UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Sistema de automatización y control de una zona de ocio con piscina y cerramiento abatible

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Amores Dolz, David

Tutor/a: Hervás Jorge, Antonio

Cotutor/a: Capilla Lladró, Roberto

CURSO ACADÉMICO: 2021/2022

Agradecimientos

Este trabajo me gustaría dedicárselo sobre todo a mi hermano, por la ayuda recibida durante toda mi vida y su apoyo, y por haber creado y desarrollado un proyecto tan grande como este conjuntamente, lo que ha hecho unirnos aún más.

A mi padre y a mi madre también por darme la fuerza para seguir luchando constantemente y afrontar todo lo que pase en la vida, ya sea bueno o malo.

A toda mi familia y en especial a mi primo Javier, por transmitirme todos los conocimientos y esa forma de vivir la vida sin presiones.

Y, por último, agradecérselo a mi grupo de amigos, en especial a Ricardo por meterme la presión de tener que terminar la carrera y por los buenos momentos vividos a lo largo de nuestra amistad y por estar siempre ahí apoyándome.

Resumen

En este trabajo de final de grado vamos a desarrollar e implementar un sistema de automatización completo para una piscina y un techo exterior abatible con todos los elementos que estos puedan llevar, ya sea desde el control completo de los elementos esenciales de la piscina, depuradora, índices de pH y cloro en el agua, como incluso elementos anexos a esta. En este caso una bomba de aire para imitar una piscina estilo jacuzzi, una bomba de agua para una cascada, luces con tecnología RGB, chorros de agua y llenado de la piscina cuando esta se encuentre por debajo del nivel de agua deseado. Todo esto será necesario para el control y realización del sistema de automatización que se va a desarrollar con el hardware Raspberry Pi, dispositivos Sonoff y elementos de control.

Por otra parte, se añadirá el manejo de un sistema de audio para la zona del techo exterior abatible con el control total sobre el mismo, iluminación con luces RGB y control de apertura y cierre del mismo. Todo este proyecto está destinado a facilitar el control y a mejorar el uso y disfrute de los elementos a través de un sistema automatizado y controlado mediante una aplicación central en cualquier dispositivo móvil o pc, llamado Home Assistant.

Este proyecto va de la mano de un segundo proyecto de final de grado, que en conjunto forman la automatización completa de todos los elementos de una casa/chalet.

Palabras clave: Raspberry Pi, sonoff, home assistant, tasmota, aplicación, sensores, medidores, luces RGB, control, automatización, electroválvula, efecto hall, servomotor, sistema hidráulico.

Abstract

In this final degree project we are going to develop and implement a complete automation system for a swimming pool and a retractable exterior roof with all the elements that these can carry, either from the complete control of the essential elements of the swimming pool, sewage treatment plant, pH and chlorine indices in the water, as well as elements attached to it. In this case, an air pump to imitate a jacuzzi-style pool, a water pump for a waterfall, lights with RGB technology, water jets and filling the pool when it is below the desired water level. All this will be necessary for the control and realization of the automation system that will be developed with the Raspberry Pi hardware, Sonoff devices and control elements.

On the other hand, the management of an audio system for the area of the folding exterior roof will be added with total control over it, lighting with RGB lights and control of opening and closing of the same. This entire project is intended to facilitate control and improve the use and enjoyment of the elements through an automated system controlled by a central application on any mobile device or PC, called Home Assistant.

This project goes hand in hand with a second final degree project, which together form the complete automation of all the elements of a house.

Keywords: Raspberry Pi, sonoff, home assistant, tasmota, app, sensors, meters, RGB lights, control, automation, solenoid, hall effect, servo motor, hydraulic system.

Resum

En aquest treball de final de grau desenvoluparem i implementar un sistema d'automatització complet per a una piscina i un sostre exterior abatible amb tots els elements que aquests puguen portar, ja siga des del control complet dels elements essencials de la piscina, depuradora, índexs de pH i clor en l'aigua, com fins i tot elements annexos a aquesta. En aquest cas una bomba d'aire per a imitar una piscina estil jacuzzi, una bomba d'aigua per a una cascada, llums amb tecnologia RGB, dolls d'aigua i ompliment de la piscina quan aquesta es trobe per davall del nivell d'aigua desitjat. Tot això serà necessari per al control i realització del sistema d'automatització que es desenvoluparà amb el maquinari Raspberry Pi, dispositius Sonoff i elements de control.

D'altra banda, s'afegirà el maneig d'un sistema d'àudio per a la zona del sostre exterior abatible amb el control total sobre aquest, il·luminació amb llums RGB i control d'obertura i tancament d'aquest. Tot aquest projecte està destinat a facilitar el control i a millorar l'ús i gaudi dels elements a través d'un sistema automatitzat i controlat mitjançant una aplicació central en qualsevol dispositiu mòbil o pc, anomenat Home Assistant.

Aquest projecte va de la mà d'un segon projecte de final de grau, que en conjunt formen l'automatització completa de tots els elements d'una casa/xalet.

Paraules clau: Raspberry Pi, sonoff, home assistant, tasmota, aplicació, sensors, mesuradors, llums RGB, control, automatització, electrovàlvula, efecte hall, servomotor, sistema hidràulic.

Tabla de contenidos

Índice de figuras

1. Introducción

Este proyecto parte de la necesidad de buscar y crear un sistema de automatización de una zona de ocio con piscina y un techo exterior abatible. Hoy en día, podemos encontrar una gran variedad de productos y dispositivos en el mercado que desempeñan estas funciones con el problema de que lo hacen por separado y no en conjunto, dificultando de esta manera el poder automatizar y centralizar todos los componentes dentro de un mismo sistema.

Cabe destacar, que los productos que se ofertan actualmente son de un alto coste económico y con muy poca posibilidad de poderse adaptar en cualquier entorno si este no ha sido preparado para ello con anterioridad. Por otra parte, a causa de esto también surge el problema de que estos productos al estar desarrollados para un sistema en concreto, dan muy poca posibilidad de poder ser modificados según las necesidades del consumidor o del usuario final.

Por todas estas causas, surge este mismo proyecto, en el que intentamos realizar un sistema automatizado que se pueda implementar en cualquier entorno y para cualquier objeto, sin la necesidad de ser los elementos propuestos en este mismo proyecto, es decir, se dan las pautas para poder realizar este mismo desarrollo con cualquier dispositivo capaz de ser controlado mediante aparatos que envían órdenes.

Por otra parte, con este proyecto también se pretende mejorar y hacer más entretenido nuestro entorno, pudiendo ser controlado y cuidado mediante el uso de una simple aplicación móvil o web.

Finalmente, y gracias a la ayuda que nos presta la tecnología de hoy en día, vemos posible mediante unos dispositivos como Raspberry Pi, Sonoff y un pequeño circuito PCB, junto con todos los elementos a querer automatizar, poder controlarlos desde una aplicación general llamada Home Assistant, que facilitará el uso de estos elementos mediante una red local conectada a Internet, para así poder acceder a ella e interaccionar con todo el sistema automatizado desde cualquier lugar.

1.1 Motivación

El planteamiento de este proyecto, surge como consecuencia de haber terminado el grado en ingeniería informática y dar la casualidad de estar compartiendo la misma profesión y afición por la informática de la mano de mi hermano y el tener que desarrollar también un proyecto. Por estas razones, llegamos a un punto de tener que desarrollar un proyecto que fuera lo suficientemente amplio para poder desarrollar dos proyectos, que juntos formaran un proyecto mucho más grande y que abarcara más conocimientos y desarrollos.

Teniendo esto claro, la motivación también surgía por la necesidad de implementar y automatizar nuestra propia casa/chalet y de paso poder probar en nuestro propio lugar todo el proyecto y poder tener una base de pruebas mucho más consistente.

A raíz de esto, surgió y se llevó a cabo el proyecto y claramente motivados por ser el trabajo propio de final de grado y poder realizarlo para nosotros mismos.

Surgieron diferentes ideas en un principio y relacionadas con el trabajo que desempeñábamos, como ya fuera el desarrollo de alguna aplicación web, algún sistema de control de empresas, pero finalmente nos decantamos por realizar un sistema de automatización general de todos los dispositivos de una casa y así poder dividir la carga de trabajo en dos proyectos, que unidos crean un sistema de automatización muy completo.

1.2 Objetivos

El objetivo principal del proyecto es la automatización de todas las zonas que teníamos a nuestra disposición.

Por todo esto, planteamos una serie de objetivos que queríamos alcanzar y que vamos a realizar en el proyecto, como son:

- Controlar los niveles de pH y cloro en el agua con sondas y enviar los datos a una aplicación.
- Poder activar el llenado de la piscina en cualquier momento sin tener que estar presentes o tener que colocar cualquier elemento para proceder a su llenado. (Esto se debía hacer porque el nivel del agua con el paso de los días terminaba disminuyendo, ya fuera por la evaporación o por el simple uso de la piscina, con el problema de que si no tiene un nivel adecuado de agua pudiera romperse la bomba depuradora).
- Activar o desactivar la bomba depuradora.
- Activar o desactivar el robot de limpieza de la piscina en cualquier momento. (Este robot se encuentra siempre sumergido, por lo tanto, solo es necesario activarlo y para ello buscábamos hacerlo de una manera remota).
- Encender o apagar las luces sumergibles de la piscina con el color deseado.
- Activar la cascada de agua de la piscina.
- Activar la bomba de aire para convertir la piscina en un jacuzzi.
- Levantar o cerrar el techo exterior abatible.
- Encender, apagar o cambiar el color de las luces del techo exterior abatible.
- Encender o apagar el equipo de música a distancia o reproducir la música deseada desde el propio móvil a través de la aplicación.
- Controlar el foco de luz exterior del recinto de la piscina.

1.3 Estructura de la memoria

Este trabajo está organizado en tres fases que se desarrollan de la siguiente forma:

- **Fase 1**: Desarrollo del sistema de automatización de la piscina.
- **Fase 2**: Desarrollo del sistema de automatización del techo exterior abatible y zona de ocio.
- **Fase 3**: Conexión con la aplicación Home Assistant.

En el capítulo 1, realizaremos un estudio de los productos que podemos encontrar hoy en día en el mercado y que se puedan asemejar en alguna funcionalidad que se va a desempeñar en este proyecto. También, realizaremos una crítica de los mismos, para finalmente dar una opinión de porque se presenta esta propuesta y que viene a intentar solucionar en el mundo de los sistemas de automatización.

A continuación, en el capítulo 2, se hará un análisis del problema, en el cual se valorará y se identificara los problemas y las diferentes soluciones que podremos valorar de realizar y por qué se descartan y cual finalmente se elige con su respectiva explicación. Se expondrá y explicará la solución propuesta, de la cual ya partirá todo el proyecto y explicaciones siguientes. En este mismo apartado, se mostrará un presupuesto orientativo, tanto del sistema automatizado de la piscina, como el del techo exterior abatible.

En el tercer capítulo, después de haber hecho una valoración sobre la propuesta pasaremos a explicar el diseño de la solución, tanto la arquitectura de los dos sistemas por separado, como del sistema central que será la aplicación, junto con toda la tecnología aplicada y utilizada y porque se ha utilizado un tipo de software y otro no, para llegar a la solución del desarrollo del proyecto. Se incluirán tanto los nombres de los dispositivos, como el lenguaje de programación utilizado y las librerías o paquetes descargados e implementados y todos los componentes del sistema.

Seguidamente, en el capítulo 4, se realizará todo el desarrollo del proyecto incluyendo partes de código de los scripts desarrollados junto con imágenes, configuraciones, instalaciones y el despliegue de la aplicación.

Por último, desarrollado en el capítulo 5 y 6, se mostrarán imágenes de la implantación y las pruebas de todo el proyecto en funcionamiento y las conclusiones a las que se han llegado después de haber realizado todo el desarrollo del sistema y posibles y futuras mejoras que se podrían realizar en el proyecto.

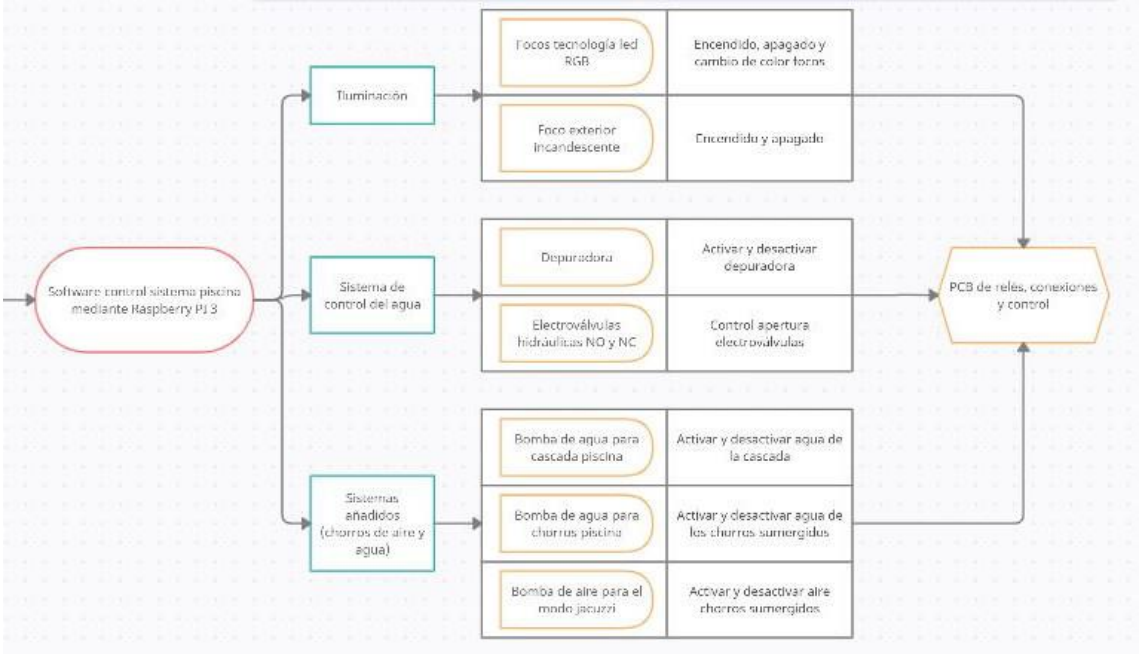


Figura 1 Esquema primera fase a desarrollar del proyecto (Piscina)

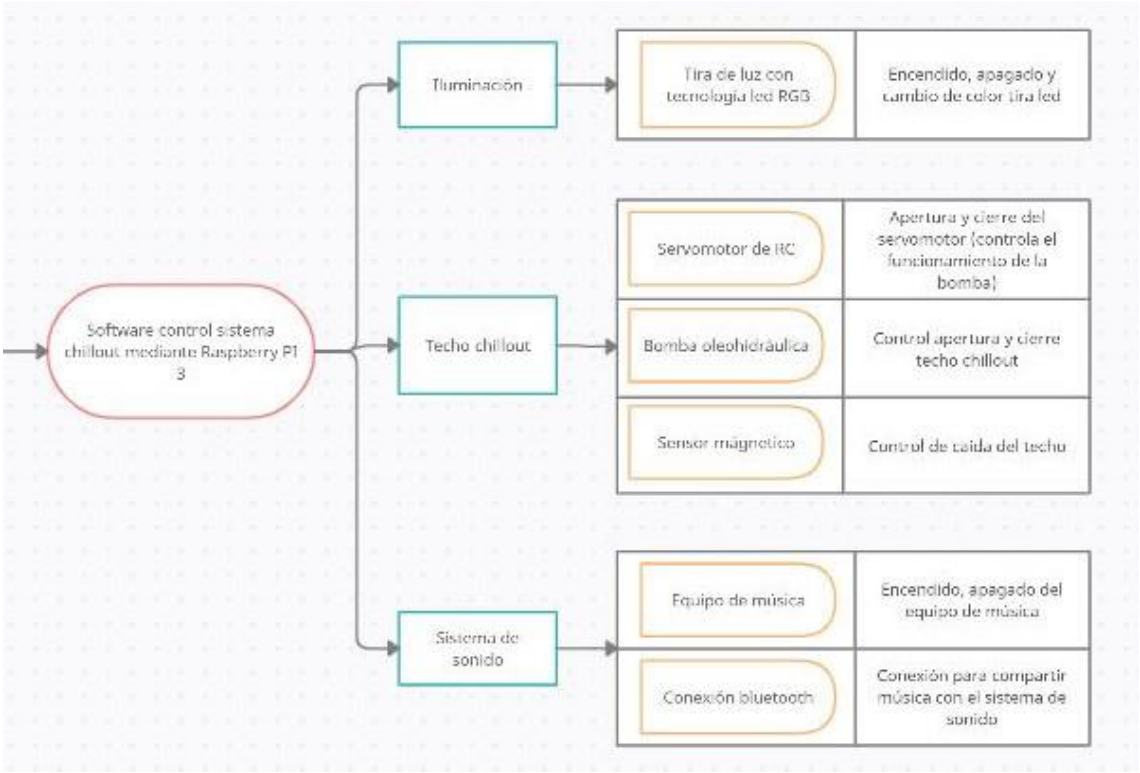


Figura 2 Esquema segunda fase a desarrollar del proyecto (Techo exterior abatible)

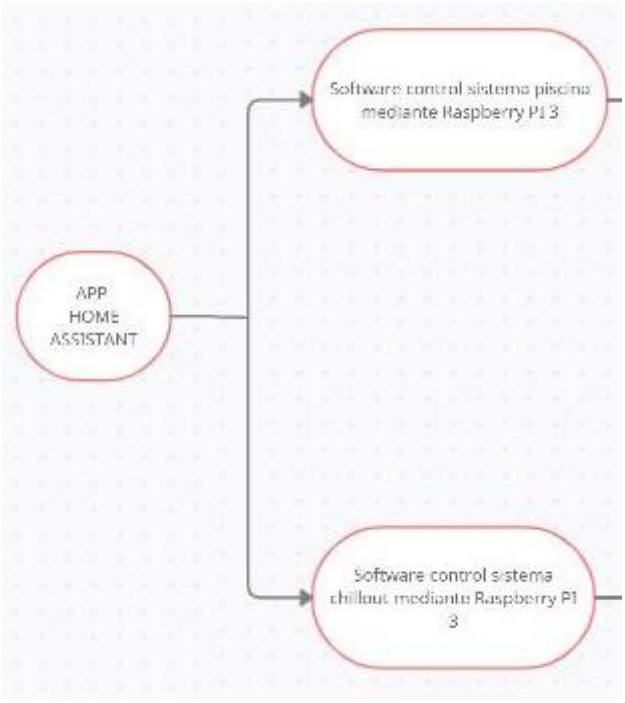


Figura 3 Esquema tercera fase a desarrollar del proyecto (Aplicación Home Assistant)

2. Estado del arte

Hoy en día, gracias al desarrollo tecnológico que vivimos, podemos encontrar una gran variedad de posibilidades de desarrollar este mismo proyecto, pero de diferentes formas, también dependiendo del objetivo que se quiera alcanzar. En nuestro caso, nos hemos basado en dos grandes alternativas, con las dos se puede desempeñar la función del proyecto sin ningún tipo de problema, y esas dos alternativas son Arduino y Raspberry PI.

Finalmente, la unión del proyecto se llevará a cabo mediante la aplicación Home Assistant, la cual existe en el mercado y es una de las aplicaciones más completas y que facilitan la funcionalidad de cualquier sistema considerablemente, no hemos encontrado competidor que desempeñe o que se pueda manejar con tanta facilidad.

2.1 Crítica al estado del arte

Buscando productos o proyectos ya existentes en el mercado actualmente, podemos encontrar las siguientes propuestas.

A continuación, detallamos algunos de los dispositivos o sistemas:

- **Armario Electronic-050**

  Este dispositivo[1] consta de un armario con magnetotérmicos eléctricos que controlan el funcionamiento de la bomba de depuración y 4 canales WiFi, que permiten el control mediante un dispositivo móvil de la propia depuradora y de las luces de la piscina.

  El problema de este dispositivo principalmente es el gran coste que tiene, pero si nos centramos en la parte de automatización, solo da la posibilidad de tener 4 canales para controlar solo 4 elementos de nuestra piscina. No podremos incluir, por supuesto, ni focos RGB, ni tener controlado los niveles de líquidos de una piscina, ni permitir el llenado de la piscina, etc. Junto con este producto, sería necesario el uso de más dispositivos para poder cumplir con las expectativas que se presentan en este proyecto.

  Esta propuesta tiene un precio en el mercado de 725€.

Figura 4 Dispositivo Electronic-050

- **Smart Pool Controller**

Con este dispositivo[2], tenemos la posibilidad de controlar mediante relojes programables el tiempo de filtración de la piscina, entradas y salidas para el sistema de iluminación tanto normal como RGB y sonda de temperatura para controlar el estado del agua.

Este producto facilita la instalación y configuración de las acciones anteriormente descritas, pero todo mediante la pantalla que lleva incorporada el dispositivo, es decir, se deben realizar estas tareas "in situ", sin la posibilidad de ser controladas mediante un dispositivo móvil o un ordenador remotamente.

Este dispositivo tiene un precio de 185€.

Figura 5 Dispositivo Smart Pool Controller

- # Fluidra Connect

Se trata de un sistema[3] tanto de hardware como software, capaz de controlar todos los elementos de una piscina, de forma remota por el usuario. Permite diagnosticar, gestionar y controlar todo el sistema desde una pantalla de ordenador o desde la propia aplicación de la empresa.

Este dispositivo/sistema se asemeja a lo que nosotros buscamos, con la diferencia de que no permite añadirle más dispositivos de los que está programado por defecto, ni tampoco el uso de luces RGB, al no tener un sistema dedicado a ello.

Consta de dos elementos, uno general que es donde se conectan todos los dispositivos y otro elemento que es el comunicador y el que posee la aplicación.

Figura 6 Fluidra connect

Los dispositivos que hemos presentado y que se podrían adaptar a lo que buscamos desarrollar, no son tan eficientes ni económicos como la solución que planteamos en nuestra propuesta. Además, las propuestas que hemos indicado no proporcionan ninguna plataforma para la gestión de las acciones.

Estas soluciones únicamente están pensadas para cubrir las necesidades básicas de una piscina, como ya puedan ser el control lumínico de la piscina, programación de los ciclos de depuración, niveles de pH y cloro, encendido o apagado de robot limpia fondos, etc. Cabe destacar que todas estas funciones nunca van englobadas en un mismo dispositivo.

2.2 Propuesta

En este caso y en esta parte del proyecto, nos hemos decantado por el uso de la Raspberry PI.

Podemos destacar que Arduino puede desarrollar la misma función que la alternativa escogida pero finalmente nos decantamos por la Raspberry PI, debido a que Arduino en su base no lleva incluido ni conectividad WiFi ni el puerto ethernet integrado, cosa que Raspberry PI en su defecto si lo lleva.

Por otra parte, Arduino en cuanto a tiempo de ejecución es mucho más rápido que la Raspberry PI, ya que esta deberá cargar un sistema operativo, que en cambio Arduino estará disponible mucho antes debido a que solo ejecutará ordenes cargadas previamente sin tener que hacer una puesta en marcha.

En conclusión, la propuesta que realizamos nosotros va mucho más allá de los productos, dispositivos y proyectos que podemos encontrar en la actualidad.

Buscamos garantizar una comodidad y fiabilidad tanto en las acciones como en las mediciones de los niveles de la piscina, facilitar el control de la misma, mejorar el disfrute, tener un control absoluto sobre todo el sistema y a la larga un ahorro tanto en faena como en los diferentes productos que lleva el mantenimiento de una piscina.

Finalmente, nuestro proyecto, busca la unión de algunos de los productos actuales en los que cada uno desempeña una función diferente, en un mismo sistema con la facilidad de tenerlo todo controlado mediante una aplicación tanto para móvil como para ordenador, como para cualquier otro dispositivo con conexión a internet.

3. Análisis del problema

Como hemos visto, no existe ningún dispositivo único capaz de controlar a la vez todo lo planteado para este proyecto.

El proyecto que queremos llevar a cabo se trata de poder fabricar un sistema que se pueda integrar de manera fácil en cualquier sistema semejante que se quiera automatizar sin depender de ninguna marca.

El principal problema de los sistemas ya desarrollados y en venta en el mercado actual, es que se tratan de sistemas que al estar desarrollados por las propias empresas no dan la oportunidad de aplicarles o integrar dispositivos externos a esas empresas. Con nuestro proyecto intentaremos suplir este problema que se plantea con el desarrollo integral de un sistema automatizado, desarrollado en su totalidad con herramientas y software de código abierto, es decir, accesible para cualquier persona hoy en día.

3.1 Identificación y análisis de soluciones posibles

Para el desarrollo de este proyecto, se han valorado diferentes soluciones posibles y todas ellas con un funcionamiento correcto, pero una destaca más sobre las otras y explicamos el porqué de haber seleccionado una de las tres soluciones que aportamos.

- **Desarrollo del sistema automatizado utilizando soluciones hardware comerciales.**

Esta solución era una de las menos contempladas a realizar, pero debíamos barajar la posibilidad de crear todo el sistema automatizado utilizando productos ya disponibles en el mercado.

El principal problema que íbamos a encontrarnos, era el elevado precio de los elementos de control para cada objeto que existía en nuestro proyecto, debido a que las empresas ya te venden el producto para funcionar, bajo su software.

El segundo problema y más importante, era la centralización de todos los dispositivos desarrollados por empresas diferentes con softwares diferentes y no accesibles al código, al estar desarrollados por ellos mismos. Este problema era el mayor a solucionar, ya que con la compra de los dispositivos de automatización de forma separada podríamos alcanzar una automatización casi completa del sistema, pero no una funcionalidad adecuada, ya que no permite una fácil conexión de todos los elementos entre si y también dependen del fabricante.

Por estos motivos, finalmente descartamos esta solución.

- **Desarrollo de la aplicación cliente en IONIC y uso de Raspberry Pi.**

Una de las primeras soluciones posibles que planteamos y desarrollamos, sin tener en cuenta la anterior, fue hacer uso solamente de dispositivos Raspberry Pi junto con una aplicación desarrollada en IONIC.

Ionic se trata de un framework basado en el lenguaje de programación Angular y que utiliza estándares como HTML, CSS y Javascript, y está enfocado para el despliegue en aplicaciones en plataformas iOS y Android.

Inicialmente, esta opción era completamente válida y asequible, pero pudimos observar diferentes problemas a lo largo del desarrollo y las pruebas que realizamos.

Uno de los problemas más graves que encontramos, es que utilizando esta estructura conseguíamos tener un sistema descentralizado en el que cada Raspberry ejecutaba las ordenes tras hacerle una petición desde la aplicación, ocasionando así el no poder consultar en tiempo real como se encontraba el dispositivo, es decir, no podíamos saber el estado del mismo hasta que no se hiciera una petición de nuevo.

Por otra parte, un punto a favor de este desarrollo era que la propia aplicación desarrollada en IONIC, daba la facilidad de poder compilar la aplicación para dispositivos multiplataforma, ya fuera iOS, Android o web.

Al partir de una aplicación basada en Angular, debíamos cargar un archivo XML con toda la configuración de direcciones de todas las raspberry que se encuentran en el sistema, y al ejecutar cualquier acción desde la app, esta envía una petición HTTP POST a la raspberry encargada de dicha función ocasionando así retardos en el funcionamiento y una incertidumbre al no saber el estado actual del propio dispositivo.

Por último, estas raspberry cargan un script desarrollado en el lenguaje Python con Webpy, las cuales recibían las peticiones POST de la aplicación y ejecutaba las funciones necesarias para llevar a cabo la funcionalidad solicitada.

Como hemos indicado al principio, esta fue una de las primeras soluciones propuestas utilizadas y la cual fue descartada al desarrollar la nueva solución que proponemos en el siguiente apartado.

Otro gran problema que debíamos solventar, era facilitar y agilizar el despliegue de la aplicación a todos los dispositivos móviles, debido a que cada vez que quisiéramos añadir un dispositivo o cambiar la funcionalidad de cualquiera de ellos, era necesario volver a compilar toda la aplicación y volver a desplegarla mediante la tienda de aplicaciones y descargando el fichero en cada uno de los dispositivos móviles que fueran a intervenir en su uso.

3.2 Solución propuesta

- **Desarrollo de la aplicación cliente con Home Assistant, uso de Raspberry Pi y dispositivos Sonoff.**

Como anteriormente hemos indicado, esta solución es la actual y la más factible y con la que se ha desarrollado el proyecto. Los problemas que encontrábamos de descentralización en la anterior solución, los hemos podido solventar mediante el uso de la aplicación Home Assistant.

Home Assistant es un software gratuito y de código abierto que sirve para la automatización de cualquier sistema preparado para ello y diseñado para ser el sistema de control central para cualquier dispositivo inteligente.

Gracias a la utilización de Home Assistant como la aplicación centralizada, conseguimos nuestro propósito de crear un sistema centralizado capaz de realizar cualquier acción o respuesta del conjunto del sistema pasando por la plataforma que a su vez hará de servidor, por lo tanto, tanto las raspberry como la app de control serán clientes del servidor y de esta manera podremos saber en cualquier momento el estado del dispositivo sin problemas.

Esta aplicación se basa en un webview que conecta a la URL del servidor Home Assistant y muestra lo mismo que se vería desde un navegador.

Tanto las raspberry como los sonoff ejecutan un firmware que soporta la comunicación por mensajes MQTT. Los sonoff ejecutan el firmware Tasmota, el cual permite con una simple parametrización la comunicación con Home Assistant. Por otra parte, las raspberry ejecutarán un script desarrollado en python el cual comunica con Home Assistant mediante MQTT haciendo uso de la librería paho-mqtt, además se encarga del control local de los actuadores y sensores, haciendo transparente esta capa a Home Assistant.

Home Assistant, también da la posibilidad de realizar automatizaciones, tarea que no permitía realizar Ionic, como por ejemplo a cierta hora de la tarde que se encienda la bomba depuradora, es decir, podremos realizar un calendario de tareas.

Finalmente, también llegamos a solventar el problema de agilizar y facilitar los cambios en el sistema y en la aplicación sin necesidad de tener que desplegar ni compilar de nuevo la aplicación, debido a que esta iba directamente en un servidor.

3.3 Presupuesto sistema automatización piscina

En este apartado vamos a mostrar una tabla resumen del presupuesto total del proyecto en cuanto a materiales, esto siempre variara dependiendo del número de elementos que quieran colocarse en el sistema automatizado, para este proyecto que hemos desarrollado el coste en material es el siguiente:

Producto	**Coste Unidad**	**Cantidad**	**Total**
Sonoff	6.50 €	1	6.50 €
Fuente alimentación 24VAC	20 €	1	20 €
Fuente alimentación 12VDC	25 €	1	25 €
Raspberry Pi 3 B+	35 €	1	35 €
Placa PCB con componentes	25 €	1	25 €
Relés 24VAC	6 €	1	6 €
Switch	10 €	1	10 €
Electroválvulas	35 €	4	140 €
Luces RGB WS2815	35 €	4	140 €
Mano de obra (horas)	45 €	72	3240 €
		TOTAL:	**3647.50 €** IVA incluido

3.4 Presupuesto sistema automatización techo exterior abatible

En este apartado vamos a mostrar una tabla resumen del presupuesto total del proyecto en cuanto a materiales, esto siempre variara dependiendo del número de elementos que quieran colocarse en el sistema automatizado, para este proyecto que hemos desarrollado el coste en material es el siguiente:

Producto	**Coste Unidad**	**Cantidad**	**Total**
Sonoff	6.50 €	1	6.50 €
Fuente alimentación 12VDC	25 €	1	25 €
Fuente alimentación servo 5VDC	5 €	1	5 €
Raspberry Pi 3 B+	35 €	1	35 €
Llave de paso con servomotor	25 €	1	25 €
Tira led RGB 1 metro	7 €	9	63 €
Sensor de efecto hall	2 €	1	2 €
Bomba hidráulica	80 €	1	80 €
Mano de obra (horas)	45 €	40	1800 €
		TOTAL:	**2041.50 €** IVA incluido

4. Diseño de la solución

En este apartado, detallaremos el diseño de la solución final seleccionada y una vez tenemos claro lo que queremos abarcar en el proyecto, necesitamos especificar qué requisitos va a emplear nuestro sistema. A continuación, vamos a realizar un listado de todos los dispositivos que se emplean en nuestro proyecto.

4.1 Arquitectura del sistema de la piscina

En esta sección mostraremos un diagrama de bloques indicando claramente cómo se estructura todo el sistema de automatización de la piscina, para así identificar claramente los componentes esenciales en la arquitectura y la relación que tienen entre ellos.

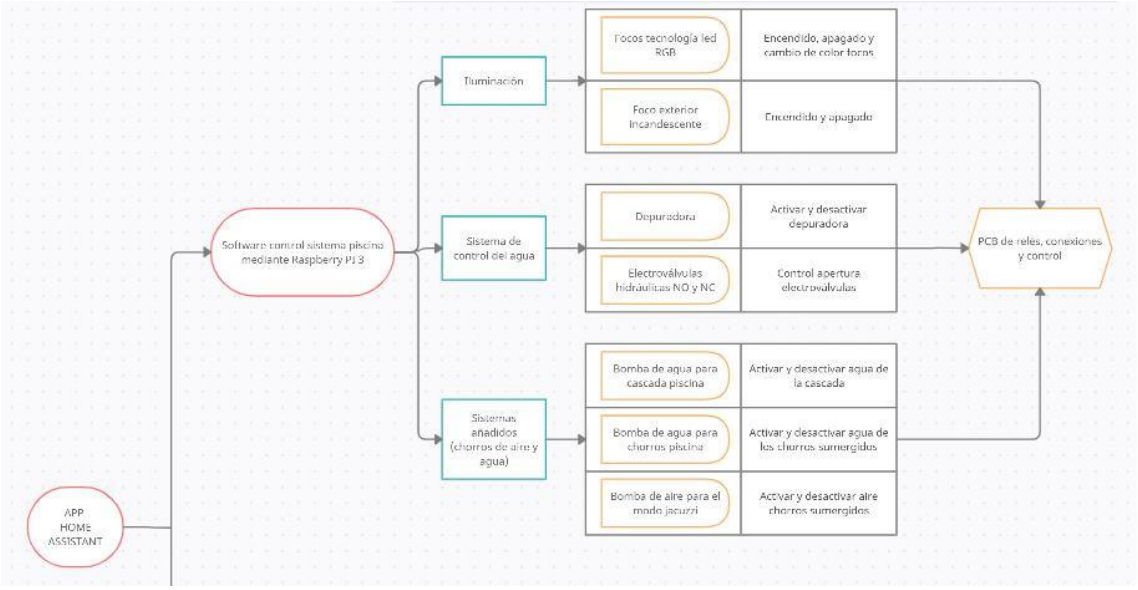


Figura 7 Diagrama de bloques sistema automatización piscina.

4.2 Arquitectura del sistema de techo exterior abatible

En esta sección mostraremos un diagrama de bloques indicando claramente cómo se estructura todo el sistema de automatización del techo exterior abatible, para así identificar claramente los componentes esenciales en la arquitectura y la relación que tienen entre ellos.

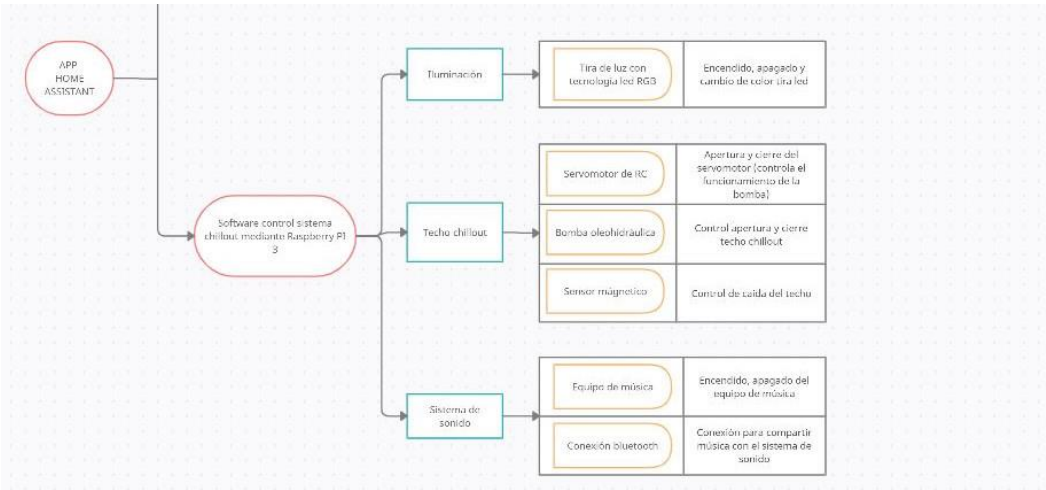


Figura 8 Diagrama de bloques sistema automatización techo exterior abatible.

4.3 Tecnología utilizada

En este apartado detallaremos en profundidad todas las herramientas software y tecnologías que se han empleado en el desarrollo del proyecto.

4.3.1 Dispositivos Raspberry PI

Como se ha explicado anteriormente, los dispositivos Raspberry Pi[4] son mini computadoras en las que se instala y se ejecuta un sistema operativo, en este caso, todas las raspberry de este proyecto han sido instaladas con el sistema operativo Raspbian, un sistema destinado para estos dispositivos y el oficial recomendado por el fabricante. Podríamos haber elegido cualquier otro sistema operativo, pero decidimos utilizar el recomendado, por estar basado en Debian bajo una distribución del sistema operativo GNU/Linux. Hemos decidido instalar una versión que no tiene interfaz gráfica, por lo tanto, todo el desarrollo, instalación y configuración ha de hacerse mediante el terminal y comandos, esto lo hemos hecho así debido a que no va a ser necesario acceder de continuo a este dispositivo una vez este configurado.

Cabe destacar, que este sistema operativo es de código abierto, lo que facilita mucho el acceso y desarrollo de aplicaciones para cualquier fin.

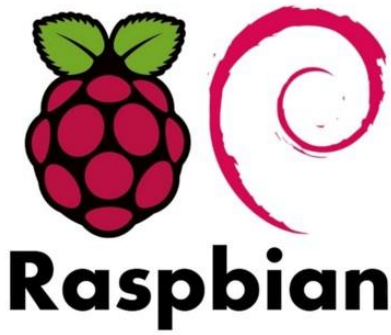


Figura 9 Logotipo del sistema operativo Raspbian

4.3.2 Dispositivo Raspberry PI Home Assistant

Este dispositivo es el principal, donde se encuentra instalada y desplegada la aplicación Home Assistant y la que hace la función de servidor, a la cual todos los sistemas envían las peticiones y la información y donde los dispositivos móviles se conectan.

A diferencia de las otras raspberry de todo el proyecto, esta será la única que lleve un sistema operativo diferente a las demás, y el cual, si tendrá entorno gráfico, necesario para la configuración de todo el servidor de Home Assistant.

El sistema operativo elegido para este dispositivo es el Home Assistant OS, que podemos encontrarlo en su propia página web[5], y es un sistema operativo optimizado para poner en marcha la aplicación con un consumo de recursos mínimo. Este paquete de instalación ya lleva pre instalados todos los add-ons y plugins y bases de datos recomendados por el fabricante de la aplicación.

Cabe destacar, que no es necesario hacerlo de esta forma ni instalar este sistema operativo, ya que Home Assistant te da la opción de instalar la aplicación libremente en cualquier otro sistema operativo, descargando el paquete de instalación de la misma.

Hemos optado por esta opción, ya que facilita mucho el trabajo de configuración y previene futuros errores causados por una mala instalación de cualquier componente o dependencia del mismo.

Figura 10 Logotipo del sistema operativo de Home Assistant

4.3.3 Lenguajes de programación

- **Python:**

Para un correcto funcionamiento y una configuración optima de todo el proyecto, es necesario hacer uso de scripts, que se ejecutarán en las propias raspberry cuando se les pida realizar cualquier acción y para llevar a cabo el desarrollo de estos scripts se ha utilizado el lenguaje de programación Python, el cual aporta una gran estabilidad, fácil legibilidad del código, programación imperativa, de código abierto y muy escalable, lo que facilita el desarrollo de las acciones que buscamos implementar.
Lo mejor de Python, es que es un lenguaje que permite plasmar o desarrollar las ideas complejas con pocas líneas de código y así conseguimos un menor tiempo de ejecución.

Figura 11 Logotipo lenguaje programación Python.

- **YAML:**

YAML es un formato de serialización de datos legibles, inspirado en otros lenguajes de programación como XML, C, Python y Perl.

Hemos utilizado este lenguaje de programación para el desarrollo y configuración de Home Assistant en la raspberry de Home Assistant.

Es necesario utilizar este lenguaje, ya que es el único disponible para poder realizar la configuración y despliegue de la aplicación.

```yaml
on:
  push:
    branches:
      - master
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v1
    - name: Install Node package
      run: npm install
    - name: Build the project
      run: npm run build
    - name: Deploy to S3
```

Figura 12 Estructura de un archivo YAML.

4.3.4 Entorno de desarrollo

Al tratarse de sistemas operativos basados en Linux y sin entorno gráfico, se han utilizado los editores de texto por consola, en estos casos el editor utilizado es el que lleva por defecto el propio sistema operativo y se trata del editor Nano. Este es un editor de texto de línea de comandos para sistemas Unix y Linux basado en curses.

Figura 13 Simulación del editor de texto GNU nano

4.3.5 Programas utilizados

Para el acceso a las raspberry y configuración del sistema y edición o subida de algunos scripts a las mismas, era necesario hacer uso de algunos programas de transferencia de datos para poder hacer la conexión a los diferentes dispositivos, para ello utilizábamos programas como Filezilla[6] para transferencia de archivos en el caso de que fuera necesario y para la conexión a las máquinas en las que se desplegaba todo el sistema automatizado utilizábamos el propio protocolo SSH que nos facilitaba Linux mediante la consola de comandos.

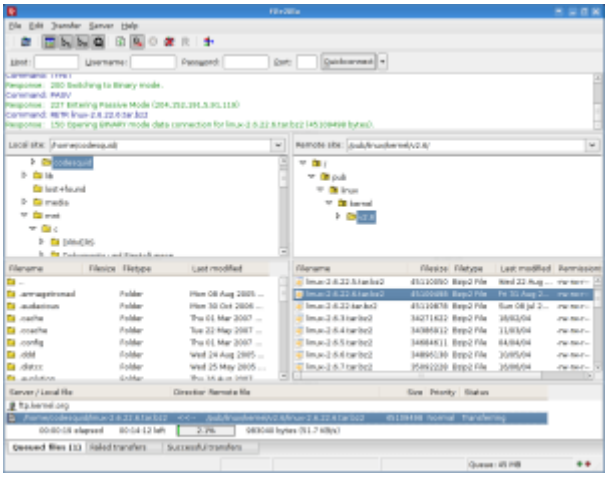


Figura 14 Interfaz de trabajo de Filezilla

Para poder cargar e instalar los sistemas operativos en los dispositivos Raspberry Pi necesitaremos hacer uso de la herramienta que nos proporciona la página oficial de Raspberry y que se llama Raspberry Pi Imager[7], valido tanto para Windows, Ubuntu y MacOS.

Figura 15 Interfaz Raspberry Pi Image

4.3.6 Bases de datos

Para el despliegue de la aplicación de Home Assistant, es necesario hacer uso de una base de datos que integre todas las tablas y registros necesarios para el correcto funcionamiento de la aplicación. Para este paso, hemos utilizado la propia base de datos que marca Home Assistant y que despliega la misma sobre SQLite.

SQLite se trata de un sistema de gestión de base de datos relacional y que depende de una biblioteca escrita en el lenguaje de programación C. Cabe recordar, que se trata de otra herramienta de software libre.

Se ha seleccionado finalmente este tipo de base de datos, debido a que permite almacenar información de dispositivos de una forma sencilla, rápida y eficaz en equipos con pocas capacidades de hardware, lo que es idóneo para nuestro proyecto desarrollado con raspberry.

Figura 16 Logotipo de SQLite

4.3.7 Librerías o paquetes

En este apartado, explicaremos las librerías que se han utilizado para el desarrollo de todo el conjunto del proyecto, indicando las fuentes de donde hemos extraído la información, junto con los propios comandos utilizados para la instalación y configuración de las mismas.

El uso de librerías que vamos a indicar, es necesario para poder desarrollar perfectamente el proyecto y algunos de ellos para facilitar la configuración e instalación de todo el sistema automatizado.

- **Paquete Pip3[8]:**

Se trata de una librería con la funcionalidad de instalar y administrar paquetes para Python, esta viene incluida en la versión de Python 3, pero es posible que el paquete de Python que instale no la lleve consigo.

```
sudo apt update && sudo apt install python3-pip
```

Con la instalación de esta librería ya podremos instalar paquetes y dependencias en Python utilizando el siguiente comando:

```
pip3 install NombreDeLaLibreria
```

- **Paquete paho-mqtt[9]:**

Con esta librería, buscaremos implementar en Python el protocolo MQTT, que se trata de un protocolo de comunicación o conectividad de máquina a máquina y nos permitirá que las aplicaciones, en este caso los dispositivos Raspberry, se puedan conectar con la Raspberry principal que contendrá a Home Assistant y a la cual deberemos enviar mensajes mediante el protocolo TCP/IP, con esto conseguiremos una comunicación cliente/servidor y resolveremos el problema de descentralización que nos ocurría con la otra opción comentada en un punto anterior.

Versión instalada 1.5.1

```
pip install paho-mqtt
```

- **Paquete RPi.GPIO[10]:**

La siguiente librería es fundamental, ya que se trata de un paquete destinado al uso de Python sobre las Raspberry. Su función, es poder controlar los pines GPIO desde los programas o script de Python de la propia Raspberry, y así poder enviar y recibir señales de estos mismos pines.

Versión instalada 0.6.3

```
pip install RPi.GPIO
```

- **Paquete rpi_ws281x[11]:**

Esta librería como la anteriormente nombrada también es fundamental, debido que de esta depende el funcionamiento de las luces RGB de la piscina que son el modelo RGB WS2815 y la cual necesitamos también para poder realizar la activación de las mismas desde la Raspberry.

Gracias a esta librería, podremos conseguir hacer funcionar los focos led RGB WS2815 de la piscina, ya que estos generalmente funcionan a 5V, lo que requiere que la señal de datos que se le envíe estén al mismo nivel, por lo tanto, esta librería nos convierte la salida de un pin GPIO/PWM de una Raspberry Pi a un voltaje más alto a través de un cambiador de nivel.

Versión instalada 4.1.0

```
pip install rpi_ws281x
```

4.4 Componentes del sistema automatización piscina

En este apartado, detallaremos todos los componentes que se utilizan y componen el proyecto de automatización de la piscina, tanto software como hardware.

4.4.1 Placa base Raspberry Pi

Este será el elemento fundamental del sistema, con el cual controlaremos toda la automatización de la piscina y recibirá los datos y los enviará a nuestra aplicación principal para poder leer y controlar todo el sistema. Se trata de un mini ordenador capaz de realizar todas las funciones necesarias mediante señales, scripts y actuadores.
En este caso en específico hemos optado por utilizar el dispositivo Raspberry Pi, se trata de un mini ordenador con la funcionalidad de poder instalarle un sistema operativo, que mediante scripts y programas será el encargado de enviar las ordenes de acción a los diferentes dispositivos que se conecten a los pines de datos.

Este dispositivo, ya lleva consigo un módulo WiFi y ethernet integrados en la placa, junto con su fuente de alimentación y los pines digitales, que serán los encargados mediante las reglas que se definan en el programa desarrollado e instalado dentro de la Raspberry Pi de ejecutar las órdenes para realizar las ordenes necesarias.

En esta primera parte del proyecto, utilizaremos la Raspberry Pi 3B+, ya que como hemos comentado en apartados anteriores, hacemos esta elección frente a la placa Arduino debido a que en esta placa no viene ningún módulo de WiFi, no dispone de sistema operativo para cargar los scripts mencionados para el desarrollo de las acciones y finalmente no es más económica que la Raspberry Pi, por lo tanto deberíamos adquirir por separado el módulo de WiFi e instalarlo en la placa Arduino, provocando así también la inutilización de pines de datos.

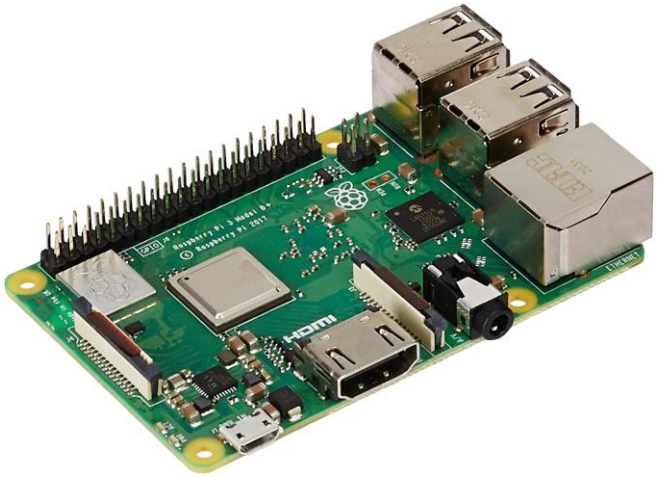


Figura 17 Raspberry Pi 3 B+

Especificaciones técnicas:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

4.4.2 Switch Ethernet

Este switch será el encargado de hacer la conexión a la red tanto del sistema de automatización de la piscina, como del sistema de automatización del techo exterior abatible, como de la otra parte del proyecto desarrollado por Pedro Amores Dolz en su propio trabajo final de grado.

Cualquier switch con 5 o más puertos es suficiente para realizar la tarea de enviar los datos de los sistemas a la aplicación principal y recibir de esta las acciones que realicemos. Nosotros hemos optado por el switch **SMC EZ Switch SMCFS5 - Conmutador - 5 x 10/100**.

Nos hemos decantado por este switch por el simple hecho del tamaño reducido que tiene y por su bajo coste, ya que como hemos explicado, cualquier switch cumpliría las funciones buscadas.

Figura 18 SMCFS5 Switch 5 x 10/100

4.4.3 Placa PCB

Este dispositivo esta creado desde cero en una placa PCB, y será el encargado de recibir los datos de la Raspberry Pi y de activar o apagar los relés encargados del funcionamiento de las luces RGB, luz exterior, ducha/cascada de la piscina, bomba de aire, bomba de agua, bomba depuradora y robot limpia fondos.

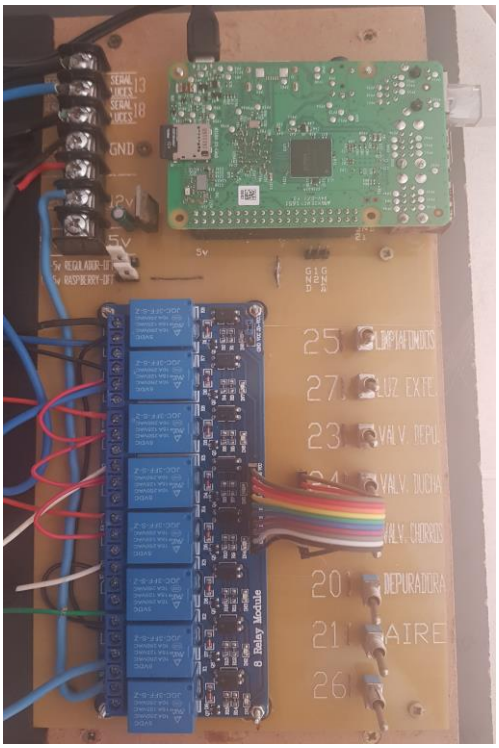


Figura 19 Circuito PCB Montado

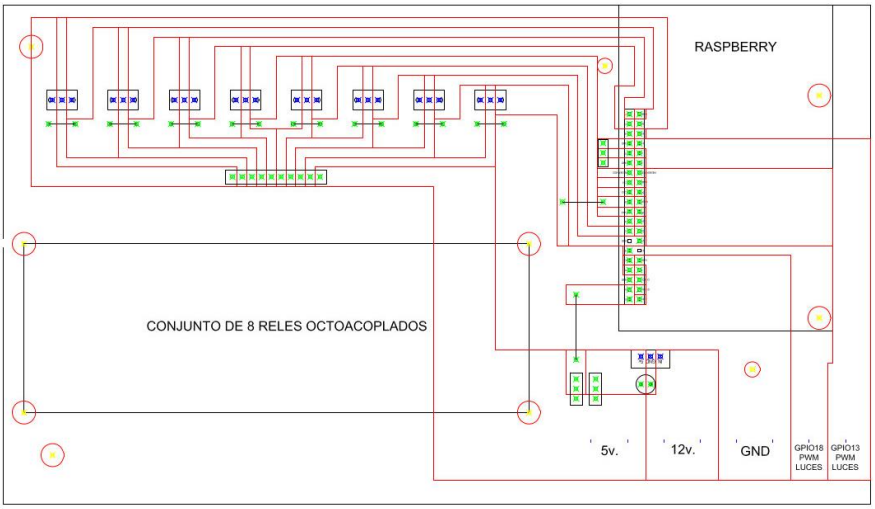


Figura 20 Esquema electrónico circuito PCB

Componentes:

- 8 Conmutadores (uno para cada acción).
- 8 Relés octoacoplados con bobina de 5V. JQC-3FF-S-Z.
- 1 Condensador de 25V de 1000 micro faradios.
- 1 Regulador de voltaje LM7805.
- 1 conjunto de conexiones de 7 elementos.
- 2 Jumpers.

4.4.4 Electroválvulas

Componente que utilizaremos para abrir o cerrar el llenado de la piscina, ducha/cascada, bomba de agua y la bomba de depuración, mediante la aplicación móvil. Este recibirá la orden del Sonoff o de la Raspberry Pi.

Figura 21 Electroválvula NC

4.4.5 Sonoff

El dispositivo Sonoff[12] es un interruptor inteligente que permite controlar el encendido o apagado de cualquier dispositivo que esté conectado a este mismo mediante una orden enviada por la red y que recibe mediante WiFi.

Figura 22 Sonoff Mini R2

Especificaciones técnicas:

- Tipo: Smart Switch module
- Voltaje: 100-240V AC 50/60Hz
- Carga máxima: 10A/2200W
- MCU: ESP8285

4.4.6 Fuente de alimentación

La fuente de alimentación, será la encargada de alimentar todo el circuito PCB montado específicamente para el sistema de automatización.

Figura 23 Fuente de alimentación LPF-60-12

Especificaciones técnicas:

- Modelo: LPF-60-12
- Voltaje de salida: 7.2 ~ 12 V
- Voltaje de entrada mínimo: 90VAC
- Voltaje de entrada máximo: 305VAC
- Potencia: 60W
- Temperatura de trabajo: -40°C ~ 80°C
- Eficiencia: 86%

4.4.7 Luces RGB WS2812

Los leds utilizados para la creación de los focos RGB de la piscina, se han llevado a cabo con el modelo WS2812. Son leds con un circuito de control integrado que incluye un puerto digital por donde envía y recibe los datos. Trabaja bajo el protocolo de comunicación NZR y con datos de 24bit.

Figura 24 Led RGB WS2812

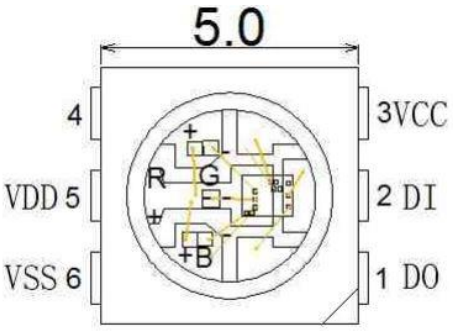


Figura 25 Esquema de conexiones del led WS2812

Especificaciones técnicas:

Prameter	Symbol	Ratings	Unit
Power supply voltage	V_{CC}	+6.0~+7.0	V
Power supply voltage	V_{DD}	+6.0~+7.0	V
Input voltage	V_I	-0.5~VDD+0.5	V
Operation junction temperature	Topt	-25~+80	℃
Storage temperature range	Tstg	-55~+150	℃

Electrical Characteristics（T_A=-20~+70℃， V_{DD}=4.5~5.5V,V_{SS}=0V,unless otherwise specified）

Prameter	Smybol	conditions	Min	Tpy	Max	Unit
Low voltage output current	I_{OL}	ROUT	——	18.5	——	mA
	I_{dout}	Vo=0.4V，D_{OUT}	10	——	——	mA
Input current	I_I	V_I=V_{DD}/V_{SS}	——	——	±1	μA
Input voltage level	V_{IH}	D_{IN}，SET	0.7V_{DD}	——	——	V
	V_{IL}	D_{IN}，SET	——	——	0.3 V_{DD}	V
Hysteresis voltage	V_H	D_{IN}，SET	——	0.35	——	V

Switching characteristics（T_A=-20~+70℃， V_{DD}=4.5~5.5V,V_{SS}=0V,unless otherwise specified）

Prameter	Symbol	Condition	Min	Tpy	Max	Unit
Operation frequency	*Fosc*2	——	——	800	——	KHz
Transmission delay time	t_{PLZ}	CL=15pF,DIN→ DOUT,RL=10KΩ	——	——	300	ns
Fall time	t_{THZ}	CL=300pF,OUTR/OUTG/OUTB	——	——	120	μs
Data transmission rate	F_{MAX}	Duty ratio50%	400	——	——	Kbps
Input capcity	C_I	——	——	——	15	pF

Figura 26 Tabla de especificaciones técnicas del led WS2812

4.4.8 Bomba de aire - 205BSP2T

Esta bomba de aire, interviene en el sistema en un segundo plano, ya que no depende de ella el funcionamiento del sistema, simplemente es un añadido que se ha puesto en el proyecto.

Figura 27 Bomba de aire 205BSP2T

4.5 Componentes del sistema automatización techo exterior abatible

En este apartado, detallaremos todos los componentes que se utilizan y componen el proyecto de automatización del techo exterior abatible, tanto software como hardware.

4.5.1 Placa base Raspberry Pi

En este sistema, utilizamos de placa base el mismo modelo de Raspberry Pi del punto 4.4.1, es decir, la misma placa que la utilizada en el sistema de automatización de la piscina.

4.5.2 Iluminación tira led RGB

A diferencia del sistema de automatización de la piscina, en este no es necesario hacer uso de leds independientes para crear un foco led, en este caso, utilizaremos una tira led RGB IP20 de exterior.

Figura 28 Tira led RGB IP20

4.5.3 Sensor de efecto Hall

Los sensores de efecto Hall, son sensores de proximidad magnéticos que trabajan por medio de la medición del voltaje cuando el dispositivo se coloca en un campo magnético.

Este sensor, será el encargado de controlar el límite de bajada y subida del techo, en el que el límite de bajada lo marcará el propio suelo y el límite de subida será un imán colocado en la posición desea, que al llegar a esa posición el sensor, este lo detectará y parará la subida del techo.

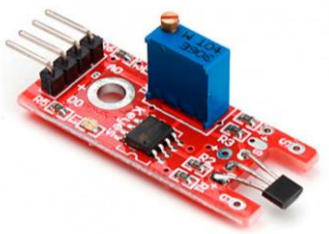


Figura 29 Sensor de efecto Hall

4.5.4 Llave de paso con servomotor

Será la encargada de activar el sistema de apertura o cierre del techo exterior. El servomotor es un dispositivo eléctrico que realiza la función de un motor, con la particularidad de que el eje de salida de este se puede mover tanto en ángulo, posición y velocidad, cosa que un motor normal no puede hacer.

La llave de paso se adaptará a este servomotor para abrir y cerrar el paso de líquido hidráulico a los brazos hidráulicos y así conseguir el movimiento de estos para abrir o cerrar el techo.

Figura 30 Servomotor

Figura 31 Llave de paso adaptada para servomotor

4.5.5 Fuente de alimentación 12VDC

Será la encargada de alimentar todo el sistema de automatización del techo exterior abatible. Como se puede comprobar en la imagen siguiente, utilizamos para ello una fuente de alimentación de ordenador modificada.

Figura 32 Fuente de alimentación mean well 12VDC

4.5.6 Sistema hidráulico

Este sistema, será el encargado de llevar a cabo tanto la apertura como el cierre del techo exterior. Consta de dos partes, la primera será la propia bomba hidráulica que hará la función de abrir o cerrar el paso de líquido hidráulico a los brazos y estos mismos serán la segunda parte del sistema y se encargarán de ejercer la fuerza para abrir y cerrar el techo exterior.

Figura 33 Bomba hidráulica

Figura 34 Brazo hidráulico

4.5.7 Sistema de audio

Sistema implementado en el proyecto, pero del cual no depende el funcionamiento de la automatización del mismo, simplemente se trata de un elemento añadido.

Figura 35 Sistema de audio XP-4080 y SAM 502

4.5.8 Sonoff

Este dispositivo es el mismo que se ha empleado en el sistema de automatización de la piscina, ya definido anteriormente en el punto 4.4.5.

5. Desarrollo de la solución propuesta

En este apartado explicaremos, de una manera más detallada y profunda, todo el sistema al completo, empezando por el sistema automatizado de la piscina, seguido del sistema automatizado del techo exterior abatible y terminando por el sistema de la aplicación Home Assistant.

5.1 Desarrollo sistema automatización piscina

Empezaremos con el desarrollo del sistema con más elementos automatizados. Para ello, empezaremos listando todos los dispositivos que intervendrán en este desarrollo como son:

- Raspberry Pi modelo 3 B+.
- Electroválvulas de líquidos.
- Switch.
- Fuente de alimentación.
- Placa PCB con sus respectivos componentes.
- Dispositivo Sonoff.
- Focos led RGB WS2815.
- Foco luz exterior.
- Bomba de aire.
- Bomba depuradora.
- Bomba de agua.

La idea principal de este sistema, es conectar todos los elementos como son la bomba de agua, la bomba de aire, la bomba depuradora, las luces y las electroválvulas a la placa PCB, la cual hará la función de englobar a todos estos elementos y esta placa conectarla a la Raspberry Pi para poder enviar y recibir los mensajes/acciones demandadas por el usuario.

5.1.1 Instalación y configuración de Raspberry Pi

5.1.1.1 Instalación

Para comenzar, será necesario descargar el sistema operativo en una tarjeta de memoria SD, para proceder a la instalación del mismo. Se aconseja que sea una tarjeta SD con bastante capacidad para no tener futuros problemas de espacio y que esta tarjeta sea de clase 10, para aumentar la eficacia tanto de la escritura como de la lectura.

Lo primero que debemos hacer será formatear la tarjeta SD con el sistema de archivos **Fat32**. Para llevar a cabo esta tarea, la realizaremos desde el sistema operativo Windows, con el cual podemos seguir diferentes caminos en los cuales el resultado será el mismo.

- Formatear la tarjeta SD mediante un programa externo como puede ser **HP USB Disk Storage Format Tool** o **SD Memory Card Formatter**, este último también es compatible con sistemas MacOS.
- Mediante el terminal de Windows y utilizando la herramienta de gestión de discos por línea de comandos que ya viene incluido en el propio sistema operativo. Deberemos acceder a la ventana de ejecutar de Windows mediante las teclas **Windows+R** y escribir en la casilla de texto **diskpart**.
  A partir de aquí, muestro los comandos necesarios para introducir en la consola de diskpart y llegar a formatear la tarjeta SD como deseamos.

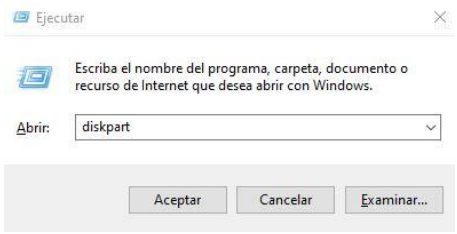


Ilustración 1 Ventana de ejecutar diskpart

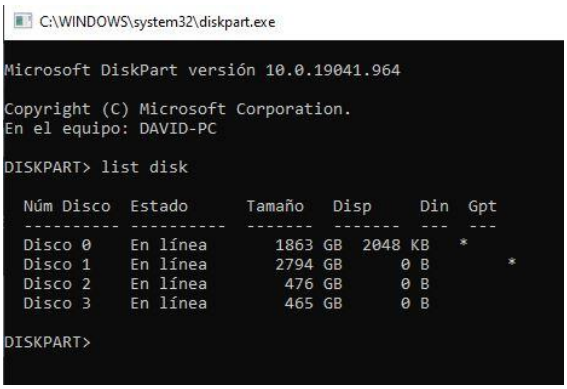


Ilustración 2 Interfaz de diskpart

Para listar todos los discos:

```
list disk
```

Para seleccionar la unidad SD:

```
select disk NÚMERO_DISCO
```

Para borrar el disco:

```
clean
```

Para crear una partición primaria en la tarjeta SD:

```
create partition primary
```

Finalmente, necesitaremos darle el formato que deseamos con el siguiente comando:

```
format fs=fat32 quick
```

Por último, tendremos esta tercera posibilidad, que será utilizando el gestor de archivos del propio equipo, donde nos aparecen todos los discos de nuestro ordenador y haciendo clic derecho sobre el mismo y seleccionando la opción de **Formatear**.

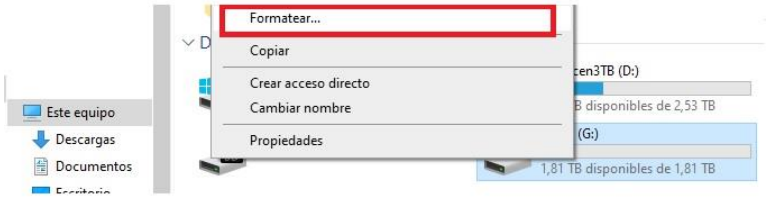


Ilustración 3 Captura para formatear mediante el sistema de archivos de Windows

Una vez tenemos ya formateada la tarjeta SD, deberemos cargar e instalar el sistema operativo en la misma, y deberemos descargar la imagen ISO del sistema operativo de la página oficial de Raspberry como hemos indicado en el punto 4.3.1 de este documento.

Para cargar e instalar el sistema operativo dentro de la Raspberry Pi, haremos uso del programa facilitado en la página oficial de Raspberry llamado Raspberry Pi Image.

Figura 36 Instalador de Raspbian en Raspberry Pi

Una vez tenemos el programa instalado, simplemente deberemos seleccionar la imagen del sistema operativo descargado, en este caso será el Raspbian 9.13 con el Kernel 4.14.79, tanto para la Raspberry Pi del sistema automatizado de la piscina, como para la del techo exterior abatible. Una vez hagamos esto, solo deberemos esperar a que se complete el proceso y ya tendremos instalado el sistema operativo en nuestros dispositivos.

5.1.1.2 Configuración

Completado el paso de la instalación, ahora deberemos acceder al sistema de archivos de la tarjeta SD y crear un archivo SSH en la partición **BOOT** que hemos creado en el comienzo del paso anterior, este archivo lo crearemos añadiendo el siguiente comando a la consola:

```
touch ssh
```

Con este sencillo archivo, ya tendremos nuestra Raspberry Pi lista para poder acceder a ella vía ssh.

Podemos realizar y activar la misma función tomando otro camino diferente, accediendo a la interfaz gráfica de la Raspberry mediante el puerto HDMI conectado a un monitor y colocando en la línea de comandos lo siguiente:

```
sudo raspi-config
```

Una vez introduzcamos ese comando, tendremos acceso al panel de herramientas de la configuración de la Raspberry Pi.

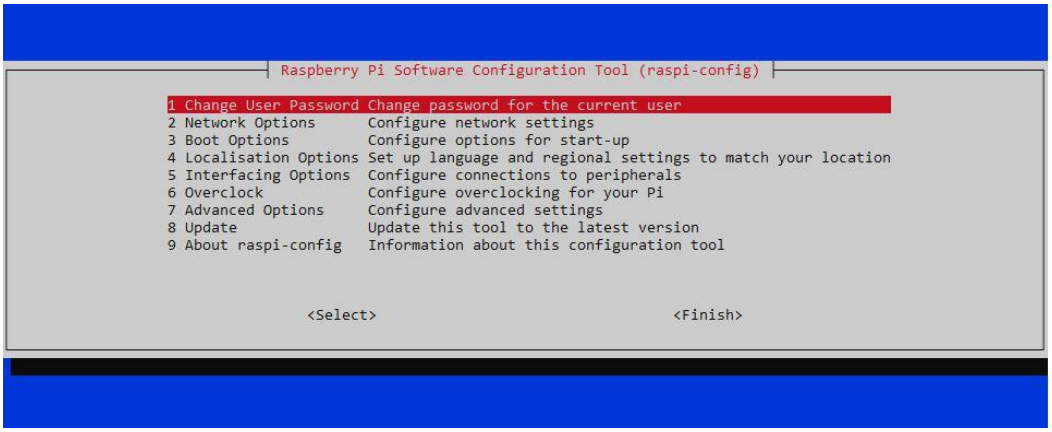


Figura 37 Panel de configuración de la Raspberry Pi

A partir de aquí, accederemos a la opción 5, Interfacing Options, una vez dentro deberemos entrar en la opción que pone SSH y activar la conexión ssh.

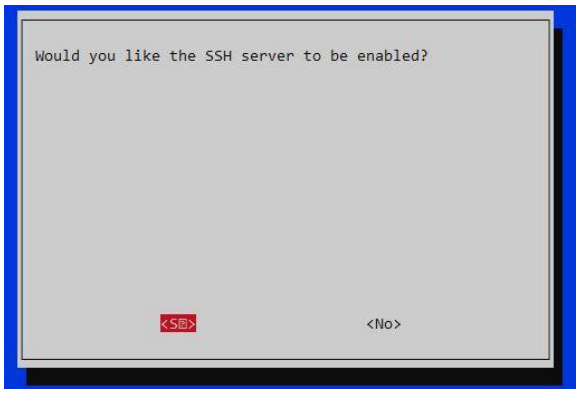


Figura 38 Activar la conexión ssh

Cabe destacar, que la opción que se ha seguido en el desarrollo del proyecto ha sido la de crear el archivo directamente en la raíz del sistema operativo, la manera más rápida, para así evitar tener que conectar el cable HDMI y tener que hacer un proceso distinto, pero con una misma finalidad.

Lo siguiente que debemos realizar, es conectar nuestra Raspberry Pi a nuestra red mediante la configuración del wifi de la misma. Para ello, debemos entrar en el sistema de archivos de la tarjeta SD y buscar dentro de la ruta **/etc/wpa-supplicant/** el archivo de configuración **wpa_supplicant.conf** y en el que deberemos añadir las siguientes líneas:

```
network = {
    ssid="nombre-red-wifi"
    psk="password-red-wifi"
    key_mgmt=WPA-PSK
}
```

Y cambiar el código de país al nuestro.

```
country = ES
```

```
pi@raspberrypi: ~
  GNU nano 2.7.4                  Fichero: /etc/wpa_supplicant/wpa_supplicant.conf

country=ES
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
        ssid="[redacted]"
        psk="[redacted]"
        key_mgmt=WPA-PSK
}
```

Figura 39 Archivo configuración Wifi Raspberry Pi

Una vez tenemos ya tenemos configurada la red wifi y el acceso ssh activado, debemos buscar en nuestra red la Raspberry Pi, haciendo uso del programa Advanced Ip Scanner, que nos facilitara la ip del dispositivo para acceder al mismo y continuar con la configuración.

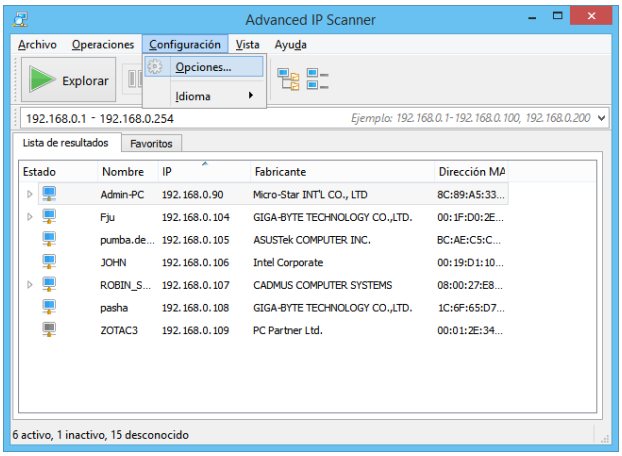


Figura 40 Interfaz de Advanced Ip Scanner

Con la dirección IP del dispositivo, podemos abrir nuestro terminal o consola de Linux y conectarnos al mismo mediante el comando ssh, en este caso, el comando a utilizar será:

```
ssh pi@10.0.0.91
```

Como se puede comprobar, ya sabemos la dirección IP y el usuario por defecto es **pi**, lo único que debemos hacer es crear una contraseña para este usuario y que se nos pedirá de aquí en adelante cada vez que nos conectemos a la misma.

Una vez dentro, debemos hacer uso de nuevo del comando de raspi-config para empezar a configurar la localización, el idioma y el timezone. Para llevar a cabo esta tarea, debemos acceder a la opción 4 del menú de configuración, **Localisation Options**, y realizar las siguientes configuraciones.

Para el idioma dejamos la opción por defecto de la Raspberry Pi.

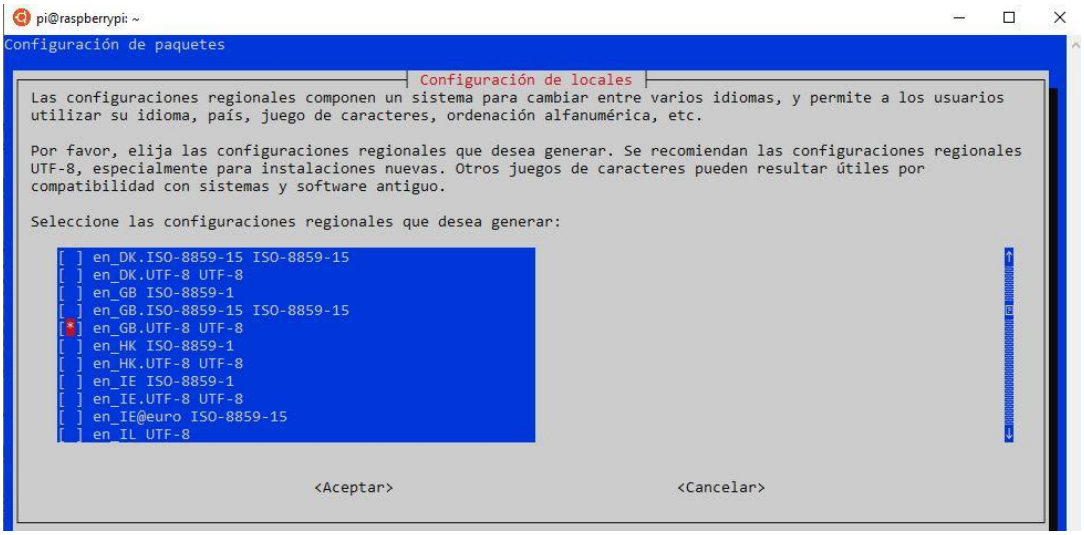


Figura 41 Menú idioma Raspberry Pi

Para la localización, timezone seleccionaremos Europa.

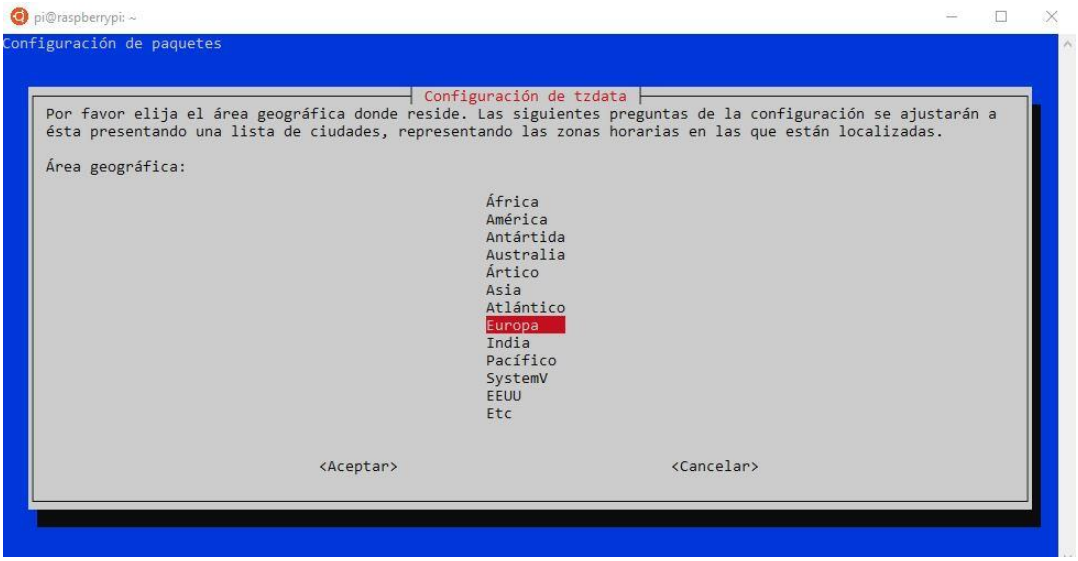


Figura 42 Menú localización Raspberry Pi

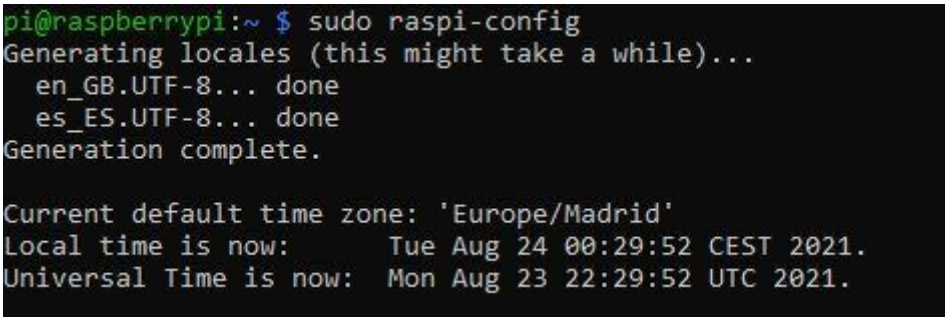


Figura 43 Resultado al seleccionar idioma y localización

Una vez realizada toda la configuración, procedemos a descargar y actualizar toda la lista de paquetes disponibles y sus versiones del sistema, esto lo haremos mediante los comandos:

```
sudo apt-get update

sudo apt-get upgrade
```

En este punto, ya tendremos el sistema de la Raspberry Pi configurado y actualizado y deberemos hacer este mismo desarrollo para la Raspberry Pi del sistema automatizado del techo exterior abatible.

5.1.2 Instalación de dependencias y librerías en la Raspberry Pi

En este punto, necesitaremos instalar todas las dependencias y librerías que utilizaremos para desarrollar la programación del sistema, y que será el principal eje del proyecto y el que dará la funcionalidad y las ordenes al sistema mediante scripts escritos con código Python.

Lo primero de todo, debemos instalar Python en nuestras Raspberrys Pi, para ello necesitaremos hacer uso de la librería o el paquete Python y descargarlo e instalarlo desde el repositorio universal, en el sistema operativo.

```
sudo apt install python3
```

El proyecto está desplegado con la versión **3.5.3-1** de Python.

Como explicábamos en el principio del punto 4.3.7, será necesario instalar el paquete **pip** de Python 3, que será necesario para poder instalar más adelante dependencias y otras librerías en Python, que serán de vital importancia para el desarrollo de los scripts.

```
sudo apt update

sudo apt install python3-venv python3-pip
```

Una vez instalado el paquete pip3, esto nos facilitará la instalación de otros paquetes ya que se trata de un sistema de gestión de paquetes que se utiliza para instalar y administrar paquetes de software escritos en el lenguaje Python.

Figura 44 Comprobación de la versión del paquete pip3 instalado.

Con la ayuda del paquete pip3, procedemos a instalar los paquetes que conformaran nuestro sistema de programación.

En primer lugar, será necesario instalar la librería **RPi.GPIO** en la versión 0.7.0 con el comando pip3.

```
sudo pip3 install RPi.GPIO
```

Este paquete será necesario para llevar a cabo el control de los pines GPIO en una Raspberry Pi desde el script desarrollado en Python.

Seguiremos con la instalación del paquete **rpi-ws281x** en la versión 4.2.5, este es necesario para realizar las instrucciones desde el script de Python y poder controlar las luces RGB WS2815 instaladas en la piscina. Se instala este paquete, debido a que las luces RGB que se utilizan, tienen el protocolo de trasferencia de datos NZR de 24bit, con este paquete podremos utilizar sus métodos y clases en el script y poder transferir la señal de datos en 24bit.

```
sudo pip3 install rpi-sw281x
```

En último lugar, será necesario instalar la librería paho-mqtt en la versión 1.5.1. Este paquete, será esencial para llevar a cabo la comunicación entre los sistemas y la conexión y envío de mensajes a la raspberry servidor en la que se ubicará la aplicación central Home Assistant.

```
sudo pip3 install paho-mqtt
```

5.1.3 Desarrollo y funcionalidad de la placa PCB

Debido a la complejidad del sistema, ha sido necesario crear un circuito electrónico en una placa PCB, con diferentes componentes, como relés, conmutadores, leds de señal, base de conexiones y la propia Raspberry Pi.

Realmente la funcionalidad de la PCB no es otra que la de facilitar las conexiones entre la raspberry y los elementos finales que la misma controla. En ella se encuentran, además de la raspberry, los siguientes elementos:

- Una fila de 8 conectores de tornillo, en los que se conecta, en uno de ellos, la señal directa de la salida de un GPIO de la raspberry a la señal de los cuatro focos RGB de la piscina, que se han fabricado con leds SMD del modelo WS2812 y que se alimentan directamente con 12 VDC, para lo que existe dos conectores únicamente puenteados entre ellos, para facilitar la conexión de la fuente de 12VDC con los focos y otros dos conectores igualmente puenteados entre ellos para conectar la GND (tierra o masa) de la fuente con la de los focos, pero además, en esta ocasión esta GND queda conectada internamente en la PCB con la GND de la raspberry, dado que si no fuera así los leds no reconocerían la señal de 3.3V que le envía la raspberry, que está alimentada desde otra fuente expresa para ella. Además, esta GND está conectada con los conmutadores colocados en la PCB que en el siguiente apartado mencionaremos.

  La mencionada señal que reciben los leds de los focos de la piscina debe ser de entre 3 y 5 voltios y sirve para definir la luminosidad de cada uno de los tres colores básicos, de cada uno de los 44 leds que componen cada foco, en cascada, con lo que queda definido el color y la intensidad de los focos, que funcionan todos ellos en paralelo, o sea que la señal que manda la raspberry a los cuatro focos es única.

  Por último, hay dos conectores de tornillo más en la fila, marcados con 5V que sirven para tener disponible 5V para cualquier necesidad imprevista o futura, a partir de un regulador de voltaje LM7805 colocado en la PCB, junto a su condensador de 16V 470 µf, que se alimenta de los 12 voltios mencionados en el apartado anterior. Mediante un jumper de la PCB se consigue alimentar con estos 5V o con los suministrados a través de un pin de la raspberry, los ocho relés referidos en el párrafo siguiente.

- Una placa con ocho relés con entradas octoacopladas, para evitar que una posible falla de cualquier relé, debido a la fuerza contra electromotriz que pueden generar sus bobinas, pueda dañar a través del GPIO conectado la propia raspberry.

  Estos relés se activan a través de los distintos GPIO’s conectados directamente a cada uno, con una señal LOW, (0 voltios, o GND, o inferior a 0.7V). Cada uno de estos relés dispone de tres conexiones en su salida, que facilitan una salida normalmente abierta (NO), o una salida normalmente cerrada (NC). En esta salida se conecta la alimentación que interesa en cada caso, por ejemplo, 220VAC para el foco piscina, o 24VAC para los relés del cuadro de protección general que comandan la bomba de aire o la bomba de agua, o 12VDC para comandar las electroválvulas.

  La alimentación de las bobinas de estos relés es de 5VDC, como se ha mencionado antes, suministrados por el mencionado regulador de voltaje o de la propia raspberry (según se encuentre situado el jumper), mientras que la señal para desactivarlos es de entre 3VDC a 5VDC, proporcionados por los GPIO, configurados como salida de la raspberry, que es de 3,3VDC, suficientes para desactivarlos, ya que estos relés se activan por bajo (LOW), por lo que las salidas GPIO están siempre en alto (HIGH), excepto cuando mandan una orden, que lo hacen situándose en bajo (LOW).

- Por último, en la PCB se han incluido ocho conmutadores de palanca, señalados con el número de GPIO que comanda el mismo relé que el conmutador y el nombre del aparato o elemento comandado.

El funcionamiento de estos conmutadores es bien sencillo, ya que conmutan la entrada de la señal de cada relé entre la salida del GPIO pertinente o el GND de la PCB, que recordemos es lo que activa el relé.

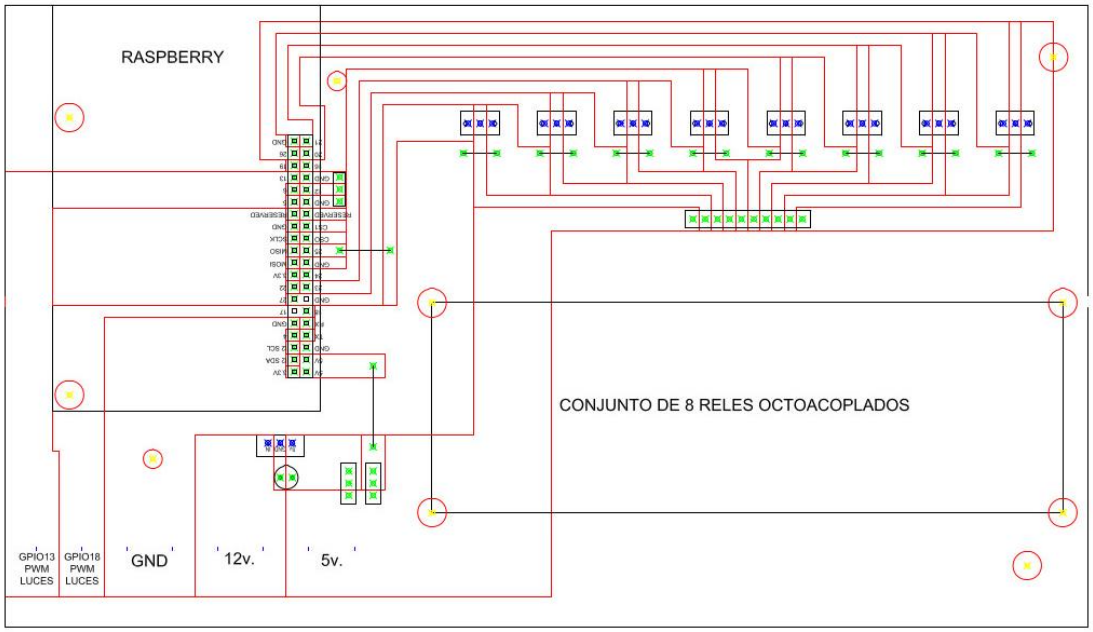


Figura 45 Esquema final circuito electrónico PCB

Figura 46 Pines de conexión Raspberry Pi

5.1.4 Desarrollo del software de automatización

Para empezar el desarrollo del software que controlará todas las acciones del sistema y la automatización del mismo, se ha creado un programa escrito en Python llamado **mqtt.py** en la raíz del sistema.

```
pi@raspberrypi:~ $ ls
cleaner_off.py  cleaner_on.py  mqtt.py
pi@raspberrypi:~ $
```

Figura 47 Ubicación del programa desarrollado en Python

Empezaremos importando todas las librerías y dependencias que utilizaremos en el software, haciendo uso del **import**.

A continuación, declaramos las variables necesarias que utilizaremos para darles el valor que corresponda al pin de la Raspberry, por el cual le enviará y recibiremos los datos de señal. Para ello, crearemos un array en el que guardaremos todos los pines declarados para poder utilizar en las diferentes funciones que definiremos más adelante.

Estos pines, que son las salidas GPIO de la propia raspberry pi, deberemos declararlos e iniciarlos al principio de la ejecución del software en un estado True, lo que significa que estarán disponibles por defecto, desde el arranque del sistema y declararlos a su vez como salidas de señal mediante el **GPIO.setup(NUMERO_PIN, GPIO.OUT)**.

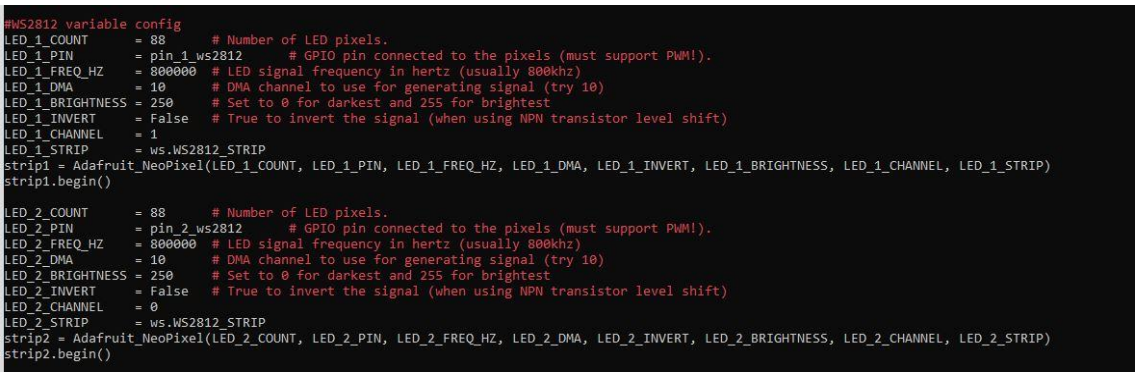
```
pin_1_ws2812 = 13
pin_2_ws2812 = 18
pin_power = 12
pin_relay = []
pin_relay.append(21)
pin_relay.append(20)
pin_relay.append(16)
pin_relay.append(24)
pin_relay.append(23)
pin_relay.append(27)
pin_relay.append(25)
pin_relay.append(26)

GPIO.setmode(GPIO.BCM)
GPIO.setup(pin_1_ws2812, GPIO.OUT)
GPIO.setup(pin_2_ws2812, GPIO.OUT)
GPIO.setup(pin_power, GPIO.OUT)
GPIO.setup(pin_relay[0], GPIO.OUT)
GPIO.setup(pin_relay[1], GPIO.OUT)
GPIO.setup(pin_relay[2], GPIO.OUT)
GPIO.setup(pin_relay[3], GPIO.OUT)
GPIO.setup(pin_relay[4], GPIO.OUT)
GPIO.setup(pin_relay[5], GPIO.OUT)
GPIO.setup(pin_relay[6], GPIO.OUT)
GPIO.setup(pin_relay[7], GPIO.OUT)

#Relay output init
GPIO.output(pin_relay[0], True)
GPIO.output(pin_relay[1], True)
GPIO.output(pin_relay[2], True)
GPIO.output(pin_relay[3], True)
GPIO.output(pin_relay[4], True)
GPIO.output(pin_relay[5], True)
GPIO.output(pin_relay[6], True)
GPIO.output(pin_relay[7], True)
```

Figura 48 Declaración e inicialización de variables del software de control de la piscina

Con el uso de focos led RGB WS2815, será necesario configurar las variables que intervendrán en el funcionamiento de los mismos y a los que habrá que indicarles e inicializar unas variables como lo son, la señal de frecuencia del led en hercios (Hz), nivel de brillo, canal de comunicación, número de leds, canal DMA utilizado, y el pin al que están conectados en la raspberry pi.



```
#WS2812 variable config
LED_1_COUNT      = 88      # Number of LED pixels.
LED_1_PIN        = pin_1_ws2812      # GPIO pin connected to the pixels (must support PWM!).
LED_1_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
LED_1_DMA        = 10      # DMA channel to use for generating signal (try 10)
LED_1_BRIGHTNESS = 250     # Set to 0 for darkest and 255 for brightest
LED_1_INVERT     = False   # True to invert the signal (when using NPN transistor level shift)
LED_1_CHANNEL    = 1
LED_1_STRIP      = ws.WS2812_STRIP
strip1 = Adafruit_NeoPixel(LED_1_COUNT, LED_1_PIN, LED_1_FREQ_HZ, LED_1_DMA, LED_1_INVERT, LED_1_BRIGHTNESS, LED_1_CHANNEL, LED_1_STRIP)
strip1.begin()

LED_2_COUNT      = 88      # Number of LED pixels.
LED_2_PIN        = pin_2_ws2812      # GPIO pin connected to the pixels (must support PWM!).
LED_2_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
LED_2_DMA        = 10      # DMA channel to use for generating signal (try 10)
LED_2_BRIGHTNESS = 250     # Set to 0 for darkest and 255 for brightest
LED_2_INVERT     = False   # True to invert the signal (when using NPN transistor level shift)
LED_2_CHANNEL    = 0
LED_2_STRIP      = ws.WS2812_STRIP
strip2 = Adafruit_NeoPixel(LED_2_COUNT, LED_2_PIN, LED_2_FREQ_HZ, LED_2_DMA, LED_2_INVERT, LED_2_BRIGHTNESS, LED_2_CHANNEL, LED_2_STRIP)
strip2.begin()
```

Figura 49 Configuración variables WS2815

```
LED_1_COUNT= 88    # Number of LED pixels.
LED_1_PIN= pin_1_ws2812  # GPIO pin connected to the pixels (must support PWM!)
LED_1_FREQ_HZ= 800000    # LED signal frequency in hertz (usually 800khz)
LED_1_DMA = 10       # DMA channel to use for generating signal (try 10)
LED_1_BRIGHTNESS = 250    # Set to 0 for darkest and 255 for brightest
LED_1_INVERT= False # True to invert the signal (when using NPN transistor level shift)
LED_1_CHANNEL= 1
LED_1_STRIP= ws.WS2812_STRIP
strip1 = Adafruit_NeoPixel(LED_1_COUNT, LED_1_PIN, LED_1_FREQ_HZ,
LED_1_DMA, LED_1_INVERT, LED_1_BRIGHTNESS, LED_1_CHANNEL, LED_1_STRIP)
strip1.begin()

LED_2_COUNT= 88    # Number of LED pixels.
LED_2_PIN= pin_2_ws2812  # GPIO pin connected to the pixels (must support PWM!)
LED_2_FREQ_HZ= 800000    # LED signal frequency in hertz (usually 800khz)
LED_2_DMA= 10       # DMA channel to use for generating signal (try 10)
LED_2_BRIGHTNESS = 250    # Set to 0 for darkest and 255 for brightest
LED_2_INVERT= False # True to invert the signal (when using NPN transistor level shift)
LED_2_CHANNEL= 0
LED_2_STRIP= ws.WS2812_STRIP
strip2 = Adafruit_NeoPixel(LED_2_COUNT, LED_2_PIN, LED_2_FREQ_HZ,
LED_2_DMA, LED_2_INVERT, LED_2_BRIGHTNESS, LED_2_CHANNEL, LED_2_STRIP)
strip2.begin()
```

Como se puede observar en el recuadro de fragmento de código anterior, se declaran dos grupos de variables, uno para el grupo LED_1 y otro para el grupo LED_2, esto es así, ya que dividimos los 4 focos led de la piscina en 2 grupos.

- **LED_1_COUNT**, hace referencia al número de leds que componen el foco, en este caso al tratarse de 2 focos, cada uno contiene 44, haciendo un total de 88 leds a controlar.
- **LED_1_PIN**, indica el número de pin al que se encuentra conectado el conjunto de focos en la raspberry pi, en el punto anterior, se detallan las variables que hacen referencia a cada pin con un nombre específico.
- **LED_1_FREQ_HZ**, se indica a la frecuencia que trabajará la señal de los focos, por defecto se deja en 800000.
- **LED_1_DMA**, deberemos decirle que canal de acceso a memoria directo se utilizara para así leer o escribir en la memoria principal del sistema.
- **LED_1_BRIGHTNESS**, se utiliza para indicar el nivel de brillo o intensidad con el que queremos que funcionen nuestros focos, siendo 0 nada de brillo y 255 el nivel de brillo máximo.
- **LED_1_INVERT**, esta variable se dejará en False, ya que no estamos utilizando en el circuito electrónico ningún transistor de tipo NPN para el control de los focos.
- **LED_1_CHANNEL**, aquí simplemente debemos indicar por el canal de comunicación que se transmitirán los datos.
- **LED_1_STRIP**,
- **strip1**, en esta variable se declara e inicializa un objeto **Adafruit_NeoPixel** con todos los parámetros definidos anteriormente. Este objeto procede de la librería anteriormente instalada.

Lo siguiente que hacemos, es crear una clase piscina (**class pool**), en la cual definiremos todas las variables en false, y una variable por cada componente que interviene en el sistema de automatización.

En este caso, inicializaremos los focos rgb a 0, ya que se tratan de valores enteros y la bomba de agua (**st_water**), la bomba de aire (**st_air**), el robot limpia fondos (**st_robot**), la cascada (**st_shower**), el foco de luz exterior (**st_lightoutdoor**) y la bomba depuradora (**st_cleaner**) en un valor False.

```
class pool:
    st_red = 0
    st_green = 0
    st_blue = 0
    st_white = 0
    st_water = False
    st_air = False
    st_cleaner = False
    st_shower = False
    st_lightoutdoor = False
    st_robot = False
```

Figura 50 Declaración de variables en la clase pool

A partir de este punto del software, empezaremos a declarar todas las funciones que controlarán y enviarán información a cada dispositivo respectivamente. Todas estas funciones las declararemos con el decorador de método estático de Python (**@staticmethod**).

- **Función __init__(self)**
  Se utilizará para inicializar la clase y crear el objeto pool.

- **Función power_on()**
  En esta función le enviaremos una señal al pin_power de la raspberry pi con un valor True, para poner en marcha el sistema.

- **Función power_off()**
  En esta función le enviaremos una señal al pin_power de la raspberry pi con un valor False, para poner apagar el sistema.

- **Función check_power()**
  En esta función comprobaremos el estado de cada componente del sistema y se llamará desde cada función, para así poder enviar a la aplicación el estado actual en el que se encuentra ese componente en el sistema, ya sea apagado o encendido, para poder mostrarlo al usuario.

- **Funciones setAirPump(value), setWaterPump(value), setShower(value), setCleaner(value), setRobot(value) y setLightOutdoor(value)**
  Todas estas funciones, se engloban en un mismo apartado debido a que la funcionalidad que desarrollan es la misma para cualquier componente relacionado con el sistema. Todas ellas, reciben un valor (value), este mismo valor lo casteamos a un entero (int(value)) y lo almacenaremos en la variable correspondiente que hemos declarado al principio de la clase pool. Una vez guardado el valor, le enviamos la señal (el valor) al pin de la raspberry que controle ese dispositivo mediante la orden GPIO.output, y por último, llamamos a la función check_power() para actualizar su estado en la aplicación y así poder mostrar el estado actual del dispositivo a tiempo real.

- **Función setRGBColor(value)**
  Esta función recibirá un mensaje con un valor que englobará a su vez 3 valores separados por comas, haciendo referencia el primer valor que define la cantidad de rojo, el segundo valor a la cantidad de verde y el tercer valor a la cantidad de azul. Esto es así porque estamos tratando con unos focos RGB (red, green, blue).
  Para poder tratar estos valores y devolver un código de color a los focos RGB, será necesario separar estos mismos con la función **decode()** y **split()**, para dividir los valores y tratar cada uno por separado.
  Finalmente, debemos realizar una operación para calcular el tanto por cien y poder enviar cada código de color a protocolo de los focos RGB.

```
@staticmethod
def setRGBColor(value):
    colors = value.decode().split(',')
    pool.st_red = round(int(colors[0]) / 255.0 * 100.0)
    pool.st_green = round(int(colors[1]) / 255.0 * 100.0)
    pool.st_blue = round(int(colors[2]) / 255.0 * 100.0)
```

Figura 51 Función de selección de color de los focos RGB

Como se puede observar, hacemos uso de la función **round()**, ya que el resultado que nos devuelva la operación deberá ser un entero, el sistema no admite ningún número con decimales (float).

- **Función setRGBPower(value)**
  En esta función, primero comprobamos si se cumple la condición de que los focos están activos, y esto lo conseguimos haciendo una suma de todos los valores que tienen las variables st_red, st_green y st_blue, si esta suma es mayor a 0 significa que están encendidas, debido a que se habían inicializado a 0 al principio del programa. Si la condición se cumple, enviamos una señal de apagado al pin de la raspberry correspondiente y volvemos a resetear los valores de color a 0 para mantener los focos apagados como en el estado inicial. Para resetear de nuevo a los valores iniciales, hacemos uso de la función **wsColorWipe()** que más abajo se explica.

```
@staticmethod
def setRGBPower(value):
    if int(value):
        if (pool.st_red + pool.st_green + pool.st_blue) > 0:
            GPIO.output(pin_relay[7], False)
            pool.wsColorWipe(strip1, Color(pool.st_red, pool.st_green, pool.st_blue), 0)
            pool.wsColorWipe(strip2, Color(pool.st_red, pool.st_green, pool.st_blue), 0)
            pool.check_power()
    else:
        GPIO.output(pin_relay[7], True)
```

Figura 52 Función de encendido de los focos RGB

- **Función setRGBBrigthness(value)**
  Esta función esta creada, por si en un futuro quisiéramos cambiar la intensidad de color/brillo de los focos led, actualmente no la utilizamos porque siempre buscamos que los focos tengan la mayor potencia de brillo posible, por eso al principio del programa se inicializan los focos con un valor en la variable de brillo de 250.

```
@staticmethod
def setRGBBrigthness(value):
    if int(value):
        pass
    pass
```

Figura 53 Función de brillo de los focos RGB

- **Función wsColorWipe(strip, color, wait_ms=50)**
  Por último, esta función será la última que intervendrá en el funcionamiento de los focos RGB de la piscina y lo único que realizamos con esta es volver a poner a 0 el color de todas las luces para así resetearlos y tener el estado inicial.

```
@staticmethod
def wsColorWipe(strip, color, wait_ms=50):
    for ie in range(20):
        for i in range(strip.numPixels()):
            strip.setPixelColor(i, color)
        strip.show()
        time.sleep(wait_ms/1000.0)
```

Figura 54 Función de reseteo del color de los focos RGB

Una vez controlamos todo el sistema y cada dispositivo por separado con su función correspondiente, es necesario que este sistema automatizado, es decir, que esta Raspberry Pi pueda enviar los estados y recibir y ejecutar las órdenes enviadas desde la aplicación a cada dispositivo conectado a esta Raspberry Pi.

Para lograr esto, se han creado dos funciones, una recibirá la respuesta del servidor, es decir, la acción pedida por el usuario mediante la aplicación que será enviada a la Raspberry Pi y esta llevará a cabo la tarea, y una segunda función que será la responsable de enviar la respuesta después de haber ejecutado alguna acción en cualquier dispositivo al servidor, es decir, a la aplicación Home Assistant.

Todas estas conexiones se realizan mediante una conexión MQTT como bien se ha explicado a lo largo del proyecto.

- **Función on_connect(client, userdata, rc, properties=None)**
  Se trata de la función de conexión del sistema automatizado, la Raspberry Pi, con el servidor donde se encuentra instalado Home Assistant. Este ejecutará el broker de MQTT para realizar la conexión y comenzar la comunicación entre los sistemas.

```
## MQTT connection
# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, rc, properties=None):
    print('Connected with result code ' + str(rc))
    for topic in devices:
        client.subscribe(topic)
        print('subscribe: '+str(topic))
        client.publish(devices[topic]['publish'], "0")
```

Figura 55 Función de la respuesta del servidor Home Assistant

- **Función on_message(client, userdata, msg)**
  Esta función será la encargada de enviar las peticiones realizadas en la aplicación a las diferentes Raspberry Pi con el uso del tópico correspondiente para cada sistema de automatización, de esta forma, cada Raspberry Pi, sabrá que mensaje debe resolver.

```
# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    getattr(pool, devices[msg.topic]['function'])(msg.payload)
    client.publish(devices[msg.topic]['publish'], msg.payload)

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.username_pw_set(username="[redacted]", password="[redacted]")
client.connect("10.0.0.89", 1883, 60)
```

Figura 56 Función del mensaje enviado al servidor

Una vez terminado el programa, deberemos crear un servicio en el sistema operativo de las Raspberry Pi, para que ejecute este programa cada vez que se enchufe la Raspberry, o en caso de perder la conexión de internet que intente reconectar el servicio y así tenerlo en funcionamiento siempre.

Para ello, debemos crear este servicio con SYSTEMD. Este paso lo llevamos a cabo, creando un archivo en la ruta **/etc/systemd/system/mqtt.service** y el archivo debe contener el siguiente código:

```
[Unit]
Description=MQTT Service
Wants=network.target
After=network.target

[Service]
Type=simple
User=root
WorkingDirectory=/home/pi
ExecStart=/usr/bin/python3 /home/pi/mqtt.py
Restart=on-failure
RestartSec=5s

[Install]
WantedBy=multi-user.target
```

Figura 57 Código del servicio MQTT

5.1.5 Conexiones del sistema

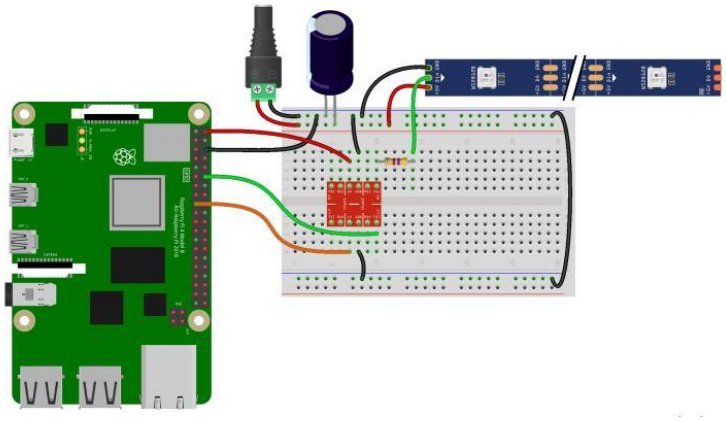


Figura 58 Diagrama de conexiones de los focos RGB simplificado

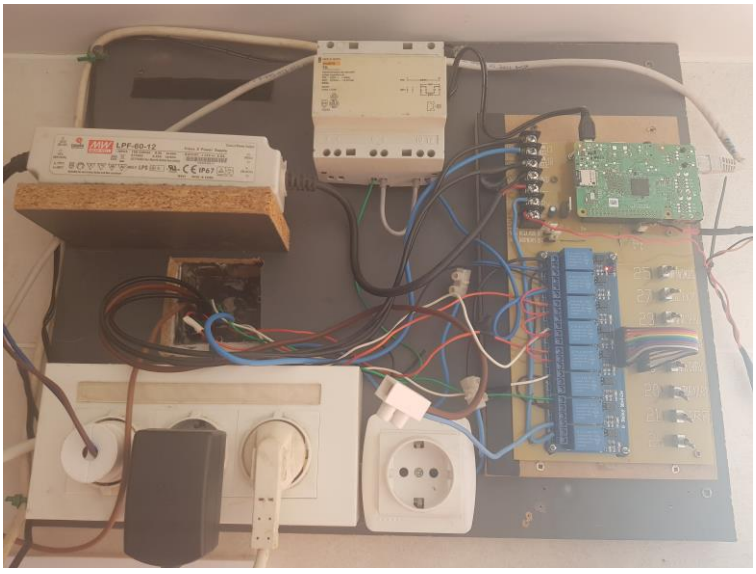


Figura 59 Conexiones del sistema de automatización de la piscina

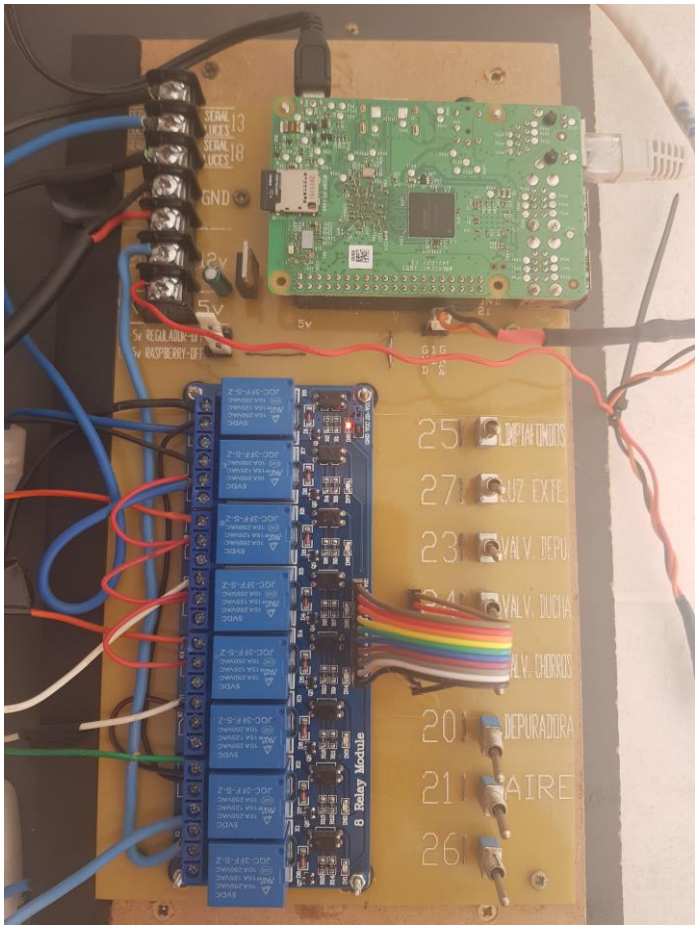


Figura 60 Conexiones placa PCB con la Raspberry Pi

Figura 61 Switch con conexiones

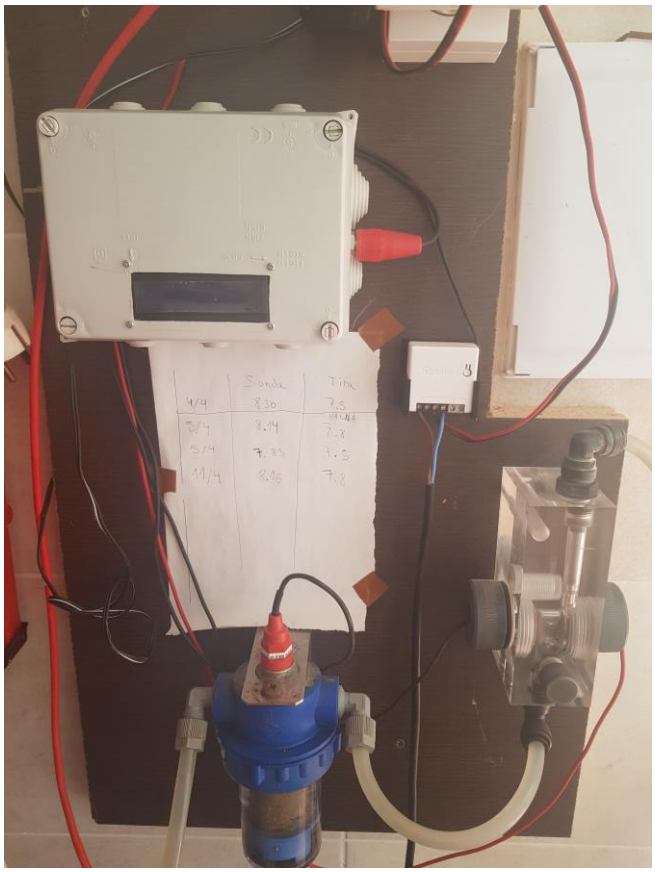


Figura 62 Sistema de control de pH y cloro de la piscina

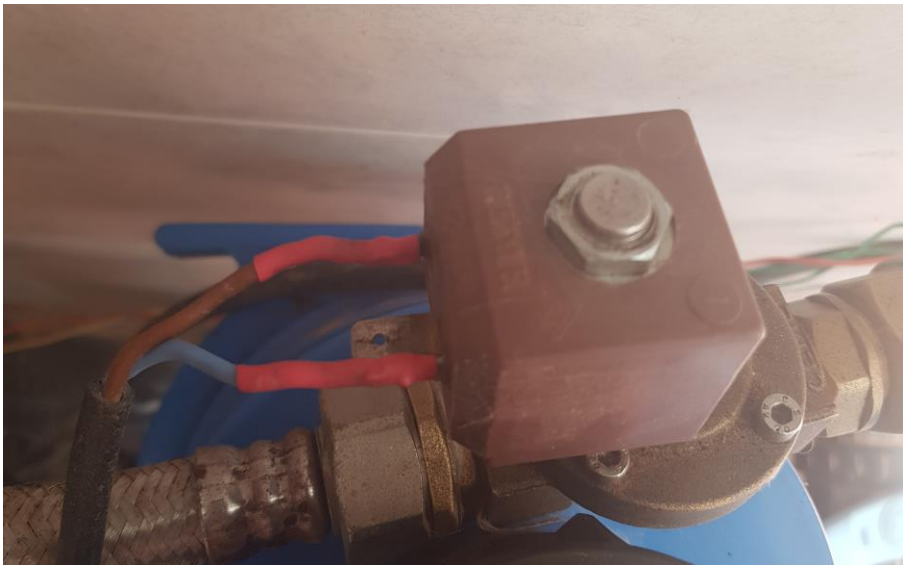


Figura 63 Conexiones electroválvula

5.2 Desarrollo sistema de automatización del techo exterior abatible

Para el desarrollo del segundo sistema automatizado, haremos como el anterior, listaremos todos los dispositivos u elementos que intervendrán en este desarrollo como son:

- Raspberry Pi modelo 3 B+.
- Llave de paso con servomotor.
- Fuente de alimentación 12VDC.
- Dispositivo Sonoff.
- Sensor de efecto hall.
- Tira led RGB.
- Sistema hidráulico.
- Sistema de audio.

La idea principal de este sistema, es poder controlar los elementos de luz, el equipo de música y la apertura y cierre del techo exterior, para poder acceder o proteger la zona de ocio que dentro se encuentra. Este sistema, va todo controlado mediante una Raspberry Pi que activará, por una parte, el servomotor para controlar el sistema hidráulico y así permitir la apertura y cierre del mismo, por otra parte, encender todo el sistema de audio y reproducir la música deseada mediante la aplicación móvil y, por último, permitir el cambio de luz de la tira led RGB.

5.2.1 Instalación y configuración de Raspberry Pi

Este apartado no lo vamos a desarrollar debido a que se trata de la misma instalación y configuración que hemos realizado para el sistema de automatización de la piscina y que ya se ha explicado en el punto 5.1.1 de este documento.

5.2.2 Instalación de dependencias y librerías en la Raspberry Pi

Como hemos explicado en el apartado anterior, este apartado se desarrollará igual que el del sistema automatizado de la piscina, con la única diferencia que se añadirá el paquete o librería de Mopidy[13], que se utilizará para configurar la conexión del software con el sistema de audio y la raspberry y así poder enviar y seleccionar la música deseada desde la aplicación Home Assistant.

Una vez realizados todos los pasos anteriores descritos en el punto 5.1.2, lo único que debemos hacer es instalar la librería Mopidy y sus dependencias.

Mopidy es una aplicación de sistema de audio integrado de Python que se ejecuta en segundo plano en una computadora, en este caso, en la Raspberry Pi. Se puede reproducir música de forma local, con conexión a Spotify, Google Play Music, etc., y te da herramientas de poder generar

listas de reproducción desde cualquier teléfono conectado a la aplicación Home Assistant.

Para la instalación del reproductor multimedia Mopidy en el sistema operativo de la Raspberry deberemos utilizar los siguientes comandos:

Versión de Mopidy 2.1.0

```
sudo pip3 install mopidy
```

Versión de Mopidy-Iris[14] 3.43.0

```
sudo pip3 install Mopidy-Iris
```

Una vez tenemos el reproductor multimedia instalado en el sistema, deberemos instalar la integración de Mopidy con Spotify, para así poder utilizar el servicio multimedia Spotify en nuestra aplicación y de esta forma tener acceso a para poder reproducir cualquier música subida en este servicio.

Para ello necesitaremos instalar la librería **libspotify12[15]** en la versión 12.1.103-0mopidy1.

```
sudo apt-get install
```

A continuación, la librería de **python-spotify[16]** en la versión 2.0.5-0mopidy1.

```
sudo apt-get install python-spotify
```

Seguiremos con la instalación del paquete **mopidy-spotify-3.0.0** en la versión 3.0.0 y se ha descargado e instalado con el setup.py de Python, que se trata de un sistema de instalación alternativo al pip3 de Python, y se utiliza cuando este último no deja instalar el paquete o librería.

Para su instalación, será necesario entrar en el directorio donde se encuentre el archivo **setup.py**, editarlo y añadir en el apartado **install_requieres**, la dirección de descarga de la librería, una vez hecho esto, volver al directorio y escribir el comando:

```
python setup.py install
```

Una vez se han realizado los pasos anteriores, pasaremos a la configuración de la integración de mopidy.

Con la instalación del reproductor multimedia Mopidy, se creará un archivo de configuración del mismo y que en nuestro caso se localiza en la ruta **/etc/mopidy/mopidy.conf**.

En este archivo, debemos añadir y activar nuestra cuenta de Spotify anteriormente creada en la web de Spotify[17], indicando el usuario y la contraseña como se muestra en la siguiente imagen.

Figura 64 Archivo de configuración de Mopidy

5.2.3 Desarrollo del software de automatización

Como hemos hecho en el punto 5.1.4, para empezar el desarrollo del software que controlará todas las acciones del sistema y la automatización del mismo, se ha creado un programa escrito en Python llamado **mqtt.py** en la raíz del sistema.

Empezaremos importando todas las librerías y dependencias que utilizaremos en el software, haciendo uso del **import**.

A continuación, declaramos las variables necesarias que utilizaremos para darles el valor que corresponda al pin de la Raspberry, por el cual le enviará y recibiremos los datos de señal. Para ello, crearemos un array en el que guardaremos todos los pines declarados para poder utilizar en las diferentes funciones que definiremos más adelante.

Estos pines, que son las salidas GPIO de la propia raspberry pi, deberemos declararlos e iniciarlos al principio de la ejecución del software en un estado True, lo que significa que estarán disponibles

por defecto, desde el arranque del sistema y declararlos a su vez como salidas de señal mediante el **GPIO.setup(NUMERO_PIN, GPIO.OUT)**.

```
pin_hall = 4
pin_pump = 17
pin_servo = 26

GPIO.setmode(GPIO.BCM)
GPIO.setup(pin_hall, GPIO.IN)
GPIO.setup(pin_pump, GPIO.OUT)
GPIO.setup(pin_servo, GPIO.OUT)

#PWM init
pwm_servo = GPIO.PWM(pin_servo, 50)
pwm_servo.start(0)
pwm_servo.ChangeDutyCycle(8.75) #close
time.sleep(5) #waiting position
pwm_servo.ChangeDutyCycle(0) #sleep

#Relay output init
GPIO.output(pin_pump, False)

#light
pin_r = 23
pin_g = 22
pin_b = 27
#pin_w = 18

GPIO.setup(pin_r,GPIO.OUT)
GPIO.setup(pin_g,GPIO.OUT)
GPIO.setup(pin_b,GPIO.OUT)
pwm_red = GPIO.PWM(pin_r,100)
pwm_green = GPIO.PWM(pin_g,100)
pwm_blue = GPIO.PWM(pin_b,100)
pwm_red.start(0)
pwm_green.start(0)
pwm_blue.start(0)
```

Figura 65 Declaración e inicialización de variables del software de control del techo abatible y la a zona de ocio

Lo siguiente que hacemos es crear una clase chillout (**class chillout**), en la cual definiremos todas las variables en false, y una variable por cada componente que interviene en el sistema de automatización.

En este caso inicializaremos los focos rgb a 0, ya que se tratan de valores enteros y el techo abatible en un estado False.

```
class chillout:
    st_roof = False
    st_red = 0
    st_green = 0
    st_blue = 0
```

Figura 66 Declaración de las variables en la clase chillout

A partir de este punto del software, empezaremos a declarar todas las funciones que controlarán y enviarán información a cada dispositivo respectivamente. Todas estas funciones las declararemos con el decorador de método estático de Python (**@staticmethod**).

- **Función setRGBColor(value)**
  Esta función recibirá un mensaje con un valor que englobará a su vez 3 valores separados por comas, haciendo referencia el primer valor que define la cantidad de rojo, el segundo valor a la cantidad de verde y el tercer valor a la cantidad de azul. Esto es así porque estamos tratando con unos focos RGB (red, green, blue).
  Para poder tratar estos valores y devolver un código de color a los focos RGB, será necesario separar estos mismos con la función **decode()** y **split()**, para dividir los valores y tratar cada uno por separado.
  Finalmente, debemos realizar una operación para calcular el tanto por cien y poder enviar cada código de color a protocolo de los focos RGB.

```
@staticmethod
def setRGBColor(value):
    colors = value.decode().split(',')
    chillout.st_red = int(colors[0]) / 255.0 * 100.0
    chillout.st_green = int(colors[1]) / 255.0 * 100.0
    chillout.st_blue = int(colors[2]) / 255.0 * 100.0
```

Figura 67 Función de selección de color de la tira led RGB

Como se puede observar, hacemos uso de la función **round()**, ya que el resultado que nos devuelva la operación deberá ser un entero, el sistema no admite ningún número con decimales (float).

- **Función setRGBPower(value)**
  En esta función, primero comprobamos si se cumple la condición de que la tira led RGB esté activa, y esto lo conseguimos haciendo una suma de todos los valores que tienen las variables st_red, st_green y st_blue, si esta suma es mayor a 0 significa que están encendidas, debido a que se habían inicializado a 0 al principio del programa. Si la condición se cumple, enviamos una señal de apagado al pin de la raspberry correspondiente y volvemos a resetear los valores de color a 0 para mantener la tira led apagada como en el estado inicial. Para resetear de nuevo a los valores iniciales, simplemente se le indicará un cambio de valor a 0.

```
@staticmethod
def setRGBPower(value):
    if int(value):
        pwm_red.ChangeDutyCycle(chillout.st_red)
        pwm_green.ChangeDutyCycle(chillout.st_green)
        pwm_blue.ChangeDutyCycle(chillout.st_blue)
    else:
        pwm_red.ChangeDutyCycle(0)
        pwm_green.ChangeDutyCycle(0)
        pwm_blue.ChangeDutyCycle(0)
```

Figura 68 Función de encendido y apagado de la tira led RGB de la zona de ocio

- **Función setRGBBrigthness(value)**
  Esta función esta creada, por si en un futuro quisiéramos cambiar la intensidad de color/brillo de la tira led, actualmente no la utilizamos porque siempre buscamos que las luces led tengan la mayor potencia de brillo posible, es decir, el valor que lleva por defecto las luces.

```
@staticmethod
def setRGBBrigthness(value):
    if int(value):
        pass
    pass
```

Figura 69 Función de brillo de la tira led RGB

- **Función setRoof(value)**
  Con esta función controlaremos la apertura y cierre del techo exterior de la zona de ocio, para ello, guardaremos en la variable **st_roof**, anteriormente declarada en la clase chillout, el nuevo valor que recogemos del mensaje enviado por la aplicación, en este caso el servidor Home Assistant.

  A continuación crearemos una condición que comprobará si la variable existe, es decir, que tenga valor. Una vez comprobamos que tenemos un valor guardado en ella, procedemos a indicarle al servomotor que debe abrirse, mediante la activación del pin al que está conectado de la Raspberry Pi y también procedemos a activar el pin de la bomba hidráulica, que será la encargada de inyectar en el sistema el líquido hidráulico para poner en funcionamiento los brazos hidráulicos.

  En caso de que la condición anterior no se cumpla, nos estará indicando que el techo hay que cerrarlo y debemos activar de nuevo los dos elementos que intervienen en el funcionamiento de la apertura y cierre del techo, y que se han explicado en el apartado anterior.

```
@staticmethod
def setRoof(value):
    chillout.st_roof = int(value)

    if chillout.st_roof:
        pwm_servo.ChangeDutyCycle(8.75) #open
        time.sleep(1)
        pwm_servo.ChangeDutyCycle(0) #sleep
        GPIO.output(pin_pump, True)
        while(not GPIO.input(pin_hall)):
            pass
        GPIO.output(pin_pump, False)
    else:
        GPIO.output(pin_pump, False)
        pwm_servo.ChangeDutyCycle(6.90) #close
        time.sleep(1)
        pwm_servo.ChangeDutyCycle(0) #sleep
```

Figura 70 Función de apertura y cierre del techo exterior

- **Función keepUpRoof(value)**
  Esta función es la encargada de mantener el techo siempre abierto y a una misma altura. Ha sido necesaria la creación de la función debido a que un brazo hidráulico de los instalados en el sistema estaba roto y perdía líquido hidráulico, lo que provoca que el propio techo con el paso del tiempo se vaya bajando poco a poco. Aquí en este momento, es donde interviene el sensor de efecto hall implementado y hace la función de un final de carrera, es decir, cuando el sensor envíe una señal de activación, será porque el imán que hay instalado en el sistema se ha puesto a la misma altura del sensor, produciendo así la activación de este y que provocará de nuevo la llamada a esta función y levantará de nuevo el techo a su límite marcado.

```
@staticmethod
def keepUpRoof(value):
    if value and chillout.st_roof and not GPIO.input(pin_hall):
        GPIO.output(pin_pump, True)
        while(not GPIO.input(pin_hall)):
            pass
        GPIO.output(pin_pump, False)
```

Figura 71 Función de mantener el techo abierto

Finalmente, se declaran las funciones de conexión del sistema automatizado, con el sistema de la aplicación Home Assistant mediante el protocolo de mensajes MQTT, explicado al final del punto 5.1.4.

```
## MQTT connection
# The callback for when the client receives a CONNACK response from the server
def on_connect(client, userdata, rc, properties=None):
    print('Connected with result code ' + str(rc))
    for topic in devices:
        client.subscribe(topic)
        print('subscribe: '+str(topic))
        client.publish(devices[topic]['publish'], "0")

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print(msg.topic)
    print(msg.payload)
    getattr(chillout, devices[msg.topic]['function'])(msg.payload)
    client.publish(devices[msg.topic]['publish'], msg.payload)

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.username_pw_set(username="[redacted]", password="[redacted]")
client.connect("10.0.0.89", 1883, 60)
```

Figura 72 Funciones de conexión del sistema automatizado con el servidor Home Assistant

Para poder ejecutar el programa en el sistema de la Raspberry Pi, será necesario crear un servicio en el sistema operativo y para ello seguiremos los mismos pasos explicados en el último párrafo del punto 5.1.4.

5.2.4 Conexiones del sistema

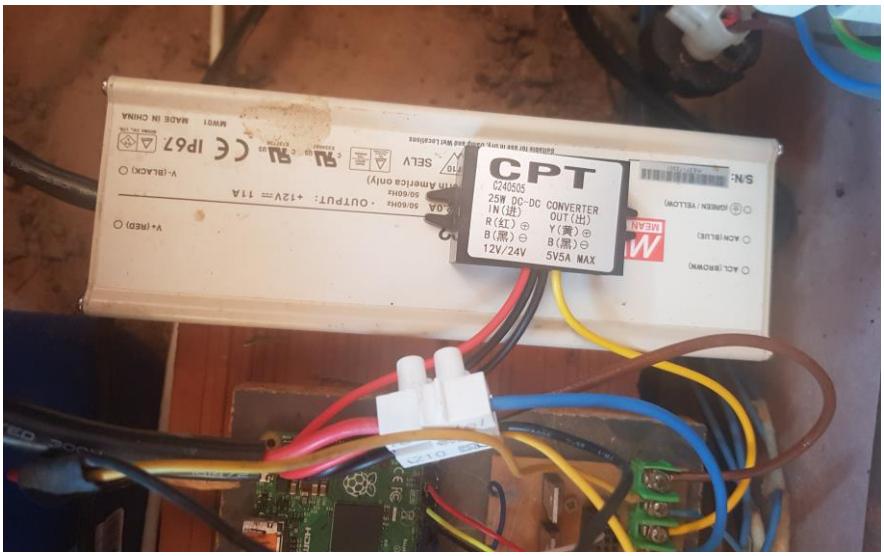


Figura 73 Conexiones fuente de alimentación

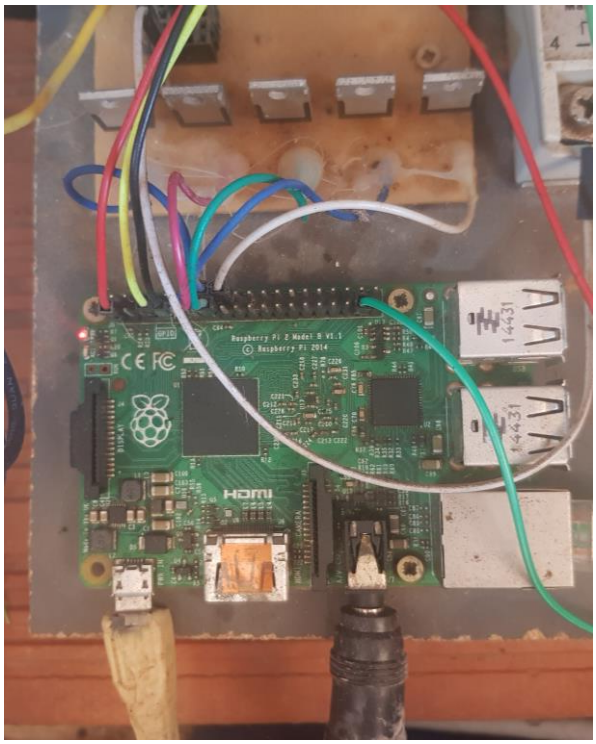


Figura 74 Conexiones Raspberry Pi del techo abatible

Figura 75 Conexiones servomotor con llave de paso

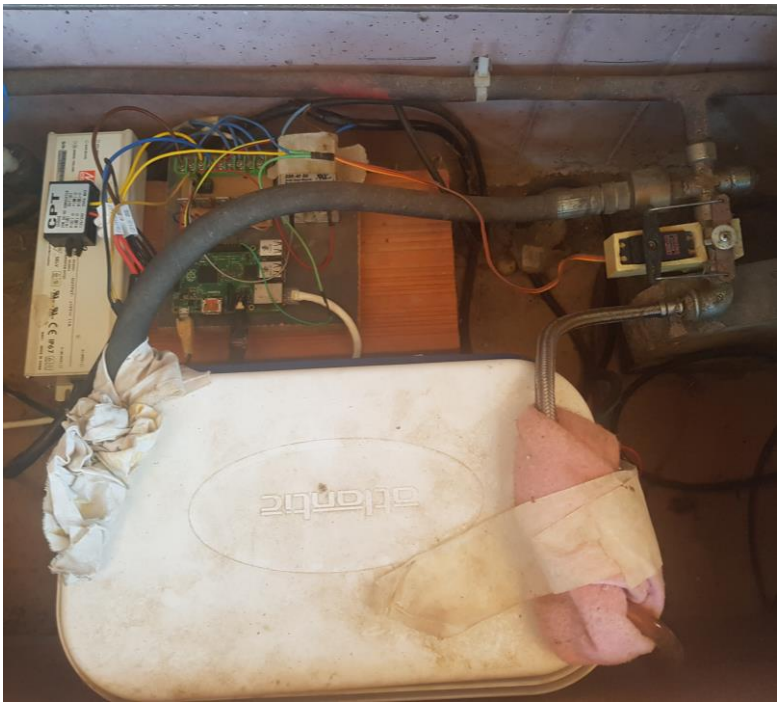


Figura 76 Sistema completo del techo abatible con las conexiones al sistema hidráulico

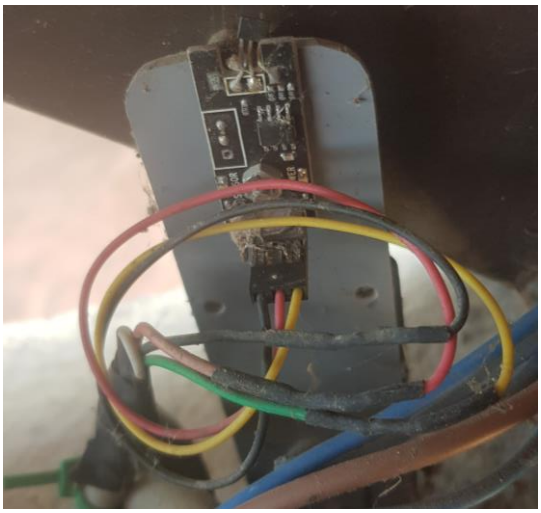


Figura 77 Conexiones del sensor de efecto hall

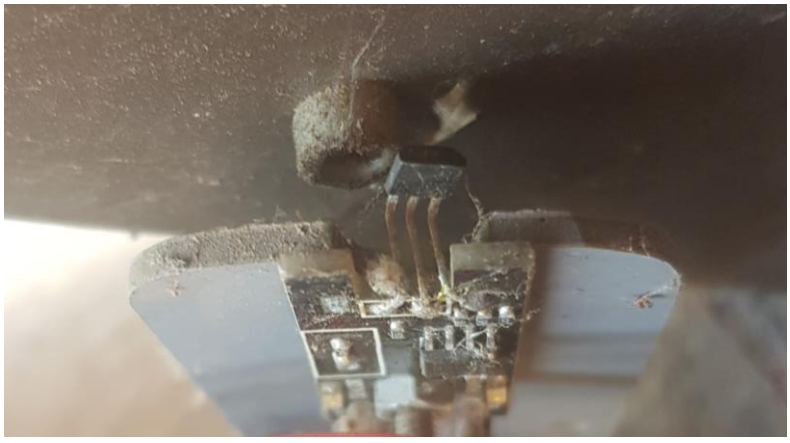


Figura 78 Posición del imán para el funcionamiento del sensor de efecto hall

6. Pruebas

A continuación, presentaremos una serie de imágenes reales tomadas de todo el sistema de automatización, tanto de los elementos del sistema como de capturas de la propia aplicación de Home Assistant en funcionamiento.

6.1 Pruebas del sistema automatizado de la piscina

Para mostrar que el funcionamiento del sistema automatizado de la piscina que buscábamos se cumple, mostraremos cada funcionalidad del sistema en imágenes, para demostrar así que el proyecto se ha llevado a cabo tanto en desarrollo como en ejecución y que a día de hoy sigue en pleno funcionamiento sin problemas.

6.1.1 Iluminación RGB

Figura 79 Encendido de luces de día de la piscina en color blanco

Figura 80 Encendido de luces de noche en color blanco

Figura 81 Encendido de luces de noche en color verde

Figura 82 Encendido de luces de noche en color azul

Como se observa en las imágenes anteriores, se han seleccionado varios colores en los focos RGB desde la aplicación, para así mostrar las combinaciones posibles que se pueden crear.

Figura 83 Activación luz verde desde la aplicación

6.1.2 Bomba de aire estilo jacuzzi

En este apartado, mostraremos las imágenes de la bomba de aire en funcionamiento junto con otro dispositivo en marcha, en este caso, las luces RGB.

Figura 84 Encendido del sistema de aire con luces

Figura 85 Encendido del sistema de aire con luces verdes

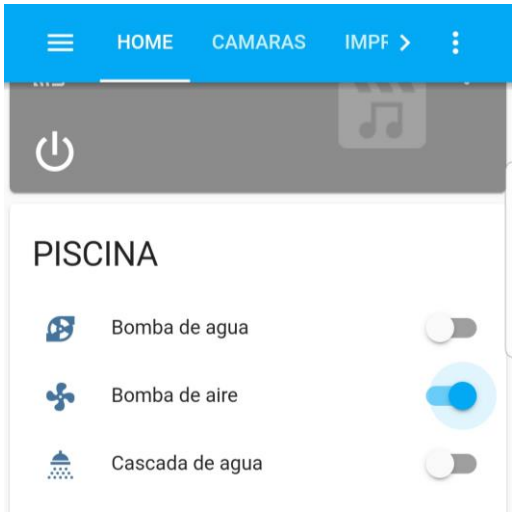


Figura 86 Activación bomba de aire desde la aplicación

6.1.3 Cascada de agua

Este elemento, depende de la bomba de agua que, al activarse, lanzará un chorro de agua en forma de cascada desde el exterior de la piscina.

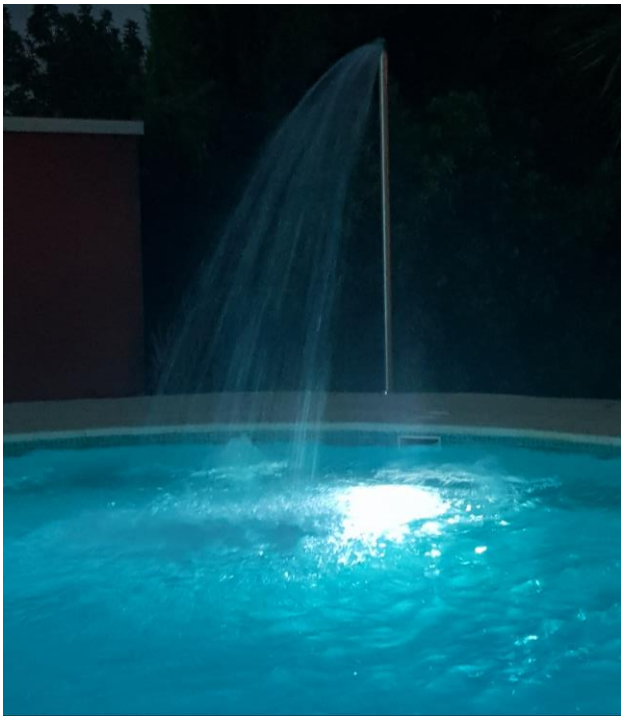


Figura 87 Encendido de la cascada

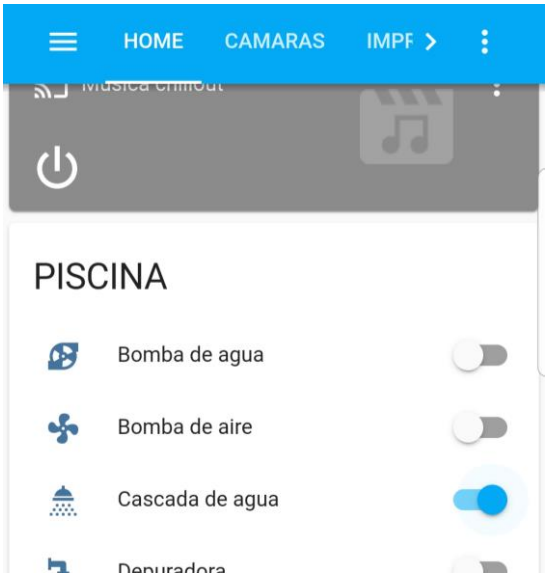


Figura 88 Activación cascada de agua en la aplicación

6.1.4 Bomba de depuración, llenado piscina y robot limpia fondos

Como no se pueden mostrar los dispositivos en funcionamiento, al no ser su funcionamiento un elemento visual, mostraremos su activación en la aplicación de Home Assistant.

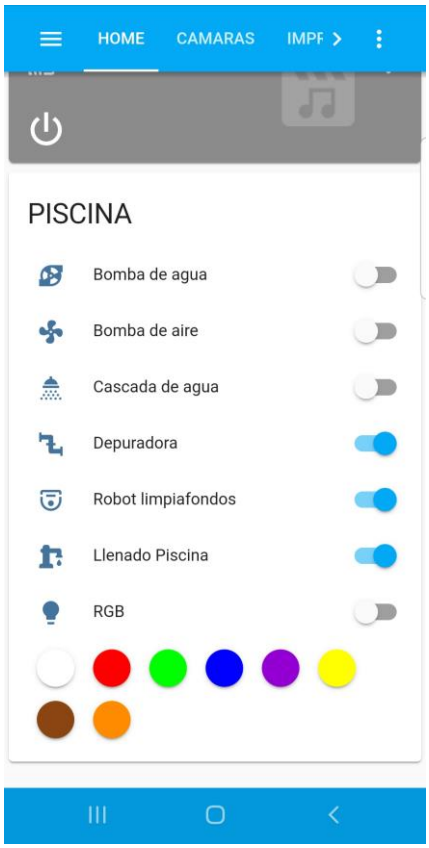


Figura 89 Activación depuradora, llenado piscina y robot limpia fondos desde la aplicación Home Assistant

6.1.5 Luz exterior

En este apartado, se puede comprobar el funcionamiento del foco de luz exterior y la activación del mismo desde la aplicación.

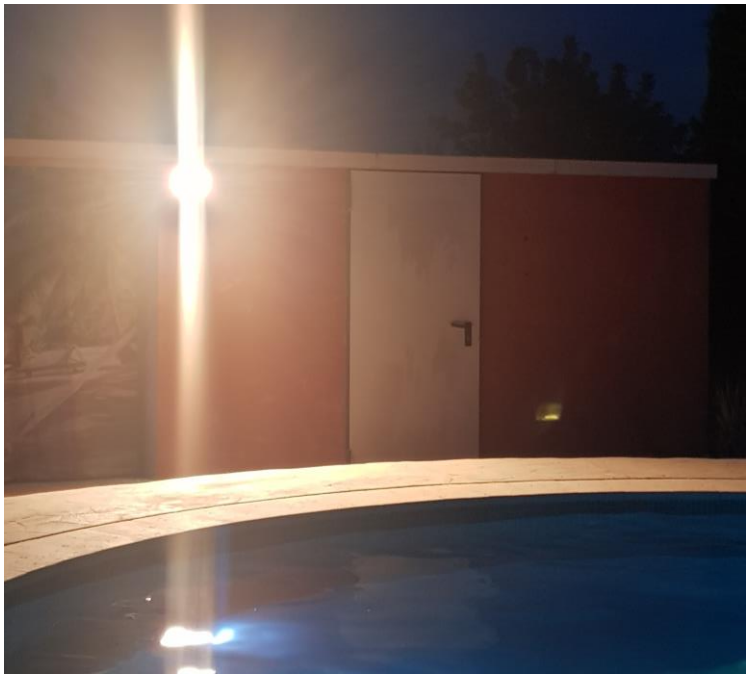


Figura 90 Encendido del foco exterior

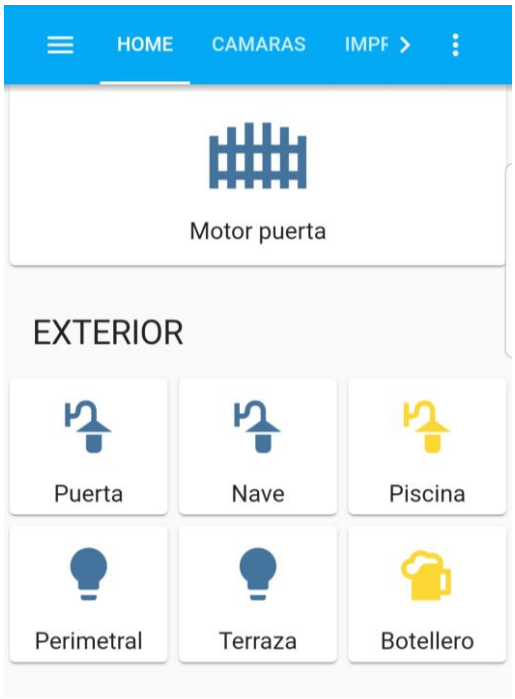


Figura 91 Activación del foco exterior en la aplicación

6.2 Pruebas del sistema automatizado del techo exterior abatible.

En este apartado, presentaremos pruebas gráficas, como en el anterior de todo el sistema automatizado y de sus elementos en funcionamiento, para demostrar el correcto funcionamiento del sistema de techo exterior abatible.

6.2.1 Apertura y cierre del techo

En las siguientes imágenes, se puede observar tanto la apertura como el cierre del techo exterior, facilitando así el resguardo del mismo cuando no se esté utilizando, todo mediante el uso de dos brazos hidráulicos que harán de mecanismo de levantar o bajar el propio techo.

Figura 92 Cierre completo del techo exterior

Figura 93 Apertura parcial del techo exterior

Figura 94 Apertura completa del techo exterior con vista lateral

Figura 95 Apertura completa del techo exterior con vista frontal

En las imágenes anteriores, se muestra todo el ciclo de apertura y cierre del techo exterior, dando acceso así al interior del mismo o a la protección de los elementos interiores en el caso de estar cerrado.

A continuación, se muestra el estado de la aplicación cuando el techo se encuentra abierto o cerrado.

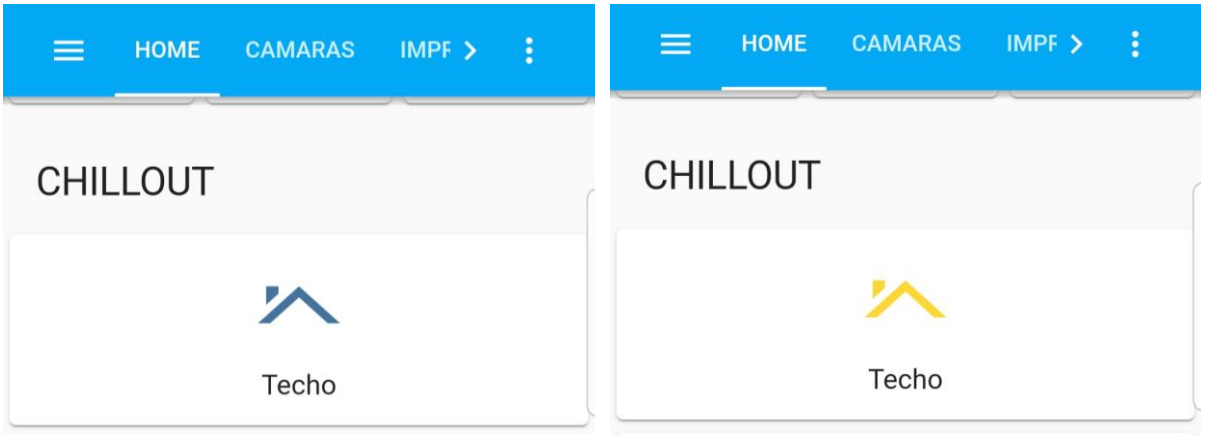


Figura 96 Estado de cierre y apertura en la aplicación

6.2.2 Iluminación RGB

Como en el caso del sistema de automatización de la piscina, en el techo exterior abatible añadimos también iluminación RGB, y en la aplicación Home Assistant el funcionamiento es el mismo que para el de la piscina, se muestra en las imágenes siguientes, la iluminación in situ y la activación y selección de color en la aplicación.

Figura 97 Iluminación en color verde del techo exterior

Figura 98 Iluminación en color azul del techo exterior

Figura 99 Activación del color azul en el techo exterior

6.2.3 Sistema de audio

En este apartado, se mostrarán imágenes del sistema de audio de la aplicación Home Assistant integrado en el sistema de automatización del techo exterior, tanto del selector de canciones a través del plugin Mopidy como de la propia aplicación Home Assistant.

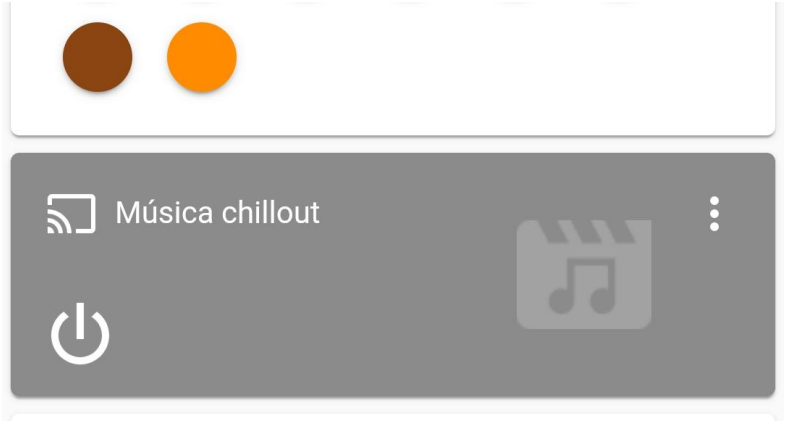


Figura 100 Estado inactivo del sistema de audio

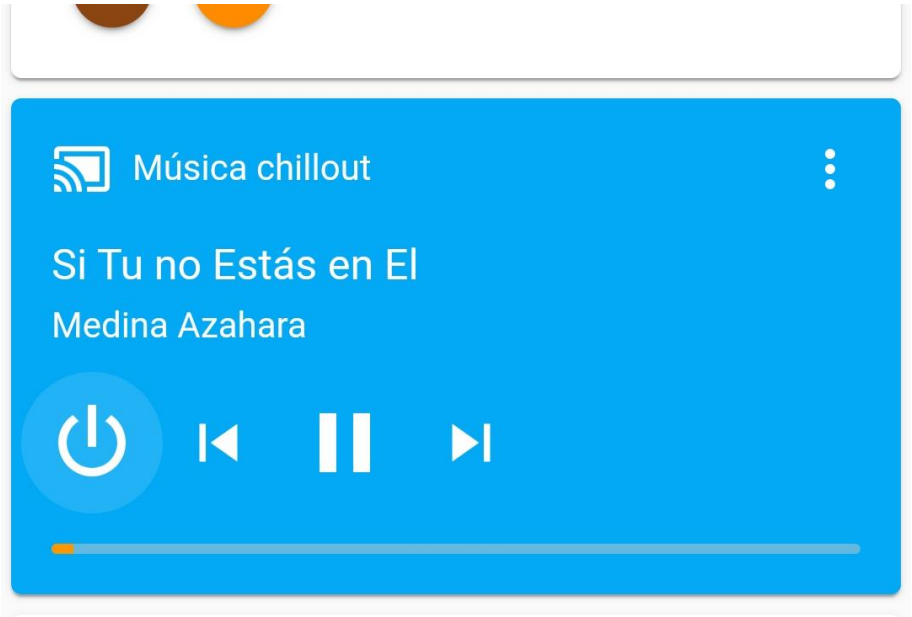


Figura 101 Estado activo del sistema de audio

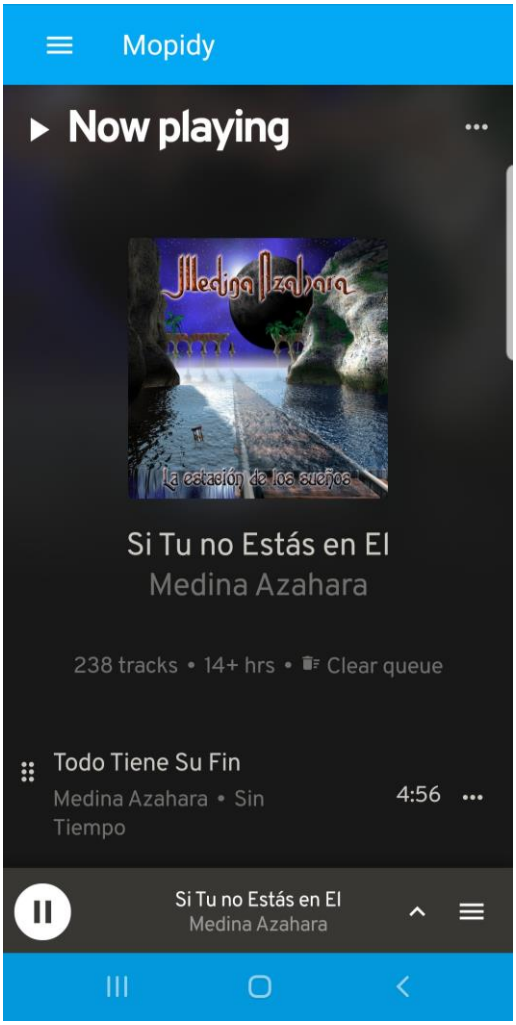


Figura 102 Reproduciendo música en la aplicación a través de Mopidy

6.3 Pruebas aplicación Home Assistant

En este punto, como ya se han mostrado capturas de pantalla de la aplicación móvil en el punto anterior, ahora incluiremos en este, capturas de pantalla de la versión pc, para así poder abarcar todos los formatos para los que se ha desarrollado la aplicación. Aun así, mostraremos capturas de pantalla de otras opciones no mostradas en el punto anterior.

No se ha llevado a cabo la explicación del desarrollo de la aplicación Home Assistant ni tampoco su configuración, porque este apartado forma parte del proyecto conjunto al que pertenece este proyecto y que está siendo desarrollado por el alumno **Pedro Amores Dolz** en su propio trabajo.

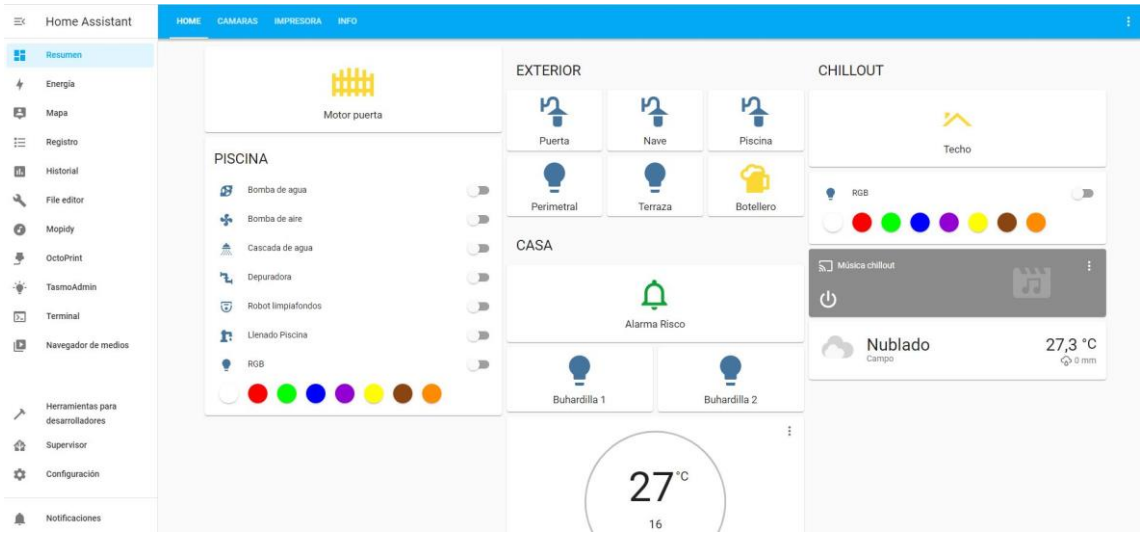


Figura 103 Interfaz web de la aplicación Home Assistant

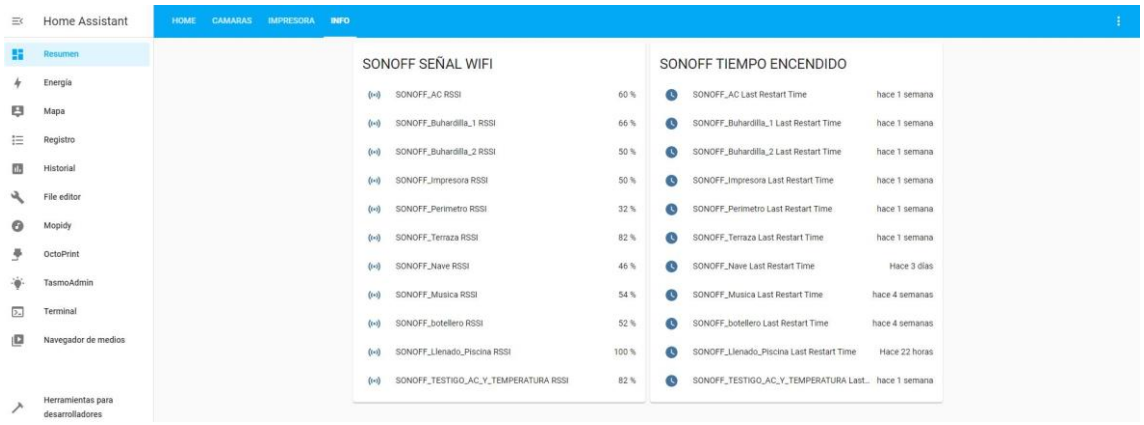


Figura 104 Panel de control de los dispositivos conectados a Home Assistant

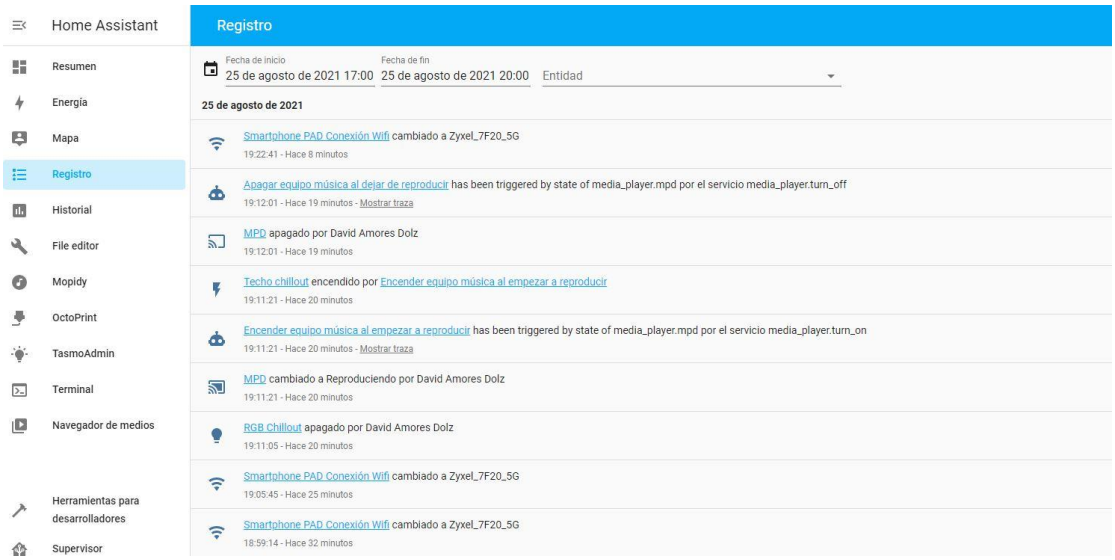


Figura 105 Interfaz del registro de movimientos y cambios de Home Assistant

7. Conclusiones

El objetivo principal del proyecto era poder crear un sistema automatizado de todos los componentes de una casa/chalet y poder controlarlos mediante una aplicación móvil o desde un ordenador, desde cualquier parte mediante internet.

Este trabajo final de grado, es una parte del proyecto general en el que automatizamos una piscina/jacuzzi, junto con un techo exterior abatible.

Ambos objetivos, tanto el de este proyecto como el desempeñado por otro compañero en su propio trabajo final de grado, se han podido llevar a cabo después de muchos meses de desarrollo y pruebas, con la finalidad de hacer un sistema automatizado capaz de competir con los productos que hay actualmente en el mercado, y haciendo un uso amplio de todos los software y hardware de código libre que podemos encontrar en la actualidad.

Durante todo el proceso de desarrollo del proyecto, nos hemos encontrado con muchos problemas, sobre todo técnicos, debido a la inexperiencia o la falta de conocimientos en algunos dispositivos, como es en el caso de los Sonoff y su software Tasmota, unos conocimientos que, a través de leer, buscar, hacer pruebas y cometer fallos, hemos podido alcanzar nuestro objetivo de una manera satisfactoria.

Con el primer método que desarrollamos, conseguimos un sistema automatizado funcional, pero con carencias que debíamos de solventar, como era el caso de no tener un sistema centralizado, y para ello tuvimos que sustituir nuestra aplicación desarrollada en Ionic y hacer uso de la aplicación Home Assistant y de los dispositivos sonoff y las nuevas configuraciones de las Raspberry Pi, para así de esta forma finalmente solventar esos problemas.

Por último, gracias al desarrollo de todo el proyecto, se han adquirido una amplia gama de conocimientos en todos los aspectos, ya sean electrónicos, eléctricos, de programación, y sobre nuevos dispositivos y por supuesto, la ampliación de conocimientos sobre las Raspberry.

7.1 Relación del trabajo desarrollado con los estudios cursados

Para poder hacer un uso de los conocimientos que he adquirido en la carrera y poder plasmarlos en el proyecto, he recurrido a los conocimientos que aprendí y desarrolle en las asignaturas de la rama, concretamente, con **DCLAN** (Diseño y Configuración de Redes de Área Local), para poder crear toda la red de conexiones y comunicación entre todo el sistema, **IAP** (Integración de Aplicaciones), para poder desarrollar e implementar la aplicación general desde la que se maneja todo el sistema y por último, las asignaturas de **RCO** (Redes de computadores) y **SSR** (Sistemas y Servicios en Red) que van ligadas, para poder implementar el uso de todo el sistema en la red y así poder comprender su comportamiento y funcionalidad.

8. Trabajos futuros

Al tratarse de un proyecto tan amplio y con un amplio abanico de posibilidades, siempre van a haber posibles ampliaciones futuras que irán surgiendo con el uso o la falta que hagan en un futuro al querer desempeñar ciertas acciones.

El mundo de la automatización es muy denso, lo que produce miles de posibles ampliaciones como ya puedan ser anexar al sistema nuevos elementos.

También hemos valorado de implementar un sistema de riego y cuidado del jardín automatizado, pero ya se convertía en un proyecto mucho más amplio y extenso y complicado a la hora de poder resumirlo dentro de todo el proyecto, por lo tanto, es algo que dejaremos como una futura ampliación del sistema.

Gracias a la tecnología que hemos utilizado, como comentaba, es posible hacer infinitos desarrollos e implementaciones.

Posibles mejoras o implementaciones al proyecto desarrollado:

- Huerto o jardín automatizado.
- Implementación de un sistema de auto control de los niveles de pH y cloro de la piscina.
- Incorporar la automatización mediante ordenes por voz.

9. Referencias y bibliografía

PRODUCTOS:

[1] Tienda web Armario Electronic-050 - [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.piscinayspa.com/es/accesorios-piscinas/61/cuadros-electricos-piscinas/6500/armario-maniobra-electronic-general-electric-1-bomba-control-iluminacion-transf-100w-wifi-4-canales/?gclid=Cj0KCQjwsZKJBhC0ARIsAJ96n3WpL8pbHRAifw-oC4XHCOHRvD6rSq-fKUnO4O9o0T3F0Bg-hmXWpOoaAmYzEALw_wcB

[2] Tienda web Smart Pool Controller - [En línea] [Último acceso: 29 de Agosto de 2021.] https://q-tech.es/products/control-integral-de-la-piscina-smart-control?utm_campaign=gs-2018-11-19&utm_content=sag_organic&utm_medium=smart_campaign&utm_source=google&variant=29257835708504

[3] Tienda web Fluidra Connect - [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.astralpool.com/es/productos/piscina/fluidra-connect/dispositivos-1/connect-box-5/

DOCUMENTACIÓN:

[4] Raspberry Pi Web Oficial - [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[5] Manual de instalación de Home Assistant en Raspberry Pi - [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.home-assistant.io/installation/raspberrypi

[6] Programa Filezilla - [En línea] [Último acceso: 29 de Agosto de 2021.] https://filezilla-project.org/

[7] Programa Raspberry Pi Image – [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.raspberrypi.org/software/

[8] Paquete pip3 - [En línea] [Último acceso: 29 de Agosto de 2021.] https://ubunlog.com/pip-instalacion-conceptos-basicos-ubuntu-20-04/

[9] Librería paho-MQTT - [En línea] [Último acceso: 29 de Agosto de 2021.] https://pypi.org/project/paho-mqtt/

[10] Librería RPi.GPIO - [En línea] [Último acceso: 29 de Agosto de 2021.] https://pypi.org/project/RPi.GPIO/

[11] Librería WS281x - [En línea] [Último acceso: 29 de Agosto de 2021.] https://github.com/richardghirst/rpi_ws281x

[12] Sonoff Web Oficial - [En línea] [Último acceso: 29 de Agosto de 2021.] https://sonoff.tech/product/diy-smart-switch/minir2/

[13] Librería Mopidy - [En línea] [Último acceso: 29 de Agosto de 2021.] https://github.com/mopidy/mopidy-spotify/tree/v3.0.0

[14] Librería Mopidy Iris - [En línea] [Último acceso: 29 de Agosto de 2021.] https://mopidy.com/ext/iris/

[15] Librería libspotify12 - [En línea] [Último acceso: 29 de Agosto de 2021.] https://pypi.org/project/Mopidy-Spotify/

[16] Librería Python spotify - [En línea] [Último acceso: 29 de Agosto de 2021.] https://github.com/plamere/spotipy

[17] Spotify Web oficial - [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.spotify.com/es/

Diagrama de conexiones NeoPixel - [En línea] [Último acceso: 29 de Agosto de 2021.] https://blog.330ohms.com/2020/06/17/como-conectar-led-rgb-neopixel-ws2812-a-raspberry-pi/

Configuración de NeoPixel en Raspberry Pi - [En línea] [Último acceso: 29 de Agosto de 2021.] https://learn.adafruit.com/getting-started-with-raspberry-pi-pico-circuitpython/neopixel-leds

Datasheet PDF Led WS2815 - [En línea] [Último acceso: 29 de Agosto de 2021.] http://www.normandled.com/upload/201808/WS2815%20LED%20Datasheet.pdf

Esquema pines de conexión Raspberry Pi - [En línea] [Último acceso: 29 de Agosto de 2021.] https://aprendiendoarduino.wordpress.com/2020/03/04/manejar-gpio-raspberry-pi/

Documento configuración SSH en Raspberry Pi - [En línea] [Último acceso: 29 de Agosto de 2021.] https://blog.ahierro.es/habilitar-acceso-ssh-a-raspberry-pi/

Documento configuración red Wifi en Raspberry Pi - [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.luisllamas.es/raspberry-pi-wifi/

Repositorio de Mopidy Spotify - [En línea] [Último acceso: 29 de Agosto de 2021.] https://github.com/mopidy/mopidy-spotify/tree/v3.0.0

Manual de instalación de Raspbian en Raspberry Pi - [En línea] [Último acceso: 29 de Agosto de 2021.] https://turinconinformatico.com/raspberrypi/instalar-raspbian/

¿Que es arduino? - [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.xataka.com/basics/arduino-raspberry-pi-que-cuales-sus-diferencias

Manual completo de Home Assistant - [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.home-assistant.io/docs/

Manual de instalación de Spotify en Raspberry Pi - [En línea] [Último acceso: 29 de Agosto de 2021.] https://circuitdigest.com/microcontroller-projects/how-to-run-spotify-on-raspberry-pi-using-mopidy-music-server

Manual de la librería Mopidy - [En línea] [Último acceso: 29 de Agosto de 2021.] https://docs.mopidy.com/en/latest/

Manual de instalación de paquetes en Python - [En línea] [Último acceso: 29 de Agosto de 2021.] https://www.activestate.com/resources/quick-reads/how-to-manually-install-python-packages/

10. Anexo

Objetivos de desarrollo sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**				**X**
ODS 4. **Educación de calidad.**				**X**
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**	**X**			
ODS 7. **Energía asequible y no contaminante.**		**X**		
ODS 8. **Trabajo decente y crecimiento económico.**				**X**
ODS 9. **Industria, innovación e infraestructuras.**	**X**			
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**		**X**		
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

ODS 6. Agua limpia y saneamiento.

Seleccionamos este objetivo de desarrollo sostenible debido a que tratamos y controlamos mediante nuestro sistema el agua de la piscina del proyecto, tratando así su pH, clorinidad y el estado en el que se encuentra el agua en ese instante. De no realizar estos controles, no podríamos conseguir un agua de calidades optimas y limpia, libre de patógenos.

ODS 7. Energía asequible y no contaminante.

Este otro ODS, también lo hemos seleccionado, ya que nuestro sistema se compone de dispositivos de bajo consumo y se lleva un control del funcionamiento de los dispositivos en el momento adecuado, es decir, no se hace una sobre utilización de los dispositivos, solo se utilizan en el momento que se requieren.

ODS 9. Industria, innovación e infraestructuras.

Este ODS, es uno de los que más destaca en nuestro trabajo final de grado, ya que hemos creado un sistema con recursos reutilizados que se les ha dado otra funcionalidad, creando así una innovación en nuestro trabajo.

Por otra parte, las infraestructuras que hemos llevado a cabo ya tienen una configuración de bajo uso de recursos y de una alta eficiencia energética.

ODS 12. Producción y consumo responsables.

Por último, este ODS también podemos relacionarlo con nuestro trabajo final de grado, ya que como se especifica en el trabajo, hemos realizado y convertido viejos dispositivos y los hemos adaptado a nuestro sistema, como es el caso de los hidráulicos del techo abatible, que antes formaban parte de un tractor que ya no estaba en funcionamiento, y los hemos aprovechado para ser reutilizados y adaptados para la apertura y cierre del techo de la zona de ocio.