni esporas de hongos del agua[2].

Existen dos tipos de filtros de carbón activo doméstico. Lo filtros de carbón activado granulados y los filtros de carbón bloqueador en polvo.
En nuestro caso utilizamos el filtro de carbón activado granulado. Este filtro está compuesto por partículas de carbono que según la marca o modelo varían en su tamaño. El carbón activado ha sido cargado con electricidad de manera que este pueda atraer compuestos que contienen carbono. Este a su vez atrae solo a compuestos orgánicos y agentes contaminantes, haciendo que solo pase el agua deteniendo estos compuestos.

[2] Consulta por internet. Disponible en: http://www.ehowenespanol.com/funciona-filtro-agua-carbon-activado-info_207845/ Fecha de consulta: 22 de marzo de 2018

FIGURA 1. Filtro de Carbón activado y de membrana

Foto autoría propia

3.2. Electroválvulas

Es una válvula electromecánica, diseñada para controlar el flujo de un fluido a través de un conducto como puede ser una tubería. La válvula está controlada por una corriente eléctrica a través de una bobina solenoidal. No se debe confundir la electroválvula con válvulas motorizadas, que son aquellas en las que un motor acciona el cuerpo de la válvula[3].

[4]Una electroválvula está compuesta por dos partes fundamentalmente: el solenoide y la válvula.

El Solenoide hace la conversión de la energía eléctrica en mecánica mediante magnetismo para poder actuar la válvula.

En algunas electroválvulas el solenoide actúa directamente sobre la válvula proporcionando toda la energía necesaria para su movimiento. Es corriente que la

[3] Consulta por internet. Disponible en: http://lexicoon.org/es/electrovalvula Fecha de consulta: 22 de marzo de 2018

[4] Consulta por internet. Disponible en https://es.wikipedia.org/wiki/Electrov%C3%A1lvula Fecha de consulta: 22 de marzo de 2018

válvula se mantenga cerrada por la acción de un muelle y que el solenoide la abra venciendo la fuerza del muelle. Esto quiere decir que el solenoide debe estar activado y consumiendo energía mientras la válvula deba estar abierta. La electroválvula que utilizamos fue la ebchq 2w-160-15.

FIGURA 2 Y FIGURA 3. Electroválvula

Foto tomada de https://articulo.mercadolibre.com.co/MCO-452218289-electrovalvula-110vac-12-2w-160-15-_JM

[5]Válvula Solenoide de referencia 2W160-15, permitiendo una apertura de 16mm, compatible con rosca 1/2", se encuentra normalmente cerrada. El embobinado que tiene esta válvula es de alambre de cobre con módulo de refrigeración revestida de aluminio. Esta puede ser instalada en cualquier ángulo, pese a ello, se recomienda posicionarla en dirección horizontal.

Modelo	2W160-15
Medio de funcionamiento	Aire, agua, aceite, gas
Voltaje de operación	110VAC
Modo de operación	Acción directa

[5] Consulta por internet. Disponible en: https://www.vistronica.com/valvulas/electrovalvula-110vac-12-2w-160-15-detail.html. Fecha de consulta: 22 de marzo de 2018

Tipo de accionamiento	Normalmente cerrada
Tamaño de puerto	G1/2
Operación de viscosidad de fluido	Por debajo de 20 CST
Operación de presión	· Aire: 0Mpa~1.0Mpa · Agua: 0Mpa~0.7Mpa · Aceite: 0.9Mpa
Resistencia máxima de presión	1.0Mpa
Temperatura de operación	-5°C~+120°C
Rango de voltaje	±10%
Material de la estructura	Cobre
Material de sello de aceite	NBR o VITON

TABLA 1

3.3. Electrobombas

Las electrobombas son máquinas que transforman energía aplicándola para mover el agua normalmente de manera ascendente. Estás son accionadas eléctricamente. En nuestro sistema utilizamos dos electrobombas de referencia PW-60980C.

Estas electrobombas funcionan a 115 V.

FIGURA 4. Electrobomba

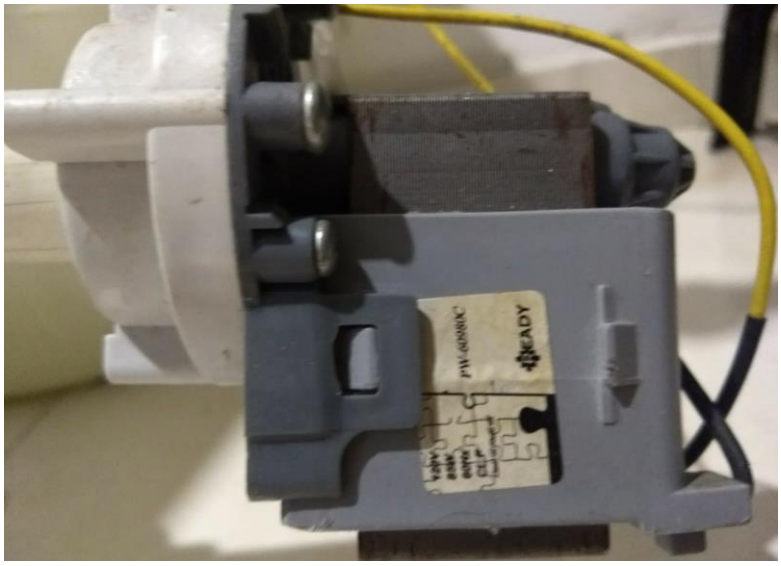


Foto autoría propia

3.4. Microcontrolador ESP32

EL ESP32 es un microcontrolador de 3.3 voltios, que contiene un módulo Wi-Fi y Bluetooth de 2.4 GHz, creado por la empresa ESPRESSIF, principalmente para aplicaciones de Internet de las cosas (IoT). Este microcontrolador cuenta con una antena de circuito impreso y cumple con los protocolos 802.11 b/ g/ n del IEEE (Institute of Electrical and Electronics Engineers).

La empresa ESPRESSIF dispone de diferentes versiones del ESP32 y en nuestro caso hemos usado el ESP32-WROOM-32, el cual cuenta con una antena de circuito impreso. Otras características principales son:

- Procesador Tensilica Xtensa de doble núcleo a 32 bits LX6
- ROM de 448kB
- SRAM de 520 kB
- Temporizadores de 64 bits con watchdog
- 34 GPIOs programables (pines de entrada y salida)
- Conversor análogo/digital de 12 bits

- 2 conversores digital/análogo de 8 bits
- 10 GPIOs especializados para sensores táctiles
- 4 SPI (Serial Peripheral Interface)
- Interfaz Ethernet MAC con DMA dedicado y soporte IEEE 1588
- 3 UART (Universal Asynchronous Receiver-Transmitter)
- Módulo PWM

El diagrama de bloques del ESP32-WROOM-32 suministrado por el fabricante es el siguiente

FIGURA 5. Diagrama de bloque del ESP32

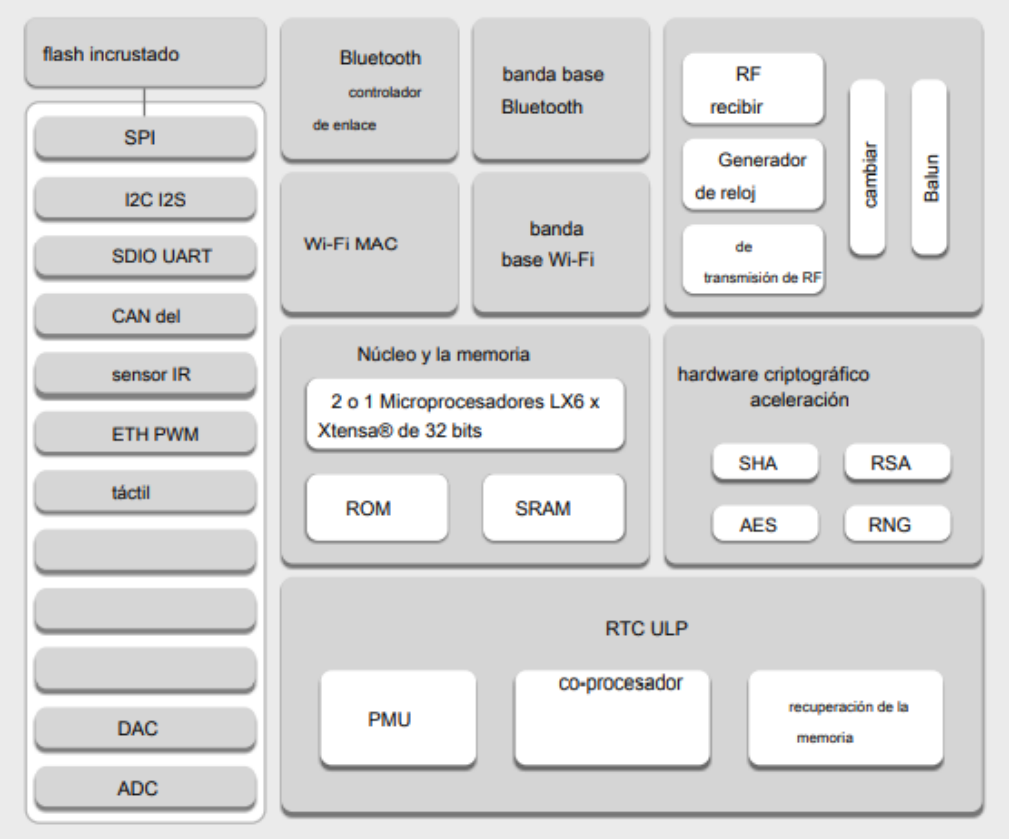


Foto tomada de

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

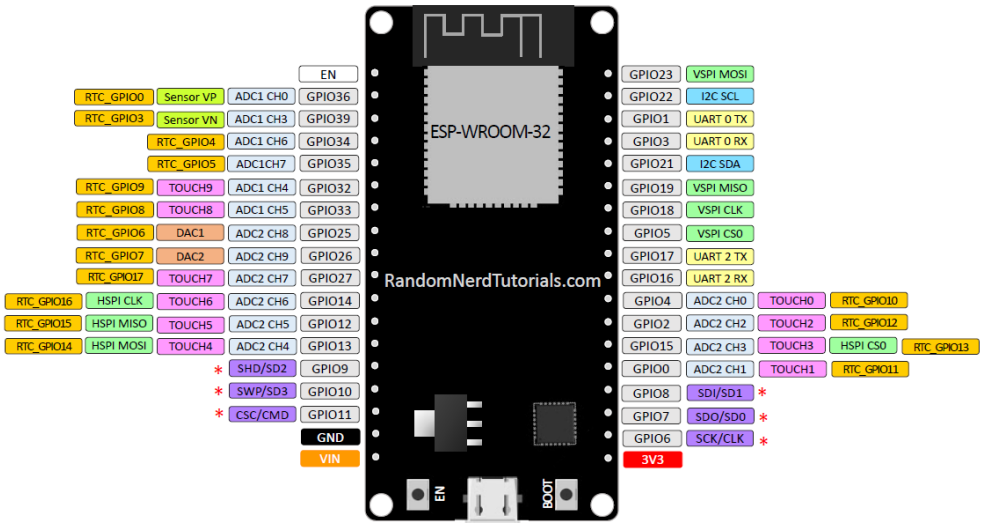
Todas las características del ESP32 se pueden consultar en la hoja de datos del fabricante.

Para nuestro proyecto hemos usado el ESP32-DevKitC de la empresa DOIT, la cual está autorizada por ESPRESSIF, para su fabricación. El ESP32-DevKitC

cuenta con los siguientes componentes que facilitan el uso del ESP32-WROOM-32.

- ESP32-WROOM-32
- Interfaz USB-UART con puerto micro-USB que facilita la carga del código fuente desde la computadora
- Led de encendido que indica la existencia de una fuente de alimentación de voltaje en el pin VIN o por el puerto micro-USB
- Botón de booteo, mediante el cual se puede actualizar sistema operativo del ESP32. El sistema operativo es FreeRTOS
- Botón de reinicio
- Memoria flash de 4MB

FIGURA 5.1 diagrama de pines del ESP32-DevKitC



Tomada de https://raw.githubusercontent.com/RuiSantosdotme/ESP32-Course/master/img/ESP32-DOIT-DEVKIT-V1-Board-Pinout-36-GPIOs.png

3.5. Interruptores de nivel

Un interruptor de nivel es un dispositivo que, instalado sobre un tanque u otro recipiente en que hay almacenamiento de sólidos o líquidos, permite discriminar si la altura o nivel que el material o elemento almacenado alcanza o

excede un nivel predeterminado. Al producirse dicha condición, este dispositivo cambia de estado y genera una acción que evita que el nivel siga subiendo[6].

FIGURA 6. Interruptor de Nivel

Tomada de http://www.directindustry.es/prod/standex-meder-electronics-gmbh/product-23046-777259.html

[7]La serie LS03 sensores de nivel de líquido es un sensor compacto utilizado en la detección de un solo nivel. Este sensor ha sido incorporado en un flotador magnético y tiene que ser montado horizontalmente para obtener mejores resultados. Se montaje se efectúa como lo muestra la imagen arriba para una configuración normalmente abierta o también se puede girara 180° para operaciones normalmente cerrado.

Este sensor viene con un cable de PVC con una sección transversal de 0,14mm2 y una longitud de 500mm. El cable puede ser modificado o cambiado a petición del comprador. H

Hay dos versiones disponibles

- PP (Polipropileno): aplicaciones de agua y ácidos diluidos.

[6] Consulta por internet. Disponible en: http://www.ecured.cu/Interruptor_de_nivel Fecha de consulta: 30 de marzo de 2018

[7] Consulta por Internet. Disponible en http://www.directindustry.es/prod/standex-meder-electronics-gmbh/product-23046-777259.html Fecha de consulta 30 de marzo de 2018

- PA (Poliamida): líquidos de frenos, gasolina, aceite

3.6. Lámparas Ultravioleta

Para explicar la utilidad de las lámparas ultravioleta debemos entender la importancia y el principio de funcionamiento de estas mismas o de los rayos UV.
Los rayos UV es una forma de energía radiante que proviene del sol[8].
Las diversas formas de radiación se clasifican según la longitud de onda medida en nanómetros (nm), que equivale a un millonésimo de milímetro. Cuanto más corta sea la longitud de onda, mayor energía tendrá la radiación.
Existen tres categorías de radiación UV:

- ✓ UV-A, entre 320 y 400 nm
- ✓ UV-B, entre 280 y 320 nm
- ✓ UV-C, entre 200 y 280 nm

La radiación UV-A es la menos nociva y la que llega en mayor cantidad a la Tierra. Casi todos los rayos UV-A pasan a través de la capa de ozono.
La radiación UV-B puede ser muy nociva. La capa de ozono absorbe la mayor parte de los rayos UV-B provenientes del sol. Sin embargo, el actual deterioro de la capa aumenta la amenaza de este tipo de radiación.

La purificación del agua mediante rayos ultravioleta es un método rápido y único para desinfectarla sin utilizar productos químicos ni calor. Los Purificadores de agua utilizan lámparas germicidas de ultravioleta que producen radiaciones de pequeñas ondas que son letales para las bacterias, virus y otros microorganismos presentes en el Agua común. [9]Cuando las bacterias, los virus y los protozoos se exponen a las longitudes de onda germicidas de la luz UV, se vuelven incapaces

[8] Consulta por internet. Disponible en: http://blog.ciencias-medicas.com/archives/1423 Fecha de consulta 23 de marzo de 2018

[9] Consultado por internet. Disponible en http://www.trojanuv.com/es/uv-basics. Fecha de consulta 28 de marzo de 2018

de reproducirse e infectar. Se ha demostrado que la luz UV es eficaz frente a microorganismos patógenos, como los causantes del cólera, la polio, la fiebre tifoidea, la hepatitis y otras enfermedades bacterianas, víricas y parasitarias.

Estos rayos son más cortos en longitud de onda que los rayos más cortos de ultravioleta, que penetran la atmósfera de la Tierra, desde el Sol[10].

Las lámparas germicidas son ampliamente utilizadas para desinfectar el aire en hospitales, clínicas veterinarias y laboratorios para proteger al personal y prevenir infecciones. Los distribuidores farmacéuticos, manipuladores de alimentos y embotelladores utilizan lámparas germicidas para prevenir la contaminación del producto y en otros casos para la esterilización fría de los productos acabados.

FIGURA 7. Lámpara Ultravioleta

Tomada de http://www.sylvania-lighting.com/product/es-es/products/0000517

Descripción del producto	
Código IPC	**517**
Nombre del producto	**G15W T8**
Tecnología	**Fluorescente**

[10] Consultado por internet. Disponible en: http://www.aguamarket.com/productos/productos.asp?producto=1671 Fecha de consulta: 28 de marzo de 2018

Potencia (clasificada) (W)	**15.00**
Forma de la lámpara	**Lineal**
Tipo	**T8**
Casquillo/Base	**G13**
Acabado de la lámpara	**Clara**
Clasificación de dispositivos	**Open**
Aplicación general	**Hospitality, Retail, Residential & Consumer**
Certificaciones	**0**
Clase ETIM	**EC002962**
Color de la luz	**UV**
Potencia (nominal) (W)	**15.00**
Tensión (V)	**55**
Regulable	**Sí**
Promedio de vida útil (h) Ta 25° C	**8000**
Código EAN	**5,41029E+12**

TABLA 2

3.7. Balastro Electrónico

Un Balastro Electrónico es un dispositivo de control Electrónico para Lámparas fluorescentes compactas. El Balastro garantiza una larga vida útil a la lámpara y Evita que se vea afectada por un encendido/apagado frecuente. La referencia del balastro que utilizamos en el sistema es: Osram qtp 2x32

- Multivoltaje 120 - 277 volts.
- Universal para tubos fluorescentes T-8 incluido curvalum
- 2x32 watts
  2x25 watts
  2x17 watts

- Encendido instantáneo.

FIGURA 8. Balastro Electrónico

Tomada de https://articulo.mercadolibre.com.ve/MLV-464488567-balastro-electronico-3x32-osram-_JM

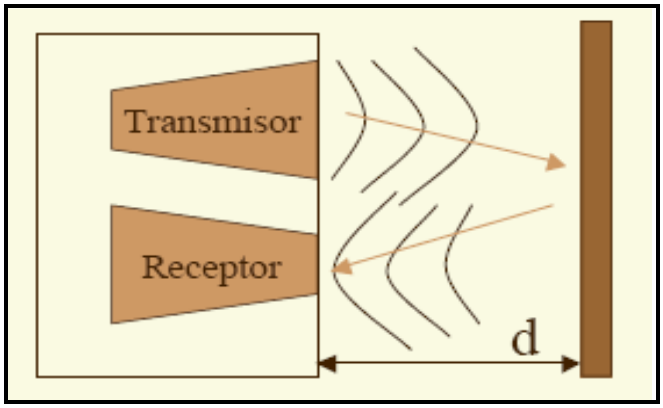
3.8. Sensor de ultrasonidos

Los ultrasonidos son antes que nada sonido, exactamente igual que los que oímos normalmente, salvo que tienen una frecuencia mayor que la máxima audible por el oído humano. Ésta comienza desde unos 16 Hz y tiene un límite superior de aproximadamente 20 KHz, mientras que nosotros vamos a utilizar sonido con una frecuencia de 40 KHz. A este tipo de sonidos es a lo que llamamos Ultrasonidos[11].

El funcionamiento básico de los ultrasonidos como medidores de distancia se muestra de una manera muy clara en el siguiente esquema, donde se tiene un receptor que emite un pulso de ultrasonido que rebota sobre un determinado objeto y la reflexión de ese pulso es detectada por un receptor de ultrasonidos:

FIGURA 9. Sensor por Ultrasonidos.

[11] Consulta por internet. Disponible en: http://www.todopic.com.ar/foros/index.php?topic=10892.10;wap2 Fecha de consulta: 30 de marzo de 2016



Tomada de http://picmania.garcia-cuervo.net/recursos/redpictutorials/sensores/sensores_de_distancias_con_ultrasonidos.pdf

La mayoría de los sensores de ultrasonido se basan en la emisión de un pulso de ultrasonido cuyo lóbulo, o campo de acción, es de forma cónica. Midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda sonora, mediante la fórmula:

$$d = \frac{1}{2} V \cdot t$$

Donde V es la velocidad del sonido en el aire y t es el tiempo transcurrido entre la emisión y recepción del pulso. En la formula tomamos el ½ porque es el recorrido que se toma al golpear la señal con el obstáculo.

3.8.1. Sensor SFR05

El SFR05 es la actualización del SFR04 con el que es compatible, pero además añadiendo nuevas funciones y características. En el modo estándar, el SRF05 se comporta igual que el SRF04 con la diferencia de que el rango de trabajo se ha aumentado de 3 a 4 metros. Esto significa que todo el software que funciona con el SRF04, funciona con el SRF05. Por otro lado, el SRF05 cuenta

con un nuevo modo de trabajo que emplea un solo pin para controlar el sensor y hacer la lectura de la medida[12].

FIGURA 10. SRF05

Tomada de http://www.superrobotica.com/s320111.htm

El SRF05 lo que se hace es mandar un impulso para iniciar la lectura y luego poner el pin en modo entrada. Después basta con leer la longitud del pulso devuelto por el sensor, que es proporcional a la distancia medida por el sensor. El SRF05 es mecánicamente igual al SRF04, por lo que puede ser un sustituto de este. El sensor SRF05 incluye un breve retardo después del pulso de eco para dar a los controladores más lentos como Basic Stamp y Picaxe el tiempo necesario para ejecutar sus pulsos en los comandos. El sensor SRF05 tiene dos modos de funcionamiento, según se realicen las conexiones[13].

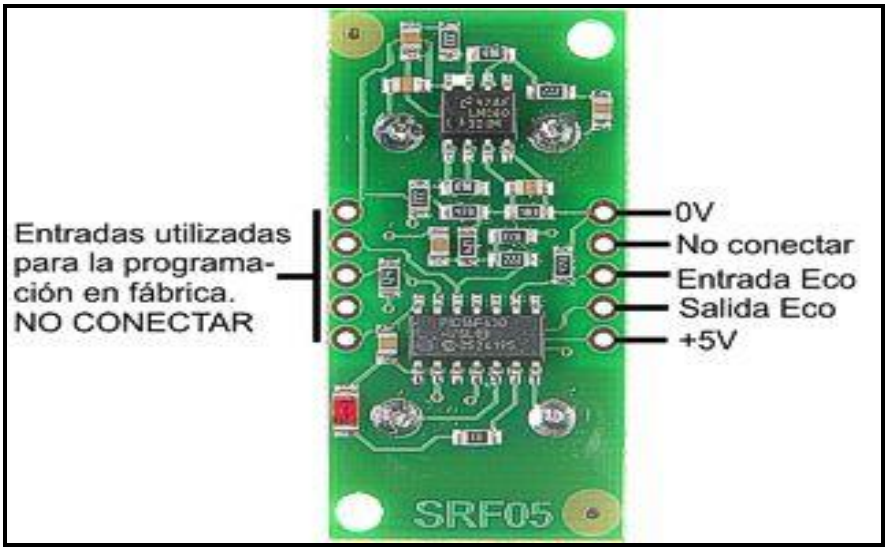
Modo 1 - Compatible con SRF04 - Señal de activación y eco independientes
Este modo utiliza pines independientes para la señal de inicio de la medición y para retorno del eco, siendo el modo más sencillo de utilizar. Todos los ejemplos de códigos para el sensor SRF04 funcionarán para SRF05 en este modo. Para utilizar este modo, simplemente deberá dejar sin conectar el pin de modo - el SRF05 integra una resistencia pull-up en este pin[14].

[12] Consulta por internet. Disponible en: http://www.robot-electronics.co.uk/htm/srf05tech.htm Fecha de consulta: 02 de abril de 2016

[13] Ibíd.

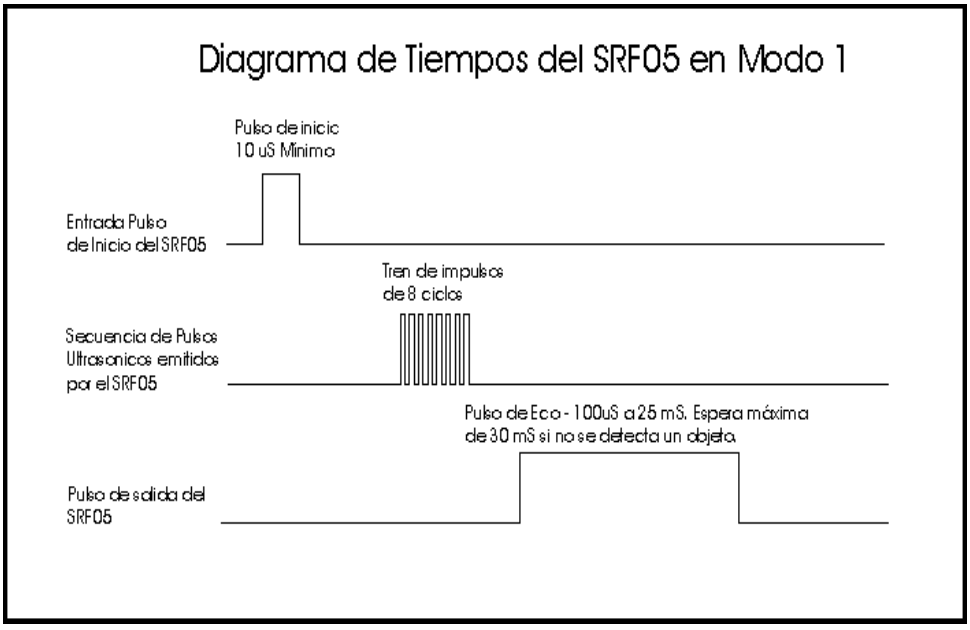
[14] Ibíd.

FIGURA 11. Diagrama de conexión del SFR05 en modo 1.



Tomada de http://www.superrobotica.com/s320111.htm

FIGURA 12. Diagrama de tiempos del SFR05 en modo 1.



Tomada de http://www.superrobotica.com/s320111.htm

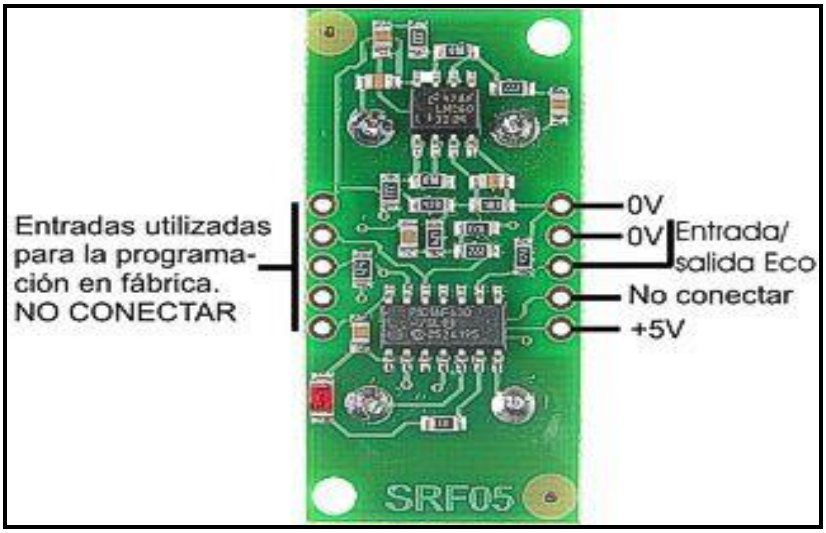
Modo 2 - Pin único para la señal de activación y eco

Este modo utiliza un único pin para las señales de activación y eco, y está diseñado para reducir el número de pines en los microcontroladores. Para utilizar este modo, conecte el pin de modo al pin de tierra de 0v. La señal de eco

aparecerá en el mismo pin que la señal de activación. El SRF05 no elevará el nivel lógico de la línea del eco hasta 700uS después del final de la señal de activación[15].

Dispone de ese tiempo para cambiar el pin del disparador y convertirlo en una entrada para preparar el código de medición de pulsos. El comando PULSIN integrado en la mayor parte de los controladores del mercado lo hace automáticamente.

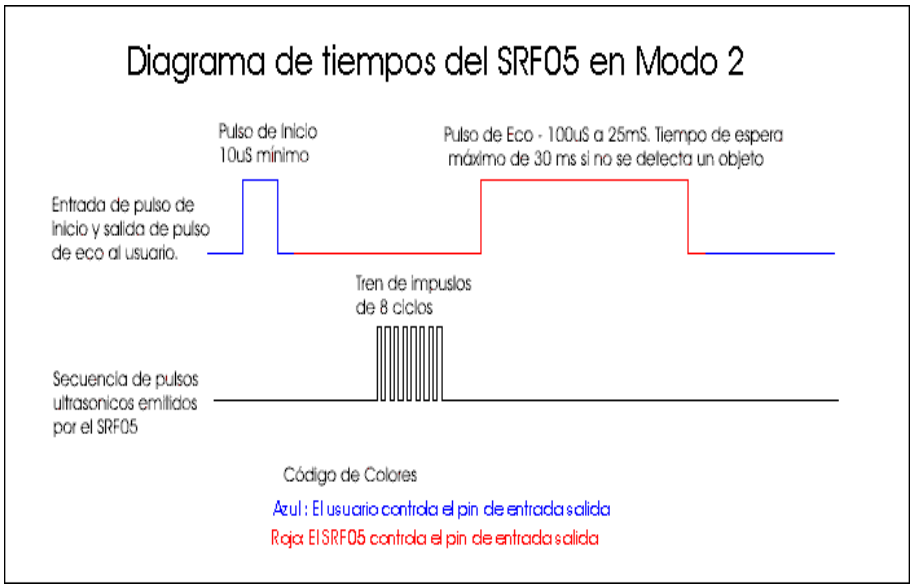
FIGURA 13. Diagrama de conexión del SFR05 en modo 2.



Tomada de http://www.superrobotica.com/s320111.htm

[15] Ibíd.

FIGURA 14. Diagrama de tiempos del SFR05 en modo 2.



Tomada de http://www.superrobotica.com/s320111.htm

El SRF05 puede activarse cada 50mS, o 20 veces por segundo. Debería esperar 50ms antes de la siguiente activación, incluso si el SRF05 detecta un objeto cerca y el pulso del eco es más corto. De esta manera se asegura que el "bip" ultrasónico ha desaparecido completamente y no provocará un falso eco en la siguiente medición de distancia.

4. DESCRIPCION DEL SISTEMA

4.1. Montaje y explicación del sistema

El sistema está construido con un objetivo principal que es el de poder reutilizar el agua lluvia y las aguas grises de la vivienda para otras actividades de esta misma.

En nuestro caso tomamos el agua que vamos tratar de tres puntos específicos que son el agua de la lavadora, el agua del lavamanos y las aguas lluvias. Estos puntos se escogieron porque son aguas que no contienen altos

contenidos de grasa ya que en nuestro sistema no está contemplado ningún filtro atrapa grasa.

El sistema está compuesto por 4 tanques. En el tanque #0 se introduce el agua sucia recolectada de los puntos anteriormente mencionados pasando primero por un filtro de malla que retiene las partículas más gruesas y solidas que tenga el agua. Dentro del tanque tenemos un filtro de gravilla que construimos de la siguiente manera: de abajo hacia arriba del tanque se fueron colocando capas gruesas de piedras empezando en la parte más baja por piedras más grandes y gruesas.

Luego colocamos otra capa de piedras medianas, seguidas entonces por una capa de piedras más pequeñas que la anterior y luego colocamos una capa de arena finalizando con una capa de piedras de las más grandes nuevamente para evitar que al introducir el agua al tanque la capa de arena se disperse.

FIGURA 15. Capa inferior del filtro de gravilla.

Foto de autoría propia

FIGURA 16. Capa intermedia del filtro de gravilla.

Foto de autoría propia

FIGURA 17. Capa superior de filtro de gravilla.

Foto de autoría propia

Podemos observar que el tanque que llamamos tanque #0 tiene un interruptor de nivel y un orificio de salida.

FIGURA 18. Tanque 0 del sistema.

Foto de autoría propia

FIGURA 19. Tanque 1 y 2 del sistema.

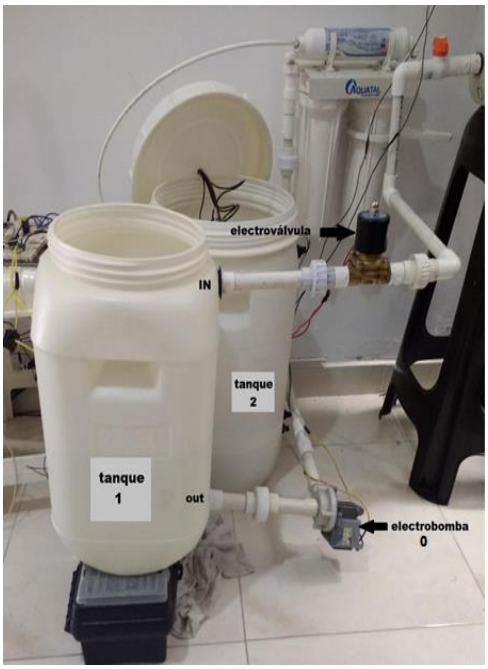


Foto de autoría propia

En el tanque #1 tenemos un orifico de entrada (in) y uno de salida (out) al igual que dos interruptores de nivel, uno colocado en la parte inferior del tanque

que me indica cuando el tanque esta vacio y el otro colocado en la parte superior del tanque que me indica cuando el tanque esta lleno.

Entre la salida del tanque #0 y la entrada del tanque #1 tenemos una electroválvula que se activa o desactiva dependiendo del estado de los tanques #0 y #1 . La funcion del interruptor de nivel que está en el tanque #0 y de la electroválvula es garantizar que el nivel de agua en el tanque #0 nunca esté por debajo del filtro de gravilla ya que la vida util de este filtro depende de que éste mantega su humedad. El tanque #1 tiene a su salida una electrobomba #0 que es la que nos ayuda a conducir el agua hasta el filtro de carbon activado, dado que este filtro requiere de una presion de 1,5 a 8 bar.

En el tanque # 2 tenemos dos las lamparas UV manejadas por un balastro de referencia OSRAM QTP 2X32T8 el cual va conectado directamente a nuestro circuito.

FIGURA 20. Vista iterior del tanque 2.

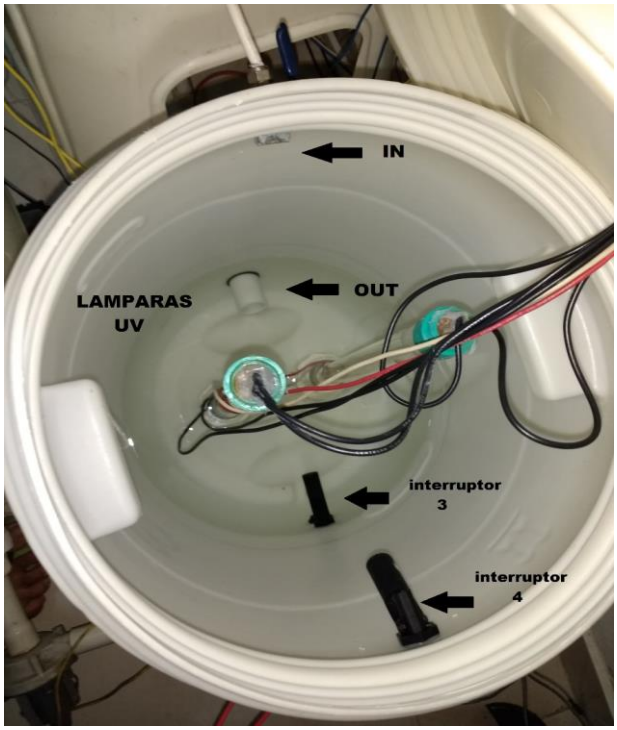


Foto de autoria propia

Entre el tanque #1 y el tanque #2 hemos colocado un filtro de carbón activado. Este filtro cumple la misma función del filtro de gravilla, pero de una manera más especializada.

Para que el agua pase del tanque #1 al filtro de carbón activado la electrobomba 0 debe encenderse, y para que esto ocurra deben cumplirse estas condiciones:
Que el nivel del agua en el tanque #1 se encuentre por encima del interruptor de nivel 2. La otra condición es que el tanque #2 tenga agua sucia o que esté vacío.

Y la última es que las lamparas UV no estén encendidas.
la electrobomba 0 se enciende y lleva el agua hasta el filtro de carbón activado. Y de la salida de este filtro obtenemos agua muy limpia, de baja turbiedad y libre de partículas sólidas.

El agua que sale del filtro entra por el orificio de entrada (IN) del tanque #3 donde encontramos las lamparas UV que se encienden cuando el tanque está completamente lleno por agua sucia. La duración del encendido de éstas lo programamos a 30 segundos que es un tiempo más que suficiente para asegurarnos que la eliminación de microorganismos y bacterias se completen correctamente.

FIGURA 21. Tanque 3 del Sistema

Foto de autoría propia

En el tanque #3 colocamos un sensor de ultrasonido de referencia SFR 05 ubicado en el centro del atapa del tanque, el cual mide la distancia hasta el nivel del agua de éste. Esta distancia es enviada a Firebase mediante el ESP32 del circuito. A la entrada del tanque #3 colocamos una válvula de antirretorno de agua.

FIGURA 22. Sensor SFR05 en el tanque 3.

Foto de autoría propia

La electrobomba 1 se enciende cuando exista agua limpia en el tanque #2 y la distancia del SRFO5 al nivel del agua en el tanque #2 sea mayor o igual a 20 cm y se apagará cuando no exista agua limpia en el tanque #2 o la distancia del sensor sea menor o igual a 10 cm.

Ya contamos con agua tratada, clara, limpia y libre de microorganismos que se puede extraer fácilmente abriendo una llave cada vez que lo necesitemos a la salida del tanque #3 para ser reutilizada en labores del hogar como lo son riego de jardines, descarga de sanitarios, lavado de ropa, lavado del carro etc.

FIGURA 23. Sistema Completo

Foto de autoría propia

4.2. Desarrollo y explicación del circuito.

4.2.1. Fuente de poder de 5V

El siguiente circuito es la fuente de poder de 5 voltios que alimenta el ESP32 y el sensor SRF05

FIGURA 24. Diagrama de poder de 5V

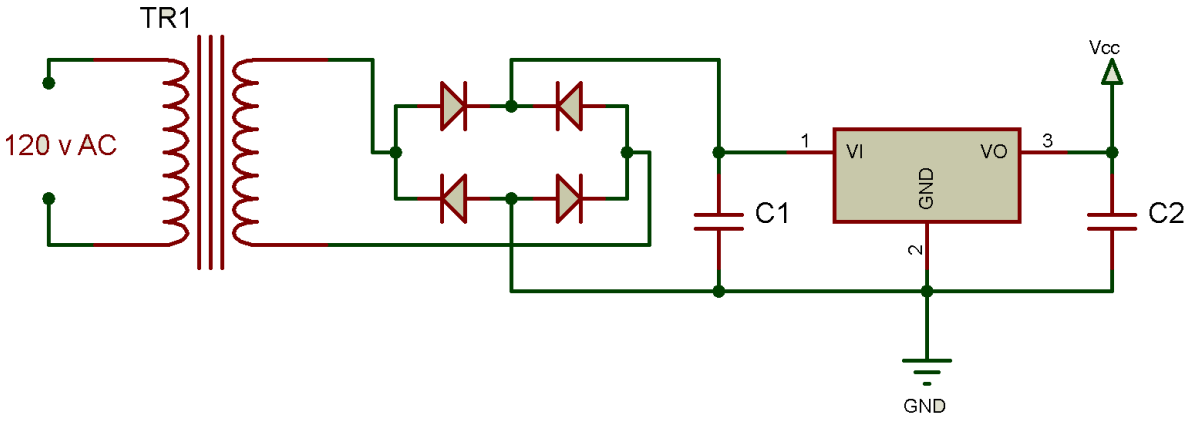


Imagen de autoria propia

El circuito está conformado por el transformador TR1, el cual es alimentado en el devanado primario por una señal AC (corriente alterna) domiciliaria de 110 voltios a 60 Hz y en el devanado secundario se obtiene una señal AC de 9 voltios.

La señal AC de 9 voltios es la entrada del puente rectificador de diodos de onda completa, se usó el 3N256, el cual es un puente rectificador de onda completa encapsulado de 2 amperios que soporta hasta 280 voltios.

En la salida positiva del puente rectificador se obtiene una señal CD (corriente directa) pulsante, la cual ingresa al capacitor C1, que funciona como filtro para reducir el rizado de la señal CD pulsante y así obtener una señal CD no pulsante

Calculo del capacitor C1:

[16]Haciendo uso de la ecuación

$$V_{riz} = \frac{I}{fC} \qquad (1)$$

y los lineamientos de diseño propuestos en este mismo libro, en el que se indica que el rizado pico a pico de la señal CD debe ser aproximadamente de un 10% del voltaje pico

[17]Dado que nuestro voltaje rms (root mean square) o eficaz es de 9 voltios y con la ecuación

$$V_p = V_{rms} * \sqrt[3]{2}$$

se obtiene un voltaje pico

$$V_p = 9 * \sqrt[3]{2}$$

$$V_p = 12.72\ v$$

[16] Consulta en MALVINO, ALBERT PAUL. (1989) Principios de Electrónica. Pag 72, ec 3-11

[17] Ibíd.

con lo cual obtenemos un voltaje de rizado

$$V_{riz} = 12.72 * 10\%$$

$$V_{riz} = 1.27\ v$$

Como usamos un rectificador de onda completa, nuestra señal de 9 voltios tiene una frecuencia de 120 Hz.

Esta fuente de voltaje alimentará el ESP32, el cual en su hoja de datos especifica que se debe suministrar una corriente mínima de 500 mA, por lo cual realizaremos los cálculos teniendo en cuenta una corriente de 700 mA para el ESP32.

La fuente de poder también alimentará al sensor de ultrasonido SRF05, el cual consume una corriente de 30 mA, esta corriente se halla realizando mediciones con un amperímetro, ya que el fabricante en su hoja de datos no suministra este parámetro.

Teniendo en cuenta los componentes que alimentará la fuente de poder, tenemos una corriente de 730 mA

De la ecuación (1) tenemos:

$$C1 = \frac{I}{f * V_{riz}}$$

$$C1 = \frac{730\ mA}{120\ Hz * 1.27\ v}$$

$$C1 = 4790\ \mu F$$

Para nuestro proyecto utilizamos y capacitor de 4700μF.

Como regulador de voltaje se utilizó el L78S05CV, que en su salida entrega 5 voltios DC y opera a una corriente máxima de 2 amperios

El capacitor C2 es de 0.1 μF y es usado por especificaciones en la hoja de datos del L78S05CV.

4.2.2. Etapa de Control.

La etapa de control está basada, fundamentalmente en el ESP32-WROOM-32, el cual tiene grabado todo el código con la lógica de control. El ESP32 tiene como framework oficial para su programación el ESP-IDF, pero la empresa fabricante, ESPRESSIF, ha creado todo un proyecto con varias librerías para que se pueda usar el IDE (*Integrated Development Environment*) Arduino, el cual fue usado para nuestra programación.

Las instrucciones de instalación del proyecto para programar el ESP32 en el IDE Arduino en los sistemas operativos Windows, MAC OS, Debian, Fedora u OpenSUSE se encuentran en la cuenta oficial GITHUB de ESPRESSIF (https://github.com/espressif/arduino-esp32#installation-instructions)

4.2.3. Código fuente ESP32-WROOM-32: (ver ANEXO 1)

El ESP32-WROOM-32 en nuestro circuito tiene las siguientes conexiones:

FIGURA 25. Diagrama ESP32

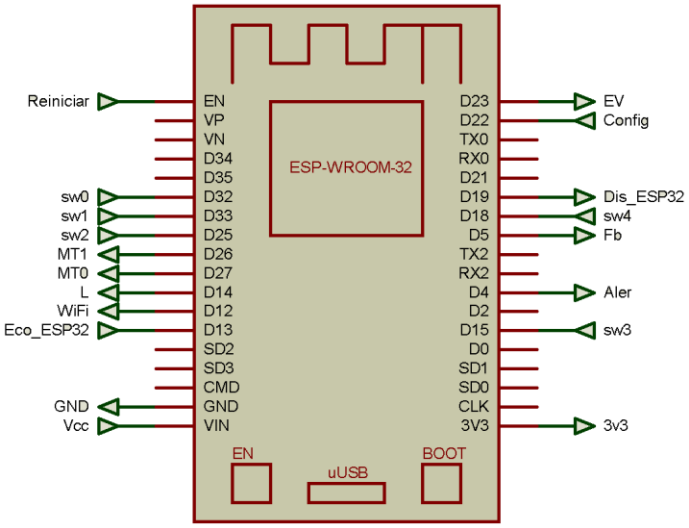


Imagen de autoria propia

- Pin EN: pulsador para reiniciar el sistema.
- Pin D22: pulsador para que el ESP32 entre en modo AP (Acces Point) y así poder ingresar las credenciales de red (nombre de red y contraseña) del modem al cual se debe conectar para tener acceso a internet. Cuando el ESP32 está en modo AP, emite una red, que en nuestro caso le hemos dado el nombre de “AQUA VIVA” con contraseña aquaviva01. Desde un SmartPhone, tableta o PC se puede conectar a esta red.

FIGURA 26. Señal de red ESP32

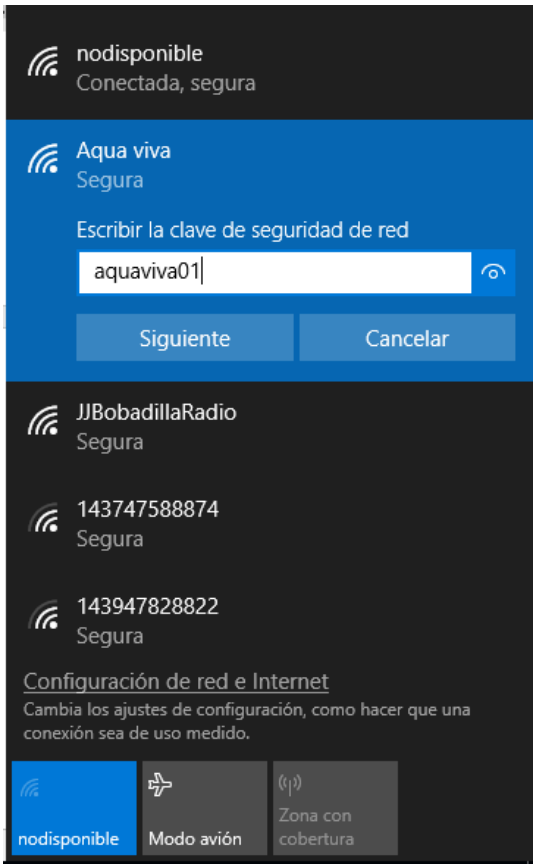


Imagen de autoría propia

En este modo (AP), en el ESP32 se ha configurado un servidor web con dirección IP (Internet Protocol) 192.168.0.1, en la que se encuentra una página web para poder ingresar, desde cualquier dispositivo como Smartphone, tableta o PC y mediante cualquier navegador web, las credenciales del modem al que se debe conectar el ESP32 para acceder a internet.

Al ingresar a esta dirección IP, se obtendrá la siguiente página:

FIGURA 27. Página Web1 del ESP32

Imagen de autoria propia

Al dar click en “GUARDAR”, el ESP32 almacena el nombre de red y contraseña en su memoria EEPROM para que el usuario no deba estar ingresando estos datos cada vez que va a encender el sistema

Luego de ingresar las credenciales del modem, el ESP32 responderá con la siguiente página en la que se ve el mensaje de que se ingresó correctamente las credenciales.

FIGURA 28. Página Web2 del ESP32

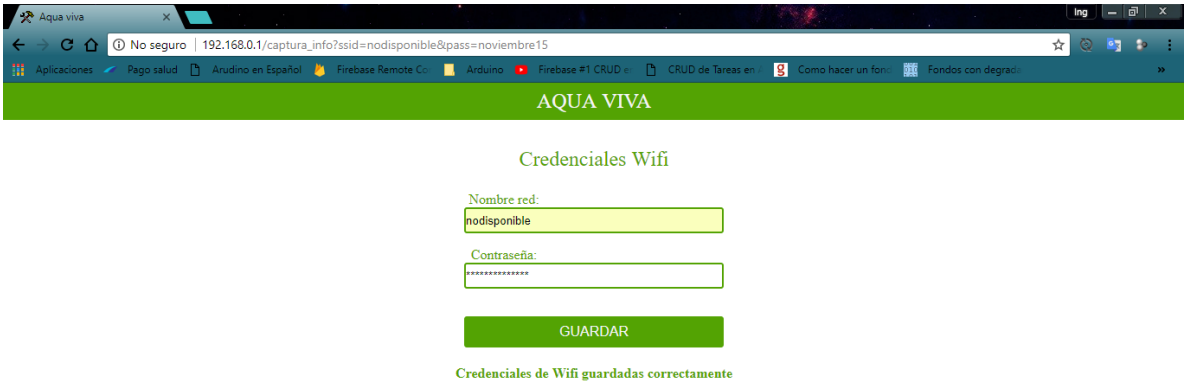


Imagen de autoria propia

La conexión de los pulsadores al ESP32 es:

FIGURA 29. Conexión de pulsadores al ESP32

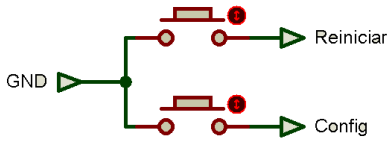


Imagen de autoria propia

- Pin D32: interruptor de nivel sw0 ubicado en el tanque T0.
- Pin D33: interruptor de nivel sw1 ubicado en la parte inferior del tanque T1
- Pin D25: interruptor de nivel sw2 ubicado en la parte superior del tanque T1
- Pin D15: interruptor de nivel sw3 ubicado en la parte inferior del tanque T2
- Pin D18: interruptor de nivel sw4 ubicado en la parte superior del tanque T2

FIGURA 30. Conexión de Interruptores al ESP32

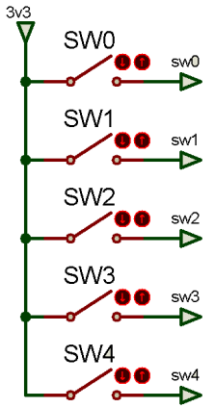
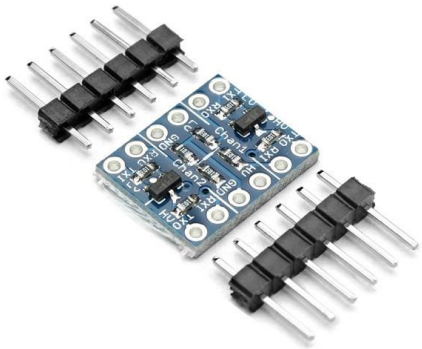


Imagen de autoria propia

- Pin D19: Mosfet BSS138 de canal N para convertir el nivel de 3.3v del ESP32 a 5v, para el envío del pulso de disparo al SRF05 con el cual iniciará una medición de la distancia del SRF05 al nivel del agua del tanque T3.
- Pin D13: divisor de tensión para convertir el nivel de 5v del SRF05 a 3.3v, para el envío del pulso de eco al ESP32, con el cual se realizará el cálculo de la distancia del SRF05 al nivel del agua del tanque T3

Para realizar el proceso de conversión de niveles lógicos se utilizó la siguiente placa integrada:

FIGURA 31. Convertidor de nivel Lógico



Tomada de https://electronilab.co/tienda/conversor-niveles-2-canales-bidireccional-5v-33v/

La conexión de la placa integrada (Mosfet y el divisor de tensión) al ESP32 es:

FIGURA 32. Conexión del convertidor de nivel Lógico al ESP32

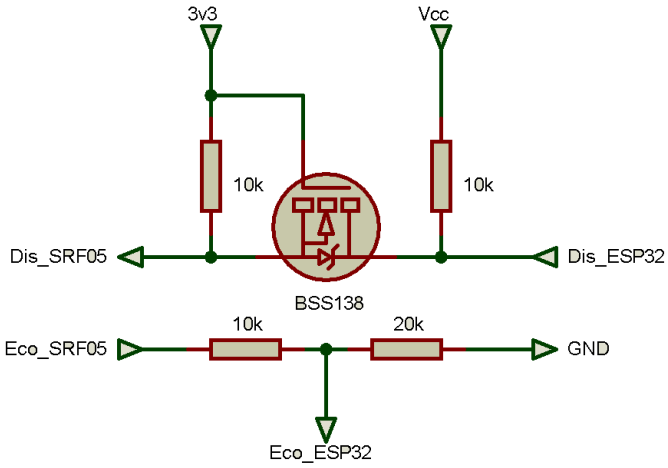


Imagen de autoria propia

La conexión de la placa integrada (Mosfet y el divisor de tensión) al SRF05 es:

FIGURA 33. Diagrama del SRF05

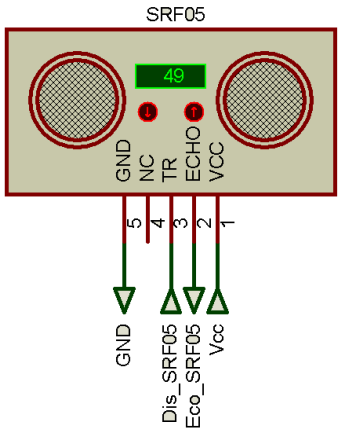


Imagen de autoria propia

- Pin D12: resistencia R1 y led que indica si existe conexión del ESP32 a la red indicada por el usuario.
- Pin D5: resistencia R2 y led que indica si existe conexión del ESP32 a Firebase.
- Pin D4: resistencia R3 y led que indica si existe errores en el sistema.
- Pin D23: resistencia R4 y led que indica si la electroválvula está encendida o apagada y a este pin también se conectará un opto-triac para el manejo de la electroválvula.
- Pin D27: resistencia R5 y led que indica si la electrobomba MT0 está encendida o apagada y a este pin también se conectará un opto-triac para el manejo de la electrobomba MT0.
- Pin D26: resistencia R6 y led que indica si la electrobomba MT1 está encendida o apagada y a este pin también se conectará un opto-triac para el manejo de la electrobomba MT1.
- Pin D24: resistencia R7 y led que indica si las lámparas ultravioleta están encendidas o apagadas y a este pin también se conectará un opto-triac para el manejo de las lámparas.

La conexión de estas resistencias y led’s al ESP32 es:

FIGURA 34. Conexión de Leds al ESP32

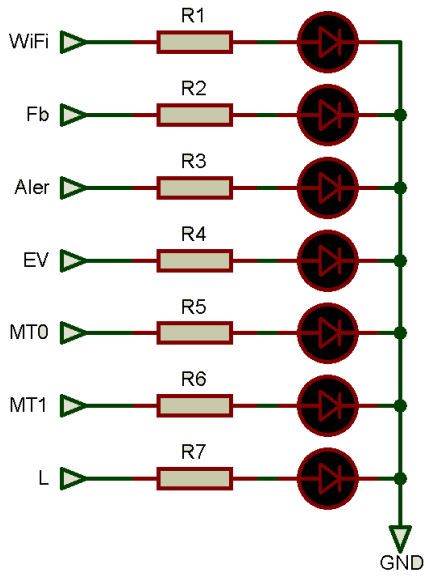


Imagen de autoria propia.

Calculo de resistencias (R1, R2, R3, R4, R5, R6, R7): cada uno de los led's requiere 10 mA y una caída de tensión de 1.7 vol., el ESP32 suministra un voltaje de 3.3 vol en sus pines de salida cuando se encuentran en alto.

Haciendo uso de la ley de OHM tenemos:

$$R = \frac{V}{I}$$

$$R = \frac{3.3\ v - 1.7\ v}{10\ mA} R = \ 160\ ohm$$

Para muestro circuito se utilizaron resistencias de 150 ohm

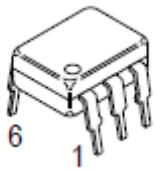
4.2.4. Etapa de Aislamiento entre etapa de Control y de Potencia

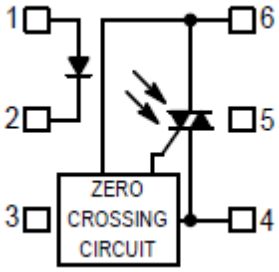
Para aislar la etapa de control de la etapa de potencia se usó el opto-triac MOC3041 que cuenta con un diodo emisor que requiere una corriente de 15 mA y

caída de voltaje de 1.25 vol para entrar en conducción. Posee un triac con detector de cruce por cero que soporta hasta 400 vol y 1 amperio.

En la etapa de potencia tenemos 4 componentes de alta potencia, una electroválvula, dos electrobombas y dos lámparas ultravioleta manejadas por un balastro de referencia OSRAM QTP 2x32T8, por lo que se utilizaron 4 MOC3041

FIGURA 35. Encapsulado MOC 3041 FIGURA 36. Diagrama MOC 3041





Imágenes tomadas de

http://exa.unne.edu.ar/ingenieria/sysistemas/public_html/Archi_pdf/HojaDatos/Optoelectronica/moc3041.pdf

FIGURA 37. Conexión del MOC 3041 al ESP32

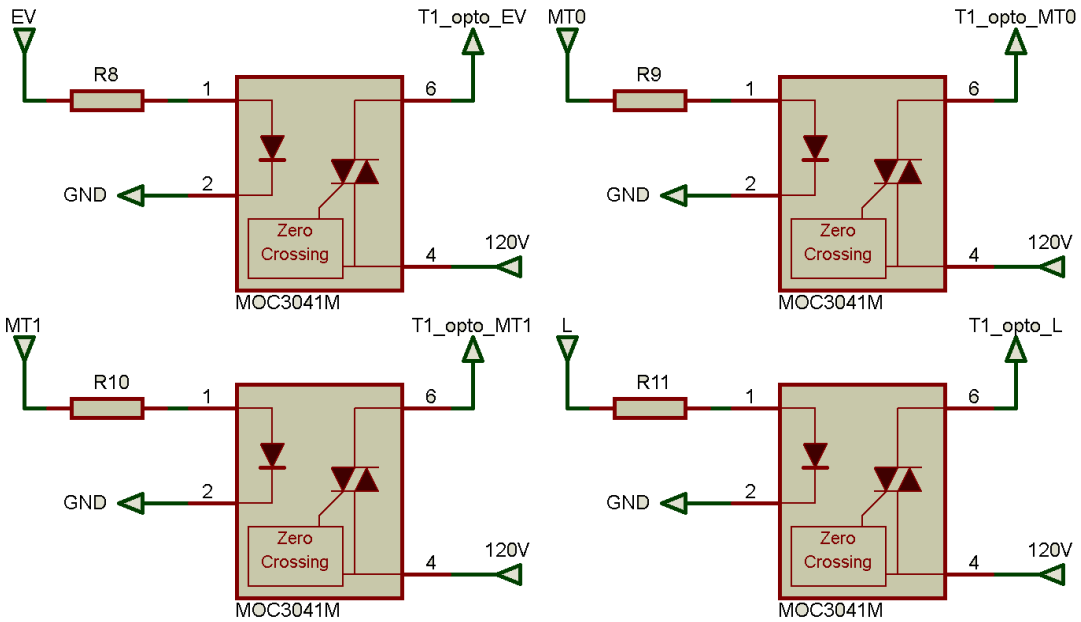


Imagen de autoria propia

Calculo de resistencias (R8, R9, R10, R11): haciendo uso de la ley de OHM y con las características eléctricas anteriormente descritas, según hoja de datos del fabricante y recordando que el ESP32 suministra un voltaje de 3.3 vol en sus pines de salida cuando se encuentran en alto tenemos.

$$R = \frac{3.3\ v - 1.25\ v}{15\ mA}$$

$$R = 136\ ohm$$

Para nuestro circuito se hace uso de resistencias de 120 ohm.

4.2.5. Etapa de Potencia.

Para el manejo de los componentes de alta potencia se ha utilizado el triac BT139 600E con características eléctricas de 600 vol y 16 amperios.

El triac BT139 es activado por orden del microcontrolador ESP32 mediante el opto-triac MOC3041.

FIGURA 38. Conexión del BT139 al MOC 3041

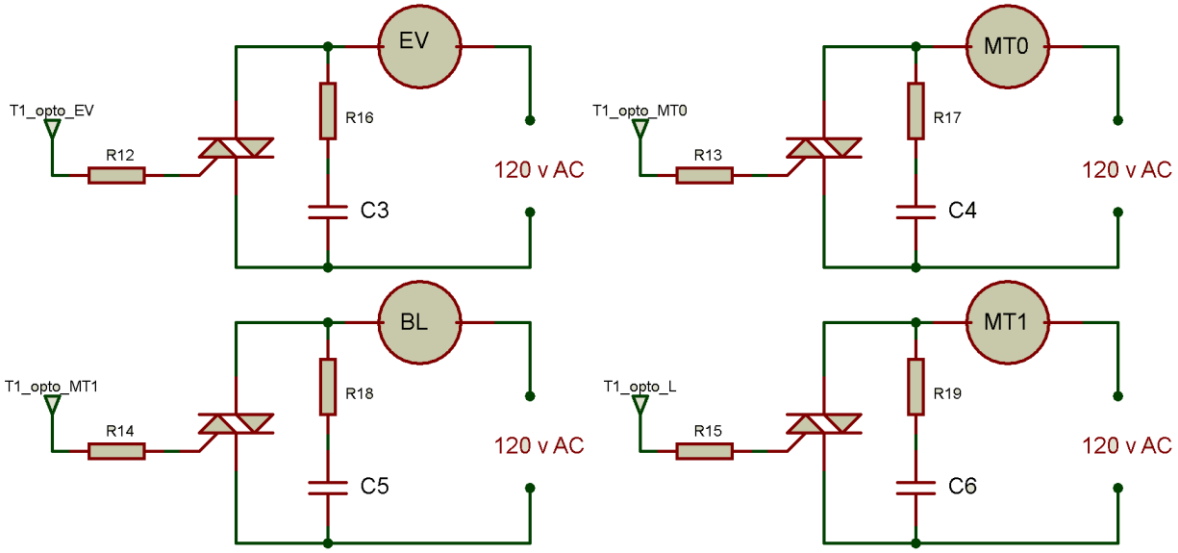


Imagen de autoría propia

Calculo de resistencias (R2, R13, R14, R15):

Cada una de estas resistencias están conectadas a los MOC3041, por lo que primero debemos calcular el valor mínimo de estas resistencias, eso dependiendo de la corriente máxima que puede manejar el triac del opto-triac, que para los MOC3041 es de 1 A.

$$R_{min} = \frac{V_{inPico}}{I_{max}}$$

Nuestro voltaje efectivo es de 120 vol., por lo cual el voltaje pico es:

$$V_{inPico} = 120\ v * \sqrt{2}$$

$$V_{inPico} = 169.7\ v$$

El valor mínimo de nuestras resistencias es:

$$R_{min} = \frac{169.7\ v}{1\ A}$$

$$R_{min} = 169.7\ ohm$$

La hoja de datos del fabricante especifica que en la practica el valor de esta resistencia es de 150 o 180 ohm, por lo que usamos resistencias de 180 ohm.

Podemos observar que los triac BT139 tienen un circuito RC conocido como circuito de freno, esto es para evitar variaciones de voltaje muy altas en el triac y evitar que se active indeseadamente.[18]

$$\frac{dv}{dt} = \frac{0.632 V_s}{R_s C_s}$$

$$R_s = \frac{V_s}{I_{TD}}$$

[18] RASHID, Electrónica de Potencia. Pág. 103. Ec 4-8 ; 4-9

En donde $\frac{dv}{dt}$ es la variación máxima de voltaje que puede soportar el BT139 sin que se active de forma indeseada, la hoja de datos especifica que es de 250 V/µs; V_s es el voltaje pico de entrada, que para nuestro caso es 120*$\sqrt{2}$ e I_{TD} es la corriente máxima que puede manejar el triac BT139, la cual es de 16 A; R_s y C_s corresponden al circuito RC que queremos calcular.

El valor de nuestras resistencias es:

$$R = \frac{120\sqrt{2}\,v}{16\,A}$$

$$R = 10.6\ ohm$$

Usaremos resistencias de 12 ohm

El valor de nuestros capacitores es:

$$C_s = \frac{0.632 V_s}{R_s \frac{dv}{dt}}$$

$$C = \frac{0.632 * \ 120\sqrt{2}\,v}{12\ ohm * \ 250\frac{V}{\mu s}}$$

$$C = 0.036 \mu F$$

Usaremos capacitores de 0.04µF (referencia 403).

El triac con propósito de manejar las lámparas ultravioleta en realidad tiene conectado el balastro OSRAM QTP 2x32T8 y a este las lámparas ultravioleta de la forma que lo indica el diagrama del mismo balastro:

FIGURA 39. Diagrama de Conexión de lámparas UV al Balastro

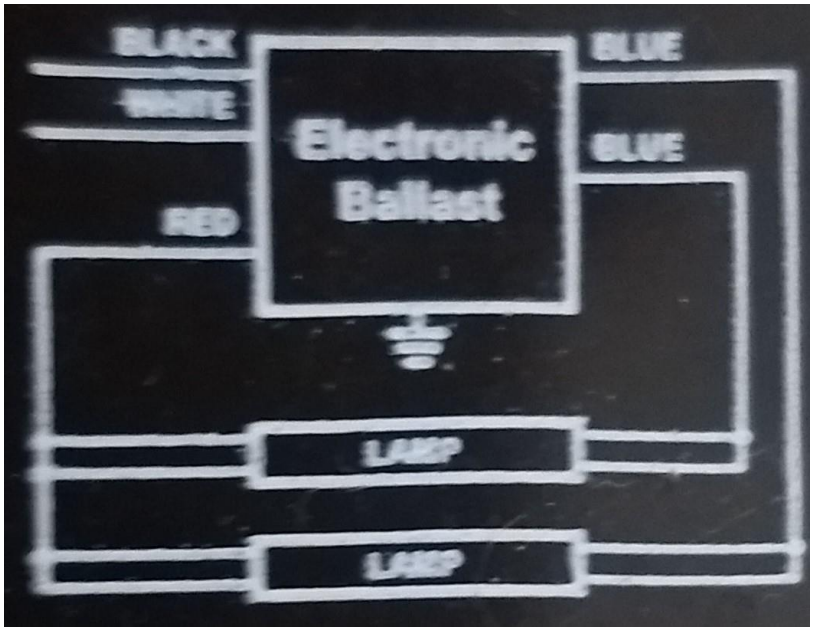
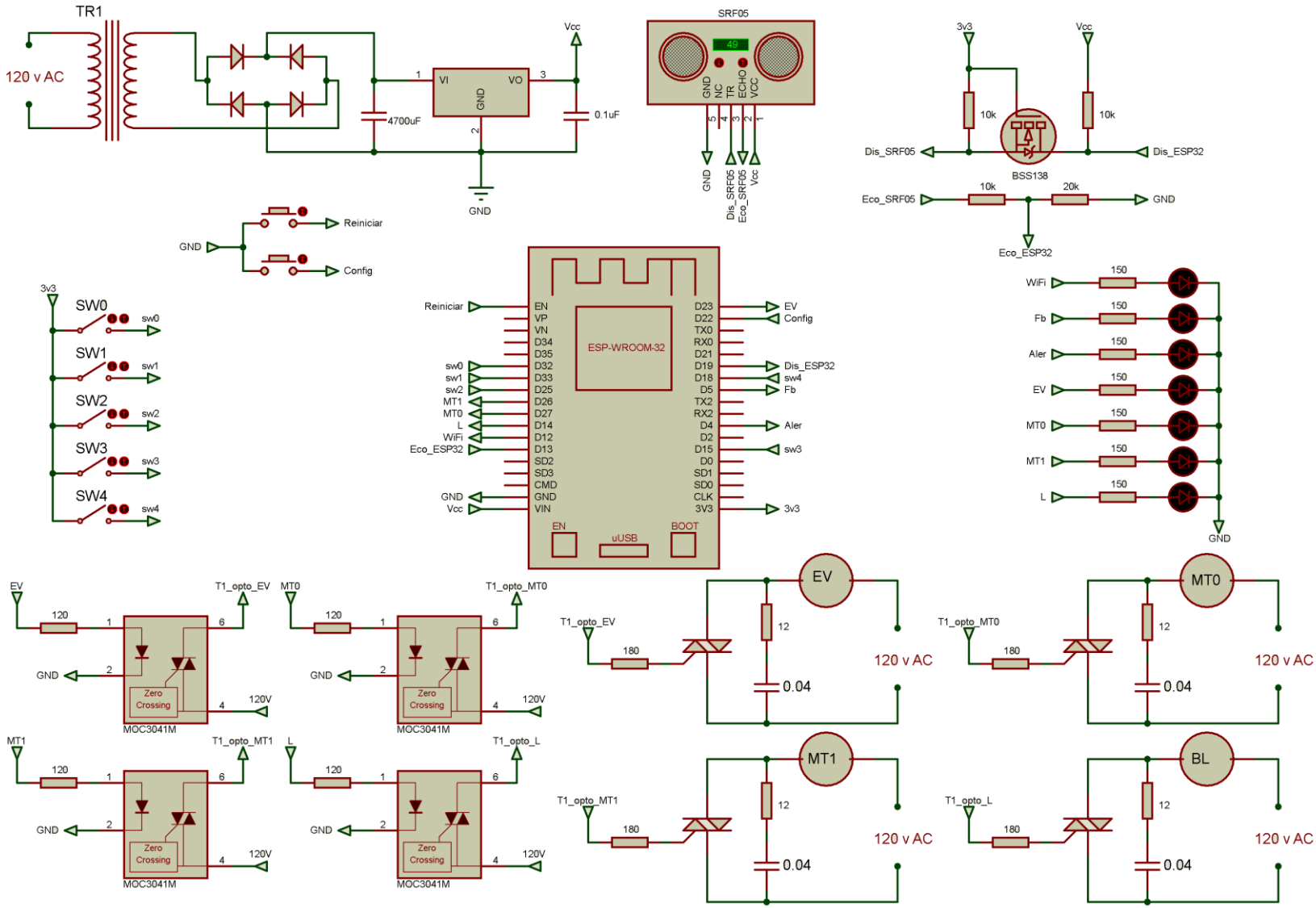


Imagen de autoria propia

4.2.6. Diagrama Completo del Circuito (Figura 40)


TR1
120 v AC
4700uF
VI
VO
GND
0.1uF
Vcc
SRF05
GND
NC
TR
ECHO
VCC
Dis_SRF05
Eco_SRF05
3v3
10k
BSS138
20k
Dis_ESP32
Eco_ESP32
Reiniciar
Config
SW0
SW1
SW2
SW3
SW4
ESP-WROOM-32
EN
VP
VN
D34
D35
D32
D33
D25
D26
D27
D14
D12
D13
SD2
SD3
CMD
VIN
D23
D22
TX0
RX0
D21
D19
D18
D5
TX2
RX2
D4
D2
D15
D0
SD1
SD0
CLK
3V3
uUSB
BOOT
WiFi
Fb
Aler
EV
MT0
MT1
L
150
MOC3041M
Zero Crossing
T1_opto_EV
T1_opto_MT0
T1_opto_MT1
T1_opto_L
120V
180
12
0.04
BL

5. SISTEMA DE BASE DE DATOS Y APLICACIÓN MOVIL (app) “AQUA VIVA”

5.1. Firebase

Como se ha indicado el circuito se conecta a internet mediante el ESP32-WROOM-32, el cual debe conectarse a un modem que le suministre acceso a internet. Una vez conectado a internet este envía la información a la base de datos RealTime Database de Firebase.

Firebase es una plataforma de la empresa Google en la que ofrece múltiples servicios en la nube, principalmente para el desarrollo de aplicaciones móviles.

Para hacer uso de los servicios de Firebase, se debe registrar y crear el proyecto, el cual debe ser conectado a la aplicación móvil (app). En la página de Firebase (https://firebase.google.com/) existe toda la documentación y manuales para hacer uso de cada uno de sus servicios.

Para el caso de nuestra app se ha usado el servicio de Realtime Database y Firebase Authentication.

[19]**Realtime Database** es una base de datos NoSQL que sincroniza con todos los clientes (usuarios que tienen instalada la aplicación móvil) en tiempo real, es decir que cuando un dato de la base de datos (BD) cambia, se ve reflejado inmediatamente en la app sin que el usuario, que debe estar conectado, tenga que refrescarla. Los datos se almacenan en formato JSON (JavaScript Object Notation).

[20]**Firebase Authentication** proporciona servicios de Backend (capa de acceso a datos), SDK (Software Development Kit) y librerías para autenticar a los usuarios en la app. Admite la autenticación mediante contraseñas, números de teléfono,

[19] Consulta por Internet. Disponible en https://firebase.google.com/docs/database/) Fecha de consulta 30 de marzo de 2018

[20] Consulta por Internet. Disponible en https://firebase.google.com/docs/auth/ Fecha de consulta 30 de marzo de 2018

proveedores de identidad federados populares, como Google, Facebook y Twitter, y mucho más.

Firebase Authentication se integra estrechamente con otros servicios de Firebase y aprovecha los estándares de la industria como OAuth 2.0 y OpenID Connect, por lo que se puede integrar fácilmente con tu backend personalizado.

5.2. Estructura de la base de datos.

Nuestra base de datos puede tener n nodos dado que fue diseñada pensando en la comercialización del proyecto de reciclaje de agua. Pensemos en que se han fabricado 5 cantidades de este producto (reciclaje de agua AQUA VIVA) para su comercialización. Cada sistema a comercializar tiene un código único que se encuentra grabado en el ESP32 y corresponde, de igual manera, a un nodo de nuestra BD donde reposará toda la información o data del sistema, enviada por el ESP32.Para nuestro ejemplo de 5 cantidades fabricadas, tendríamos nuestra BD con los siguientes nodos:

FIGURA 41. Base de datos REALTIME DATABASE – FIREBASE

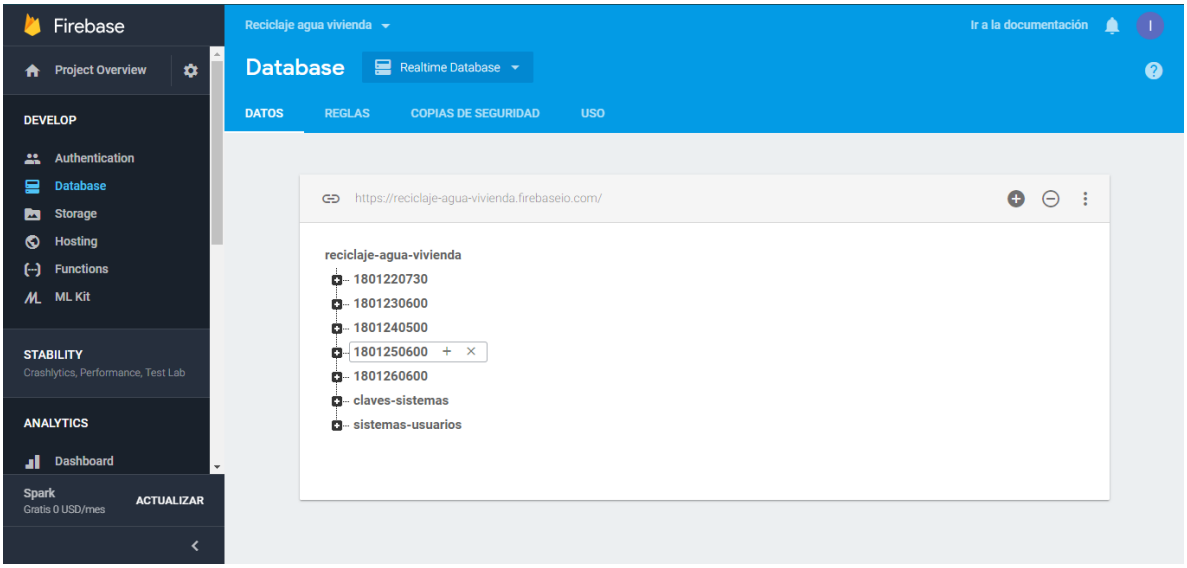


Imagen de autoria propia

Podemos clasificar estos nodos en tres principales:

Nodo “claves-sistemas”:

Figura 42. Nodo “clave-sistemas”

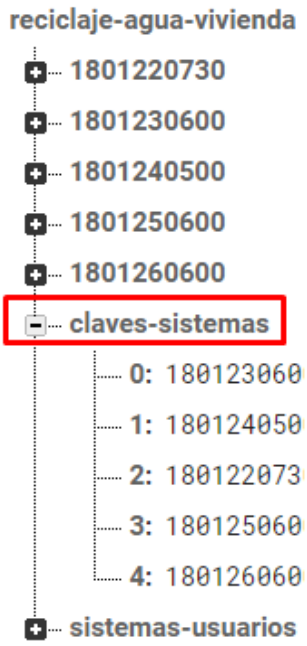


Imagen de autoria propia

En este nodo se encuentran todas las claves de los sistemas fabricados y cada vez que se fabrica un nuevo sistema se debe crear un subnodo con la clave del nuevo sistema.

Nodo “sistemas-usuarios”:

FIGURA 43. Nodo “sistemas-usuarios”

Imagen de autoria propia

Cuando una persona (usuario) compra uno de los sistemas, debería registrarse, mediante la app y allí también registrar la clave que identifica al sistema para poder estar monitoreándolo mediante la app. Un usuario puede monitorear varios sistemas siempre y cuando tenga las claves de estos. Cuando un usuario se registra, Firebase le asigna un UID que es un código único creado por Firebase en el momento del registro y este código es interno en Firebase, es decir que el usuario nunca lo conoce, explicado esto, podemos entender que este nodo relaciona el UID de cada usuario registrado con las de los sistemas que el usuario a ingresado mediante la app.

Podemos observar que este nodo tiene un nodo hijo que corresponde al UID que Firebase asigno al usuario cuando se registró, este a su vez puede tener varios nodos, uno por cada sistema que el usuario haya ingresado y contienen la clave del sistema y el nombre que el usuario quiso dar en el momento del registro, por ejemplo el nodo “U79sZtsgx5V5ZiJ3k93IPFypHFZ2”, correspondiente al UID de un usuario, tiene dos nodos hijos que corresponden a los sistemas que el usuario a ingresado (Universidad y Casa)

Nodo con la data de cada sistema:

FIGURA 44. Nodo “sistemas-usuarios”

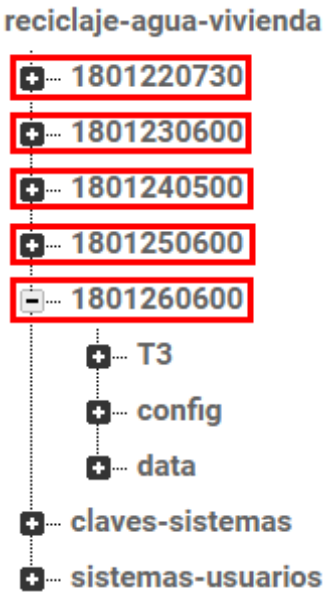


Imagen de autoria propia.

Dado nuestro ejemplo de 5 sistemas fabricados y vendidos podemos ver que existe 5 nodos principales con las calves de cada sistema, estos nodos se crean automáticamente cuando un usuario registra un sistema por primera vez y contiene tres nodos hijos, T3, config y data, donde se almacena toda la data del sistema correspondiente.

Nodo “T3”:

FIGURA 45. Nodo “T3”

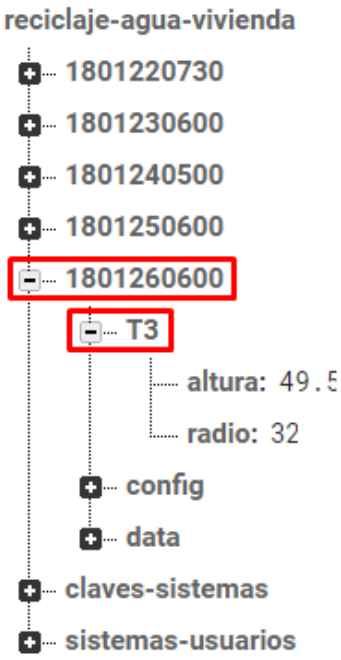


Imagen de autoria propia

Este nodo tiene las dimensiones altura y radio del tanque T3 que es donde se almacena el agua que ya ha sido tratada, con el fin de que la app, teniendo el dato de la distancia entregada por el sensor SRF05, pueda calcular el volumen del agua en litros disponibles en este tanque.

Nodo “config”:

FIGURA 46. Nodo “config”

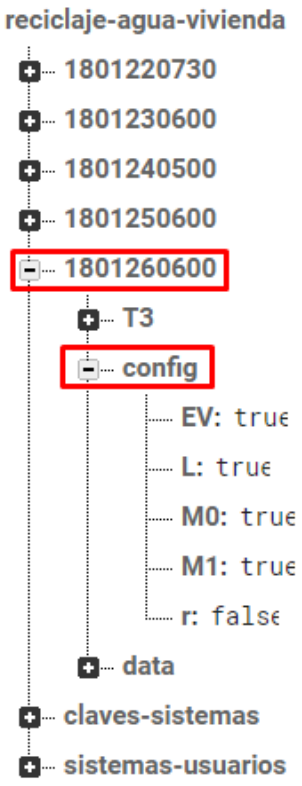


Imagen de autoria propia

Este nodo está siendo monitoreado por el ESP32 para identificar cualquier cambio en él y actuar dependiendo del dato, el usuario puede cambiar este nodo mediante la aplicación.

Los 4 primeros campos (EV, L, M0, M1) indican si la electroválvula, lámparas o electrobombas están habilitadas o no para que el circuito las active cuando corresponda. El valor “true” indica que están habilitadas y el circuito puede activarla cuando corresponda y el valor “false” indica que no pueden ser activadas por el circuito.

El campo “r” es para poder reiniciar el ESP32 desde la app, cuando el usuario indica desde la app que se reinicie el sistema, esta envía un true a este campo e inmediatamente el ESP32 se reinicia, de tal manera que cuando se ha reiniciado el ESP32, este envía un “false” al campo “r” y de esta manera la app entiende que el sistema fue reiniciado y puede notificarlo al usuario.

Nodo “data”:

FIGURA 47. Nodo “data”

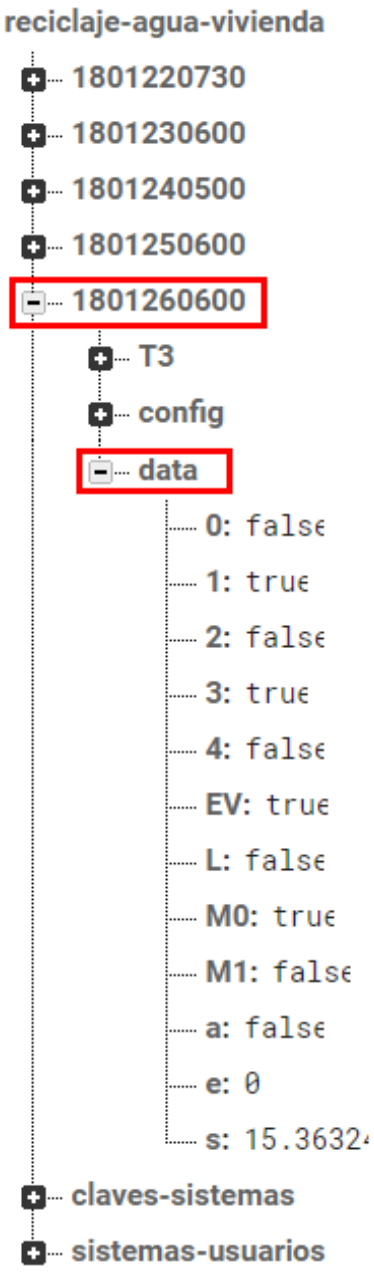


Imagen de autoría propia

Este nodo tiene toda la data principal que indica el estado actual del sistema; los campos “0, 1, 2, 3 y 4” indican el estado de los interruptores de nivel, cuando

están en “true” se indica que el interruptor de nivel está cerrado y en “false” se indica que está abierto.

Los campos “EV, L, M0 y M1” indican el estado de la electroválvula, las lámparas y las electrobombas, cuando están en “true” se indica que están encendidas o activas y en “false” se indica que están apagadas o inactivas;

En el campo “e” se guarda los posibles errores que se encuentren en el sistema y pueden existir cuatro valores, 0 o 1 o 2 o 3. En donde “0” indica que no existen errores, “1” que existe un error con los interruptores de nivel del tanque 1, “2” que existe un error con los interruptores de nivel del tanque 2 y “3” que existen errores en la lectura del sensor.

En el campo “s” se almacena la distancia que el sensor SRF05 entregó.

5.3 Herramientas de desarrollo de la aplicación móvil

La app, a la cual le hemos dado el nombre comercial “AQUA VIVA”, fue desarrollada con el framework (marco de desarrollo) IONIC y está disponible como APK para los sistemas operativos IOS y Android.

[21]IONIC: es un marco de desarrollo de aplicaciones móviles HTML5 destinado a la creación de aplicaciones móviles híbridas. Las aplicaciones híbridas son esencialmente sitios web pequeños que se ejecutan en un shell de navegador en una aplicación que tiene acceso a la capa de plataforma nativa. Las aplicaciones híbridas tienen muchos beneficios sobre las aplicaciones nativas puras, específicamente en términos de soporte de plataforma, velocidad de desarrollo y acceso a código de terceros.

Ionic viene con elementos de interfaz de usuario móvil de estilo nativo y diseños que obtendría con un SDK nativo en iOS o Android, pero que realmente no

[21] Consulta por Internet. Disponible en https://ionicframework.com/docs/v1/guide/preface.html Fecha de consulta 30 de marzo de 2018

existían antes en la web. Ionic también le brinda algunas maneras obstinadas pero poderosas de crear aplicaciones móviles que eclipsen los marcos de desarrollo de HTML5 existentes.

IONIC es de código abierto (Open Source) y está basado en el framework Angular, que es desarrollado por la empresa Goolge. Angular se basa en el lenguaje de programación TypeScript, que fue desarrollado por la empresa Microsoft.

Una aplicación móvil de IONIC se basa en un stack de páginas; básicamente una página es un componente angular. Una página de IONIC tiene tres plantillas, una plantilla TypeScript, donde se encuentra toda la lógica de la página, otra plantilla HTML5 (HyperText Markup Language) que contiene todas etiquetas que conforman la presentación de la página y otra plantilla CSS (Cascading Style Sheets) para darle estilos a la página

5.4 Aplicación Móvil.

La app AQUA VIVA permite a los usuarios estar monitoreando los sistemas de reciclaje que tiene registrados con su usuario.

Icono y splash de la app AQUA VIVA [22]

FIGURA 48. Icono y splash de aplicación móvil

Imagen tomada de https://www.freepik.com/free-vector/green-environmental-icons-collection_905036.htm

[22] Tomada de (https://www.freepik.com/free-vector/green-environmental-icons-collection_905036.htm)

5.4.1. Páginas de la aplicación móvil:

- Página de registro de usuarios: en esta página los usuarios nuevos usuarios se pueden registrar con un correo electrónico y contraseña. Una vez se registren, se enviará un link al correo electrónico con el que se registró para confirmar el proceso de registro.

FIGURA 49. Página de registro de usuarios

Imagen de autoria propia

- Página de inicio de sesión: cuando el usuario ya se encuentra registrado en el sistema y quiere acceder a los sistemas, primero debe iniciar sesión en esta página. En esta página también se puede realizar el proceso de restablecimiento de contraseña cuando el usuario la ha olvidado.

FIGURA 50. Página de inicio de sesión

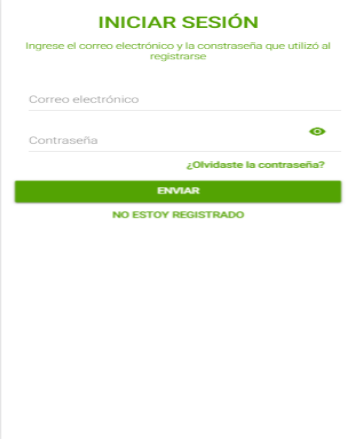


Imagen de autoria propia

- Página para registrar sistemas y seleccionar sistema a monitorear: en esta página el usuario debe registrar el o los sistemas con la clave correspondiente.

FIGURA 51. Página para registrar sistemas

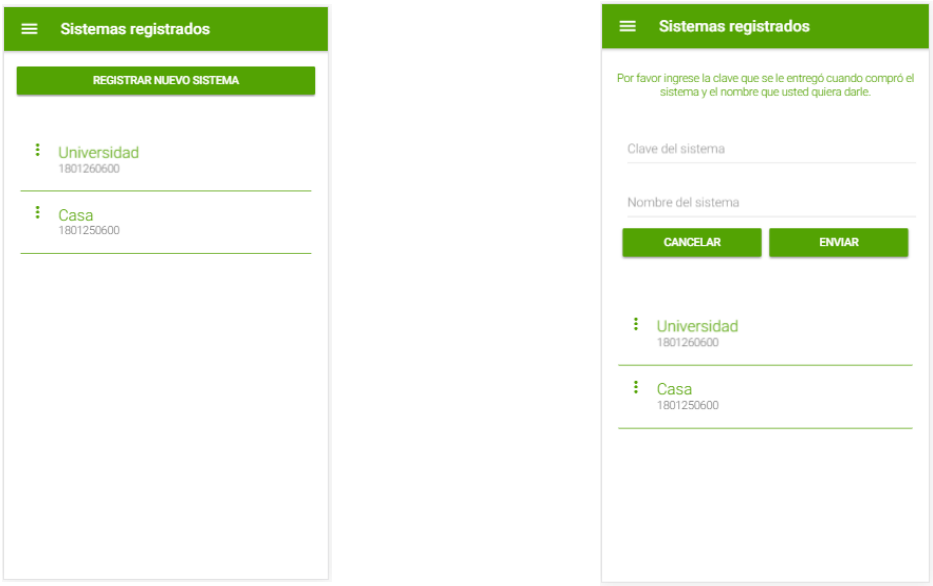


Imagen de autoría propia

Menú lateral sin sistema seleccionado: si el usuario toca el botón de menú o desliza el dedo en la pantalla de izquierda a derecha puede observar el menú.

Mientras el usuario no ha seleccionado un sistema solo existen dos opciones en el menú, una para cerrar sesión y otra para cerrar el menú.

FIGURA 52. Menú lateral sin sistema seleccionado

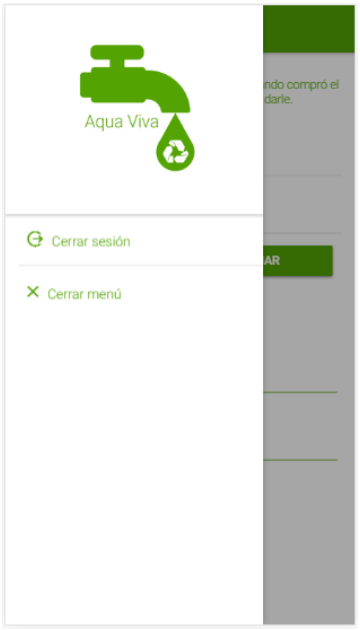


Imagen de autoria propia.

Página inicial del sistema: cuando el usuario selecciona, en la página anterior, el sistema que desea monitorear se abre esta página en la que se encuentra los botones de acceso a las páginas de cada uno de los tanques y también se observa el volumen, en litros, del agua en el tanque 3.

FIGURA 53. Página inicial del sistema

Imagen de autoria propia

- Menú lateral con sistema seleccionado: cuando el usuario ya ha seleccionado un sistema, el menú lateral tendrá las opciones para inhabilitar o habilitar los actuadores y para reiniciar el ESP32. Cuando un actuador está inhabilitado se mostrará en rojo.

FIGURA 54. Menú lateral con sistema seleccionado

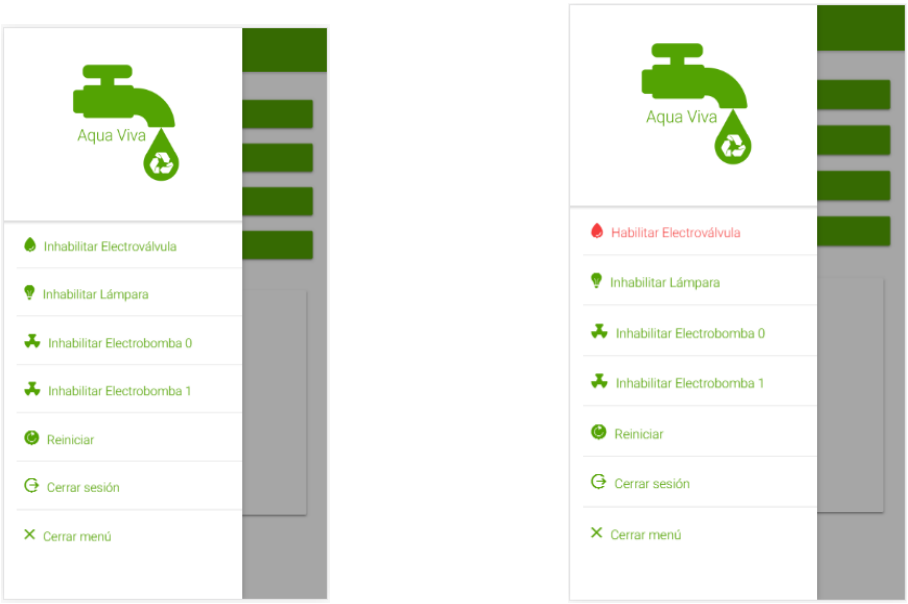


Imagen de autoria propia

- Páginas de los tanques: tocando el botón de cada uno de los tanques en la página inicial del sistema se accede a la página que contiene la información del estado de cada tanque y el estado actual de los actuadores. Cuando la imagen de un actuador está en negro, se indica que está apagado o inactivo y cuando está en rojo se indica que está encendido o activo. Cuando existe un error en el sistema se observará una “X” roja con el mensaje que indica el motivo del error.

FIGURA 55. Páginas de los tanques

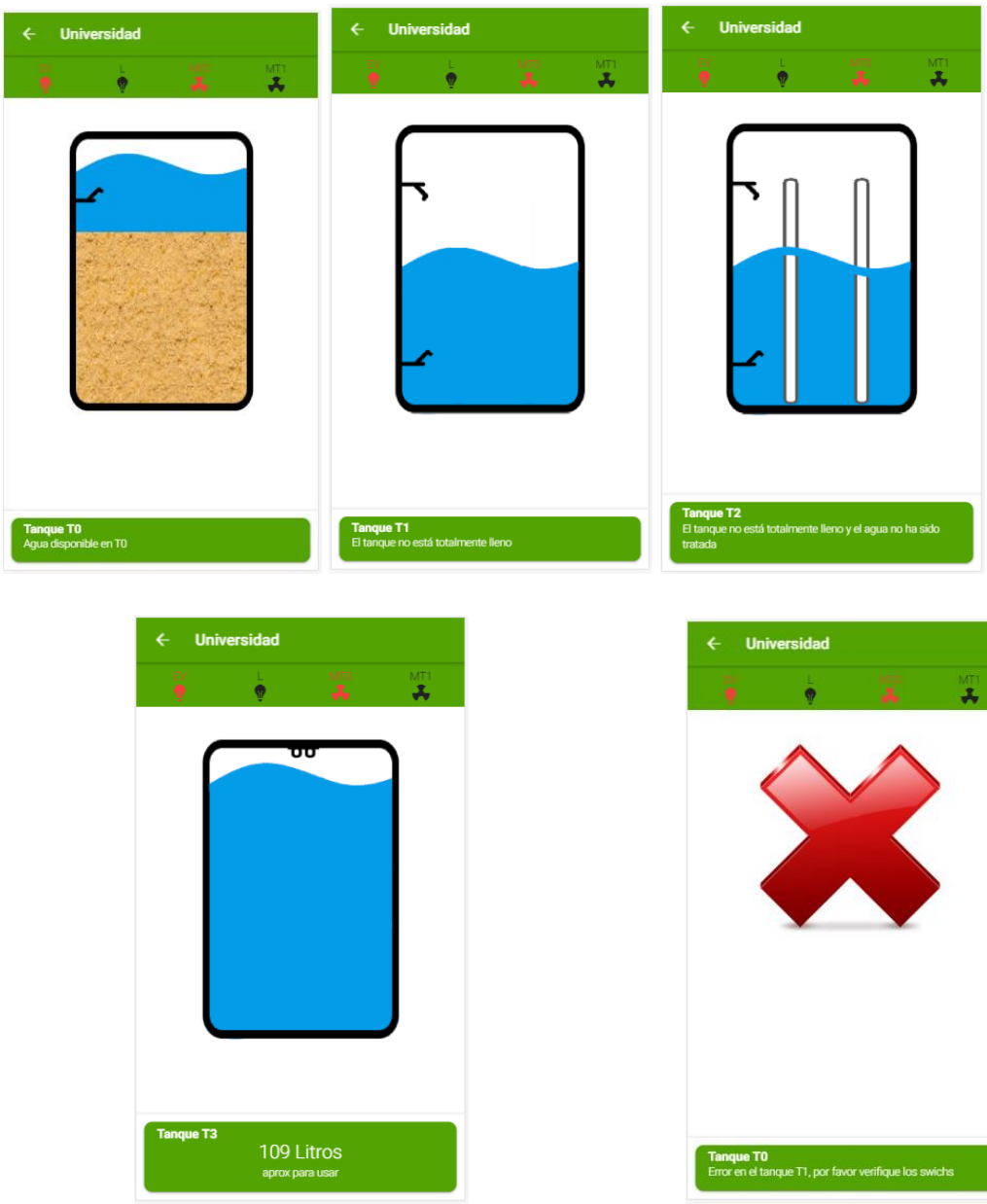


Imagen de autoria propia

5.4.2. Código fuente de la aplicación AQUA VIVA (ver ANEXO 2)

6. COSTOS DEL SISTEMA

CANTIDAD	PRODUCTO	V. UNIDAD	V. TOTAL
3	Tanques 162 Litros	$ 40.000,00	$ 120.000,00
1	Tanque de 102 Litros	$ 30.000,00	$ 30.000,00
-	Tubería PVC	$ 50.000,00	$ 50.000,00
50	cm tela para filtro	$ 2.500,00	$ 2.500,00
-	Gravilla	$ 15.000,00	$ 15.000,00
1	Filtro carbón activado	$ 120.000,00	$ 180.000,00
1	Electroválvula	$ 115.000,00	$ 115.000,00
2	Electrobombas	$ 55.000,00	$ 110.000,00
2	Lamparas UV G15WT8	$ 48.500,00	$ 97.000,00
1	Balastro osram qtp 2x32	$ 18.500,00	$ 18.500,00
1	Transformador a 9 vol	$ 18.500,00	$ 18.500,00
5	Interruptores de nivel	$ 10.000,00	$ 50.000,00
1	Sensor SRF05	$ 15.000,00	$ 15.000,00
1	ESP32 DevKit	$ 32.000,00	$ 32.000,00
4	MOC3141	$ 2.000,00	$ 8.000,00
4	BT139-600	$ 2.000,00	$ 8.000,00
1	Convertidor de nivel lógico BSS138	$ 3.800,00	$ 3.800,00
1	Rectificador de onda completa	$ 833,00	$ 833,00
1	Regulador de L78S05CV	$ 2.000,00	$ 2.000,00
1	Capacitor 4700 uF/50V	$ 3.500,00	$ 3.500,00
1	Capacitor 0,1 uF/50V	$ 60,00	$ 60,00
2	Pulsadores	$ 300,00	$ 600,00
7	Led's	$ 250,00	$ 1.750,00
17	Resistencias 1/4 watts	$ 150,00	$ 2.550,00
4	Capacito 104	$ 70,00	$ 280,00
20	Mts Cable 18 AWG	$ 300,00	$ 6.000,00
	TOTAL		$ 890.873,00

TABLA. 3

7. CONCLUSIONES Y RECOMENDACIONES.

La mayor parte del consumo elevado de agua en la vivienda corresponde al utilizado en el lavado de ropa.

La implementación de un sistema de reutilización de agua de la lavadora es rentable en la medida que la sociedad otorgue un valor realmente

Los costos para la implementación del sistema son mínimos en comparación a la gran utilidad y ahorro que este trae para nuestro bolsillo

El sistema será más comercial en la medida que se logre adaptar un solo contenedor en el que se pueda implementar todo el sistema para que haga todos sus procesos.

De ninguna manera el agua tratada por el sistema es apta para el consumo humano.

Es recomendable que el sistema sea ubicado en un lugar de fácil acceso para hacer el adecuado mantenimiento y su limpieza.

Es recomendable reutilizar nuevamente el agua que queda en los tanquespara vaciado del sanitario y/o aseo del hogar.

Se recomienda utilizar para el lavado de ropa un detergente biodegradable para que sea más fácil su tratamiento y sin problema se pueda reutilizar en riego de jardín.

8. REFERENCIAS BIBLIOGRAFICAS

LAPEÑA RIGOLA, Miguel. Tratamiento de aguas industriales: aguas de proceso y residuales. 1a ed. Barcelona España; MARCOMBO, S.A., 1990.

Malvino, Albert Paul. Principios de electrónica. Sexta edición. 2000 y Séptima edición 2007. Mc Graw Hill

RASHID, Muhammad H. “Electrónica de Potencia, Circuitos, dispositivos y Aplicaciones” México tercera edición. Prentice Hall. 2004

KOLBAN´s book ESP32. Libro virtual. https://leanpub.com/kolban-ESP32

9. ANEXOS

9.1 ANEXO 1**. Código fuente ESP32-WROOM-32:**

```
/*-----------------------------------------------LIBRERÍAS-----------------------------------*/
/*WiFi.h: Librería creada por la empresa fabricante del ESP32, espressif, facilita el uso del módulo wifi del ESP32.
Repositorio: https://github.com/espressif/arduino-esp32

IOXhop_FirebaseESP32.h: Librería creada por la empresa IOXhop, la cual facilita la conexión del ESP32 con Firebase.
Repositorio: https://github.com/ioxhop/IOXhop_FirebaseESP32

ArduinoJson.h: Librería creada por bblanchon, la cual facilita el uso de objetos json para el envío y recepción de información de Firebase
Repositorio: https://github.com/bblanchon/ArduinoJson

FS.h: Librería creada por la empresa fabricante del ESP32, espressif. Es una librería que gestiona el sistema de archivos del ESP32 y aunque no utilizamos el sistema de archivos, se requiere incluirla ya que la librería ESPAsyncWebServer.h es una dependencia de esta.
Repositorio: https://github.com/espressif/arduino-esp32

AsyncTCP.h: Librería creada por me-no-dev, librería TCP destinada a habilitar un entorno de red de conexión múltiple para el ESP32. se requiere incluirla ya que la librería ESPAsyncWebServer.h es una dependencia de esta.
Repositorio: https://github.com/me-no-dev/ESPAsyncWebServer

ESPAsyncWebServer.h: Librería creada por me-no-dev, la cual facilita la configuración del ESP32 como un servidor HTTP.
Repositorio: https://github.com/me-no-dev/ESPAsyncWebServer
```

```
EEPROM.h: Librería creada por la empresa fabricante del ESP32, espressif, facilita el uso de la memoria EEPROM del ESP32.
Repositorio: https://github.com/espressif/arduino-esp32
*/
#include <WiFi.h>
#include <IOXhop_FirebaseESP32.h>
#include <ArduinoJson.h>

#include <FS.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

#include <EEPROM.h>

/*-----------------------------------DECLARACIÓN DE PINES--------------------------------*/
/*
sw0 es el interruptor que está en T0 por encima del filtro de arena y se encarga garantizar que el filtro de gravilla no quede sin agua.

sw1(en la base de T1) y sw2(en la parte media de T1) son los interruptores de nivel, encargados de indicar los niveles del agua en T1, de tal manera que ayuden a decidir cuándo se debe encender o apagar la electroválvula y la electrobomba MT0.

sw3(en la base de T2) y sw4(en la parte media de T2) son los interruptores de nivel, encargados de indicar los niveles del agua en T2, de tal manera que ayuden a decidir cuándo se deben encender o apagar las electrobombas MT0 y MT1 y la Lámpara ultravioleta.
```

EV es la electroválvula ubicada entre T0 y T1, encargada de controlar el paso del agua de T0 a T1 y garantizar el que el filtro de gravilla siempre tenga agua.

L es la lámpara ultravioleta ubicada en T2, la cual se enciende cuando T2 se ha llenado, cumpliendo la función de desinfección de microorganismos; el tiempo de encendido es de 40s aproximadamente.

MT0 es la electrobomba ubicada entre T1 y el filtro de carbón activado (filtro ubicado antes de T2), la cual se encarga de enviar el agua con presión al filtro, ya que se requiere de presión para que el filtro funcione correctamente.

MT1 es la electrobomba ubicada entre T2 y T3, la cual se encarga de enviar el agua a T2, tanque elevado para disposición del agua tratada.

alerta es un indicador visual (led-rojo) de los errores que su pudieran estar presentando en todo el sistema. Se pueden identificar 3 errores.
Error 1: cuando sw1=0 y sw2=1, el cual indica un fallo en los interruptores de nivel de T1, ya que no se puede presentar que el nivel del agua esté por encima de sw2 y al mismo tiempo por debajo de sw1.
Error 2: cuando sw3=0 y sw4=1, el cual indica un fallo en los interruptores de nivel de T2, ya que no se puede presentar que el nivel del agua esté por encima de sw4 y al mismo tiempo por debajo de sw3.
Error 3: cuando el SRF05 genere una lectura incorrecta, como por ejemplo que la distancia medida sea mayor que la distancia del SRF05 al fondo de T3 o que sea menor a la distancia mínima que debe existir entre el SRF05 y el nivel del agua.

disparo es pin del ESP32 que se encarga de generar la orden al SRF05 de iniciar una medición.

eco es el pin del ESP32 que recibe la señal del SRF05 que permite realizar la medición de distancia.

led_wf: es un indicador visual que ayuda a identificar el estado de conexión wifi del ESP32. Dependiendo de su estado o parpadeo se puede identificar lo siguiente:
Apagado: Indica que no se han ingresado credenciales de la red del modem.
Encendido: Indica que el ESP32 está conectado al modem.
Parpadeo de 500 ms: Indica que el ESP32 está intentando conectarse a la red indicada.
Parpadeo de 250 ms: Indica que el ESP32 se encuentra conectado a la red y está en proceso de que se le asigne una dirección IP.
Apagado después de tres parpadeos: Indica que el ESP32 ha ingresado en modo AP(access point), pero no se ha podido asignar la ip predefinida (192.168.0.1) para el servidor HTTP.
Encendido después de tres parpadeos: Indica que el ESP32 ha ingresado en modo AP(access point) y ya obtuvo la ip predefinida (192.168.0.1) para el servidor HTTP. En este momento se puede ingresar las credenciales de la red a la cual se debe conectar el ESP32.

led_fb: es un indicador visual que ayuda a identificar el estado de la conexión del ESP32 a Firebase. Dependiendo de su estado o parpadeo se puede identificar lo siguiente:
Apagado: no existe conexión con Firebase y tampoco existe conexión con la red indicada.
Encendido: Indica que existe conexión con Firebase.
Parpadeo de 500 ms: indica que se está intentando conectar con Firebase.

pul_config: es un pulsado que permite habilitar el ESP32 en modo AP para poder ingresar las credenciales de la red para conexión a internet.
*/
```
#define SW0         32
#define SW1         33
#define SW2         25
```

```
#define SW3         15
#define SW4         18
#define EV          23
#define MT0         27
#define MT1         26
#define L           14
#define alerta      4
#define disparo     19
#define eco         13
#define led_wf      12
#define led_fb      5
#define pul_config  22

/*----------------------------------DECLARACIÓN DE VARIABLES--------------------------*/
/*
```

ssid_esp32: es el nombre de la red que habilita el ESP32 en modo AP para que el usuario pueda ingresar las credenciales de la red a la que se debe conectar el ESP32 cuando esté en modo STA(estación) para acceso a internet.

pass_esp32: es la contraseña de la red que habilita el ESP32 en modo AP.

local_IP: es la dirección IP que asigna el ESP32 al servidor HTTP cuando está en modo AP.

gateway: es la puerta de enlace predeterminada del ESP32 en modo AP.

subnet: es la máscara de subred del ESP32 en modo AP.

server: es una variable de tipo AsyncWebServer que permite configurar el servidor. Como parámetro recibe el puerto 80, que es el que se usa para comunicación HTTP.

```
*/

const char* ssid_esp32 = "Aqua viva";
const char* pass_esp32 = "aquaviva01";

IPAddress local_IP(192, 168, 0, 1);
IPAddress gateway(192, 168, 0, 1);
IPAddress subnet(255, 255, 0, 0);

AsyncWebServer server(80);

/*
HTML y HTML1 son la paginas web que provee el servidor del ESP32 cuando está en modo AP.
HTML es la página que se provee cuando el usuario accede a la IP 192.168.0.1 y está habilita un formulario de dos campos para poder ingresar las credenciales de la red a la que se debe conectar el ESP32 cuando está en modo STA(estación) para acceder a internet.

HTML1 es la página web que el ESP32 entrega cuando las credenciales han sido guardadas en la EEPROM y es básicamente la misma página web que HTML, solo que contiene un mensaje indicando que las credenciales han sido guardadas.

PROGMEM es una palabra clave que le indica al compilador que la información HTML y HTML1 debe guardarse en la memoria flash en vez de hacerlo en la memoria RAM.
*/

const char HTML[] PROGMEM = "<!DOCTYPE html><html lang=\"en\"><head>
<meta charset=\"UTF-8\"><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\"> <meta http-equiv=\"X-UA-Compatible\" content=\"ie=edge\">
```

<title>Aqua viva</title> <style>body{margin: 0;}.titulo{margin-bottom: 1.5rem; color: #53A303; font-size: 1.5rem;}div{text-align: center; margin-bottom: 1rem;}label{text-align: left; color: #53A303; width: 80px; padding-right: 13rem;}input{border-color: #53A303; border-style: solid; border-radius: 0.2rem; width: 18.5rem; height: 1.5rem;}.barra{background-color: #53A303; padding: 0.5rem; margin-bottom: 2rem; font-size: 1.5rem; color: #f4f4f4;}button{margin-top: 1rem; background-color: #53A303; color: #f4f4f4; width: 18.7rem; height: 36px; font-size: 1rem; border-color: #53A303; border-style: solid; border-radius: 0.2rem;}</style></head><body> <div class=\"barra\">AQUA VIVA <div class=\"formulario\"> <div class=\"titulo\">Credenciales Wifi <form action=\"captura_info\" method=\"get\"> <label>Nombre red:</label> <input type=\"text\" name=\"ssid\" required> <label>Contraseña:</label> <input type=\"text\" name=\"pass\" required> <button type=\"submit\"> GUARDAR </button> </form> </body></html>";

const char HTML1[] PROGMEM = "<!DOCTYPE html><html lang=\"en\"><head> <meta charset=\"UTF-8\"> <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\"> <meta http-equiv=\"X-UA-Compatible\" content=\"ie=edge\"> <title>Aqua viva</title> <style>body{margin: 0;}.titulo{margin-bottom: 1.5rem; color: #53A303; font-size: 1.5rem;}div{text-align: center; margin-bottom: 1rem;}label{text-align: left; color: #53A303; width: 80px; padding-right: 13rem;}input{border-color: #53A303; border-style: solid; border-radius: 0.2rem; width: 18.5rem; height: 1.5rem;}.barra{background-color: #53A303; padding: 0.5rem; margin-bottom: 2rem; font-size: 1.5rem; color: #f4f4f4;}button{margin-top: 1rem; background-color: #53A303; color: #f4f4f4; width: 18.7rem; height: 36px; font-size: 1rem; border-color: #53A303; border-style: solid; border-radius: 0.2rem;}h4{color: #53A303; text-align: center;}</style></head><body> <div class=\"barra\">AQUA VIVA <div class=\"formulario\"> <div class=\"titulo\">Credenciales Wifi <form action=\"captura_info\" method=\"get\"> <label>Nombre red:</label> <input type=\"text\" name=\"ssid\" required> <label>Contraseña:</label> <input type=\"text\" name=\"pass\" required> <button type=\"submit\">

```
GUARDAR </button> </form> </div><h4>Credenciales de Wifi guardadas correctamente</h4></body></html>";

/*
ssid_wf: Es una matriz de una fila y 50 columnas que contendrá el nombre de la red a la que se debe conectar el ESP32 para tener acceso a internet.

pass_wf: Es una matriz de una fila y 50 columnas que contendrá la contraseña de la red a la que se debe conectar el ESP32 para tener acceso a internet.

long_ssid: Variable en la que se almacenará la longitud del nombre de red ingresado por el usuario, sirve para determinar si al iniciar el sistema existen credenciales de red ingresadas.

FIREBASE_HOST: contiene la url del proyecto (base de datos) en Firebase.
*/

char ssid_wf[50];
char pass_wf[50];
int long_ssid;
#define FIREBASE_HOST "reciclaje-agua-vivienda.firebaseio.com"

float dist_SRF05 = 0; /*Almacena la distancia en centímetros medida por el sensor.*/
float dist_SRF05_ant = 0; /*Almacena la distancia medida por el SRF05 en el ciclo anterior*/
float dist_ant_fb = 0; /*Almacena la distancia en centímetros medida por el sensor en el último envío a Firebase.*/
float dist_SRF05_Fondo_T3 = 49.5; /*Almacena la distancia entre el SRF05 y el fondo de T3.*/
```

```
float dist_min_SRF05 = 10; /*Almacena la distancia mínima que debe medir el SRF05, para garantizar que T2 no se rebose o el agua se acerque mucho al SRF05*/
int cont_error_SRF05 = 0; /*Almacena la cantidad de veces que el SRF05 realiza una medición incorrecta y si es frecuente el error, entonces detener el sistema*/
int cont_dist = 0; /*Almacena las veces que el SRF05 de manera consecutiva genera una medición con cambio de más de 2cm respecto a la medición anterior, para poder descartar las mediciones incorrectas*/

/*Variables que almacenaran los valores de interruptores de nivel y actuadores*/
bool sw0;
bool sw1;
bool sw2;
bool sw3;
bool sw4;
bool ev;
bool l;
bool mt0;
bool mt1;
bool estado_agua = 0;  /*Variable bandera que indica si el agua en T2 ya fue tratada (true) o no (false)*/
```

cont_conxs_rechazadas: variable que cuenta las veces en que existe un error al leer o escribir en Firebase.

error_conex_fb: variable que indica si existe un error en la conexión del ESP32 con Firebase, con el fin de no enviar o recibir información de Firebase y también para que el ESP32 reestablezca la conexión con Firebase. True indica que no hay conexión con Firebase y False indica que existe conexión.

stream_existe: variable que indica que el ESP32 está suscrito al servidor de Firebase, escuchando el nodo config. True indica que está suscrito y false que no lo está.

toggle: variable que permite cambiar de estado un led.

ip_obtenida: variable que indica si el modem asignó, recientemente, una dirección IP al ESP32 con el fin de reestablecer la conexión con Firebase.

reconx_inmediata: variable que indica si debe realizarse un intento de reconexión inmediata a Firebase, esto ocurre cuando el modem ha asignado una IP a Firebase después de que este ya ha estado funcionando.

red_no_encontrada: variable que indica si al iniciar el ESP32 se encontró la red especificada por el usuario.

_evento: variable que almacena el evento actual del estado de conexión del ESP32 al modem
*/

```
int cont_conxs_rechazadas = 0;
bool error_conex_fb;
bool stream_existe = 0;
bool toggle;
bool ip_obtenida = 0;;
bool reconx_inmediata = 1;
bool red_no_encontrada = 1;
WiFiEvent_t _evento;
```

/*
ms_ant_SRF05: almacena el valor de la función millis() cuando finaliza el proceso de medición por el SRF05 para garantizar que se realicen mediciones con

intervalos de tiempo no menores a 20 ms, tal como lo indica el fabricante del SRF05 en su hoja de datos.

ms_ant_fb: almacena el valor de la función millis() cuando finaliza el proceso de análisis para el envío de información a Firebase, con el fin de ejecutar en intervalos de tiempo no menor a 3s.

ms_reconx_fb: almacena el valor de la función millis() cuando finaliza el intento de reconexión del ESP32 a Firebase.

ms_ant_led_toggle: almacena el valor de la función millis() cuando finaliza el proceso de cambio de estado de un led.

ms_buscar_red: almacena el valor de la función millis() cuando finaliza el proceso de búsqueda de la red para conectarse a internet.

ms_ant_L: almacena el valor de millis() cuando L fue encendida, con el fin de mantenerla encendida por 20s.
*/
```
unsigned long ms_ant_SRF05 = 0;
unsigned long ms_ant_fb = 0;
unsigned long ms_reconx_fb = 0;
unsigned long ms_ant_led_toggle = 0;
unsigned long ms_buscar_red = 0;
unsigned long ms_ant_L = 0;
```

/*
Arreglo que almacenará las variables de los interruptores de nivel, los actuadores y estado_agua. Un arreglo almacena los datos del ciclo actual y el otro los datos del último ciclo enviado a Firebase.
*/

```
bool var_data_act[9];
bool var_data_ant[9];

/*Variables que almacenarán los valores del nodo config de Firebase*/
bool EV_hab = 1;
bool L_hab = 1;
bool MT0_hab = 1;
bool MT1_hab = 1;
bool reiniciar = 0;
int error = 0;

/*-----------------------------------------------FUNCIONES---------------------------------------*/
/*Función que cambia el estado de un led cada cierto tiempo*/
void toggle_led(int pin, int tiempo){
  if((abs(millis() - ms_ant_led_toggle)) >= tiempo){
    toggle = !toggle;
    digitalWrite(pin, toggle);
    ms_ant_led_toggle = millis();
  }
}

/*
Función que retorna la distancia en centímetros del objeto al SRF05. Se ingresa como parámetros el pin de disparo y el pin de eco.
vsonido: velocidad del sonido en cm/us (346,8 m/s a 26°c. Aumenta 0.6m/s por cada °c)

Tpulso_us: almacena el tiempo durante el cual permanece en alto el pin eco.

La función pone el pin disparo en 1 y espera 20us, según lo indicado por el fabricante en su hoja de datos, para volver a ponerlo en 0.
```

La función pulseIn() espera que el pin eco se ponga en 1 y almacena, en Tpulso, el tiempo durante el cual permanece en ese estado.

La función retorna la distancia en centímetros de acuerdo a la medición entregada por el SRF05. La distancia es el tiempo medido * la velocidad del sonido dividido entre 2, ya que la señal realiza el recorrido de ida y regreso.
*/

```
float cal_dist_srf05(int pin_disparo, int pin_eco){
  float vsonido = 0.03468;
  float Tpulso_us = 0;
  digitalWrite(pin_disparo, 1);
  delayMicroseconds(20);
  digitalWrite(pin_disparo, 0);
  Tpulso_us = pulseIn(pin_eco, 1);
  return Tpulso_us * vsonido / 2;
}
```

/*
Función que verifica errores de conexión con Firebase y lo muestra en el puerto serial. Recibe como parámetro un char que indica el lugar desde donde fue llamada la función.

Luego de detectar tres errores consecutivos se cambia la variable error_conex_fb para que no se intente enviar información a Firebase y en cambio se realice una reconexión.
*/

```
bool verif_error_fb(const char* lugar){
  if(Firebase.failed()){
    cont_conxs_rechazadas++;
    Serial.print("Error: ");
    Serial.print(Firebase.failed());
```

```
    Serial.print(" en ");
    Serial.print(lugar);
    Serial.print(". ");
    Serial.println(Firebase.error());
    if(cont_conxs_rechazadas >= 3){
      error_conex_fb = 1;
      cont_conxs_rechazadas = 0;
      WiFi.reconnect();
    }
    return true;
  }
  else{
    digitalWrite(led_fb, 1);
    cont_conxs_rechazadas = 0;
    error_conex_fb = 0;
    Serial.print("Se envió a Firebase desde: ");
    Serial.println(lugar);
    return false;
  }
}

/*
Función que se encarga de armar el objeto json que se enviará al nodo data de Firebase.
Se crea el objeto con la capacidad necesaria y los datos a enviar, para después enviar el objeto a Firebase.
*/
void enviarDataJsonFirebase(){
    StaticJsonBuffer<200> jsonBuffer;
    JsonObject& json = jsonBuffer.createObject();
    json["0"] = sw0;
```

```
    json["1"] = sw1;
    json["2"] = sw2;
    json["3"] = sw3;
    json["4"] = sw4;
    json["s"] = dist_SRF05;
    json["EV"] = ev;
    json["M0"] = mt0;
    json["M1"] = mt1;
    json["L"] = l;
    json["a"] = estado_agua;
    json["e"] = error;

    Firebase.set("1801260600/data", json);
    verif_error_fb("set json");
}

void enviarConfigJsonFirebase(){
/*
Se crea un objeto para reiniciar los valores del nodo config de Firebase, habilitando los actuadores y poniendo en false el campo r(reiniciar), para luego enviar el objeto a Firebase.
*/
  EV_hab = 1;
  L_hab = 1;
  MT0_hab = 1;
  MT1_hab = 1;
  reiniciar = 0;

  StaticJsonBuffer<200> jsonBuffer1;
  JsonObject& json_config = jsonBuffer1.createObject();
  json_config["EV"] = EV_hab;
```

```
  json_config["L"] = L_hab;
  json_config["M0"] = MT0_hab;
  json_config["M1"] = MT1_hab;
  json_config["r"] = reiniciar;

  Firebase.set("1801260600/config", json_config);
}

/*
Función que suscribe el ESP32 a Firebase para estar monitoreando el nodo config. cada vez que existe un cambio en este nodo, Firebase notifica al ESP32.
*/

void firebaseStream(){
  /*
  Código que se ejecuta cuando existe un cambio en el nodo config de Firebase.
  Actualiza la variable que ha cambiado (MT0_hab, MT1_hab, L_hab). Si el dato
  que cambió es r(reiniciar), se realiza el reinicio del ESP32.
  */
  Firebase.stream("1801260600/config", [](FirebaseStream stream) {
    Serial.println(stream.getEvent());
    String getevento = stream.getEvent();
    if((getevento != "put") && (getevento != "patch") && (getevento != "keep-alive")){
      Serial.println("Hola");
    }
    if(stream.getEvent() == "put"){
      JsonObject& json_config1 = stream.getData();
      EV_hab = json_config1["EV"];
      L_hab = json_config1["L"];
      MT0_hab = json_config1["M0"];
      MT1_hab = json_config1["M1"];
```

```
      reiniciar = json_config1["r"];

      if(reiniciar){
        ESP.restart();
      }
    }
  });
}

/*Función que recibe los eventos de cambio de estado en la conexión wifi en modo STA*/
void eventosWifi(WiFiEvent_t evento){
  Serial.print("Evento Wifi es: ");
  Serial.println(evento);

  switch(evento){
/*
Evento que se genera cuando se detiene el modulo wifi del ESP32. Cuando esto ocurre se modifican todas las variables necesarias para que el sistema funcione sin conexión a internet e intente reconectarse.
*/
    case SYSTEM_EVENT_STA_STOP:
      _evento = evento;
      Serial.print("Evento SYSTEM_EVENT_STA_STOP, codigo: ");
      Serial.println(evento);
      error_conex_fb = 1;
      digitalWrite(led_fb, 0);
      EV_hab = 1;
      L_hab = 1;
      MT0_hab = 1;
      MT1_hab = 1;
```

```
      if(stream_existe){
        Firebase.stopStream();
        stream_existe = 0;
        delay(500);
      }
      WiFi.reconnect();
      break;


/*Evento que se genera cuando el modulo wifi del ESP32 se conecta al modem.*/
    case SYSTEM_EVENT_STA_CONNECTED:
      _evento = evento;
      Serial.print("Evento SYSTEM_EVENT_STA_CONNECTED, codigo: ");
      Serial.println(evento);
      break;

/*Evento que se genera cuando el modulo wifi del ESP32 se desconecta del
modem.*/
    case SYSTEM_EVENT_STA_DISCONNECTED:
      _evento = evento;
      Serial.print("Evento SYSTEM_EVENT_STA_DISCONNECTED, codigo: ");
      Serial.println(evento);
      error_conex_fb = 1;
      digitalWrite(led_fb, 0);
      EV_hab = 1;
      L_hab = 1;
      MT0_hab = 1;
      MT1_hab = 1;

      if(stream_existe){
```

```
      Firebase.stopStream();
      stream_existe = 0;
      delay(500);
    }
    WiFi.reconnect();
    break;

/*Evento que se genera cuando el modulo wifi del ESP32 obtiene una ip del modem.*/
  case SYSTEM_EVENT_STA_GOT_IP:
    _evento = evento;
    if(stream_existe){
      Serial.println("stop stream");
      Firebase.stopStream();
      stream_existe = 0;
      delay(500);
    }
    Serial.print("Evento SYSTEM_EVENT_STA_GOT_IP, codigo: ");

    /*Se inicia la conexión con el proyecto en Firebase*/
    digitalWrite(led_wf, 1);
    Serial.println("fin evento");
    ip_obtenida = 1;
    break;

  default:
    Serial.print("Entró default");
    Serial.println(evento);
  }
}
```

```
/*Función que realiza la búsqueda de la red registrada en la EEPROM*/
bool buscarRed(){
  Serial.println("Buscando red");
  int redes = WiFi.scanNetworks();
  if (redes != 0){
    Serial.print("Se encontraron ");
    Serial.print(redes);
    Serial.println(" redes");
    for (int i = 0; i < redes; ++i){
      Serial.print("Wifi ");
      Serial.print(i+1);
      Serial.print(": ");
      Serial.println(WiFi.SSID(i));
      if(WiFi.SSID(i) == ssid_wf){
        Serial.println("Wifi encontrada");
        return true;
      }
    }
    return false;
  }
  else {
    Serial.println("No existen redes");
    return false;
  }
}

/*Función para grabar en la EEPROM*/
void grabarEeprom(int ubicacion, String dato){
  int long_dato = dato.length();
  char arreglo[50];
  dato.toCharArray(arreglo, long_dato+1);
```

```
  for(int i = 0; i < long_dato; i++){
    EEPROM.write(ubicacion+i, arreglo[i]);
  }
  for(int i = long_dato; i < 50; i++){
    EEPROM.write(ubicacion+i, 255);
  }
  EEPROM.commit();
}

/*Función para leer la EEPROM*/
String leerEeprom(int posicion){
  byte caracter;
  String dato;
  for(int i = posicion; i < posicion+50; i++){
    caracter = EEPROM.read(i);
    if(caracter != 255){
      dato += (char)caracter;
    }
  }
  return dato;
}
```

/*

Función para configurar las credenciales de la red con wifi.

Se configura el ESP32 en modo AP con la dirección IP fija

Se genera un parpadeo de un led indicador para informar que el ESP32 entró en modo AP

Se verifica que el ESP32 halla obtenido la IP especificada. Cuando obtiene la IP se enciende el led y se habilita la red del ESP32 en modo AP

```
*/
void configuracionCredencialesWifi(){

  WiFi.mode(WIFI_AP);
  WiFi.softAPConfig(local_IP, gateway, subnet);

  Serial.println("Pulsador en 1");
  digitalWrite(led_wf, 1);
  delay(250);
  digitalWrite(led_wf, 0);
  delay(250);
  digitalWrite(led_wf, 1);
  delay(250);
  digitalWrite(led_wf, 0);
  delay(250);
  digitalWrite(led_wf, 1);
  delay(250);
  digitalWrite(led_wf, 0);

  int cont = 0;
  while(WiFi.softAPIP() != local_IP){
    Serial.print("cont es: ");
    Serial.println(cont);
    cont++;
    if(cont >= 20){
      Serial.println("reinicio ipconfig");
      WiFi.softAPConfig(local_IP, gateway, subnet);
      cont = 0;
    }
  }
```

```
  digitalWrite(led_wf, 1);
  WiFi.softAP(ssid_esp32, pass_esp32);

  Serial.println();
  Serial.print("IP address: ");
  Serial.println(WiFi.softAPIP());

  /*Función que entrega la página inicial de configuración de credenciales wifi*/
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *peticion){
    peticion->send(200, "text/html", HTML);
  });

/*
Función que entrega la página indicando que las credenciales se ingresaron correctamente y obtiene el nombre y contraseña ingresados por el usuario y se graba en la EEPROM.
*/
  server.on("/captura_info", [](AsyncWebServerRequest *peticion){

    int cant_parametros = peticion->params();

    for(int i=0; i<cant_parametros; i++){
      AsyncWebParameter* parametro = peticion->getParam(i);
      if(parametro->name() == "ssid"){
        grabarEeprom(0, parametro->value());
      }
      else{
        grabarEeprom(50, parametro->value());
      }
      Serial.print("parametro es: ");
      Serial.println(parametro->value());
```

```
    }

    peticion->send(200, "text/html", HTML1);
  });

  server.begin();
  while (true);
}


/* CÓDIGO DE CONFIGURACIÓN */
void setup() {
  EEPROM.begin(512);                /*Se inicia la EEPROM*/
/*
Se define el modo de funcionamiento (salida o entrada) de los pines que se usaran en el ESP32, para los pines donde se conectan los interruptores de nivel se habilitan las resistencias de pulldown.
*/
  pinMode(SW0, INPUT_PULLDOWN);
  pinMode(SW1, INPUT_PULLDOWN);
  pinMode(SW2, INPUT_PULLDOWN);
  pinMode(SW3, INPUT_PULLDOWN);
  pinMode(SW4, INPUT_PULLDOWN);
  pinMode(EV, OUTPUT);
  pinMode(MT0, OUTPUT);
  pinMode(MT1, OUTPUT);
  pinMode(L, OUTPUT);
  pinMode(alerta, OUTPUT);
  pinMode(disparo, OUTPUT);
  pinMode(eco, INPUT);
  pinMode(led_wf, OUTPUT);
  pinMode(led_fb, OUTPUT);
```

```
  pinMode(pul_config, INPUT_PULLDOWN);


  /*Se inicia en bajo todos los pines de salida*/
  digitalWrite(EV, 0);
  digitalWrite(MT0, 0);
  digitalWrite(MT1, 0);
  digitalWrite(L, 0);
  digitalWrite(alerta, 0);
  digitalWrite(disparo, 0);
  digitalWrite(led_wf, 0);
  digitalWrite(led_fb, 0);

  Serial.begin(115200);                 /*Se inicia un puerto serial*/

  if(digitalRead(pul_config)){           /*Si el pulsador está sostenido se inicia el*/
    configuracionCredencialesWifi();     /*proceso de configuración de credenciales
wifi*/
  }

  /*Se verifica la longitud del nombre de red y luego se lee la EEPROM*/
  long_ssid = leerEeprom(0).length();
  Serial.print("longitud de ssid es: ");
  Serial.println(long_ssid);

  leerEeprom(0).toCharArray(ssid_wf, 50);
  leerEeprom(50).toCharArray(pass_wf, 50);

/*Si la longitud es 0, entonces no existen credenciales de wifi, de lo contrario si
existen*/
  if(!long_ssid){
```

```
    Serial.println("Sin credenciales de red");
    red_no_encontrada = 1;
    error_conex_fb = 1;
  }
  else{
/*
Se inicia la conexión wifi a la red definida y mientras se establece conexión se indica en el monitor serie que se está realizando la conexión.
Se intenta conectar a wifi por máximo 40s, mientras se establece conexión se enciende y apaga el led indicador wifi(led_wf), si no hay conexión se apaga el led indicador.
*/
    if(buscarRed()){
      WiFi.begin(ssid_wf, pass_wf);
      Serial.println("connecting");
      unsigned long millis_wifi = millis();
      while(WiFi.status() != WL_CONNECTED){
        toggle = !toggle;
        digitalWrite(led_wf, toggle);
        delay(500);

        if((millis() - millis_wifi) >= 40000){
          red_no_encontrada = 1;
          error_conex_fb = 1;
          return;
        }
      }

/*Si el ESP32 está conectado se establece conexión con Firebase, de lo contrario es porque no se encuentra la red guardada en la EEPROM*/
      if(WiFi.status() == WL_CONNECTED){
```

```
      red_no_encontrada = 0;
      digitalWrite(led_wf, 1);
      Firebase.begin(FIREBASE_HOST);  /*Se inicia la conexión con el proyecto en Firebase*/
      delay(1000);

/*
 Se lee el campo a(estado_agua) del nodo data de Firebase, el cual indica si el agua de T2 ya fue tratada, para que cada vez que el sistema se reinicie no se vuelva encender la lámpara UV innecesariamente.

Se reinicia los valores de config en Firebase.
*/
      estado_agua = Firebase.getBool("1801260600/data/a");
      verif_error_fb("get estado_agua setup");

      enviarConfigJsonFirebase();
      if(!verif_error_fb("Inicio config")){
        firebaseStream();
        stream_existe = 1;
      }
      Serial.println("fin ip obtenida");
    }
  }
  else{
    Serial.println("No se encontró la red al inicio");
    red_no_encontrada = 1;
    error_conex_fb = 1;
  }

  /*Se activa la detección de eventos del módulo wifi*/
```

```
    WiFi.onEvent(eventosWifi);

    /*Se crea el arreglo var_data_ant*/
    var_data_ant[0] = digitalRead(SW0);
    var_data_ant[1] = digitalRead(SW1);
    var_data_ant[2] = digitalRead(SW2);
    var_data_ant[3] = digitalRead(SW3);
    var_data_ant[4] = digitalRead(SW4);
    var_data_ant[5] = 0;
    var_data_ant[6] = 0;
    var_data_ant[7] = 0;
    var_data_ant[8] = 0;
    var_data_ant[9] = estado_agua;
  }

  /*Se realiza la primera medición del SRF05*/
  dist_SRF05 = cal_dist_srf05(disparo, eco);
  dist_SRF05_ant = dist_SRF05;
}

/* CÓDIGO PRINCIPAL */
void loop() {
/*Si la longitud del nombre de la red es diferente de cero*/
  if(long_ssid != 0){

/*Cuando la red no se ha encontrado se realiza la busqueda cada 10s*/
    if(red_no_encontrada && (abs(millis() - ms_buscar_red) >= 10000)){
      if(buscarRed()){
        Serial.println("regresó que se encontró red");
        WiFi.begin(ssid_wf, pass_wf);
        red_no_encontrada = 0;
```

```
    }
  }

/*Cada vez que se asigna una dirección IP al ESP32 se reestablece la conexión con Firebase. Esta es una reconexión inmediata*/
  if(ip_obtenida && reconx_inmediata){
    Firebase.begin(FIREBASE_HOST);              /*Se inicia la conexión con el proyecto en Firebase*/
    delay(1000);
    enviarConfigJsonFirebase();
    if(!verif_error_fb("primera Reconexión Firebase")){
      firebaseStream();
      stream_existe = 1;
      ip_obtenida = 0;
    }
    else{
      reconx_inmediata = 0;
      ms_reconx_fb = millis();
    }
  }

/*Si no se pudo conectar a Firebase en la reconexión inmediata, se realiza el proceso cada 30s*/
  if(ip_obtenida && !reconx_inmediata && ((millis() - ms_reconx_fb) >= 30000)){
    Firebase.begin(FIREBASE_HOST);              /*Se inicia la conexión con el proyecto en Firebase*/
    delay(1000);
    enviarConfigJsonFirebase();
    if(!verif_error_fb("Reconexión Firebase posterior")){
      firebaseStream();
      stream_existe = 1;
```

```
      reconx_inmediata = 1;
      ip_obtenida = 0;
    }
    else{
      ms_reconx_fb = millis();
    }
  }

/*Cuando el modulo wifi se desconecta se indica con un parpadeo de 500ms en el led wifi*/
  if(_evento == SYSTEM_EVENT_STA_DISCONNECTED){
    toggle_led(led_wf, 500);
  }

/*Cuando el modulo wifi se conecta y está esperando que el modem le asigne una ip, se indica con un parpadeo de 250ms en el led wifi*/
  if(_evento == SYSTEM_EVENT_STA_CONNECTED){
    toggle_led(led_wf, 250);
  }

/*Cuando el modulo wifi obtiene la IP y se está conectando a Firebase, se indica con un parpadeo de 500ms en el led Firebase*/
  if(ip_obtenida){
    toggle_led(led_fb, 500);
  }
 }

/*Lectura de todas las variables. Las MT y la EV no se leen porque no se utilizan para tomar decisiones*/
 sw0 = digitalRead(SW0);
 sw1 = digitalRead(SW1);
```

```
  sw2 = digitalRead(SW2);
  sw3 = digitalRead(SW3);
  sw4 = digitalRead(SW4);
  l = digitalRead(L);

/*
Verificación del correcto estado de los sw de T1. Si sw2 está On y sw1 está OFF se genera una alerta ya que algo debe estar fallando en los sw. Por seguridad se apaga la EV, las MT's y la L, se genera la alarma (error 1) y se envía los datos al nodo data de Firebase.
Permanece en un ciclo while esperando a que los sw funcionen correctamente o a ser reiniciado por la app. También se puede reiniciar por hardware.
*/
  if(!sw1 && sw2){
    digitalWrite(alerta, 1);
    digitalWrite(EV, 0);
    digitalWrite(MT0, 0);
    digitalWrite(MT1, 0);
    digitalWrite(L, 0);
    if(!error_conex_fb){
      error = 1;
      ev = 0;
      mt0 = 0;
      mt1 = 0;
      l = 0;
      enviarDataJsonFirebase();
    }

    if(WiFi.status() != WL_CONNECTED){
      digitalWrite(led_wf, 0);
    }
```

```
    while(!sw1 && sw2){
      Serial.println("Error en T1. Verifique Switch");
      sw1 = digitalRead(SW1);
      sw2 = digitalRead(SW2);
      if(!(!sw1 && sw2)){
        digitalWrite(alerta, 0);
        error = 0;
        if(!error_conex_fb){
          enviarDataJsonFirebase();
        }
        return;
      }
    }
  }

/*
Verificación del correcto estado de los sw de T1. Si sw4 está On y sw3 está OFF se genera una alerta ya que algo debe estar fallando en los sw. Por seguridad se apaga la EV y las MT's y la L, se genera la alarma (error 3) y se envía los datos al nodo data de Firebase.
Permanece en un ciclo while esperando a que los sw funcionen correctamente o esperando a ser reiniciado por la app. También se puede reiniciar por hardware.
 */
  if(!sw3 && sw4){
    digitalWrite(alerta, 1);
    digitalWrite(EV, 0);
    digitalWrite(MT0, 0);
    digitalWrite(MT1, 0);
    digitalWrite(L, 0);
    if(!error_conex_fb){
```

```
    error = 2;
    ev = 0;
    mt0 = 0;
    mt1 = 0;
    l = 0;
    enviarDataJsonFirebase();
  }

  if(WiFi.status() != WL_CONNECTED){
    digitalWrite(led_wf, 0);
  }

  while(!sw3 && sw4){
    Serial.println("Error en T2. Verifique Switch");
    sw3 = digitalRead(SW3);
    sw4 = digitalRead(SW4);
    if(!(!sw3 && sw4)) {
      digitalWrite(alerta, 0);
      error = 0;
      if(!error_conex_fb){
        enviarDataJsonFirebase();
      }
      return;
    }
  }
 }

/*
Control de encendido y apagado de la electroválvula(EV).
On: cuando sw0==0 (agua por encima de sw0) y sw2==0 (T1 no está lleno) y
EV_hab==1 (EV habilitada).
```

```
OFF: cuando no se cumple alguna de las condiciones anteriores.
*/
  if(!sw0 && !sw2 && EV_hab){
    digitalWrite(EV, 1);
    ev = 1;
  }
  else{
    digitalWrite(EV, 0);
    ev = 0;
  }

/*
Control de encendido y apagado de la electrobomba MT0.
On: cuando estado_agua==0 (agua en T2 sucia) y L==0 (lámpara UV apagada) y
sw2==1 (nivel de T1 por encima de sw2) y sw4==0 (T2 desocupado) y
MT0_hab==1 (MT0 habilitada).
OFF: cuando estado_agua==1 (agua en T2 limpia) o L==1 (lámpara UV
encendida) o sw4==1 (T2 lleno) o MT0_hab==0 (MT0 deshabilitada) o sw1==0 (T1
desocupado).
*/
  if(!estado_agua && !l && sw2 && !sw4 && MT0_hab){
    digitalWrite(MT0, 1);
    mt0 = 1;
  }
  else if (estado_agua || l || sw4 || !MT0_hab || !sw1){
    digitalWrite(MT0, 0);
    mt0 = 0;
  }

/*
Control de encendido y apagado de L.
```

```
On: estado_agua==0 (agua en T2 sucia) y sw4==1 (T2 lleno) y L_hab==1 (Lámpara habilitada).
La lámpara permanece en ON por 30s.
OFF: cuando ha pasado 30 ms o más después de ser encendida.
*/
  if(!estado_agua && sw4 && L_hab){
    if(!l){
      digitalWrite(MT0, 0);
      mt0 = 0;
      digitalWrite(L, 1);
      l = 1;                /*ele(l) = uno(1)*/
      ms_ant_L = millis();
    }
  }

  if(l && (abs(millis() - ms_ant_L) >= 30000)){
    digitalWrite(L, 0);
    l = 0;                /*letra ele(l) = 0*/
    estado_agua = 1;
  }

/*
Cuando nivel de T2 desciende por debajo de sw3, se debe cambiar estado_agua a 0 para indicar que el agua limpia se acabó y ahora puede empezar a llenarse con agua sucia.
*/
  if (!sw3) estado_agua = 0;

/*
Medición de distancia con el SRF05.
Si ha transcurrido 200ms o más desde la última medición se realiza una nueva
```

Si la diferencia de la medición en el ciclo anterior con la medición actual es superior a 2cm, no se tiene en cuenta la medición actual, porque es una medida incorrecta, pero si esta diferencia es constante por 5 mediciones consecutivas, se tiene en cuenta la medición actual.
*/

```
  if (abs(millis() - ms_ant_SRF05) >= 200){
    dist_SRF05 = cal_dist_srf05(disparo, eco);
    if(abs(dist_SRF05 - dist_SRF05_ant) > 2){
      cont_dist++;
      Serial.print("cont_dis es: ");
      Serial.println(cont_dist);
      if(cont_dist >= 5){
        cont_dist = 0;
        dist_SRF05_ant = dist_SRF05;
        return;
      }
      dist_SRF05 = dist_SRF05_ant;
    }
    else{
      cont_dist = 0;
      dist_SRF05_ant = dist_SRF05;
    }
    Serial.print("Distancia medida es: ");
    Serial.println(dist_SRF05);
```

/*
Se verifica que la medición no sea mayor a la distancia entre el SFR05 y el fondo de T2 (dist_SRF05_Fondo_T3 + 2) o menor a la mínima permitida (dist_min_SRF05 - 2). Si esto es verdadero entonces se incrementa el contador de errores de medición, de tal manera que, si ocurre más de 5 mediciones incorrectas

consecutivas, se detiene el sistema, entrando en un ciclo infinito, para que se realice la verificación del hardware. Sale del ciclo infinito con un reset.

Luego de realizar la medición se almacena el valor de millis en ms_ant_SRF05, para poder determinar cuando pasen 20ms y así poder iniciar una nueva medición */

```
  if((dist_SRF05 >= dist_SRF05_Fondo_T3 + 2) || (dist_SRF05 <= dist_min_SRF05 - 2)){
    cont_error_SRF05++;
    if(cont_error_SRF05 >= 5){
      error = 3;
      digitalWrite(alerta, 1);
      digitalWrite(EV, 0);
      digitalWrite(MT0, 0);
      digitalWrite(MT1, 0);
      digitalWrite(L, 0);

      if(!error_conex_fb){
        ev = 0;
        mt0 = 0;
        mt1 = 0;
        l = 0;
        enviarDataJsonFirebase();
      }

      if(WiFi.status() != WL_CONNECTED){
        digitalWrite(led_wf, 0);
      }

      while(true){
        Serial.println("Lect incorrecta en tanque T3");
```

```
          Serial.print("Distancia medida es: ");
          Serial.println(dist_SRF05);
          delay(40);
          dist_SRF05 = cal_dist_srf05(disparo, eco);
        }
       }
     }
     else cont_error_SRF05 = 0;

     ms_ant_SRF05 = millis();
  }

/*
Control de encendido y apagado de MT1.
On: estado_agua==1 (agua en T2 limpia) y dist_medida_SFR05_cm>=20 (solo cuando hay, como mínimo, 20cm de espacio en T2 se puede encender MT1, esto para evitar que MT1 se esté encendiendo y apagando constantemente) y MT1_hab==1 (MT1 habilitada).
OFF: estado_agua=0 (agua en T2 sucia) o dist_cm<=8cm (T2 lleno) o MT1_hab==0 (MT1 deshabilitada).
*/
  if(estado_agua && dist_SRF05 >= 20 && MT1_hab ){
    digitalWrite(MT1, 1);
    mt1 = 1;
  }
  else if(!estado_agua || dist_SRF05 <= dist_min_SRF05 || !MT1_hab){
    digitalWrite(MT1, 0);
    mt1 = 0;
  }

/*
```

```
Envío de objeto json al nodo data de Firebase.
Si ha transcurrido 3s o más desde la última vez que se envió el objeto y existe un
cambio en la data, se envía de nuevo.

Luego de realizar el envío se almacena el valor de millis en ms_ant_fb, para poder
determinar cuando pasen 3s y así poder realizar un nuevo envío del objeto.
*/

  if (!error_conex_fb && (abs(millis() - ms_ant_fb) >= 3000)){
    float rest_dist_firebase = abs(dist_SRF05 - dist_ant_fb);

    var_data_act[0] = sw0;
    var_data_act[1] = sw1;
    var_data_act[2] = sw2;
    var_data_act[3] = sw3;
    var_data_act[4] = sw4;
    var_data_act[5] = ev;
    var_data_act[6] = mt0;
    var_data_act[7] = mt1;
    var_data_act[8] = l;            /*Letra ele(l)*/
    var_data_act[9] = estado_agua;

    for(int i = 0; i<=8; i++){
      if((var_data_act[i] != var_data_ant[i]) || (rest_dist_firebase >= 2)){

        enviarDataJsonFirebase();

        var_data_ant[0] = sw0;
        var_data_ant[1] = sw1;
        var_data_ant[2] = sw2;
        var_data_ant[3] = sw3;
```

```
        var_data_ant[4] = sw4;
        var_data_ant[5] = ev;
        var_data_ant[6] = mt0;
        var_data_ant[7] = mt1;
        var_data_ant[8] = l;
        var_data_ant[9] = estado_agua;

        dist_ant_fb = dist_SRF05;
        ms_ant_fb = millis();
        return;
      }
    }
  }
}
```

9.2 ANEXO 2. **Código fuente de la aplicación AQUA VIVA**

Código credenciales Firebase: son las credenciales del proyecto en Firebase para vincularlos a la app.

```
export const firebaseConfig = {
  apiKey: "AIzaSyCgZFn9Cfzw3UkVwh-9nEBovjHAfES6034",
  authDomain: "reciclaje-agua-vivienda.firebaseapp.com",
  databaseURL: "https://reciclaje-agua-vivienda.firebaseio.com",
  projectId: "reciclaje-agua-vivienda",
  storageBucket: "reciclaje-agua-vivienda.appspot.com",
  messagingSenderId: "1048238480554"
};
```

Componente popover: es el que se usa para poder modificar el nombre de un sistema o eliminarlo.

Plantilla TypeScript:

```
import { Component, OnInit } from '@angular/core';
import { AlertController, NavParams, ViewController, ToastController } from
'ionic-angular'

//Interfaces
import { SistemaCN } from '../../interfaces/intefaces';

//Servicios
import { DbSistUsuariosProvider } from '../../providers/db-sist-usuarios/db-
sist-usuarios';
import { Subscription } from 'rxjs';

@Component({
  selector: 'popover',
  templateUrl: 'popover.html'
})
export class PopoverComponent implements OnInit{

  sistemasU:SistemaCN[];
  sistema:SistemaCN;
  sus_popover:Subscription
  key_nodo:string;

  constructor(
    private alertCtrl:AlertController,
    private navParams: NavParams,
    private viewCtrl: ViewController,
```

```
  private _dbSistemasU: DbSistUsuariosProvider,
  private toastCtrl: ToastController
) { }

ngOnInit(){
  this.sistema = this.navParams.get('sistema');
}

modificarNombre(){
  this.viewCtrl.dismiss();
  this.mostrarConfrmActualizar();
}

eliminarSistema(){
  this.viewCtrl.dismiss();
  this.mostrarConfirmEliminar();
}

mostrarConfrmActualizar() {
  this.alertCtrl.create({
    title: 'Modificar nombre',
    message: `Ingrese el nuevo nombre para este sistema. Nombre actual:
      "${this.sistema.nombre}"`,
    inputs: [
      {
        name: 'nombre',
        placeholder: 'Nuevo nombre'
      },
    ],
    buttons: [
      {
        text: 'Cancelar'
      },
      {
        text: 'Guardar',
        handler: data => {
          if (data.nombre == '') {
            this.mostrarToast();
            return false;
          } else {
            this.obtenerKeySisSel().then(key=>{
              this._dbSistemasU.ActualizarSistema(key, data).then(_=>{
                this.viewCtrl.dismiss();
              });
            });
```

```
          }
        }
      }
    ]
  }).present();
}

mostrarConfirmEliminar() {
  this.alertCtrl.create({
    title: '¿Desea eliminar el sistema?',
    message: `Está seguro de eliminar el sistema
"${this.sistema.nombre}"`,
    buttons: [
      {
        text: 'SI',
        handler: () => {
          this.obtenerKeySisSel().then(key=>{
            this._dbSistemasU.eliminarSistema(key).then(_=>{
              this.viewCtrl.dismiss();
            }).catch(error=>error);
          });
        }
      },
      {
        text: 'NO'
      }
    ]
  }).present();
}

obtenerKeySisSel(){
  return new Promise((resolve,reject)=>{
    this.sistemasU = this.navParams.get('sistemas_usuario_obj');
    resolve();
  }).then(_=>{
    for (let key in this.sistemasU) {
      if (this.sistemasU[key].clave === this.sistema.clave) {
        return key;
      }
    }
  }).catch(error=>error);
}

mostrarToast(){
  this.toastCtrl.create({
```

```
      message: 'Ingrese el nuevo nombre para el sistema',
      duration: 3000
    }).present();
  }
}
```

Plantilla HTML5:

```
<ion-list >
  <ion-list-header>{{sistema.nombre}} - {{sistema.clave}}</ion-list-header>
  <button class="lista" ion-item (click)="modificarNombre()">
    Modificar nombre
  </button>

  <button class="lista" ion-item (click)="eliminarSistema()">
    Eliminar sistema
  </button>
</ion-list>
```

Plantilla CSS:

```
popover {
  .lista{
    color: color($colors, primary);
  }
}
```

Componente toolbar-actuadores: es la barra que se encuentra en todas las páginas de los tanques y que contiene el estado de los actuadores.

Plantilla TypeScript:

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

import { NavParams} from 'ionic-angular';
import { Subscription } from 'rxjs/Subscription';

//Servicio
import { DbSistemasProvider } from '../../providers/index.servicios';

//Interfaces
import { Data, Sistema } from '../../interfaces/intefaces'
```

```
@Component({
  selector: 'toolbar-actuadores',
  templateUrl: 'toolbar-actuadores.html'
})
export class ToolbarActuadoresComponent implements OnInit, OnDestroy {

  sistema:Sistema;
  data_sistema:Data;
  sus_tool_actuadores:Subscription;

  constructor(
    private navParams: NavParams,
    private _dbSistemas: DbSistemasProvider
  ) {}

  ngOnInit(){
    //Se obtiene la data actual del sistema
    this.sistema = this.navParams.get('sistema');
    this.data_sistema = this.sistema.data;
    //Nos subcribimos a la referencia de la BD en el servicio para saber
cuando ocurran
    //cambios en la BD
    this.sus_tool_actuadores =
this._dbSistemas.Sistema.subscribe((sistema:any)=>{
      //Cada vez que cambie la data del sistema se obtiene la información de
      //los actuadores
      this.data_sistema = sistema.data
    });
  }

  ngOnDestroy(){
    //Se cancela la suscripción
    this.sus_tool_actuadores.unsubscribe();
  }
}
```

Plantilla HTML5:

```
<ion-toolbar

  color='primary'
  text-center>
  <ion-grid >
    <ion-row icon-only>

      <ion-col *ngIf='!data_sistema.EV'
```

```
  icon-only
  class="col_on">
  EV
  <ion-icon name="ios-bulb" color='dark' class="icon"></ion-icon>
</ion-col>
<ion-col *ngIf='data_sistema.EV'
  icon-only
  class="col_off">
  EV
  <ion-icon name="ios-bulb" color='danger' class="icon"></ion-icon>
</ion-col>

<ion-col *ngIf='!data_sistema.L'
  icon-only
  class="col_on">
  L
  <ion-icon name="ios-bulb" color='dark' class="icon"></ion-icon>
</ion-col>
<ion-col *ngIf='data_sistema.L'
  icon-only
  class="col_off">
  L
  <ion-icon name="ios-bulb" color='danger' class="icon"></ion-icon>
</ion-col>

<ion-col *ngIf='!data_sistema.M0'
  icon-only
  class="col_on">
  MT0
  <ion-icon name="md-nuclear" color='dark' class="icon"></ion-icon>
</ion-col>
<ion-col *ngIf='data_sistema.M0'
  icon-only
  class="col_off">
  MT0
  <ion-icon name="md-nuclear" color='danger' class="icon"></ion-icon>
</ion-col>

<ion-col *ngIf='!data_sistema.M1'
  icon-only
  class="col_on">
  MT1
  <ion-icon name="md-nuclear" color='dark' class="icon"></ion-icon>
</ion-col>
<ion-col *ngIf='data_sistema.M1'
```

```
        icon-only
        class="col_off">
        MT1
        <ion-icon name="md-nuclear" color='danger' class="icon"></ion-icon>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-toolbar>
```

Plantilla CSS:

```
toolbar-actuadores {
  .grid{
    padding: 0;
  }

  .col_on{
    padding: 0;
    color: color($colors, dark);
  }

  .col_off{
    padding: 0;
    color: color($colors, danger);
  }

  .icon{
    display: block;
    padding-top: 0.5rem;
  }
}
```

Código del servicio de autenticación: solo contiene una plantilla TypeScript

```
import { Injectable } from '@angular/core';
import { AlertController, ToastController } from 'ionic-angular';

//Firebase
import { AngularFireAuth } from 'angularfire2/auth';
import * as firebase from 'firebase/app';
import 'rxjs/add/operator/map';


@Injectable()
export class AutenticacionProvider {
```

```
  constructor(
    private afAuth:  AngularFireAuth,
    private alertCtrl: AlertController,
    private toastCtrl: ToastController
  ) { }

  registerUser(email:string, password:string){
    return new Promise( (resolve, reject)=>{
      this.afAuth.auth.createUserWithEmailAndPassword(email, password)
      .then(userData => {
        //Guardo el uid en realtime database
        resolve(userData);
      })
      .catch((err)=>{
        switch (err.code) {
          case 'auth/email-already-in-use':
            this.showInformativeAlert(
              'El correo electrónico ya existe',
              'El correo electrónico está siendo utilizado por otra cuenta.
Por\
               favor cierre este mensaje e inicie sesión');
          break;

          case 'auth/invalid-email':
            this.presentToast('Registre un correo electrónico valido');
          break;

          case 'auth/weak-password':
              this.presentToast('La contraseña debe tener almenos 6
caracteres');
          break;

          default:
          break;
        }
        reject(err);
      });
    });
  }

  loginUser(email:string, password:string){
    return new Promise( (resolve, reject)=>{
      this.afAuth.auth.signInWithEmailAndPassword(email, password)
      .then((userData) => {
        resolve(userData);
```

```
      })
      .catch((err)=>{
        if (email === '') {
          this.presentToast('Por favor ingrese un correo electrónico.');
          return;
        }

        switch (err.code) {
          case 'auth/invalid-email':
            this.presentToast('Ingrese un correo electrónico valido.');
          break;

          case 'auth/user-disabled':
            this.showAlertAccountDisabled();
          break;

          case 'auth/user-not-found':
              this.showInformativeAlert(
                'Correo electrónico no encontrado',
                'El correo electrónico ingresado nunca ha estado asociado a
una cuenta,\
                si desea registrarse, cierre este mensaje y vaya a la
sección\
                "Registarse".'
              )
          break;

          case 'auth/wrong-password':
            this.presentToast('Contraseña invalida');
          break

          default:
          break;
        }
        reject(err);
      })
    });
  }

  logoutUser(){
    return new Promise( (resolve, reject)=>{
      this.afAuth.auth.signOut()
      .then(() => {
        resolve();
      })
```

```
    .catch(error=>error)
  });
}

getStateAuthUser(){
  return this.afAuth.authState.map(auth => auth);
}

obtenerUsuario(){
  return new Promise((resolve, reject)=>{
    resolve(this.afAuth.auth.currentUser);
  });
}


sendEmailVerifyAccount(){
  return new Promise((resolve, reject)=>{
    this.afAuth.auth.currentUser.sendEmailVerification()
    .then(()=> {
      resolve();
    })
    .catch(error=>error)
  });
}

sendEmailResetPassword(email:string){
  return new Promise((resolve, reject)=>{
    this.afAuth.auth.sendPasswordResetEmail(email)
    .then(()=>{
      resolve();
    })
    .catch((err)=>{
      if (email === '') {
        this.presentToast('Por favor ingrese un correo electrónico.');
        return;
      }

      switch (err.code) {
        case 'auth/invalid-email':
          this.presentToast('Ingrese un correo electrónico valido.');
        break;

        case 'auth/user-not-found':
            this.showInformativeAlert(
              'Correo electrónico no encontrado',
```

```
                'El correo electrónico ingresado nunca ha estado asociado a
una cuenta,\
                si desea registrarse, cierre este mensaje y seleccione la\
                opción "No me he registrado aún".'
              )
            break;

            default:
            break;
          };
          reject(err);
        });
      });
    }

    private showInformativeAlert(title:string, subTitle:string) {
      this.alertCtrl.create({
        title: title,
        subTitle: subTitle,
        buttons: ['Entendido']
      }).present();
    }

    private showAlertAccountDisabled() {
      this.alertCtrl.create({
        title: 'Cuenta deshabilitada',
        subTitle: 'Parece que su cuenta ha sido deshabilitada.',
        buttons: ['Cerrar']
      }).present();
    }

    private presentToast(message:string) {
      this.toastCtrl.create({
        message: message,
        duration: 3000,
        position: 'top',
        dismissOnPageChange: true
      }).present();
    }
}
```

Código del servicio de claves del sistema: solo tiene una plantilla TypeScript

```
import { Injectable } from '@angular/core';
```

```
import { AngularFireDatabase, AngularFireList } from
'angularfire2/database';
import { Observable } from 'rxjs/Observable';

@Injectable()
export class DbClavesProvider {

  private dbClavesRef: AngularFireList<any>;
  private claves: Observable<any[]>;

  constructor(private afDB: AngularFireDatabase) {
    this.dbClavesRef = this.afDB.list('claves-sistemas');
    this.claves = this.dbClavesRef.valueChanges();
  }

  //Metodo que regresa un arreglo con las claves del sistema
  ConsultarClaves(){
    return new Promise((resolve, reject)=>{
      this.claves.subscribe(claves=>resolve(claves));
    });
  }
}
```

Código del servicio de los sistemas relacionados al usuarios: solo tiene una plantilla TypeScript.

```
import { Injectable} from '@angular/core';
import { AngularFireDatabase, AngularFireObject } from
'angularfire2/database';
import { Observable } from 'rxjs/Observable';

import { AutenticacionProvider } from '../index.servicios'
import { User } from 'firebase/app';
import { SistemaCN } from '../../interfaces/intefaces';


@Injectable()
export class DbSistUsuariosProvider{

  private dbSistemasUsuariosRef:AngularFireObject<any>;
  sistemasUsuariosC:Observable<any>
  uidUsuario:string;
  sistemas_actuales;
```

```
  constructor(
    private afDB: AngularFireDatabase,
    private _aut: AutenticacionProvider
  ) {       }

  //
  iniciarRef(){
    return new Promise((resolve, reject)=>{
      this.obtenerUidUsuario()
          .then(uidUsuario=>{
            this.dbSistemasUsuariosRef = this.afDB.object('sistemas-
usuarios/'+this.uidUsuario);
            this.sistemasUsuariosC =
this.dbSistemasUsuariosRef.snapshotChanges();

            resolve();
          })
          .catch(error=>error)
    });
  }

  //Devuelve el uid del uausrio logeado
  private obtenerUidUsuario(){
    return new Promise((resolve, reject)=>{
      this._aut.obtenerUsuario()
        .then((usuario:User)=>{
          this.uidUsuario = usuario.uid;
          resolve(this.uidUsuario);
        })
        .catch(error=>error);
    });
  }

  guardarNuevoSistemaL(sistema:SistemaCN){
    return new Promise((resolve,reject)=>{
      this.afDB.list('sistemas-usuarios/'+this.uidUsuario).push(sistema);
      resolve()
    })
  }

  eliminarSistema(nodo){
    return new Promise((resolve, reject)=>{
      this.afDB.list('sistemas-
usuarios/'+this.uidUsuario+'/'+nodo).remove();
      resolve();
```

```
    })
  }

  ActualizarSistema(key:string, objeto:object){
    return new Promise((resolve, reject)=>{
      this.afDB.list('sistemas-usuarios/'+this.uidUsuario).update(key,
objeto);
      resolve();
    })
  }

  destruirRef(){
    this.dbSistemasUsuariosRef = null;
    this.sistemasUsuariosC = null;
  }
}
```

Código del servicio de los sistemas: solo tiene una plantilla TypeScript.

```
//Serivicio que dispone los nodos del sistema seleccionado por el usuario
logeado.
//Se dispone los nodos config y data como una lista cada uno
import { Injectable } from '@angular/core';
import { AngularFireDatabase, AngularFireObject } from
'angularfire2/database';
import { Observable, BehaviorSubject } from 'rxjs/Rx';

import { SistemaCN } from '../../interfaces/intefaces';

@Injectable()
export class DbSistemasProvider {

  private dbSistemaRef: AngularFireObject<any>;
  public Sistema: Observable<any[]>;
  private clave_sistema = new BehaviorSubject('');
  public clave$ = this.clave_sistema.asObservable();
  private clave:string;

  sistema_seleccionado: SistemaCN;

  constructor(private afDB: AngularFireDatabase) {
      this.clave$.subscribe((clave:string)=>{
        if (clave != '') {
          this.clave = clave;
```

```
          this.dbSistemaRef = this.afDB.object(clave);
          this.Sistema = this.dbSistemaRef.valueChanges();
        }
      })
  }

  sistemaSeleccionado(sistema: SistemaCN){
    this.sistema_seleccionado = sistema;
    if (this.sistema_seleccionado == null) {
      this.clave_sistema.next('');
    } else {
      this.clave_sistema.next(this.sistema_seleccionado.clave);
    }
  }

  updateData(objecto:object){
    this.afDB.list(this.clave).update('config', objecto);
  }

  setData(objecto:object){
    this.afDB.list(this.clave).set('config', objecto);
  }
}
```

Código de las interfaces usadas:

```
export interface SistemaCN{
  clave?:string,
  nombre?:string
}

export interface Config{
  EV?:boolean,
  L?:boolean,
  M0?:boolean,
  M1?:boolean,
  r?:boolean
}

export interface Data{
  EV?:boolean,
  L?:boolean,
  M0?:boolean,
  M1?:boolean,
```

```
  a?:boolean,
  e?:number,
  s?:number,
  0?:boolean,
  1?:boolean,
  2?:boolean,
  3?:boolean,
  4?:boolean
}

export interface Sistema{
  config:{
    EV?:boolean,
    L?:boolean,
    M0?:boolean,
    M1?:boolean,
    r?:boolean
  },

  data:{
    EV?:boolean,
    L?:boolean,
    M0?:boolean,
    M1?:boolean,
    a?:boolean,
    e?:number,
    s?:number,
    0?:boolean,
    1?:boolean,
    2?:boolean,
    3?:boolean,
    4?:boolean
  },

  T3:{
    altura?:number,
    radio?:number
  }
}
```

Código de app.module: donde se configura todo los módulos, páginas, servicios, etc, que se van a usar en las app.

```
import { BrowserModule } from '@angular/platform-browser';
import { ErrorHandler, NgModule } from '@angular/core';
import { IonicApp, IonicErrorHandler, IonicModule } from 'ionic-angular';
import { SplashScreen } from '@ionic-native/splash-screen';
import { StatusBar } from '@ionic-native/status-bar';

import { MyApp } from './app.component';

//Páginas
import {
  InicioPage,
  T0Page,
  T1Page,
  T2Page,
  T3Page,
  RegistrarUsuarioPage,
  LoginPage,
  RegistrarSistemaPage
} from '../pages/index.paginas';

//Componentes
import { ToolbarActuadoresComponent } from '../components/toolbar-
actuadores/toolbar-actuadores'
import { PopoverComponent } from '../components/popover/popover';

//Servicios
import {
  DbClavesProvider,
  DbSistemasProvider,
  DbSistUsuariosProvider
} from '../providers/index.servicios'


import { AutenticacionProvider } from
'../providers/autenticacion/autenticacion';

//config firebase
import { firebaseConfig } from '../config/firebase.config';

//Firebase
import { AngularFireModule } from 'angularfire2';
import {
  AngularFireDatabaseModule,
  AngularFireDatabase,
 } from 'angularfire2/database';
```

```
import { AngularFireAuthModule } from 'angularfire2/auth';

@NgModule({
  declarations: [
    MyApp,
    InicioPage,
    T0Page,
    T1Page,
    T2Page,
    T3Page,
    RegistrarUsuarioPage,
    LoginPage,
    RegistrarSistemaPage,
    ToolbarActuadoresComponent,
    PopoverComponent
  ],
  imports: [
    BrowserModule,
    IonicModule.forRoot(MyApp),
    AngularFireModule.initializeApp(firebaseConfig),
    AngularFireDatabaseModule,
    AngularFireAuthModule
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp,
    InicioPage,
    T0Page,
    T1Page,
    T2Page,
    T3Page,
    RegistrarUsuarioPage,
    LoginPage,
    RegistrarSistemaPage,
    ToolbarActuadoresComponent,
    PopoverComponent
  ],
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler},
    AutenticacionProvider,
    AngularFireDatabase,
    AutenticacionProvider,
    DbClavesProvider,
```

```
    DbSistemasProvider,
    DbSistUsuariosProvider
  ]
})
export class AppModule {}
```

Código principal o raíz (app):

Plantilla TypeScript:

```
import { Component, OnDestroy } from '@angular/core';
import { Platform, MenuController,
         AlertController, LoadingController,
         ToastController } from 'ionic-angular';
import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';
import { Subscription } from 'rxjs/Subscription';
import { Observable, BehaviorSubject } from 'rxjs/Rx';

//Servicios
import {
  AutenticacionProvider,
  DbSistemasProvider
 } from '../providers/index.servicios'

//Páginas
import { LoginPage, RegistrarSistemaPage } from '../pages/index.paginas';

//Interfaces
import { Config } from '../interfaces/intefaces';

@Component({
  templateUrl: 'app.html'
})
export class MyApp implements OnDestroy{

  rootPage:any;

  sus_aut:Subscription;
  sus_data:Subscription;
  sus_clave:Subscription;

  config_sistema:Config;

  reiniciando = new BehaviorSubject(false);
```

```
  reiniciando$ =this.reiniciando.asObservable();

  sistema_menu:boolean;

  constructor(
    private menuCtrl: MenuController,
    private alertCtrl:AlertController,
    platform: Platform,
    statusBar: StatusBar,
    splashScreen: SplashScreen,
    private _aut: AutenticacionProvider,
    private _dbSistema:DbSistemasProvider,
    private loadingCtrl:LoadingController,
    private toastCtrl: ToastController
  ) {
    platform.ready().then(() => {
      this.sus_aut = this._aut.getStateAuthUser().subscribe(auth=>{
        //Se verifica si auth existe
        if (auth) {
          //Como auth existe se verifica que el email esté verificado
          if (auth.emailVerified) {
            //Si el email está verificado va al RegistrarSistemaPage
            this.rootPage = RegistrarSistemaPage;
            //Se habilita el menú, pero sin las opciones para inhabilitar
los actuadores y la
            //opción de reiniciar
            this.menuCtrl.enable(true, 'menu');
            //Nos suscribimos al observable clave$
            this.sus_clave = this._dbSistema.clave$.subscribe(clave=>{
              if (clave) {
                //Si se ha seleccionado un sistema se muestra las opciones
de habilitar los
                //actuadores en el menú y nos suscribimos al observador del
sistema seleccionado
                this.sus_data =
this._dbSistema.Sistema.subscribe((data_sistema:any)=>{
                  this.config_sistema = data_sistema.config;
                  this.sistema_menu = true;
                  //Cada vez que cambia el nodo config se evalua si existe
una orden de
                  //de reinicio para poner en true la variable reiniciando
de tal manera que
                  //el observable reiniciando$ actue
                  if (!this.config_sistema.r) {
                    this.reiniciando.next(false);
```

```
                }
                else{
                  this.reiniciando.next(true);
                }
              });
            } else {
              //Si no se ha seleccionado un sistema no se muestra las
opciones de habilitar
              //los actuadores en el menú.
              //Si existe suscribción a la data nos desuscribimos
                if (this.sus_data) {
                  this.sus_data.unsubscribe();
                }
                this.sistema_menu = false;
            }
          });
        }
        else{
          //Si email no está verificado se inhabilita el menú, va a la
página login y
          //se realiza el logout
          this.menuCtrl.enable(false, 'menu');
          this.rootPage = LoginPage;
          this._aut.logoutUser();
        }
      }
      else {
        //Si auth no existe se inhabilita el menú, va a la página login
        this.menuCtrl.enable(false, 'menu');
        this.rootPage = LoginPage;
      }
    });

    statusBar.styleDefault();
    splashScreen.hide();
  });
}

//Metodo que deshabilita el actuador seleccionado
deshab(actuador:string){
  switch (actuador) {
    case 'EV':
    if (this.config_sistema.EV) {
      let titulo = `¿Quieres deshabilitar la Electroválvula?`;
```

```
        let mensaje = `Si deshabilitas la Electroválvula el control del
sistema no podrá
        encenderla hasta que vuelvas a activarla o reinicies el sistema.
¿Quieres deshabilitar
        la Electroválvula?`;
        this.alerta_actuador(titulo, mensaje, {EV: false,
                                               L: this.config_sistema.L,
                                               M0: this.config_sistema.M0,
                                               M1: this.config_sistema.M1,
                                               r: this.config_sistema.r});
      }
      else {
        let titulo = `¿Quieres habilitar la Electroválvula?`;
        let mensaje = `Habilita la Electroválvula para que el control del
sistema pueda
        encenderla ¿Quieres habilitar la Electroválvula?`;
        this.alerta_actuador(titulo, mensaje, {EV: true,
                                               L: this.config_sistema.L,
                                               M0: this.config_sistema.M0,
                                               M1: this.config_sistema.M1,
                                               r: this.config_sistema.r});
      }
      break;

      case 'L':
        if (this.config_sistema.L) {
          let titulo = `¿Quieres deshabilitar la Lámpara?`;
          let mensaje = `Si deshabilitas la Lámpara el control del sistema
no podrá encenderla hasta\
          que vuelvas a activarla o reinicies el sistema. ¿Quieres
deshabilitar la Lámpara?`;
          this.alerta_actuador(titulo, mensaje, {EV: this.config_sistema.EV,
                                                 L: false,
                                                 M0: this.config_sistema.M0,
                                                 M1: this.config_sistema.M1,
                                                 r: this.config_sistema.r});
        }
        else {
          let titulo = `¿Quieres habilitar la Lámpara?`;
          let mensaje = `Habilita la Lámpara para que el control del sistema
pueda encenderla\
          ¿Quieres habilitar la Lámpara?`;
          this.alerta_actuador(titulo, mensaje, {EV: this.config_sistema.EV,
                                                 L: true,
                                                 M0: this.config_sistema.M0,
```

```
                                                M1: this.config_sistema.M1,
                                                r: this.config_sistema.r});
        }
        break;

      case 'M0':
        if (this.config_sistema.M0) {
          let titulo = `¿Quieres deshabilitar la Electrobomba 0?`;
          let mensaje = `Si deshabilitas la Electrobomba 0 el control del
sistema no podrá encenderla hasta\
          que vuelvas a activarla o reinicies el sistema. ¿Quieres
deshabilitar la Electrobomba 0?`;
          this.alerta_actuador(titulo, mensaje, {EV: this.config_sistema.EV,
                                                L: this.config_sistema.L,
                                                M0: false,
                                                M1: this.config_sistema.M1,
                                                r: this.config_sistema.r});
        }
        else {
          let titulo = `¿Quieres habilitar la Electrobomba 0?`;
          let mensaje = `Habilita la Electrobomba 0 para que el control del
sistema pueda encenderla\
          ¿Quieres habilitar la Electrobomba 0?`;
          this.alerta_actuador(titulo, mensaje, {EV: this.config_sistema.EV,
                                                L: this.config_sistema.L,
                                                M0: true,
                                                M1: this.config_sistema.M1,
                                                r: this.config_sistema.r});
        }
        break;

        case 'M1':
        if (this.config_sistema.M1) {
          let titulo = `¿Quieres deshabilitar la Electrobomba 1?`;
          let mensaje = `Si deshabilitas la Electrobomba 1 el control del
sistema no podrá encenderla hasta\
          que vuelvas a activarla o reinicies el sistema. ¿Quieres
deshabilitar la Electrobomba 1?`;
          this.alerta_actuador(titulo, mensaje, {EV: this.config_sistema.EV,
                                                L: this.config_sistema.L,
                                                M0: this.config_sistema.M0,
                                                M1: false,
                                                r: this.config_sistema.r});
        }
        else {
```

```
          let titulo = `¿Quieres habilitar la Electrobomba 1?`;
          let mensaje = `Habilita la Electrobomba 1 para que el control del
sistema pueda encenderla\
          ¿Quieres habilitar la Electrobomba 1?`;
          this.alerta_actuador(titulo, mensaje, {EV: this.config_sistema.EV,
                                                 L: this.config_sistema.L,
                                                 M0: this.config_sistema.M0,
                                                 M1: true,
                                                 r: this.config_sistema.r});
        }
        break;

      default:
        break;
    }
  }

  //Metodo que actualiza el campo r(reiniciar) a true, para dar la orden al
ESP32 de
  //reiniciarse
  reiniciar(){
    this.menuCtrl.close();
    this._dbSistema.setData({EV: this.config_sistema.EV,
                             L: this.config_sistema.L,
                             M0: this.config_sistema.M0,
                             M1: this.config_sistema.M1,
                             r: true});
    this.loading();
  }

  //Se cierra el menú
  cerrarMenu(){
    this.menuCtrl.close();
  }

  //Metodo para cerrar sesión y desucribirse de la clave seleccionada
  cerrarSesion(){
    this._aut.logoutUser()
    .then(()=>{
      this.sus_clave.unsubscribe();
    })
    .catch(error=>error)
    this.menuCtrl.close();
  }
```

```
  //Metodo que genera la alerta cuando un actuador se va a habilitar o
dehabilitar
  alerta_actuador(titulo:string, mensaje, objeto:object) {
    return this.alertCtrl.create({
      title: titulo,
      message: mensaje,
      buttons: [
        {
          text: 'SI',
          handler: () => {
            this._dbSistema.setData(objeto);
          }
        },
        {
          text: 'NO'
        }
      ]
    }).present();
  }

  //Metodo que genera una alerta informativa basica
  alertaInformativa(titulo:string, subtitulo:string){
    this.alertCtrl.create({
      title: titulo,
      subTitle: subtitulo,
      buttons: ['OK']
    }).present();
  }

  //Metodo que muestra el loading cuando se está reiniciando el sistema
  //Este metodo espera hasta 20s la respuesta del ESP32
  loading(){
    let loading = this.loadingCtrl.create({
      content: `Se está reiniciando el sistema, esto debe tardar unos
segundos,
      por favor espere`
    })

    loading.present().then(_=>{
      let sus_reiniciando = this.reiniciando$.subscribe(valor=>{
        if (!valor) {
          loading.dismiss();
          clearTimeout(timer);
        };
      });
```

```
    }).catch(error=>error);

    let timer = setTimeout(()=>{
      if (this.config_sistema.r) {
        loading.dismiss();
        let titulo:string = 'No se reinició';
        let subtitulo:string = `No fue posible reiniciar el sistema, por
favor verifique que
        se encuentra encendido`
        this.alertaInformativa(titulo, subtitulo);
      }
    }, 20000);

    loading.onDidDismiss(()=>{
      if (!this.config_sistema.r) {
        this.mostrarToast();
      }
    });
  }

  //Metodo que muestra un toast informativo cuando el sistema se ha
reiniciado
  mostrarToast(){
    this.toastCtrl.create({
      message: 'Sistema reiniciado',
      duration: 3000
    }).present();
  }

  //Metodo del ciclo de vida del componente que se ejecuta cuando el
componente se destruye
  //Lo que hace es desuscribirse del observable de autenticación.
  ngOnDestroy(){
    this.sus_aut.unsubscribe();
  }
}
```

Plantilla HTML5:

```
<ion-menu id="menu" [content]="content">
    <ion-header class="menu" color='primary' menuToggle>
        <img src="./assets/imgs/menu.png" alt="">
    </ion-header>
    <ion-content>
```

```
<ion-list>
  <div *ngIf='sistema_menu'>
    <button *ngIf='config_sistema.EV'
      ion-item
      class="btn_hab"
      icon-left
      (click)="deshab('EV')">
      <ion-icon class="icon" name="ios-water"></ion-icon>
      Inhabilitar Electroválvula
    </button>
    <button
      ion-item *ngIf='!config_sistema.EV'
      class="btn_deshab"
      icon-left
      (click)="deshab('EV')">
      <ion-icon class="icon" name="ios-water"></ion-icon>
      Habilitar Electroválvula
    </button>

    <button *ngIf='config_sistema.L'
      ion-item
      class="btn_hab"
      icon-left
      (click)="deshab('L')">
      <ion-icon class="icon" name="ios-bulb"></ion-icon>
      Inhabilitar Lámpara
    </button>
    <button
      ion-item *ngIf='!config_sistema.L'
      class="btn_deshab"
      icon-left
      (click)="deshab('L')">
      <ion-icon class="icon" name="ios-bulb"></ion-icon>
      Habilitar Lámpara
    </button>

    <button *ngIf='config_sistema.M0'
      ion-item
      class="btn_hab"
      icon-left
      (click)="deshab('M0')">
      <ion-icon class="icon" name="md-nuclear"></ion-icon>
      Inhabilitar Electrobomba 0
    </button>
    <button *ngIf='!config_sistema.M0'
```

```
    ion-item
    class="btn_deshab"
    icon-left
    (click)="deshab('M0')">
    <ion-icon class="icon" name="md-nuclear"></ion-icon>
    Habilitar Electrobomba 0
  </button>

  <button *ngIf='config_sistema.M1'
    ion-item
    class="btn_hab"
    icon-left
    (click)="deshab('M1')">
    <ion-icon class="icon" name="md-nuclear"></ion-icon>
    Inhabilitar Electrobomba 1
  </button>
  <button *ngIf='!config_sistema.M1'
    ion-item
    class="btn_deshab"
    icon-left
    (click)="deshab('M1')">
    <ion-icon class="icon" name="md-nuclear"></ion-icon>
    Habilitar Electrobomba 1
  </button>


  <button
    ion-item
    class="btn_hab"
    icon-left
    (click)="reiniciar()">
    <ion-icon class="icon" name="ios-refresh-circle"></ion-icon>
    Reiniciar
  </button>
</div>

<button
  ion-item
  class="btn_hab"
  icon-left
  (click)="cerrarSesion()">
  <ion-icon class="icon" name="md-log-out"></ion-icon>
  Cerrar sesión
</button>
```

```
        <button
          ion-item
          class="btn_hab"
          icon-left
          (click)="cerrarMenu()">
          <ion-icon class="icon" name="md-close"></ion-icon>
          Cerrar menú
        </button>
      </ion-list>
    </ion-content>
  </ion-menu>

<ion-nav  #content [root]="rootPage" ></ion-nav>
```

Plantilla css

```
.btn_hab {
  color: color($colors, primary);
}

.btn_deshab {
  color: color($colors, danger);

}

.icon{
  padding: 1rem;
}

p {
  text-align: center;
}
```

Código de la página para registrar usuarios:

Plantilla TypeScript:

```
import { Component } from '@angular/core';
import {
  NavController,
  ToastController,
  AlertController
} from 'ionic-angular';
```

```
//Servicios
import { AutenticacionProvider } from
'../../providers/autenticacion/autenticacion'

//Páginas
import { LoginPage } from '../index.paginas'

@Component({
  selector: 'page-registrar-usuario',
  templateUrl: 'registrar-usuario.html',
})
export class RegistrarUsuarioPage {

  email_ingresado:string = '';
  contrasena1_ingresada:string = '';
  contrasena2_ingresada:string = '';
  tipo_campo_contrasena: string = 'password';
  icono_mostrar_contrasena: string = 'eye';

  constructor(
    private _auth: AutenticacionProvider,
    private toastCtrl: ToastController,
    private navCtrl: NavController,
    private alertCtrl: AlertController
  ) {
  }

  //Metodo que crea un nuevo usuario
  onEnviarAdicionarUsuario(){
    //Se verifica si escribió en el campo email
    if (this.email_ingresado === '') {
      this.presentarToast('Por favor ingrese un correo electrónico');
      return;
    }

    //Se verifica si las contraseñas coinciden
    if (this.contrasena1_ingresada !== this.contrasena2_ingresada) {
      this.presentarToast('Las contraseñas no coinciden');
      return;
    }

    //Se llama el metodo para registrar al usuario. Si es posible
registrarlo, el metodo
    //devuelve toda la data del usuario
```

```
    this._auth.registerUser(this.email_ingresado,
this.contrasena1_ingresada)
    .then((userData)=>{
      //Se envía el email con el link de confirmación de confirmación
      this._auth.sendEmailVerifyAccount()
        .then(_=>{
          //Se presenta una alerta indicando que se realizó el registro.
          let titulo:string = 'Registro realizado';
          let subtitulo:string = 'Se ha realizado el registro y se ha
enviado un link al correo\
          electrónico suministrado, el cual debe abrir para que podamos
vericarlo. Luego de abrir\
          el link por favor inicie sesión en esta app'
          this.alerta(titulo, subtitulo);
        })
        .catch(error=>error);
    })
    .catch(error=>error);
  }

  //Metodo que presenta un toast informativo
  presentarToast(mensaje:string) {
    this.toastCtrl.create({
      message: mensaje,
      duration: 3000,
      position: 'top'
    }).present();
  }

  //Metodo que cambia el tipo del campo contraseña de password a text y la
imagen del ojo
  //para mostrar o no la contraseña
  onMostrarContrasena(){
    if (this.tipo_campo_contrasena === 'password') {
      this.tipo_campo_contrasena = 'text';
      this.icono_mostrar_contrasena = 'eye-off';
    } else {
      this.tipo_campo_contrasena = 'password';
      this.icono_mostrar_contrasena = 'eye';
    }
  }

  //Metodo que muestra una alerta informativa
  alerta(titulo:string, subtitulo:string){
    return this.alertCtrl.create({
```

```
      title: titulo,
      subTitle: subtitulo,
      buttons: [
        {
          text: 'Entendido',
          handler: () => {
            this.navCtrl.push(LoginPage);
          }
        }
      ]
    }).present();
  }
}
```

Plantilla HTML5:

```
<ion-header>

    <ion-navbar color='primary'>
    <ion-title>RECICLAJE DE AGUA</ion-title>
    </ion-navbar>

  </ion-header>

<ion-content padding>

    <h1>REGISTRARSE</h1>
    <p class="titulo-parrafo">Ingrese un correo electrónico y una
      constraseña para registrarse</p>

    <form (ngSubmit)="onEnviarAdicionarUsuario()">

      <ion-item>
        <ion-label floating>Correo electrónico</ion-label>
        <ion-input type="email"
                    name='email'
                    [(ngModel)]='email_ingresado'>
        </ion-input>
      </ion-item>


      <ion-item>
        <ion-label floating>Contraseña</ion-label>
        <ion-input type="{{tipo_campo_contrasena}}"
```

```
                    name='contrasena1_ingresada'
                    [(ngModel)]='contrasena1_ingresada'>
        </ion-input>
        <button ion-button
                clear
                type="button"
                item-right
                icon-only
                (click)="onMostrarContrasena()">
          <ion-icon name="{{icono_mostrar_contrasena}}"> </ion-icon>
        </button>
      </ion-item>

      <ion-item>
        <ion-label floating>Repita la contraseña</ion-label>
        <ion-input type="{{tipo_campo_contrasena}}"
                    name='contrasena2_ingresada'
                    [(ngModel)]='contrasena2_ingresada'>
        </ion-input>
      </ion-item>

      <button ion-button
              block
              type="submit">
        Enviar
      </button>
    </form>

</ion-content>
```

Plantilla CSS:

```
page-registrar-usuario {
  h1{
    color: color($colors, primary);
    padding-top: 1rem;
    text-align: center;
  }

  .titulo-parrafo{
    color: color($colors, primary);
    font-size: 1.8rem;
    text-align: center;
  }
```

```
  .btn-siguiente{
    margin-top: 2rem;
  }

  .btn-ir-login{
    text-transform: none;
    font-size: 2rem;
  }
}
```

Código de la página de inicio de sesión:

Plantilla TypeScript:

```
import { Component } from '@angular/core';
import { AlertController, NavController } from 'ionic-angular';

//Servicios
import { AutenticacionProvider } from '../../providers/index.servicios';
import { User } from 'firebase/app';

//Páginas
import { RegistrarUsuarioPage } from '../index.paginas'

@Component({
  selector: 'page-login',
  templateUrl: 'login.html',
})
export class LoginPage {

  email_ingresado:string = '';
  contrasena_ingresada:string = '';
  tipo_campo_contrasena: string = 'password';
  icono_mostrar_contrasena: string = 'eye';

  constructor(
    private alertCtrl: AlertController,
    private _aut: AutenticacionProvider,
    private navCtrl: NavController
  ) {  }

  //Metodo que se ejecuta cuando se da click en el botón Enviar
  //Realiza el login del usuario ingresado. Si los datos son correctos el
servicio _aut
  //regresa la data del usuario ingresado
```

```
  onSiguienteLoginUsuario(){
    this._aut.loginUser(this.email_ingresado, this.contrasena_ingresada)
    .then((userData: User)=>{
      //Se identifica si el email está verificado.
      //Si no está verificado se muestra un mensaje con la opción de enviar
un link al correo
      //para que verifique el email
      if (!userData.emailVerified) {
        this.alertCtrl.create({
          title: 'Cuenta no verificada',
          message: 'Su cuenta debe ser verificada para poder continuar, si
desea que\
          enviemos un link de verificación a "' + this.email_ingresado + '"
seleccione la\
          opción "ENVIAR LINK", de lo contrario seleccione "CANCELAR".',
          buttons: [
            {
              text: 'Enviar link',
              handler: () => {
                this._aut.sendEmailVerifyAccount()
                .then(()=>{
                  this.mostrarAlertaInformativa(
                    'Correo electrónico enviado',
                    'Hemos enviado un link al correo electrónico "' +
this.email_ingresado +
                    '" para que realice la verificación de su cuenta'
                  )
                })
                .catch(()=>{});
              }
            },
            {
              text: 'Cancelar'
            }
          ]
        }).present();
      }
    })
    .catch((err)=>{
    });
  }

  //Metodo que permite enviar un email con un link para el reestablecimiento
de la contraseña
  onResetContrasena(){
```

```
    //Se valida si ha ingresado un email en el campo
    if (!this.email_ingresado) {
      let titulo:string = 'Correo electrónico no ingresado';
      let subtitulo:string = 'Por favor ingrese un correo electrónico para
poder reestablecer\
      la contraseña'
      this.mostrarAlertaInformativa(titulo, subtitulo);
    } else {
      //Si ya ingresó el email entonces se muestra una alerta indicando que
sí desea recibir
      //el link para reestablecer la contraseña
      this.alertCtrl.create({
        title: 'Reestablecer contraseña',
        message: 'Si olvidó la contraseña, seleccione la opción "ENVIAR
LINK" para que\
        enviemos un correo electrónico con un link a "' +
this.email_ingresado + '" y pueda\
        reestablecer la contraseña de su cuenta, de lo contrario seleccione
"CANCELAR".',
        buttons: [
          {
            text: 'Enviar link',
            handler: () => {
              this._aut.sendEmailResetPassword(this.email_ingresado)
              .then(()=>{
                this.mostrarAlertaInformativa(
                  'Correo electrónico enviado',
                  'Hemos enviado un link al correo electrónico "' +
this.email_ingresado +
                  '" para que pueda reestablecer la contraseña de su cuenta
y pueda\
                  ingresar'
                )
              })
              .catch((error)=>{
                console.log('Error: ', error);
              });
            }
          },
          {
            text: 'Cancelar'
          }
        ]
      }).present();
```

```
    }
  }

  //Metodo que cambia el tipo del campo contraseña de password a text y la
imagen del ojo
  //para mostrar o no la contraseña
  onMostrarContrasena(){
    if (this.tipo_campo_contrasena === 'password') {
      this.tipo_campo_contrasena = 'text';
      this.icono_mostrar_contrasena = 'eye-off';
    } else {
      this.tipo_campo_contrasena = 'password';
      this.icono_mostrar_contrasena = 'eye';
    }
  }

  //Metodo que permite mostrar una alerta informativa
  mostrarAlertaInformativa(titulo:string, subtitulo:string) {
    this.alertCtrl.create({
      title: titulo,
      subTitle: subtitulo,
      buttons: ['Entendido']
    }).present();
  }

  //Metodo que abre la página RegistrarUsuarioPage
  onIrPagRegistrarUsuario(){
    this.navCtrl.push(RegistrarUsuarioPage);
  }
}
```

Plantilla HTML5:

```
<ion-content padding>

  <h1>INICIAR SESIÓN</h1>
  <p class="titulo-parrafo">Ingrese el correo electrónico y la constraseña
que
    utilizó al registrarse</p>

  <form (ngSubmit)="onSiguienteLoginUsuario()">

    <ion-item>
      <ion-label floating>Correo electrónico</ion-label>
```

```
  <ion-input type="email"
             name='email_ingresado'
             [(ngModel)]='email_ingresado'>
  </ion-input>
</ion-item>

<ion-item>
  <ion-label floating>Contraseña</ion-label>
  <ion-input type="{{tipo_campo_contrasena}}"
             name='contrasena_ingresada'
             [(ngModel)]='contrasena_ingresada'>
  </ion-input>
  <button ion-button
          clear
          type="button"
          item-right
          icon-only
          (click)="onMostrarContrasena()">
    <ion-icon name="{{icono_mostrar_contrasena}}"> </ion-icon>
  </button>
</ion-item>

<ion-buttons end>
  <button ion-button
          clear
          type="button"
          class="btn-reset-contrasena"
          (click)='onResetContrasena()'>
    ¿Olvidaste la contraseña?
  </button>
</ion-buttons>


<button ion-button
        block
        type="submit">
  Enviar
</button>

<button ion-button
        block
        clear
        class="btn-ir-login"
        type="button"
        (click)='onIrPagRegistrarUsuario()'>
```

```
            No estoy registrado
    </button>

  </form>
</ion-content>
```

Plantilla CSS:

```
page-login {
  h1{
    color: color($colors, primary);
    padding-top: 1rem;
    text-align: center;
  }

  p{
    color: color($colors, primary);
  }

  .tituolo-parrafo{
    color: color($colors, primary);
    font-size: 1.8rem;
    text-align: center;
  }

  .btn-reset-contrasena{
    text-transform: none;
  }
}
```

Código de la página de registro de sistemas:

Plantilla TypeScript:

```
import { Component, OnDestroy } from '@angular/core';
import { ToastController, NavController, PopoverController } from 'ionic-
angular';
import { Subscription } from "rxjs/Subscription";

//Componentes
import { PopoverComponent } from '../../components/popover/popover';

//Servicios
import {
```

```
  DbSistUsuariosProvider,
  DbClavesProvider,
  DbSistemasProvider
} from '../../providers/index.servicios';

//Interfaces
import { SistemaCN } from '../../interfaces/intefaces'

//Páginas
import { InicioPage } from '../index.paginas'

@Component({
  selector: 'page-registrar-sistema',
  templateUrl: 'registrar-sistema.html',
})
  export class RegistrarSistemaPage implements OnDestroy{

    nuevo_sistema:boolean;
    clave_ingresada:string;
    nombre_ingresado:string;
    sistemas_usuario_arr:SistemaCN[];
    sistemas_usuario_obj:object;

    sus_registrar_sistemas: Subscription;

    constructor(
      private toastCtrl:ToastController,
      private _dbSistemasUsuarios: DbSistUsuariosProvider,
      private _dbClaves: DbClavesProvider,
      private _dbSistemas: DbSistemasProvider,
      private navCtrl: NavController,
      private popoverCtrl: PopoverController
    ) { }

  //Metodo del ciclo de vida de la página que se ejecuta cuando la página se
va a cargar.
  ionViewDidLoad() {
    //Se obtiene los sistemas del usuario antes de que se cargue la página
    this._dbSistemasUsuarios.iniciarRef().then(_=>{
      //Nos suscribimos al observable que monitorea los sistemas registrados
al usuario
      this.sus_registrar_sistemas =
this._dbSistemasUsuarios.sistemasUsuariosC
        .subscribe((sistemasUsuarios:any)=>{
          //Se verifica si el usuario tiene sistemas relacionados
```

```
          if (sistemasUsuarios.payload.val()) {
            //Se convierte el objeto en un array de objetos para poder
iterar en el html
            this.sistemas_usuario_arr =
Object.values(sistemasUsuarios.payload.val());
            //Se obtiene el objeto de los sistemas para poderlo enviar al
popover cuando
            //el usuario quiera eliminar o modificar el sistema seleccionado
            this.sistemas_usuario_obj = sistemasUsuarios.payload.val();
          }
          else{
            //Si no tiene sistemas relacionados la variable se pone en null
            this.sistemas_usuario_arr = null;
          }
        });
    });
  }

  //Metodo que muestra u oculta los campos para registrar un nuevo sistema
al usuario.
  nuevoSistema(){
    if (this.nuevo_sistema) {
      this.nuevo_sistema = false;
    } else {
      this.nuevo_sistema = true;
    }
  }

  //Metodo que permite guardar un nuevo sistema al usuario
  onEnviarSistema(){
    //Se verifica si ha ingresado una clave en el campo
    if (!this.clave_ingresada) {
      let mensaje:string = 'Ingrese la clave del sistema que compró';
      this.mostrarToast(mensaje);
      return;
    }

    //Se verifica si ha ingresado un nombre para el nuevo sistema
    if (!this.nombre_ingresado) {
      let mensaje:string = 'Ingrese un nombre para el sistema'
      this.mostrarToast(mensaje);
      return;
    }

    //Se verifica si la clave ingresada no tiene 10 caracteres
```

```
    if (this.clave_ingresada.length != 10) {
      let mensaje:string = 'La clave no es correcta';
      this.mostrarToast(mensaje);
      return;
    }

    //Se obtiene las claves de los sistemas creados en la BD
    this._dbClaves.ConsultarClaves()
      .then((claves:any)=>{
        //Despues de obtener las claves se verifica si la ingresada por el
uausrio
        //existe en la BD. Se compara.
        //Si la clave ingresada existe, se continua, pero si no existe se
muestra
        //un toast informando que la clave es incorrecta.
        for (let i = 0; i < claves.length; i++) {
          if (claves[i] == this.clave_ingresada) {
            //Si la clave ingresada existe en la BD se continua.
            //Se verifica si el usuario tiene sistemas asociados. Si no
tiene
            //se guarda el sistema, pero si tiene sistemas se verifica si la
clave ingresada
            //ya está asociada.
            if(!this.sistemas_usuario_arr){
              this.procesoGuardarSistema();
              return;
            }
            else {
              for (let i = 0; i < this.sistemas_usuario_arr.length; i++) {
                if (this.sistemas_usuario_arr[i].clave ==
this.clave_ingresada) {
                  //Como existe se muestra el toast y sale de este metodo
                  let mensaje:string = 'Esta clave ya está registrada en su
perfil';
                  this.mostrarToast(mensaje);
                  return;
                }
              }
              //Como la clave no está asociada al usuario entonces se guarda
el sistema
              this.procesoGuardarSistema();
            }
            return;
          }
        }
```

```
        //Como la clave ingresada no existe se muestra un toast.
        let mensaje = 'La clave no es correcta';
        this.mostrarToast(mensaje);
      })
      .catch(error=>error);
  }

  //Metodo que se ejecuta cuando selecciona un sistema y abre la página de
inicio y envía
  //el sistema seleccionado al servicio dbSistemas para que las páginas de
los tanques
  //puedan obtener la data necesaria
  sistemaSeleccionado(sistema:SistemaCN){
    this.navCtrl.push(InicioPage);
    this._dbSistemas.sistemaSeleccionado(sistema);
  }

  //Metodo que muestra un toast informativo
  mostrarToast(mensaje:string){
    return this.toastCtrl.create({
      message: mensaje,
      duration: 3000
    }).present();
  }

  //Metodo que guarda el sistema que se está registrando
  procesoGuardarSistema(){
    //Promesa que envía la información registrada del nuevo sistema al
servicio para que se
    //guarde en la base de datos
    return new Promise((resolve, reject)=>{
      let sistema:SistemaCN = {
        clave: this.clave_ingresada,
        nombre: this.nombre_ingresado
      }
      this._dbSistemasUsuarios.guardarNuevoSistemaL(sistema)
        .then(()=>{
          //luego de guardar el sistema se borra los campos y se deja de
mostrar el formulario
          this.nuevo_sistema = false;
          this.clave_ingresada = '';
          this.nombre_ingresado = '';
        })
        .catch(error=>error);
        resolve();
```

```
    })
  }

  //Metodo que abre el popover cuando el usuario quiere elimininar o
modificar un sistema
  abrirPopover(evento, sistema, sistemas_usuario_obj){
    this.popoverCtrl.create(PopoverComponent, {sistema,
sistemas_usuario_obj}).present({
      ev: evento
    });
  }

  //Metodo del ciclo de vida del componente que se ejecuta cuando el
componente (Página)
  //se va a destruir
  //Se desuscribre del observable que monitorea los sistemas del usuario y
llama al metodo
  //del servicio _dbSistemasUsuarios destruirRef el cual pone en null la
referencia del
  //nodo que contiene los sistemas del usuario
  ngOnDestroy(){
    if (this.sus_registrar_sistemas) {
      this.sus_registrar_sistemas.unsubscribe();
    }
    this._dbSistemasUsuarios.destruirRef();
  }
}
```

Plantilla HTML5:

```
<ion-header>

  <ion-navbar color='primary'>
    <button
      ion-button
      icon-only
      menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>

    <ion-title>Sistemas registrados</ion-title>
  </ion-navbar>

</ion-header>
```

```
<ion-content padding>
  <button *ngIf='!nuevo_sistema'
    class="btn-nuevo-sistema"
    ion-button
    block
    (click)='nuevoSistema()'>
    Registrar nuevo sistema
  </button>

  <form *ngIf='nuevo_sistema'
    class="formulario"
    (ngSubmit)='onEnviarSistema()'>

    <p>
      Por favor ingrese la clave que se le entregó cuando compró el sistema
y
      el nombre que usted quiera darle.
    </p>

    <ion-item>
      <ion-label floating>Clave del sistema</ion-label>
      <ion-input type="text"
                  name='clave'
                  [(ngModel)]='clave_ingresada'>
      </ion-input>
    </ion-item>

    <ion-item>
      <ion-label floating>Nombre del sistema</ion-label>
      <ion-input type="text"
                 maxlength="25"
                 name='nombre'
                 [(ngModel)]='nombre_ingresado'>
      </ion-input>
    </ion-item>

    <ion-grid>
      <ion-row>
        <ion-col >
          <button
            ion-button
            block
            type="button"
            (click)='nuevoSistema()'>
            Cancelar
```

```
          </button>
        </ion-col>

        <ion-col >
          <button
            ion-button
            block
            type="submit">
            Enviar
          </button>
        </ion-col>
      </ion-row>
    </ion-grid>

  </form>

  <ion-list no-lines>
    <ion-grid>
      <ion-row *ngFor="let sistema of sistemas_usuario_arr" class="fila">
        <ion-col col-auto>
          <button
            ion-button
            icon-only
            round clear
            (click)='abrirPopover($event, sistema, sistemas_usuario_obj)'>
            <ion-icon name="md-more"></ion-icon>
          </button>
        </ion-col>

        <ion-col >
          <button
            class="btn_lista"
            ion-item
            (click)='sistemaSeleccionado(sistema)'>
            {{sistema.nombre}}
            <p class="clave_lista">{{sistema.clave}}</p>
          </button>
        </ion-col>
      </ion-row>
    </ion-grid>
  </ion-list>

</ion-content>
```

Plantilla CSS:

```
page-registrar-sistema {
  .btn-nuevo-sistema{
    margin-bottom: 4rem;
  }

  .formulario{
    margin-bottom: 4rem;
  }

  p{
    color: color($colors, primary);
  }

  .btn_lista{
    font-size: 2rem;
    padding: 0;
    color: color($colors, primary)
  }

  .clave_lista{
    text-align: left;
  }

  .fila{
    border-bottom: solid 0.1rem;
    border-color: color($colors, primary);
  }
}
```

Código de la página del tanque T0:

Plantilla TypeScript:

```
import { Component, OnInit } from '@angular/core';
import { NavParams} from 'ionic-angular';
import { Subscription } from "rxjs/Subscription";

//Servicios
import { DbSistemasProvider } from '../../providers/db-sistemas/db-
sistemas';

//Intefaces
import { SistemaCN, Data, Sistema } from '../../interfaces/intefaces'

@Component({
```

```
  selector: 'page-t0',
  templateUrl: 't0.html',
})
export class T0Page implements OnInit {

  sistema_seleccionado:SistemaCN;
  data_sistema:Data;
  sistema:Sistema;

  img_T0:string;
  card_footer:string;

  sus_t0: Subscription;

  constructor(
    private navParams: NavParams,
    private _dbSistemas: DbSistemasProvider
  ) {}

  //Metodo del ciclo de vida del componente (página) que se ejecuta al
iniciar el componente,
  //despues de que se ejecuta el constructor
  ngOnInit(){
    //Nos subcribimos al observable que monitorea el sistema seleccionado
por el usuario
    //para saber cuando ocurran cambios.
    this.sus_t0 =
this._dbSistemas.Sistema.subscribe((sistema_config_data:any)=>{
      this.data_sistema = sistema_config_data.data
      //Cada vez que cambie la data del sistema se realiza la verificación
del estado
      //de sw0 para volver a renderizar la página
      this.presentacion();
    });
    //Se obtiene el sistema seleccionado y su data, esto se debe hacer
    //porque al subscribirse, como la data no ha cambiado, no se puede
obtener nada
    return new Promise((resolve, reject)=>{
      this.sistema_seleccionado =
this.navParams.get('sistema_seleccionado');
      this.sistema = this.navParams.get('sistema');
      this.data_sistema = this.sistema.data;
      resolve();
    })
      .then(()=>{
```

```
        //se veririfica el estado de sw0 para renderizar la página
        this.presentacion();
      })
      .catch(error=>error)
  }

  //Metodo que verifica la data de t0 y renderiza la página
  presentacion(){
    //Se verifica si existen errores
    if (this.data_sistema.e !== 0 ) {
      this.img_T0 = './assets/imgs/error.png';
      switch (this.data_sistema.e) {
        case 1:
          this.card_footer = 'Error en el tanque T1, por favor verifique los
swichs';
          break;

        case 2:
          this.card_footer = 'Error en el tanque T2, por favor verifique los
swichs';
          break;

        case 3:
          this.card_footer = 'Error en el tanque T3, por favor verifique el
sensor';
          break;
      }
      return;
    }

    //Si sw0 es false, se muestra la imagen y el footer correspondiente
    if (!this.data_sistema["0"]) {
      this.img_T0 = './assets/imgs/T0_lleno.png';
      this.card_footer = `Agua disponible en T0`;
    }
    else{
      this.img_T0 = './assets/imgs/T0_vacio.png';
      this.card_footer = `Sin agua en T0`;
    }
  }

  //Metodo del ciclo de vida de la página que se ejecuta cuando se va a
destruir la página.
  //Se desuscribe del observable que monitorea el sistema seleccionado
  ionViewWillUnload(){
```

```
    if (this.sus_t0) {
      this.sus_t0.unsubscribe();
    }
  }
}
```

Plantilla HTML5:

```
<ion-header>
  <ion-navbar color='primary'>
    <ion-title>{{sistema_seleccionado.nombre}}</ion-title>
  </ion-navbar>
</ion-header>

<ion-content>

  <toolbar-actuadores></toolbar-actuadores>

  <div class="div_img">
    <img src="{{img_T0}}" class="tanque">
  </div>

</ion-content>

<ion-footer>
  <ion-card class="ion_card" color='primary'>
    <ion-card-header class="card_title">
      Tanque T0
    </ion-card-header>
    <ion-card-content>
      {{card_footer}}
    </ion-card-content>
  </ion-card>
</ion-footer>
```

Plantilla CSS:

```
page-t0 {

  .div_img{
    text-align: center;
  }

  .tanque{
    max-width: 60%;
```

```
    padding-top: 4rem;
  }

  .ion_card{
    border-radius: 1rem;
  }

  .card_title{
    font-weight: bold;
    padding-top: 0.5rem;
    padding-bottom: 0;
  }
}
```

Código de la página del tanque T1:

Plantilla TypeScript:

```
import { Component, OnInit } from '@angular/core';
import { NavParams} from 'ionic-angular';
import { Subscription } from "rxjs/Subscription";

//Servicios
import { DbSistemasProvider } from '../../providers/db-sistemas/db-
sistemas';

//Intefaces
import { SistemaCN, Data, Sistema } from '../../interfaces/intefaces'

@Component({
  selector: 'page-t1',
  templateUrl: 't1.html',
})
export class T1Page implements OnInit {

  sistema_seleccionado:SistemaCN;
  data_sistema:Data;
  sistema:Sistema;
  sus_t1:Subscription;

  img_T1:string;
  card_footer:string;

  constructor(
    private navParams: NavParams,
```

```
    private _dbSistemas: DbSistemasProvider
  ) { }

  ngOnInit(){
    //Nos subcribimos a la referencia de la BD en el servicio para saber
cuando ocurran
    //cambios en la BD
    this.sus_t1 =
this._dbSistemas.Sistema.subscribe((sistema_config_data:any)=>{
      this.data_sistema = sistema_config_data.data
      //Cada vez que cambie la data del sistema se realiza la verificación
del estado
      //de t0_sw0 para volver a renderizar la página
      this.presentacion();
    });
    //Se obtiene el sistema seleccionado y su data, esto se debe hacer
    //porque al subscribirse, como la data no ha cambiado, no se puede
obtener nada
    return new Promise((resolve, reject)=>{
      this.sistema_seleccionado =
this.navParams.get('sistema_seleccionado');
      this.sistema = this.navParams.get('sistema');
      this.data_sistema = this.sistema.data;
      resolve();
    })
      .then(()=>{
        //se veririfica el estado de t0_sw0 para renderizar la página
        this.presentacion();
      })
      .catch(error=>error)
  }
  presentacion(){
    //Se verifica si existen errores
    if (this.data_sistema.e !== 0 ) {
      this.img_T1 = './assets/imgs/error.png';
      switch (this.data_sistema.e) {
        case 1:
          this.card_footer = `Error en el tanque T0, parece que T0 se ha
quedado sin agua
          y el filtro de gravilla se puede deteriorar`;
          break;

        case 2:
          this.card_footer = 'Error en el tanque T1, por favor verifique los
swichs';
```

```
          break;

        case 3:
          this.card_footer = 'Error en el tanque T2, por favor verifique los
swichs';
          break;

        case 4:
          this.card_footer = 'Error en el tanque T3, por favor verifique el
sensor';
          break;
      }
      return;
    }

    if (!this.data_sistema["1"] && !this.data_sistema["2"]) {
      this.img_T1 = './assets/imgs/T1_vacio.png';
      this.card_footer = 'El tanque está vacio';
      return
    }

    if (this.data_sistema["1"] && !this.data_sistema["2"]) {
      this.img_T1 = './assets/imgs/T1_medio_lleno.png';
      this.card_footer = 'El tanque no está totalmente lleno';
      return;
    }

    if (this.data_sistema["1"] && this.data_sistema["2"]) {
      this.img_T1 = './assets/imgs/T1_lleno.png';
      this.card_footer = 'El tanque está lleno';
      return;
    }
  }

  ionViewWillUnload(){
    if (this.sus_t1) {
      this.sus_t1.unsubscribe();
    }
  }
}
```

Plantilla HTML5:

```
<ion-header>
```

```
    <ion-navbar color='primary'>
      <ion-title>{{sistema_seleccionado.nombre}}</ion-title>
    </ion-navbar>
  </ion-header>

  <ion-content>

    <toolbar-actuadores></toolbar-actuadores>

    <div class="div_img">
      <img src="{{img_T1}}" class="tanque">
    </div>

  </ion-content>

  <ion-footer>
    <ion-card class="ion_card" color='primary'>
      <ion-card-header class="card_title">
        Tanque T1
      </ion-card-header>
      <ion-card-content>
        {{card_footer}}
      </ion-card-content>
    </ion-card>
  </ion-footer>
```

Plantilla CSS:

```
page-t1 {

  .div_img{
    text-align: center;
  }

  .tanque{
    max-width: 60%;
    padding-top: 4rem;
  }

  .ion_card{
    border-radius: 1rem;
  }

  .card_title{
    font-weight: bold;
```

```
    padding-top: 0.5rem;
    padding-bottom: 0;
  }
}
```

Código de la página del tanque T2:

Plantilla TypeScript:

```
import { Component, OnInit } from '@angular/core';
import { NavParams} from 'ionic-angular';
import { Subscription } from "rxjs/Subscription";

//Servicios
import { DbSistemasProvider } from '../../providers/db-sistemas/db-
sistemas';

//Intefaces
import { SistemaCN, Data, Sistema } from '../../interfaces/intefaces'

@Component({
  selector: 'page-t2',
  templateUrl: 't2.html',
})
export class T2Page implements OnInit {

  sistema_seleccionado:SistemaCN;
  data_sistema:Data;
  sistema:Sistema;
  sus_t2:Subscription;

  img_T2:string;
  card_footer:string;

  constructor(
    private navParams: NavParams,
    private _dbSistemas: DbSistemasProvider
  ) {}

  ngOnInit(){
    //Nos subcribimos a la referencia de la BD en el servicio para saber
cuando ocurran
    //cambios en la BD
```

```
    this.sus_t2 =
this._dbSistemas.Sistema.subscribe((sistema_config_data:any)=>{
      this.data_sistema = sistema_config_data.data
      //Cada vez que cambie la data del sistema se realiza la verificación
del estado
      //de t0_sw0 para volver a renderizar la página
      this.presentacion();
    });
    //Se obtiene el sistema seleccionado y su data, esto se debe hacer
    //porque al subscribirse, como la data no ha cambiado, no se puede
obtener nada
    return new Promise((resolve, reject)=>{
      this.sistema_seleccionado =
this.navParams.get('sistema_seleccionado');
      this.sistema = this.navParams.get('sistema');
      this.data_sistema = this.sistema.data;
      resolve();
    })
    .then(()=>{
      //Se renderiza la página despues de que se obtiene los datos
      this.presentacion();
    })
    .catch(error=>error)
  }

  presentacion(){
    //Se verifica si existen errores
    if (this.data_sistema.e !== 0 ) {
      this.img_T2 = './assets/imgs/error.png';
      switch (this.data_sistema.e) {
        case 1:
          this.card_footer = `Error en el tanque T0, parece que T0 se ha
quedado sin agua
          y el filtro de gravilla se puede deteriorar`;
          break;

        case 2:
          this.card_footer = 'Error en el tanque T1, por favor verifique los
swichs';
          break;

        case 3:
          this.card_footer = 'Error en el tanque T2, por favor verifique los
swichs';
          break;
```

```
        case 4:
          this.card_footer = 'Error en el tanque T3, por favor verifique el
sensor';
          break;
      }
      return;
    }

    if (!this.data_sistema["3"] && !this.data_sistema["4"]) {
      this.img_T2 = './assets/imgs/T2_vacio.png';
      this.card_footer = 'El tanque está vacio';
      return;
    }

    if (this.data_sistema["3"] && !this.data_sistema["4"]) {
      this.img_T2 = './assets/imgs/T2_medio_lleno.png';
      if (this.data_sistema.a) this.card_footer = 'El tanque no está
totalmente lleno y el agua ya fue tratada';
      else this.card_footer = 'El tanque no está totalmente lleno y el agua
no ha sido tratada';
      return;
    }

    if (this.data_sistema["3"] && this.data_sistema["4"]) {
      if (this.data_sistema.L) {
        this.img_T2 = './assets/imgs/T2_lleno_L_on.png';
        this.card_footer = 'El tanque está lleno y el agua se está
tratando';
      }
      else {
        this.img_T2 = './assets/imgs/T2_lleno_L_off.png';
        this.card_footer = 'El tanque está lleno y el agua ya fue tratada';
      }
      return;
    }
  }

  ionViewWillUnload(){
    if (this.sus_t2) {
      this.sus_t2.unsubscribe();
    }
  }
}
```

Plantilla HTML5:

```
<ion-header>
  <ion-navbar color='primary'>
    <ion-title>{{sistema_seleccionado.nombre}}</ion-title>
  </ion-navbar>
</ion-header>

<ion-content>
  <toolbar-actuadores></toolbar-actuadores>

  <div class="div_img">
    <img src="{{img_T2}}" class="tanque">
  </div>

</ion-content>

<ion-footer>
  <ion-card class="ion_card" color='primary'>
    <ion-card-header class="card_title">
      Tanque T2
    </ion-card-header>
    <ion-card-content>
      {{card_footer}}
    </ion-card-content>
  </ion-card>
</ion-footer>
```

Plantilla CSS:

```
page-t2 {

  .div_img{
    text-align: center;
  }

  .tanque{
    max-width: 60%;
    padding-top: 4rem;
  }

  .ion_card{
    border-radius: 1rem;
  }

  .card_title{
```

```
    font-weight: bold;
    padding-top: 0.5rem;
    padding-bottom: 0;
  }
}
```

Código de la página del tanque T3:

Plantilla TypeScript:

```
import { Component, OnInit } from '@angular/core';
import { NavParams} from 'ionic-angular';
import { Subscription } from "rxjs/Subscription";

//Servicios
import { DbSistemasProvider } from '../../providers/db-sistemas/db-
sistemas';

//Intefaces
import { SistemaCN, Data, Sistema } from '../../interfaces/intefaces'

@Component({
  selector: 'page-t3',
  templateUrl: 't3.html',
})
export class T3Page {

  sistema_seleccionado:SistemaCN;
  data_sistema:Data;
  sistema:Sistema;

  img_T3:string;
  sus_t3:Subscription;
  volt3:number;
  card_footer:string;
  error:boolean;

  constructor(
    private navParams: NavParams,
    private _dbSistemas: DbSistemasProvider
  ) { }

  ngOnInit(){
    //Nos subcribimos a la referencia de la BD en el servicio para saber
cuando ocurran
```

```
    //cambios en la BD
    this.sus_t3 = this._dbSistemas.Sistema.subscribe((sistema:any)=>{
      this.data_sistema = sistema.data
      //Se obtiene el volumen en T3
      this.volt3 = Math.trunc((Math.PI * (Math.pow(sistema.T3.radio, 2) *
        (sistema.T3.altura - sistema.data.s)))/1000);

      if (this.volt3 < 0) {
        this.volt3 = 0;
      }

      //Cada vez que cambie la data del sistema se realiza la verificación
del estado
      //de t0_sw0 para volver a renderizar la página
      this.presentacion();
    });
    //Se obtiene el sistema seleccionado y su data, esto se debe hacer
    //porque al subscribirse, como la data no ha cambiado, no se puede
obtener nada
    return new Promise((resolve, reject)=>{
      this.sistema_seleccionado =
this.navParams.get('sistema_seleccionado');
      this.sistema = this.navParams.get('sistema');
      this.data_sistema = this.sistema.data;
      resolve();
    })
      .then(()=>{
        //se veririfica el estado de t0_sw0 para renderizar la página
        //Se obtiene el volumen en T3
        this.volt3 = Math.trunc((Math.PI * (Math.pow(this.sistema.T3.radio,
2) *
          (this.sistema.T3.altura - this.sistema.data.s)))/1000);

        if (this.volt3 < 0) {
          this.volt3 = 0;
        }

        this.presentacion();
      })
      .catch(error=>error)
  }

  presentacion(){
    //Se verifica si existen errores
    if (this.data_sistema.e !== 0 ) {
```

```
      this.error = true;
      this.img_T3 = './assets/imgs/error.png';
      switch (this.data_sistema.e) {
        case 1:
          this.card_footer = 'Error en el tanque T1, por favor verifique los
swichs';
          break;

        case 2:
          this.card_footer = 'Error en el tanque T2, por favor verifique los
swichs';
          break;

        case 3:
          this.card_footer = 'Error en el tanque T3, por favor verifique el
sensor';
          break;
      }
    }
    else {
      this.error = false;
    }

    if (this.data_sistema.s >= this.sistema.T3.altura * 0.66) {
      this.img_T3 = './assets/imgs/T3_vacio.png'
    }
    else if (this.data_sistema.s <= this.sistema.T3.altura * 0.33){
      this.img_T3 = './assets/imgs/T3_lleno.png';
    }
      else{
        this.img_T3 = './assets/imgs/T3_medio_lleno.png';
      }
  }

  ionViewWillUnload(){
    if (this.sus_t3) {
      this.sus_t3.unsubscribe();
    }
  }
}
```

Plantilla HTML5:

```
<ion-header>
```

```
  <ion-navbar color='primary'>
    <ion-title>{{sistema_seleccionado.nombre}}</ion-title>
  </ion-navbar>
</ion-header>


<ion-content>
  <toolbar-actuadores></toolbar-actuadores>

  <div class="div_img">
    <img src="{{img_T3}}" class="tanque">
  </div>
</ion-content>

<ion-footer>
  <ion-card class="ion_card" color='primary'>
    <ion-card-header class="card_title">
      Tanque T3
    </ion-card-header>
    <ion-card-content *ngIf='!error' text-center>
      <h1>{{volt3}} Litros</h1>
      aprox para usar
    </ion-card-content>

    <ion-card-content *ngIf='error' text-center>
      {{card_footer}}
    </ion-card-content>
  </ion-card>
</ion-footer>
```

Plantilla CSS:

```
page-t3 {

  .div_img{
    text-align: center;
  }

  .tanque{
    max-width: 60%;
    padding-top: 4rem;
  }

  .ion_card{
```

```
    border-radius: 1rem;
  }

  .card_title{
    font-weight: bold;
    padding-top: 0.5rem;
    padding-bottom: 0;
  }
}
```