Trabajo Fin de Máster
Máster en Sistemas de Energía Eléctrica

Implementación de un sistema domótico mediante la plataforma de código abierto Home Assistant

Autor: Raúl del Pino Villanueva
Tutor: Juan Manuel Mauricio

Dpto. de Ingeniería Eléctrica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Máster
Máster en Sistemas de Energía Eléctrica

Implementación de un sistema domótico mediante la plataforma de código abierto Home Assistant

Autor:
Raúl del Pino Villanueva

Tutor:
Juan Manuel Muricio
Profesor Contratado Doctor

Dpto. de Ingeniería Eléctrica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Trabajo Fin de Máster: Implementación de un sistema domótico mediante la plataforma de código abierto Home Assistant

Autor: Raúl del Pino Villanueva

Tutor: Juan Manuel Mauricio

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mi mujer

A mis padres

Resumen

En los días que corren cada vez es más frecuente poder controlar y automatizar cualquier dispositivo o tarea diaria, con el objetivo de hacernos la vida más fácil. Aquí entra el famoso concepto *Internet de las Cosas*, que se refiere a la interconexión digital de objetos cotidianos con Internet, convirtiéndose así en inteligentes.

Hace unos años esto era impensable. La domótica sólo estaba al alcance de unos pocos privilegiados por su excesivo precio. Eran sistemas propietarios de determinados fabricantes, que te obligaban a utilizar su gama de productos y cuya instalación y configuración no eran nada triviales.

Afortunadamente, la irrupción de numerosas empresas que ofrecen productos de calidad a precios muy competitivos y de fácil instalación, ha hecho que haya una gran competencia en el mercado por lo que siempre favorecerá al consumidor. Esto hace que cualquier persona pueda tener un sistema domótico en su hogar.

El problema surge cuando se quieren tener los mejores productos de cada marca, ya sea por elección propia o porque determinada empresa no dispone de algún dispositivo en concreto. En este caso, se necesitaría instalar una app por cada producto y, además, no se podrán relacionar entre ellos.

Ahí es donde nace este trabajo, de la necesidad de poder integrar cualquier producto de cualquier fabricante en un único sistema domótico. Y para ello se va a utilizar la plataforma de código abierto Home Assistant, siendo actualmente la más potente y con mayor proyección entre las distintas opciones que existen. Este sistema programado en Python 3 tiene un sinfín de posibilidades de personalización, con una interfaz moderna y dando soporte a multitud de componentes de distintas marcas.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN A LOS SISTEMAS DOMÓTICOS

1.1. Introducción

Los sistemas domóticos implementados en los hogares están siendo cada vez más frecuentes, debido en gran medida a sus precios competitivos y a su facilidad de instalación. Aunque uno de sus principales problemas, como siempre ha ocurrido, es que depende en gran parte de utilizar únicamente los dispositivos y sensores de un solo fabricante. Si se quieren adquirir varios productos de diferentes marcas, surge la problemática de no poder interconectarlos entre ellos y, además, se tiene que utilizar una aplicación diferente por cada dispositivo que no pertenezca al mismo fabricante.

Todo esto hace que vayan surgiendo nuevas plataformas abiertas para poder integrarlo todo en un único sistema domótico, independientemente de la marca a la que pertenezca o a su protocolo de comunicación. Se consigue que todos los dispositivos y sensores puedan interactuar entre sí, creando un entorno muy personalizable y accesible a casi cualquier persona.

Para este trabajo se ha elegido la plataforma de código abierto Home Assistant. Su código está desarrollado en Python 3 y su configuración se realiza mediante ficheros YAML, que es un formato para el intercambio de información que tiene como objetivo facilitar el mapeo de estructuras de datos más complejas en un documento de texto plano legible para un ser humano.

De un gran proyecto sale también una gran comunidad de usuarios, y este es el caso de Home Assistant. La actividad diaria en el foro oficial es muy alta, dando un gran soporte a la plataforma. Se resuelven dudas, se aportan soluciones a los errores que van surgiendo y se comparten proyectos propios. Aunque su contenido es en inglés, cada vez es más frecuente encontrar en la red contenidos en español. Además, las actualizaciones son constantes, cada poco tiempo (una o dos semanas) salen nuevas vesiones con mejoras, correcciones y nuevas integraciones de componentes.

1.2. Internet de las cosas (IoT)

Lo primero que cabe destacar es que la expansión de internet y su acceso han crecido de forma exponencial en los últimos años. Se vive en una sociedad hiperconectada, ya que se pueden realizar comunicaciones con cualquier persona del mundo en cuestión de segundos, y se sabe lo que ocurre en cualquier parte del planeta en tiempo real. Los dispositivos que se llevan en los bolsillos son más potentes que muchos ordenadores de hace unos años pero hasta hace poco esta comunicación ha estado limitada a ser entre seres humanos, ya que eran los únicos creadores y consumidores de la información que se generaba.

Desde el nacimiento de los ordenadores eran las personas los que tenían que introducir la información al sistema, ya fuera tecleando, rellenando formularios o sacando fotografías. Pero cuando la conectividad y la electrónica se vuelven tan asequibles se puede dotar de sistemas de comunicación a cualquier elemento. Estos objetos serán capaces de ofrecer información, enviarla directamente a la base de datos para que los servidores la procesen, notificar sobre comportamientos erróneos o enviar sugerencias sobre su uso.

Y cuando se hablan de elementos se puede pensar en cualquier objeto de la vida cotidiana, desde gafas, zapatillas y camisetas, hasta tostadoras, hornos, frigoríficos y grifos, pasando por semáforos, coches, aceras o árboles. Los sensores y actuadores de los que dispondrán los objetos comunicarán sus valores a la red de forma autónoma, y por tanto los humanos ya no seremos la única fuente de introducción de datos.

Una de las consecuencias más importantes de este nuevo paradigma será la comunicación entre todos los dispositivos, que generará un tráfico de datos nunca visto hasta la fecha. Además del reto tecnológico que supone desplegar una red de comunicaciones que soporte tal movimiento de bits, aún mayor es el reto que permite almacenar y procesar esa montaña de información. Los servidores que alberguen y procesen dichos datos tendrán que implementar nuevos algoritmos capaces de enfrentarse a todos esos unos y ceros para poder sacar valor añadido. Por tanto, además de las innumerables ventajas que aportará el internet de las cosas, también traerá consigo una serie de retos tanto sociales como tecnológicos.

Desde el punto de vista tecnológico, se podría destacar la seguridad como uno de los puntos críticos del sistema. Pasar de un sistema en el que los objetos son puramente locales a uno en el que todo está conectado a la red, y por tanto expuesto a un control externo, puede resultar muy peligroso para cualquiera: que alguien consiga controlar remotamente los semáforos de una gran ciudad, los contadores de energía de una compañía eléctrica, el sistema de videovigilancia de un edificio importante o el marcapasos inteligente de alguien que ha sufrido un infarto recientemente.

Por otro lado, el control remoto de los objetos hace que el comprador de un dispositivo no sea el único que tenga acceso al mismo, ya que el fabricante o proveedor del servicio también podrá ser tan dueño del dispositivo como el usuario final. Las preguntas sobre qué derechos tiene el fabricante sobre el dispositivo, si pueden recabar y almacenar todos los datos reportados por el objeto y qué puede hacer con esos datos, son obligatorias. Todo esto se tiene que legislar convenientemente con respecto a la privacidad y el anonimato, siendo conceptos clave en este nuevo escenario, ya que los perfiles que se puedan crear para cada usuario serán muy detallados.

Uno de los campos en los que más va a penetrar el internet de las cosas es en el de los hogares, a través de lo que ya se conoce hoy en día como las Smart Homes. La domótica se puede enfocar desde distintos puntos de vista: la seguridad, la eficiencia energética, el confort, la salud, etc. En pocos años, dispositivos como cámaras IP, cerraduras, termostatos, neveras, hornos, camas y espejos inteligentes, serán tan habituales como hoy en día lo es tener una televisión o un teléfono. La casa será capaz de detectar el horario de entrada y salida, el termostato ajustará el encendido de la calefacción para ahorrar energía maximizando el confort, las luces serán capaces de regular su intensidad y color dependiendo de la actividad que se esté realizando, la cama hará desperta a la persona lentamente sin que el resto de vecinos tengan que escuchar el despertador y la tostadora dejará el pan al gusto.

Además, las formas de interactuar con el sistema se minimizarán al máximo porque la casa será capaz de anticiparse a los movimientos, y en los casos en los que el usuario tenga que tomar el control lo hará a través de sistemas de voz o gestos. Se espera un futuro (ya presente) revolucionario a la par que inquietante, por eso es muy importante tener conocimiento sobre la tecnología que subyace, los pros y contras de este tipo de sistemas y estar informado al respecto.

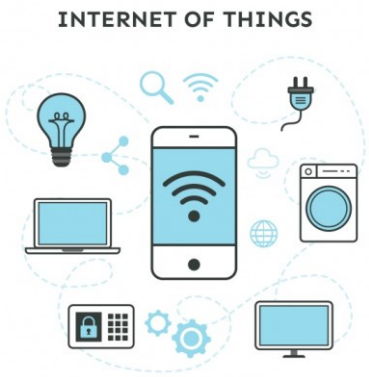


Figura 1-1. Internet de las cosas.

1.3. Controladores domóticos

Los controladores domóticos son equipos con aplicaciones especialmente diseñadas para controlar dispositivos en nuestros hogares. Deben controlar entradas, salidas, ejecutar comandos o gestionar datos. Se puede comprar con el software ya instalado o montarlos uno mismo utilizando un pequeño ordenador, que es lo que se va a realizar en este proyecto. En realidad, cualquier tipo de ordenador puede realizar las funciones de controlador domótico, pero hay que tener en cuenta que éste va a permanecer en todo momento alerta a la espera de recibir comandos e información, es decir, se va a tener encendido las 24 horas del día. Por ello, es conveniente que el consumo del equipo y su potencia de cálculo estén acordes a las tareas que debe realizar.

Si se monta un sistema domótico sobre un PC funcionará sin ningún problema pero se tendrá un equipo caro para las funciones que debe realizar, trabajando además a un nivel muy bajo de capacidad y con un consumo

de energía más alto con respecto a un equipo domótico. Por ello, los controladores domóticos son equipos con todas las funcionalidades de un PC pero adaptados en cuanto a potencia y tareas que deben ejecutar. En esta definición encajan a la perfección los sistemas embebidos como la popular Raspberry Pi (aunque hay muchos más), ya que al contrario de lo que ocurre con los ordenadores de propósito general que están diseñados para cubrir un amplio rango de necesidades, éstos se diseñan para cubrir necesidades específicas. En estos sistemas la mayoría de los componentes se encuentran incluidos en la placa base, tomando de ahí su nombre.

En cuanto a los modelos, existen diversos tipos que se diferencian tanto en su precio como en sus prestaciones. En este proyecto se va a utilizar una Raspberry Pi 3 como controlador, ya que sus prestaciones ideales para este propósito. Sus características son: un procesador de 64 bits a una frecuencia de 1,2 GHz, memoria RAM de 1 GB, conectividad Wifi y Bluetooth, y diferentes entradas y salidas como un puerto HDMI, un puerto de audio, cuatro puertos USB, un puerto Ethernet y 40 pines GPIO, al igual que una ranura con lector de tarjetas microSD donde se intriducirá el sistema operativo.

En cuanto al sistema operativo, para usar este ordenador se necesita un sistema adaptado. Ya que se trata de un hardware basado en Linux, existen numerosas variantes de este sistema que son compatibles. El sistema operativo principal es Raspbian, una versión de Debian adaptada a la placa Raspberry Pi, aunque existen muchas alternativas a éste. Y para acceder a este sistema existen diferentes formas: de forma local, siendo necesario un teclado y un ratón USB, y un monitor HDMI. O bien por conexión SSH, que permitirá acceder remotamente desde otro dispositivo sin necesidad de utilizar ningún otro tipo de hardware.

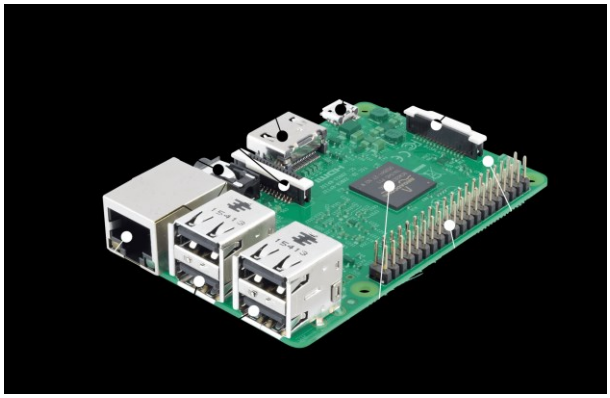


Figura 1-2. Raspberry Pi 3 modelo B.

1.4. Plataforma Home Assistant

También llamada de forma abreviada HA o HASS, es una plataforma abierta de automatización del hogar creada en Python 3, y que permite a través de diferentes módulos la interacción con diferentes plataformas, servicios y dispositivos. Su fundador fue Paulus Schoutsen, y se puede fijar el año 2014 como el inicio, aunque realmente todo comenzó en 2012 cuando inició un pequeño proyecto como hobby para controlar las bombillas Philips Hue poco después de su lanzamiento.

Actualmente, el proyecto cuenta con más de 1.000 componenetes disponibles, perfectamente organizados en varias categorías como Climate, Health o Weather, y con la posibilidad de utilizar también un buscador. La idea de los componentes es proveer una interfaz de comunicación entre la plataforma y los diferentes elementos externos, que pueden ser plataformas online como IFTTT, iCloud o APIs de terceros; sistemas que comparten un elemento físico y una plataforma cloud como pueden ser Nest o las Philips HUE; elementos puramente físicos que se pueden conectar directamente o a través de un protocolo como MQTT; y elementos software que permitan emitir notificaciones a bots de Telegram.

El modelo de creación de componentes se basa en el desarrollo directo de los responsables del proyecto así como de la gran comunidad existente, que crea y envía los componentes que genera para cubrir sus necesidades. En la página oficial se puede encontrar mucha documentación de calidad, con información relativa a la configuración de los componentes y su uso, acompañando siempre las explicaciones con ejemplos que facilitan su comprensión. Home Assistant también permite el uso de scripts escritos en Python para extender la funcionalidad y flexibilidad de la plataforma.

Aunque todo esté en inglés, es un idioma que se debe manejar sin demasiada dificultad. Además, está creciendo mucho el contenido en español de Home Assistant en Internet, tanto en páginas webs como en grupos específicos de usuarios que utilizan Telegram como medio para exponer sus proyectos y resolver sus dudas. Tiene una interfaz simple, clara y moderna, perfectamente adaptable a cualquier dispositivo, y sacan nuevas actualizaciones cada muy poco tiempo (1-2 semanas) para corregir errores o integrar nuevos componentes.

A pesar de todo, configurar un sistema de este tipo siempre tiene su dificultad, en este caso también se va a tener que utilizar en más de una ocasión el prueba y error.

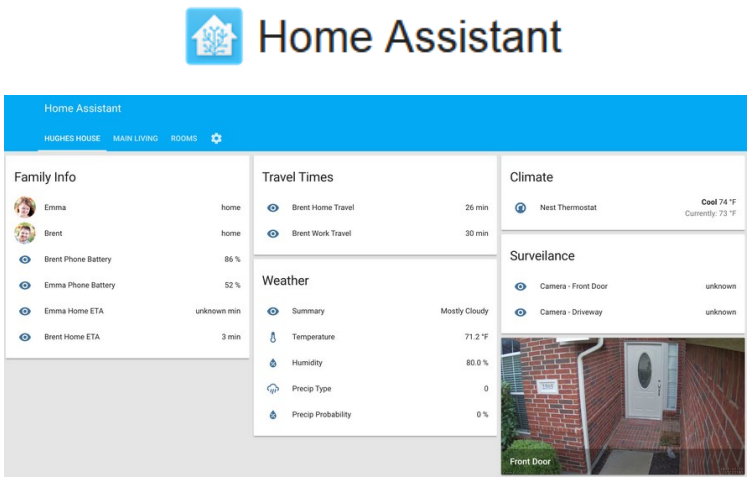


Figura 1-3. Logo e interfaz gráfica de Home Assistant.

1.5. Otras plataformas abiertas

En este apartado se van a comentar algunas de las alternativas más importantes que existen sobre las plataformas de código abierto para su uso como sistemas domóticos.

1.5.1. Domoticz

Es un software libre de control domótico disponible para las plataformas Windows y Linux y caracterizado por consumir muy pocos recursos del sistema, lo que lo convierte en una solución ideal para combinarlo con una Raspberry Pi. Fue una de las primeras plataformas de este tipo, ya que su primer lanzamiento fue en diciembre de 2012.

Ofrece soporte a un gran número de protocolos domóticos como pueden ser EnOcean, X10 o Z-Wave, así como una buena integración con diferentes dispositivos inalámbricos como mini estaciones meteorológicas o cámaras IP. Cuenta además con una interfaz web y aplicaciones móviles que convierten su uso multiplataforma en una tarea sencilla.

Por el contrario, su interfaz está algo desactualizada respecto a las tendencias actuales y, lo que es más importante, los diferentes módulos que lo componen están un poco desacoplados del núcleo del sistema, lo que hace más complicado el desarrollo de nuevos módulos y funcionalidades. Esta situación le ha llevado a que el número tecnologías disponibles, aunque amplio, sea algo menor que en otras opciones.

El software está escrito en C++ y la documentación no resulta demasiado completa ni atractiva lo que hace que gran parte de las dudas relacionadas con el software se acaben debatiendo e intentando solucionar en el foro de este proyecto, lo que no es una forma nada recomendable para recopilar información.

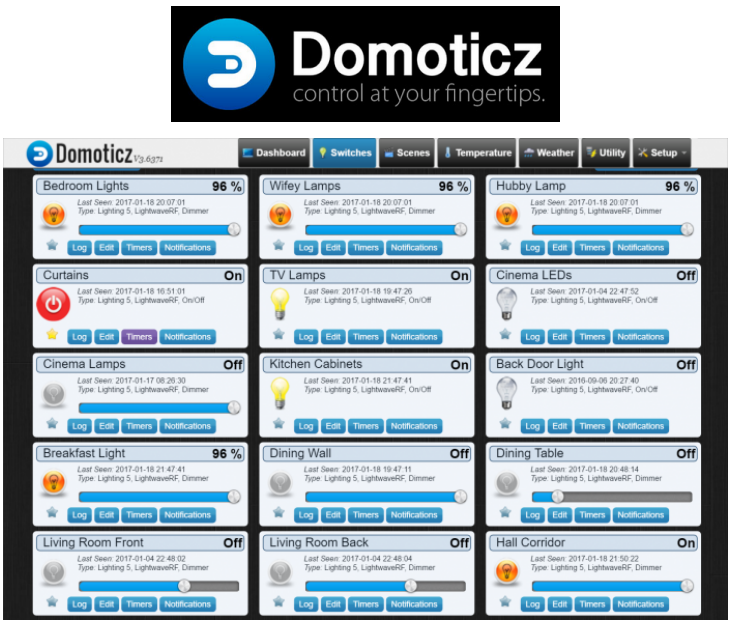


Figura 1-4. Logo e interfaz gráfica de Domoticz.

1.5.2. Jeedom

Es un software francés nacido en 2014, mucho más moderno en cuanto a su creación y a sus planteamientos, que se ha convertido en una solución relativamente conocida dentro del mercado doméstico europeo. Esta plataforma pertenece a la compañía que lo desarrolla y gracias a la cual se podría comprar un dispositivo con el software ya instalado y listo para funcionar. Sin embargo, el software es libre y se puede crear un sistema propio basado en éste.

Su arquitectura interna es mucho más modular aunque su juventud hace que el sistema todavía no sea demasiado robusto. La aparición de actualizaciones y nuevas funcionalidades es más lenta que en otras plataformas y tiene el inconveniente de que la mayor parte de la documentación e información se encuentra en francés y es bastante incompleta. En este caso también es fácil tener que recurrir al foro oficial para consultar toda clase de dudas relacionas con su implementación, donde se tendrá la barrera del idioma.

La plataforma está basada en una tienda o market donde está disponible un listado de todos los elementos que se pueden instalar mediante el plugin correspondiente. Muchos de los plugins son gratuitos, aunque hay algunos de pago, y su instalación es muy visual y sencilla.

Figura 1-5. Logo e interfaz gráfica de Jeedom.

1.5.3. OpenHAB

Fue iniciado por Kai Kreuzer en 2010 y tiene muchos co-desarrolladores. OpenHAB presume de ser un sistema domótico abierto que cubre una gran cantidad de sistemas: Windows, OSX, Linux, sistemas embebidos como NAS, etc. Su diseño se basa en una arquitectura totalmente modular donde se tiene el núcleo del sistema y sobre el que se construyen alrededor los diferentes módulos que le añaden funcionalidad.

Está escrito en Java y su modularidad ha permitido que cubra gran cantidad de elementos hardware y protocolos. Se puede trabajar con relativa sencillez con KNX, ZWave, X10, el termostato Nest, las bombillas HUE, bases de datos, servicios Web como IFTTT o una API de previsión del tiempo.

Si se tienen conocimientos sobre Java, aunque es una tarea algo compleja, se podría modificar algún módulo para adaptarlo a cualquier necesidad, lo que le aporta un plus de configurabilidad. El diseño visual es relativamente moderno, la documentación es muy completa y además cuenta con aplicaciones oficiales para iOS y Android. Es un proyecto más grande que los dos anteriores y cuenta con una amplia comunidad que se reúnen alrededor del foro oficial, aunque al ser un proyecto alemán se utiliza esta lengua para la resolución de dudas.

Esta plataforma aporta muchas ventajas, aunque también tiene la desventaja de ser la opción más pesada en cuanto a consumo de recursos y a espacio en disco. También puede resultar relativamente compleja en algunos aspectos de su configuración.

Figura 1-6. Logo e interfaz gráfica de OpenHAB.

2 SISTEMAS OPERATIVOS BASADOS EN LINUX

2.1. Un poco de historia

En realidad GNU/Linux, pero más conocido como Linux, es un sistema operativo compatible Unix, es decir, se creó basándose en este sistema propietario: nadie más tiene permiso para modificarlo y actualizarlo.

El sistema operativo Unix nació a principios de la década de los 70 por los desarrolladores Ken Thompson y Dennis Ritchie en los Laboratorios Bell, que pertenecen a la famosa compañía AT&T. Fue creado como un sistema operativo para manejar servidores, donde los comandos tienen casi todo el protagonismo.

Linux fue creado por Linus Torvalds a principios de los 90. Por aquel entonces un estudiante de informática de la Universidad de Helsinki, empezó, como proyecto de fin de carrera y sin poder imaginar en lo que se llegaría convertir, a programar las primeras líneas de código de este sistema operativo.

El sistema ha sido diseñado y programado por multitud de programadores alrededor del mundo. El núcleo sigue en continuo desarrollo bajo la coordinación de Linus Torvalds. A partir del año 1.991, que fue la primera distribución, se empezaron a crear muchas otras distribuciones basadas en Linux así como muchos escritorios.

Actualmente, existen distribuciones que están soportadas comercialmente, como Fedora (Red Hat), openSUSE (Novell), Ubuntu (Canonical Ltd.) y Mandriva; distribuciones mantenidas por la comunidad, como Debian y Gentoo; y distribuciones que no están relacionadas con ninguna empresa o comunidad, como es el caso de Slackware.

Hay dos características muy peculiares que lo diferencian del resto de los sistemas que podemos encontrar en el mercado: la primera es que es libre para su uso, o lo que es lo mismo, gratuito. La segunda, es que el sistema viene acompañado del código fuente, es decir, Linux se distribuye bajo la Licencia Pública General GNU (GPL), por lo tanto, este código fuente tiene que estar siempre accesible.

La parte central y más importante de un sistema operativo es el núcleo o kernel, que es el que se encarga de realizar toda la comunicación segura entre el software y el hardware del ordenador. En un sistema GNU/Linux, Linux es el núcleo. El resto del sistema está formado por un gran número de programas y librerías que hacen posible su utilización, muchos de los cuales fueron escritos por o para el proyecto GNU.

2.2. Debian y Raspbian

Debian fue la primera distribución de Linux en incluir un sistema de gestión de paquetes para permitir una fácil instalación y desinstalación del software. Además, también fue la primera que podía actualizarse sin necesidad de una reinstalación.

La combinación de la filosofía y metodología de Debian, las herramientas GNU, el núcleo Linux, y otro software libre importante, forman una distribución de software única llamada Debian GNU/Linux. Esta distribución está formada por un gran número de paquetes, y cada paquete contiene ejecutables, scripts, documentación e información de configuración. Estos paquetes tienen unos encargados que son los responsables de mantenerlo actualizado, hacer un seguimiento de los informes de fallos y comunicarse con los autores principales del programa empaquetado.

Raspbian es el sistema operativo de software libre recomendado para el uso en una Raspberry Pi. Está basado en Debian y optimizado para su hardware. Viene con más de 35.000 paquetes, software precompilado incluido en un buen formato para una fácil instalación. Raspbian es un proyecto comunitario en desarrollo activo, con énfasis en mejorar la estabilidad y el rendimiento de tantos paquetes Debian como sean posibles.

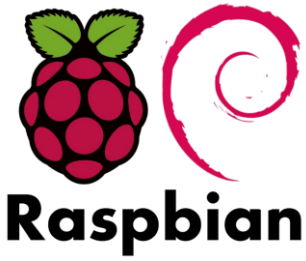


Figura 2-1. Raspberry Pi + Debian = Raspbian.

2.3. Características y funciones principales

En Linux hay un usuario especial conocido como el superusuario o administrador del sistema, que generalmente recibe el nombre de usuario *root*. El superusuario tiene acceso ilimitado al ordenador y puede hacer casi cualquier cosa. Cuando se ejecutan comandos como administrador siempre hay que tener cuidado con lo que se instala, ya que se puede dañar el sistema.

La administración de usuarios en Raspbian, cuando se inicia sesión en una Raspberry Pi, se realiza en la línea de comandos o en una ventana de terminal en caso de tener una interfaz gráfica. El usuario predeterminado es *pi* y la contraseña *raspberry*, y por defecto comienza en su carpeta de inicio */home/pi*. Es un sistema multiusuario, se pueden agregar tantos usuarios como se quieran y cambiar la contraseña de cada uno de ellos.

Este usuario *pi* en Raspbian es un *sudoer*. Esto le da la posibilidad de ejecutar comandos con privilegios de administrador cuando está precedido por ***sudo***, y para cambiar al usuario *root* con la orden ***sudo su***. Aunque también es posible conceder estos permisos a otros usuarios, se tendrá que tener cuidado por el tema de la seguridad.

Existen diferentes formas de instalar software en su Raspberry Pi, dependiendo de dónde se encuentre la fuente del software. El más común es a través de la herramienta de línea de comandos APT (Advanced Packaging Tool). Algunos softwares también se pueden instalar usando otros gestores de paquetes.

• Apt: La forma más fácil de administrar la instalación, actualización y eliminación de software es utilizar Apt de Debian. Si un paquete de software está empaquetado en Debian y funciona en la arquitectura Arm de Raspberry Pi, también debería estar disponible en Raspbian. Para instalar o eliminar paquetes, se necesitan permisos de administrador.

Apt mantiene una lista de las fuentes de software en su Pi en un archivo en la ruta */etc/apt/sources.list*. Antes de instalar el software, se debe actualizar su lista de paquetes disponibles y sus versiones con ***sudo apt-get update***, pero no instala o actualiza ningún paquete. Esta lista la coge de los servidores con repositorios que tenemos definidos en el *sources.list*.

Una vez el comando anterior ha descargado la lista de software disponible y la versión en la que se encuentra, podemos actualizar dichos paquetes usando el comando ***sudo apt-get upgrade***. Instalará las nuevas versiones respetando la configuración del software cuando sea posible. Para actualizar un paquete específico se puede usar ***sudo apt-get install somepackage***.

• Python: algunos softwares pueden instalarse usando el administrador de paquetes de Python ***pip***. Otros paquetes de Python se pueden encontrar en los archivos de Raspbian y se pueden instalar usando Apt. Este último es el método preferido de instalación de software, ya que significa que los módulos que instala pueden actualizarse fácilmente con los comandos ***sudo apt-get update*** y ***sudo apt-get upgrade***.

Los paquetes Python en Raspbian que son compatibles con Python 2.x siempre tendrán un prefijo ***python-*** y los paquetes de Python 3 se escribirán como ***python3-***. Un ejemplo de instalación sería ***sudo apt-get install python3-picamera*** y de eliminación ***sudo apt-get remove python-picamera***.

Pero no todos los paquetes de Python están disponibles en los archivos de Raspbian, y los que están a veces pueden estar desactualizados. Si no se encuentra una versión adecuada en los archivos de Raspbian, se puede

instalar paquetes del Python Package Index (PyPI).

Para hacerlo se usa la herramienta pip o pip3, en función de la versión Python utilizada (si no vienen instaladas en el sistema se usará el comando ***sudo apt-get install python-pip*** o ***python3-pip*** respectivamente). Un ejemplo de instalación de librería en Python 2 sería con el comando ***pip install unicornhat*** y otro de desinstalación en Python 3 sería ***pip3 uninstall unicornhat***.

Otra de las funcionalidades que existen es la de ejecutar tareas al iniciar el sistema. Si por ejemplo se quiere ejecutar un script al iniciarse la Raspberry Pi, se pondría ***@reboot python /home/pi/myscript.py***. Esto ejecutará su secuencia de comandos de Python cada vez que se reinicie la Raspberry Pi. Si se quiere que se ejecute siempre en un segundo plano mientras está en funcionamiento, hay que agregar ***(espacio)&*** al final del anterior comando indicado.

2.4. Entorno virtual

Básicamente, consiste en realizar una instalación de Python aislada para poder administrar la instalación de paquetes de forma independiente a los ya instaladas en el sistema principal o en otros entornos.

Se pueden crear tantos entornos virtuales como se quieran para distintos proyectos y no tener que preocuparse por corromper los paquetes ya instalados. Esto ocurre con los desarrollos de las librerías y sus distintas versiones cuando introducen mejoras y necesitan hacer pruebas.

Una vez creado el Entorno Virtual Python y antes de empezar a instalar y utilizar los paquetes, hay que activarlo. Esto es necesario para situarnos en el entorno que queramos trabajar. Los paquetes de Python instalados en el EV se almacen en la ruta *./usr/local/lib/python3.5/site-packages/*.

2.5. Comandos Linux

Para poder realizar este trabajo fin de máster se han tenido que utilizar todos estos comandos de Linux:

ls: enumera el contenido, archivos y carpetas, del directorio donde te encuentres (***ls***) o uno que se especifique (***ls /ruta/del/directorio***).

cd: cambia el directorio actual por el especificado (***cd /home/pi/directorio***). Si se utiliza el comando sólo (***cd***) y se está dentro de un directorio, se sale de toda la ruta donde estés.

mkdir: crea un nuevo directorio, en el que estés por defecto (mkdir /nuevodirectorio) o en una ruta especificada (***mkdir /ruta/del/nuevodirectorio***).

touch: crea un archivo vacío si éste no existe (***touch archivo.py***).

rm: elimina algún archivo (***rm archivo.txt***) o directorio (***rm -r /directorio***). También se pueden usar rutas en los dos casos.

cp: copia archivos de una ruta a otra (***cp /rutaorigen/archivo.txt /rutadestino/archivo.txt***).

mv: mueve (corta) un archivo o directorio a una ruta especificada. Funciona igual que el comando cp.

cat: para ver el contenido de un archivo pero sin poder editarlo (***cat archivo.py***).

wget: descarga un archivo desde internet hasta tu ordenador (***wget dirección_web***).

nano: para editar un archivo (***nano /ruta/archivo.txt***).

su: cambia a modo Superusuario o "root" (***su***). Sería el acceso como administrador del sistema, necesario para hacer cambios importantes o acceder a determinados archivos. También se usa para cambiar de usuario (***su pi*** ó ***su juan***).

sudo: para ejecutar alguna orden en modo Superusuario (***sudo comando***) pero como otro usuario, es decir, simula ser el otro usuario sin cambiar verdaderamente el usuario actual. Ello implica que se puede ejecutar un comando como administrador y, al segundo siguiente, volverá a tener los privilegios del usuario que se estaba usando antes.

exit: para salir del usuario que esté y volver al de defecto. También para salir de la sesión SSH.

passwd: cambiar la contraseña de usuario (***passwd***). Si queremos cambiar la contraseña de *root* sería ***sudo passwd***. Hay que tener cuidado al cambiar las contraseñas con las minúsculas y mayúsculas ya que no se visualiza en el terminal lo que se está tecleando.

adduser: crea un usuario nuevo (***sudo adduser nombre***). Se puede dejar sin contraseña pulsando intro cuando la pida.

userdel: elimina un usuario en el sistema. Si se utiliza –r se elimina también sus carpetas asociadas (***sudo userdel -r usuario***).

reboot: reinicia el sistema operativo. Se necesita *sudo* delante.

poweroff: apaga el sistema operativo. Se necesita *sudo* delante.

clear: limpia el texto del terminal.

ping: verifica si se puede establecer comunicación con otro host. Se puede usar especificando un nombre de host (***ping raspberrypi.org***) o una dirección IP (***ping 8.8.8.8***).

ifconfig: visualiza los detalles de configuración de la red para las interfaces en el sistema actual (***ifconfig***).

curl: permite descargar archivos de internet, publicar datos en un formulario web o subir archivos a través de FTP.

chmod: significa *cambiar modo*, es el comando utilizado para cambiar (añadir o quitar) los permisos de lectura, escritura y ejecución de archivos.

cron: significa *tiempo*. Es un administrador regular de procesos en segundo plano (demonio) que los ejecuta a intervalos regulares. Los procesos que deben ejecutarse y las opciones de ejecución se especifican en el fichero crontab.

crontab: es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario. Crontab verificará la fecha y hora en que se debe ejecutar el script o el comando y los permisos de ejecución, y lo realizará en segundo plano. Por ejemplo, el comando ***crontab -e*** edita el archivo crontab del usuario.

pip3 show *librería* | grep Version: muestra la versión de la librería indicada que está instalada en el sistema.

pip3 list installed --format=columns: da por pantalla la lista de librerías instaladas y sus versiones.

raspi-config: con este comando precedido de ***sudo*** se podrá realizar una configuración inicial de la Raspberry Pi. En este menú se puede cambiar el idioma, la localización, la zona horaria, el teclado, el nombre del equipo, agrandar la partición de la tarjeta SD y otros parámetros.

lsb_release -a: para saber la versión de Linux que se tiene instalada.

Control + c: sale del ping que se esté ejecutando por pantalla. También se usa en otros comandos que dan información por pantalla y hasta que no se para por el usuario no dejan de ejecutarse.

Copiar algún texto de la ventana del terminal de comandos: seleccionar el texto deseado y automáticamente se copia en el portapapeles. No funciona el *control* + *c* de Windows.

Pegar algún texto en la ventana del terminal de comandos: botón derecho del ratón donde se encuentra el cursor en la ventana del terminal y automáticamente se pega. No funciona *control* + *v* de Windows.

3 REQUISITOS HARDWARE Y SOFTWARE

3.1. Hardware

Para el desarrollo de este sistema domótico se han utilizado los siguientes elementos físicos:

- Raspberry Pi 3 modelo B + transformador de alimentación de 2,5 A + carcasa de plástico + disipadores térmicos.
- 2 Tarjetas microSD clase 10 de 32 GB marca *SanDisk*.
- SAI marca *Riello*.
- 2 Regletas de 4 tomas con conexión C14 IEC320 para el SAI.
- Cámara IP Wireless HD, modelo C1 versión 3 de *Foscam*.
- 3 enchufes adaptadores UE-China.
- Gateway de *Xiaomi* 3 sensores de Temperatura y Humedad de *Xiaomi*.
- Sensor de Temperatura, Humedad y Presión de *Xiaomi*.
- Sensor de movimiento de *Xiaomi*.
- Pulsador Wireless de *Xiaomi*.
- Bombilla Wifi RGB de *Yeelight*.
- Controlador universal IR de *Broadlink*.
- Enchufe Wifi SP3 de *Broadlink*.
- Regleta Wifi MP1 de *Broadlink*.
- Tableta Lenovo Tab 4 de 8 pulgadas.
- Robot de limpieza Mi Vacuum, versión 1 de *Xiaomi*.
- Cable Ethernet.
- Cable USB tipo A-B para comunicación del SAI.
- Conector IEC hembra a SCHUKO para alimentación del SAI.
- Portátil personal marca *Toshiba*.
- Dispositivo móvil iPhone 6s.
- Sistema de climatización centralizado marca *Acson*.
- Router Sercomm H500-S de la compañía *Vodafone*.
- TV *Samsung* modelo UE40D6500.

3.2. Software

Para el diseño e implementación de este sistema domótico se han utilizado los siguientes sistemas operativos y aplicaciones:

- Windows 7, en el portátil personal.
- Sistema Operativo iOS versión 11.3.1, en el iPhone.
- Sistema Operativo Android Nougat versión 7.1.1, en la tableta *Lenovo*.
- Hassbian v.1.3.2 basado en Debian 9.4 Stretch, en el sistema operativo.
- Home Assistant versión 0.66.1, en el sistema domótico.
- *Telegram* app.
- *Life360* app.
- *Foscam* app.
- Mi Home app de *Xiaomi*.
- e-Control app de *Broadlink*
- *Yeelight* app.

4 FUNCIONAMIENTO DE HOME ASSISTANT

4.1. Archivos de configuración YAML

Home Assistant se instala en un entorno virtual en la ruta */srv/homeassistant*. Se inicia como un servicio ejecutado por el usuario *homeassistant* y su configuración se encuentra localizada en la dirección */home/homeassistant/.homeassistant*. Con esto se asegura que las instalaciones de Python y de Home Assistant no interfieren entre sí.

Para poner en marcha los componentes y configurar la instalación, se tendrá que trabajar con los archivos de configuración de Home Assistant. Cuando se inicia por primera vez, HA escribe un archivo de configuración predeterminado que permite el acceso mediante interfaz web y el descubrimiento de los dispositivos que estén en la red de casa. Este primer inicio puede tardar hasta un minuto.

Actualmente existe la opción de manejar la configuración directamente desde la interfaz web, aunque todavía no está muy conseguido. La mejor opción es conectarse mediante SSH y editar los archivos con un editor de texto desde el mismo Windows (teniendo instalado Samba), o incluso mediante comandos desde el terminal (utilizando Putty).

En este caso, para un sistema operativo Linux, el archivo principal que contiene los componentes que se cargarán con sus configuraciones se llama *configuration.yaml* y se encuentra en la ruta por defecto comentada en la introducción. Una vez se realicen los cambios en este archivo, se deberá reiniciar el servicio homeassistant para que tengan efecto. Se puede hacer desde el intérprete de comandos con *sudo systemctl restart home-assistant@homeassistant.service*, o desde la interfaz web de HA, haciendo clic en el icono superior izquierdo, seleccionando *Configuration* y activando la opción *Restart* desde la sección *Server Management*.

Home Assistant usa el formato YAML que significa *Another Another Markup Language* y, al igual que Python, utiliza espacios en blanco (no tabulaciones) para delimitar las secciones de código. Por defecto, emplea una sangría de dos espacios para cada sección anidada y en caso de problemas, se reciben mensajes de error al iniciar el servicio.

Se podrán poner comentarios aclaratorios en el código de los archivos YAML poniendo el símbolo # delante del texto, lo que hará que el sistema los ignore. Ejemplo:

```
input_select:
  threat:
    name: Threat level
    # A collection is used for options
    options:
      - 0
      - 1
      - 2
      - 3
    initial: 0
```

Para mejorar la legibilidad del código del archivo de configuración principal, se recomienda separar en distintos archivos YAML en función de los componentes declarados, con una sintaxis del tipo *lights: !include lights.yaml*. Esta declaración le dice a Home Assistant que inserte el contenido de *lights.yaml* en ese punto. Hay que tener en cuenta que solo puede haber un *!include* para cada componente.

Otra de las opciones que hay será la de incluir uno o varios ficheros .yaml dentro de una carpeta del directorio principal de *homeassistant*. Para ello se añadirá *device_tracker: !include_dir_merge_list device_tracker* el cual devolverá el contenido del directorio *device_tracker* como una lista, siempre que su código empiece por guión.

También es importante no almacenar datos privados (contraseñas, claves API, etc.) directamente en el archivo *configuration.yaml*. Estos datos se pueden almacenar en un archivo separado como *secret.yaml*, lo que evita problemas de seguridad.

4.2. Configuración de dispositivos y problemas comunes

Home Assistant puede descubrir automáticamente muchos de los dispositivos y servicios que se tengan configurados en la red. Para instalarlos e integrarlos correctamente en el sistema, se tendrá que consultar la página de componentes de la web oficial de HA e incluso el foro, donde los usuarios exponen sus problemas y soluciones.

Las *entities* o entidades son variables que proporcionan datos, como un sensor o un interruptor. Las plataformas, como *Dark Sky* o *Wunderground*, normalmente proporcionan acceso a múltiples *entities* (temperaturas mínimas/máximas, pronóstico, etc.). Se puede ver una lista de las entidades, sus nombres y sus valores, en la interfaz web en el menú *Developer tools* → *States* (< >) → *Entities*.

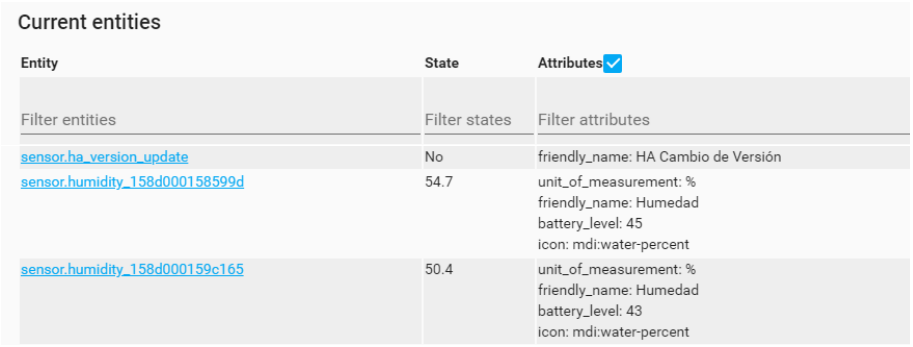


Figura 4-1. Lista de entidades.

Es común que ocurran problemas al configurar Home Assistant, ya que es posible que un componente no se muestre o que esté actuando de manera errónea. Todas estas advertencias se almacenarán en *home-assistant.log* que se encuentra en el directorio principal de configuración y se puede consultar. Este archivo se reinicia al inicio de HA.

Se podrían dividir en dos tipos los errores de configuración:

1. Por un lado, los errores de formato en los archivos *.yaml*. Se tienen que tener muy en cuenta los espacios que se indican al declarar cualquier componente, siendo siempre pares, y también el espaciado de los distintos niveles de anidamiento del código.

También es importante saber que los únicos caracteres válidos en los nombres de las entidades son las letras minúsculas, los números y los subrayados. Si se crea una entidad con otros caracteres, puede que Home Assistant no genere un error para esa entidad, es decir, que fallen en silencio.

2. Por otro lado, los errores propios de Home Assistant o del Sistema Operativo. Es importante que se tenga siempre todo actualizado y se revise el foro oficial para comprobar la posible solución a los errores que se pueden dar en las sucesivas actualizaciones de HA y del SO. Estos errores se pueden dar en los códigos de los archivos Python del sistema o en las versiones de las librerías que se tengan instaladas y que no funcionen con las nuevas actualizaciones.

Por eso, antes de actualizar, lo mejor será leer las notas de la versión y comprobar que la actualización no vaya a estropear ninguna configuración anterior, ya que a veces se rediseña el código de configuración de los componentes existentes y se tiene que modificar su integración para que HA lo reconozca correctamente.

4.3. Organizando las vistas

A medida que se van agregado componentes, la interfaz web empieza a desordenarse con gran cantidad de elementos dispersos por toda la página. Para evitar esto, se pueden utilizar grupos y vistas para limpiar la interfaz y colocar elementos relacionados en una determinada pestaña.

Un *group* o grupo es simplemente un objeto que contiene una lista de entidades. Visualmente, un *group* se

representa como un panel o una ficha. Por defecto, el grupo *group. all_devices* existe y almacena los elementos descubiertos por la plataforma de seguimiento de dispositivos. Los grupos normalmente contienen una lista de entidades.

Una *view* o vista se presenta como una pestaña separada dentro de Home Assistant. Las *views* son en realidad grupos de grupos y difieren de los grupos normales por tener la propiedad *view: yes*. También se pueden agregar entidades individuales así como grupos a una vista.

Normalmente, cada entidad necesita su propia entrada en el archivo *configuration.yaml*. Hay dos estilos para entradas múltiples:

1. Agrupar todas las entidades debajo de una principal:

```
sensor:
  - platform: mqtt
    state_topic: "home/bedroom/temperature"
    name: "MQTT Sensor 1"
  - platform: mqtt
    state_topic: "home/kitchen/temperature"
    name: "MQTT Sensor 2"
  - platform: rest
    resource: http://IP_ADDRESS/ENDPOINT

switch:
  - platform: vera
```

2. Indicar cada dispositivo separadamente. Para ello hay que diferenciar las entradas con un número o cadena única:

```
media_player livingroom:
  platform: mpd
  server: IP_ADDRESS
media_player kitchen:
  platform: plex
camera 1:
  platform: generic
camera 2:
  platform: mjpeg
```

Por defecto, el primer grupo se llamará *default_view* y es el que se muestra al inicio de sesión. Las pestañas se pueden definir para que aparezcan con un nombre o con un icono, y cada entidad sólo se puede utilizar una única vez en el grupo donde queramos que aparezca, no pudiéndose usar en varios grupos.

A continuación se muestra un ejemplo de configuración, donde el código y la interfaz quedarían así:

```
group:
  default_view:
    view: yes
    icon: mdi:home
    entities:
      - group.kitchen
      - group.awesome_people
      - group.climate
  upstairs:
    name: Kids
    icon: mdi:account-multiple
    view: yes
    entities:
      - light.bedroom
      - media.player.bedroom
```

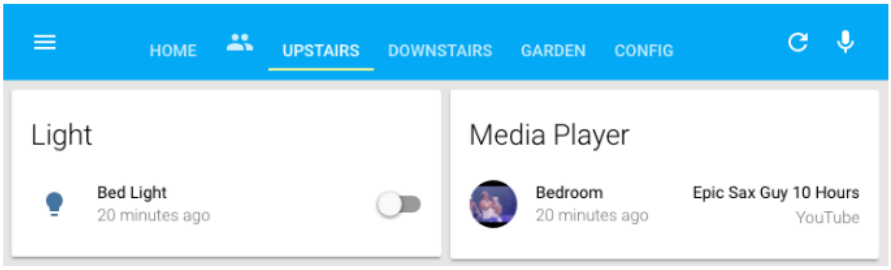


Figura 4-2. Interfaz limpia con *views* y *groups*.

4.4. Personalización de nombres e iconos

De forma predeterminada, todos los dispositivos estarán visibles y tendrán un nombre y un icono por defecto, determinado por su dominio. Se puede personalizar el aspecto de su página principal modificando algunos de estos parámetros, anulando los atributos de cada entidad específica.

Para realizar todos estos cambios, se tendrá que añadir una sección personalizada al principio del archivo de configuración, bajo la etiqueta *homeassistant*: y sangrada por dos espacios.

Se va a mostrar un ejemplo para cambiar los iconos de los switches por algo más apropiado. Se pueden utilizar los iconos de Material Design o unas imágenes propias. El archivo *configuration.yaml* quedará así:

```
homeassistant:
  name: Home
  unit_system: metric
  # etc

  customize:
    sensor.living_room_motion:
      hidden: true
    thermostat.family_room:
      entity_picture: https://example.com/images/nest.jpg
      friendly_name: Nest
    switch.wemo_switch_1:
      friendly_name: Toaster
      entity_picture: /local/toaster.jpg
      initial_state: 'on'
      icon: mdi:kettle
    switch.rfxtrx_switch:
      assumed_state: false
```

Otros parámetros que se pueden personalizar son: el estado inicial (*on/off*), la unidad de medida o la forma de visualización en que aparezca un interruptor (*assumed_state*).

4.5. Automatizaciones

Las automatizaciones constan de tres partes diferentes: *Trigger* (disparador), *Condition* (condición) y *Action* (acción). Un ejemplo de regla de automatización del hogar sería la siguiente: cuando Pepe llega a casa (disparador) y es después del atardecer (condición), entonces enciende las luces en la sala de estar (acción).

La primera línea es el disparador de la regla de automatización. Éstos describen eventos que deberían desencadenar la regla, siendo posible especificar múltiples disparadores para la misma regla. Una vez que se inicia un disparador, Home Assistant validará las condiciones, si las hay, y llamará a la acción. En este caso, se trata de una persona que llega a casa, y por tanto el estado de Pepe cambiaría de *not_home* a *home*.

La segunda línea es la condición. Las condiciones son opcionales y se pueden usar para evitar que ocurra una acción cuando se activa. Las condiciones parecen muy similares a los disparadores pero son muy diferentes. Un disparador observará los eventos que suceden en el sistema, mientras que una condición solo observa cómo se ve el sistema en este momento. Un disparador puede observar que se está encendiendo un interruptor, sin embargo una condición solo puede ver si un interruptor está actualmente encendido o apagado (estado actual). En este caso, solo se quiere actuar cuando el sol se haya puesto.

La tercera parte es la acción, que se realizará cuando se desencadene una regla y se cumplan todas las condiciones. Aquí también se pueden ejecutar múltiples acciones, procesándose una detrás de otra en el orden en que se ingresaron. Las acciones tienen que ver con los servicios de llamadas de Home Assistant y para explorar los que están disponibles se tiene que abrir la herramienta de desarrollador *Servicios*. Cada servicio tiene un dominio y un nombre, pudiendo por ejemplo encender cualquier luz (*light.turn_on*), activar un script (*script.turn_on*) o apagar un interruptor (*switch.turn_off*).

```
automation:
  alias: Turn on the lights when the sun sets
  trigger:
    platform: sun
    event: sunset
    offset: '-01:00:00'
  condition:
    condition: state
    entity_id: group.all_devices
    state: 'home'
  action:
    service: light.turn_on
    entity_id: light.living_room
```

4.6. Scripts

Los scripts son una secuencia de acciones que Home Assistant ejecutará. Estas secuencias están disponibles como una entidad a través de un componente independiente, aunque también puede incorporarse en una automatización.

La estructura básica de la sintaxis del script es una lista de acciones que contienen las acciones a realizar. Si una secuencia de comandos contiene solo una acción, la lista se puede omitir quitando los guiones.

Al ejecutar un script, también se puede agregar una o varias condiciones para detener la ejecución posterior. Cuando una condición no se cumple, el script finalizará. Un ejemplo sería el siguiente:

```
script:
  example_script:
    sequence:
      - service: light.turn_on
        data:
          entity_id: light.ceiling
      - service: notify.notify
        data:
          message: 'Turned on the ceiling light!'
```

Aquí también es común incorporar retrasos o *delays* para suspender temporalmente el script e iniciarlo más adelante, o jugar con la secuencia de tiempo en el que ejecutamos una serie de acciones.

4.7. Sensor template

Una funcionalidad muy importante en Home Assistant es la plataforma de *template* o plantilla, utilizada para sensores que se pueden dividir en atributos de estado. Un ejemplo sería que se muestre en HA el ángulo que va teniendo el Sol en todo momento. Para ello se utiliza uno de los atributos que tiene el componente *sun*:

```
sensor:
  - platform: template
    sensors:
      solar_angle:
        friendly_name: "Sun Angle"
        unit_of_measurement: '°'
        value_template: "{{ '%+.1f'|format(states.sun.sun.attributes.elevation) }}"
```

Otro ejemplo sería poder cambiar el icono y renombrar el nombre de salida en función de si es día o noche:

```
sensor:
  - platform: template
    sensors:
      day_night:
        friendly_name: "Day/Night"
        value_template: >-
          {% if is_state('sun.sun', 'above_horizon') %}
            Day
          {% else %}
            Night
          {% endif %}
        icon_template: >-
          {% if is_state('sun.sun', 'above_horizon') %}
            mdi:weather-sunny
          {% else %}
            mdi:weather-night
          {% endif %}
```

Más utilidades que tiene este *template* serían para el cambio de la unidad de medida o la foto de entidad.

4.8. Instrucciones más importantes

Iniciar/parar/reiniciar Home Assistant:

```
sudo systemctl start home-assistant@homeassistant.service
sudo systemctl stop home-assistant@homeassistant.service
sudo systemctl restart home-assistant@homeassistant.service
```

Activar el Entorno Virtual (estando con el usuario *pi*):

```
sudo su -s /bin/bash homeassistant   #entra como usuario homeassistant
source /srv/homeassistant/bin/activate   #activa el Entorno Virtual
```

Ejecutar manualmente Home Assistant (dentro del Entorno Virtual):

```
hass
```

Comprobar la configuración de HA (dentro del Entorno Virtual):

```
hass --script check_config
```

Leer el archivo log de errores de HA (estando con el usuario *pi*):

```
sudo su -s /bin/bash homeassistant
cd /home/homeassistant/.homeassistant
nano home-assistant.log
```

Editar el archivo de configuración de HA (estando con el usuario *pi*):

```
sudo su -s /bin/bash homeassistant
cd /home/homeassistant/.homeassistant
nano configuration.yaml
```

4.9. Seguridad y acceso remoto

Si se tiene pensado usar Home Assistant desde fuera de la LAN, por ejemplo desde Internet, se tienen varias opciones. Uno de ellas es habilitar el soporte HTTPS y redireccionar el puerto 8123 en tu router. Esto permite realizar conexiones cifradas pero expone la instalación a Internet, es decir, podrían existir vulnerabilidades que permitirían a los atacantes tomar el control del sistema local.

Una segunda opción es configurar una VPN en tu router que permite realizar una conexión y un acceso a HA de forma segura. Si se va a utilizar HTTPS, para que funcionen todos los servicios hay que proporcionar certificados SSL válidos (no autofirmados). Para obtener certificados válidos se necesita tener un nombre DNS público (por ejemplo, usando un servicio DNS dinámico como *duckdns.org*) y recurriendo a *letsencrypt.org* para configurar un certificado SSL válido para la instalación.

Independientemente del método de acceso (HTTP o HTTPS), se tendrá que configurar una contraseña. La plataforma no soporta múltiples cuentas de usuario pero se puede definir una contraseña API que se necesitará para iniciar sesión en la interfaz web. La mejor forma de hacer esto es creando un archivo que almacene todos los datos confidenciales, como contraseñas, códigos API y URLs. Se llamará *secrets.yaml* y se referenciará en el archivo *configuration.yaml*.

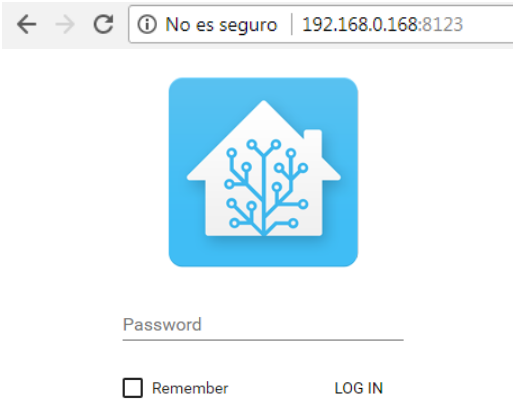


Figura 4-3. Autentificación en Home Assistant.

Con todo esto se intenta evitar que cualquiera pueda acceder a tu sistema domótico simplemente conociendo la dirección HTTPS. Si se quiere introducir directamente en el archivo *configuration.yaml* se tendría que poner el código:

```
http:
  api_password: bienveni2
```

Si se quiere introducir en *secrets.yaml*, se tendría que poner en el archivo principal de configuración:

```
http:
  api_password: !secret http_password
```

Y dentro del archivo *secrets.yaml* se pondría la contraseña asignada con su correspondiente identificador:

```
http_password: bienveni2
```

4.10. Interfaz de usuario

A continuación se va a explicar el funcionamiento a nivel web de Home Assistant. Aunque a día de hoy cada vez se pueden hacer más cosas a través de esta interfaz web, en este trabajo se ha optado por realizar las configuraciones sin ayuda de algunas de las herramientas disponibles, es decir, editando directamente los archivos del sistema.

Home Assistant cuenta con una aplicación nativa para iOS, mientras que para los clientes Android se puede configurar la página como pantalla de inicio, en Chrome, dirección http://ip-raspberry-pi:8123, Menu, Add to

homescreen, o instalar una app no oficial que tiene el mismo nombre que la plataforma.

Una vez que se realiza la instalación del Sistema Operativo, se accede a la página personal de Home Assistant. Para ello, se ingresa en la barra de direcciones del navegador la dirección IP asignada a la Raspberry Pi, seguido del número de puerto 8123 para que se pueda conectar: ***http://192.168.0.168:8123/***

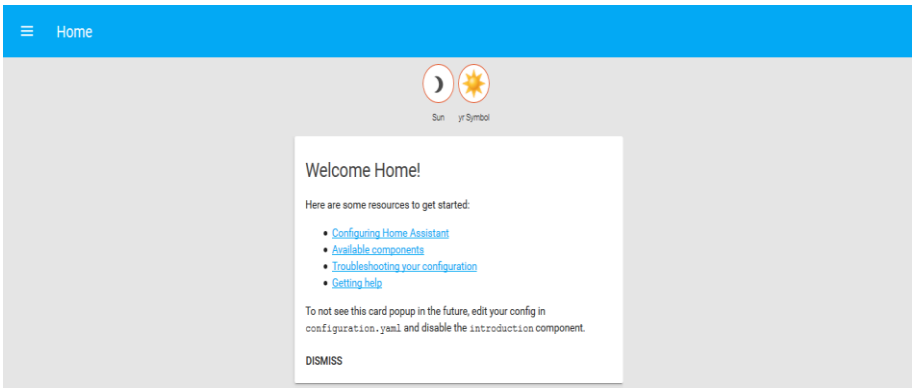


Figura 4-4. Pantalla inicial.

En este primer acceso donde todavía no hay nada configurado, únicamente se verán dos símbolos meteorológicos (sensores) que vienen por defecto ya configurados, y un mensaje de aviso con información sobre cómo empezar a configurarlo. En la zona donde están los círculos es dónde irán apareciendo todos los sensores que se instalen y, en la zona del mensaje, los interruptores, cámaras, automatizaciones y scripts que se vayan configurando. Todo lo que se integre en HA se puede organizar por pestañas y/o grupos en función del diseño que se le quiera dar a la página.

Cuando se accede al menú principal, arriba a la izquierda, se despliegan los diferentes apartados de los que consta la plataforma:

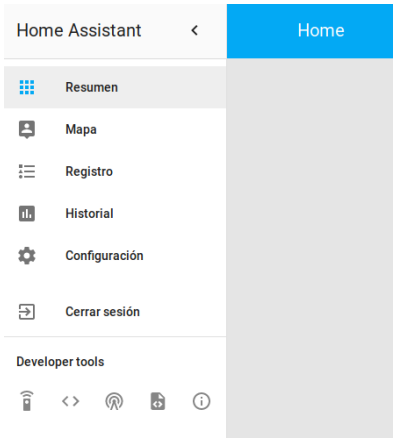


Figura 4-5. Menú lateral.

Resumen: es la pantalla principal que aparece cuando se conecta a Home Assistant, donde aparecerá el cuadro de mandos domótico de todo lo que se configure.

Mapa: muestra la ubicación que se obtiene por medio de la IP, o bien, porque en la configuración se hayan indicado las coordenadas GPS.

Registro: muestra un registro secuencial donde se indica cómo van cambiando los estados de los dispositivos que hay integrados, indicando la hora y el orden en que se van sucediendo.

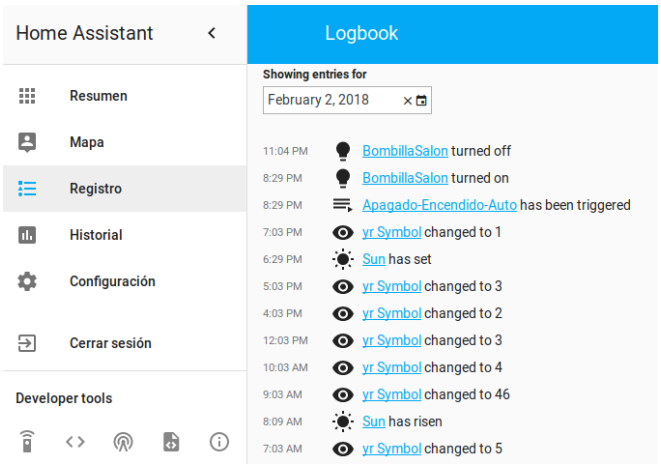


Figura 4-6. Registro.

Historial: es parecido al registro. En este caso, se ven todos los componentes por colores y todas las activaciones y cambios que ha habido a lo largo del día. También se puede consultar un periodo determinado a nuestra elección.

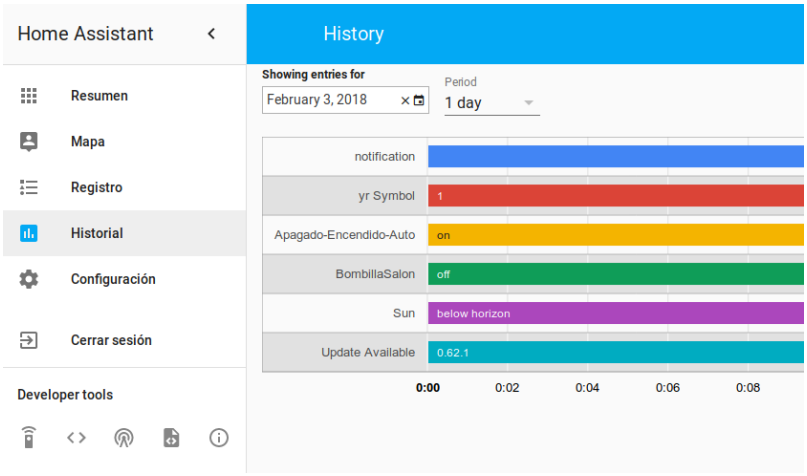


Figura 4-7. Historial.

Configuración: en este apartado se puede realizar la configuración de los dispositivos y automatizaciones. Dentro se pueden ver diferentes apartados:

- *Home Assistant Cloud*: es un servicio opcional creado para mantener el proyecto ofreciendo alojamiento en la nube.
- *General*: fundamentalmente se usa para validar el archivo de configuración, es decir, te dice si éste tiene errores de escritura.
- *Customization*: aquí se pueden personalizar los dispositivos, como por ejemplo dándole un nombre o un icono específico.
- *Automation*: se pueden configurar los automatismos que queramos introducir en nuestro sistema domótico. Un ejemplo sería: si el sensor X detecta movimiento, entonces avisa por Telegram y enciende la alarma.
- *Script*: es igual que el apartado anterior, pero aquí sólo se definen los scripts o comandos que se ejecutan dentro de *Automation*.

Herramientas para desarrolladores/*developer tools*: por último, al final del margen izquierdo, existen unas

herramientas muy importantes para la configuración y gestión de Home Assistant.

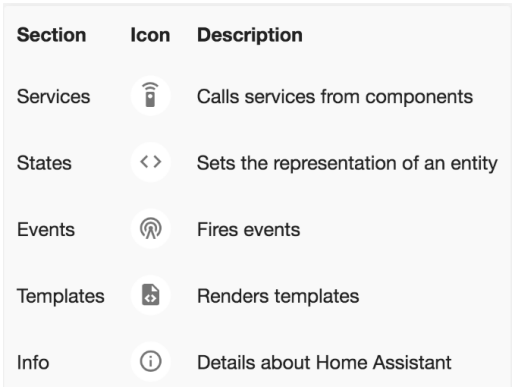


Figura 4-8. Herramientas de desarrollador.

- *Services*: permite realizar llamadas a los diversos servicios que presentan los componentes. Se pueden hacer cosas como activar una automatización, volver a cargar la configuración de HA o enviar una notificación de prueba al móvil. Los servicios disponibles pueden variar en función de los componentes que se tengan activos en la configuración. Se puede probar cualquier acción antes de añadirla a una automatización.

  Cuando se hace un cambio en el archivo principal *configuración.yaml*, siempre se tiene que guardar y reiniciar Home Assistant. Hay dos formas de hacerlo: por comandos como se indicó anteriormente, y a través de un servicio de reinicio, el cual tiene una gran utilidad.

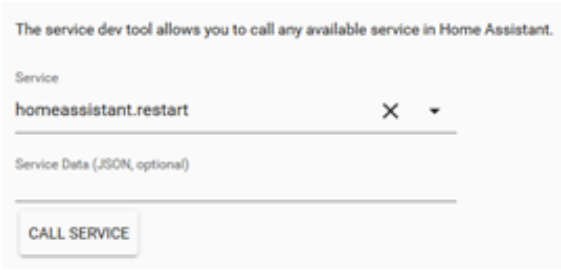


Figura 4-9. Servicio de reinicio.

- *States*: se enumeran todas las entidades, con su estado y atributos actuales. Se utiliza esta lista para encontrar una entidad concreta, ya sea para saber cómo hacer referencia a ella en la configuración, por ejemplo las entidades visibles en una vista, o para usarla en una plantilla.
- *Events*: permite generar un evento en el bus de eventos. Existen varios eventos disponibles, aunque en la práctica no se suele utilizar mucho.
- *Templates*: El sistema de plantillas de HA utiliza la sintaxis de plantillas de Jinja2 añadiendo algunas variables internas. El objetivo es procesar los datos de entrada o salida y darles un formato específico.

  La vista de Plantillas proporciona un espacio de trabajo donde se puede experimentar y probar la sintaxis antes de pasarla al archivo de configuración. Cuando se cargue la página por primera vez, aparecerá un ejemplo de sintaxis que, entre otras cosas, pasará por todos los sensores mostrando sus valores actuales.
- *Info*: te muestra la versión actual, así como los errores y avisos que se han ido registrando al iniciarse Home Assistant, al igual que se hace en el archivo *home-assistant.log*.

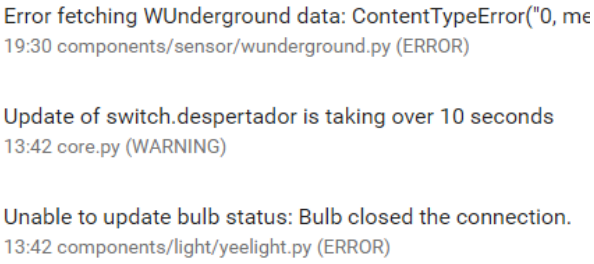


Figura 4-10. Información de avisos y errores.

5 CONFIGURACIÓN DEL SISTEMA DOMÓTICO

5.1. Sistema operativo

En este proyecto se va a utilizar Hassbian versión 1.31. Es un sistema operativo personalizado para Raspberry Pi 3 y la forma más fácil de instalar Home Assistant. Éste consiste en un sistema dedicado basado en Raspbian Stretch (versión ligera) que lleva HA ya incorporado. Éste a su vez se basa la versión 9.4 de Debían Stretch, de donde coge el segundo nombre, y sustituye al anterior sistema llamado Raspbian Jessie. La principal novedad en esta versión es que ahora Home Assistant se ejecuta en Python 3.5.

Como en las demás versiones, incorpora un entorno virtual ya creado para el uso de Home Assistant y varios scripts ya instalados para su uso, como Mosquitto (MQTT) o Samba.

La otra alternativa consiste en instalar el sistema operativo Raspbian Stretch y luego Home Assistant. Este caso da la opción de poder utilizar la Raspberry Pi para otros proyectos al ser un sistema no dedicado. La contra a su versatilidad es una instalación un poco más larga y tediosa, como por ejemplo el entorno virtual.

La última versión de Hassbian siempre se va a encontrar en este repositorio de GitHub. Una vez bajada la imagen (.img) del sistema operativo se grabará en una tarjeta microSD, y se recomienda que tenga una capacidad mínima de 4GB aunque en mi caso utilizaré una tarjeta de 32GB. Antes que nada se formatea la tarjeta, por ejemplo con el programa gratuito SDFormatter. Una vez realizado, se grabará la imagen de Hassbian en la microSD con el programa libre Win32DiskImager.

Esta imagen instalará la última versión de Home Assistant en el arranque inicial. Se tiene que tener en cuenta que tras el primer inicio o un inicio tras una actualización, puede ser bastante lento el arranque por lo que se tiene que dejar que se ejecute el sistema durante unos minutos hasta que se pueda acceder a la interfaz web de la plataforma.

5.2. Conexión mediante SSH y primeras configuraciones

Una vez que ya se tiene la imagen grabada en la tarjeta, se introduce en la ranura de la Raspberry Pi y se le conecta también un cable Ethernet desde el router. El objetivo será conocer la dirección IP local que le adjudica el router para la conexión a internet.

Para ello hay que entrar en la página de configuración del router. En este caso se dispone del modelo Sercomm H500-S de la compañía Vodafone, y se accede mediante la dirección http://192.168.0.1 en el navegador de internet. Nada más entrar, aparece una visión general de los dispositivos que están conectados por wifi y por cable de red. En esta sección de red es donde se mira la IP de mi sistema: 192.168.0.168.

Esta dirección IP es la que se necesita para conectarse mediante terminal (sólo por comandos, sin interfaz gráfica) a mi sistema domótico. Para ello se utiliza el programa con licencia libre *Putty*, que es un cliente multiplataforma para conexiones remotas.

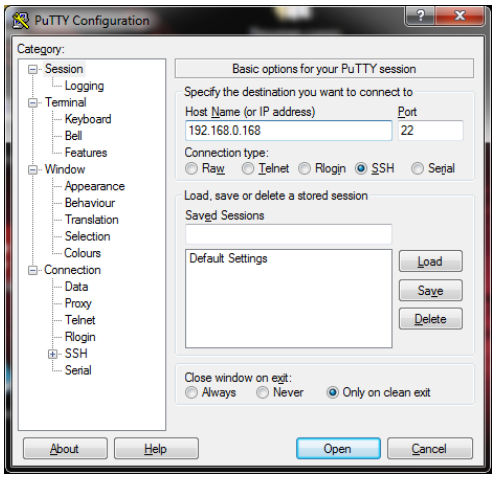


Figura 5-1. Conexión SSH mediante Putty.

En este caso se ejecuta en Windows y al iniciar el programa se ingresa la dirección IP de la Raspberry Pi y el puerto TCP 22. Se marca el tipo de conexión como SSH y se pulsa *Abrir,* donde aparece una ventana de terminal en el que pide las credenciales para el acceso al sistema. Por defecto, el nombre de usuario es *pi* y la contraseña *frambuesa*.

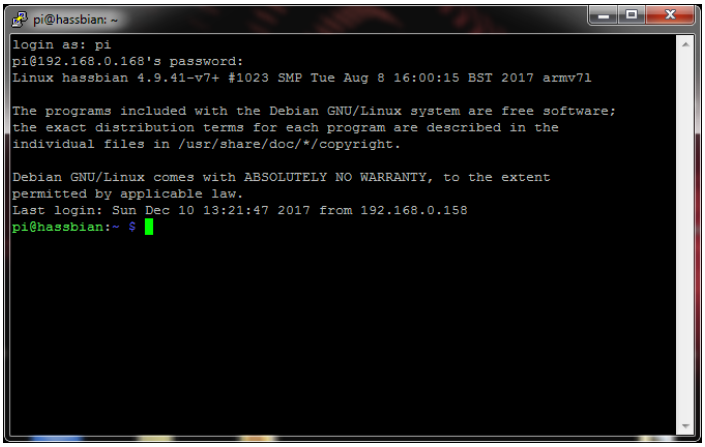


Figura 5-2. Terminal de comandos.

Para poder cambiarla por una de nueva elección se teclea el comando ***sudo passwd pi***. Pedirá que se ingrese una nueva contraseña de UNIX, teniendo en cuenta que distingue entre mayúsculas y minúsculas. Esta es la contraseña de acceso Root al sistema por lo que es muy importante cambiarla para evitar que alguien externo se pueda conectar y realizar cualquier cambio malintencionado.

Ahora se entrará en la pantalla de configuración con el comando ***sudo raspi-config***. Para realizar el desplazamiento por los menús se usan las flechas de arriba y abajo y para entrar en éstos la tecla intro. Dentro del menú 4 *Opciones de localización*, se cambiará tanto la zona horaria (submenú I2), eligiendo primero *Europa* y luego *Madrid*, como el idioma (submenú I1), marcando la opción *es_ES.UTF-8* mediante la tecla espaciadora.

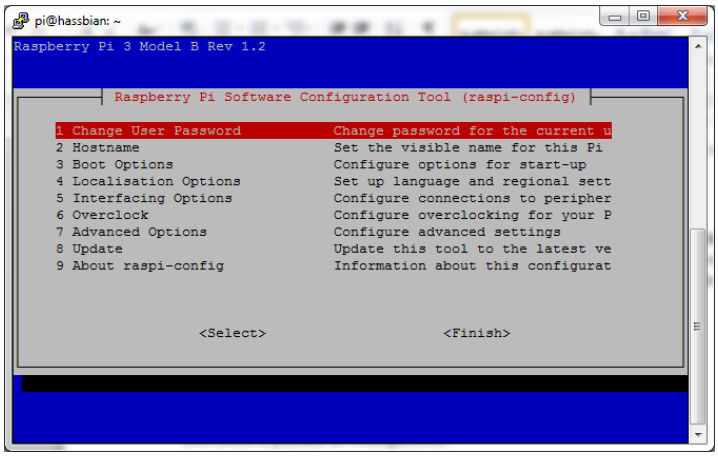


Figura 5-3. Menú de configuración Raspberry.

5.3. Conexión Wifi y asignación de IP local estática

Para no tener siempre que realizar la conexión a través de cable Ethernet y poder poner el dispositivo en cualquier lugar de casa, se va a habilitar el acceso Wifi. En una Raspberry Pi 3 ya viene incorporado por lo que no es necesario un USB Wifi externo. Se escribirá en la ventana del terminal el comando ***sudo nano /etc/wpa_supplicant/wpa_supplicant.conf***.

Y en la parte inferior del archivo se añadirán los datos de nuestro Wifi:

```
network={
 ssid="vodafone3D60"
 psk="0123456789"
        }
```

Se guarda todo mediante la secuencia de teclas: *Control + O*; *Enter*; *Control + X*, y luego se reinicia con ***sudo reboot***. Desconectamos el cable de red y la Pi se conectará a través de Wifi. El único problema es que ahora se tendrá que volver a buscar la nueva dirección IP ya que lo normal es que haya cambiado. La solución consistirá en asignar una IP local fija.

Esta configuración se necesita para acceder al servidor desde el exterior y poder redireccionar las peticiones que llegan al router. Para ello se necesita obtener al menos tres direcciones: la IP estática que se quiera asignar en la Raspberry que debe estar en la misma subred, la dirección del router y la dirección (o direcciones) de los DNS.

Se escribe ***ifconfig*** en el terminal para que muestre distinta información de la red, siendo la importante la que viene en el apartado *wlan0*. Ahí se puede ver la IP actual, en este caso 192.168.0.168, que se usará como IP estática para la Raspberry. También se puede ver la máscara de subred (255.255.255.0). Estos dos datos indican que la subred es 192.168.0.X, donde X será un valor entre 0 y 255. Es decir, todos los equipos de la red doméstica tendrán una IP local con el formato 192.168.0.X.

Ahora sólo queda averiguar la IP del router y para ello se escribe ***route –ne***. Aquí hay que fijarse en la puerta de enlace (*Gateway*) que esté en la fila donde *Destination* es 0.0.0.0, en este caso la dirección es 192.168.0.1. Básicamente lo que dice esa línea es que en la red local todo el tráfico que tenga como destino internet (0.0.0.0) se redirecciona al dispositivo con IP 192.168.0.1, es decir, nuestro router.

```
pi@hassbian:~ $ route -ne
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.0.1     0.0.0.0         UG        0 0          0 wlan0
192.168.0.0     0.0.0.0         255.255.255.0   U         0 0          0 wlan0
```

Figura 5-4. IP del router.

Una vez ya se tienen estos dos datos apuntados, se puede configurar la IP estática. Para establecerla ase añade la información anterior al fichero */etc/dhcpcd.conf*, siendo necesario editarlo como root: ***sudo nano /etc/dhcpcd.conf***. Sólo es necesario añadir las siguientes líneas al final del fichero, sin modificar nada más:

 # static IP configuration

interface wlan0
static ip_address=192.168.0.168/24
static routers=192.168.0.1
static domain_name_servers=8.8.8.8 8.8.4.4

En la línea que comienza por *interface* se establece la interfaz de red que se va a configurar. En este caso, al ser por Wifi, será *wlan0* (si fuera por cable se pondría *eth0*). En la siguiente línea (*static ip_address*) se escribe la IP estática que se quiere establecer. El /24 del final no es necesario pero no está de más ponerlo si la máscara de subred es 255.255.255.0. En *static routers* se escribe la IP del router y en *static domain_name_servers* la IP o IPs separadas por un espacio de los servidores DNS que se quiera utilizar, en este caso proporcionados por Google. Ya sólo queda guardar el fichero, reiniciar y comprobar que nuestra IP es la que se quería.

El router normalmente tiene habilitado el DHCP por lo que asignará automáticamente direcciones IP a los distintos dispositivos que se conecten a él. Por tanto, interesa que la IP estática no se asigne a otro dispositivo para evitar un conflicto de IPs y que deje sin conexión a Internet a la Raspberry. Esto se soluciona entrando en la configuración del router, apartado configuración LAN y DHCP estático - Red local. Ahí, se añade el dispositivo con la IP que se quiera y su dirección MAC. Se aplican los cambios y ya estaría todo configurado según las necesidades.

Figura 5-5. Añadir IP estática en el router.

Es importante asignar una IP fija a todos los dispositivos que se integren en el sistema domótico, ya que si no se hace no servirán las configuraciones que se hagan en Home Assistant y cuando el router les asigne otras IPs nos aparecerán como no disponibles.

5.4. Instalación y configuración de scripts

5.4.1. Samba

Permite el acceso desde el PC (Windows) a la Raspberry (Hassbian) que se encuentran en la misma red, y así se puede editar y compartir los archivos de configuración más fácilmente. Se escribirá el comando: ***sudo hassbian-config install samba***. Una vez instalado, se verá como una unidad de red a la que se puede acceder

como cualquier otra unidad de disco.

5.4.2. Mosquitto (MQTT)

Es un protocolo de comunicaciones rápido, seguro y altamente versátil para la comunicación máquina a máquina. Está orientado a la comunicación de dispositivos IOT, debido a que requiere muy poco ancho de banda, tiene un consumo muy bajo y precisa de muy pocos recursos para su funcionamiento.

Dentro de Home Assistant funcionará como un servidor MQTT, o también llamado *broker*, que aceptará los mensajes de cada dispositivo y los difundirá entre todos los demás, operando por defecto en el puerto 1883. Éste se encuentra instalado en Hassbian pero no activado, por lo que habrá que escribir el siguiente código: ***sudo hassbian-config install mosquitto***.

Al instalarlo pedirá que se ingrese un nombre de usuario y contraseña para acceder al servidor. Estos datos son los que todos los dispositivos MQTT conectados deberán usar si se conectan a su servidor. Para probar la conexión MQTT se puede utilizar el siguiente código, donde *micasa* es el usuario y *micasa1234* es la contraseña MQTT: ***mosquitto_sub -d -u micasa -P micasa1234 -t test/#.***

Para incorporar MQTT en HA, se introducirá el siguiente código en el archivo *configuration.yaml*:

```
mqtt:
  broker: IP Address
  port: 1883
  client_id: home-assistant-1
  keepalive: 60
  username: micasamqtt
  password: micasa1234
```

5.4.3. DuckDNS y Let´s Encrypt

DuckDNS es un sencillo y gratuito gestor de DNS dinámicas para poder acceder a través de internet a la Raspberry Pi. Para poder enviar y recibir datos se necesita disponer de la dirección IP asignada. Actualmente, al contratar el servicio de internet la dirección IP que se nos asigna el proveedor puede cambiar en cualquier momento en función de sus necesidades. Estas IPs dinámicas suponen un problema a la hora de intentar acceder a los datos de los ordenadores desde fuera de nuestra red interna, ya que si cambian no se sabrá dónde hay que conectarse.

Los servidores de DNS dinámicas permiten asociar la dirección IP a un nombre de internet del tipo *minombre.duckdns.org*. El servidor se encargará de actualizar la información, de tal forma que el nombre siempre esté apuntando a esa dirección IP. Los pasos a seguir para configurar DuckDNS serán los siguientes:

1. Se establece una IP estática. Esto se explicó en el apartado *Configurar IP local estática*.

2. Se crea una cuenta en DuckDNS. Se puede usar la cuenta de usuario de Gmail para loguearse. Una vez validado, aparecerá una pantalla con la información de la cuenta y el *token* que se ha generado. A continuación, se introduce el nombre que se quiera registrar, siempre que no esté cogido por otra persona, y se añade el dominio.

Figura 5-6. Registro en DuckDNS.

Con esto se crea la dirección *micasa.duckdns.org*, de tal forma que siempre que se acceda a ésta en realidad se estará accediendo al router de la conexión a internet. Ahora sólo queda el último paso, que es automatizar el cambio de dirección IP.

Para ello se irá al menú que se encuentra en la parte superior de la página web de DuckDNS y se selecciona la opción *install*. En ella, se elige el sistema operativo, en este caso *linux cron*, y el nombre de dominio que se registró anteriormente y para el que se quieren los códigos para su configuración. Se ejecutaran por orden y con el usuario *pi*:

ps -ef | grep cr[o]n
mkdir duckdns
cd duckdns
nano duck.sh

Dentro de este script se pega el siguiente texto, en el que se incluye el dominio y el *token* que se generó en el registro. Se guarda y se sale del editor:

echo url="https://www.duckdns.org/update?domains=micasa&token=bd7d7abc-ac3c-40a2-9026-c689ebr90c92&ip=" | curl -k -o ~/duckdns/duck.log -K -

Ahora se ejecutan estos comandos:

chmod 700 duck.sh
crontab -e

Y se pega el siguiente texto para que el script *duck.sh* se ejecute cada 5 minutos. Se guarda y se sale:

****/5 * * * * ~/duckdns/duck.sh >/dev/null 2>&1***

Para testear que funciona se ejecutan los siguientes comandos:

./duck.sh
cat duck.log

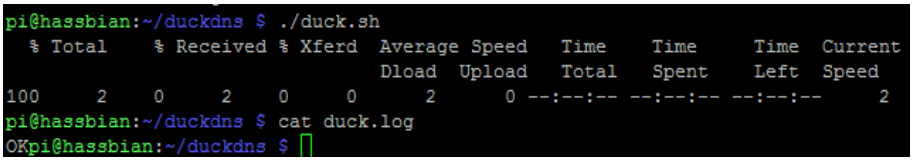


Figura 5-7. Comandos para comprobar el script duck.sh.

Por último, para iniciar el servicio *cron* sin tener que reiniciar el sistema se utilizará ***sudo service cron start***.

3. Redirección del puerto 8123. Para poder acceder desde fuera de casa a Home Assistant a través del puerto 8123 establecido para ello, hay que redireccionarlo en el router. Se entra de nuevo en la configuración avanzada del router y en el apartado *Internet* hay una opción que indica *Redirección de Puertos*. Se añade la asignación del puerto con los datos de la IP de la Raspberry, el protocolo y el puerto, quedando configurado como se indica en la siguiente imagen.

Figura 5-8. Redirección de puertos en el router.

4. Configuración del script *dehydrated*. Se hace la configuración con el usuario *homeassistant*. Primero, se descarga una copia del script *dehydrated* haciendo una clonación del repositorio:

git clone https://github.com/lukas2511/dehydrated.git

```
(homeassistant) homeassistant@hassbian:/home/pi $ cd
(homeassistant) homeassistant@hassbian:~ $ git clone https://github.com/lukas2511/dehydrated.git
Cloning into 'dehydrated'...
remote: Counting objects: 1774, done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 1774 (delta 74), reused 111 (delta 71), pack-reused 1653
Receiving objects: 100% (1774/1774), 561.83 KiB | 0 bytes/s, done.
Resolving deltas: 100% (1103/1103), done.
(homeassistant) homeassistant@hassbian:~ $
```

Figura 5-9. Clonación del script dehydrated.

Dentro de la carpeta dehydrated que se ha creado, se crea un archivo de texto con ***nano domains.txt*** en el que se incluirá la dirección *micasa.duckdns.org*. También se necesita otro archivo, ***nano config***, en el que se incluirá el siguiente texto:

 # Which challenge should be used? Currently http-01 and dns-01 are supported
CHALLENGETYPE="dns-01"

 # Script to execute the DNS challenge and run after cert generation
HOOK="${BASEDIR}/hook.sh"

Por último, se necesita un script que haga el cambio de DNS cuando se reinicia Home Assistant. Para ello, se crea un archivo *hook.sh* con ***nano hook.sh*** y se le pega el contenido indicado en el Anexo A.1, teniendo en cuenta que en *domain* y *token* se ponen los datos que se obtuvieron en *duckdns.org*.

Y se hace el script ejecutable: ***chmod 755 hook.sh***.

5. Se establece el usuario *homeassistant* como root. Se necesita incluir al usuario *homeassistant* como usuario root para que pueda ejecutar los comandos sin que nos pida contraseña y así poder continuar con la configuración sin problema: ***sudo adduser homeassistant sudo.***

Se agrega el usuario a sudoers: ***sudo visudo*** y se añade la línea ***homeassistant ALL=NOPASSWD: ALL***. Se reinicia el sistema.

6. Se genera el certificado SSL. Primero, se registra una nueva clave privada con *letsencrypt*:

./dehydrated --register --accept-terms

```
(homeassistant) homeassistant@hassbian:~/dehydrated $ ./dehydrated --register --
accept-terms
# INFO: Using main config file /home/homeassistant/dehydrated/config
+ Generating account key...
+ Registering account key with ACME server...
+ Done!
(homeassistant) homeassistant@hassbian:~/dehydrated $
```

Figura 5-10. Registro de clave en dehydrated.

Luego, se genera el certificado y se tendrá un certificado válido: ***./dehydrated -c***

Ahora, se tienen que dar permisos de lectura y escritura a las carpetas y archivos de dentro de la carpeta */dehydrated* que necesitan ser modificados por el usuario *homeassistant*. Estos códigos se ejecutan fuera de dicha carpeta por lo que primero se ejecuta el comando ***cd*** y luego:

chmod 755 /home/homeassistant/dehydrated/
chmod 755 /home/homeassistant/dehydrated/certs/
chmod 755 /home/homeassistant/dehydrated/certs/micasa.duckdns.org/
chmod 755 /home/homeassistant/dehydrated/certs/micasa.duckdns.org/cert-1519507441.csr

chmod 755 /home/homeassistant/dehydrated/certs/micasa.duckdns.org/cert-1519507441.pem
chmod 755 /home/homeassistant/dehydrated/certs/micasa.duckdns.org/chain-1519848781.pem
chmod 755 /home/homeassistant/dehydrated/certs/micasa.duckdns.org/fullchain-1519848781.pem
chmod 755 /home/homeassistant/dehydrated/certs/micasa.duckdns.org/privkey-1519848781.pem

7. Renovación del certificado automático. El certificado proporcionado por *Let´s Encrypt* expira a los 90 días, por lo que se necesita que se renueve automáticamente. Para ello, simplemente se hace una llamada al script *dehydrated* vía *cron* para que se ejecute, por ejemplo, el primer día de cada mes:

crontab –e

Y se pega el siguiente código:

0 1 1 * * /home/homeassistant/dehydrated/dehydrated -c

8. Configuración en Home Assistant. En el archivo *configuration.yaml* se añade el nuevo certificado en la sección *http*:

```
http:
  api_password: !secret http_password
  ssl_certificate: /home/homeassistant/dehydrated/certs/micasa.duckdns.org/fullchain.pem
  ssl_key: /home/homeassistant/dehydrated/certs/micasa.duckdns.org/privkey.pem
  base_url: micasa.duckdns.org:8123
```

Se guarda, se reinicia y ya se tendrá acceso al sistema domótico desde el exterior: *https://micasa.duckdns.org:8123*.

9. Problema y soluciones: IP NAT Loopback.

En el caso del router Vodafone, y en muchos de ellos, existe el problema de que no tiene implementada la función IP NAT Loopback, por lo que no permite acceder desde la red de casa a la dirección de DuckDNS, ya que ésta apunta a la IP pública y el router descarta la conexión.

Por tanto, algunas de las soluciones para poder realizar la conexión desde dentro de casa sería utilizando los datos móviles 4G en teléfonos o tabletas, conectando el router de la compañía a un segundo router que sí lo permita o conectando directamente a la dirección IP de la LAN *https://192.168.0.168:8123/*.

En algunos Sistemas Operativos como Windows o Linux, existe la opción de añadir una entrada al fichero *hosts* para que la dirección *https://micasa.duckdns.org:8123* apunte a la dirección IP local, siempre que el PC esté conectado a esta misma LAN. Este fichero se encuentra en *C:\Windows\System32\drivers\etc\hosts* o en */etc/hosts* si es Linux. Se abre con un editor de texto y simplemente se incluirá la siguiente línea:

192.168.0.168 micasa.duckdns.org

5.5. Integración de componentes

5.5.1. Sensores Xiaomi

Figura 5-11. Componentes Xiaomi en HA.

Dispositivos:

- Gateway.
- Sensores de Temperatura y Humedad.

- Sensor de Temperatura, Humedad y Presión.
- Sensor de movimiento.
- Pulsador Wireless.

Los dispositivos de Xiaomi Smart Home se integran muy bien en Home Assistant, y en cuanto a la relación calidad-precio es de lo mejor que se puede encontrar en el mercado. Se necesita disponer del *Gateway* ya que sin él no será posible coordinar los dispositivos ya que éste hace de puerta de enlace para todos. Para la comunicación utiliza el protocolo Zigbee y los datos pasan por los servidores de Xiaomi en China. Realmente, una vez configurado en HA cabe la posibilidad de capar el acceso a internet de la app para que no se conecte a esos servidores y que sólo lo haga de forma local a través de Home Assistant.

Para la configuración se va a utilizar un proyecto creado por *lazcad* y disponible en GitHub, que incluye soporte para todos estos dispositivos domóticos. En realidad, no sólo existe una forma de poder integrarlos y esto es gracias al trabajo de la comunidad de HA.

Lo primero de todo será instalar la app *Mi Home* de Xiaomi, que está disponible tanto en iOS como en Android, y configurar ahí cada uno de los componentes. Es muy sencillo, sólo hay que seguir las instrucciones indicadas en cada producto para que se conecten al Wifi de casa y que empiecen a funcionar.

Una vez se haga esto, se realizará la conexión al sistema Hassbian a través de Samba. Se crea una carpeta con el nombre *custom_components* dentro del directorio principal */home/homeassistant/.homeassistant/*. Dentro de esta carpeta se descomprimen los archivos que se han bajado del repositorio de GitHub. Básicamente se trata de varios archivos Python, con un código general y otro por cada dispositivo, que hacen que sean reconocidos por Home Assistant simplemente añadiendo unas líneas en el *configuration.yaml*.

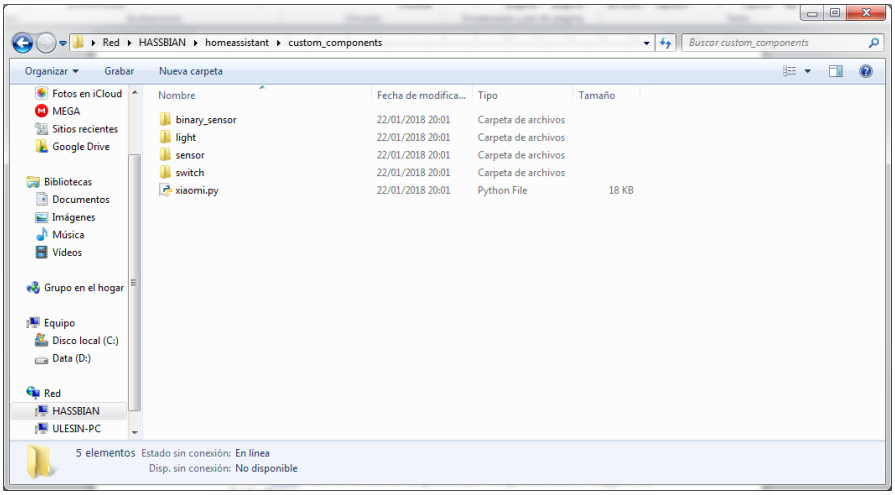


Figura 5-12. Componentes Xiaomi en HA.

Antes se necesita averiguar la *key* de nuestro *Gateway*. Para extraerla se habilitará el modo desarrollador dentro de la app. Se entra en el dispositivo *Gateway*, en el menú de los tres puntos de arriba a la derecha y luego en el primer menú de *Acerca de*/*About*. Una vez dentro, se toca con el dedo varias veces en la pantalla blanca hasta que aparezcan varios menús ocultos en idioma Chino. Se entra en el tercer menú empezando por arriba, se activa el protocolo de comunicación LAN (1) y se obtiene la *key* (2). En el cuarto menú también se puede obtener la MAC del *Gateway* donde pone *gw_mac*.

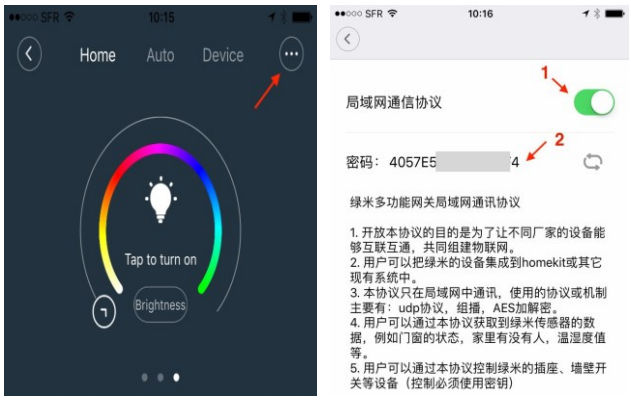


Figura 5-13. App Mi Home de Xiaomi.

Ahora sí, para habilitar el Xiaomi *Gateway* en Home Assistant se agrega el siguiente código en el archivo de configuración. La *MAC* del dispositivo se puede dejar en blanco, y en el *discovery_retry* ponemos 30, que son las veces que intentará realizar la conexión de Home Assistant al *Gateway* en caso de que falle.

```
xiaomi_aqara:
  discovery_retry: 30
  gateways:
    - mac:
      key: 4073195C71634160
```

Como ya se comentó anteriormente, una vez reiniciado el sistema aparecen los nuevos dispositivos como sensores dentro de un círculo y con nombres algo complejos. Aunque en función del gusto de cada persona se pueden dejar así, en este trabajo se personalizarán más adelante los nombres y se dispondrán en grupos ordenados.

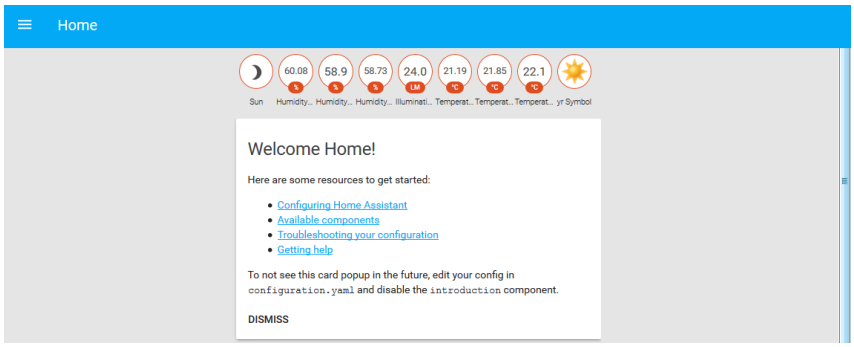


Figura 5-14. Interfaz inicial de HA.

5.5.2. Bombilla Yeelight

Figura 5-15. Componente Yeelight en HA.

Dispositivos:

- Bombilla Wifi RGB.

Se instalamos la app de la marca *Yeelight*, en iOS o Android, y se configura la bombilla. También se tiene que habilitar el modo LAN. Se entra en el menú de la bombilla, opciones de control LAN y se activa en el botón del siguiente menú.

Otra cosa que se necesita saber es la IP de la bombilla por lo que se volverá a ver en la página de configuración del router como se indicó en el apartado *Conexión mediante SSH y primeras configuraciones*. Se tiene que mirar en la sección Wifi del menú principal, donde estará la IP asignada, en este caso 192.168.0.157.

Por último, en el *configuration.yaml* se introducen las siguientes líneas en la sección *light*, donde *name* indica el nombre que se quiera mostrar en HA y *transition* el tiempo en milisegundos para las transiciones.

```
light:
- platform: yeelight
  devices:
    192.168.0.157:
      name: Luz entrada
      transition: 1000
```

Las opciones que hay disponibles en la bombilla aparecerán de la siguiente manera al pinchar sobre ella:

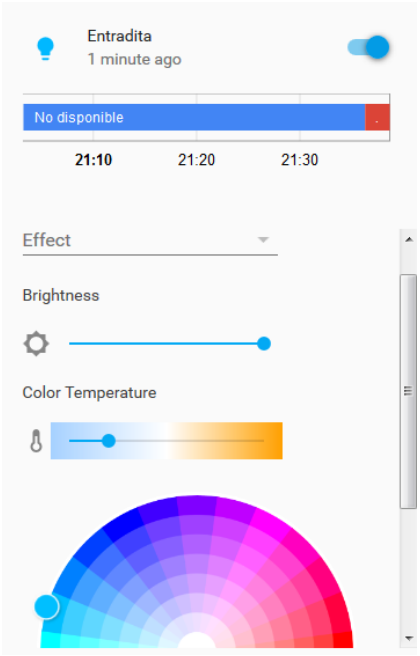


Figura 5-16. Menú de opciones Yeelight.

5.5.3. Dispositivos Broadlink

Figura 5-17. Componentes Broadlink en HA.

Dispositivos:

- Controlador universal IR.

- Enchufe Wifi SP3.
- Regleta Wifi MP1.

La función del emisor de infrarrojos será la de controlar la climatización centralizada, y la de los enchufes, será la de apagar y encender cualquier aparato que se quiera controlar.

Para integrarlo todo, primero se instalará la app *e-control* de Broadlink y se configurarán todos los dispositivos. Dentro del menú del dispositivo IR, se tiene que hacer que aprenda los comandos infrarrojos del mando del climatizador de casa de la marca Acson. Se irá pulsando cada botón del mando delante del dispositivo para que vaya aprendiendo la codificación de los infrarrojos, y en la app se creará un mando digital igual que si fuera el mando físico.

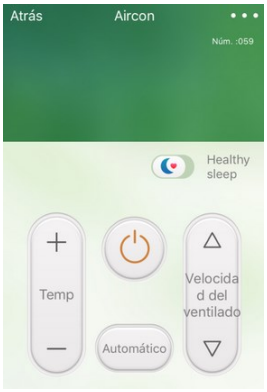


Figura 5-18. App e-Control de Broadlink.

Por otro lado, también se tiene que conocer la IP y la MAC que tiene cada uno de los productos entrando en la configuración del router, para poder integrarlos en Home Assistant. Los datos de la MAC vienen indicados como información cuando se añaden los dispositivos para fijar su IP en el router.

En el código de integración se añaden 30 segundos como duración máxima de conexión a los dispositivos (*timeout*), el modelo (*type*) de producto que corresponde cada uno y el nombre que se quiera poner.

Para la configuración del emisor IR se tendrá que añadir más líneas de configuración. Se pondrá la palabra *switches*, un identificador, el nombre y el comando del botón que corresponda. Para la primera configuración, se dejan en blanco todos los códigos que están entre comillas simples para luego ir completándolos.

Una vez se detectan por Home Assistant, se podrá utilizar el servicio que tiene éste y que ya se comentó en el apartado de introducción a HA, para ir sacando los códigos en Base64. Para ello, se selecciona *switch.broadlink_learn_command* de la lista de servicios disponibles y se presiona *call service*. A continuación, se tendrán 20 segundos para presionar un botón del mando virtual de la app para que el código se imprima por pantalla como una notificación persistente en la interfaz web. Es importante que a cada código descubierto se le añada al final dos iguales (==) para que funcione.

Para la climatización de casa, se va a utilizar el botón de encendido y apagado, que tienen el mismo código, el subir y bajar temperatura, el cambio de modo de funcionamiento y el cambio de velocidad del ventilador.

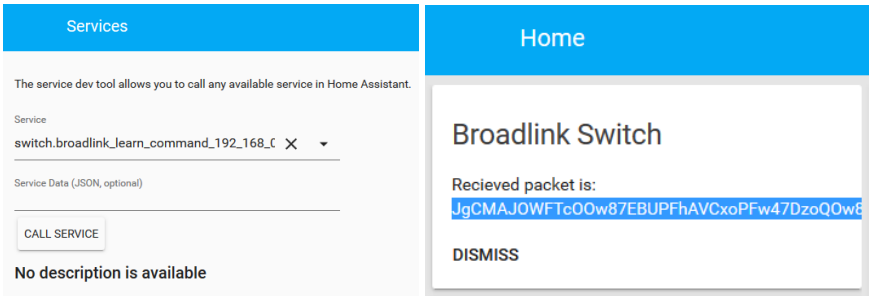


Figura 5-19. Códigos IR Broadlink en HA.

Una vez realizado todo esto, se meten los siguientes códigos en el archivo de configuración, dentro de la

sección *switch*:

```
switch:
  - platform: broadlink
    host: 192.168.0.166
    mac: '44:EA:34:F5:BC:C6'
    type: sp3
    timeout: 30
    friendly_name: 'Calientacamas Raúl'
  - platform: broadlink
    host: 192.168.0.156
    mac: '44:EA:34:E5:62:90'
    type: mp1
    timeout: 30
    friendly_name: 'Regleta'
    slots:
      slot_1: 'Cargador'
      slot_2: 'Lamparita'
      slot_3: 'Despertador'
      slot_4: 'Calientacamas'
  - platform: broadlink
    host: 192.168.0.159
    mac: '44:EA:34:88:D5:66'
    timeout: 30
    friendly_name: Climatización
    switches:
      aire:
        friendly_name: "Encendido"
        command_on:
'JgBQAAABJZURORAUEBQTERAUEBQQFBAUEBQQOg85ETgROBI3EzYTNhMREzYQFBAUEDk
QFBAUEBUPOREVDjoPORMREzYQORI3EwAGbAABJUwQAA0FAAAAAAAAAAA=='
        command_off:
'JgBQAAABJZURORAUEBQTERAUEBQQFBAUEBQQOg85ETgROBI3EzYTNhMREzYQFBAUEDk
QFBAUEBUPOREVDjoPORMREzYQORI3EwAGbAABJUwQAA0FAAAAAAAAAAA=='
…
```

El resto del código que falta para configurar los otros cuatro botones se indica en el Anexo A.2.

5.5.4. TV Samsung

Figura 5-20. Componente Samsung TV en HA.

Dispositivos:

- Smart TV modelo UE40D6500.

Home Assistant tiene muchas marcas y modelos de TVs para ser integradas de forma muy sencilla. Simplemente tiene que estar la TV conectada a internet de casa, ya sea por Wifi o por cable, y saber la IP que

tiene, como ya se indicó anteriormente. La primera vez que la televisión se enciende, hay que aceptar un mensaje para permitir la comunicación entre Home Assistant y la TV. Una vez realizado esto, se pondrá el siguiente código en el *configuration.yaml*:

media_player:
- platform: samsungtv
host: 192.168.0.155

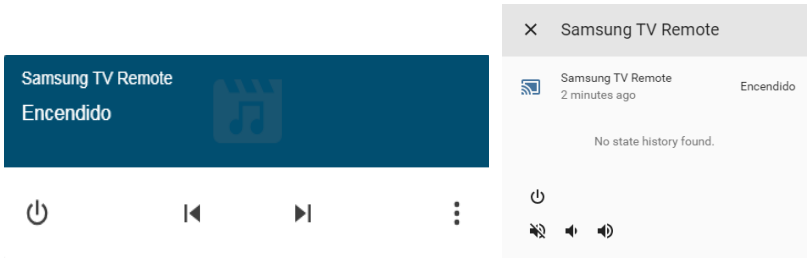


Figura 5-21. Menú de opciones Samsung TV.

5.5.5. Robot de limpieza Xiaomi

Figura 5-22. Componente Mi Robot Vaccum en HA.

Dispositivos:

- Robot de limpieza v1.

Para poder controlar el *Xiaomi Mi Robot Vacuum* (versión 1) se necesita instalar la librería *python-miio*. Ésta es la encargada de interactuar con los dispositivos que usan el protocolo binario encriptado *Mi Home de Xiaomi*, basado en UDP (*User Datagram Protocol*) y como designación de puerto el 54321.

Para que funcione correctamente se tienen que instalar otras librerías adicionales como son *libffi-dev*, *libssl-dev* y tener actualizada *setuptools*. Para ello se ejecutan los siguientes códigos mientras me encuentro en el usuario *pi*:

sudo apt-get install libffi-dev
sudo apt-get install libssl-dev
pip3 install -U setuptools

Después de esto ya se puede instalar la librería *python-miio*, se detiene primero el servicio de Home Assistant y luego se entra con el usuario *homeassistant* para realizar la instalación:

sudo systemctl stop home-assistant@homeassistant.service
sudo su -s /bin/bash homeassistant
source /srv/homeassistant/bin/activate
pip3 install python-miio / python-mirobo
exit
sudo systemctl start home-assistant@homeassistant.service

Es importante mencionar uno de los problemas que se tuvo en este punto del trabajo para lograr que funcionara esta librería, ya que daba errores al iniciar HA. En los foros oficiales se encontró el motivo, y era la versión de otra librería llamada *construct* de la que necesita que sea compatible. La versión que es compatible

hasta el momento de instalarla era la 2.9.41, por lo que se tenía que hacer una instalación manual de la versión concreta que se necesite y no utilizar la última versión disponible (estando con el usuario *homeassistant*):

pip3 install construct==2.8.22

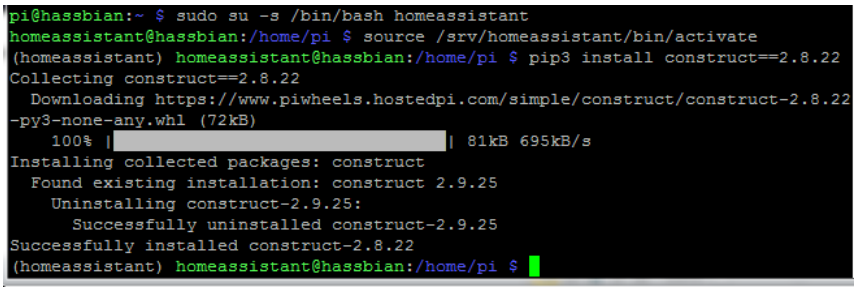


Figura 5-23. Instalación librería construct.

La librería *python-miio* está en constante actualización en el repositorio de GitHub del creador, puesto que se van descubriendo incompatibilidades con las librerías mencionadas o con las nuevas versiones de Home Assistant. Para ir actualizando la librería se utilizará el siguiente código, que entre otras cosas también te instala la versión correcta de *construct*:

pip3 install -U https://github.com/rytilahti/python-miio/archive/master.zip

Por tanto, lo más rápido para solucionar cualquier problema que surja es entrando en el foro oficial y siguiendo las instrucciones que se comentan, aunque otra opción es esperar un tiempo a que lo actualicen en el repositorio e instalarlo desde ahí.

Para el código de configuración de Home Assistant, se necesitará saber la IP asignada y un *token* de 32 caracteres hexadecimales que es único para cada robot de limpieza. Para sacarlo se utilizará un teléfono o una tableta con sistema operativo Android. Aunque hay alguna opción más para obtener el *token*, la más sencilla y la que mejor funciona es la que se comenta a continuación.

La aplicación *Mi Home* ya no almacena el *token* dentro de su base de datos local, ya que desde la versión 5.0.31 y superiores lo hace en los servidores de Xiaomi en China. Debido a esto, actualmente la única solución conocida es instalar una versión degradada mediante una apk. Esta versión concreta se puede descargar de cualquiera de las páginas que salen en Google al buscar *apk mi home 5.0.31*.

Otra de las cosas que se hará será tener activada la depuración por USB en el sistema Android para cuando se conecte al ordenador. Los pasos para realizar esto son los siguientes: se entra en Menú, Ajustes, Información del dispositivo (teléfono o tableta), y se pulsa varias veces hasta que aparezca un aviso que indica que las opciones de desarrollo están activas. Ahora, al volver al menú de Ajustes aparecerá otro apartado llamado Opciones de desarrollo. Cuando se entra en este apartado, se tendrá que activar la opción Depuración USB.

Una vez se instala la app y se configura el robot, quedará descargar la herramienta llamada *MiToolkit.1.6*. En este repositorio se descarga el .zip y se descomprime en el ordenador para su ejecución.

Se conecta el dispositivo Android al PC mediante cable y se pulsa *Extract Token* de la herramienta. Se abrirá otra pantalla y de nuevo se pulsa *Extract Token* sin establecer ninguna contraseña. Ahora, en el Android saldrá una pantalla de confirmación para realizar una *copia de seguridad de mis datos*, es decir, de la app *Mi Home*. Cuando termina el proceso, confirma que se ha realizado correctamente y que se va a proceder a la extracción de los datos. Pasados unos segundos mostrará un listado de todos los dispositivos que se tengan configurados en la app, y se buscará la línea que pone *rockrobo.vacuum.v1* como se puede ver en la siguiente imagen.

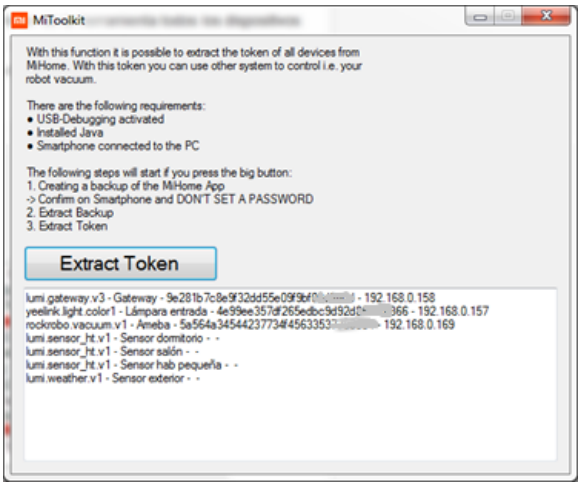


Figura 5-24. Herramienta para extraer token de Mi Robot.

Ejecutando el siguiente código en el terminal con el usuario *homeassistant*, se puede comprobar si todo ha salido bien y si se realiza la conexión correctamente con el robot de limpieza:

mirobo --ip 192.168.0.169 --token 5a564a3454423773 4f45633532888900 -d

Para introducir el *token* en la configuración de HA:

```
vacuum:
  - platform: xiaomi_miio
    host: 192.168.0.169
    token: 5a564a3454423773 4f45633532888900
    name: Ameba
```

Las opciones e información que estarán disponibles serán las siguientes:

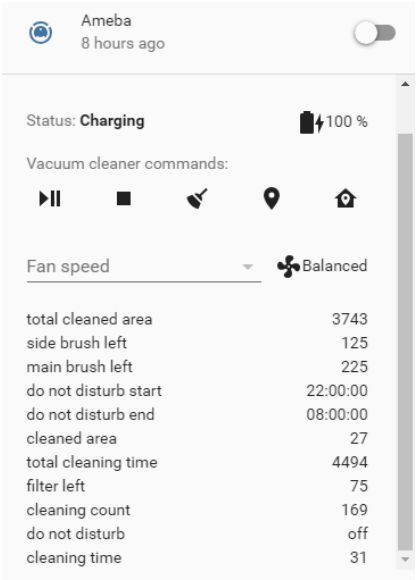


Figura 5-25. Menú de opciones Robot Xiaomi.

5.5.6. SAI Riello

Figura 5-26. SAI Riello.

Dispositivos:

- SAI Riello iDialog 800VA.
- Conector IEC hembra a SCHUKO.
- Regletas (2) de 4 tomas con conexión C14 IEC320.
- Cable USB tipo A-B.

Este dispositivo no tiene componente específico aunque se configura añadiendo la herramienta NUT (Network UPS Tools).

Un sistema de alimentación ininterrumpida (SAI), también conocido en inglés como *uninterruptible power supply* (UPS), realiza la función de asegurar el suministro eléctrico a los dispositivos que se encuentren conectados en caso de apagón. Tiene una respuesta muy rápida (de 2 a 6 ms), en el momento que detecta la caída de tensión pasan a funcionar las baterías del producto para sustituir la alimentación de la red. Y en cuanto vuelve la luz a casa, deja de alimentarse por el SAI y vuelve a hacerlo por la red. En caso de que el apagón dure más que la capacidad que tiene éste, activará la función de apagado controlado del sistema cuando le queda poco tiempo de alimentación.

Todo esto es importantísimo para que no haya pérdida de datos en el sistema domótico, y que no se corrompa la tarjeta microSD, ya que son muy sensibles cuando se apagan bruscamente. También hace que se pueda seguir controlando la casa en caso de que se esté fuera, ya que además de la Raspberry Pi 3 se conectará el router para no perder la conexión a internet, el *Gateway* de Xiaomi para no perder la información de los sensores, y la cámara Foscam para poder seguir viendo las imágenes de casa.

Este SAI está catalogado dentro de los interactivos, ya que no sólo tiene la funcionalidad de actuar cuando hay un fallo de tensión, sino que también incorpora un filtro para actuar cuando hay caídas o subidas de tensión, cuando existe ruido eléctrico o electromagnético e incluso cuando hay inestabilidad en la frecuencia.

Dimensionamiento de un SAI: éstos se clasifican en función de la Potencia aparente *S* (VA), y se relaciona con la Potencia activa *P* (W) mediante el factor de potencia (FP). Los más básicos tienen un FA de 0,6, es decir, son menos eficientes, por lo que tendrán mayor consumo propio y menor autonomía. Para saber el equivalente de la potencia en vatios se hará con la fórmula:

$S = P / FP \rightarrow P = S \times FP = 800 \times 0{,}6 = 480$ W

Ahora se calcula el consumo de los dispositivos que se van a conectar. Lo normal es realizar el cálculo sobre el consumo medio, ya que el máximo sólo lo alcanzan en momentos puntuales y dependiendo del uso que se le dé. En este caso, los valores son un poco por encima de la media para dar un pequeño margen de error.

Tabla 5–1. Consumo total dispositivos conectados al SAI

Dispositivo utilizado	Potencia
Router	20 W
Raspberry Pi 3	1,5 W
Gateway Xiaomi	40 W
Cámara Foscam	5 W
Total	66,5 W

Según los datos técnicos que se aportan en el manual del SAI Riello para la carga calculada, daría una autonomía de aproximadamente 45 minutos.

Una vez se enchufa el SAI a la red eléctrica y en la regleta se enchufan los dispositivos que se han comentado, se tendrá que conectar el cable USB tipo A-B del SAI a la Raspberry para que se puedan comunicar entre ellas mediante el software que se le va a instalar a continuación (usuario *pi*):

sudo apt-get install nut

Ahora se modificarán los siguientes ficheros para realizar la configuración:

- **nut.conf**: establece la configuración general de la herramienta NUT y tipo de sistema del SAI (independiente, servidor en red o cliente), además de los parámetros generales de upsd, upsmon y del apagado.
- **ups.conf**: define los SAIs que queremos controlar.
- **upsd.conf**: define el puerto, la dirección IP en la que escuchará el *daemon*, certificados a usar, etc.
- **upsd.users**: usuarios y grupos con acceso a la gestión de los SAIs
- **upsmon.conf**: configuración de monitorización para cada uno de los SAIs del sistema.

Se entra en el directorio con ***cd /etc/nut/*** y se van a realizar las siguientes modificaciones en cada uno de los 5 archivos:

sudo nano nut.conf

Se le indica que es un sistema independiente ya que es un único SAI:

MODE=standalone

sudo nano ups.conf

Se especifica el driver que le corresponde (se puede mirar en la página de Network UPS Tools), el puerto físico y una descripción sencilla. La última línea del fichero que indica *maxretry = 3* se ha tenido que quitar con una # ya que daba un error.

```
[riello]
    driver = riello_usb
    port = auto
    desc = "Riello IDG 800"
```

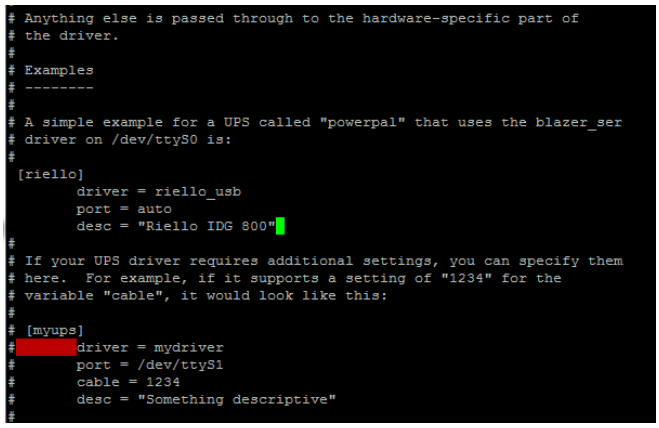


Figura 5-27. Contenido archivo ups.conf.

sudo nano upsd.conf

Aquí se configura el demonio *upsd*. Como no se van a permiten conexiones externas ni se va a usar cifrado, bastaría con añadir la siguiente línea, donde 3493 es el puerto de conexión estándar: *LISTEN 127.0.0.1 3493*

sudo nano upsd.users

En el siguiente archivo se define un usuario *admin*, que se usará para monitorizar el SAI, con permisos para ejecutar todos los comandos:

[admin]
password = admin1234
actions = SET
instcmds = ALL

sudo nano upsmon.conf

En este archivo hay bastantes opciones para configurar aunque lo único que hay que añadir será la siguiente línea: *MONITOR riello@localhost 1 admin admin1234 master*

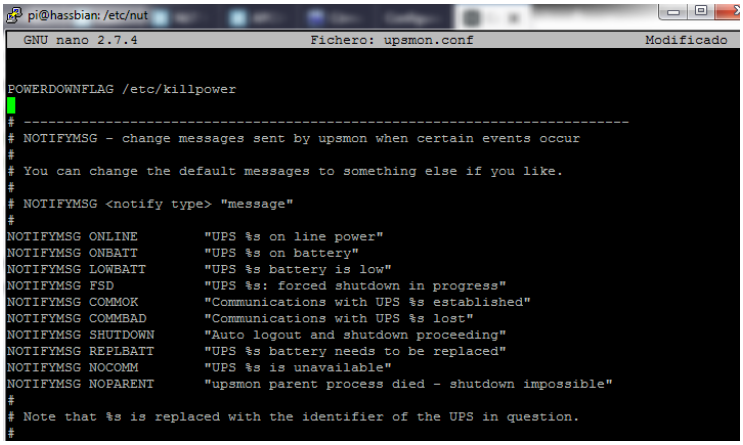


Figura 5-28. Contenido archivo upsmon.conf.

Se pueden ver todas las opciones que se han configurado en este archivo mediante el comando:

sudo awk 'NF' /etc/nut/upsmon.conf | grep -v "^#"

Por último, para evitar denegación de acceso por el usuario *nut* a la ruta donde se encuentran los archivos *.conf*, se tendrán que ejecutar los comandos:

sudo chown root:nut /etc/nut/*
sudo chmod 640 /etc/nut/*

Se reinicia el sistema y se comprueba que los servicios funcionan correctamente. Para ver estos parámetros, como la tensión de entrada y salida, temperatura, frecuencia, carga, tiempo de uso y otros datos, se ejecutará el siguiente comando:

upsc riello@localhost

En el caso de los SAI Riello se han detectado valores erróneos en algunos de los parámetros que muestran, como son la temperatura del SAI, la carga que tiene y el tiempo de funcionamiento. Los valores que no son correctos son los que indican un 255, en los dos primeros, y 3932100 en el último. Esto se puede deber a que la herramienta NUT no es capaz de sacar correctamente los datos, por lo que deberían hacer una actualización para corregir el problema.

La integración y configuración en Home Assistant se explicará en el siguiente apartado "otros sensores".

5.5.7. Cámara Foscam

Figura 5-29. Componente cámara Foscam en HA.

Dispositivos:

- Cámara IP Wireless HD 720p, Foscam modelo C1 v3.

En el caso de la integración de cámaras IP en Home Assistant, la marca *Foscam* es de las mejores. Después de probar también un modelo *Yi Dome 1080p*, la comparación es muy favorable a la primera. La *Yi* necesita primero una modificación del firmware y además cuando se integra en HA no actualiza bien la imagen.

La integración en Home Assistant, a nivel de programación, se hace mediante los comandos CGI proporcionados por *Foscam*, por lo que es una marca abierta para propósitos domóticos como éste.

Algunas de las características más importantes de esta cámara son: resolución HD 720p, ángulo de visión 115º, micrófono y altavoz incorporado, visión nocturna, ranura para tarjeta microSD, conectividad Wifi, Ethernet y consumo muy bajo de 5W máximo.

Igual que para otros componentes, primero se necesita instalar y configurar en un dispositivo móvil la app *Foscam*. Hay que registrarse aportando un usuario y contraseña que se necesitará para la configuración en HA. La cámara posee una pegatina con un código QR que se tendrá que escanear con la propia aplicación y añadir la red Wifi a la que se conectará. Una vez realizado, se incluirá el siguiente código en el *configuration.yaml*:

```
camera:
  - platform: foscam
    ip: 192.168.0.167
    username: micasa
    password: micasa1234
    name: Foscam
```

Las imágenes de vídeo en esta plataforma si se visualizan en la interfaz web se actualizan cada 10 segundos. Una vez que se entra en ella pulsando en la imagen, se amplía en grande y se pueden ver las imágenes en tiempo real con bastante fluidez.

Figura 5-30. Imagen cámara Foscam.

5.6. Notificaciones

Home Assistant dispone de un componente de notificación que hace posible enviar notificaciones a una amplia variedad de plataformas. Una vez cargada, se dispondrá de un servicio que se podrá utilizar para enviar mensajes al móvil cuando haya un evento importante en el sistema, o cuando se quiera recibir un aviso sobre alguno de los dispositivos o automatizaciones creadas. También tiene la posibilidad de enviar fotos o vídeos.

Dos de las plataformas más importantes y que se van a utilizar en este proyecto serán la del servicio de mensajería *Telegram*, disponible para todos los sistemas operativos de móviles, y la de iOS, que funciona mediante la app oficial de Home Assistant y que es específica para los usuarios de Apple. En el caso de Android no dispone de una app oficial.

5.6.1. Telegram

Para la configuración de Telegram, al ser una API pública pone a disposición de cualquier usuario la posibilidad de crear bots para recibir las notificaciones que se configuren en HA.

Para ello, una vez instalada la app, se abrirá un chat a *@BotFather* y se deberá escribir el siguiente comando: */newbot*. Pedirá un nombre para el bot que se quiera crear: *Ule*, y luego un usuario para ese bot con la condición que debe acabar en bot: *MiCasaBOT*. Una vez se hace esto, sale un mensaje con información sobre el enlace al bot creado y un *token*, un número muy largo de números y letras que se utilizará para la configuración en Home Assistant.

Ahora se creará el grupo de *Telegram* con los miembros se quieran incluir para recibir las notificaciones, se llamará *Equipo HA*. En él se incluirá el bot que se ha creado, y otro bot del sistema *Telegram* llamado *@myidbot* que servirá para conocer el *IDchat* que se utilizará en Home Assistant. Una vez dentro estos bots, se pondrá en el grupo el comando */getgroupid* para que se muestre la *IDchat*. Por último, se expulsa el *IDBot* ya que no será necesario tenerlo dentro.

Para la integración en HA se incluirá en el *configuration.yaml* el siguiente código, siendo el chat ID un número negativo:

```
telegram_bot:
  - platform: polling
    api_key: 517817600:ABFcWJVvUF8jvH9y90XZgwC7C3YIZMp6K8s
    allowed_chat_ids:
      - -211790591
```

Se añadirá también el código ***notify: !include notify.yaml*** y se crea en el directorio principal de homeassistant el archivo *notify.yaml* con el siguiente contenido:

```
- name: Equipo HA
  platform: telegram
  api_key: 517817600:ABFcWJVvUF8jvH9y90XZgwC7C3YIZMp6K8s
    chat_id: -211790591
```

Se guarda y se reinicia el sistema y ya se podrían utilizar las notificaciones en *Telegram* dentro de las automatizaciones que se quieran diseñar. También se puede realizar una comprobación para ver si todo funciona correctamente. Dentro de los servicios de HA, se pondría lo indicado en la imagen y se debería recibir una notificación en *Telegram* con el mensaje Prueba.

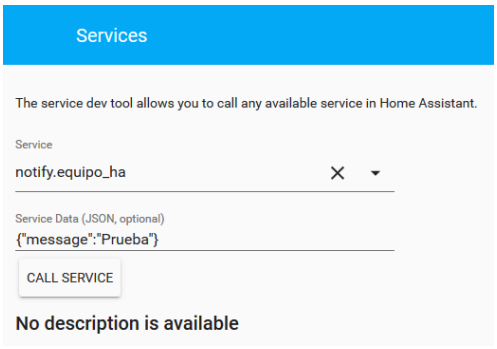


Figura 5-31. Servico para probar Telegram.

Un ejemplo de código dentro de la acción de una automatización sería:

```
…
action:
  - service: notify.equipo_ha
    data:
      message: 'La temperatura de la CPU ha llegado a los {{ states.sensor.cpu_temperatura.state }} ºC'
```

5.6.2. iOS app

Para configurar la app de iOS de Home Assistant y activar las notificaciones, una vez instalada en el iPhone, se añadirá al archivo *configuration.yaml* el código:

ios:

Se reinicia Home Assistant para que sea detectado y se configura la app introduciendo la dirección URL de DuckDNS y la contraseña de acceso que se puso. En los ajustes también se tendrá que habilitar las notificaciones. Se reinicia de nuevo y se verá que tanto el componente de iOS como la plataforma de notificaciones se encuentran cargadas.

Al igual que se hizo con la prueba de *Telegram*, se puede hacer con las notificaciones de iOS con solo cambiar el servicio a ***notify.ios_ulephone***, donde *ulephone* es el ID o nombre del dispositivo. Si no se recibe el mensaje, habrá que reiniciar tanto HA como la app y volver a probar.

Con esto ya se podrá utilizar tanto uno como otro para recibir en el móvil las notificaciones más importantes de Home Assistant que se configuren en las automatizaciones.

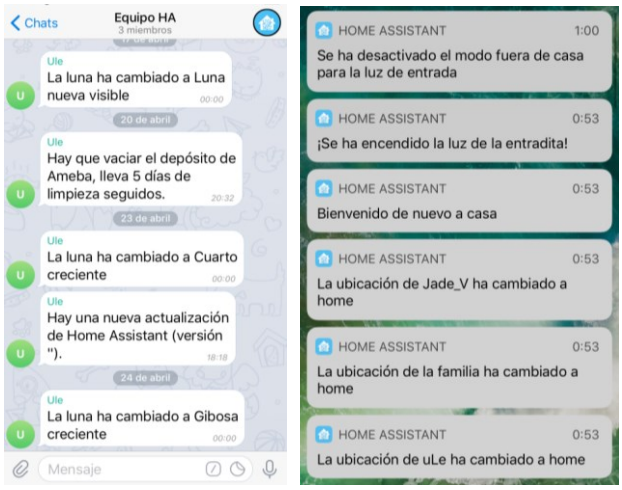


Figura 5-32. Notificaciones em Telegram y app iOS.

5.7. Localización de dispositivos móviles para seguimiento de ubicación

Conocido en inglés como *device tracker*, Home Assistant te da la opción de poder utilizar varias de las tecnologías existentes para el rastreo de dispositivos móviles, y en consecuencia, la ubicación de las personas. Las opciones que hay disponibles son varias: mediante GPS como pueden ser *Owntracks* o *iCloud*, o bien con tecnologías como *Nmap* para el escaneo de dispositivos dentro de la red Wifi local.

En este caso se va a usar una combinación de varios métodos. Se basará principalmente en una app gratuita llamada *Life360* de ubicación mediante GPS, a la vez que utiliza también *Owntracks* y *Nmap*. También se tiene que tener instalado MQTT ya que es necesario para que *Owntracks* funcione.

Para la configuración de todo ello se instalará la app *Life360*. Se creará un usuario y una contraseña cuando se realiza el registro y se tendrá que recordar para el siguiente paso. Se debe instalar y registrar en tantos móviles como personas se quieran *trackear*.

Para integrarlo en HA, se tiene que crear dentro del directorio principal */.homeassistant* una carpeta llamada */shell_commands*, donde se guardará un fichero llamado *360_mqtt_broker.sh* que se podrá descargar desde la siguiente página, y cuya configuración inicial se tendrá que modificar con nuestros datos de *Life360* y MQTT. Sólo hay que incluir los usuarios y contraseñas, el resto se deja como viene por defecto:

username360="micasa@gmail.com"
password360="micasa360"
mosquitto_pub="/usr/bin/mosquitto_pub"
mqtt_host="127.0.0.1"
mqtt_port="1883"
mqtt_user="micasamqtt"
mqtt_pass="micasa1234"
timeout=150

Hay más líneas de código en ese archivo para la configuración del acceso a Life360 y a MQTT pero no se va a pegar el código debido a su extensión. Se puede consultar descargando el archivo.

Ahora hay que dar permisos de ejecución al script anterior, para que se vaya conectando a *Life360* cada 150 segundos (2 minutos y medio), y se descargue la información al MQTT local.

sudo chmod +x 360_mqtt_broker.sh

También se tiene que hacer que el script se inicie al encenderse la Raspberry Pi, por lo que para configurar el autoarranque se ejecutará:

sudo nano /lib/systemd/system/life360.service

Y se pegará el siguiente código:

[Unit]
Description=Life360 service
After=multi-user.target

[Service]
Type=simple
ExecStart=/bin/bash /home/homeassistant/.homeassistant/shell_commands/360_mqtt_broker.sh
Restart=on-abort

[Install]
WantedBy=multi-user.target

Se tiene que tener en cuenta que en ExecStart se pondrá la ruta donde se encuentra el script. Se guarda el archivo y se sale. Ahora se habilita el servicio, se dan permisos al script y se inicia *Life360* con los siguientes códigos ejecutados en este orden:

sudo systemctl enable life360.service
sudo chmod 644 /lib/systemd/system/life360.service
sudo chmod +x /lib/systemd/system/life360.service
sudo systemctl daemon-reload
sudo systemctl start life360.service
sudo systemctl status life360.service

```
pi@hassbian:~ $ sudo systemctl status life360.service
● life360.service - Life360 service
   Loaded: loaded (/lib/systemd/system/life360.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2018-04-15 19:18:04 CEST; 12min ago
 Main PID: 6891 (bash)
   CGroup: /system.slice/life360.service
           ├─6891 /bin/bash /home/homeassistant/.homeassistant/shell_commands/360_mqtt_broker.sh
           └─7550 sleep 150
```

Figura 5-33. Estado del servicio Life360.

Para que funcione se debe tener instalada la librería *jq*: ***sudo apt-get install jq***. A continuación, se crea dentro de */.homeassistant* la carpeta */device_tracker*, y dentro de ésta el archivo *device_tracker.yaml* con el siguiente código:

- platform: nmap_tracker
hosts: 192.168.0.1/24
home_interval: 15
interval_seconds: 30
track_new_devices: False

- platform: owntracks
max_gps_accuracy: 100

Con esta configuración de parámetros, la precisión del GPS será de 100 metros. En cuanto al escaneo de IPs con *Nmap* se introducirá la dirección de la red a escanear, considerando el tiempo de 15 minutos de intervalo de no escaneo cuando detecta que se está en casa, y cuando se está fuera escanea cada 30 segundos.

Se debe comentar que *Nmap* no sirve en la mayoría de dispositivos móviles cuando se está en casa, sino sólo cuando entras en la red por primera vez. Estos dispositivos cuando entran en reposo desactivan el Wifi para ahorrar energía, por lo que considera que se está fuera de casa hasta que no se desbloquee. Por eso se combinan los dos métodos, para utilizar lo mejor de cada uno y que el error sea lo más pequeño posible.

En *configuration.yaml* se tiene que añadir la siguiente línea para que detecte el código del *device_tracker.yaml*:

device_tracker: !include_dir_merge_list device_tracker

Para que funcione todo correctamente se tiene que instalar *Nmap* mediante los comandos:

sudo apt-get update
sudo apt-get install nmap

Después de esto se reinicia el sistema. Al iniciar, se encontrará en la carpeta */.homeassistant* un archivo llamado *known_devices.yaml* en el que aparecen los dispositivos conectados a la red. Se tienen que buscar los móviles que están vinculados a *Life360*, y que aparecerán así:

ule_777ceb62783441f9974559b39c44bd7:
hide_if_away: false
icon:
mac:
name: uLe
picture: /local/sully.png
track: true
vendor:

jade_v_d13ba49fdb064b0bb0071ce3c11bd1bd:
hide_if_away: false
icon:
mac:
name: Jade V
picture: /local/bu.png
track: true
vendor:

En *picture*, se puede añadir una foto para que aparezca en el mapa de HA. Para ello, se pegan las imágenes en la carpeta */www* dentro del directorio principal de */.homeassistant*. Y para situar los móviles/personas sobre el mapa se necesita añadir al *configuration.yaml* la siguiente palabra:

map:

Una vez se reinicie el sistema, aparecerá la ubicación de los móviles que se hayan configurado, excepto si están en la ubicación Home que no aparecen. La configuración de casa se realiza al principio del archivo de configuración, donde se le dan los parámetros de longitud y latitud de la ubicación. También se puede configurar para que aparezca el estado de la batería, aunque esto se explicará en un apartado posterior.

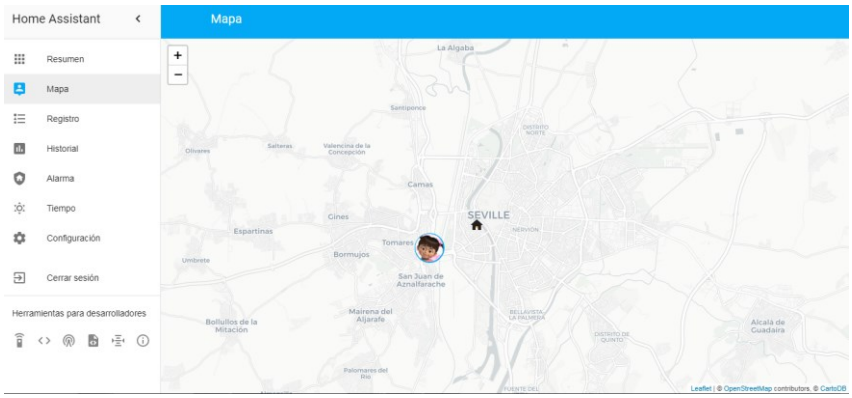


Figura 5-34. Mapa en Home Assistant.

5.8. Sistema de alarma

En Home Assistant se puede montar un sistema de alarma anti intrusos utilizando la bombilla *Yeelight*, el *Gateway* de Xiaomi, el sensor de movimiento, el pulsador y la cámara Foscam. La idea será la de activar la alarma con el pulsador cuando se vaya a salir de casa y no haya nadie dentro. La interfaz y funcionamiento de la alarma forman parte de un proyecto de *gazoscalvertos* y se puede encontrar en su página de GitHub. Lo primero de todo será agregar en el archivo *configuration.yaml* las siguientes líneas:

alarm_control_panel: !include alarm.yaml
panel_custom: !include panel_custom.yaml

Y crear los archivos indicados en el directorio principal de homeassistant. En *alarm.yaml* se indicará el código de desarme de la alarma (*code*), el tiempo que tarda en armar la alarma (*pending_time*), el tiempo que dura la alarma cuando se activa (*trigger_time*), y dentro de los sensores, los que se quieren utilizar y en función de cómo se quiera que actúen, en un apartado u otro.

En este diseño, se activará inmediatamente la alarma nada más abrir la puerta por lo que se incluye en *inmediate*. Existe la opción de que no se active inmediatamente y que deje unos segundos (*delayed*) después de que se active el sensor, para que se pueda desarmar con el pulsador si es alguien de la familia es el que entra. En este caso se desactivará desde fuera, antes de llegar a casa, accediendo a la interfaz web mediante la app. Además se podría incluir un código para cuando detecte a alquien de la familia en la zona home se desactive la alarma.

La alarma en modo noche no se va a utilizar pero sirve cuando se está durmiendo. En este caso, sería importante utilizar dos sensores, uno de movimiento que se incluiría en *homemodeignore,* ya que si alguien se levanta por la noche lo detectaría y saltaría la alarma, y otro sensor de apertura de puerta, que se incluiría en el *inmediate*.

Contenido del archivo *alarm.yaml*:

platform: bwalarm
name: house

code: 1234

pending_time: 25
trigger_time: 600

alarm: automation.alarm_triggered
warning: automation.alarm_warning

clock: True
perimeter_mode: False

COLOURS

warning_colour: 'orange'
pending_colour: 'orange'
disarmed_colour: '#03A9F4'
armed_home_colour: 'black'
armed_away_colour: 'black'
triggered_colour: 'red'

SENSOR GROUPS

immediate:

- binary_sensor.motion_sensor_158d000171c445

delayed:

homemodeignore:

override:

perimeter:

En *panel_custom.yaml* se indicará el siguiente código. Es importante que en *alarmid*, detrás del punto, se ponga el mismo nombre que se puso en el anterior archivo, en este caso house.

- name: alarm
sidebar_title: Alarma
sidebar_icon: mdi:security-home
config:
alarmid: alarm_control_panel.house

Ahora se creará una carpeta dentro de */www* que se llamará */lib* para dejar un par de archivos de dependencias JS. Para ello, se entra por SSH con nuestro usuario *homeassistant* y se pone el siguiente código:

cd /home/homeassistant/.homeassistant/www
mkdir lib
cd lib
wget http://domology.es/wp-content/uploads/2018/01/countdown360.js
wget http://domology.es/wp-content/uploads/2018/01/jquery-3.2.1.min.js

No se van a poner los códigos porque son bastante extensos, se pueden ver mediante cualquier editor de texto cuando se descargan. También se creará otra carpeta en */custom_components* llamada */alarm_control_panel* para meter un archivo Python *bwalarm.py* con el código para el funcionamiento principal de la alarma. Es muy extenso y no se va a poner el código ya que se puede descargar en la página de GitHub.

Con esto ya se tendría montado el sistema de alarma. Ahora faltaría el diseño de las automatizaciones para hacer funcionar las luces de la bombilla y los sonidos del Gateway. Las imágenes se verán accediendo a la cámara Foscam.

Figura 5-35. Sistema de alarma desactivada.

El código para las automatizaciones donde se introducen los diferentes estados de la alarma y su diseño, se tienen que introducir en el archivo *automations.yaml* y hacer referencia dentro del archivo de configuración poniendo ***automation: !include automations.yaml***. Los estados de la alarma son los indicados por Home Assistant y el código final quedaría de la siguiente manera:

```
##########################################
######### ESTADOS ALARMA ##########
##########################################

# Estado de la alarma cuando se está activando (tarda 25 segundos).
# Se introduce un sonido en el Gateway de cuenta atrás para saber que se está
# armando la alarma.

- id: alarm_arming_away
  alias: '[Alarm] Away Mode Arming'
  trigger:
    - platform: state
      entity_id: alarm_control_panel.house
      to: 'pending'
  action:
    - service: xiaomi_aqara.play_ringtone
      data:
        gw_mac: 38:6C:07:77:8A:C1
        ringtone_id: 3
        ringtone_vol: 20
```

```
  - service: notify.ios_ulephone
    data:
      message: "Activando alarma de casa... *Estado: pending"
```

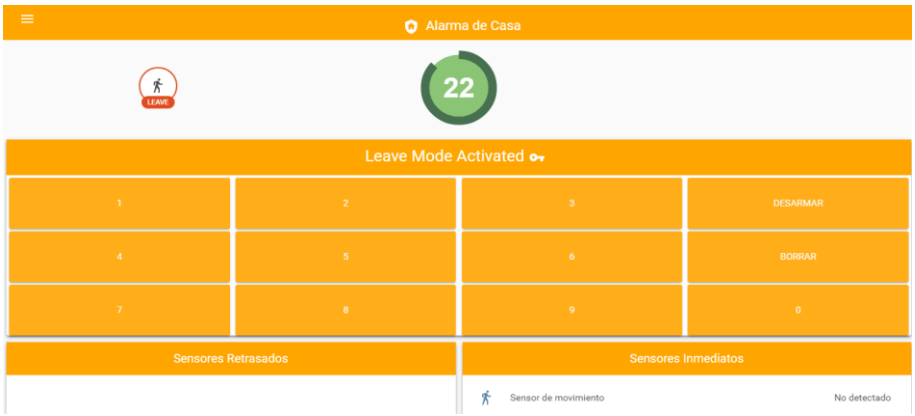


Figura 5-36. Activando sistema de alarma.

```
# Estado de la alarma cuando pasa a armada.
- id: alarm_armed_away
  alias: '[Alarm] Away Mode Armed'
  trigger:
    - platform: state
      entity_id: alarm_control_panel.house
      to: 'armed_away'
  action:
    - service: notify.ios_ulephone
      data:
        message: "¡Alarma de casa activada! *Estado: armed_away"
```

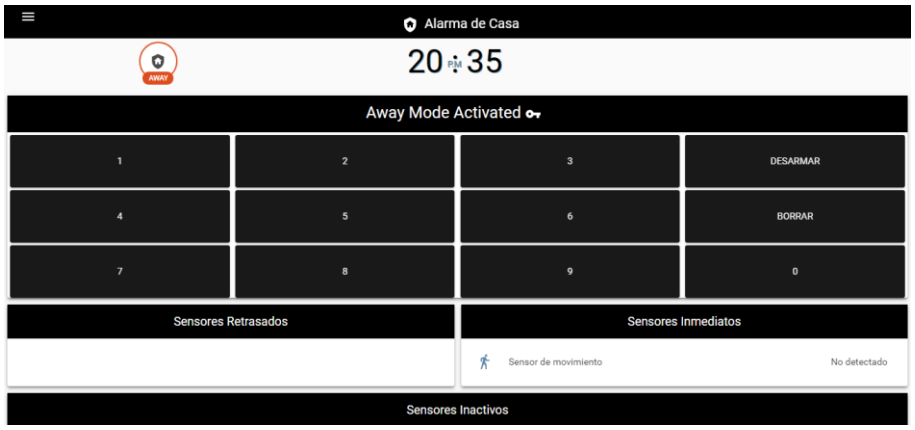


Figura 5-37. Sistema de alarma activada.

El resto del código con los demás estados de la alarma se indica en el Anexo A.3. Además se añade el código con la automatización de la sirena.

Por último, tanto para activar como desactivar la automatización de la alarma, se va a configurar el pulsador de Xiaomi para cuando se pulse de forma mantenida. Los estados tendrán que estar en desarmado o armado respectivamente. Cuando se aprieta para su desarmado se tiene que añadir el código de desarme para que funcione:

```
- alias: activar desactivar alarma
```

```
trigger:
  platform: event
  event_type: click
  event_data:
    event_id: binary_sensor.switch_158d0001b12b32
    click_type: long_click_press
action:
  - service_template: >
      {% if is_state('alarm_control_panel.house', 'disarmed') == false %}
        alarm_control_panel.alarm_disarm
      {% else %}
        alarm_control_panel.alarm_arm_away
      {% endif %}

    entity_id: alarm_control_panel.house
    data:
      code: 1234
```

5.9. Diseño y configuración de la interfaz de usuario

5.9.1. Sistema de climatización

Para el diseño de la climatización frío/calor de casa se necesitan varias configuraciones. La idea será simular tanto el mando físico como el *display* para comprobar el estado actual: si está apagado o encendido, la temperatura fijada, el modo de funcionamiento y la velocidad del ventilador.

Más arriba se realizó la integración del emisor de infrarrojos de Broadlink, por lo que ya se tienen los *switches* o interruptores que serán los botones del mando para manejar la climatización.

Figura 5-38. Mandos del sistema de climatización.

Se tienen que definir una lista de valores *input_select* para que se pueda seleccionar tanto el modo de funcionamiento como la velocidad del ventilador. También se define una lista de números *input_number* que será el rango de temperaturas posible, y el icono que se quiera que salga al lado del nombre de cada uno. Dentro de *configuration.yaml* se pondrá:

```
input_select:
  ac_operation_mode:
    name: Modo de funcionamiento
    options:
      - DRY
      - FAN
      - COOL
```

```
    - HEAT
   icon: mdi:settings
  ac_fan_mode:
   name: Modo del ventilador
   options:
    - LOW
    - MED
    - HIGH
    - AUTO
   icon: mdi:fan
input_number:
  ac_temperature:
   name: Temperatura
   min: 16
   max: 30
   step: 1
   icon: mdi:thermometer
```

Para el funcionamiento principal se incluirán los siguientes códigos dentro de *automation: !include automations.yaml*. El código se compone de un desencadenante que será cuando de pulsa el botón, de una condición que se debe cumplir que será que la climatización esté encendida, y de una acción que hará que se incremente, decremente o seleccione un valor. Esto último es posible gracias a los servicios que están disponibles en Home Assistant, y para el cambio de temperatura quedaría así:

```
# Temperatura del display disminuye
- alias: Temperatura display disminuye
  trigger:
  - platform: state
    entity_id: switch.bajar_temperatura
    from: 'off'
    to: 'on'
  condition:
  - condition: state
    entity_id: switch.encendido
    state: 'on'
  action:
  - service: input_number.decrement
    data:
      entity_id: input_number.ac_temperature
# Temperatura del display aumenta
- alias: Temperatura display aumenta
  trigger:
  - platform: state
    entity_id: switch.subir_temperatura
    from: 'off'
    to: 'on'
  condition:
  - condition: state
    entity_id: switch.encendido
    state: 'on'
  action:
  - service: input_number.increment
    data:
      entity_id: input_number.ac_temperature
```

El código para el cambio de velocidad del ventilador y el modo de funcionamiento se incluye en el Anexo A.4, siendo iguales que los de temperatura pero cambiando el tipo de servicio y la entidad.

Para simular el *display* se utilizará un *sensor template* para que cuando se pulse un botón del mando, cambie también el valor en la pantalla. Dentro de *sensor: !include sensores.yaml* se pondrá:

```
- platform: template
  sensors:
    ac_setting_mode:
      friendly_name: 'Current mode'
      value_template: '{{ states.input_select.ac_operation_mode.state }}'
    ac_fan_mode:
      friendly_name: 'Current fan'
      value_template: '{{ states.input_select.ac_fan_mode.state }}'
    ac_temp_display:
      friendly_name: 'Current Temp'
      value_template: '{{ states.input_number.ac_temperature.state | int }}'
      unit_of_measurement: 'ºC'
```

También se tiene que realizar unos *scripts*, que será lo que se refleje en el *display*, y donde al pulsar el botón correspondiente hará que se ponga a cero su valor. Se incluirá en *script: !include scripts.yaml*:

```
aire_modo:
  sequence:
    - service: switch.turn_on
      entity_id: switch.modo
    - service: switch.turn_off
      entity_id: switch.modo
```

Para la subida y bajada de temperatura y el ventilador será el código igual pero cambiando el nombre del switch correspondiente. Se incluyen los códigos en el Anexo A.5.

Para la personalización de los nombres e iconos que aparecerán en Home Assistant, se incluirá dentro de la primera línea *homeassistant:* del archivo de configuración la línea *customize: !include customize.yaml*, con el siguiente código:

```
# Nombre de los botones e iconos
switch.encendido:
  icon: mdi:power
  assumed_state: false
script.aire_bajar_temp:
  friendly_name: "Bajar Temperatura"
  icon: mdi:arrow-down-bold
script.aire_subir_temp:
  friendly_name: "Subir Temperatura"
  icon: mdi:arrow-up-bold
script.aire_ventilador:
  friendly_name: "Velocidad del Ventilador"
  icon: mdi:fan
script.aire_modo:
  friendly_name: "Modo de funcionamiento"
  icon: mdi:settings

# Objetos ocultos: no se necesitan mostrar los interruptores puesto que se utilizarán los scripts
# que se han diseñado para la apariencia del display. Se indica como hidden: true.

switch.bajar_temperatura:
  assumed_state: false
  hidden: true
switch.subir_temperatura:
  assumed_state: false
  hidden: true
switch.ventilador:
```

```
  assumed_state: false
  hidden: true
switch.modo:
  assumed_state: false
  hidden: true
```

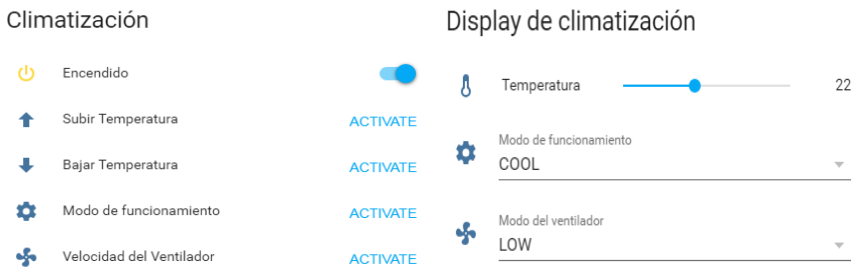


Figura 5-39. Apariencia de los mandos de climatización.

5.9.2. Robot de limpieza

Una vez se tiene integrado en Home Assistant, la idea será introducirle varias funcionalidades: programación semanal, contador de usos y control remoto para la limpieza del depósito.

• Programación semanal: se necesita hacer un código para que el robot se inicie a una hora determinada, tanto a diario como en fines de semana, según las necesidades. Primero se tienen que declarar todas las variables que se van a utilizar en el *configuration.yaml*, tanto números como booleanos:

```
input_number:
  # Hora para activar el encendido por programación
  hora_on_p1:
    name: Hora inicio
    icon: mdi:timer
    min: 00
    max: 23
    step: 1

  # Minutos para activar el encendido por programación
  minutos_on_p1:
    name: Minutos inicio
    icon: mdi:timer
    min: 00
    max: 59
    step: 1

input_boolean:
  # Activar/Desactivar el encendido por programación
  # Si solo está activado esto funcionará SOLO una vez
  estado_on_p1:
    name: Encendido Programa 1
    icon: mdi:alarm-check

  # Activar/Desactivar la condición de uso de Lunes a Viernes (incluidos).
  # Debe estar activada también "estado_on_p1".
  entresemana_p1:
    name: Entresemana
    icon: mdi:alarm-check

  # Activar/Desactivar la condición de uso de Sábado y Domingo.
  # Debe estar activada también "estado_off_p1".
  finde_p1:
```

```
    name: Fin de semana
    icon: mdi:alarm-check
```

Dentro de *sensores.yaml* se usará un *template* para incluir un *slider* o control deslizante para poder elegir el horario al que se quiera programar el robot.

```
############################
#   Sensor Programación 1   #
############################

## Resumen de hora:minutos para el encendido por programación

- platform: template
  sensors:
    inicio_p1:
      friendly_name: 'Hora de encendido'
      icon_template: mdi:timer
      value_template: '{{ (states.input_number.hora_on_p1.state)|int }}:{% if
states.input_number.minutos_on_p1.state|int < 10 %}0{% endif %}{{
(states.input_number.minutos_on_p1.state)|int }}'
```

Ahora se tiene que escribir el código principal del programador dentro de *automations.yaml* para que realice el funcionamiento indicado. Funciona como todas las automatizaciones: un desencadenante, que será el tiempo que se indique para que se inicie; varias condiciones, en función de si es fin de semana o entresemana y de si están activados los botones de encendido de la programación; y varias acciones, de puesta en marcha del robot y de comprobación de estados de las variables booleanas. Debido a su longitud se ha indicado el código en el Anexo A.6.

• Contador de usos: en este caso HA tiene un problema con la variable *counter* y es que no guarda su estado al reiniciar el sistema como sí lo hacen todos los *input*, por lo que siempre tomará el valor de 0 o del que se configure como valor inicial. Es por eso que se tiene que hacer una duplicación del contador mediante un *input_number* para que guarde el estado antes de reiniciar el sistema, y cuando se inicie de nuevo adquiera el valor que tenía antes. Por tanto, se empezará por declarar las variables que se necesitan:

```
# Variable contador con el nombre del robot "ameba"
counter:
  ameba:
    step: 1
    icon: mdi:counter

# Valores numéricos para el display del contador y la programación diaria.
input_number:
  contador:
    name: Contadorrr
    min: 0
    max: 10
    step: 1
```

En *automations.yaml* se incorpora el código de funcionamiento: cuando se use el robot (*on*) y cumpla la condición de que el script de control remoto que se usa para la limpieza se encuentre apagado durante 3 segundos, se incrementa el contador en 1 y se asigna el valor que tiene *counter* al *contador* que se ha creado, para ir almacenando su valor para cuando haya un reinicio del sistema.

La condición que se ha puesto es para que no cuente uno más al iniciar el control remoto de limpieza, ya que el robot se enciende y pasa a estado *on*. Con esto no se incrementa el contador porque el script estaría activo también:

```
- alias: Contador de usos
  trigger:
    - platform: state
      entity_id: vacuum.ameba
```

```
    to: 'on'
  condition:
    - condition: state
      entity_id: script.vacuum_service_mode
      state: 'off'
      for:
        seconds: 3
  action:
    - service: counter.increment
      entity_id: counter.ameba
    - service: input_number.set_value
      data_template:
        entity_id: input_number.contador
        value: "{{ states.counter.ameba.state }}"
```

En *scripts.yaml* se tiene que crear un código para asignar el valor del *contador* a *counter* cuando se vuelve a iniciar el sistema después de un reinicio. Tiene que ir comprobando si su valor es menor e incrementándolo hasta que lo iguale, y se utilizará para ello el servicio de incremento de contador (+1).

```
counter_inc:
  sequence:
  # Se incrementa una vez cuando entra
    - service: counter.increment
      entity_id: counter.ameba
    - service_template: >
        {% if states('counter.ameba')|default(0)|int < states('input_number.contador')|int %}
          script.turn_on
        {% else %}
          script.turn_off
        {% endif %}

      entity_id: script.counter_inc_loop

# Se necesita crear un bucle que vaya recorriendo los scripts
counter_inc_loop:
  sequence:
    - delay:
        milliseconds: 1
    - service: script.turn_on
      entity_id: script.counter_inc
```

Y ahora en *automations.yaml* se crea un código para llamar a esos scripts siempre que Home Assistant se inicie. Se pondrá la condición de que el *contador* no tenga valor cero para entrar en la comprobación, ya que por defecto *counter* tiene ese valor inicial:

```
- alias: 'inicio home assistant'
  trigger:
    platform: homeassistant
    event: start
  condition:
    condition: numeric_state
    entity_id: input_number.contador
    above: 0
  action:
    service: script.turn_on
    entity_id: script.counter_inc
```

• Control remoto para limpieza del depósito: se utilizarán los servicios que tiene Home Assistant para mover el robot de limpieza de Xiaomi mediante el control remoto. El objetivo de esto será sacar el robot de debajo del mueble para poder levantar la tapa y vaciar el depósito. A la vez que se pulsa este botón, el contador de usos se

reiniciará a 0 para empezar de nuevo a contar. Se introducirá el depósito limpio en el robot y se pulsará el botón de la *casita* que tiene éste en la parte superior para que vuelva a su base.

Se han puesto varios retrasos (*delays*) dentro de la ejecución del código y configurado el tiempo y la velocidad del robot para que se pare justo en un hueco que hay entre el mueble donde se encuentra y una cómoda que hay enfrente de éste:

```
vacuum_service_mode:
  alias: Control remoto
  sequence:
    - service: vacuum.xiaomi_remote_control_start
      entity_id: vacuum.ameba
    - delay: '00:00:08'
    - service: vacuum.xiaomi_remote_control_move
      entity_id: vacuum.ameba
      data:
        duration: 1800
        velocity: 0.2
        rotation: 0
    - service: counter.reset
      entity_id: counter.ameba
    - service: input_number.set_value
      data_template:
        entity_id: input_number.contador
        value: 0
    - delay: '00:00:02'
    - service: vacuum.xiaomi_remote_control_stop
```

También se incorporará un *sensor template* para que en la interfaz de Home Assistant indique si hay que vaciar el depósito del robot. Cuando llegue a 5 usos indicará *Sí* a la pregunta formulada y cambiará el icono a uno de alerta:

```
- platform: template
  sensors:
    limpieza_ameba:
      friendly_name: '¿Hay que vaciar el depósito?'
      value_template: >-
        {%- if (states.counter.ameba.state | int > 4) -%}
          Sí
        {%- else -%}
          No
        {%- endif -%}
      icon_template: >-
        {%- if (states.counter.ameba.state | int > 4) -%}
          mdi:comment-alert-outline
        {% else %}
          mdi:comment-check-outline
        {% endif %}
```

Y nos enviará una notificación al móvil cuando esto ocurra. En el archivo *automations.yaml*:

```
- alias: 'Notificación vaciar depósito'
  trigger:
    platform: numeric_state
    entity_id: counter.ameba
    above: 4
  action:
    - delay:
        minutes: 1
```

```
  - service: notify.ios_ulephone
    data:
      message: Hay que vaciar el depósito de Ameba, lleva {{ states.counter.ameba.state }} días de limpieza seguidos.
```

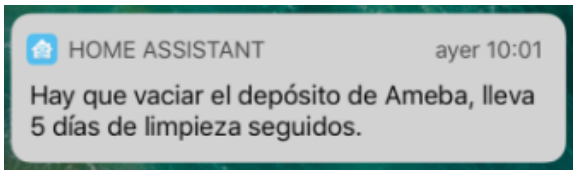


Figura 5-40. Notificación para limpiar depósito del robot.

En *customize.yaml* se personalizará el nombre del contador y el icono del control remoto:

```
counter.ameba:
  friendly_name: "Contador de usos"
script.vacuum_service_mode:
  icon: mdi:broom
```

Para terminar, es muy importante que se incorporen las siguientes líneas en el archivo de configuración. El *recorder* sirve para que HA guarde el estado de los dispositivos al reiniciar el sistema, como pueden ser luces, interruptores, sensores o variables como los *input* que se han utilizado para el contador y la programación:

```
recorder:
  purge_keep_days: 5
  include:
    domains:
      - sensor
      - light
      - switch
      - input_boolean
      - input_number
      - input_select
```

La apariencia del diseño implementado aparecerá en Home Assistant como se indica en la figura.

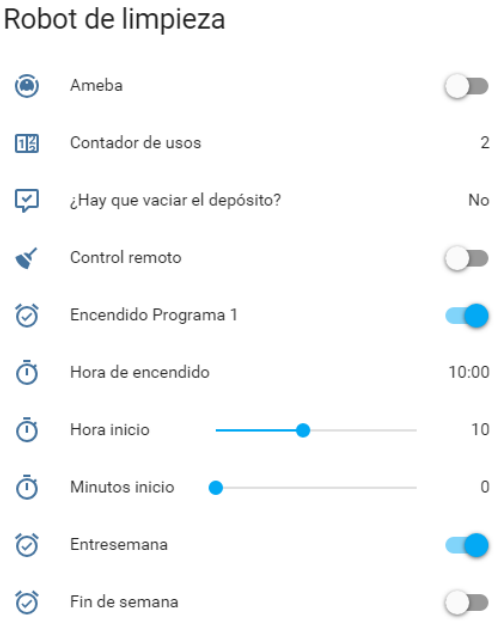


Figura 5-41. Diseño del robot en HA.

5.9.3. Sensores, luces y enchufes

Los dispositivos que se han integrado anteriormente en HA salen con nombres poco operativos como se pueden ver en el apartado *estados* de las herramientas para desarrolladores. Para personalizar estos nombres y cambiar los iconos que salen por defecto, se hará en el archivo *customize.yaml*:

```
# Luces
light.yeelight_rgb_7811dc6840bc:
  friendly_name: Luz entrada
light.gateway_light_286c07888ac2:
  friendly_name: Luz gateway

# Sensores de temperatura y humedad, y también uno de presión atmosférica
sensor.temperature_158d000159c165:
  friendly_name: Temperatura
sensor.humidity_158d000159c165:
  friendly_name: Humedad
  icon: mdi:water-percent
```

El resto de los sensores de temperatura y humedad personalizados se indican en el Anexo A.7. Cuando se pulsa en el nombre aparecerá la evolución histórica de los valores que ha tenido el sensor en el último día. Como se ha comentado anteriormente, para que Home Assistant te guarde estos datos se tiene que incluir el dominio sensor (en este caso) en el *recorder* del archivo de configuración.

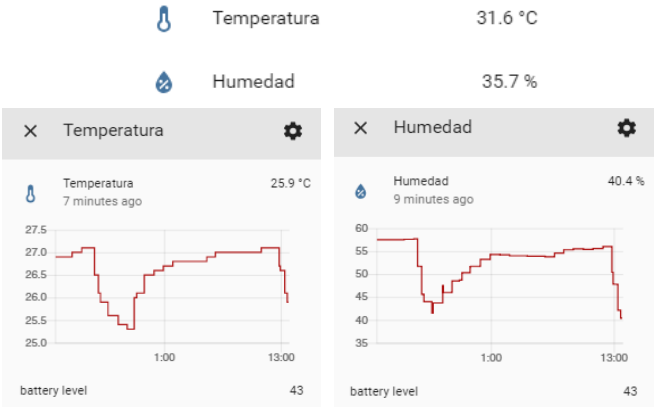


Figura 5-42. Sensor de temperatura y humedad.

```
# Sensor de Luminosidad del Gateway
sensor.illumination_286c07888ac2:
  friendly_name: Cantidad de luz
  icon: mdi:theme-light-dark

# Los nombres de los enchufes ya se personalizaron cuando se definieron en
# la plataforma broadlink. Sólo se cambiarán los iconos.
switch.cargador:
  icon: mdi:battery-charging-30
switch.despertador:
  icon: mdi:alarm
switch.lamparita:
  icon: mdi:lamp
switch.calientacamas:
  icon: mdi:hotel
switch.calientacamas_raul:
    icon: mdi:hotel
```

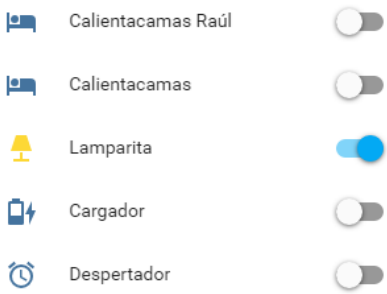


Figura 5-43. Enchufes personalizados.

 # Sensor de movimiento y pulsador
binary_sensor.motion_sensor_158d000171c445:
friendly_name: Sensor de movimiento
binary_sensor.switch_158d0001b12b32:
friendly_name: Pulsador

Para poder monitorizar el estado de las baterías de los sensores utilizados, sus pilas, se va a introducir un *sensor template* en el que si su valor es igual o menor a 5% que cambie el icono a una pila vacía para advertir de que está cerca de agotarse. El código para uno de los dispositivos se indica a continuación y para el resto se puede mirar en el Anexo A.8:

```
- platform: template
  sensors:
    movimiento_battery:
      friendly_name: 'Batería sensor movimiento'
      value_template: >
        {%- if states.binary_sensor.motion_sensor_158d000171c445.attributes.battery_level %}
          {{ states.binary_sensor.motion_sensor_158d000171c445.attributes.battery_level|round }}
        {% else %}
          {{ states.binary_sensor.motion_sensor_158d000171c445.state }}
        {%- endif %}
      icon_template: '{%- if  states.binary_sensor.motion_sensor_158d000171c445.attributes.battery_level <=
5 %}mdi:battery-outline{%- elif
states.binary_sensor.motion_sensor_158d000171c445.attributes.battery_level >= 95 %}mdi:battery{% else
%}mdi:battery-{{(states.binary_sensor.motion_sensor_158d000171c445.attributes.battery_level|float /
10)|round*10}}{%- endif %}'
      unit_of_measurement: "%"
```

Figura 5-44. Enchufes personalizados.

5.9.4. Batería de los dispositivos móviles

Cuando se integraron los dos *device_tracker* ya se configuró tanto el nombre como la foto que iba a aparecer en cada uno. Ahora, para conocer el estado de las baterías de los móviles, se incluirá en *sensores.yaml* un *template* con el siguiente código, donde el nombre de cada dispositivo se obtiene del archivo *known_devices.yaml*. Se incluye también un código para que vaya cambiando el icono de la batería en función del porcentaje de carga que tenga:

```
- platform: template
  sensors:
    battery_ule:
      friendly_name: 'Batería móvil uLe'
      unit_of_measurement: '%'
      value_template: >
        {%- if states.device_tracker.ule_777ceb627834417f9974559b39c44bd7.attributes.battery %}
           {{ states.device_tracker.ule_777ceb627834417f9974559b39c44bd7.attributes.battery|round }}
        {% else %}
           {{ states.sensor.battery_ule.state }}
        {%- endif %}
      icon_template: >
        {%- if is_state("sensor.battery_ule", "unknown") %}
           mdi:battery-unknown
        {%- elif is_state_attr("device_tracker.ule_777ceb627834417f9974559b39c44bd7", "battery_status",
"Charging") %}
           mdi:battery-charging
        {%- elif states.device_tracker.ule_777ceb627834417f9974559b39c44bd7.attributes.battery <= 5 %}
           mdi:battery-outline
        {%- elif states.device_tracker.ule_777ceb627834417f9974559b39c44bd7.attributes.battery >= 95 %}
           mdi:battery
        {% else %}
           mdi:battery-{{(states.device_tracker.ule_777ceb627834417f9974559b39c44bd7.attributes.battery|float
/ 10)|round*10}}
        {%- endif %}
```

El código para la batería del otro móvil se puede ver en el Anexo A.9.

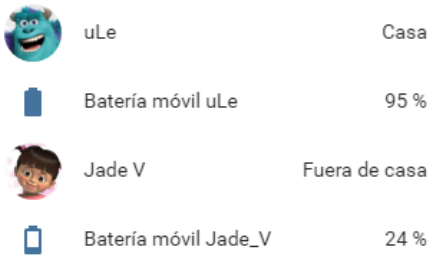


Figura 5-45. Localización y estados baterías.

Para poder comprobar mediante notificaciones al móvil cuando detecta la ubicación en casa o fuera, y poder realizar automatizaciones en función de la localización de las personas que viven en casa, se añadirá la siguiente automatización de aviso:

```
- alias: 'Aviso cambio estado ule home'
  trigger:
    platform: state
    entity_id: device_tracker.ule_767ceb627834417f9974559b39c44bd1
    from: 'not_home'
```

```
    to: 'home'
  action:
    - service: notify.ios_ulephone
      data:
        message: "La ubicación de uLe ha cambiado a {{
states.device_tracker.ule_767ceb627834417f9974559b39c44bd1.state }}"
```

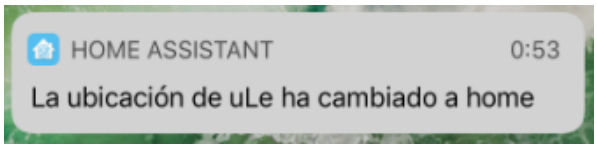


Figura 5-46. Notificación de ubicación.

Las demás notificaciones con los distintos estados y personas utilizan el mismo código y se pueden ver en el Anexo A.10.

5.9.5. Dispositivo SAI

Dentro de *sensores.yaml* se tiene que incluir la plataforma *nut* (*Network UPS Tools*) que se instaló anteriormente para poder monitorizar todas las variables que proporciona la herramienta. Para que se muestren en HA se incluirán las siguientes:

```
- platform: nut
  resources:
    - ups.status
    - ups.power.nominal
    - ups.realpower.nominal
    - input.voltage
    - battery.voltage
```

También se introducirá un *template* para traducir los códigos de todos los posibles estados de funcionamiento del SAI y el icono que aparecerá en función del estado en que se encuentre:

```
- platform: template
  sensors:
    ups_status:
      friendly_name: 'Estado de funcionamiento'
      value_template: >-
        {% if is_state('sensor.nut_ups_status_data', 'OL CHRG') %}
          En línea, cargando batería
        {% elif is_state('sensor.nut_ups_status_data', 'OB DISCHRG') %}
          Batería en uso
        {% elif is_state('sensor.nut_ups_status_data', 'LB') %}
          Batería baja
        {% elif is_state('sensor.nut_ups_status_data', 'OL') %}
          En línea
        {% elif is_state('sensor.nut_ups_status_data', 'SD') %}
          Shutdown load
        {% elif is_state('sensor.nut_ups_status_data', 'CP') %}
          Cable power
        {% elif is_state('sensor.nut_ups_status_data', 'CTS') %}
          Clear to Send
        {% elif is_state('sensor.nut_ups_status_data', 'RTS') %}
          Ready to send
        {% elif is_state('sensor.nut_ups_status_data', 'DCD') %}
```

```
    Data carrier detect
  {% elif is_state('sensor.nut_ups_status_data', 'RNG') %}
    Ring indicate
  {% elif is_state('sensor.nut_ups_status_data', 'DTR') %}
    Data terminal ready
  {% elif is_state('sensor.nut_ups_status_data', 'ST') %}
    Send break
  {% endif %}
 icon_template: >-
  {% if is_state('sensor.ups_status_data', 'OL CHRG') %}
    mdi:battery-charging-70
  {% elif is_state('sensor.ups_status_data', 'OB DISCHRG') %}
    mdi:battery-charging-wireless-alert
  {% else %}
    mdi:battery-charging
  {% endif %}
```

Los iconos y nombres que se quieran cambiar se personalizan en el archivo *customize.yaml*:

```
sensor.nut_ups_nominal_power:
  friendly_name: Potencia
  icon: mdi:flash
sensor.nut_ups_nominal_real_power:
  friendly_name: Potencia real
  icon: mdi:flash-circle
sensor.nut_ups_input_voltage:
  friendly_name: Tensión de entrada
  icon: mdi:power-socket-eu
sensor.nut_ups_battery_voltage:
  friendly_name: Tensión batería
  icon: mdi:battery-positive
```

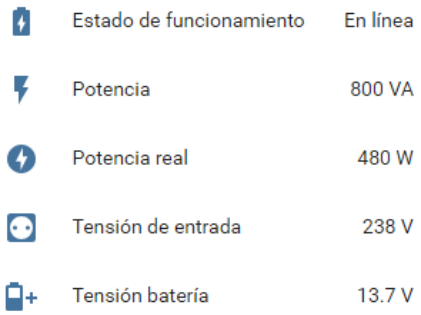


Figura 5-47. Datos del SAI.

Se pueden configurar algunas notificaciones para recibir avisos cuando el SAI pase a algún estado concreto, así se sabrá por ejemplo cuando se ha ido la luz en casa. En este caso enviará la información a través de la app de Home Assistant en iOS. Se añadirá en el archivo *automations.yaml* el siguiente código:

```
- alias: 'Notificación SAI cargando'
  trigger:
    platform: state
    entity_id: sensor.nut_ups_status_data
    to: 'OL CHRG'
  action:
    - service: notify.ios_ulephone
```

```
  data:
    message: "¡¡Ha vuelto la luz!! El SAI se está cargando"
```

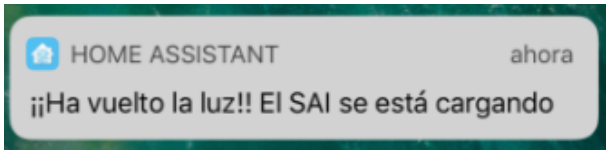


Figura 5-48. Notificación sobre el funcionamiento del SAI.

En el Anexo A.11 se incluyen un par de notificaciones más sobre el estado del SAI.

5.9.6. Información sobre Home Assistant

Dentro de la plataforma de HA existen varios sensores propios que proporcionan datos del sistema. El objetivo será conocer la versión actual de Home Assistant que está instalada en la Raspberry Pi, que avise si hay alguna actualización nueva y cuál es esa versión, y el tiempo de funcionamiento que lleva HA desde el último reinicio.

Para ello se utilizará la plataforma *sensor version* que muestra la versión actual de la plataforma y el *sensor scrape* que obtiene información determinada de las páginas webs. El sensor carga una página HTML, busca la información indicada y obtiene el dato que se quiera, aunque en principio sólo sirve para páginas con códigos de programación simples.

También se incluirán unos *templates* para realizar la comparación de versiones e indicar en la interfaz web si hay una actualización nueva:

```
################################
#      Sensores Versión HA     #
################################

- platform: version
  name: Versión actual instalada

# Un sensor para comprobar la última versión disponible.
- platform: scrape
  resource: https://home-assistant.io
  name: Última versión disponible
  select: ".current-version h1"
  value_template: '{{ value.split(":")[1] }}'
  scan_interval: 3600

# Otro sensor con distinta frecuencia de muestreo para usarlo como control de cambio
# de versiones, ya que uno siempre cambiará antes que el otro.
- platform: scrape
  resource: https://home-assistant.io
  name: Nueva versión disponible
  select: ".current-version h1"
  value_template: '{{ value.split(":")[1] }}'
  scan_interval: 4200
```

Con esto buscará cada cierto tiempo en la página web de Home Assistant el trozo de código indicado en el *scrape* y devolverá su valor.

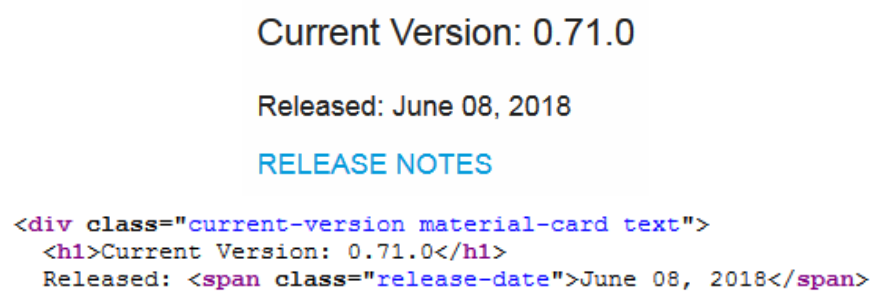


Figura 5-49. Información gráfica y código HTML de la web de HA.

El código del primer *template* sirve para comparar los dos sensores *scrape* y el que se usará para las sucesivas notificaciones de actualización de versión, excepto el primer cambio, que se realiza con el segundo *template*. Éste último también es el que muestra por pantalla un *Sí* o un *No* y un icono distinto en función de si existe nueva versión para instalar:

```
- platform: template
  sensors:
    ha_version_update:
      friendly_name: 'HA Cambio de Versión'
      value_template: >-
        {%- if states.sensor.ultima_version_disponible.state != states.sensor.nueva_version_disponible.state -
%}
          Sí
        {%- else -%}
          No
        {%- endif -%}


- platform: template
  sensors:
    ha_update_available:
      friendly_name: '¿Actualización disponible?'
      value_template: >-
        {%- if states.sensor.ultima_version_disponible.state != "unavailable" and
states.sensor.ultima_version_disponible.state != states.sensor.version_actual_instalada.state -%}
          Sí
        {%- else -%}
          No
        {%- endif -%}
      icon_template: >-
        {%- if states.sensor.ultima_version_disponible.state != "unavailable" and
states.sensor.ultima_version_disponible.state != states.sensor.version_actual_instalada.state -%}
          mdi:new-box
        {% else %}
          mdi:verified
        {% endif %}
```

Los iconos de los dos sensores que indican las versiones se cambiaran en el *customize.yaml*:

```
sensor.version_actual_instalada:
  icon: mdi:home-assistant
sensor.ultima_version_disponible:
  icon: mdi:alert-circle-outline
```

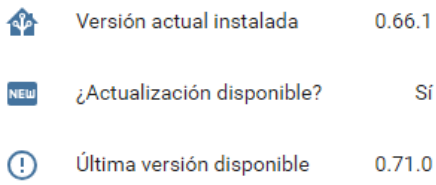


Figura 5-50. Información sobre la plataforma HA.

El código para las notificaciones sobre el primer cambio de versión y para las siguientes actualizaciones se muestra en el Anexo A.12.

Por último, para saber el tiempo que lleva funcionando Home Assistant desde el último reinicio, se utilizará el *sensor uptime*. El problema que tiene es que da el tiempo en minutos y cuando se llevan varios días es complicado de calcular. Para personalizarlo se añadirá un *template* para convertir los minutos en días, horas y minutos. Este último código se mostrará en el Anexo A.13 ya que es un poco extenso:

```
- platform: uptime
  name: Tiempo funcionando
  unit_of_measurement: minutes
```

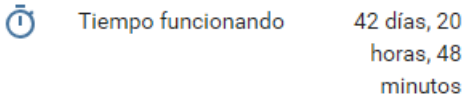


Figura 5-51. Información sobre tiempo de funcionamento de HA.

5.9.7. Aviso caducidad certificado SSL

Anteriormente se configuró el certificado de la página *DuckDNS* y su renovación automática que se debe hacer unos 30 días antes de su vencimiento. Si esto no ocurre, se activará una alerta cuando el tiempo de caducidad del certificado se reduzca de 21 días. Esto dará 3 semanas para solucionar el problema, instalar el nuevo certificado y obtener otros 90 días de conexiones seguras en Home Assistant.

Se tiene que instalar la siguiente librería por lo que se pondrá en terminal: ***sudo apt-get install ssl-cert-check***.

Para integrarlo en HA se utilizará la plataforma sensor *command_line*, que emite comandos específicos para obtener los datos que se quieran. El código de configuración del sensor queda así:

```
- platform: command_line
  name: SSL cert expiry
  unit_of_measurement: días
  scan_interval: 10800
  command: "ssl-cert-check -b -c /home/homeassistant/dehydrated/certs/ulesin.duckdns.org/cert.pem | awk '{ print $NF }'"
```

La automatización para la notificación al terminal móvil:

```
# Notificación para cuando quedan menos de 21 días para caducar
- alias: 'SSL expiry notification'
  trigger:
    platform: numeric_state
    entity_id: sensor.ssl_cert_expiry
    below: 21
  action:
```

```
  service: notify.ios_ulephone
  data:
    message: '¡Cuidado! El certificado SSL expira en 21 días y no se ha renovado automáticamente'
```

Y se personaliza el nombre y el icono en el *customize.yaml*:

```
sensor.ssl_cert_expiry:
 friendly_name: Caducidad certificado SSL
 icon: mdi:calendar-clock
```

Figura 5-52. Tiempo de caducidad certificado SSL.

5.9.8. Información sobre la Raspberry Pi 3

Para poder monitorizar el estado del servidor, usaré la plataforma *systemmonitor* que da información sobre el uso de la tarjeta microSD, de la memoria RAM y de la CPU, y el tiempo y la fecha del último reinicio de la Raspberry. Para obtener la temperatura de la CPU utilizaré la plataforma *línea de comandos*:

```
- platform: command_line
  name: CPU temperatura
  command: "cat /sys/class/thermal/thermal_zone0/temp"
  unit_of_measurement: "°C"
  value_template: '{{ value | multiply(0.001) | round(1) }}'
  scan_interval: 45

- platform: systemmonitor
  name: RPI Host
  scan_interval: 45
  resources:
    - type: disk_use_percent
      arg: /
    - type: disk_use
      arg: /
    - type: disk_free
      arg: /
    - type: memory_free
    - type: memory_use
    - type: memory_use_percent
    - type: processor_use
    - type: last_boot
    - type: since_last_boot
```

Al igual que ocurría con el *sensor uptime*, se tiene que usar un *template* para convertir los minutos en días, horas y minutos. El código se muestra en el Anexo A.14 y la personalización de todos los nombres e iconos en el Anexo A.15.

Figura 5-53. Datos sobre la Raspberry Pi.

Una notificación que se va a configurar es que avise cuando la Raspberry Pi 3 pase de una determinada temperatura, por ejemplo 60 ºC, para así poder monitorizar y actuar en caso necesario. En el *automations.yaml* se añade el siguiente código:

```
- alias: 'Notificación temperatura máxima CPU'
  trigger:
    platform: numeric_state
    entity_id: sensor.cpu_temperatura
    above: 60
  action:
    - service: notify.ios_ulephone
      data:
        message: 'La temperatura de la CPU ha llegado a los {{ states.sensor.cpu_temperatura.state }} ºC'
```

Figura 5-54. Temperatura de la CPU y notificación.

5.9.9. Información sobre internet

Para conocer datos sobre la conexión a internet se utilizará la plataforma sensor *speedtest*, que utiliza el servicio web *speedtest.net* para medir el rendimiento del ancho de banda de la red. Dará tres parámetros cada 45 minutos (se puede cambiar), la velocidad de bajada, de subida y el *ping*, de un servidor concreto que será el más cercano al hogar y que se podrá averiguar en el listado que facilitan desde su propia web. Se integra en HA con el código:

```
- platform: speedtest
  server_id: 3465
  minute:
    - 45
  monitored_conditions:
    - ping
    - download
    - upload
```

Y se personalizan los nombres en el *customize.yaml*:

```
sensor.speedtest_download:
  friendly_name: Velocidad Bajada
sensor.speedtest_ping:
  friendly_name: Ping
sensor.speedtest_upload:
  friendly_name: Velocidad Subida
```

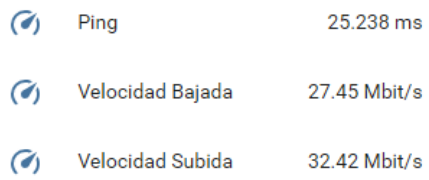


Figura 5-55. Datos sobre conexión a internet.

5.9.10. Añadir elementos en el menú lateral

En este caso se va a configurar el panel lateral de Home Assistant para incorporar el mapa del tiempo y poder ver la posibilidad de lluvias en directo. Se utilizará el *panel_iframe* en el *configuration.yaml* para que aparezca en el menú lateral y se incorporará la dirección de la página web donde aparece el mapa del tiempo, en este caso de *rainviewer.com*.

```
panel_iframe:
  tiempo:
    title: 'Tiempo'
    url: 'https://www.rainviewer.com/map.html?loc=40.298793,-
3.772578,6&oFa=0&oC=0&oU=0&oCUB=1&oCS=0&oF=0&oAP=1&rmt=1'
    icon: mdi:weather-sunny
```

También se puede configurar el acceso a una página web determinada o a la dirección del router de casa:

```
  router:
    title: 'Router'
    url: ' http://192.168.0.1'
    icon: mdi: router-wireless
```

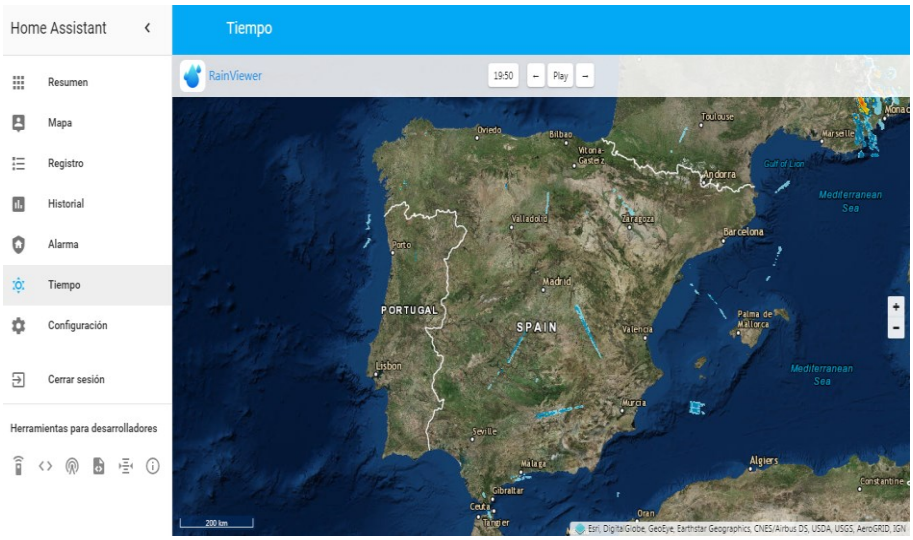


Figura 5-56. Panel lateral con mapa del tiempo.

5.9.11. Meteorología y climatología

Para tener un sistema domótico lo más completo posible no puede faltar la información relativa al tiempo. Con esto se puede jugar para crear avisos o automatizaciones que dependan por ejemplo de la estación del año, de la posición del sol durante el día o de las fases lunares. Existen varios tipos de sensores para ello.

Por un lado está la plataforma *sun*, que ya se comentó anteriormente y que aporta datos sobre el amanecer y el atardecer, su elevación y el ángulo de incidencia de los rayos de sol. Además, las otras dos plataformas que se usarán serán los sensores *season* y *moon*. El primero da la estación del año (primavera, verano, otoño o invierno) y el otro realiza un muestreo diario de la luna e indica en cual de las 8 fases se encuentra.

Por otro lado, aunque hay otras páginas disponibles, se ha elegido la plataforma *wunderground* que utiliza la web *Weather Underground* como fuente de información meteorológica. Ésta da un montón de datos meteorológicos incluido previsiones para los siguientes días. Para poder utilizarla se necesita hacer un registro gratuito en su página web para que aporten una clave API que será la que obtenga los datos que se reflejaran en la página de Home Assistant. Opcionalmente se puede elegir la estación de datos que esté más cerca de casa, la localización del hogar y el idioma.

Para integrarlo en HA se pondrá el código en *sensores.yaml* y luego se podrán agrupar por tipo de datos cuando se organicen las pestañas y grupos:

```
- platform: wunderground
  api_key: 1f548fa1397d50bb
  pws_id: IALSEVIL4
  latitude: 38.3925
  longitude: -7.9940
  lang: SP
  monitored_conditions:
    - alerts
    - feelslike_c
    - heat_index_c
    - location
    - precip_today_metric
    …
```

El resto de parámetros, que son 34 en total, se muestran en el Anexo A.16.

Figura 5-57. Predicción y datos históricos de la temperatura.

Los nombres personalizados para que sean entendibles se muestran en el Anexo A.17.

Figura 5-58. Predicción y datos históricos de la temperatura.

Ahora se muestra el sensor *season* para indicar la estación del año:

- platform: season

Para hacer la traducción al español del nombre de la estación y ponerle un icono distinto para cada una de ellas, se introducirá un *template* con el siguiente código:

```
- platform: template
  sensors:
    estacion:
      friendly_name: 'Estación'
      value_template: >
        {% set season ={"spring": "Primavera", "summer": "Verano", "autumn": "Otoño", "winter": "Invierno"} %}
        {{ season[states("sensor.season")] }}
      icon_template: >
        {% if is_state('sensor.season', 'spring') %}
          mdi:flower
        {% elif is_state('sensor.season', 'summer') %}
          mdi:weather-sunny
```

```
    {% elif is_state('sensor.season', 'autumn') %}
      mdi:leaf
    {% elif is_state('sensor.season', 'winter') %}
      mdi:snowflake
    {% endif %}
```

Y así se integra el sensor *moon* para indicar las fases lunares:

```
- platform: moon
```

Aquí también se van a personalizar los nombres en español y a poner una foto de cada estado lunar en vez de un icono. Para ello se pegan todas las fotos en la carpeta local */www* que está dentro del directorio principal de *homeassistant*. El código completo de este *template* se incluye en el Anexo A.18.

```
- platform: template
  sensors:
    moon_phase:
      friendly_name: 'Fase Lunar'
      value_template: >
        {% if is_state('sensor.moon', 'New moon') %}
          Luna nueva
        …
      entity_picture_template: >-
        {% if is_state('sensor.moon', 'New moon') %}
          /local/nueva.jpg
        …
        {% endif %}
```

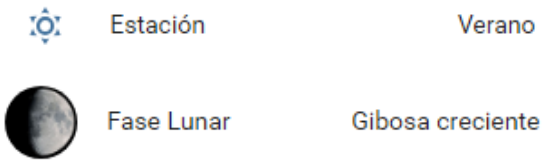


Figura 5-59. Estación y fase lunar personalizada.

Se pueden configurar notificaciones para que informe del cambio de estación y de las fases de la luna. Para ello, se indicará en el código de integración de la automatización un evento del tipo *state_changed*, es decir, siempre que cambie un estado realiza la notificación. En este caso para las cuatro estaciones y las ocho fases lunares. Se pueden ver los dos códigos en el Anexo A.19.

Otra configuración que se puede realizar será para que muestre el horario del amanecer y atardecer, personalizando los atributos de la plataforma *sun*. Para ello se crea un *template* que dará el horario de cada uno con el formato hora y minuto, y con el icono correspondiente:

```
- platform: template
  sensors:
    sunrise:
      friendly_name: 'Amanecer'
      value_template: '{{ as_timestamp(states.sun.sun.attributes.next_rising) |
timestamp_custom("%H:%M")}}'
      icon_template: mdi:weather-sunset-up
    sunset:
      friendly_name: 'Puesta de Sol'
      value_template: '{{ as_timestamp(states.sun.sun.attributes.next_setting) |
timestamp_custom("%H:%M")}}'
      icon_template: mdi:weather-sunset-down
```

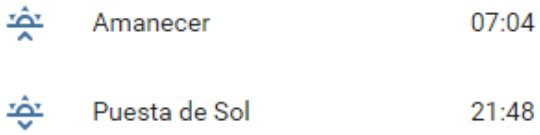


Figura 5-60. Horarios del amanecer y atardecer.

5.9.12. Fecha personalizada

Para que aparezca la fecha actual, con el día de la semana y el mes traducidos al español, se aplicará un *template* con el siguiente código:

```
- platform: template
  sensors:
    date_long:
      friendly_name: 'Fecha'
      value_template: >
        {% set months = ["Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto",
"Septiembre", "Octubre", "Noviembre", "Diciembre"] %}
        {% set month = months[now().strftime('%m') | int -1] %}
        {% set dias = ["Lunes ", "Martes ", "Miércoles ", "Jueves ", "Viernes ", "Sábado ", "Domingo "]
%}
        {% set dia = dias[now().strftime('%u') | int -1] %}
        {% set year = now().strftime('%Y') %}
        {{ dia + ',' + ' ' + now().strftime('%d') + ' ' + month + ' ' + year }}
      icon_template: mdi:calendar
```

Figura 5-61. Fecha personalizada.

5.9.13. Añadir imagen animada

Dentro del apartado *camera* existe la opción de utilizar la plataforma genérica para insertar una imagen estática (.jpg) o dinámica (.gif) en la interfaz gráfica de HA. En este caso se va a incorporar un pequeño mapa del tiempo que se reproducirá continuamente. Basta con saber la dirección URL de la imagen gif, en este caso de la web meteorológica *sat24.com*:

```
camera:
  - platform: generic
    still_image_url:
https://api.sat24.com/animated/SP/infraPolair/1/Romance%20Standard%20Time/3774382
    name: Mapa Tiempo
```

Figura 5-62. Mapa del tiempo animado.

5.9.14. Notificaciones accionables para encender la climatización

Se van a crear dos notificaciones con uno de los sensores de temperatura de la casa para cuando baje o suba de una temperatura determinada, avise al móvil y dé la opción de poder encender el aire (frío o calor) sin tener que entrar en la interfaz web de Home Assistant.

Esto sólo se puede hacer con la app de iOS y se llaman *notificaciones accionables*. Cada vez que se crea o se actualiza una notificación accionable en HA, hay que actualizar las configuraciones de dentro de la aplicación de iOS. Para ello, se entra en Ajustes, menú Configuración de las notificaciones y se pulsa en Actualizar ajustes push.

El código que hay que meter en el archivo *configuration.yaml* debajo de la categoría *ios:* será el siguiente. El otro caso de temperatura máxima funciona de la misma manera y se incluirá en el Anexo A.20. Es importante poner tanto el *identificador* como el *nombre de acción* en mayúsculas para que funcione:

```
ios:
  push:
    categories:
      - name: Temperatura minima
        identifier: temperatura_minima
        actions:
          - identifier: ENCENDER_AIRE
            title: 'Encender el aire'
            activationMode: 'background'
            authenticationRequired: no
            destructive: yes
            behavior: 'default'
      - name: Temperatura maxima
        identifier: temperatura_maxima
        actions:
          - identifier: ENCENDER_AIRE
            title: 'Encender el aire'
            activationMode: 'background'
            authenticationRequired: no
            destructive: yes
            behavior: 'default'
```

En *automations.yaml* se tendrán que poner los dos siguientes códigos, uno para la notificación con el mensaje de aviso cuando la temperatura baja de 20.8 ºC, y otro para encender el *switch* correspondiente a la climatización de casa una vez que se pulsa el mensaje *Encender el aire*:

```
# Temperatura mínima: Actionable Push Message
- alias: 'Temperatura mínima'
```

```
  trigger:
    platform: numeric_state
    entity_id: sensor.temperature_158d000158599d
    below: 20.8
  action:
    - service: notify.ios_ulephone
      data:
        message: "La temperatura del dormitorio es demasiado baja ({{
states.sensor.temperature_158d000158599d.state }}ºC), ¿quieres poner el aire?"
        data:
          push:
            badge: 5
            category: "temperatura_minima"

# Temperatura mínima: Actionable Push Response
- alias: 'Push notify action temperatura'
  initial_state: 'on'
  trigger:
    platform: event
    event_type: ios.notification_action_fired
    event_data:
      actionName: ENCENDER_AIRE
  action:
    service: homeassistant.turn_on
    entity_id: switch.encendido
```

Para que funcione, una vez que llega el mensaje se desliza hacia la izquierda, se pulsa *ver* y saldrá otra pantalla con el mismo mensaje pero con un botón que pone *Encender el aire*. Si se quiere encender se pulsa y si no se cierra el mensaje.

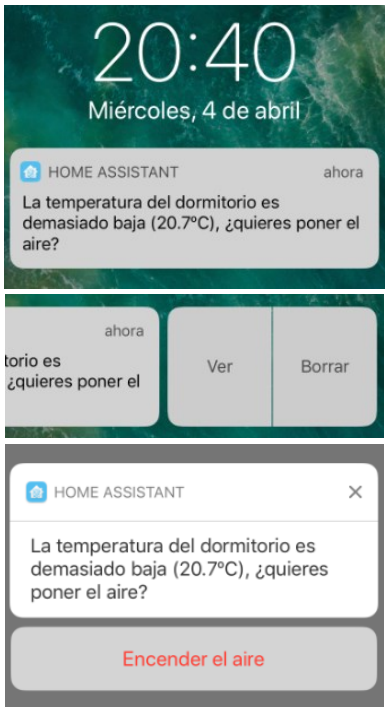


Figura 5-63. Notificación accionable en app iOS.

5.9.15. Automatizaciones personalizadas

En este apartado se van a añadir cuatro automatizaciones diseñadas para completar el sistema domótico en función de determinadas necesidades.

La primera será la de encender la luz del *Gateway* del salón por la mañana en un horario determinado, para tener una luz ambiente y que no esté a oscuras el salón. Se encendería de lunes a viernes de 6:30 a 7:00.

```
- alias: Encendido gateway por la mañana
  trigger:
    platform: time
    at: '06:30:00'
  condition:
    condition: time
    weekday:
      - mon
      - tue
      - wed
      - thu
      - fri
  action:
    - service: light.turn_on
      entity_id: light.gateway_light_286c07888ac2
    - delay: '00:30:00'
    - service: light.turn_off
      entity_id: light.gateway_light_286c07888ac2
```

La segunda será la de crear un modo vacaciones en el que se desactiven las automatizaciones que no se quieran ejecutar en ese periodo y, además, encenderá la luz de la entradita todos los días de 22:00 a 23:59. Aquí no se activa la alarma, es independiente a esta automatización. Este modo se activará haciendo doble clic en el botón *Wireless*, y más en concreto, desactiva el encendido del *Gateway* por las mañanas y la programación semanal del robot de limpieza. También llegará al móvil un mensaje personalizado:

MODO VACACIONES

```
- alias: Activar modo vacaciones
  trigger:
    platform: event
    event_type: click
    event_data:
      event_id: binary_sensor.switch_158d0001b12b32
      click_type: double
  condition:
    condition: state
    entity_id: automation.modo_vacaciones
    state: 'off'
  action:
    - service: automation.turn_on
      entity_id: automation.modo_vacaciones
    - service: automation.turn_off
      entity_id: automation.encendido_gateway_por_la_manana
    - service: homeassistant.turn_off
      entity_id: input_boolean.estado_on_p1
    - service: notify.ios_ulephone
      data:
        message: 'Se ha activado el modo vacaciones, ¡buen viaje!'
```

Para desactivarlo también se pulsaría dos veces en el botón. Se comprueba que el modo vacaciones está activo para volver a activar las automatizaciones que se apagaron anteriormente. También se hace que suene una música de bienvenida en el *Gateway* para comprobar que se ha ejecutado correctamente y un mensaje

personalizado.

```
- alias: Desactivar modo vacaciones
  trigger:
    platform: event
    event_type: click
    event_data:
      event_id: binary_sensor.switch_158d0001b12b32
      click_type: double
  condition:
    condition: state
    entity_id: automation.modo_vacaciones
    state: 'on'
  action:
    - service: automation.turn_off
      entity_id: automation.modo_vacaciones
    - service: xiaomi_aqara.play_ringtone
      data:
        gw_mac: 38:6C:07:77:8A:C1
        ringtone_id: 25
        ringtone_vol: 20
    - service: automation.turn_on
      entity_id: automation.encendido_gateway_por_la_manana
    - service: automation.turn_on
      entity_id: input_boolean.estado_on_p1
    - service: notify.ios_ulephone
      data:
        message: 'Se ha desactivado el modo vacaciones, ¡bienvenido de nuevo!'
```

Cuando se activa el modo vacaciones, la automatización hace que se encienda la luz de la entrada durante dos horas. El código es el siguiente:

```
- alias: Modo vacaciones
  initial_state: 'off'
  trigger:
    platform: time
    at: '22:00:00'
  action:
    - service: light.turn_on
      entity_id: light.yeelight_rgb_7811dc6840bc
    - delay: '01:59:00'
    - service: light.turn_off
      entity_id: light.yeelight_rgb_7811dc6840bc
```

La tercera será la de crear un modo fuera de casa que encienda la luz de la entrada cuando el sensor de movimiento detecte que se entre en casa, ya que el interruptor no está a mano. Las condiciones para que se encienda la luz serán que no haya nadie en casa y que se pueda encender a partir de una hora antes del atardecer.

Para activarlo se ha puesto que lo haga cuando todos los miembros de la casa estén fuera, utilizando el *device_tracker*, y siempre que haya condiciones de oscuridad a partir de una hora antes del atardecer y hasta el amanecer. También enviará un mensaje de confirmación al móvil que el modo se ha activado:

```
# MODO FUERA DE CASA
- alias: activar luz entrada cuando detecte not home
  trigger:
    platform: state
    entity_id: group.all_devices
    from: 'home'
```

```
    to: 'not_home'
  condition:
    condition: and
    conditions:
      - condition: state
        entity_id: automation.luz_movimiento_entrada
        state: 'off'
      - condition: or
        conditions:
          - condition: sun
            after: sunset
            after_offset: '-1:00:00'
          - condition: sun
            before: sunrise
  action:
    - service: automation.turn_on
      entity_id: automation.luz_movimiento_entrada
    - service: notify.ios_ulephone
      data:
        message: 'Se ha activado el modo fuera de casa para la luz de entrada'
```

Para desactivarlo se hará mediante una pulsación en el botón de *Xiaomi*. Sonará una música en el *Gateway* para confirmar que se ha desactivado correctamente y se enviará una notificación al móvil. Este modo no activa la alarma.

```
- alias: Desactivar modo fuera casa
  trigger:
    platform: event
    event_type: click
    event_data:
      event_id: binary_sensor.switch_158d0001b12b32
      click_type: single
  condition:
    condition: state
    entity_id: automation.luz_movimiento_entrada
    state: 'on'
  action:
    - service: automation.turn_off
      entity_id: automation.luz_movimiento_entrada
    - service: xiaomi_aqara.play_ringtone
      data:
        gw_mac: 38:6C:07:77:8A:C1
        ringtone_id: 12
        ringtone_vol: 20
    - service: light.turn_off
      entity_id: light.yeelight_rgb_7811dc6840bc
    - service: notify.ios_ulephone
      data:
        message: 'Se ha desactivado el modo fuera de casa para la luz de entrada'
```

Cuando se activa el modo fuera de casa, la automatización hace que cuando el sensor de movimiento detecte presencia se encienda la luz de la entrada siempre y cuando sea en condiciones de oscuridad: una hora antes del atardecer y hasta el amanecer. Cuando se enciende la luz envía un mensaje de bienvenida a casa. El código es el siguiente:

```
- alias: Luz movimiento entrada
  initial_state: 'off'
  trigger:
    - platform: state
```

```
    entity_id: binary_sensor.motion_sensor_158d000171c445
    from: 'off'
    to: 'on'
  condition:
   condition: and
   conditions:
    - condition: state
      entity_id: light.yeelight_rgb_7811dc6840bc
      state: 'off'
    - condition: state
      entity_id: automation.luz_movimiento_entrada
      state: 'on'
    - condition: or
      conditions:
       - condition: sun
         after: sunset
         after_offset: '-1:00:00'
       - condition: sun
         before: sunrise
  action:
   - service: light.turn_on
     entity_id: light.yeelight_rgb_7811dc6840bc
   - service: notify.ios_ulephone
     data:
       message: 'Bienvenido de nuevo a casa'
```

La cuarta y última automatización consiste en crear un modo día de fiesta. Esto hace que no se ejecuten un par de automatizaciones que se quieren hacer cuando es un día de fiesta y no se trabaja, como son el encendido del *Gateway* por la mañana y la limpieza del suelo con el robot, que se activará manualmente a otra hora que más convenga. Se activará y desactivará mediante un interruptor virtual que se crea mediante un *booleano* en el *configuration.yaml*:

```
input_boolean:
  dia_fiesta:
    name: Modo día de fiesta
    icon: mdi:calendar-today
```

Los dos códigos para el *automations.yaml* serán los siguientes:

```
- alias: "modo dia fiesta on"
  trigger:
    - platform: state
      entity_id: input_boolean.dia_fiesta
      to: 'on'
  action:
    - service: automation.turn_off
      entity_id: automation.encendido_gateway_por_la_manana
    - service: homeassistant.turn_off
      entity_id: input_boolean.estado_on_p1

- alias: "modo dia fiesta off"
  trigger:
    - platform: state
      entity_id: input_boolean.dia_fiesta
      to: 'off'
  action:
    - service: automation.turn_on
      entity_id: automation.encendido_gateway_por_la_manana
```

```
  - service: homeassistant.turn_on
    entity_id: input_boolean.estado_on_p1
```

Figura 5-64. Botón para el modo día de fiesta.

5.10. Organización final en grupos y pestañas

Para el diseño de la página principal de Home Assistant se ha optado por dividirlo en cinco pestañas y con iconos representativos en vez de con nombres (aparecen comentados en el código). Para ello, se escribirá en el archivo de configuración el texto *group: !include groups.yaml*, y se introducirán las líneas de código en el archivo *groups.yaml*.

La primera pestaña será la vista por defecto y representa seis grupos: cuatro por cada habitación de la casa, otro de localización de personas y un último con las imágenes de la cámara.

En la segunda pestaña se incorporarán los grupos del sistema de climatización, la televisión y todo lo relacionado con el robot de limpieza.

La tercera será una pestaña de información donde vendrán datos sobre el sistema domótico, como son Home Assistant, la Raspberry Pi 3, la conexión a Internet, el estado del SAI y la carga de las baterías de los sensores.

Una cuarta pestaña en el que se incorporan todos los datos sobre meteorología y climatología, y otra última con todas las automatizaciones que se han integrado en HA para saber cuál está activa y poder desactivarlas cuando se desee.

```
##########################
#     Pestañas principales     #
##########################

default_view:
  #name: Hogar
  view: yes
  icon: mdi:home
  entities:
    - group.salon
    - group.cuarto_chico
    - group.dormitorio
    - group.exterior
    - group.presencia
    - camera.foscam

general:
  #name: General
  icon: mdi:remote
  view: yes
  entities:
    - group.climatizacion
    - group.aireacondicionado
    - media_player.samsung_tv_remote
    - group.robot

datos:
  #name: Datos
```

```
  view: yes
  icon: mdi:settings
  entities:
    - group.homeassistant
    - group.internet
    - group.sistema
    - group.sai
    - group.baterias

el_tiempo:
  #name: Tiempo
  view: yes
  icon: mdi:weather-partlycloudy
  entities:
    - group.tiempo_actual
    - group.info_tiempo
    - group.historico
    - group.prediccion
    - group.mapas

automatizaciones:
  #name: Automatizaciones
  view: yes
  icon: mdi:angularjs
  entities:
    - group.autos
    - group.scripts
```

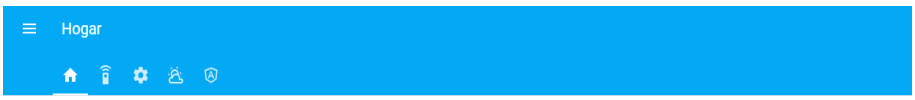


Figura 5-65. Cabecera principal de Home Assistant.

Ahora en cada grupo se tienen que incorporar todos los sensores y dispositivos que se han configurado para que aparezcan agrupados en función de la información que más se relacionen entre ellos. Igual que el anterior código también se incorpora a *groups.yaml*:

```
##############################
#    Grupos por habitación     #
##############################

salon:
  name: Salón
  view: no
  control: hidden
  entities:
    - sensor.temperature_158d000159c165
    - sensor.humidity_158d000159c165
    - sensor.illumination_286c07888ac2
    - light.yeelight_rgb_7811dc6840bc
    - light.gateway_light_286c07888ac2
    - binary_sensor.motion_sensor_158d000171c445
    - binary_sensor.switch_158d0001b12b32

exterior:
```

```
  name: Exterior
  view: no
  control: hidden
  entities:
    - sensor.temperature_158d0001f52cca
    - sensor.humidity_158d0001f52cca
    - sensor.pressure_158d0001f52cca
cuarto_chico:
  name: Cuarto chico
  view: no
  control: hidden
  entities:
    - sensor.temperature_158d0001825324
    - sensor.humidity_158d0001825324
dormitorio:
  name: Dormitorio
  view: no
  control: hidden
  entities:
    - sensor.temperature_158d000158599d
    - sensor.humidity_158d000158599d
    - switch.calientacamas_raul
    - switch.calientacamas_susana
    - switch.lamparita
    - switch.cargador
    - switch.despertador
presencia:
  name: ¿Quién anda ahí?
  view: no
  control: hidden
  entities:
    - device_tracker.ule_767ceb627834417f9974559b39c44bd1
    - sensor.battery_ule
    - device_tracker.jade_v_d23ba49fdb064b0bb0071ce3c11bd5bd
    - sensor.battery_jade_v
```

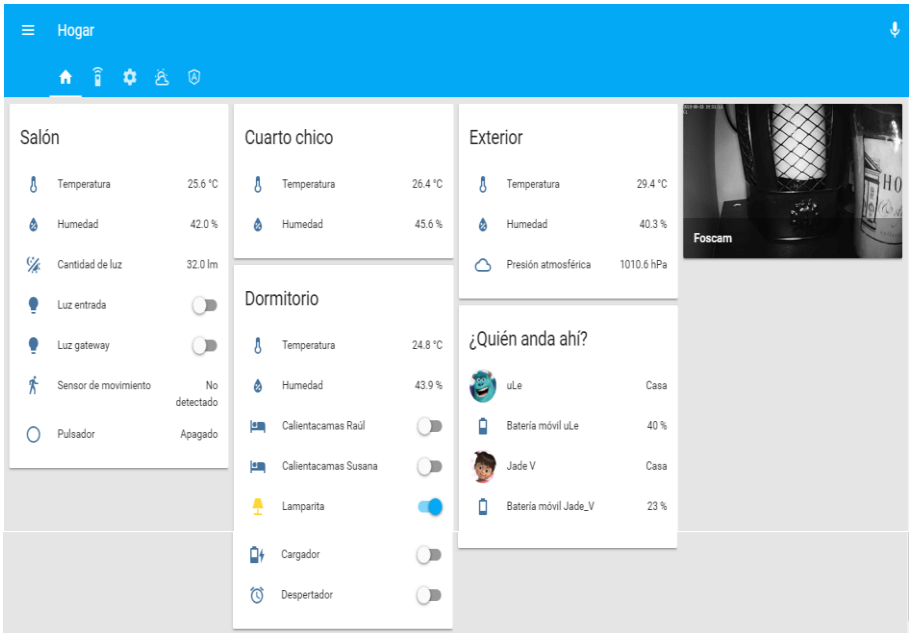


Figura 5-66. Pestaña principal por habitaciones.

```
############################
#      Grupos por servicios      #
############################

climatizacion:
  name: Climatización
  view: no
  control: hidden
  entities:
    - switch.encendido
    - script.aire_subir_temp
    - script.aire_bajar_temp
    - script.aire_modo
    - script.aire_ventilador

aireacondicionado:
  name: Display de climatización
  view: no
  control: hidden
  entities:
    - input_number.ac_temperature
    - input_select.ac_operation_mode
    - input_select.ac_fan_mode

robot:
  name: Robot de limpieza
  view: no
  control: hidden
  entities:
    - vacuum.ameba
    - counter.ameba
    - sensor.limpieza_ameba
    - script.vacuum_service_mode
    - input_boolean.estado_on_p1
    - sensor.inicio_p1
    - input_number.hora_on_p1
    - input_number.minutos_on_p1
    - input_boolean.entresemana_p1
    - input_boolean.finde_p1
```

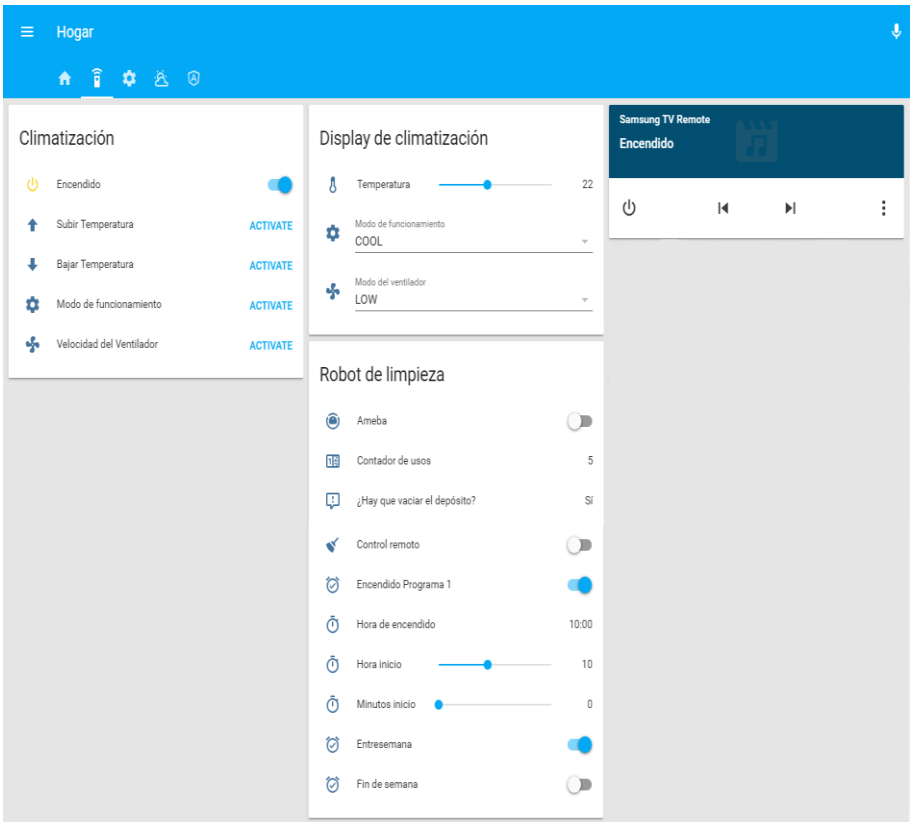


Figura 5-67. Pestaña de servicios.

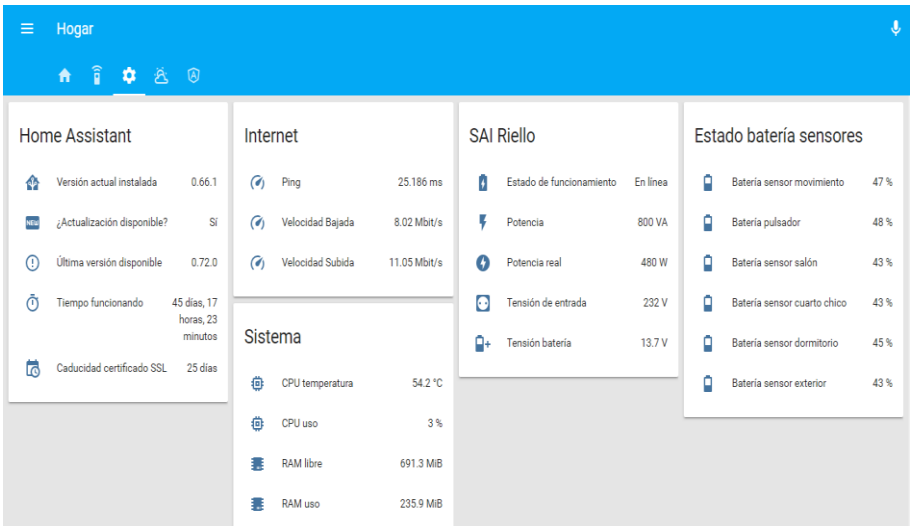
```
############################
#      Grupos por datos    #
############################

sai:
  name: SAI Riello
  view: no
  control: hidden
  entities:
    - sensor.ups_status
    - sensor.nut_ups_nominal_power
    - sensor.nut_ups_nominal_real_power
    - sensor.nut_ups_input_voltage
    - sensor.nut_ups_battery_voltage

homeassistant:
  name: Home Assistant
  view: no
  control: hidden
  entities:
    - sensor.version_actual_instalada
    - sensor.ha_update_available
    - sensor.ultima_version_disponible
    - sensor.ha_uptime_templated
    - sensor.ssl_cert_expiry

internet:
  name: Internet
  view: no
  control: hidden
```

```
  entities:
    - sensor.speedtest_ping
    - sensor.speedtest_download
    - sensor.speedtest_upload
sistema:
  name: Sistema
  view: no
  control: hidden
  entities:
    - sensor.cpu_temperatura
    - sensor.processor_use
    - sensor.memory_free
    - sensor.memory_use
    - sensor.memory_use_percent
    - sensor.disk_free_
    - sensor.disk_use_
    - sensor.disk_use_percent_
    - sensor.last_boot
    - sensor.since_last_boot_templated
baterias:
  name: Estado batería sensores
  view: no
  control: hidden
  entities:
    - sensor.movimiento_battery
    - sensor.switch_battery
    - sensor.salon_battery
    - sensor.cuarto_chico_battery
    - sensor.dormitorio_battery
    - sensor.exterior_battery
```



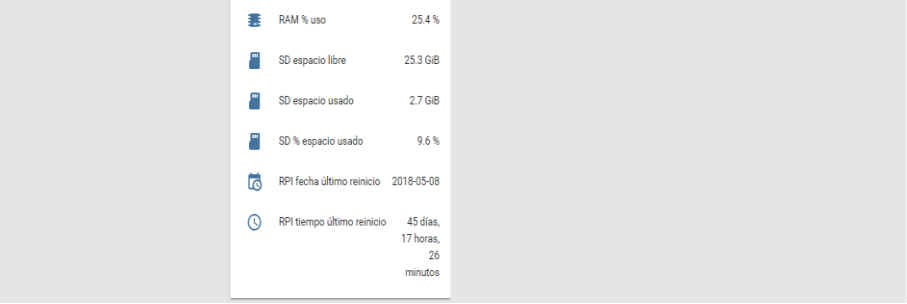


Figura 5-68. Pestaña con información general del sistema.

```
##############################
#     Grupos sobre el tiempo       #
##############################

info_tiempo:
  name: Otra información
  view: no
  control: hidden
  entities:
    - sensor.date_long
    - sensor.estacion
    - sensor.sunrise
    - sensor.sunset
    - sensor.moon_phase

tiempo_actual:
  name: Tiempo actual
  view: no
  control: hidden
  entities:
    - sensor.pws_weather
    - sensor.pws_temp_c
    - sensor.pws_temp_high_1d_c
    - sensor.pws_temp_low_1d_c
    - sensor.pws_relative_humidity
    - sensor.pws_precip_today_metric
    - sensor.pws_precip_1d
    - sensor.pws_wind_dir
    - sensor.pws_wind_kph
    - sensor.pws_pressure_mb
    - sensor.pws_UV
    - sensor.pws_alerts

historico:
  name: Histórico de temperaturas
  view: no
  control: hidden
  entities:
    - sensor.pws_temp_high_record_c
    - sensor.pws_temp_low_record_c
    - sensor.pws_temp_high_avg_c
    - sensor.pws_temp_low_avg_c

mapas:
  name: Mapas
  view: no
```

```
  control: hidden
  entities:
    - camera.mapa_tiempo
prediccion:
  name: Predicción próximos días
  view: no
  control: hidden
  entities:
    - sensor.pws_weather_1d_metric
    - sensor.pws_weather_1n_metric
    - sensor.pws_weather_2d_metric
    - sensor.pws_weather_2n_metric
    - sensor.pws_weather_3d_metric
    - sensor.pws_weather_3n_metric
    - sensor.pws_weather_4d_metric
    - sensor.pws_weather_4n_metric
```

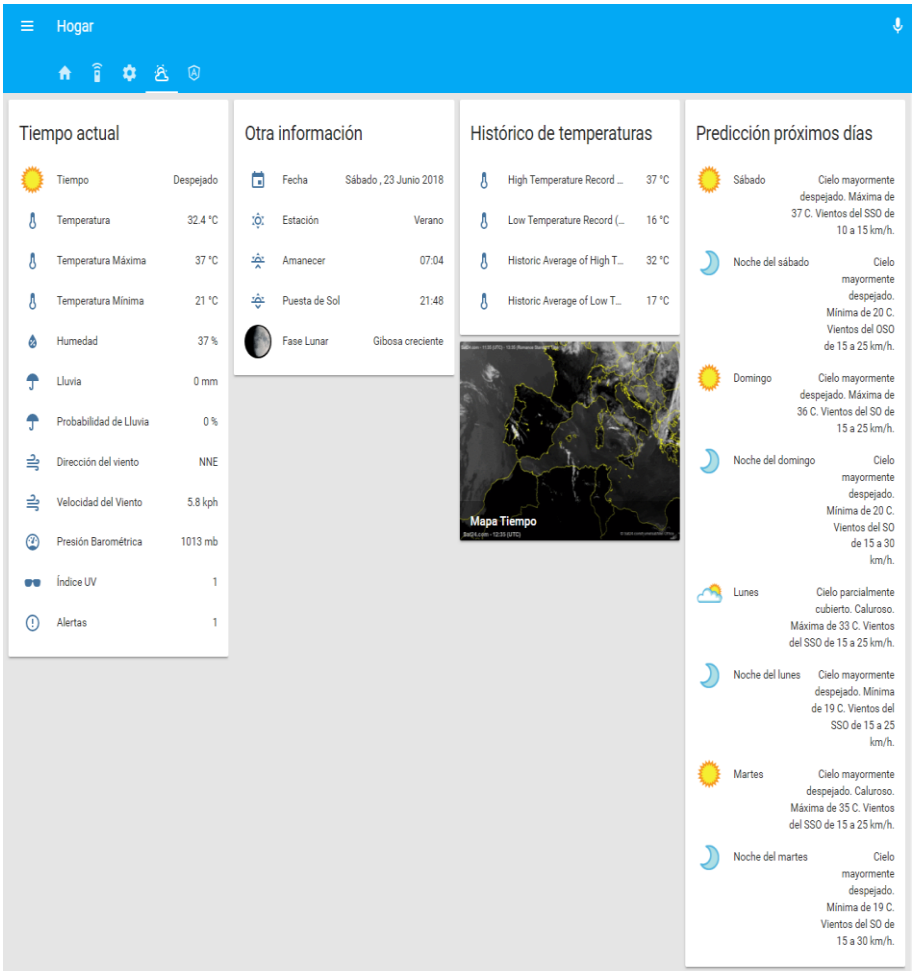


Figura 5-69. Pestaña sobre el tiempo.

```
################################
#   Grupos por automatizaciones   #
################################

autos:
  name: Automatizaciones
```

view: no
control: hidden
entities:
- input_boolean.dia_fiesta
- automation.activar_modo_fuera_casa
- automation.desactivar_modo_fuera_casa
- automation.luz_movimiento_entrada
- automation.activar_modo_vacaciones
- automation.desactivar_modo_vacaciones
- automation.modo_vacaciones
- automation.encendido_gateway_por_la_manana
- automation.encendido_calientacamas
- automation.activar_desactivar_alarma
- automation.sirena1
- automation.sirena2
- automation.battery_alert

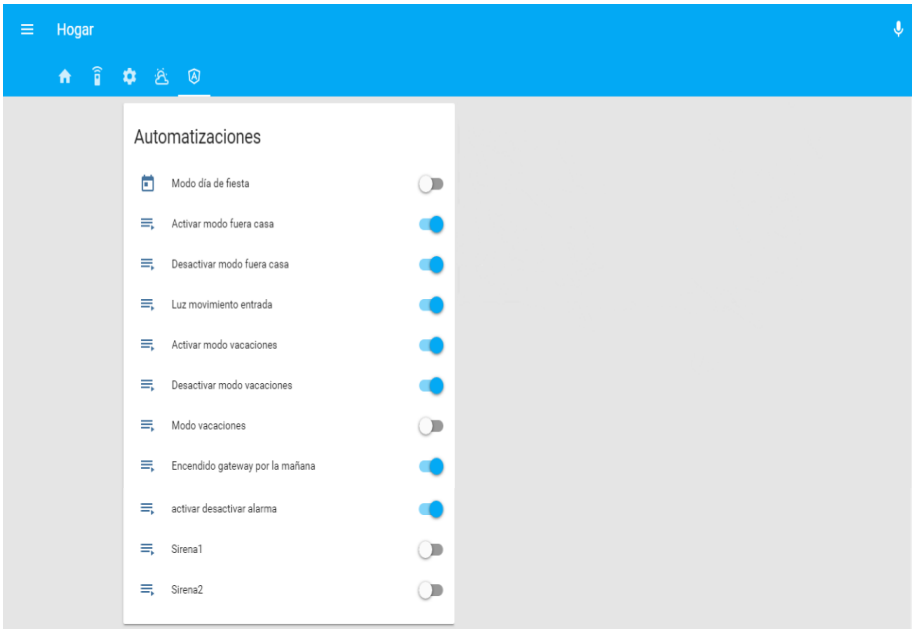


Figura 5-70. Pestaña de automatizaciones.

PRESUPUESTO

Tabla 0–1. Presupuesto empleado para el sistema domótico

Hardware utilizado	Tienda	Cantidad	Importe	Total
Rasberry Pi 3 + alimentación + carcasa + disipadores	Amazon	1	55 €	55 €
Tarjeta micro SD clase 10 de 32 GB SanDisk	Amazon	2	12 €	24 €
Regleta 4 tomas con conexión C14 IEC320 para SAI	Amazon	2	11 €	22 €
Cámara IP Wireless HD, modelo C1 v3 Foscam	Amazon	1	60 €	60 €
Enchufes adaptadores UE-China	AliExpress	3	2 €	6 €
Gateway Xiaomi	AliExpress	1	23 €	23 €
Sensor Temperatura y Humedad Xiaomi	AliExpress	3	8 €	27 €
Sensor Temperatura, Humedad y Presión Xiaomi	Gearbest	1	7 €	7 €
Sensor movimiento Xiaomi	AliExpress	1	9 €	9 €
Pulsador Wireless Xiaomi	Gearbest	1	6 €	6 €
Bombilla Wifi RGB Yeelight	Gearbest	1	12 €	12 €
Controlador universal IR Broadlink	Banggood	1	12 €	12 €
Enchufe Wifi SP3 Broadlink	AliExpress	1	17 €	17 €
Regleta Wifi MP1 Broadlink	AliExpress	1	21 €	21 €
Tableta Lenovo Tab 4 de 8 pulgadas	Fnac	1	100 €	100 €
Robot de limpieza Mi Vacuum v1 Xiaomi	Gearbest	1	250 €	250 €
Cable Ethernet	-	1	0 €	0 €
Cable USB tipo A-B para comunicación SAI	-	1	0 €	0 €
Conector IEC hembra a SCHUKO alimentación SAI	-	1	0 €	0 €
Portátil personal Toshiba	-	1	0 €	0 €
Dispositivo móvil iPhone 6s	-	1	0 €	0 €
Sistema de climatización centralizado Acson	-	1	0 €	0 €
Router Sercomm H500-S Vodafone	-	1	0 €	0 €
TV Samsung modelo UE40D6500	-	1	0 €	0 €
Total				651 €

La tableta de 8 pulgadas que se intalará en la pared para su uso como controlador del sistema desde dentro de casa es opcional. El robot de limpieza sí se ha tenido en cuenta ya que es un dispositivo importante en este sistema domótico. Otros productos no se han tenido en cuenta su precio por disponer de ellos y no tenerse que comprar.

CONCLUSIONES

Una vez realizado este trabajo se puede afirmar y confirmar que la posibilidad de realizar un sistema dómotico es una opción real y muy interesante para los hogares de hoy en día. El dinero invertido es moderadamente bajo respecto a otras opciones del mercado, pero lo más importante es que con este tipo de sistemas instalados se pueden establecer hábitos para crear casas prácticas y eficientes desde el punto de vista energético.

Si bien a priori puede parecer relativamente sencillo de implementar, realmente tiene cierta complejidad ya que se se necesitan nociones de informática, sobretodo para intentar resolver los problemas que puedan surgir durante las instalaciones y configuraciones. Se tocan varios sistemas operativos, aunque con diferencia Linux se convierte en la parte más compleja si nunca se ha utilizado.

La plataforma Home Assistant aporta un abanico de posibilidades muy grande de personalización del sistema, se puede hacer casi cualquier cosa. La comunidad que hay detrás es grande y sigue creciendo cada vez más, y el soporte oficial es continuo en cuanto a actualizaciones e incorporación de nuevos componentes. Es la opción perfecta entre las plataformas existentes, tiene un presente y un futuro muy prometedor.

El único punto que habría que dejar un poco en el aire sería el de la seguridad. A pesar de que se han establecido medidas para intentar que el sistema sea lo más robusto e inaccesible para los piratas informáticos, realmente no se sabe lo fácil o difícil que sería el intentar tomar el control o sacar información personal. Este es un tema delicado incluso para las grandes empresas que poseen carísimos sistemas de seguridad.

BIBLIOGRAFÍA

• Web oficial de Home Assistant:

https://www.home-assistant.io/

• Foro oficial de Home Assistant:

https://community.home-assistant.io/

• Grupo de Telegram en Español:

https://t.me/Domology

• Buscador general para contenidos:

https://www.google.com/

• Configuración general:

https://www.home-assistant.io/docs/scripts/service-calls/

https://www.home-assistant.io/docs/configuration/customizing-devices/

https://www.home-assistant.io/docs/configuration/splitting_configuration/

https://www.home-assistant.io/docs/automation/

https://www.home-assistant.io/docs/scripts/

• Sensores:

https://www.home-assistant.io/components/sensor.template/

https://www.home-assistant.io/components/sensor.command_line/

https://www.home-assistant.io/components/sensor.systemmonitor/

https://www.home-assistant.io/components/sensor.scrape/

https://www.home-assistant.io/components/sensor.version/

https://www.home-assistant.io/components/sensor.speedtest/

https://www.home-assistant.io/components/sensor.wunderground/

https://www.home-assistant.io/components/sensor.season/

https://www.home-assistant.io/components/sensor.moon/

https://www.home-assistant.io/components/sun/

• Interruptores:

https://www.home-assistant.io/components/switch/

https://www.home-assistant.io/components/switch.template/

• Cámara:

https://www.home-assistant.io/components/camera.generic/

• Variables input:

https://www.home-assistant.io/components/input_number/

https://www.home-assistant.io/components/input_select/

https://www.home-assistant.io/components/input_boolean/

• Contador:

https://www.home-assistant.io/components/counter/

• Iconos:

https://cdn.materialdesignicons.com/2.3.54/

• Paneles laterales:

https://www.home-assistant.io/components/panel_iframe/

• Configuración inicial y scripts de HA:

http://www.richa1.com/RichardAlbritton/installing-home-assistant-with-hassbian/

• DuckDNS y LetsEncrypt:

https://www.splitbrain.org/blog/2017-08/10-homeassistant_duckdns_letsencrypt

http://sanzperez.com/alfredo/duckdns-un-sencillo-gestor-de-dns-dinamicas/

https://www.home-assistant.io/docs/ecosystem/certificates/lets_encrypt/

• Componentes Xiaomi:

https://diyprojects.io/installing-including-xiaomi-smart-home-kit-home-assistant-hass/#comments

https://github.com/lazcad/homeassistant

https://www.home-assistant.io/components/xiaomi_aqara/

• Bombilla Yeelight:

https://getyeti.co/posts/how-to-control-yeelight-and-your-smarthome-with-yeti

https://www.home-assistant.io/components/light.yeelight/

• Broadlink:

https://www.home-assistant.io/components/switch.broadlink/

• TV Samsung:

https://www.home-assistant.io/components/media_player.samsungtv/

• Robot de limpieza:

https://www.home-assistant.io/components/vacuum.xiaomi_miio/

https://npirtube.com/tutorial-cambio-voces-xiaomi-vacuum-roborock-vacuum-cleaner/

https://python-miio.readthedocs.io/en/latest/index.html

• SAI:

https://www.home-assistant.io/components/sensor.nut/

http://trasteandoarduino.com/2016/12/18/configurar-un-sai-ups-en-linux/

http://akirasan.net/control-y-monitorizacion-de-un-ups-con-nut/

https://todosai.com/blog/como-elegir-un-sai-b46.html

https://networkupstools.org/

• Tracker Life360:

http://domology.es/localizacion-de-moviles-life360/

https://www.home-assistant.io/components/device_tracker/

https://www.home-assistant.io/components/device_tracker.nmap_tracker/

https://www.home-assistant.io/components/zone/

• Cámara Foscam:

https://www.home-assistant.io/components/camera.foscam/

• Sistema de alarma:

https://github.com/gazoscalvertos/Hass-Custom-Alarm

http://domology.es/sistema-alarma-hass-xiaomi-gateway/

https://www.home-assistant.io/components/alarm_control_panel.manual/

• Notificaciones:

http://domology.es/integracion-telegram-home-assistant/

https://www.home-assistant.io/components/notify.telegram/

https://www.home-assistant.io/components/notify/

https://www.home-assistant.io/docs/ecosystem/ios/notifications/basic/

https://www.home-assistant.io/docs/ecosystem/ios/

https://www.home-assistant.io/components/telegram_bot/

https://joshmccarty.com/install-configure-home-assistant-ios-app-enable-notifications/

https://networkhobo.com/2018/01/11/using-actionable-notifications-in-home-assistant/

• Sistemas domóticos:

https://ricveal.com/blog/

• Curso: Internet of Things (IoT) por Tknika formación:

https://www.youtube.com/playlist?list=PLOYSs5_FlYNum9JrVX1DFkzbCaSu2TXKU

ANEXO A

A.1. Contenido del archivo hook.sh:

```
#!/usr/bin/env bash
set -e
set -u
set -o pipefail

domain="micasa"
token=" bd7d7abc-ac3c-40a2-9026-c689ebr90c92"

case "$1" in
  "deploy_challenge")
    curl "https://www.duckdns.org/update?domains=$domain&token=$token&txt=$4"
    echo
    ;;
  "clean_challenge")
    curl
"https://www.duckdns.org/update?domains=$domain&token=$token&txt=removed&clear=true"
    echo
    ;;
  "deploy_cert")
    sudo systemctl restart home-assistant@homeassistant.service
    ;;
  "unchanged_cert")
    ;;
  "startup_hook")
    ;;
  "exit_hook")
    ;;
  *)
    echo Unknown hook "${1}"
    exit 0
    ;;
esac
```

A.2. Resto del código para configurar el emisor infrarrojos de Broadlink en HA:

```
    aire_subir_temp:
     friendly_name: "Subir Temperatura"
     command_on:
'JgBQAAABKJIRORAUEBQSEhISExETERMRFBETNhM2EDkQORA5EDkQOg8VEDgSEhI3EzYTE
RMREBQQORAUEDkQFBAUEDoQOBI3EgAGbAABJkwQAA0FAAAAAAAAAAA=='
    aire_bajar_temp:
     friendly_name: "Bajar Temperatura"
     command_on:
'JgBQAAABJpYROBISEBQQFBAUEBQQFQ8VEBMSNxM2EzYTNhA5EDkQORA5EDkQFRAUETcS
ExIRExETERAUEDkQORAUEDkQORA6DwAGbwABJUwQAA0FAAAAAAAAAAA=='
    aire_ventilador:
     friendly_name: "Ventilador"
     command_on:
'JgBQAAABJZYQORAUEBUQFBESEhITERMRExEQORA5EDkQORA7EDoPOhA4EjcSEhM2EzYTE
```

```
RMSDxQQFBAWDzkQFBAUEDoPORE4EgAGbQABJUwQAA0FAAAAAAAAAA=='
    aire_modo:
      friendly_name: "Modo"
      command_on:
'JgBQAAABKJMTNhMRExIPFBAUEBQQFBAUEBQQORE4EjcTNhM2EzYTNhAUEBQQFBAUEDk
QFQ8VEBQRNxI3EzYTNhMSETgSNxM2EwAGbAABJU0PAA0FAAAAAAAAAA=='
```

A.3. Resto del código con los estados del sistema de alarma para su configuración:

```
# Estado de la alarma cuando pasa a activada en modo noche.

- id: alarm_armed_home
  alias: '[Alarm] Home Mode Armed'
  trigger:
    - platform: state
      entity_id: alarm_control_panel.house
      to: 'armed_home'
  action:
    - service: notify.ios_ulephone
      data:
        message: "¡Alarma de casa activada! *Estado: armed_home"


# Estado de la alarma cuando se desarma.
# La secuencia será: parar el efecto de luz de la bombilla y apagarla (si estaba
# encendida), desactivar las dos automatizaciones de la sirena, parar el sonido
# (si estaba sonando), activar sonido en el Gateway para saber que la alarma se ha
# desactivado.

- id: alarm_disarmed
  alias: '[Alarm] Disarmed'
  trigger:
    - platform: state
      entity_id: alarm_control_panel.house
      to: 'disarmed'
  action:
    - service: light.turn_on
      data:
        entity_id: light.yeelight_rgb_7811dc6840bc
        effect: 'Stop'
    - service: light.turn_off
      entity_id: light.yeelight_rgb_7811dc6840bc
    - service: automation.turn_off
      entity_id: automation.sirena1
    - service: automation.turn_off
      entity_id: automation.sirena2
    - delay:
        seconds: 1
    - service: xiaomi_aqara.stop_ringtone
      data:
        gw_mac: 38:6C:07:77:8A:C1
    - service: xiaomi_aqara.play_ringtone
      data:
        gw_mac: 38:6C:07:77:8A:C1
        ringtone_id: 12
        ringtone_vol: 20
```

```
    - service: notify.ios_ulephone
      data:
        message: "¡Alarma desactivada! *Estado: disarmed"


# Estado de la alarma cuando pasa a activada.
# Se activa la automatización sirena1 para el sonido de la alarma.

- id: alarm_triggered
  alias: '[Alarm] Triggered'
  trigger:
    - platform: state
      entity_id: alarm_control_panel.house
      to: 'triggered'
  action:
    - service: automation.turn_on
      entity_id: automation.sirena1
    - service: notify.ios_ulephone
      data:
        message: '¡¡Atención!! ¡¡Alguien ha entrado en casa!! *Estado: triggered. {{
states[states.alarm_control_panel.house.attributes.changed_by.split(".")[0]][
states.alarm_control_panel.house.attributes.changed_by.split(".")[1]].name }}'


# Estado de la alarma cuando pasa a advertencia de activación de alarma.

- id: alarm_warning
  alias: '[Alarm] Warning'
  trigger:
    - platform: state
      entity_id: alarm_control_panel.house
      to: 'warning'
  action:
    - service: notify.ios_ulephone
      data:
        message: "¡Se ha disparado la alarma de casa! *Estado: warning"


########################################
############ S I R E N A ###############
########################################

# Estas dos automatizaciones sirven para hacer sonar el Gateway con una sirena de
# Policía e iluminar la bombilla mediante un efecto de luz azul y roja de Policía cuando
# la alarma pasa a activada.

- alias: "Sirena1"
  initial_state: False
  #hide_entity: False
  trigger:
    - platform: state
      entity_id: automation.sirena1
      from: 'off'
      to: 'on'
  condition:
    - condition: state
      entity_id: alarm_control_panel.house
      state: 'triggered'
  action:
```

```
    - service: automation.turn_off
      entity_id: automation.sirena2
    - service: xiaomi_aqara.play_ringtone
      data:
        gw_mac: 38:6C:07:77:8A:C1
        ringtone_id: 2
        ringtone_vol: 80
    - service: light.turn_on
      data:
        entity_id: light.yeelight_rgb_7811dc6840bc
        effect: 'Police'
    - delay:
        seconds: 7
    - service: automation.turn_on
      entity_id: automation.sirena2
```

 # Esta segunda automatización sirve para hacer un bucle hasta que se desactive la
 # alarma. El retardo de 7 segundos coincide con la duración del sonido para que se
 # inicie cada vez justo cuando termine.

```
- alias: "Sirena2"
  initial_state: False
  #hide_entity: False
  trigger:
    - platform: state
      entity_id: automation.sirena2
      from: 'off'
      to: 'on'
  condition:
    - condition: state
      entity_id: alarm_control_panel.house
      state: 'triggered'
  action:
    - service: automation.turn_off
      entity_id: automation.sirena1
    - service: xiaomi_aqara.play_ringtone
      data:
        gw_mac: 38:6C:07:77:8A:C1
        ringtone_id: 2
        ringtone_vol: 80
    - service: light.turn_on
      data:
        entity_id: light.yeelight_rgb_7811dc6840bc
        effect: 'Police'
    - delay:
        seconds: 7
    - service: automation.turn_on
      entity_id: automation.sirena1
```

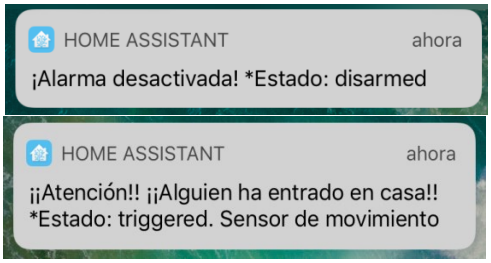


Figura 0-1. Notificaciones en el sistema de alarma.

A.4. Resto del código para la automatización de los mandos de la climatización:

```
# Cambio velocidad del ventilador
- alias: Cambio velocidad ventilador
  trigger:
  - platform: state
    entity_id: switch.ventilador
    from: 'off'
    to: 'on'
  condition:
  - condition: state
    entity_id: switch.encendido
    state: 'on'
  action:
  - service: input_select.select_next
    data:
      entity_id: input_select.ac_fan_mode
# Cambio modo de funcionamiento
- alias: Cambio modo funcionamiento
  trigger:
  - platform: state
    entity_id: switch.modo
    from: 'off'
    to: 'on'
  condition:
  - condition: state
    entity_id: switch.encendido
    state: 'on'
  action:
  - service: input_select.select_previous
    data:
      entity_id: input_select.ac_operation_mode
```

A.5. Resto del código para los scripts de los mandos de la climatización:

```
aire_bajar_temp:
  sequence:
    - service: switch.turn_on
      entity_id: switch.bajar_temperatura
    - service: switch.turn_off
      entity_id: switch.bajar_temperatura
aire_subir_temp:
  sequence:
    - service: switch.turn_on
      entity_id: switch.subir_temperatura
    - service: switch.turn_off
      entity_id: switch.subir_temperatura
aire_ventilador:
  sequence:
    - service: switch.turn_on
      entity_id: switch.ventilador
```

```
  - service: switch.turn_off
    entity_id: switch.ventilador
```

A.6. Código principal del programador para el robot de limpieza:

```
### Programa 1 ###
# Automatismo para el encendido por programación

- alias: Programa 1 ON
  trigger:
    platform: time
    # Comprobación recurrente al inicio de cada minuto.
    minutes: '/1'
    seconds: 00
  condition:
    condition: and
    conditions:
      # AND 1 (si es la hora)
      - condition: template
        value_template: '{{ (now().strftime("%H"))|int == (states.input_number.hora_on_p1.state)|int }}'
      # AND 1 (si son los minutos)
      - condition: template
        value_template: '{{ (now().strftime("%M"))|int == (states.input_number.minutos_on_p1.state)|int }}'
      # AND 3 (si está activado el encendido por programación)
      - condition: state
        entity_id: input_boolean.estado_on_p1
        state: 'on'
      # AND 4 (si está activado entresemana/findes o solo una vez)
      - condition: or
        conditions:
          # AND 4 - OR 1 (entresemana)
          - condition: and
            conditions:
              # AND 4 - OR 1 - AND 1 (si está activado entresemana)
              - condition: state
                entity_id: input_boolean.entresemana_p1
                state: 'on'
              # AND 4 - OR 1 - AND 2 (si hoy es un dia de entresemana)
              - condition: time
                weekday:
                  - mon
                  - tue
                  - wed
                  - thu
                  - fri
          # AND 4 - OR 2 (finde)
          - condition: and
            conditions:
              # AND 4 - OR 2 - AND 1 (si está activado finde)
              - condition: state
                entity_id: input_boolean.finde_p1
                state: 'on'
              # AND 4 - OR 2 - AND 2 (si hoy es un día del finde)
              - condition: time
                weekday:
                  - sat
```

```
        - sun
    # AND 4 - OR 3 (solo una sola vez)
    - condition: and
      conditions:
        # AND 4 - OR 3 - AND 1 (si NO está activado entresemana)
        - condition: state
          entity_id: input_boolean.entresemana_p1
          state: 'off'
        # AND 4 - OR 3 - AND 2 (si NO está activado finde)
        - condition: state
          entity_id: input_boolean.finde_p1
          state: 'off'
  action:
    # Aquí se introduce el robot sobre el que actúa la programación
    - service: vacuum.turn_on
      entity_id: vacuum.ameba
    #
    - delay: 00:00:05
    # Comprueba si no está activo "entresemana"
    - condition: state
      entity_id: input_boolean.entresemana_p1
      state: 'off'
    # Comprueba si no está activo "finde"
    - condition: state
      entity_id: input_boolean.finde_p1
      state: 'off'
    # Si llega hasta aquí es el caso de SOLO UNA VEZ y desactiva el encendido por programación
    - service: input_boolean.turn_off
      entity_id: input_boolean.estado_on_p1
```

A.7. Personalización de los nombres e iconos de los sensores:

```
# Sensores de temperatura y humedad, y también uno de presión atmosférica

sensor.temperature_158d0001825324:
  friendly_name: Temperatura
sensor.humidity_158d0001825324:
  friendly_name: Humedad
  icon: mdi:water-percent
sensor.temperature_158d000158599d:
  friendly_name: Temperatura
sensor.humidity_158d000158599d:
  friendly_name: Humedad
  icon: mdi:water-percent
sensor.temperature_158d0001f52cca:
  friendly_name: Temperatura
sensor.humidity_158d0001f52cca:
  friendly_name: Humedad
  icon: mdi:water-percent
sensor.pressure_158d0001f52cca:
  friendly_name: Presión atmosférica
  icon: mdi:cloud-outline
```

A.8. Resto de código para conocer el estado de las baterías de los sensores:

```
- platform: template
```

```
 sensors:
   switch_battery:
     friendly_name: 'Batería pulsador'
     value_template: >
        {%- if states.binary_sensor.switch_158d0001b12b32.attributes.battery_level %}
           {{ states.binary_sensor.switch_158d0001b12b32.attributes.battery_level|round }}
        {% else %}
           {{ states.binary_sensor.switch_158d0001b12b32.state }}
        {%- endif %}
     icon_template: '{%- if  states.binary_sensor.switch_158d0001b12b32.attributes.battery_level <= 5
%}mdi:battery-outline{%- elif states.binary_sensor.switch_158d0001b12b32.attributes.battery_level >= 95
%}mdi:battery{% else %}mdi:battery-
{{(states.binary_sensor.switch_158d0001b12b32.attributes.battery_level|float / 10)|round*10}}{%- endif %}'
     unit_of_measurement: "%"


- platform: template
  sensors:
    salon_battery:
      friendly_name: 'Batería sensor salón'
      value_template: >
        {%- if states.sensor.temperature_158d000159c165.attributes.battery_level %}
           {{ states.sensor.temperature_158d000159c165.attributes.battery_level|round }}
        {% else %}
           {{ states.sensor.temperature_158d000159c165.state }}
        {%- endif %}
      icon_template: '{%- if  states.sensor.temperature_158d000159c165.attributes.battery_level <= 5
%}mdi:battery-outline{%- elif states.sensor.temperature_158d000159c165.attributes.battery_level >= 95
%}mdi:battery{% else %}mdi:battery-
{{(states.sensor.temperature_158d000159c165.attributes.battery_level|float / 10)|round*10}}{%- endif %}'
      unit_of_measurement: "%"


- platform: template
  sensors:
    cuarto_chico_battery:
      friendly_name: 'Batería sensor cuarto chico'
      value_template: >
        {%- if states.sensor.temperature_158d0001825324.attributes.battery_level %}
           {{ states.sensor.temperature_158d0001825324.attributes.battery_level|round }}
        {% else %}
           {{ states.sensor.temperature_158d0001825324.state }}
        {%- endif %}
      icon_template: '{%- if  states.sensor.temperature_158d0001825324.attributes.battery_level <= 5
%}mdi:battery-outline{%- elif states.sensor.temperature_158d0001825324.attributes.battery_level >= 95
%}mdi:battery{% else %}mdi:battery-
{{(states.sensor.temperature_158d0001825324.attributes.battery_level|float / 10)|round*10}}{%- endif %}'
      unit_of_measurement: "%"


- platform: template
  sensors:
    dormitorio_battery:
      friendly_name: 'Batería sensor dormitorio'
      value_template: >
        {%- if states.sensor.temperature_158d000158599d.attributes.battery_level %}
```

```
        {{ states.sensor.temperature_158d000158599d.attributes.battery_level|round }}
      {% else %}
        {{ states.sensor.temperature_158d000158599d.state }}
      {%- endif %}
    icon_template: '{%- if  states.sensor.temperature_158d000158599d.attributes.battery_level <= 5
%}mdi:battery-outline{%- elif states.sensor.temperature_158d000158599d.attributes.battery_level >= 95
%}mdi:battery{% else %}mdi:battery-
{{(states.sensor.temperature_158d000158599d.attributes.battery_level|float / 10)|round*10}}{%- endif %}'
    unit_of_measurement: "%"


- platform: template
  sensors:
    exterior_battery:
      friendly_name: 'Batería sensor exterior'
      value_template: >
        {%- if states.sensor.temperature_158d0001f52cca.attributes.battery_level %}
          {{ states.sensor.temperature_158d0001f52cca.attributes.battery_level|round }}
        {% else %}
          {{ states.sensor.temperature_158d0001f52cca.state }}
        {%- endif %}
      icon_template: '{%- if  states.sensor.temperature_158d0001f52cca.attributes.battery_level <= 5
%}mdi:battery-outline{%- elif states.sensor.temperature_158d0001f52cca.attributes.battery_level >= 95
%}mdi:battery{% else %}mdi:battery-
{{(states.sensor.temperature_158d0001f52cca.attributes.battery_level|float / 10)|round*10}}{%- endif %}'
      unit_of_measurement: "%"
```

A.9. Código para conocer el estado de la batería del otro dispositivo móvil:

```
- platform: template
  sensors:
    battery_jade_v:
      friendly_name: 'Batería móvil Jade_V'
      unit_of_measurement: '%'
      value_template: >
        {%- if states.device_tracker.jade_v_d33ba49fdb064b0bb0371ce3c11bd5bd.attributes.battery %}
          {{ states.device_tracker.jade_v_d33ba49fdb064b0bb0371ce3c11bd5bd.attributes.battery|round }}
        {% else %}
          {{ states.sensor.battery_jade_v.state }}
        {%- endif %}
      icon_template: >
        {%- if is_state("sensor.battery_jade_v", "unknown") %}
          mdi:battery-unknown
        {%- elif is_state_attr('device_tracker.jade_v_d33ba49fdb064b0bb0371ce3c11bd5bd', 'battery_status',
'Charging') %}
          mdi:battery-charging
        {%- elif states.device_tracker.jade_v_d33ba49fdb064b0bb0371ce3c11bd5bd.attributes.battery <= 5 %}
          mdi:battery-outline
        {%- elif states.device_tracker.jade_v_d33ba49fdb064b0bb0371ce3c11bd5bd.attributes.battery >= 95 %}
          mdi:battery
        {% else %}
          mdi:battery-
{{(states.device_tracker.jade_v_d33ba49fdb064b0bb0371ce3c11bd5bd.attributes.battery|float /
10)|round*10}}
        {%- endif %}
```

A.10. Resto de notificaciones para conocer la ubicación de los miembros de la familia:

```
- alias: 'Aviso cambio estado jade home'
  trigger:
    platform: state
    entity_id: device_tracker.jade_v_d23ba49fdb064b0bb0071ce3c11bd5bd
    from: 'not_home'
    to: 'home'
  action:
    - service: notify.ios_ulephone
      data:
        message: "La ubicación de Jade_V ha cambiado a {{
states.device_tracker.jade_v_d23ba49fdb064b0bb0071ce3c11bd5bd.state }}"


- alias: 'Aviso cambio estado ule not home'
  trigger:
    platform: state
    entity_id: device_tracker.ule_767ceb627834417f9974559b39c44bd1
    from: 'home'
    to: 'not_home'
  action:
    - service: notify.ios_ulephone
      data:
        message: "La ubicación de uLe ha cambiado a {{
states.device_tracker.ule_767ceb627834417f9974559b39c44bd1.state }}"


- alias: 'Aviso cambio estado jade not home'
  trigger:
    platform: state
    entity_id: device_tracker.jade_v_d23ba49fdb064b0bb0071ce3c11bd5bd
    from: 'home'
    to: 'not_home'
  action:
    - service: notify.ios_ulephone
      data:
        message: "La ubicación de Jade_V ha cambiado a {{
states.device_tracker.jade_v_d23ba49fdb064b0bb0071ce3c11bd5bd.state }}"


- alias: 'Aviso cambio estado familia'
  trigger:
    platform: event
    event_type: state_changed
    event_data:
      entity_id: group.all_devices
  action:
    - service: notify.ios_ulephone
      data:
        message: "La ubicación de la familia ha cambiado a {{ states.group.all_devices.state }}"
```

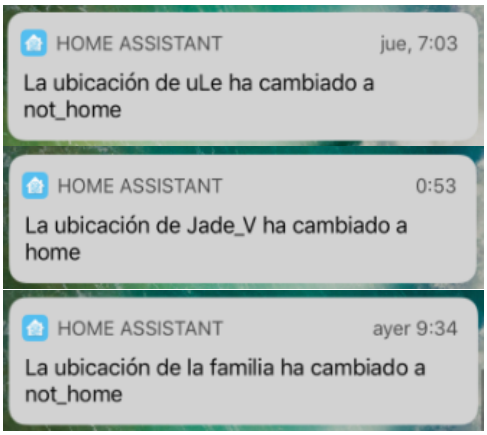


Figura 0-2. Notificaciones para conocer la ubicación.

A.11. Otras notificaciones para conocer el estado de funcionamiento del SAI:

```
- alias: 'Notificación SAI funcionando'
  trigger:
    platform: state
    entity_id: sensor.nut_ups_status_data
    to: 'OB DISCHRG'
  action:
  - service: notify.ios_ulephone
    data:
      message: "¡¡Atención!! ¡¡Se ha ido la luz en casa!!"


- alias: 'Notificación SAI batería baja'
  trigger:
    platform: state
    entity_id: sensor.nut_ups_status_data
    to: 'LB'
  action:
  - service: notify.ios_ulephone
    data:
      message: "¡¡Atención!! ¡¡La batería del SAI está acabándose!!"
```

A.12. Notificaciones para cambio de versiones en Home Assistant:

```
- alias: 'Update Notification'
  trigger:
    platform: state
    entity_id: sensor.ha_version_update
    from: 'No'
    to: 'Sí'
  action:
    service: notify.ios_ulephone
    data:
      message: “Hay una nueva actualización de Home Assistant”


- alias: 'Update Primer Cambio Version'
  trigger:
    platform: state
    entity_id: sensor.ha_update_available
```

```
  from: 'No'
  to: 'Sí'
 action:
  - service: notify.equipo_ha
    data:
      message: "Primer cambio de versión, hay una nueva actualización de Home Assistant"
```

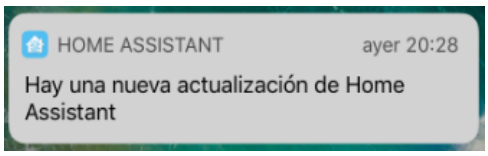


Figura 0-3. Notificación sobre actualización de HA.

A.13. *Template* que modifica el sensor *uptime* para pasar el tiempo de minutos a días, horas y minutos:

```
- platform: template
  sensors:
    ha_uptime_templated:
      friendly_name: 'Tiempo funcionando'
      value_template: >-
        {% set uptime = states.sensor.tiempo_funcionando.state | int %}
        {% set minutes = ((uptime % 60) / 1) | int %}
        {% set hours = ((uptime % 1440) / 60) | int %}
        {% set days = (uptime /1440) | int %}

        {%- if uptime < 1 -%}
          Menos de un minuto
        {%- else -%}
          {%- if days > 0 -%}
            {%- if days == 1 -%}
              1 día
            {%- else -%}
              {{ days }} días
            {%- endif -%}
          {%- endif -%}
          {%- if hours > 0 -%}
            {%- if days > 0 -%}
              {{ ', ' }}
            {%- endif -%}
            {%- if hours == 1 -%}
              1 hora
            {%- else -%}
              {{ hours }} horas
            {%- endif -%}
          {%- endif -%}
          {%- if minutes > 0 -%}
            {%- if days > 0 or hours > 0 -%}
              {{ ', ' }}
            {%- endif -%}
            {%- if minutes == 1 -%}
              1 minuto
            {%- else -%}
              {{ minutes }} minutos
```

```
      {%- endif -%}
    {%- endif -%}
  {%- endif -%}
 icon_template: mdi:timer
```

A.14. *Template* que modifica el sensor *since_lasta_boot* para pasar el tiempo de minutos a días, horas y minutos:

```
- platform: template
  sensors:
    since_last_boot_templated:
      friendly_name: 'RPI tiempo último reinicio'
      value_template: >-
        {%- if states.sensor.since_last_boot -%}
          {%- set slb = states.sensor.since_last_boot.state.split(' ') -%}
          {%- set count = slb | length -%}
          {%- set hms = slb[count - 1] -%}
          {%- set hms_split = hms.split(':') -%}
          {%- set hours = hms_split[0] | int -%}
          {%- set minutes = hms_split[1] | int -%}
          {%- if count == 3 -%}
            {%- set days = slb[0] | int -%}
          {%- else -%}
            {%- set days = 0 -%}
          {%- endif -%}

          {%- if days == 0 and hours == 0 and minutes == 0 -%}
            Menos de un minuto
          {%- else -%}
            {%- if days > 0 -%}
              {%- if days == 1 -%}
                1 día
              {%- else -%}
                {{ days }} días
              {%- endif -%}
            {%- endif -%}
            {%- if hours > 0 -%}
              {%- if days > 0 -%}
                {{ ', ' }}
              {%- endif -%}
              {%- if hours == 1 -%}
                1 hora
              {%- else -%}
                {{ hours }} horas
              {%- endif -%}
            {%- endif -%}
            {%- if minutes > 0 -%}
              {%- if days > 0 or hours > 0 -%}
                {{ ', ' }}
              {%- endif -%}
              {%- if minutes == 1 -%}
                1 minuto
              {%- else -%}
                {{ minutes }} minutos
              {%- endif -%}
            {%- endif -%}
```

```
      {%- endif -%}
    {%- else -%}
      n/a
    {%- endif -%}
  icon_template: mdi:clock
  entity_id: sensor.since_last_boot
```

A.15. Personalización de los sensores que aportan los datos de la Raspberry Pi:

```
sensor.processor_use:
  friendly_name: CPU uso
sensor.cpu_temperatura:
  icon: mdi:memory
sensor.memory_free:
  icon: mdi:chip
  friendly_name: RAM libre
sensor.memory_use:
  icon: mdi:chip
  friendly_name: RAM uso
sensor.memory_use_percent:
  icon: mdi:chip
  friendly_name: RAM % uso
sensor.disk_free_:
  friendly_name: SD espacio libre
  icon: mdi:micro-sd
sensor.disk_use_:
  friendly_name: SD espacio usado
  icon: mdi:micro-sd
sensor.disk_use_percent_:
  friendly_name: SD % espacio usado
  icon: mdi:micro-sd
sensor.last_boot:
  friendly_name: 'RPI fecha último reinicio'
  icon: mdi:calendar-clock
```

A.16. Código completo de los datos sobre meteorología de wunderground:

```
- platform: wunderground
  api_key: 1f548fa1397d50bb
  pws_id: IALSEVIL4
  latitude: 38.3925
  longitude: -7.9940
  lang: SP
  monitored_conditions:
    - alerts
    - feelslike_c
    - heat_index_c
    - location
    - precip_today_metric
    - precip_1d_mm
    - precip_1d
    - pressure_mb
    - relative_humidity
    - station_id
    - temp_c
```

- temp_high_record_c
- temp_low_record_c
- temp_high_avg_c
- temp_low_avg_c
- temp_high_1d_c
- temp_low_1d_c
- UV
- weather
- weather_1h
- wind_dir
- wind_gust_kph
- wind_gust_1d_kph
- wind_kph
- wind_1d_kph
- wind_string
- weather_1d_metric
- weather_1n_metric
- weather_2d_metric
- weather_2n_metric
- weather_3d_metric
- weather_3n_metric
- weather_4d_metric
- weather_4n_metric

A.17. Personalización de nombres de los datos de wunderground:

sensor.pws_precip_1d:
friendly_name: Probabilidad de Lluvia
sensor.pws_temp_high_1d_c:
friendly_name: Temperatura Máxima
sensor.pws_temp_low_1d_c:
friendly_name: Temperatura Mínima
sensor.pws_alerts:
friendly_name: Alertas
sensor.pws_weather:
friendly_name: Tiempo
sensor.pws_uv:
friendly_name: Índice UV
sensor.pws_relative_humidity:
friendly_name: Humedad
sensor.pws_wind_dir:
friendly_name: Dirección del viento
sensor.pws_temp_c:
friendly_name: Temperatura
sensor.pws_feelslike_c:
friendly_name: Sensación Térmica
icon: mdi:account-alert
sensor.pws_heat_index_c:
friendly_name: Temperatura/Humedad
icon: mdi:thermometer-lines
sensor.pws_precip_today_metric:
friendly_name: Lluvia
sensor.pws_pressure_mb:
friendly_name: Presión Barométrica
sensor.pws_wind_kph:
friendly_name: Velocidad del Viento

A.18. Código completo para personalizar los nombres y las imágenes de las fases lunares:

```
- platform: template
  sensors:
    moon_phase:
      friendly_name: 'Fase Lunar'
      value_template: >
        {% if is_state('sensor.moon', 'New moon') %}
          Luna nueva
        {% elif is_state('sensor.moon', 'Waxing crescent') %}
          Luna nueva visible
        {% elif is_state('sensor.moon', 'First quarter') %}
          Cuarto creciente
        {% elif is_state('sensor.moon', 'Waxing gibbous') %}
          Gibosa creciente
        {% elif is_state('sensor.moon', 'Full moon') %}
          Luna llena
        {% elif is_state('sensor.moon', 'Waning gibbous') %}
          Luna gibosa menguante
        {% elif is_state('sensor.moon', 'Last quarter') %}
          Cuarto menguante
        {% elif is_state('sensor.moon', 'Waning crescent') %}
          Luna menguante
        {% endif %}
      entity_picture_template: >-
        {% if is_state('sensor.moon', 'New moon') %}
          /local/nueva.jpg
        {% elif is_state('sensor.moon', 'Waxing crescent') %}
          /local/25-2.jpg
        {% elif is_state('sensor.moon', 'First quarter') %}
          /local/cuarto.jpg
        {% elif is_state('sensor.moon', 'Waxing gibbous') %}
          /local/75-2.jpg
        {% elif is_state('sensor.moon', 'Full moon') %}
          /local/llena.jpg
        {% elif is_state('sensor.moon', 'Waning gibbous') %}
          /local/75.jpg
        {% elif is_state('sensor.moon', 'Last quarter') %}
          /local/cuarto2.jpg
        {% elif is_state('sensor.moon', 'Waning crescent') %}
          /local/25.jpg
        {% endif %}
```

A.19. Códigos para notificaciones de cambio de estado, estación y fases lunares:

```
- alias: 'Cambio de estación'
  trigger:
    platform: event
    event_type: state_changed
    event_data:
      entity_id: sensor.season
  action:
    - service: notify.ios_ulephone
      data:
```

```
      message: "Hemos cambiado de estación a {{ states.sensor.season.state }}"


- alias: 'Cambio de fases lunares'
  trigger:
    platform: event
    event_type: state_changed
    event_data:
      entity_id: sensor.moon
  action:
    - service: notify.ios_ulephone
      data:
        message: "La luna ha cambiado a {{ states.sensor.moon_phase.state }}"
```

A.20. Códigos de notificación accionable para encender el aire cuando alcanza una temperatura máxima de 27ºC:

```
ios:
  push:
    categories:
      - name: Temperatura maxima
        identifier: temperatura_maxima
        actions:
          - identifier: ENCENDER_AIRE
            title: 'Encender el aire'
            activationMode: 'background'
            authenticationRequired: no
            destructive: yes
            behavior: 'default'

# Temperatura máxima: Actionable Push Message
- alias: 'Temperatura máxima'
  trigger:
    platform: numeric_state
    entity_id: sensor.temperature_158d000158599d
    above: 27
  action:
    - service: notify.ios_ulephone
      data:
        message: "La temperatura del dormitorio es demasiado alta ({{
states.sensor.temperature_158d000158599d.state }}ºC), ¿quieres poner el aire?"
        data:
          push:
            badge: 5
            category: "temperatura_minima"

# Temperatura máxima: Actionable Push Response
- alias: 'Push notify action temperatura'
  initial_state: 'on'
  trigger:
    platform: event
    event_type: ios.notification_action_fired
    event_data:
      actionName: ENCENDER_AIRE
  action:
    service: homeassistant.turn_on
    entity_id: switch.encendido
```