Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior de Jaén

SISTEMA DE DOMÓTICA IoT MULTIPLATAFORMA

Alumno: Nicolás Moral de Aguilar

Tutor: Antonio Abarca Álvarez

Dpto: Ingeniería Electrónica y Automática

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don ANTONIO ABARCA ÁLVAREZ , tutor del Proyecto Fin de Carrera titulado: Sistema de Domótica IoT Multiplataforma, que presenta NICOLÁS MORAL DE AGUILAR, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, SEPTIEMBRE de 2019

El alumno:

MORAL DE AGUILAR NICOLAS - 53596327W
Firmado digitalmente por MORAL DE AGUILAR NICOLAS - 53596327W
Fecha: 2019.09.02 14:30:39 +02'00'

NICOLÁS MORAL DE AGUILAR

Los tutores:

ABARCA ALVAREZ ANTONIO - 23786769P
Firmado digitalmente por ABARCA ALVAREZ ANTONIO - 23786769P
Nombre de reconocimiento (DN): c=ES, serialNumber=23786769P, sn=ABARCA ALVAREZ, givenName=ANTONIO, cn=ABARCA ALVAREZ ANTONIO - 23786769P
Fecha: 2019.09.02 11:44:37 +02'00'

ANTONIO ABARCA ÁLVAREZ

A mis padres, por hacerme ser quien soy. Por permitirme crear mi futuro, incluso en momentos en los que ni siquiera era capaz de ver el presente.

A Teresa, por acompañarme siempre, sin excepción. Por enseñarme lo que es la perseverancia con nada más que la suya.

A Antonio, por enseñarme por primera vez un ESP-8266. Por dejarme volar en este proyecto y confiar en mi para que todo llegase a buen puerto.

Índice

1. Introducción

1.1. Motivación

Hace años comencé un proyecto personal para permitir la apertura remota de la puerta de un garaje. Las posibilidades de la domotización siempre me han parecido muy interesantes y comencé a implementar este sistema con los pocos conocimientos que tenía sobre programación para Android, algo de electrónica y un sencillo servidor en Python. El problema que surgió entonces era la inexistencia de un Arduino con conexión a la red con un precio razonable para ampliar las posibilidades de la aplicación. Por aquel entonces, usé directamente una Raspberry Pi 1. Sin embargo, apareció en el mercado un nuevo microcontrolador fabricado por Espressif, el "ESP-8266" que disponía de WiFi y bajo consumo a un precio realmente bajo (actualmente 1€). Programable de la misma forma que el ya archiconocido Arduino, cambió mi percepción de las posibilidades de llevar a cabo la implementación de un sistema domótico IoT que fuese "low cost".

1.2. Propuesta

Aún con el chip "ESP-8266", existen 2 problemas básicamente. El primero es que se necesitan conocimientos de electrónica relativamente avanzados para la conexión de este chip con sensores y actuadores para el sistema, además comprarlos por separado es caro. El segundo es que la aplicación original de mi implementación estaba desarrollada para Android, y quería que fuese controlable en todas las plataformas posibles.

La resolución del segundo problema es relativamente sencilla: Se necesita una aplicación web. Puede encontrarse un *framework* de domótica que tenga una interfaz web para facilitar la implementación del sistema.

La solución al primer problema se vuelve trivial gracias a la multitud de marcas de bajo coste que han aparecido debido a Alexa, Siri y Google Home, ya que ofrecen dispositivos que incluyen el ESP-8266 con el hardware necesario para controlar desde la simple activación de un relé, hasta reguladores lumínicos o sensores realmente completos.

1.3. Objetivos

1.3.1. Bajo Coste

Disponer de un sistema domótico en una vivienda es viable a día de hoy. Sin embargo, es caro, y te hace depender de aplicaciones para los dispositivos de cada marca, que además limitan bastante las posibilidades de programación del sistema.

Con el chip ESP-8266 es posible conseguir que este sistema tenga un precio muy económico y que sea fácilmente implementable en nuevas viviendas e incluso en antiguas. Prácticamente la totalidad de dispositivos domóticos comerciales a nivel usuario hacen uso de este chip, así que debo recalcar que no concibo la implementación de este TFG sin él.

1.3.2. Procesamiento local

El procesamiento local es una de las cosas que más echo en falta en las aplicaciones de domótica actuales. Los sensores y actuadores de un sistema domótico de este tipo transmiten mucha información personal entre ellos y sobre todo hacia un “sistema de procesamiento central”. En caso de usar marcas comerciales, estos sistemas son servidores que disponen de nuestra información personal. En cambio, implementando el intercambio de información de todos nuestros dispositivos a través de un servidor propio y local, aumentamos bastante la privacidad del sistema.

1.3.3. Seguridad y tolerancia a fallos

Un sistema domótico debe ser seguro y contemplar los posibles escenarios de fallo que puedan suceder sea debido a factores internos del sistema como debido a factores externos (caídas de la red eléctrica o problemas de conectividad en la red). Se implementan en este trabajo desde conexiones HTTPS hasta estrategias de “*hardening*“ para sistema operativo en el que reside e incluso un proxy inverso. Todos estos procedimientos se explican detalladamente en el apartado 7, relativo a la seguridad del sistema. Además de esto, la resistencia a posibles fallos se ha considerado estableciendo reglas y programaciones independientes del sistema central en cada uno de los dispositivos conectados, evitando que se produzcan errores que puedan dejar bloqueados los dispositivos. De esta manera, estos dispositivos funcionan de forma independiente por sí solos, con un menor grado de inteligencia artificial, pero sin dejar de atender la funcionalidad que desempeñan.

1.3.4. Curva de aprendizaje mínima

Cuando implementas un sistema de estas características, al elegir un *framework* desde el que empezar, es conveniente hacerlo de tal forma que no necesites mucho tiempo en aprender a manejarlo. Se hablará de los diferentes *framework*s que he considerado, sin embargo, no solo debe de tenerse en cuenta a nivel del desarrollador, si no a nivel del futuro usuario que hará uso del sistema. De esta forma, Home Assistant, ofrece tanto facilidad para el desarrollador, como una interfaz realmente bonita para los usuarios finales que puede modificarse fácilmente incluso desde sí misma.

1.4. Estimación de costes

1.4.1. Temporal

En este diagrama de Gantt se refleja la asignación temporal de tareas durante la realización del proyecto. Se ha diseñado con un software especializado para la gestión de tiempos en proyectos se llama “OmniGraffle”.

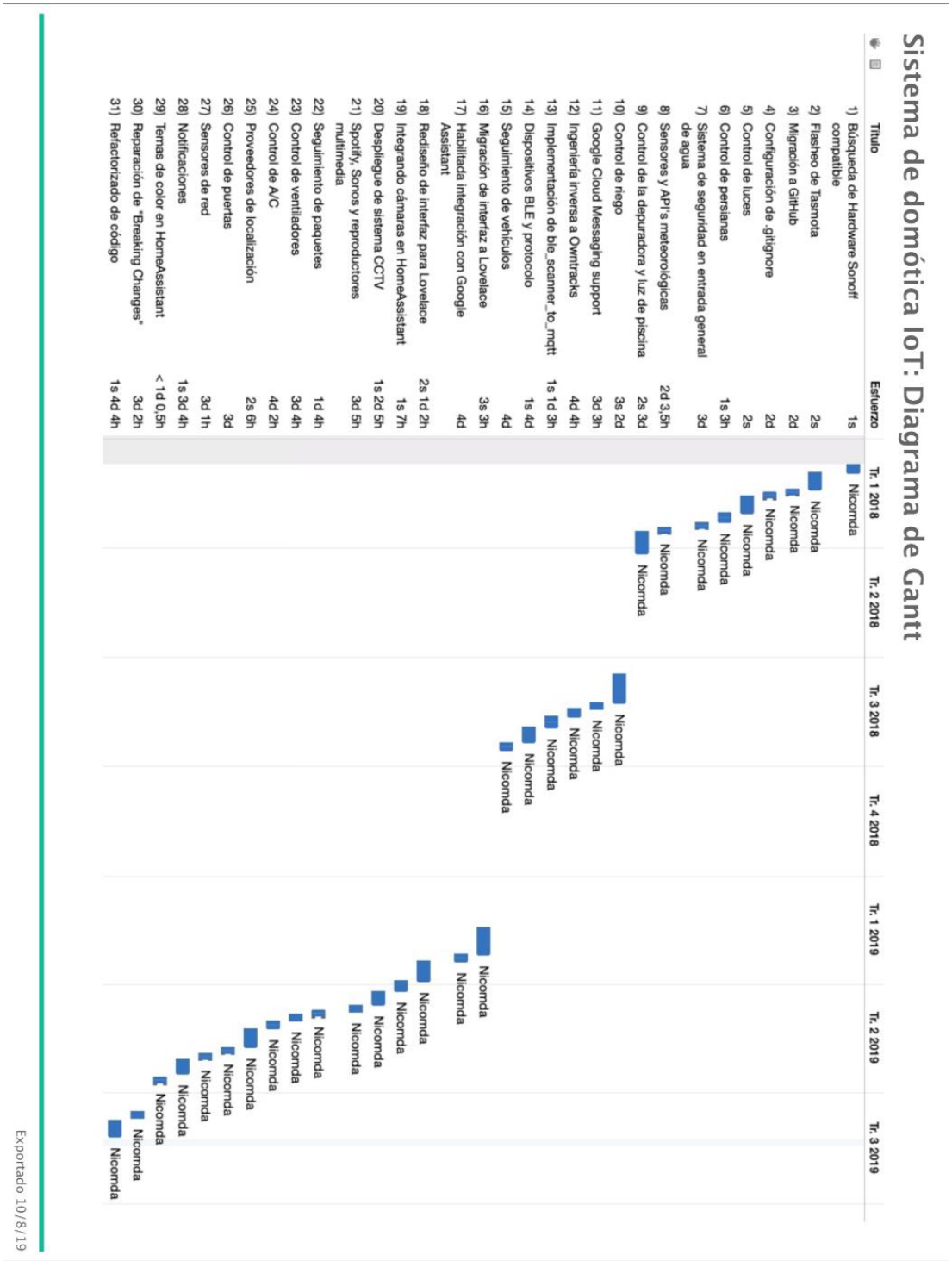


Ilustración 1: Diagrama de Gantt

También he considerado conveniente incluir una tabla con los costes asociados al tiempo asignado para la realización de cada etapa. Está en este caso separado del presupuesto ya que es necesario diferenciar el coste de inversión en investigación del coste de implementación.

Sistema de domótica IoT: Cronograma de recursos

Nombre	Fecha de inicio	Fecha final	Duración	% completado	Coste de la asignación
Nicomda	**22/1/18 8:00**	**5/8/19 12:00**	**< 38s 3d 2h**	**100 %**	**23.189,55 €**
Búsqueda de Hardware Sonoff compatible	22/1/18 8:00	26/1/18 17:00	1s	100 %	600,00 €
Flasheo de Tasmota	29/1/18 8:00	9/2/18 17:00	2s	100 %	1.200,00 €
Migración a GitHub	12/2/18 8:00	13/2/18 17:00	2d	100 %	240,00 €
Configuración de .gitignore	14/2/18 8:00	15/2/18 17:00	2d	100 %	240,00 €
Control de luces	16/2/18 8:00	1/3/18 17:00	2s	100 %	1.200,00 €
Control de persianas	2/3/18 9:00	9/3/18 12:00	1s 3h	100 %	645,00 €
Sistema de seguridad en entrada general de agua	9/3/18 14:30	14/3/18 14:30	3d	100 %	360,00 €
Sensores y API's meteorológicas	14/3/18 15:30	19/3/18 10:00	2d 3,5h	100 %	292,50 €
Control de la depuradora y luz de piscina	19/3/18 10:30	5/4/18 10:30	2s 3d	100 %	1.560,00 €
Control de riego	16/7/18 8:00	7/8/18 17:00	3s 2d	100 %	2.040,00 €
Google Cloud Messaging support	8/8/18 9:00	13/8/18 12:00	3d 3h	100 %	405,00 €
Ingeniería inversa a Owntracks	13/8/18 14:00	20/8/18 9:00	4d 4h	100 %	540,00 €
Implementación de ble_scanner_to_mqtt	20/8/18 10:00	28/8/18 14:00	1s 1d 3h	100 %	765,00 €
Dispositivos BLE y protocolo	28/8/18 15:00	10/9/18 15:00	1s 4d	100 %	1.080,00 €
Seguimiento de vehículos	11/9/18 8:00	14/9/18 17:00	4d	100 %	480,00 €
Migración de interfaz a Lovelace	12/2/19 13:00	5/3/19 16:00	3s 3h	100 %	1.845,00 €
Habilitada integración con Google Assistant	6/3/19 8:00	11/3/19 17:00	4d	100 %	480,00 €
Rediseño de interfaz para Lovelace	12/3/19 11:00	27/3/19 14:00	2s 1d 2h	100 %	1.350,00 €
Integrando cámaras en HomeAssistant	27/3/19 16:00	4/4/19 15:00	1s 7h	100 %	705,00 €
Despliegue de sistema CCTV	5/4/19 9:00	16/4/19 15:00	1s 2d 5h	100 %	915,00 €
Spotify, Sonos y reproductores multimedia	17/4/19 11:00	22/4/19 17:00	3d 5h	100 %	435,00 €
Seguimiento de paquetes	23/4/19 10:00	24/4/19 15:00	1d 4h	100 %	180,00 €
Control de ventiladores	25/4/19 8:00	30/4/19 12:00	3d 4h	100 %	420,00 €
Control de A/C	30/4/19 15:00	6/5/19 17:00	4d 2h	100 %	510,00 €
Proveedores de localización	7/5/19 14:00	22/5/19 11:00	2s 6h	100 %	1.290,00 €
Control de puertas	22/5/19 13:00	27/5/19 12:00	3d	100 %	360,00 €
Sensores de red	28/5/19 8:00	31/5/19 9:00	3d 1h	100 %	375,00 €
Notificaciones	31/5/19 13:00	12/6/19 17:00	1s 3d 4h	100 %	1.020,00 €
Temas de color en HomeAssistant	18/6/19 8:00	19/6/19 8:28	< 1d 0,5h	100 %	127,05 €
Reparación de "Breaking Changes"	16/7/19 8:00	19/7/19 10:00	3d 2h	100 %	390,00 €
Refactorizado de código	23/7/19 8:00	5/8/19 12:00	1s 4d 4h	100 %	1.140,00 €

Ilustración 2: Cronograma y costes de la investigación y primera implementación

1.4.2. Técnica (GitHub)

La organización técnica del proyecto puede verse claramente reflejada en los commits de Git, a fecha del 29 de Julio de 2019 son 128 registrados, a lo que habría que incluir las primeras horas de trabajo del proyecto ya que no existía un repositorio desde el comienzo. GitHub proporciona varios gráficos para que pueda visualizarse correctamente la organización técnica de los proyectos que utilizan esta plataforma.

Ilustración 3: Adiciones y borrados por semana en GitHub

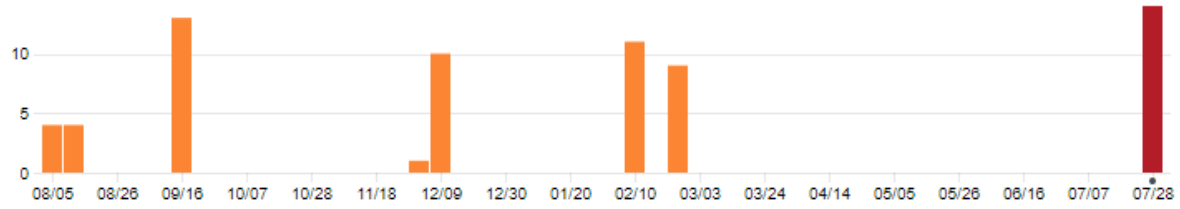


Ilustración 4: Referencia de commits semanales en GitHub

1.4.3. Presupuesto

Artículo	Descripción	Coste Unitario	Uds	Coste Total
Raspberry Pi 3B	Miniordenador SBC	36,05 €	1	36,05 €
Sandisk SD 32GB Class 10	Tarjeta MicroSD de clase 10	6,90 €	1	6,90 €
Asus RT-AC68U	Router (Incluye DDNS)	127,44 €	1	127,44 €
Sonoff Basic	Placa 1 relé, Interruptor básico	3,18 €	2	6,36 €
Sonoff 4ch DIY	Placa 4 relés	14,60 €	2	29,20 €
Sonoff TH10 + DHT22	Placa 1 relé con sensor de humedad y temperatura	11,77 €	1	11,77 €
Sonoff Dual	Placa 2 relés	9,08 €	2	18,16 €
Sonoff T1	Interruptor 1 relé	11,38 €	1	11,38 €
Sonoff T1 Doble	Interruptor 2 relés	16,60 €	1	16,60 €
Sonoff S26	Enchufe Wi-Fi	7,78 €	1	7,78 €
Xiaomi IR Remote	Emisor de infrarrojos	12,74 €	1	12,74 €
Amazon Dash Button	Botón Wi-Fi	1,99 €	8	15,92 €
BLE Beacon	Beacon de Bluetooth Low Energy	9,80 €	2	19,60 €
VStarcam C7824WIP	Cámara IP de interior	24,76 €	2	49,52 €
VStarcam C16S	Cámara IP de exterior	29,62 €	1	29,62 €
Dominio DDNS	Dominio para el acceso externo al sistema	0,00 €	1	0,00 €
Asus Lyra Mini	Sistema de red mallada, 3 dispositivos	97,99 €	1	97,99 €
Google Home Mini	Altavoz inteligente con Google Assistant	29,90 €	2	59,80 €
Yeelight Xiaomi	Bombillas RGB inalámbricas	13,46 €	4	53,84 €
FT232 USB UART	Conversor USB a UART	1,80 €	1	1,80 €
Cables Dupont	Cables para conectar placas por USB	0,90 €	1	0,90 €
Sensor DS18B20	Sensor de temperatura	1,30 €	2	2,60 €
Wemos D1 Mini	Módulo ESP-8266 mini	3,65 €	1	3,65 €
Instalación	Instalación del sistema en una vivienda (Horas)	18,00 €	72	1.296,00 €
		TOTAL		1.915,62 €

2. Tecnologías usadas. Estado del arte

2.1. Contraposición con redes 5G

Actualmente nos encontramos en un momento turbulento en lo que se refiere a tecnologías de red. Tanto la proliferación de la fibra óptica a nivel doméstico, como el próximo despliegue a gran escala de la tecnología 5G a nivel mundial, hacen que no solo mejore la velocidad y la calidad de las conexiones, sino también la forma de desarrollar e implementar los sistemas en red. En este momento, generalmente es necesaria una red local para un sistema de domótica. Sin embargo, es probable que con estas nuevas conexiones 5G, cambie lo que conocemos como entorno de red local. Este sistema se ha realizado todo dentro de una red local para preservar la privacidad y abaratar los costes, pero según los esquemas de topología de red del despliegue del 5G, van a desaparecer las diferencias entre red local e internet, conectando todos los dispositivos directamente a internet tal y como se muestra en la siguiente imagen.

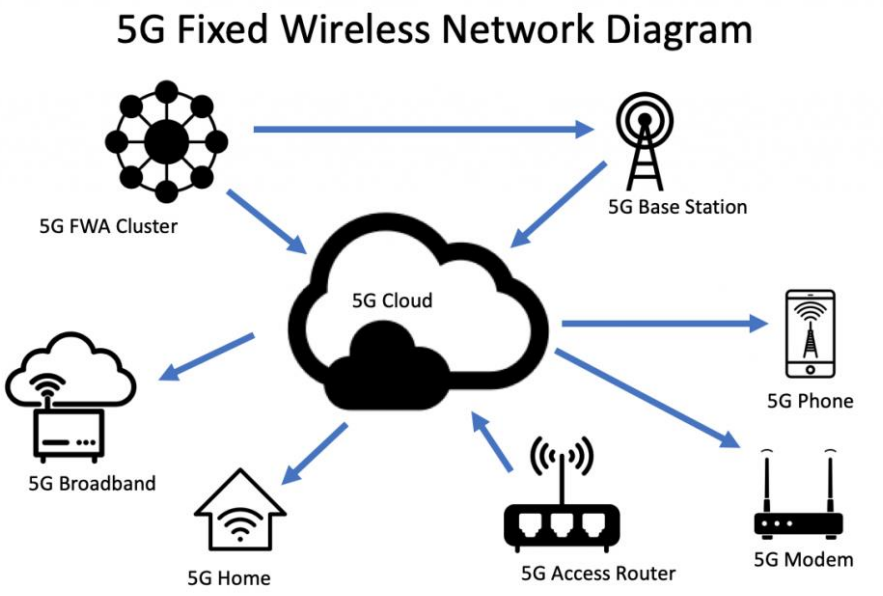


Ilustración 5: Topología de red 5G

2.2. Protocolos de domótica actuales

2.2.1. MQTT

Es un protocolo desarrollado por IBM, y actualmente se ha convertido en el más utilizado en aplicaciones IoT. Su nombre, son las siglas de “Message Queue Telemetry Transport” y se enfoca en la conectividad M2M. Como desde el principio ha sido un protocolo enfocado al funcionamiento en posibles dispositivos con hardware y software empotrado, está diseñado para consumir muy poco ancho de banda, así se reduce el consumo y es posible manejarlo con dispositivos de muy bajos recursos. Con todas estas ventajas en lo que a IoT se refiere, se emplea principalmente en la conexión entre dispositivos domóticos, desde sensores hasta relés, y su visión para el uso a nivel empresarial hace que sea completo para cualquier tipo de entorno relacionado con domótica.

Su arquitectura sigue una topología de estrella, principalmente existe un nodo central, también llamado “broker”, que es capaz de trabajar con un gran número de dispositivos de forma simultánea. MQTT permite también el establecimiento de comunicaciones cifradas mediante *SSL* (Secure Socket Layer).

Para entender su funcionamiento es necesario conocer el concepto de “topic”. Un topic, o tema en español, es un canal de comunicación que establecen uno o varios dispositivos a través del bróker. Siempre que un dispositivo se suscribe a un tema, éste puede publicar y recibir información a través del mismo. El protocolo mantiene una fuerte estructura jerárquica, de tal forma, si suscribimos un dispositivo a un topic padre, este recibe la información de sus hijos. Una arquitectura básica es la que vemos en la ilustración.

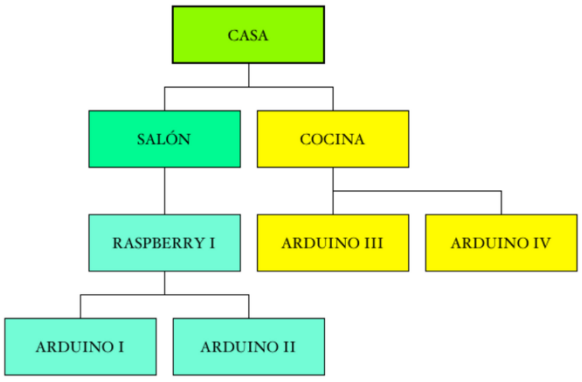


Ilustración 6: Jerarquía MQTT

2.2.2. Zigbee

Al igual que MQTT, Zigbee se basa en el reducido consumo de energía y recursos, pero en este caso, Zigbee no es solo un protocolo de comunicación de red para dispositivos de IoT si no que incluye unas especificaciones de hardware. Para fabricar dispositivos que utilicen este protocolo es necesario pertenecer a la Zigbee Alliance y el protocolo solo puede ser utilizado por dispositivos que pertenezcan a ella. Siguen la norma del IEE 802.15.4, y tiene que pasar los estándares para poder ser comercializados. Existen 3 tipos de dispositivos:

- **Zigbee Coordinator**

  Solo existe 1 dispositivo coordinador para la red. Controla la red y los caminos de esta para que los dispositivos se interconecten.

- **Zigbee Router**

  Interconecta dispositivos separados en la red. Es con el que se comunica el usuario.

- **Zigbee End Device**

  Puede comunicarse solo con su padre ya sea router o coordinador. Sin embargo, no existe una administración directa hacia él mismo. Esto permite que aumente la vida útil de la batería, pero nos limita en control.

No se ha utilizado en este proyecto porque al incluir revisiones y estándares de hardware y software suelen ser más caros, además limita bastante la cantidad de dispositivos disponibles.

2.2.3. Z-Wave

El último de los grandes protocolos enfocados a la domótica, utiliza una red mallada de radiofrecuencia de bajo consumo para establecer la comunicación entre los dispositivos de la red. Son necesarios nodos de enlace con soporte de conexión de red LAN para poder controlarlos de forma inalámbrica. Son interoperables entre los diferentes fabricantes de Z-Wave, aunque este sea un protocolo cerrado. Z-Wave permite transmitir la información de un dispositivo a otro hasta llegar al Hub, o nodo

de enlace, sin embargo está limitado a cuatro saltos. El alcance, es mayor que el de Zigbee al usar radiofrecuencia, de 10-20 metros que soporta Zigbee hasta 100 metros que es capaz de soportar Z-Wave.

Z-Wave tiene bastante reducido el número de dispositivos que podemos conectar en la red, siendo el máximo 232, mientras que en Zigbee son 65.000.

2.3. Frameworks de domótica actuales

2.3.1. Home Assistant

Es una plataforma de código abierto de domótica para el hogar desarrollada en Python 3. Permite a través de lo que ellos llaman integraciones, la interacción con diferentes *API*'s, plataformas, y dispositivos. Actualmente dispone de más de 1400 componentes, desde sistemas como Nest, Philips Hue o Google Assistant, hasta componentes genéricos para adaptar cualquier marca con protocolos habituales. La interfaz gráfica es bastante moderna, el sistema es estable, y se actualiza de forma continua a ritmo de casi una vez por semana. Próximamente llegará a la primera versión estable de la plataforma y hay una cantidad enorme de desarrolladores activos mejorando el sistema. El lenguaje para su programación a nivel de módulos es *YAML* y para ampliar su funcionalidad con script realmente programables utiliza Jinja2.

2.3.2. OpenHAB

Esta plataforma, se desarrolla en Java. También es de código abierto y es más antigua que Home Assistant. También tiene bastantes componentes y *addons*, prácticamente la misma cantidad que el anterior. Es un software más maduro, sin embargo con mucho más tiempo que Home Assistant, tiene la misma cantidad de módulos. Su lenguaje es Xbase, bastante sencillo, al igual que *YAML*.

2.3.3. Domoticz

Empezó como un producto especializado en dispositivos de radiofrecuencia. Está escrito en C++ y aunque se ha actualizado bastante, su última versión estable es de 2018. La interfaz por defecto da aspecto de antigua y el número de componentes e integraciones es bastante menor que en Home Assistant u OpenHAB. El software es más ligero que los anteriores, pero a no ser que se necesite para hacer algo sencillo

basado en radiofrecuencia o que estemos realmente muy limitados a nivel de Hardware, no tiene sentido utilizarla.

2.4. ¿Por qué Home Assistant y MQTT?

Tras la visión general mostrada en el apartado anterior sobre las plataformas domóticas más utilizadas actualmente, cabe explicar la elección de Home Assistant. Python 3 es un lenguaje sencillo. Esto hace que cada día lleguen más desarrolladores a la plataforma para aportar actualizaciones y mejoras. La interfaz está inspirada en *Material Design* de Google, y las posibilidades de personalización tanto visual como a nivel de automatizaciones y scripts, no solo mediante *YAML* si no también gracias a Jinja2, son infinitas.

YAML es muy fácil de convertir a *JSON*, facilitándonos el acceso a la creación de más componentes para el sistema, dada la cantidad de *API*'s que utilizan este lenguaje. Esto y las continuas actualizaciones de la plataforma semanalmente hacen de Home Assistant un lugar perfecto para empezar en el terreno del IoT a nivel doméstico.

Home Assistant, soporta Z-Wave y Zigbee. A pesar de ello, MQTT es un protocolo más versátil, más extendido, y más prometedor a nivel de futuro que los anteriores. Tanto es así, que existen varios conversores vía software para integrar el control de dispositivos basados en las tecnologías anteriores, dentro del protocolo MQTT haciéndolo transparente para nosotros y permitiendo gestionar todo a través de nuestro bróker MQTT. Es sencillo, y principalmente, compatible con el firmware que usan la mayoría de los dispositivos utilizados en este proyecto, el cual detallaremos más adelante en el punto 4.1.

2.5. Mosquitto Client/Server

Una vez que se ha clarificado el protocolo a utilizar, se necesita el software adecuado para crear el bróker, y para que podamos hacer las pruebas pertinentes en el sistema. Para ello usaremos Mosquitto, el software de Linux más famoso para sistemas MQTT. Es un servidor (bróker) de bajo consumo y código abierto escrito en C y que tiene las librerías necesarias para utilizarlo también como cliente MQTT. Para ver su funcionamiento utilizamos básicamente dos comandos. El primero, para

publicar mensajes en el bróker. Los argumentos son -h para la IP del bróker, -t para el topic y -m para el mensaje (*payload*).

El segundo comando te permite suscribirte a un tema para recibir la información del

```
osboxes@osboxes:~$ mosquitto_pub -h 46.183.114.53 -t "/casa/salon/bombilla/1" -m "Bombilla 1: ON"
osboxes@osboxes:~$ mosquitto_pub -h 46.183.114.53 -t "/casa/salon/bombilla/2" -m "Bombilla 2: ON"
osboxes@osboxes:~$ mosquitto_pub -h 46.183.114.53 -t "/casa/salon/bombilla/3" -m "Bombilla 3: OFF"
osboxes@osboxes:~$
```

Ilustración 7: Mosquitto (Publicación)

mismo. Los argumentos de la línea de comandos se mantienen igual que en el anterior, excepto que al suscribirse no se envía un mensaje.

```
osboxes@osboxes:~$ mosquitto_sub -h 46.183.114.53 -t "/casa/salon/bombilla/#"
Bombilla 1: ON
Bombilla 2: ON
Bombilla 3: OFF
```

Ilustración 8: Mosquitto (Suscripción)

3. Hardware

3.1. ESP-8266

3.1.1. Características

Es básicamente un chip integrado, que tiene conexión Wi-Fi y es compatible con el protocolo TCP/IP. Con él conseguimos la funcionalidad básica de un Arduino, pero con la potencia de tener acceso a internet. Se programa mediante el mismo lenguaje que se usa en Arduino habitualmente, una especie de pseudo C. La empresa que desarrolló el chip y que actualmente continúa siendo fabricante del mismo es Espressif.

Hardware:

- CPU Tensilica L106 32-bit
- Voltaje de operación 3-3,6V
- Corriente de operación 80mA
- Temperatura de operación -40-125º

Conectividad:

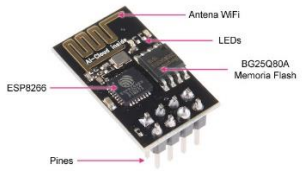
- Soporta protocolo IPv4, incluyendo esto TCP/UDP/HTTP/FTP e incluso HTTPS mediante una capa de software. Actualmente TLS 1.2.

Modos de operación:

- Active Mode: Funcionalidad completa, consumo aproximado al de operación
- Sleep Mode: Solo mantiene activo RTC (Reloj en tiempo real) para mantener la sincronización, y queda alerta de eventos que lo despierten. Mantiene datos de conexión en memoria para evitar restablecimiento de la conexión Wi-Fi. Consumo: Entre 0,6 y 1mA.
- Deep Sleep: RTC encendido pero no operativo, debe existir una transición al Sleep Mode previa al modo activo. Pierde los datos almacenados ya que no alimenta la memoria. Consume 20µA.

El fabricante AI-Thinker, que fue el primero en adoptar en sus placas la serie ESP, desarrolló diferentes módulos para adaptarse a usos distintos, que incluyen además del ESP-8266, un módulo de memoria, una antena, y proporciona pines para facilitar la conexión del dispositivo.

Módulos ESP-XX:

ESP-01	ESP-05	ESP-12	NodeMCU
			
Es el módulo más popular, de 1.5-4€/ud. 2 pines *GPIO* digitales.	Es el más simple, se considera un Shield Wifi para Arduino, 3€. No dispone de *GPIO*	Convirtiéndose en el buque insignia. 11 puertos, 1 es analógico. Necesario soldar.	Basado en ESP-12. Incluye adaptador USB. Dispone de firmware para programarlo en LUA, Python o JS.

Existen más módulos aparte de estos, fabricados por otras empresas, por ejemplo, son muy utilizados todos los de Wemos. Por lo general, en los productos que se utilizan en este proyecto, se incluye el módulo PSF-B85 que es una variante encapsulada del ESP-12. También se incluyen en los módulos más modernos el ESP-8285, que es un 8286 con el módulo de memoria ya incluido en el interior.

3.1.2. Pin-Out y diagrama electrónico

Para el pin-out se muestra NodeMCU ya que es más completo y tiene disponibles más *GPIO* para proyectos de domótica, y sobre todo una entrada analógica para sensores. En principio, el ESP8266 tiene 17 pines de propósito general. Sin embargo, solo podemos usar 11 de ellos ya que hay 6 pines que se usan para conectar el chip de memoria flash. En el lado izquierdo, podemos ver, el pin A0, que es la entrada analógica para sensores, pines de alimentación y control y la interfaz SPI.

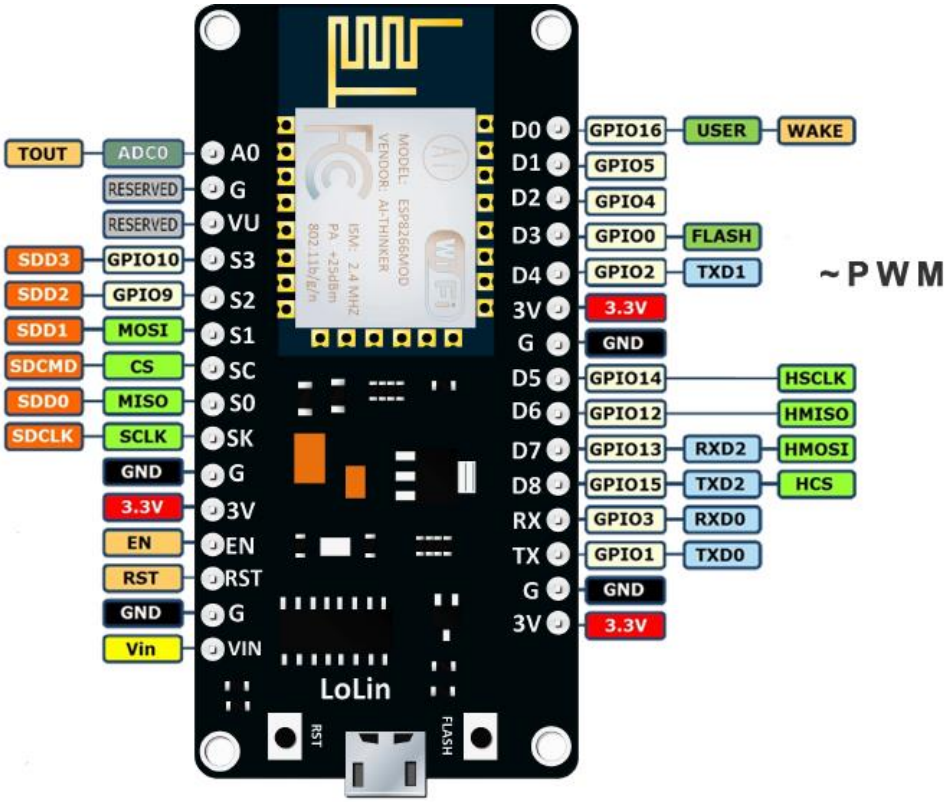


Ilustración 9: NodeMCU Pinout

El diagrama electrónico del ESP8266 es bastante sencillo. He elegido el ESP-01 para mostrarlo donde pueden verse la interconexión entre el chip, la memoria y el reloj.

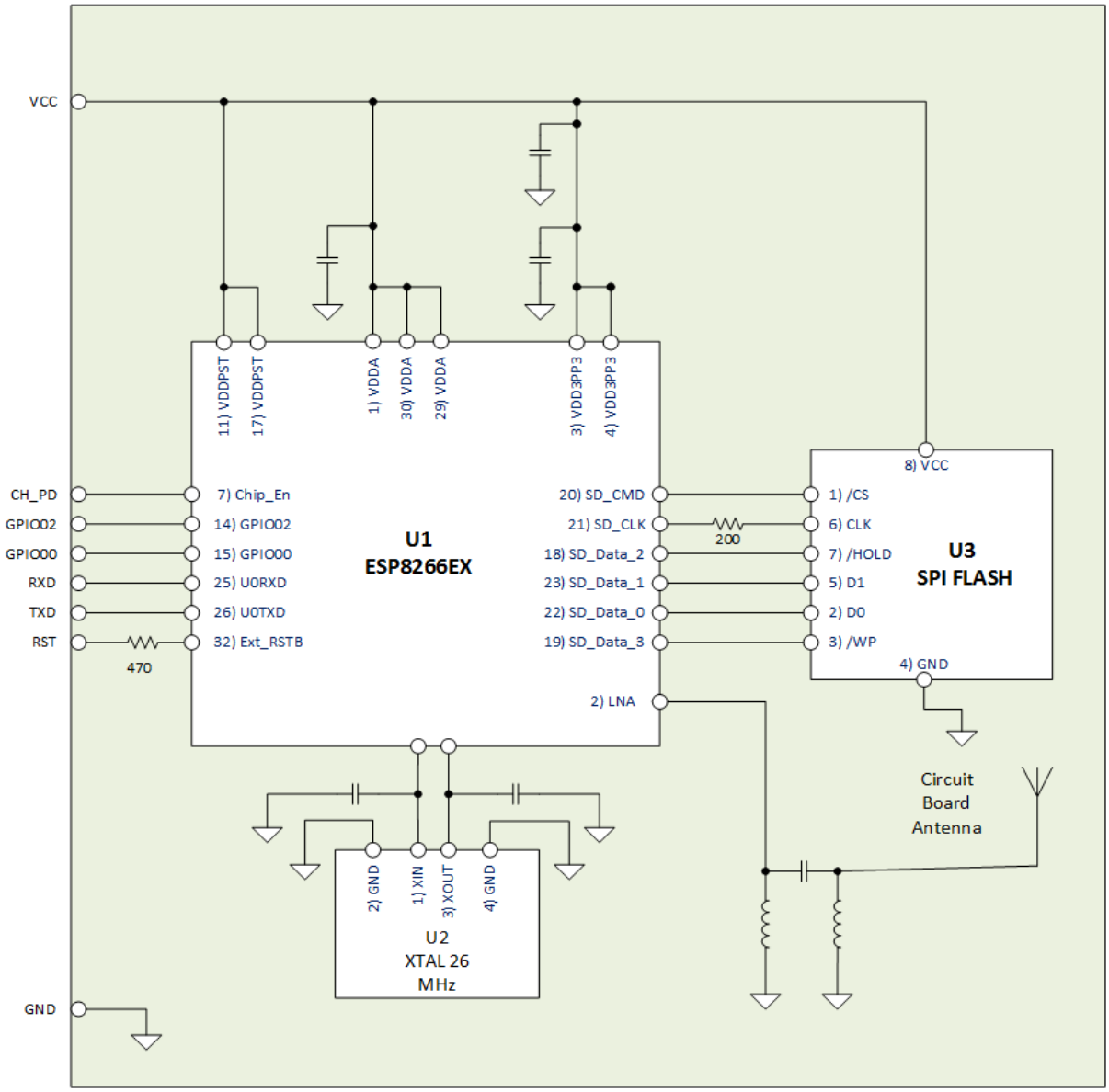


Ilustración 10: Diagrama electrónico ESP-01

3.1.3. Interfaz de conexión UART a USB

Aunque no es necesario para trabajar con módulos de pruebas como NodeMCU, generalmente en cualquier dispositivo en el que esté empotrado un ESP-8266 se encuentran expuestos los pines de conexión UART (TX,RX,VCC,GND). Con estos pines, estableceremos una conexión serie desde USB hasta el ESP-8266 para poder reprogramarlo y utilizar un firmware con soporte de MQTT. Para ello necesitamos un adaptador USB-UART. Este adaptador alimenta el circuito a través de los pines VCC y GND, generalmente disponen de un jumper para seleccionar alimentación a 5V o a

3,3V. En esta parte del proceso es muy importante seleccionar bien la tensión ya que si alimentamos un ESP-8266 a 5V puede quedar inutilizable. A continuación se muestra un diagrama de conexión con un ESP-01.

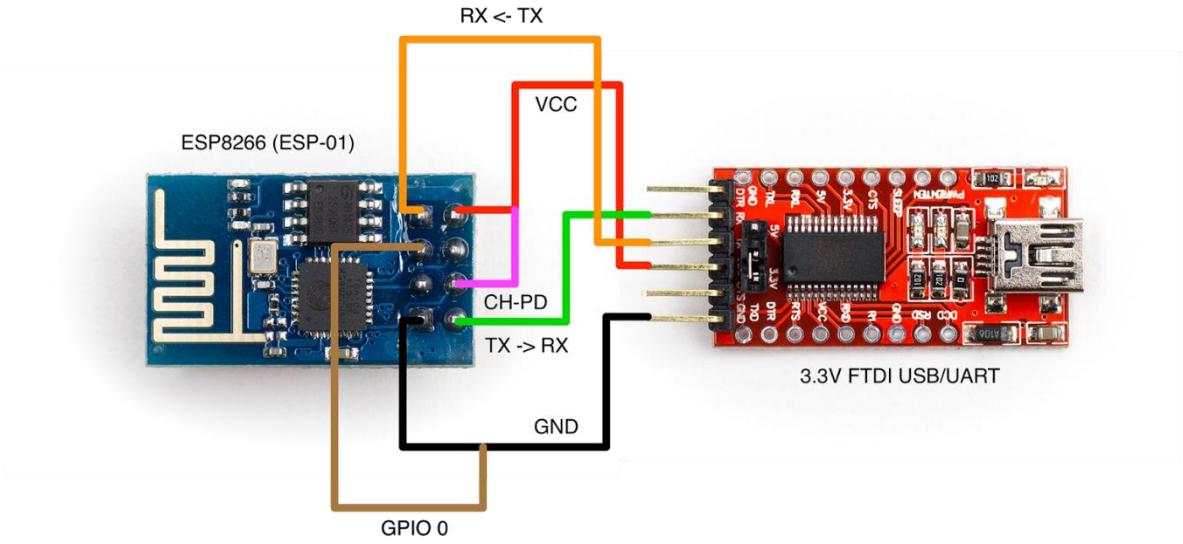


Ilustración 11: Conexión USB-UART

En el diagrama observamos lo siguiente:

- Pines de transmisión y recepción (TX/RX) deben ir cruzados.
- Hay 2 pines en el FTDI que quedan sueltos porque viene preparado para una interfaz RS232, el USB solo necesita 4.
- La conexión morada del esquema no es estrictamente necesaria.
- *GPIO*-0 está puenteado a tierra. Esta es la forma de activar el modo programación en un ESP-8266, no siempre hay que puentear, dependiendo de la placa puede ser solo pulsando un botón.

3.1.4. Proceso de flasheo

Una vez tengamos los 4 pines necesarios conectados a FTDI, simplemente conectamos el USB al ordenador y si hemos instalado los drivers necesarios (CH340) de forma correcta, tendremos un puerto COM funcional disponible para reprogramar el chip. Tanto los dispositivos Sonoff como los Wemos que se han utilizado, necesitan modificar el firmware a uno mucho más potente y versátil llamado Tasmota. Es completamente configurable y tiene soporte directo con MQTT, seguridad, programaciones a nivel interno, reglas e incluso OTA (Actualización vía Wi-Fi).

Aunque lo explicaremos más adelante en el punto 4, existen varios firmwares para ESP-8266 que son muy completos, pero el proceso de flasheo es el mismo para todos. Cuando se empezó a desarrollar este proyecto se necesitaba el IDE de Arduino oficial para compilar el firmware y subirlo a las placas, teniendo incluso que añadir librerías externas para soportarlas. Más tarde, apareció *PlatformIO*, que, aunque se considere una extensión de editores como Atom o Visual Studio Code, por su magnitud puede considerarse un IDE completo como el de Arduino, aunque mucho más moderno y cómodo que este. Al principio, eran estas las opciones posibles porque había que recompilar el firmware añadiendo, como mínimo, datos de conexión como el *SSID* y la contraseña Wi-Fi. Sin embargo, el firmware ha avanzado mucho y crea una red provisional de forma automática, a la que podemos conectarnos para configurarlo y que se conecte a la nuestra, por tanto, ya que no es necesario recompilar, ahora la opción más cómoda pasa por utilizar una aplicación en la que directamente seleccionamos los siguientes parámetros:

- Puerto COM
- Imagen del firmware que queremos subir a la placa descargada directamente desde el apartado de “releases” de Tasmota en GitHub
- Velocidad en baudios (por defecto 115200)
- Modo de flasheo, que debe ser siempre Dual Output (DOUT)
- Si queremos o no borrar la memoria interna. (No es necesario si simplemente queremos actualizar vía USB)

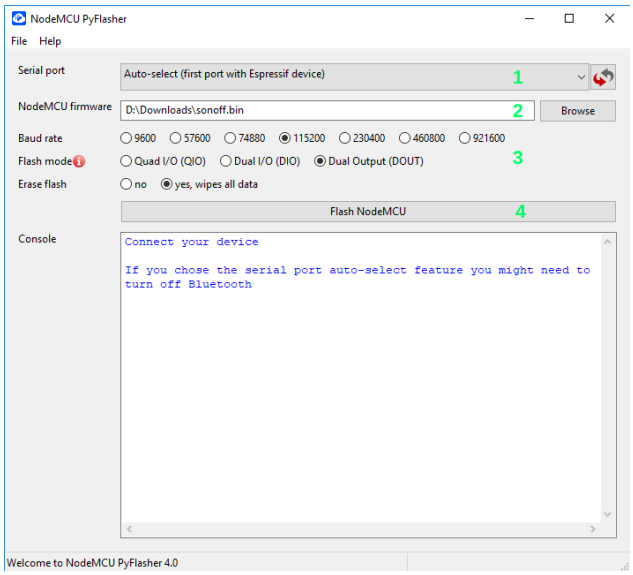


Ilustración 12: Software para flashear ESP-8266

3.1.5. Dispositivos utilizados tras flasheo

Durante la implementación de este proyecto se han utilizado diferentes dispositivos de la marca Sonoff, y un Wemos para controlar dispositivos e información de sensores. A continuación, se detalla el uso de cada uno de los modelos utilizados. Las imágenes que se aportan reflejan los pines de conexión para el flasheo.

Sonoff Basic

El más pequeño de la serie, dispone de solo un relé y un pequeño interruptor para activarlo y desactivarlo de forma manual. Es el más barato de todos y con el que empezó a realizarse este proyecto. Se ha utilizado para varios interruptores que operaban de forma manual, por ejemplo, en el equipo de sonido del sótano y en la alimentación de la cámara IP del interior de la casa.

Ilustración 13: Sonoff Basic

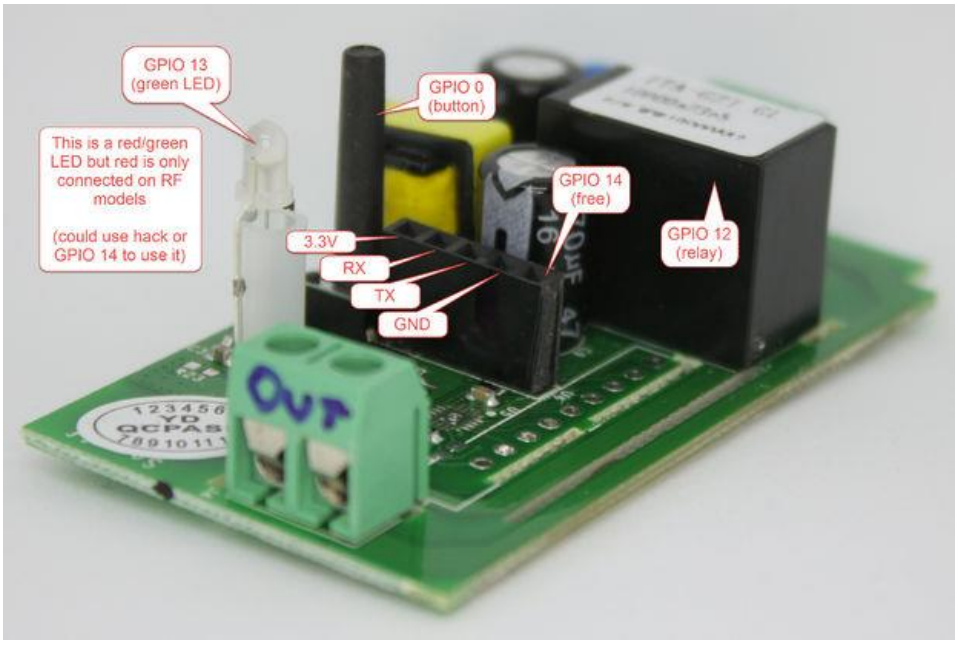


Ilustración 14: Sonoff Basic abierto

Sonoff 4CH DIY

Es una placa de la serie DIY (Do it yourself) que consta de 4 relés. En la instalación se han utilizado 2:

- Placa 1 (RIEGO):
  - 1 relé para cada una de las electroválvulas de riego
- Placa 2 (CONTROL GARAJE):
  - 1 Relé para activar la depuradora
  - 1 Relé para activar la luz de la piscina
  - 1 Relé para activar la puerta del garaje
  - 1 Relé queda libre actualmente

Actualmente la información de flasheo que consta en el GitHub del firmware de Tasmota, se aportó directamente tras investigarla en este proyecto. Es la primera placa en cuyas cabeceras de soldadura no existen los pines TX y RX para establecer la conexión UART. De esta forma, fue necesario hacerse con el esquema electrónico del chip para saber la correspondencia de los pines en el chip directamente. El proceso de flasheo es bastante complejo. Es necesario tener mucho cuidado al disponer el cableado en el chip, ya que dos de los cables *dupont* que se conectan desde el FTDI USB-UART, no van a un "*header*" si no directamente al chip. Se observa mejor en la imagen.

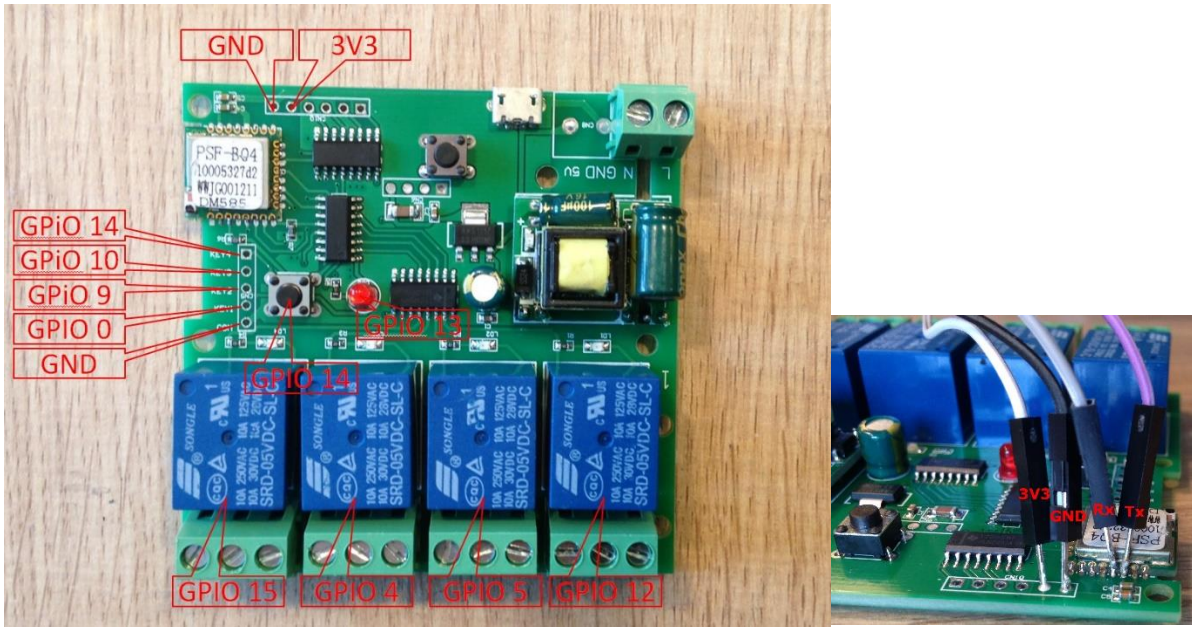


Ilustración 15: Sonoff 4CH DIY (Vista superior y lateral del chip)

Sonoff 4CH Pro R2

Es una placa similar a la DIY de cuatro canales. Es más cara, permite montaje en railes DIN, y dispone de botones en la parte izquierda para activar de forma individual cada uno de los 4 relés disponibles en ella. Además, aunque en el proyecto se hace vía software directamente en el firmware, esta placa dispone de microinterruptores para seleccionar varios modos de operación con los que podemos configurar si pueden o no activarse varios relés, si solo pueden activarse de forma individual…

La instalación del firmware es bastante más sencilla que en la anterior, pues en esta, sí que están accesibles los pines necesarios para la conexión vía UART. Se ha utilizado para la instalación de las persianas del salón.

Ilustración 16: Sonoff 4CH PRO

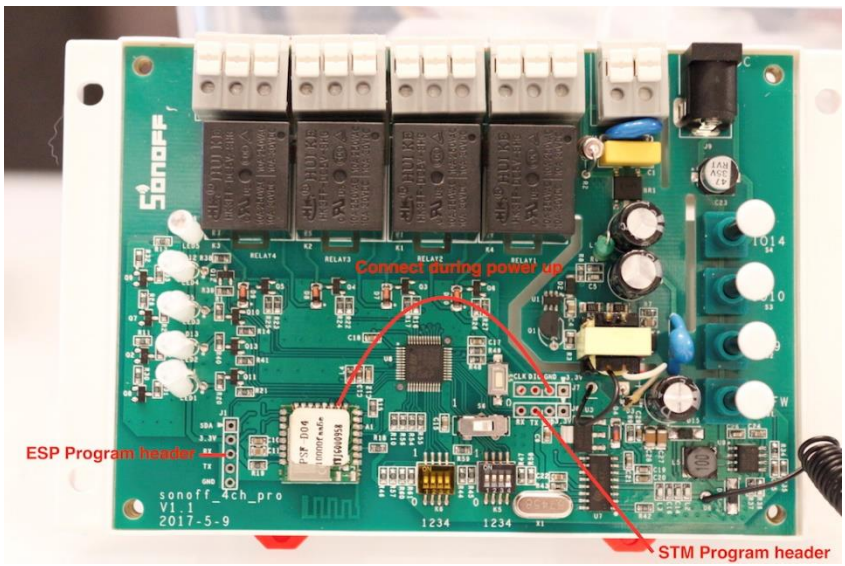


Ilustración 17: Sonoff 4CH PRO abierto

Sonoff TH10

Es una placa que dispone de solo un relé como Sonoff Basic, aunque de aspecto se parece bastante más al Dual. Dicho esto, su diferencia más interesante es un conector minijack lateral donde podemos conectar un sensor de temperatura y/o humedad. Se ha utilizado para disponer de información sobre ambas medidas en el sótano, además de como interruptor de emergencia. Es uno de los interruptores más interesantes del proyecto, diseñado para funcionar a prueba de fallos. Este Sonoff está instalado de tal forma que alimenta la Raspberry Pi 3B que aloja a Home Assistant y el router del sótano. Es un relé aislado al que no tenemos acceso desde la interfaz de Home Assistant, su configuración por defecto en caso de fallo en la red eléctrica es de cerrarse al encenderse para volver a activar el sistema. Tiene conectado un sensor AM2301. Es el último Sonoff que se ha añadido al proyecto, para evitar caídas del sistema. Puede accederse a su interfaz directamente a través de internet, pero no a través del sistema. En teoría debe de permanecer siempre encendido a no ser que haya algún problema mayor. Su proceso de flasheo es bastante sencillo ya que tenemos los pines directamente disponibles en la placa, y el *GPIO*-0 se puentea con tierra al pulsar el botón del interruptor.

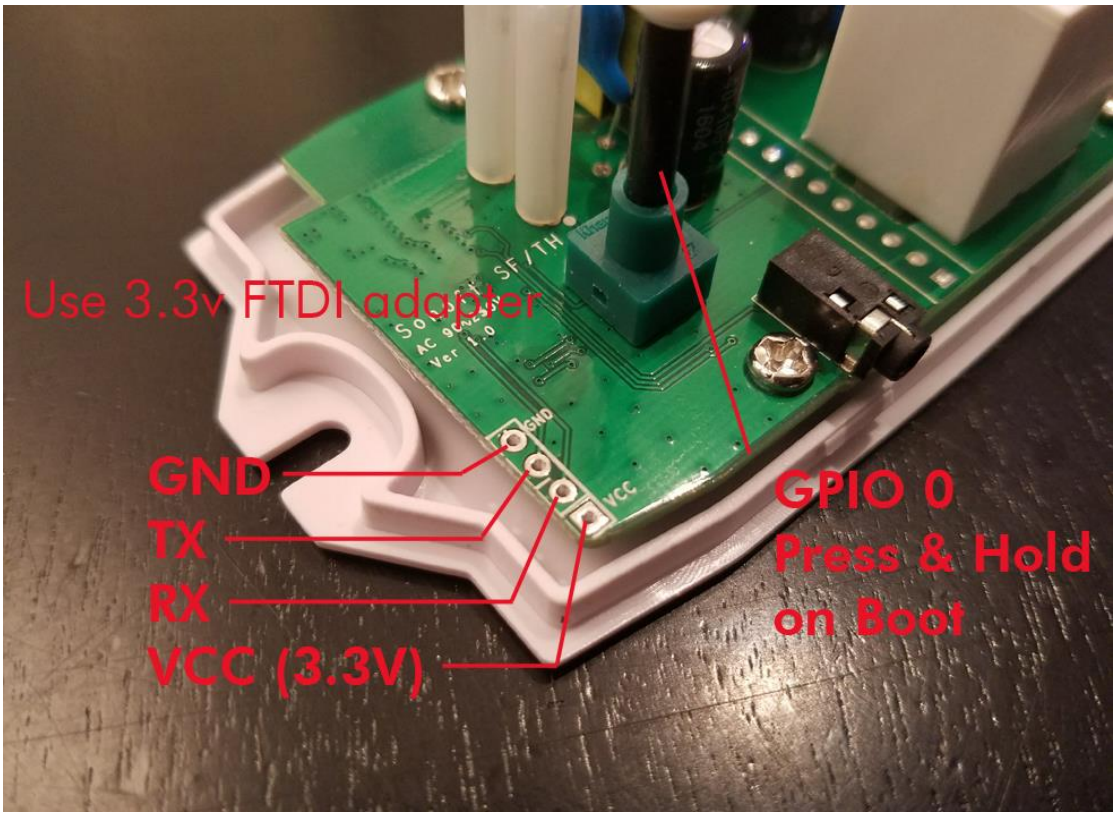


Ilustración 18: Sonoff TH10

Sonoff Dual R2

Esta placa dispone de 2 relés, básicamente es igual que un Basic pero con un relé más disponible para conectar. En el proyecto se han utilizado para la electroválvula de entrada de agua general, y para las persianas individuales, pues son ideales para este uso, ya que el motor necesita dos salidas, una para bajar y otra para subir.

Ilustración 19: Sonoff Dual

El aspecto exterior del dual es similar al del TH10, sin embargo, carece de entrada de minijack y tiene un relé más, pero el flasheo es idéntico. Una vez instalado hay que tener cuidado de activar el interbloqueo, ya que ambos relés no deben nunca de estar instalados a la vez. En la imagen se muestra el diagrama de conexión. Las flechas indican la conexión de los interruptores. Como ya existían interruptores enclavados mecánicamente, se han utilizado para ser lo menos intrusivo posible.

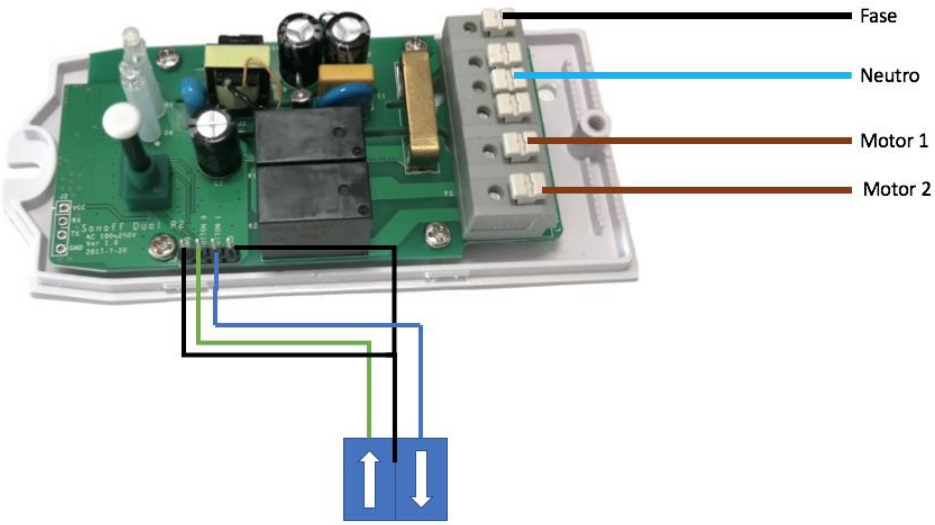


Ilustración 20: Sonoff Dual (Diagrama persianas)

Sonoff Touch (Doble y simple)

Esta placa está diseñada para sustituirla por interruptores de pared directamente. En este caso, se han instalado en el sótano para el control de luces, y en la puerta de la calle para activar el relé de la cerradura. Existen opciones desde 1 a 3 relés en el interior.

Ilustración 21: Sonoff Touch (1 y 2 relés)

El proceso de flasheo de un Sonoff Touch es igual independientemente de los interruptores que tenga. Es bastante sencillo porque tiene los pines UART con un adaptador ya soldado. Además, el *GPIO*-0 para poner el ESP-8266 en modo flasheo es sencillo de encontrar.

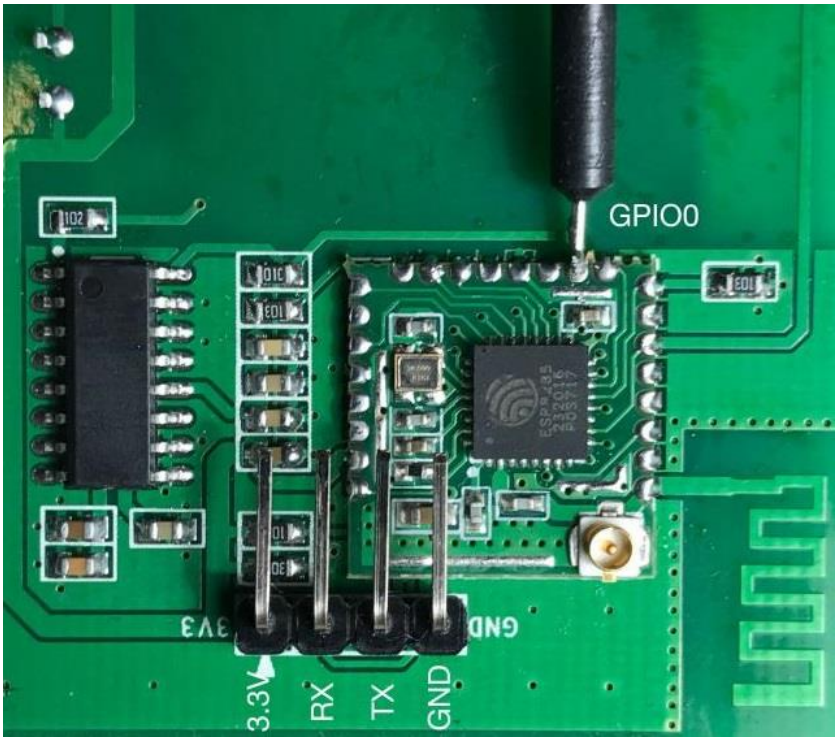


Ilustración 22: Sonoff Touch (Interior)

Sonoff S26

Es un módulo de Sonoff con una entrada de conector *schuko* macho, y una salida hembra. Así, utiliza el relé para unir entrada y salida permitiéndonos conectar y desconectar de forma remota o incluso automática lo que hay conectado en el enchufe. En el proyecto se ha utilizado para controlar de forma remota la alimentación de los monitores y el estudio de música. No hay problema en darle este uso, pero siempre teniendo en cuenta el amperaje de los equipos que conectemos, ya que el relé soporta hasta 10A.

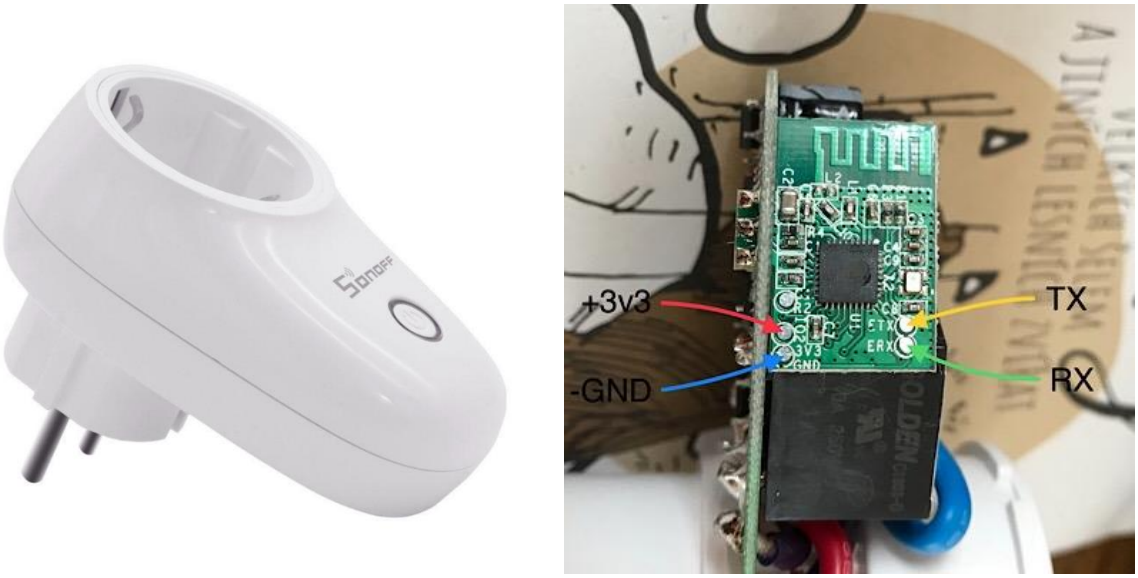


Ilustración 23: Sonoff S26 EU (Exterior e interior)

El flasheo de este dispositivo es algo más engorroso porque hay muy poco espacio en la placa, además es bastante difícil sujetarla. Así que, a pesar de que los pines están bastante expuestos, es uno de los más difíciles de conectar al puerto serie para cargar el nuevo firmware. Pero una vez conectados los pines necesarios, al igual que los anteriores no hay más que conectarlo al USB y cargar el firmware Tasmota. Una vez hecho esto tendremos todo lo necesario para conectarlo a Home Assistant.

Cabe decir que a la hora de adquirir esta placa hay que tener bastante cuidado ya que existen varias ediciones dependiendo del tipo de enchufe del territorio en el que nos encontremos. Debemos comprar la versión EU.

Sonoff iFan02

Es una de las placas más modernas de la empresa. Está diseñada para conectarla a ventiladores de techo. Es bastante delgada y en el proyecto no ha habido problemas para dejarla dentro del ventilador. Tiene conexión Wi-Fi y *RF*, por defecto trae un mando con el que podemos controlar tanto la luz como las velocidades del ventilador al que conectamos la placa. En este caso, la imagen que se ha adjuntado refleja una simulación del diagrama de conexión.

Ilustración 24: Sonoff iFan02 (Diagrama de conexión)

Una vez que se ha abierto la placa, la conexión es sencilla. Los pines están expuestos totalmente y el *GPIO*-0 accesible para conectarlo a tierra y activar el modo de flasheo.

La imagen es del proceso que se expone en el GitHub de Tasmota. En cambio, no recomiendo soldar el cable amarillo que aparece en la imagen, sino simplemente puentearlo con un *dupont*, ya que ha de desconectarse tras el flasheo.

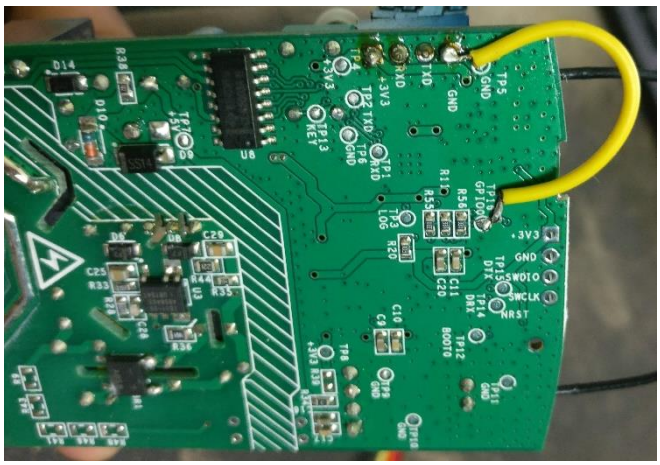


Ilustración 25: iFan02 (Interior)

Wemos D1 Mini

De todos los dispositivos sobre los que hemos hablado en este punto, el D1 mini es el más sencillo de programar puesto que la interfaz USB está disponible directamente en placa, y además tiene el botón reset puenteando *GPIO*-0 a tierra de tal forma que simplemente conectando el USB con el botón de reset pulsado, Wemos D1 mini pasará al modo de flasheo. También es compatible con Tasmota, y aunque en algunas de las placas mencionadas anteriormente haya disponibles algunos *GPIO* del ESP-8266/85, si va a trabajarse con sensores, es recomendable utilizar Wemos, por las siguientes razones:

- Todos los pines a excepción de los necesarios para el módulo de memoria están expuestos por defecto, ya que es una placa diseñada para desarrollar proyectos DIY en IoT.
- Es bastante más barato que los módulos de Sonoff
- Posibilidad de soldar pines para hacer pruebas con conectores *dupont* o de dejarlos directamente para soldar tanto sensores como actuadores a la placa.

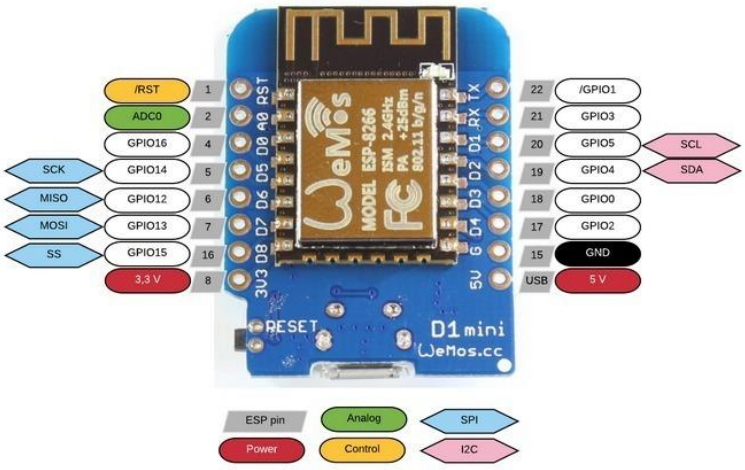


Ilustración 26: Wemos D1 mini pin-out

La temperatura interna de la planta baja y del exterior de la casa utiliza una de estas placas y 2 sensores D18B20.

Además de todas estas ventajas comentadas, es una placa que conocemos y hemos controlado anteriormente, puesto que se ha usado en la asignatura “Microprocesadores y Microcontroladores” de este grado.

3.2. Comparativa tecnológica

La razón por la que se está implementando en prácticamente todos los dispositivos domóticos a la venta es sencilla. Un módulo básico de ESP8266 tiene un precio aproximado de 1.50€, mientras que su competidor directo, Arduino MKR1000 cuesta alrededor de 35€. Esto ha hecho que la proliferación del ESP-8266 y del ESP-8285 (Es un ESP-8266 con el módulo de memoria integrado) sea tan rápida. Los módulos de conexión tanto LAN como Wi-Fi de Arduino siempre han sido bastante caros. Sin embargo, ESP-8266 tenía las características ideales para triunfar en el mercado. Es barato, pequeño, fácil de programar y aunque a nivel de entradas analógicas está más limitado que Arduino, tenemos conexión a internet sin necesidad de ningún otro shield o módulo. El acceso a la red de redes es tan potente que una vez que lo tenemos, descubrimos que mucha parte de la información de los sensores que antes necesitábamos conectar a la placa, ahora puede obtenerse fácilmente desde diferentes *API*'s.

No por ello debemos de olvidarnos de Arduino. Existen muchos proyectos que utilizan la conexión Wi-Fi de un ESP-8266 combinándola con un Arduino UNO o MEGA, que tienen disponibles muchas más entradas y salidas. Puede establecerse, al igual que se hace a la hora de programarlos, una conexión serie entre ambos dispositivos para que se comuniquen entre ellos y funcionen de forma autónoma, o para mostrar datos en una interfaz web, que puede construirse perfectamente en un ESP-8266.

4. Firmware y programación

4.1. Tasmota

4.1.1. Configuración inicial

Tasmota, tal y como especifica en la web del proyecto es un firmware alternativo para dispositivos basados en ESP-8266/85 que dispone de interfaz web, reglas, temporizadores, actualizaciones online, soporte para dispositivos personalizados y sensores. Además, aunque no se utilice en este proyecto, tiene soporte por defecto para la detección automática de los dispositivos en Home Assistant.

La configuración inicial supone una red local válida ya configurada y un servidor MQTT funcional, que explicaremos más adelante en el punto 6 de forma detallada.

Lo primero que tenemos que hacer tras haber flasheado el dispositivo y alimentarlo, es conectarnos al punto de acceso que crea de forma automática para que se conecte a nuestra red. Una vez conectado seleccionamos el tipo de dispositivo en el apartado de opciones “Configure module”, guardamos y vamos al apartado de MQTT. En este apartado hemos de configurar la IP del host, el puerto, el usuario y contraseña del bróker y el topic. El nombre del cliente y el topic completo no son obligatorios.

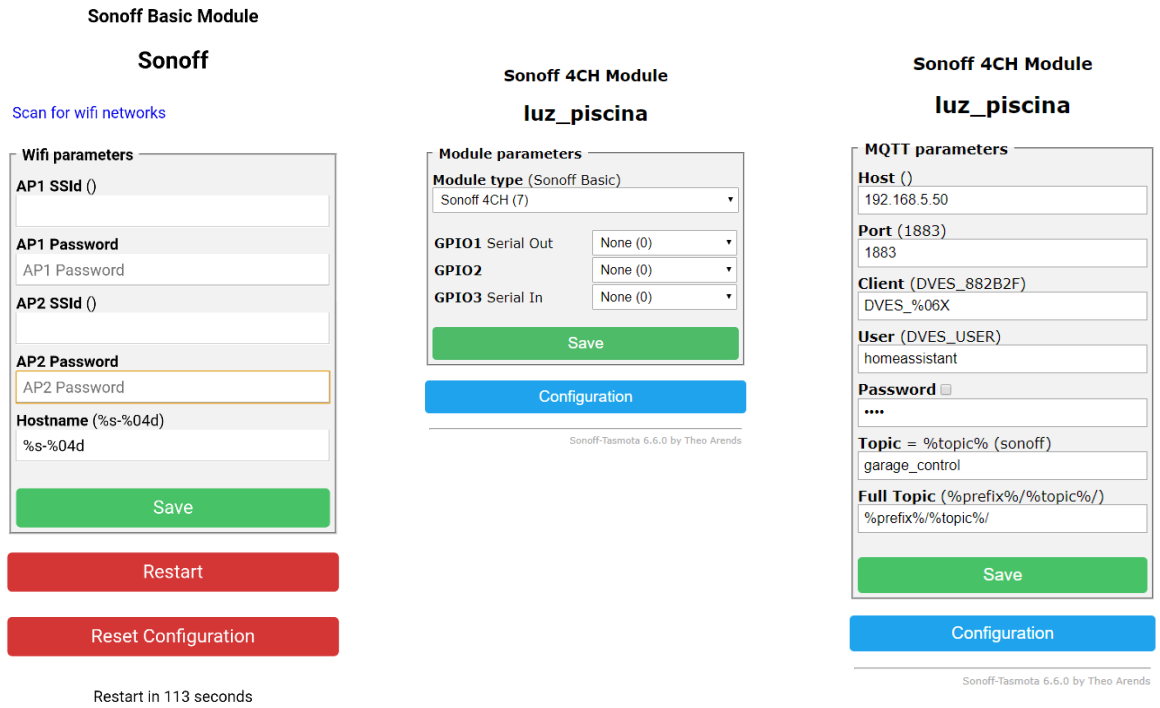


Ilustración 27: Configuración Wi-Fi, tipo de módulo y MQTT

Opcionalmente es posible configurar algunas cosas más que pueden resultar interesantes. Una de las que ha sido introducida en Tasmota en las últimas actualizaciones, son los temporizadores. Ahora es posible configurar hasta 16 temporizadores dentro de un dispositivo con el firmware instalado. Es bastante interesante ya que nos permite proteger el sistema ante fallos. En caso de desconexión total de Home Assistant, podemos activar los temporizadores desde la interfaz, y el dispositivo funcionará de forma autónoma. La interfaz es tal y como se muestra en la siguiente imagen.

Tiene varias opciones de configuración, pero se configura básicamente mediante las siguientes opciones:

- Enable Timers: Activa todos los temporizadores, si están armados.
- Output: Especifica el relé que activa el temporizador.
- Action: Especifica la acción que realizará el relé.
- Arm: Arma el temporizador, su funcionamiento solo es efectivo si “Enable timers” está activo.
- Repeat: Si queremos que Arm no se desactive después de ser armado la primera vez.

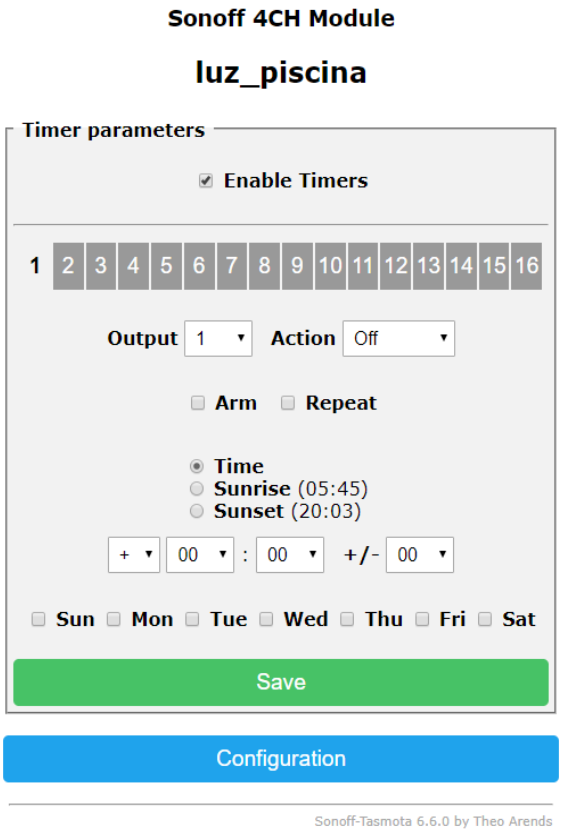


Ilustración 28: Temporizadores en Tasmota

El resto de opciones son para ajustar si queremos una hora y unos días de la semana, o queremos que se active en la salida o la puesta de sol.

En Tasmota también existen lo que en la Wiki llaman “Rules”, o sea reglas. Nos permite una programación básica del dispositivo incluso con condicionales y variables. Actualmente no tiene interfaz, igual que no la tenían anteriormente

los temporizadores. Sin embargo, es interesante para mejorar la seguridad. En caso de desconexión del servidor MQTT durante más tiempo del establecido, el dispositivo activará los temporizadores de forma automática, siendo así totalmente independiente del sistema si se producen fallos mayores.

```
Rule1 on Http#Initialized do Timers 1 endon on Mqtt#Connected do backlog Rule3 0; Rule2 1; Timers 0 endon
Rule2 on Mqtt#Disconnected do backlog Rule3 1; RuleTimer1 450; Rule2 0 endon
Rule3 on Rules#Timer=1 do Timers 1 endon
```

Ilustración 29: Reglas de Tasmota a prueba de fallos

- Regla 1: Activa los temporizadores por defecto. Tras la conexión MQTT, desactiva regla 3, activa regla 2 y desactiva los temporizadores de Tasmota.
- Regla 2: En caso de desconexión del servidor MQTT, activa la regla 3 e inicializa contador a 30 segundos. Se apaga a sí misma.
- Regla 3: Activa temporizadores si el contador llega a 0.

Por último, en lo que a configuración se refiere, existe un apartado que se llama “Configure Other” donde podemos establecer una contraseña de entrada al panel de administración web, cosa que es totalmente recomendable hacer. Además, podemos activar y desactivar MQTT y cambiarle el nombre a cada uno de los relés (el nombre del dispositivo dentro de la elipse azul en la figura, siempre es el nombre del primer “Friendly Name”).

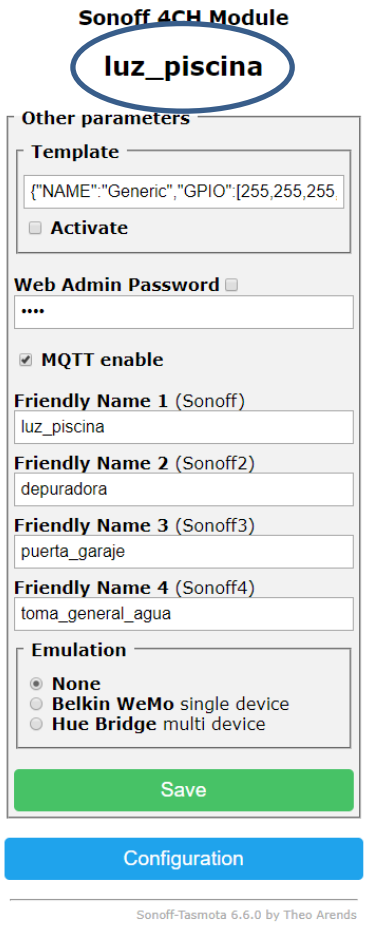


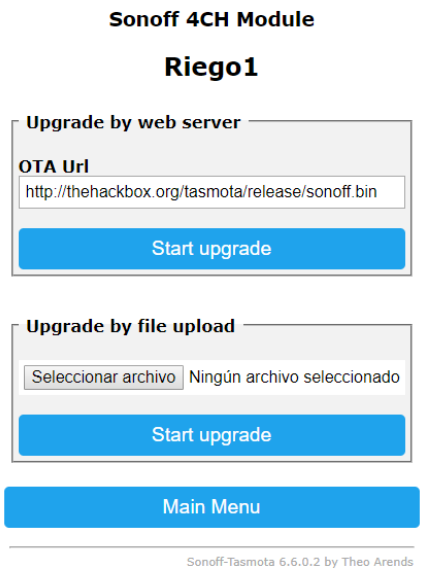
Ilustración 30: Otra configuración disponible en Tasmota

4.1.2. OTA

Cuando comenzó el desarrollo de este proyecto, no había disponible actualización vía Wi-Fi en Tasmota, y era necesario volver a abrir los dispositivos para flashear la actualización vía USB. Más tarde comenzó a ser necesario subir 2 archivos. El primero es el llamado “minimal-firmware”, que es una imagen con la única funcionalidad de subir la nueva actualización, dejando más espacio disponible para escribir el firmware debido a las grandes limitaciones de memoria que tenemos en los dispositivos Sonoff (1MB). Una vez conectado y con el “minimal-firmware” instalado, se actualiza al firmware nuevo. Siempre es recomendable guardar un backup de la configuración por si hay algún problema durante la actualización. Actualmente, el firmware viene además con una dirección web de un servidor que permite actualizar directamente los dispositivos con tan solo pulsar el botón actualizar, haciendo el proceso del “minimal-firmware” transparente para el usuario. La interfaz de usuario disponible en el firmware para actualizar es bastante sencilla y puede verse en la siguiente imagen.

Aunque vía OTA sea la opción más sencilla para actualizar el sistema existen 3 formas para hacerlo:

- Actualización manual mediante conexión USB-UART
- Actualización desde la interfaz web pero de forma manual
- Actualización vía OTA. Aquí el “minimal-firmware” se flashea de forma automática durante el proceso



Las imágenes de Tasmota pueden descargarse desde el propio GitHub de Tasmota, o desde la web donde el firmware obtiene las actualizaciones vía OTA. Existen ediciones compiladas con diferentes versiones de las librerías de Arduino para el ESP-8266. Actualmente, la versión estable aún usa las librerías 2.3.0.

4.2. HomeAssistant

4.2.1. Arquitectura

Para hablar de la arquitectura de Home Assistant, debemos saber que el núcleo de este sistema es el responsable del control de un hogar inteligente. Tiene básicamente 4 partes bien diferenciadas que permiten que esto pueda llevarse a cabo. Aunque por separado pueden no parecer importantes, en conjunto hacen que todo el sistema funcione. Están programadas en Python 3, y cada componente, entidad, automatización y script necesitan que estas cuatro partes estén presentes en el sistema para poder funcionar adecuadamente. Posteriormente explicaremos cada una de estas por separado, pero por ahora, conozcamos las partes indispensables que dan vida al núcleo de Home Assistant.

- Bus de eventos

  El bus de eventos básicamente facilita la escucha y la activación de eventos, y es el centro neurálgico de Home Assistant. Gracias a este bus puede establecerse en términos generales la EDA (Arquitectura dirigida por eventos).

- Máquina de estados

  La máquina de estados monitoriza el estado de cada ente del sistema y dispara un evento “state_changed” cuando el estado de alguna cambia.

- Registro de servicios

  Escucha eventos del tipo “call_service” en el bus y permite a otras partes del sistema registrar servicios.

- Temporizador

  Envía un evento tipo “time_changed” cada segundo hacia el bus de eventos.

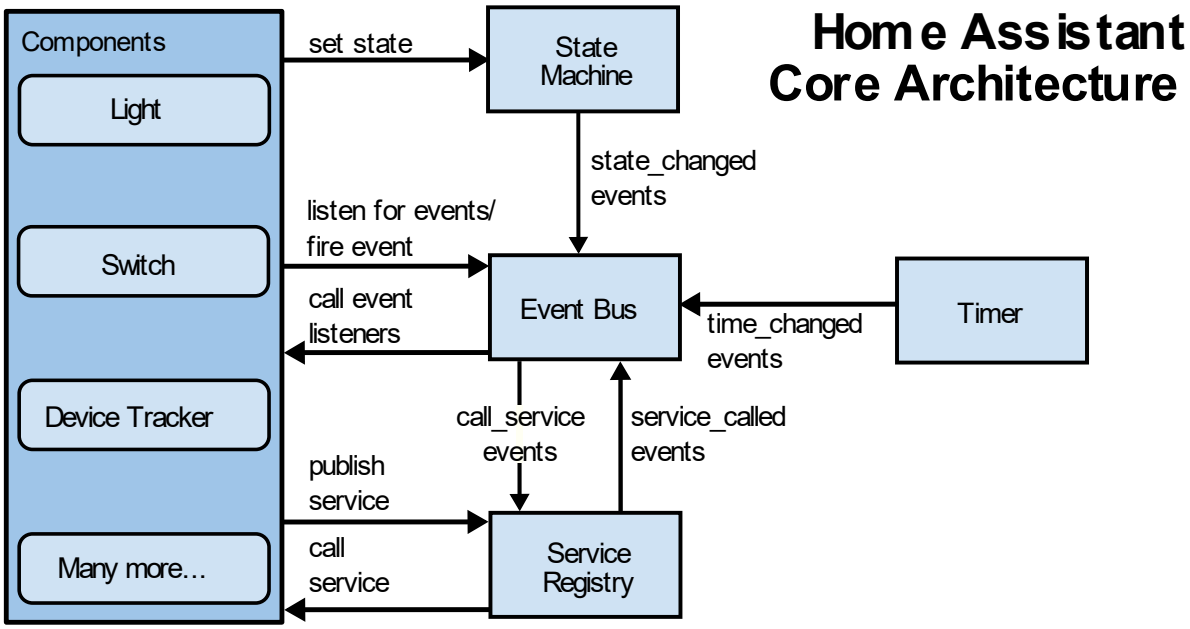


Ilustración 31: Arquitectura del núcleo de Home Assistant

4.2.2. YAML

A pesar de que muchos piensan que *YAML* es un lenguaje de maquetado, en realidad es un lenguaje de serialización de datos. Está diseñado para ser leído y escrito por humanos. Toda su funcionalidad está basada en *JSON* y la conversión entre ambos es realmente sencilla. Sabiendo que la mayoría de las *API*'s en internet generan información en *JSON* y que en prácticamente todos los lenguajes existen multitud de librerías para parsear los datos en *JSON*, es obvia la potencia de *YAML*. La indentación está inspirada en Python pero *YAML* usa doble espacio en vez de tabulaciones. Es el lenguaje elegido por Home Assistant para la configuración de componentes, entidades, automatizaciones, scripts y escenas, o sea, todo lo configurable a nivel usuario.

Es un lenguaje bastante sencillo, y una vez que entendemos su funcionamiento no tardaremos demasiado tiempo en dominarlo. Es una de las razones por las que Home Assistant ha tenido tan buena acogida entre la comunidad de usuarios. La sencillez del mismo, y la intencionalidad por parte de los desarrolladores de Home Assistant de evitarle al usuario final cualquier tipo de interacción con un lenguaje de programación como tal, hace que el sistema entero sea fácil de entender para los que

no son expertos en programación. A continuación se muestra un uso básico del mismo para que entendamos como se establecen los elementos en *YAML*.

Una estructura de datos simple con variables y listas:

```
-  martin:
    name: “Martin D'vloper”
    job: Developer
    skills:
      - python
      - perl
      - pascal
-  tabitha:
    name: “Tabitha Bitumen”
    job: Developer
    skills:
      - lisp
      - fortran
      - erlang
```

Un bloque doblado, se guarda como una sola línea literal, pero cambia saltos de línea por espacios. Se usarán para combinar *YAML* con Jinja2 que conoceremos en el apartado siguiente.

```
fold_newlines: >
            this is really a
            single line of text
            despite appearances
```

Las variables nos serán útiles para los “templates” en Jinja2. Se manejan de la siguiente forma.

```
foo: "{{ variable }}/additional/string/literal"
```

Estos son los conceptos básicos para poder configurar y utilizar *YAML* en Home Assistant. En el caso de necesitar más información sobre el mismo, se recomienda acudir a la web de referencia de *YAML*[1], actualmente se utiliza la versión 1.1 porque Home Assistant incluye el soporte al lenguaje a través de la librería PyYAML, que se encuentra en esta versión.

[1] https://yaml.org/

4.2.3. JINJA2

Es un lenguaje de plantillas para Python, rápido, seguro y ejecutado dentro de un "*sandbox*". Nos permite usar su sintaxis y hacer disponibles algunas variables durante la ejecución. Soporta operaciones matemáticas, lógica y comparaciones. Vamos a ver algunos de los elementos que nos serán útiles en su integración en Home Assistant.

Comprobar estados

```
{{ states('device_tracker.paulus') }}
```

Condicionales

```
{% if is_state('device_tracker.paulus', 'home') %}
  Ha, Paulus is home!
{% else %}
  Paulus is at {{ states('device_tracker.paulus') }}.
{% endif %}
```

Bucles

```
{% for state in states.sensor %}
  {{ state.entity_id }}={{ state.state }},
{% endfor %}
```

Jinja2 puede utilizarse en HomeAssistant desde *YAML*, en todas las Automatizaciones y scripts, y en algunos componentes. Se llama soporte para plantillas y se escribe dentro de un bloque doblado de *YAML*.

Tiene también soporte para controlar tiempo, distancia, formatos, funciones numéricas, filtros y expresiones regulares. Podemos procesar los datos de entrada externos y utilizarlos dentro de *YAML*. Hay más información en el apartado de "templates" de la web de Home Assistant[2].

[2] https://www.home-assistant.io/docs/configuration/templating/

4.2.4. Componentes

Un componente es una especie de "*addon*" para incluir dispositivos y entidades dentro de Home Assistant. En la web oficial existen actualmente casi 1500 componentes, llamados también "Integrations". Estos componentes nos permitirán añadir desde sensores, luces o interruptores, hasta conexiones con servicios y *API*'s de internet. También podemos encontrar componentes no soportados de forma oficial llamados "custom-components". En el punto 6.3 se estudiarán con profundidad los empleados en la realización de este proyecto. Para hacerse una idea no hay más que echar un vistazo al apartado de integraciones de la web oficial.

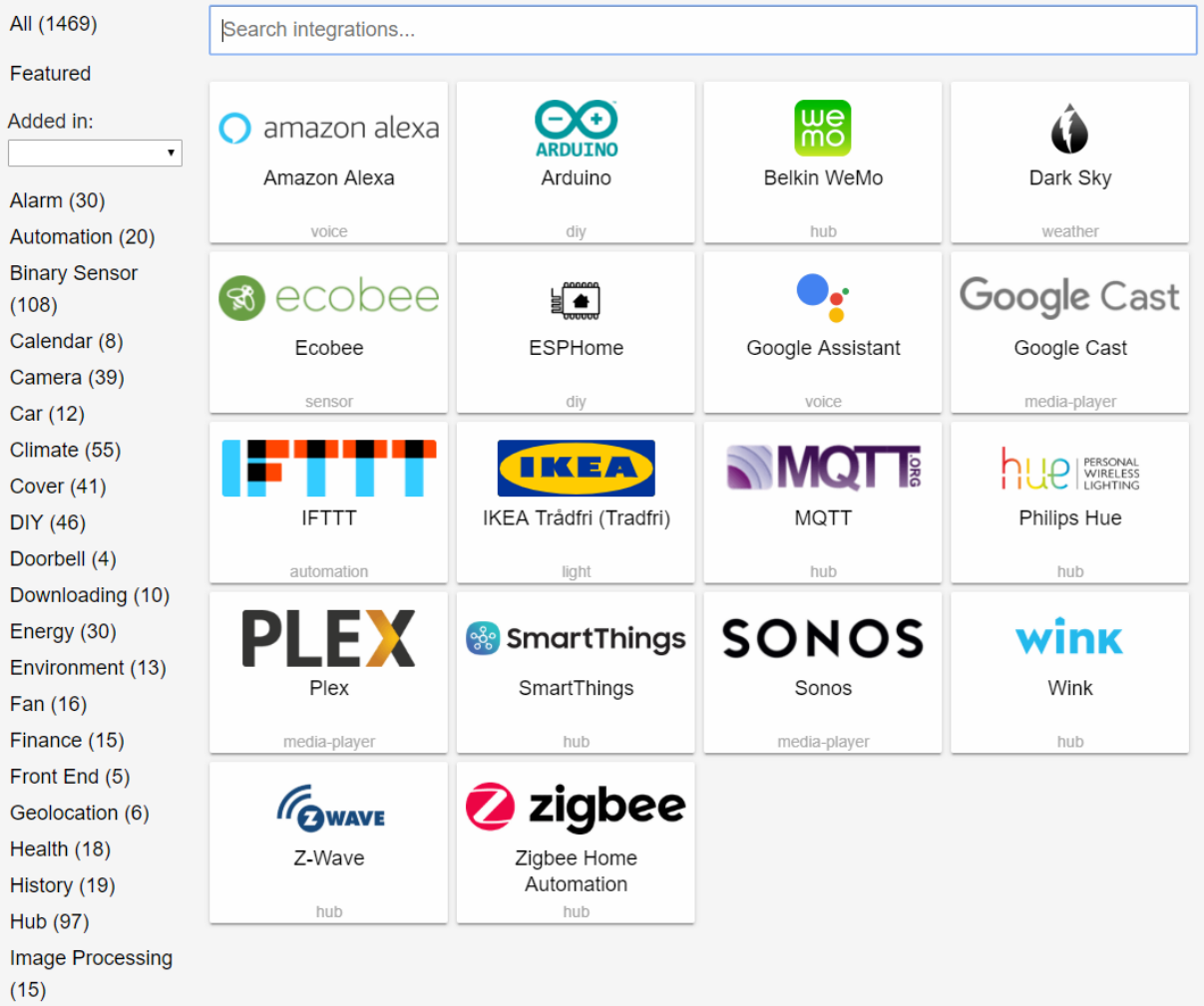


Ilustración 32: Integraciones en la web de Home Assistant

Están separadas por categorías de funcionalidad. Muchas de las marcas de IoT habituales están actualmente soportadas, pero es aconsejable buscar dispositivos de marcas compatibles para utilizarlas en nuestro sistema domótico.

4.2.5. Entidad

Cada uno de los dispositivos que forman parte de nuestro sistema domótico está representado en Home Assistant como una entidad. Se encargan de abstraer el funcionamiento interno del sistema, así no tenemos que preocuparnos de conocer como usa los servicios o la máquina de estados para funcionar. Nos permiten extender un tipo de entidad implementando solamente las propiedades y métodos necesarios para funcionar. Cuando configuramos un nuevo dispositivo, se genera una entidad nueva que podemos personalizar desde *YAML*, requiriendo ésta unos atributos mínimos para funcionar, y dejando otros como opcionales.

4.2.6. Automatizaciones

Una automatización se considera un componente en Home Assistant, aunque puede decirse que es un tanto especial. Pues nos permite utilizarlo para crear disparadores que ejecutan acciones en el sistema. Consta de tres partes.

Disparador

Es el primer atributo requerido de una automatización. Describe los eventos que deberían dispararla. Por ejemplo, cuando alguien llega a casa, cuando una habitación alcanza una temperatura determinada, o cuando una luz ha sido activada.

Condición

Son comprobaciones opcionales para limitar el funcionamiento de una regla en casos específicos. Por ejemplo, si queremos que, a pesar de haberse disparado, solo se ejecute la acción en caso de que sea de noche.

Acción

Ocurre cuando se dispara la regla y las condiciones se cumplen. Por ejemplo, los ocupantes de la vivienda están en casa, y es de noche, encendemos las luces.

Opcionalmente podemos utilizar templates de Jinja2 en este componente. Hay más información sobre automatizaciones[3] en la web.

[3] https://www.home-assistant.io/docs/automation/

4.2.7. Scripts

Un script es una secuencia de acciones que ejecutará el sistema. Están disponibles como una entidad, ya que son un componente, igual que las automatizaciones. Además, también pueden incrustarse dentro de las automatizaciones, juntos, podemos hacer prácticamente cualquier cosa dentro del sistema. La sintaxis básica es una lista que contiene acciones. Se entenderá mejor viendo uno de ellos.

```
# Script de ejemplo para visualizar la sintaxis
# Abre la puerta del garaje y notifica a papá
script:
  encender_luz_y_notificar:
    sequence:
      - service: switch.turn_on
        data:
          entity_id: switch.puerta_garaje
      - service: notify.html5
        data:
          message: 'Se ha activado la puerta del garaje'
          target: papa
```

Existen varias opciones disponibles para agregar a la secuencia de un script. Llamar un servicio, comprobar una condición, esperar un tiempo determinado, disparar un evento, etc… Si se necesita más información al respecto podemos encontrarla en la documentación sobre scripts en la web de Home Assistant[4].

Puede que solo se necesiten scripts, pero si el caso de uso lo requiere, combinar este script con una automatización haría posible, por ejemplo, notificar solamente si no hay nadie en casa en ese momento.

[4] https://www.home-assistant.io/docs/scripts/

4.2.8. UI Lovelace

La interfaz de usuario Lovelace lleva funcionando pocos meses en Home Assistant. Sin embargo, ha sido uno de los mayores cambios que este proyecto ha tenido que afrontar. Aunque a día de hoy ya no puede encontrarse documentación sobre la interfaz antigua en la web oficial, aún sigue en funcionamiento. Para configurar la interfaz antigua era necesario crear grupos para las entidades. Aunque no era difícil configurarla, era un engorro porque cada vez que queríamos reestructurar mínimamente la interfaz era necesario un reinicio completo del sistema.

Lovelace permite la modificación de la interfaz con el sistema en funcionamiento, y no solo eso, también permite editarla desde la propia interfaz, a nivel de código *YAML* o mediante interfaz gráfica, dependiendo de la complejidad que queramos implementar.

Se edita mediante un sistema de tarjetas. Hay disponibles desde tarjetas para listas de entidades, hasta para termostatos o luces directamente. También podemos encontrar algunas no oficiales. Es totalmente personalizable, en este proyecto el resultado actual es éste.

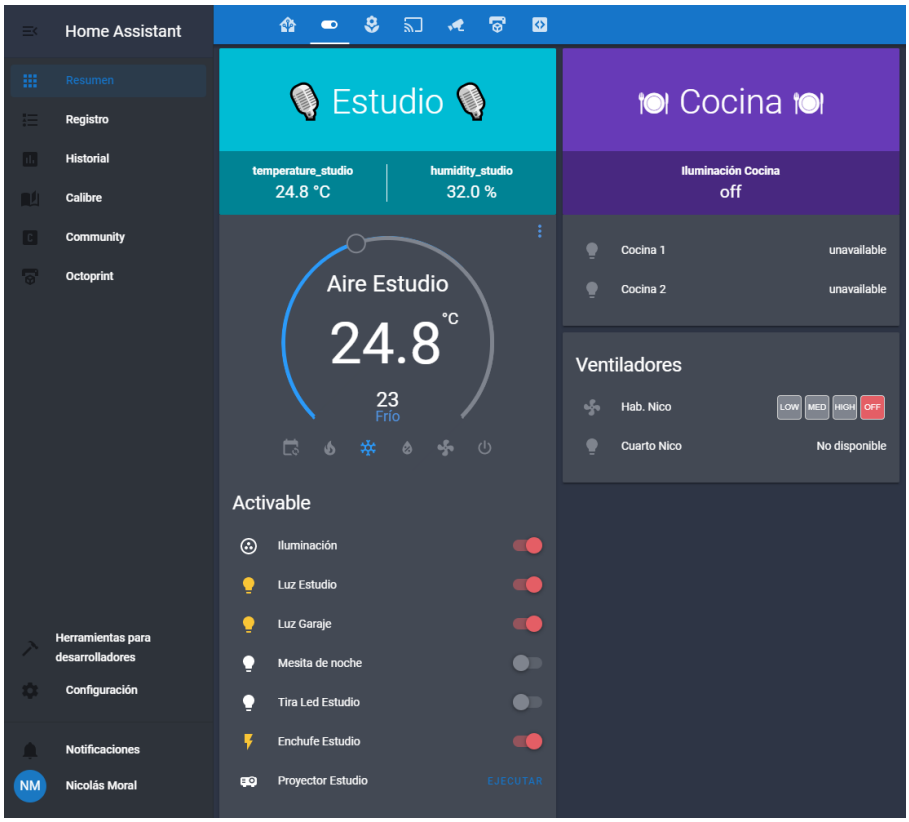


Ilustración 33: Estado actual de Lovelace UI

Veamos un ejemplo del código *YAML* de la habitación Estudio del proyecto. En ella podemos ver varias tarjetas configuradas y funcionales con entidades del proyecto. Usa las siguientes tarjetas.

- Banner-card no oficial para el título y los sensores.
- Termostato para el control del aire acondicionado
- Lista de entidades activables
- Pila vertical que engloba las anteriores para mantenerlas unidas

Hay tarjetas dentro de Lovelace que actualmente no necesitan configuración *YAML*, ya que incorporan un asistente de configuración mediante la propia interfaz incluyendo incluso una previsualización de la tarjeta, sin embargo, el Estudio utiliza como tarjeta padre una pila vertical, que no tiene disponible el editor vía interfaz gráfica. En el proyecto, prácticamente no se ha utilizado este editor para usuarios ya que cuando manejas bien *YAML* es más rápido y más cómodo hacerlo directamente.

```yaml
cards:
  - background: '#00BCD4'
    entities:
      - entity: sensor.temperature_studio
      - entity: sensor.humidity_studio
    heading: "\U0001F399 Estudio \U0001F399"
    row_size: 2
    type: 'custom:banner-card'
  - entity: climate.air_conditioner
    name: Aire Estudio
    type: thermostat
  - entities:
      - group.garage_lights
      - light.luz_estudio
      - light.luz_garaje
      - entity: light.nueva_1
        name: Mesita de noche
      - light.tira_led_estudio
      - switch.enchufe_estudio
      - entity: script.button_projector
        icon: 'mdi:projector'
    id: d63993a5358249ffa7ce41ca9bbf58a8
    show_header_toggle: false
    title: Activable
    type: entities
type: 'custom:vertical-stack-in-card'
```

Ilustración 34: Estudio UI YAML

A pesar de que no se haya utilizado, es una opción estupenda para cualquier persona que quiera empezar con Home Assistant. A medida que el sistema se actualiza se incorporan nuevas tarjetas para añadir a la interfaz mediante este editor más sencillo. El siguiente ejemplo nos permitirá hacernos una idea de cómo funciona esta interfaz.

Ilustración 35: Configurador de tarjeta de Entidades en Lovelace

Además, desde este editor al que podemos acceder cuando seleccionamos el modo edición en el menú de la esquina superior derecha de la interfaz, podemos agregar nuevas tarjetas, y ver cuales hay disponibles oficialmente en Home Assistant. Al intentar agregar una nueva pulsando, desde el modo edición, en el botón con el símbolo “+” de la esquina inferior derecha, aparece un cuadro de diálogo con todas las tarjetas que podemos utilizar.

Ilustración 36: Tarjetas disponibles en el modo edición de Lovelace

5. Sensores

5.1. Meteorológicos

5.1.1. DS18B20

Es un sensor digital de temperatura con un circuito integrado para convertir las medidas analógicas del transductor a digitales. Tiene una tolerancia a error de ± 0.5ºC y según las características del chip, tiene una precisión de hasta 12 bits. Funciona con un voltaje de entrada entre 3 y 5.5 V, y es bastante barato. El rango de temperatura de operación es de -55 a 125ºC. Consta de 3 pines, dos son para alimentación VCC y GND y el último es el de datos.

Emplea un bus de comunicación tipo 1-Wire. Puede alimentarse directamente por la línea de datos, o por una línea adicional mediante el voltaje anteriormente mencionado. Dentro del bus 1-Wire, podemos instalar varios sensores. Además, dispone de una memoria de alarma, que se usa para consultar posteriormente si alguno de los sensores que pertenecen al bus han superado la temperatura, ya sea límite superior o inferior.

DS18B20 es un sensor con una muy buena calidad/precio. Puede encontrarse por 0,45€ en la versión integrada TO-92, y por 1€ en su versión sumergible, que es la utilizada en este proyecto.

Está diseñado para funcionar tanto en ambientes domésticos, como en ambientes industriales mediante 1-Wire creando redes de sensores. En este proyecto se usa para controlar la temperatura externa e interna de la vivienda. Están conectados a un Wemos D1 Mini, y el esquema de conexión está reflejado en la ilustración 37.

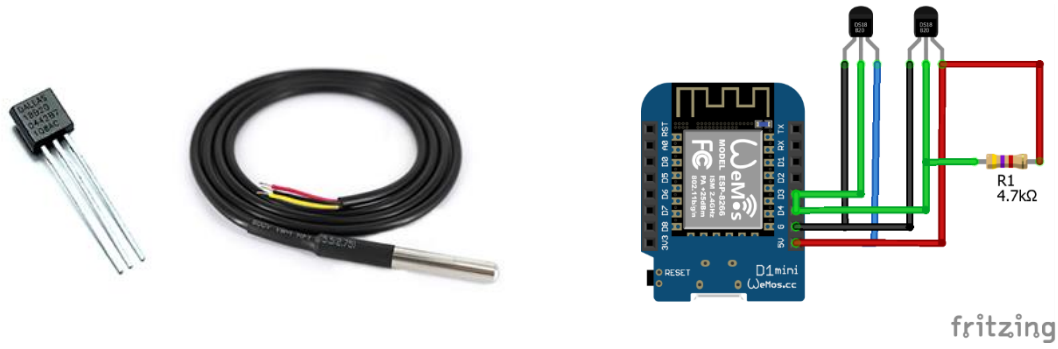


Ilustración 37: Versiones comerciales de DS18B20 y esquema de conexión con Wemos D1 Mini

5.1.2. DHT21 (AM2301)

AM2301 es una versión con cable del sensor DHT21 en una carcasa de plástico grande con soporte para sujeción por tornillo. Es un sensor de bajo coste de temperatura y humedad. Utiliza un termistor para la temperatura y un sensor de humedad capacitivo y envía la señal en formato digital. La frecuencia máxima de lectura es de 2 segundos. Consta de 3 cables, uno de datos, y los dos habituales para alimentar el dispositivo entre 3.3 y 5.5v.

- Rango de medición de la temperatura es de -40 a -80ºC
- Precisión de ±0.5ºC
- La resolución es de 0.5ºC
- Rango de medición de humedad de 0 a 100% RH
- Precisión de 3%
- La resolución es de 0.1% RH

Es uno de los sensores que pueden comprarse por defecto con el módulo Sonoff TH10 para conectarse al puerto minijack. En el proyecto se utiliza para medir la temperatura del sótano en el interruptor de emergencia del sistema.

Ilustración 38: Sensor AM2301

5.2. Caudal

La incorporación del sensor de caudal YF-S201 es una de las medidas de seguridad más útiles en lo que se refiere a domótica. Se ha instalado en paralelo en la toma general del agua de la casa junto a una electroválvula controlada por un Sonoff Dual. La idea es utilizar el sensor para detectar fugas prolongadas de agua en la casa. De esta forma, si la tubería general supera el límite de caudal establecido durante un tiempo determinado, la electroválvula se abrirá, cortando la entrada de agua, evitando así posibles inundaciones en la vivienda.

El sensor se basa en el efecto magnético Hall, tiene una turbina que se mueve al pasar el agua. Una de sus aspas tiene un imán que el sensor magnético reconoce al pasar.

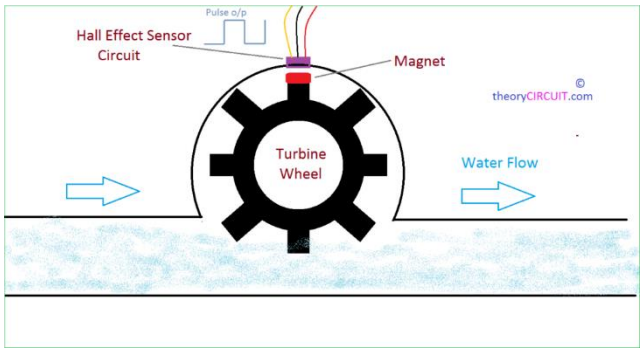


Ilustración 39: Diagrama de funcionamiento de sensor de flujo

Dependiendo del número de veces que gire la turbina, envía pulsos a un Wemos D1 Mini mediante PWM, permitiendo saber cuál es el flujo del agua mediante una entrada digital de la placa. La fórmula utilizada calcula la variación de volumen con respecto al tiempo.

$$Q = \frac{\Delta V}{\Delta t}$$

Basándonos en esta ecuación teórica, podemos despejarla para afirmar que:

$$V = V_0 + Q\Delta t$$

El factor de conversión promedio que proporciona el fabricante para el sensor de 1/2" es f(Hz)=7.5 x Q(L/min) y es el que debemos utilizar en Wemos para calcular el caudal que pasa a través de la tubería. En este caso Tasmota no soporta el sensor, así que debemos programarlo de forma manual tanto el funcionamiento del mismo como la conexión y publicación de resultados al servidor MQTT. No obstante, la programación es bastante sencilla y fácil de encontrar ya implementada en otros sistemas.

5.3. Nivel de agua

Durante los estudios de Grado en Ingeniería Informática que estoy finalizando, trabajamos en una práctica con un sensor de proximidad HC-SR04. Su funcionamiento consiste en un emisor y un receptor de ultrasonidos a una frecuencia de 40KHz.

Es el sensor que suele utilizarse en proyectos de IoT para detectar obstáculos. Y consta de 4 pines, 2 de alimentación, un disparador y un receptor. Las características básicas del HC-SR04 son:

- Alimentación 5V
- Distancia 2-400cm (A partir de 250cm las medidas empiezan a perder fiabilidad)
- Resolución 0.3cm

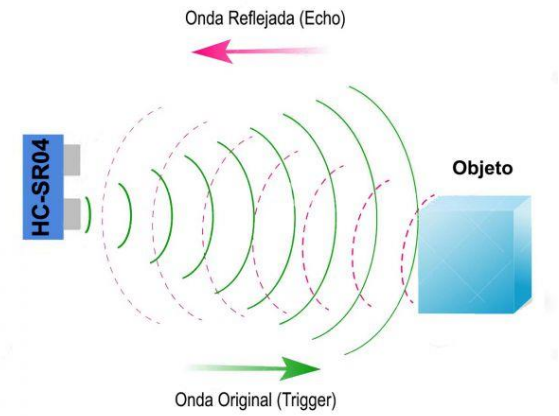


Ilustración 40: Diagrama de funcionamiento del sensor HC-SR04

La idea que surgió en el proyecto al entender el funcionamiento de este sensor, es la de sustituir el sensor de nivel actual del aljibe del sótano, por uno conectado a Home Assistant y mucho más preciso. Tiene una profundidad de 2 metros. Si medimos la distancia hasta el suelo con el depósito vacío, y volvemos a hacerlo con el depósito lleno, obtenemos un medidor con una resolución de 0.3cm en el nivel del agua, mucho más preciso que el que utilizábamos actualmente (resolución de 1000L). En este caso si hay soporte directo desde Tasmota, sin embargo, deberemos aplicar algunos ajustes sobre la lectura original para detectar los litros equivalentes a la distancia proporcionada por el sensor. Puede hacerse de dos formas diferentes, una mucho más sencilla que la otra.

- Directamente desde Home Assistant, ajustando la medida mediante plantillas usando Jinja2 en la lectura del sensor MQTT (Fácil)
- Modificando las fórmulas de lectura del sensor en Tasmota para adecuarlas a nuestra variante de uso (Difícil)

5.4. Presencia

En este apartado vamos a profundizar sobre los sensores de presencia vía BLE (Bluetooth Low Energy). Aunque el sistema utiliza 3 proveedores de localización diferentes, dos de ellos son integraciones de Home Assistant, por tanto, la implementación de estos en el sistema es trivial.

En el desarrollo del proyecto, utilizamos BLE *Beacon*s para detectar la presencia en diferentes sitios. Son unos dispositivos conectados por USB, en este caso a ambos coches. Usamos una aplicación llamada OwnTracks disponible en Android. Esta aplicación permite registrar BLE *Beacon*s para que cuando sean detectados por el Smartphone, se emita un evento a Home Assistant que actualice la localización del usuario a la zona asignada al *Beacon*. Pero antes de explicar cómo utilizarlo, veamos el funcionamiento de esta tecnología.

Tiene un rango aproximado de 80m, y la distancia se mide dependiendo del nivel de señal (TX power), que varía entre -30dBm y +4dBm. Existen 2 protocolos extendidos, uno desarrollado por Apple, i*Beacon*, y otro abierto desarrollado por Google, Eddystone, sin embargo, este último ha sido sustituido por el nuevo NearBee, que no depende del hardware de los dispositivos.

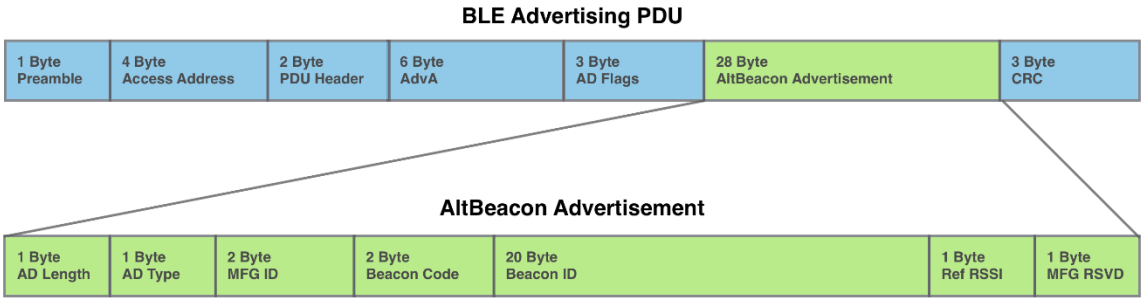


Ilustración 41: Estructura de un paquete BLE Beacon

La tecnología funciona sobre el protocolo Bluetooth y se implementa en el “Data *Payload*”. Los parámetros interesantes son el *Beacon* UUID, que es un identificador único para cada dispositivo, y el RSSI, que es la potencia TX de la que inducimos la distancia hasta el mismo.

El apartado de configuración de *Beacon*s dentro de Owntracks enviaba las peticiones al bróker MQTT actualizando la posición, pero en una actualización de la

aplicación esta función dejó de estar disponible. Tras perder la localización vía BLE en el sistema se necesitaba otra manera de conseguir esa detección de presencia.

Para solucionar esto, se estudió el funcionamiento a nivel de paquetes de red de Owntracks, mediante Wireshark (escáner de red) y tras conocer que es posible simular un *Beacon* desde Android, la decisión que se tomó fue la de realizar ingeniería inversa al protocolo para hacerlo funcionar al contrario. La idea era seguir utilizando la integración de Owntracks en Home Assistant simulando el envío de mensajes al bróker MQTT mediante un servidor en desarrollado en NodeJS, que usaba el hardware bluetooth que se encuentra disponible en la Raspberry Pi 3B+ para detectar la proximidad de los Smartphones y los coches. Así, podíamos saber si algún usuario o los coches estaban en el garaje.

Se diseñó un servidor funcional. Además, este nuevo programa utilizaba un concepto interesante para el posicionamiento de dispositivos, el llamado Filtro de Kalman. Es un concepto popularizado en aplicaciones para estimar la posición de vehículos cuando existen señales de GPS inestables. También se conoce como estimación linear cuadrática.

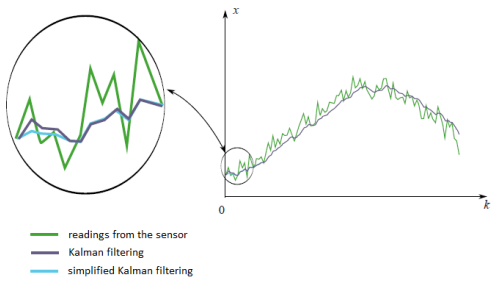


Ilustración 42: Filtro de Kalman

Aunque este servidor ha estado funcionando durante meses. Había un problema principal, solo era capaz de detectar la posición en el estudio y el sótano. Esto limitaba bastante las posibilidades de detección de presencia en otras habitaciones, pero en la versión de Home Assistant 0.83, aparece por primera vez como componente un detector de presencia vía bluetooth a través de los dispositivos Google Home. En este proyecto hay dos de ellos, uno en la cocina y otro en la planta de arriba. Por tanto, junto con el componente de detección de presencia a través del BLE de la RPi ya existente, no tenía sentido seguir utilizando el servidor. Así que actualmente toda la detección de presencia se hace mediante integraciones del propio Home Assistant y sensores MQTT personalizados con plantillas. Se detallará en el punto 6.3.1.

5.5. Sensores virtuales adicionales

Aunque casi todos se consideran integraciones dentro del sistema, es posible programar sensores mediante plantillas. Estos sensores pueden recoger información tanto del interior del sistema como de cualquier *API* de internet. Las posibilidades con el acceso a internet del sistema y la cantidad de *API*'s que hay disponibles son inmensas. Veamos los más relevantes.

Sensor binario

Existen versiones binarias para prácticamente todos los tipos que se explican a continuación. Un sensor binario es aquel que solo tiene dos estados posibles, verdadero o falso. Son, por consiguiente, bastante sencillos de configurar y muy útiles a la hora de programar automatizaciones y scripts dentro del sistema.

Sensor bayesiano

Permite observar el estado de múltiples sensores y usa el teorema de Bayes[5] para calcular la probabilidad de que un evento haya sucedido dado el estado de los sensores observados. Si la estimación supera el umbral de probabilidad, el sensor tomará el valor "on", en cualquier otro caso será "off".

Sensor de filtrado

Aplica técnicas de procesamiento de señales a los estados previos y posteriores de un sensor, y genera un nuevo estado a partir de ellos. Por ejemplo, la siguiente gráfica muestra las lecturas de un sensor de humedad real con respecto a ese mismo sensor filtrado. Hay diferentes opciones de filtros disponibles en la documentación.

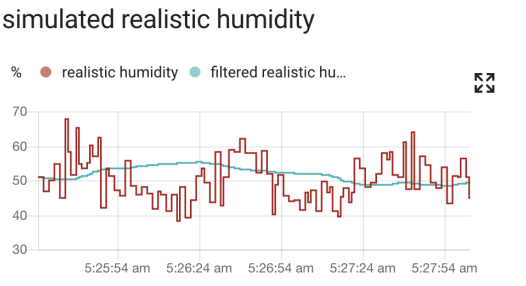


Ilustración 43: Grafica simulando valores de un sensor de humedad real y uno ya filtrado

[5] https://es.wikipedia.org/wiki/Teorema_de_Bayes

Sensor estadístico

Recoge el estado de otros sensores y calcula la media por defecto, estableciendo como atributos medidas estadísticas típicas. Si se asocia a un sensor binario solo cuentan los cambios de estado.

Sensor HTTP

Permiten recibir información desde otros servidores vía HTTP mediante el método POST. Es necesario tener acceso para editar la url de destino y el "*payload*" de datos.

Sensor RESTful

Consume datos desde el "*endpoint*" de una *API* REST. Tiene soporte para métodos GET y POST y probablemente sea el sensor que más información nos permite obtener desde fuera de nuestro sistema, debido a la gran cantidad de *API*'s disponibles en este formato.

Sensor Ping

Realiza un ping a la IP que especifiquemos esperando recibir un "*echo*" mediante peticiones *ICMP*. Puede usarse incluso como detector de presencia de dispositivos. También para saber si hay equipos encendidos, o nuevos equipos en la red.

6. Implementación del sistema

6.1. Creación de la imagen de Hassbian

A la hora de desplegar una instancia de Home Assistant, la web oficial tiene disponibles varias opciones a considerar:

- Hass.io (Sistema Linux reducido para ejecutar Home Assistant)
  - Imagen de Hass.io para volcarla directamente en una SD de Raspberry Pi.
  - HassOS y Docker[6] nos permiten instanciar el sistema Hass.io en un hipervisor. Y de esta forma estará ejecutándose dentro de un sistema de forma encapsulada.
- Hassbian (Distribución Raspbian con Home Assistant instalado en un entorno virtual de Python)
  - Imagen de Hassbian para volcarla en SD para Raspberry Pi.
- Home Assistant (Software disponible en cualquier distribución de Linux que soporte Python superior a 3.6)
  - Instalación a través del gestor de paquetes habitual del sistema (apt, yum…)

Hass.io está realmente limitado. Es seguro y más sencillo de configurar que el resto, sin embargo, a posteriori, nos limita las cosas que podemos hacer en el sistema pues este está totalmente encapsulado. Por ello, partiendo de que iniciamos una instalación desde cero en una Raspberry Pi, se eligió Hassbian.

Para crear la imagen en una tarjeta SD primero debemos descargarla desde el apartado de “releases” del github de Home Assistant.

https://github.com/home-assistant/pi-gen/releases

Una vez descargada la imagen necesitamos un programa con el que poder volcarla en la SD. Es recomendable hacerse con una tarjeta de al menos clase 10. La tarjeta será el disco duro de la Raspberry que alojará Home Assistant, así que nos interesa disponer de velocidades de lectura/escritura decentes.

[6] https://www.redhat.com/es/topics/containers/what-is-docker

Existen varias aplicaciones que nos permiten volcar una imagen en una tarjeta SD. Usaremos balenaEtcher que es de las más sencillas.

https://www.balena.io/etcher/

Una vez descargada e instalada la aplicación, la abrimos y aparece una interfaz con 3 pasos. El proceso tardará dependiendo de la velocidad que soporte la SD.

1. Seleccionar la imagen de Hassbian que acabamos de descargar (no es necesario descomprimirla)
2. Seleccionar la SD en la que queremos volcar la imagen
3. Pulsar el botón “Flash!”

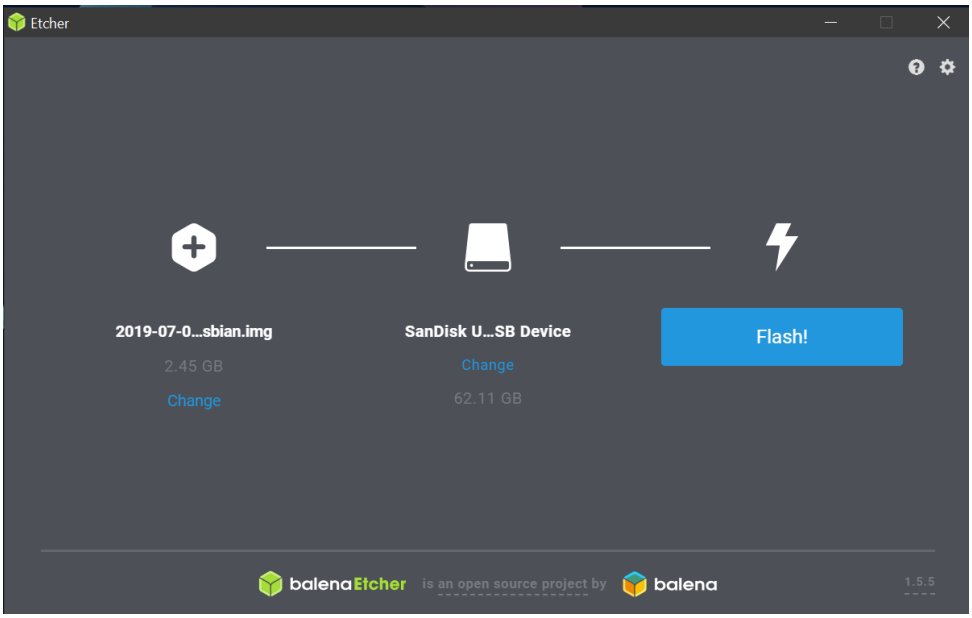


Ilustración 44: Interfaz de balenaEtcher

Por defecto la imagen trae los siguientes extras:

- *GPIO* listos para el uso
- Bluetooth instalado
- *SSH* activado
- Hassbian-config instalado (es una herramienta que permite actualizar el Hassbian y agregar algunos componentes adicionales de forma automática mediante scripts desde un repositorio, como Mosquitto, AppDaemon, o Zigbee2mqtt)

Cuando termine el proceso de volcado de la imagen, tendremos disponible una partición de llamada “boot” que aparecerá como un dispositivo de almacenamiento en el sistema en que nos encontremos. Aunque se recomienda la conexión vía ethernet, es posible usar Wi-Fi. Para activarlo por defecto, creamos un archivo llamado “wpa-supplicant.conf” y establecemos los datos de conexión en este formato.

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

network={
    ssid="YOUR_SSID"
    psk="YOUR_PASSWORD"
}
```

Ilustración 45: Ejemplo de wpa-supplicant

Una vez editados los datos para conectarnos a nuestra red inalámbrica, salvamos el archivo y Hassbian copiará la configuración al arrancar para establecer la conexión.

Durante el arranque, un instalador se ejecutará en segundo plano para descargar la última versión de Home Assistant. Tarda alrededor de 10 minutos, y cuando finaliza, una instancia del sistema estará disponible en la siguiente url. El usuario y la contraseña por defecto son “pi” y “raspberry” respectivamente.

http://hassbian.local:8123

También tenemos disponible el acceso por *SSH* con los mismos datos, aunque se recomienda encarecidamente cambiar el password por defecto.

El usuario por defecto que ejecuta el servidor es homeassistant. Y la configuración está disponible en:

“/home/homeassistant/.homeasistant”

6.2. Configuración inicial de HomeAssistant

Una vez que se conocen las rutas y parámetros básicos para conectarnos al sistema. Debemos saber cómo se estructura la configuración. A excepción del *log* y la base de datos, todos los archivos están codificados en *YAML*.

En la implementación del proyecto, se han estructurado muchos de los componentes en archivos separados para hacer más comprensible la estructura. Los archivos y directorios más importantes serán detallados.

```
pi@hassbian:/home/homeassistant/.homeassistant $ ls
automations.yaml          deps                              input_selects.yaml    scripts.yaml
automations.yaml.save     deps_3.5.3                        input_texts.yaml      secrets.yaml
binary_sensors_mqtt.yaml  device_trackers.yaml              intents.yaml          sensors_mqtt.yaml
blinds_scripts            fans.yaml                         known_devices.yaml    sensors.yaml
cameras.yaml              fcm_android_registrations.conf    lights.yaml           switches.yaml
climates.yaml             groups.yaml                       locks.yaml            themes.yaml
cloud.yaml                home-assistant.log                media_players.yaml    tts
configuration.yaml        home-assistant_v2.db              notifiers.yaml        ui-lovelace.yaml
covers.yaml               home-assistant_v2.db.try          packages              www
custom_components         input_booleans.yaml               remotes.yaml          zones.yaml
customize.yaml            input_numbers.yaml                Roda-YKR-H-102E.yaml
```

Ilustración 46: Ejecución del comando "ls" en el directorio principal de Home Assistant

configuration.yaml

Es el archivo principal de configuración de Home Assistant. En él se establecen los valores de configuración, y en nuestro caso, como va a hacerse una implementación separada en varios archivos, se incluyen los archivos necesarios para todos los componentes utilizados.

secrets.yaml

Es un archivo que no se sincroniza en GitHub porque contiene las credenciales y las url necesarias para que funcionen todos los componentes. Tiene una estructura de llave/valor y se utiliza en cualquier otro archivo con la sintaxis “!secret llave”.

custom_components/

En esta carpeta residen los componentes no oficiales que estemos usando en la implementación. Aunque lo estudiaremos en el apartado 6.8, HACS también usa esta ruta para instalar los componentes que seleccionemos en la tienda.

www/

Este directorio está expuesto al exterior. Contiene los archivos javascript de tarjetas adicionales de lovelace, imágenes y cualquier fichero que necesitemos exponer hacia el exterior.

packages/

Contiene configuraciones de *YAML* adicionales que se insertan en configuration.yaml como si se tratase de un paquete. En nuestro caso se utiliza para todos las automatizaciones y scripts del sistema de riego.

Estos son los ficheros y carpetas del proyecto que deben conocerse. Todos los demás archivos *YAML* tienen el nombre del componente que se implementa en su interior.

6.3. Configuración de Componentes

6.3.1. Sensor de ubicación

El sensor de ubicación es un sensor de plantilla, que unifica todos los proveedores de ubicación que se utilizan en este proyecto para detectar la localización de los usuarios. Está programado combinando *YAML* y Jinja2 para asignar estados adicionales y prioridades dependiendo del rango de distancia que soportan las diferentes tecnologías utilizadas.

- Sensor BLE de Raspberry Pi que detecta si los usuarios están en el Estudio donde reside el servidor central
- Sensor BLE de los dos Google Assistant disponibles en la casa. Uno situado en la cocina y otro en la planta superior
- Sensor Wi-Fi del router Asus, que tiene integración en Home Assistant
- Sensor GPS a través del componente mobile_app que nos envía la información al sistema a través de las coordenadas de los móviles con la aplicación Ariela instalada

Para ver las prioridades y el funcionamiento de la implementación del sensor de plantilla es mejor visualizarlo a través de un diagrama de flujo.

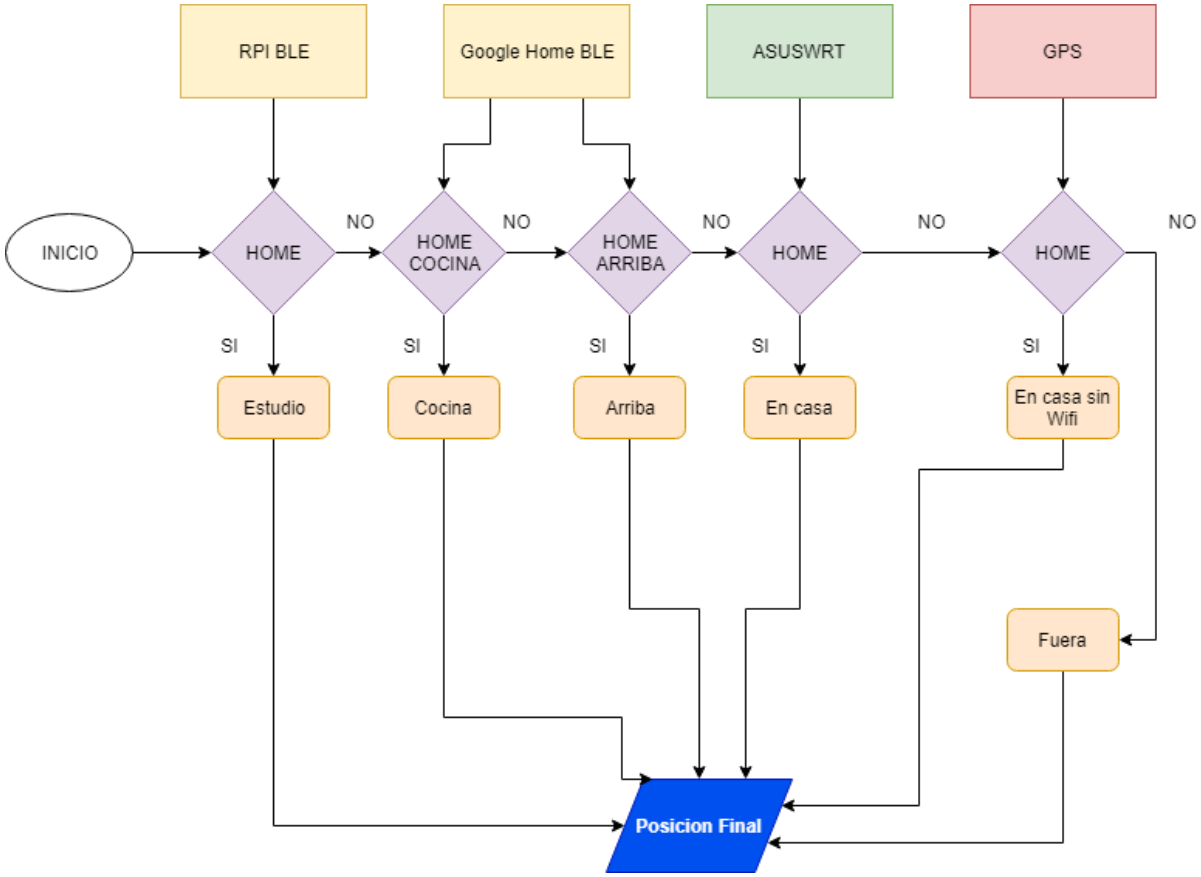


Ilustración 47: Diagrama de flujo del funcionamiento del sensor para la localización de usuarios

6.3.2. Sensores MQTT y OpenWeatherMap

En el apartado 5 del proyecto se explican detalladamente los sensores que han sido utilizados en la casa. Sin embargo, para integrar todas estas magnitudes físicas en Home Assistant desde el firmware Tasmota, es necesario un sensor MQTT que extraiga los datos desde el topic del dispositivo seleccionado.

```
- platform: mqtt
  name: "temperature_outside_1"
  state_topic: "tele/wemos_sensor_1/SENSOR"
  value_template: "{{ value_json['DHT11'].Temperature }}"
  unit_of_measurement: "ºC"
```

Ilustración 48: Sensor MQTT de temperatura exterior

Hay varios sensores de este tipo en la casa distribuidos por diferentes dispositivos con el firmware Tasmota, pero todos los datos se extraen de la misma forma, simplemente es necesario cambiar el topic de referencia y el valor que se extrae del *JSON*.

No solo existen sensores medioambientales, Ariela, nos aporta información de cada uno de los dispositivos móviles que poseen los usuarios del sistema. Sensores de batería, de información de red, de conexiones bluetooth e incluso hasta de imágenes tomadas por las cámaras de los dispositivos.

Otro de los sensores más importantes del sistema es la integración con la *API* de OpenWeatherMap. Aporta información meteorológica bastante completa que en nuestro caso va a utilizarse para evitar que el sistema de riego active la programación si va a llover, ya que en este caso será la lluvia la que riegue el jardín. Además, están implementándose mejoras en el sistema de riego para aumentar o disminuir el tiempo de activación de las electroválvulas dependiendo de la temperatura que haya habido durante el día anterior. Al estar como componente en Home Assistant, la disponibilidad de los sensores en el sistema como entidades es directa, y se configuran los atributos disponibles desde la propia integración.

6.3.3. Interruptores MQTT

Todos los interruptores o relés que controlamos en el sistema están controlados por un ESP-8266 con el firmware Tasmota. Esto nos permite la comunicación MQTT con todos ellos, y aunque Home Assistant tiene disponible el descubrimiento automático en el sistema, el proyecto lo implementa de forma manual pues es más versátil a la hora de configurar las diferentes funcionalidades y restricciones. Aunque si miramos la web de Home Assistant relativa a los interruptores MQTT[7] existen bastante parámetros configurables para el componente, pero no todos son estrictamente necesarios. Sin embargo, dependiendo del interruptor puede resultar interesante establecer parámetros extra para obtener una mejor funcionalidad. Por defecto, en la web de Tasmota aparece una configuración mínima para un interruptor gestionado por su firmware.

[7] https://www.home-assistant.io/components/switch.mqtt/

```
- platform: mqtt
  name: "Zona 1"
  state_topic: "stat/irrigation/RESULT"
  value_template: "{{ value_json.POWER1 }}"
  command_topic: "cmnd/irrigation/POWER1"
  availability_topic: "tele/irrigation/LWT"
  payload_on: "ON"
  payload_off: "OFF"
  payload_available: "Online"
  payload_not_available: "Offline"
  qos: 1
  retain: false
```

Ilustración 49: Interruptor MQTT que activa la Zona 1 de riego

Atributos configurados:

name: Establece el nombre de la entidad (los espacios se sustituyen por "_")

state_topic: Topic para recuperar el estado de los relés desde Home Assistant

value_template: Plantilla para especificar el valor *JSON* a recuperar

command_topic: Topic para enviar comandos de activación al interruptor

availability_topic: Topic para consultar la disponibilidad del dispositivo

*payload*_on: Comando que se envía al activar el interruptor

*payload*_off: Comando que se envía al desactivar el interruptor

*payload*_available: Comando que se envía si está disponible el dispositivo

*payload*_not_available: Comando que se envía si no está disponible

qos: Nivel que se establece como calidad de servicio MQTT

retain: *Flag* que especifica si queremos enviar un comando retenido en el topic. Cuando este u otro dispositivo se suscribe al topic, recibirá el último comando retenido.

Con todos estos valores asignados correctamente tendremos el interruptor MQTT listo para funcionar como una entidad más en Home Assistant.

6.3.4. Iluminación MQTT y Yeelight

La implementación de luces MQTT Home Assistant se realiza exactamente de la misma manera que un interruptor básico, sin embargo, si las luces son RGB o permiten variar la luminosidad, hay que añadir algunos atributos adicionales relativos al brillo, al color, e incluso permite configurar efectos y transiciones. Esto hace que aumente bastante la complejidad para configurar las luces de esta manera. Además, no es sencillo encontrar bombillas que soporten por defecto el protocolo MQTT.

Por ello, en el proyecto se han utilizado bombillas de la marca Yeelight (by Xiaomi) que disponen de un componente directo que permite configurarlas especificando solo la IP. Es una de las marcas de bombillas más conocidas en domótica, y no son excesivamente caras. Una vez conectadas a través de la aplicación oficial de Yeelight para smartphones, solo hay que configurar la IP correspondiente en el componente y estarán integradas en el sistema con una interfaz que soporta colores, transiciones y modificación del brillo de forma directa.

Ilustración 50: Bombilla Yeelight v2 e interfaz de la integración de una bombilla RGB en Home Assistant

6.3.5. Ventiladores con luz

El dispositivo iFan02 que vimos en el apartado de hardware permite la conexión de un ventilador con luz al sistema. Se integra como un ventilador MQTT con plantillas para las velocidades del ventilador y una luz básica MQTT.

La peculiaridad de este dispositivo es que dispone de varios *payload*s que enviar dependiendo de la velocidad que queramos establecer en el ventilador. Existen cinco estados (apagado 0, encendido 4, velocidad 1, velocidad 2, velocidad 3).

Por defecto, la visualización a nivel de interfaz gráfica en Lovelace no está demasiado conseguida, pero existe una tarjeta no oficial para ventiladores que nos ofrece una visualización bastante práctica.

Ilustración 51: Aspecto visual de un ventilador con luz en Lovelace

6.3.6. Riego y piscina

La programación de los circuitos de riego de la casa es probablemente uno de los puntos más complejos del proyecto, la personalización de la interfaz con programación de las cuatro electroválvulas consta de bastantes automatizaciones y scripts para su funcionamiento. Al principio del desarrollo del proyecto la interfaz era completamente distinta a la actual, existían 4 grupos diferenciados, uno para cada zona y eran configurables de forma individual. Sin embargo, tras utilizarlo durante un tiempo la solución que se había implementado no era ergonómica, así que se replanteó la idea desde cero, unificando la configuración de las cuatro zonas en un panel central donde la diferencia entre ellas es simplemente el tiempo de activación.

Todas las zonas son activables de forma manual, por separado, y están limitadas vía Tasmota para que no pueda haber zonas activadas de forma simultánea. Para integrarlas en la interfaz de forma elegante se tomaron unas fotografías de la zona del jardín asociada a cada circuito de riego y usamos tarjetas de Lovelace tipo “Picture Glance”

Existe también un panel de información del ciclo actual, donde puede verse información relativa al último ciclo de riego. Está disponible el tiempo de activado para cada uno de los relés conectados a las electroválvulas, el estado actual del ciclo, la zona activa y el tiempo restante de la zona.

Por último, se ha añadido recientemente a Lovelace un panel con el gráfico de activación de cada una de las zonas.

Con respecto a la piscina, se ha diseñado una interfaz lo más simple posible. Actualmente funciona también sobre una tarjeta “picture_glance” donde podemos activar de forma manual la luz y la depuradora, o seleccionar dos modos de depuración, uno para verano, que comprende el horario de depuración establecido durante la temporada de baño, y otro para invierno, que reduce las horas de funcionamiento solo para el mantenimiento del agua en buen estado. La depuración utiliza dos scripts que activan el relé durante un tiempo determinado, que están asociadas a dos automatizaciones que se disparan de forma distinta dependiendo del modo que hayamos seleccionado mediante la interfaz.

Visualmente, las tarjetas “picture-glance” disponibles en la interfaz Lovelace, tienen un muy buen acabado. Se han utilizado tanto para la configuración relativa a la depuradora como para activar el riego de forma manual.

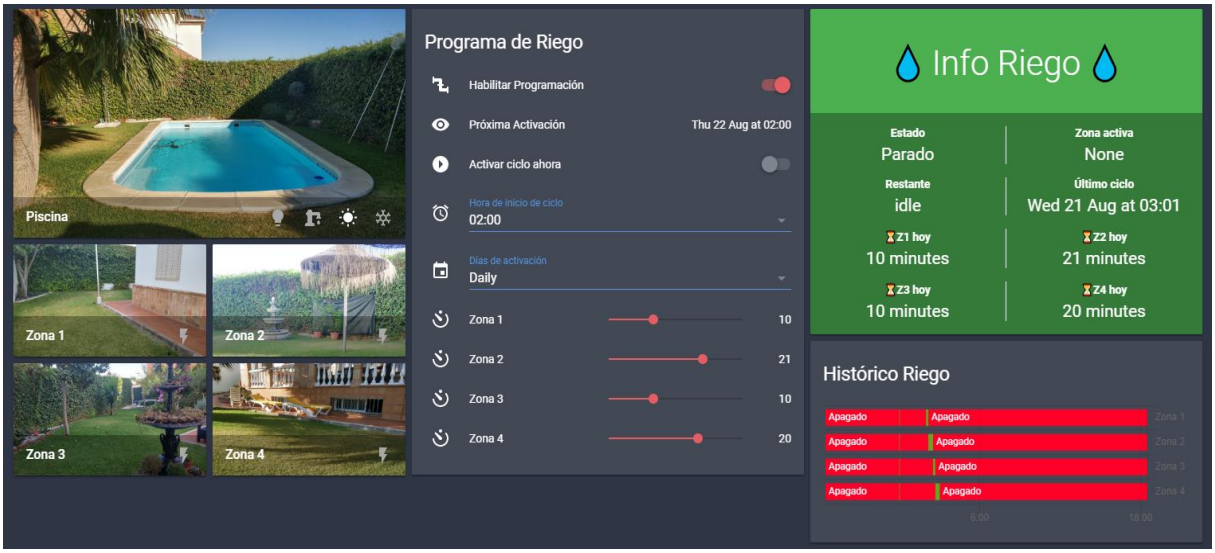


Ilustración 52: Sección del jardín en Lovelace

6.3.7. Persianas

En la casa en la que se implementa Home Assistant, existen 4 persianas motorizadas. Dos grandes en el salón, una en la terraza superior y otra en una habitación.

Las persianas tienen asociadas automatizaciones y scripts para controlarlas de forma automática tratando de mejorar la conservación de la temperatura del interior de la casa.

Programación de persianas:

- Durante el invierno, cuando la temperatura exterior sobrepasa el umbral superior establecido, todas suben para permitir la entrada de calor en el interior. En cambio, si la temperatura desciende del umbral inferior, se bajan de forma automática para conservar el calor interior.
- Durante el verano funciona a la inversa, si la temperatura está por encima del umbral superior, bajan todas las persianas para conservar la temperatura interna. Si la temperatura baja del umbral inferior, se suben para dejar entrar el frío en la casa.
- Además, en ambos periodos estacionales, siempre que no se infrinjan las normas anteriores, las persianas se abren 3 horas después del amanecer, y se cierran al anochecer.

Hay que tener en cuenta que en este caso no debemos de poner la placa Sonoff en paralelo con los interruptores, si no que debemos conectar los interruptores en la placa. Aunque los interruptores por defecto para persianas motorizadas están enclavados de forma mecánica (los motores funcionan con un neutro y doble fase), al realizar el paralelo entre el Sonoff y los interruptores, cabe la posibilidad de que habiendo dejado activado un interruptor físico, se active la fase contraria en el Sonoff. Para que esto no se produzca en ningún caso, se conectan los interruptores desde el propio VCC a 3.3V del Sonoff a un *GPIO* disponible, así pasarán a ser botones en el firmware Tasmota. De esta forma, y activando a través de la consola el interbloqueo de relés[8], nunca podrán activarse dos fases de forma simultánea.

[8] https://github.com/arendst/Sonoff-Tasmota/wiki/Commands (InterLock)

6.3.8. Cámaras

Aunque ahora pueden integrarse las cámaras por defecto en el sistema como diferentes componentes dependiendo del protocolo que usen (http, onvif, rtsp), en versiones anteriores de Home Assistant, no era posible. Por ello cuando empezó el desarrollo, se pensó en una segunda Raspberry Pi que sirviese sistema de videovigilancia, para generar flujos de video compatibles con Home Assistant. El servidor de videovigilancia por antonomasia es ZoneMinder. Tiene un componente para integrarlo directamente en Home Assistant, pero no está pensado para funcionar en dispositivos con una potencia limitada. El uso de *ffmpeg* para cargar los tres flujos de vídeo en nuestro sistema, hace que la Raspberry Pi 3B utilice más de un 90% del procesador de forma continua para la gestión de los flujos.

Además, el paquete por defecto disponible en los repositorios estándar de Raspbian no tenía soporte para el procesamiento a través de GPU, lo cual hacía esta opción prácticamente inviable. Después de recompilar manualmente integrando todo lo necesario para hacer funcionar *ffmpeg* con soporte para la GPU de Raspberry Pi, los resultados obtenidos tampoco fueron satisfactorios.

Una vez descartado el uso de ZoneMinder, se buscó otro servidor local de video vigilancia que tuviese soporte para flujos en H264, ya que este formato es mucho más ligero que el anterior, y el consumo de CPU baja drásticamente (del 90% a menos del 10%). Tras valorar diferentes opciones, elegimos Shinobi, es un software de código abierto, muy nuevo y con una alta participación en su repositorio oficial.

Aunque lo hayamos configurado y estuviese funcionando durante un tiempo incluso evaluando pruebas con ALPR[9] y cascadas de Haar para reconocimiento de matrículas y objetos (funcional, pero inviable por la reducida potencia de RPi), se ha obviado extender la explicación, pues actualmente, los tres flujos de video funcionan de forma directa integrados en Home Assistant con el componente genérico para cámaras.

Se migró recientemente desde el componente *ffmpeg*, debido al consumo y a las nuevas características implementadas por el componente genérico, entre ellas la

[9] Advanced License Plate Recognition (Reconocimiento avanzado de matrículas)

exposición de las entidades mediante Google Assistant y el soporte para la visualización remota en dispositivos Chromecast.

Por privacidad, todas las cámaras tienen bloqueado por defecto el acceso externo a internet desde el router. De esta forma, solo pueden enviar el flujo de vídeo al servidor donde se encuentra Home Assistant. Además, la cámara situada en el interior de la vivienda tiene agregado un Sonoff Basic, que la activa mediante una automatización solo cuando todos los usuarios del sistema están fuera de la vivienda.

Con esta integración directa, desechamos la necesidad de otra Raspberry Pi, con su respectiva fuente de alimentación y tarjeta SD. Por tanto, abaratamos costes sin perder funcionalidad. Si en algún momento es necesario un uso más avanzado a nivel de CCTV, la mejor opción existente es la mencionada anteriormente, mediante Shinobi.

Las cámaras se integran en la interfaz de Lovelace mediante unas tarjetas llamadas “picture_entity”, pero si se incluye alguna entidad adicional se hace con “picture_glance”. Actualmente el stream de las cámaras en Home Assistant se ejecuta en tiempo real y produce aproximadamente una imagen cada dos segundos.

Si nos fijamos en la cámara interna, se ha agregado un acceso al interruptor MQTT de la cámara por si necesitamos activarlo en cualquier momento sin necesidad de que se disparen las automatizaciones que lo activan de forma automática.

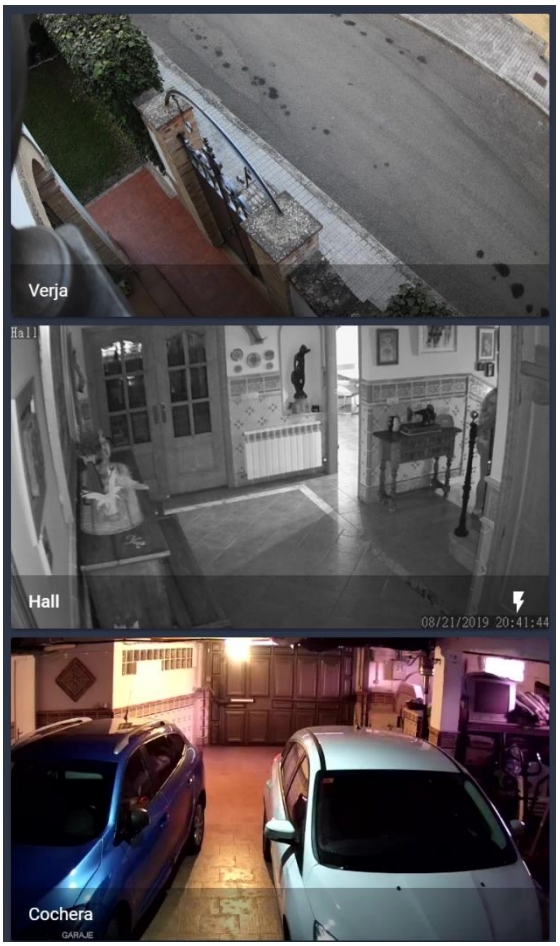


Ilustración 53: Cámaras en Lovelace UI

6.3.9. Aire Acondicionado

Existen numerosos complementos para controlar aires acondicionados en Home Assistant. Hay bastantes marcas compatibles, pero en caso de no poseer ninguna de ellas siempre puede obtenerse un termostato que soporte *HVAC* y que sea compatible con el protocolo MQTT. En cambio, la implementación del funcionamiento del aire en este proyecto se ha hecho a través de un emisor/receptor IR de Xiaomi.

El Split de aire acondicionado que hay instalado en el estudio no es compatible con Wi-Fi, pero si tiene un mando a distancia infrarrojos para controlarlo. Existe un complemento en Home Assistant para integrar el controlador remoto por infrarrojos de Xiaomi. El principal problema para configurar este dispositivo es que necesitamos un token que nos da acceso a la *API*. Para obtenerlo existen varios métodos diferentes:

- Instalar una versión antigua de la aplicación de Xiaomi, Mi Home en Android y mediante acceso root recuperar la información exportando la base de datos de la aplicación.
- Instalar la aplicación en iOS y recuperar el token a través de un visor de copias de seguridad compatible con iOS.
- Crear una máquina virtual de Android mediante BlueStacks y usar un software llamado BlueStacks Tweaker para acceder al sistema de archivos
- Instalar un programa de nodeJS mediante el gestor de paquetes npm llamado “miio” y ejecutarlo en la terminal mediante la orden “miio discover”

Por sencillez, se ha usado la cuarta opción, que es la última que ha sido añadida. Esta aplicación escanea los dispositivos en red y produce una salida como la siguiente, de la que nos interesa el token que guardaremos como llave en el archivo secrets.yaml.

```
Device ID: 48765421
Model info: zhimi.airpurifier.m1
Address: 192.168.100.9
Token: token-as-hex-here via auto-token
Support: At least basic
```

Ilustración 54: Salida en consola que produce la ejecución de miio discover

6.3.10. Seguimiento de envíos

Para realizar el seguimiento de envíos en el sistema, se utiliza un complemento que integra la *API* de AfterShip. Es una aplicación compatible con dispositivos móviles que hace el seguimiento de paquetes de muchas compañías de transportes. Para poder utilizar el complemento debemos crear una cuenta en AfterShip y generar una clave de *API*. Con esta clave, que también reside como llave en secrets.yaml, tendremos vinculada una nueva entidad al sistema que muestra el seguimiento de todos los paquetes que agreguemos a través de la aplicación móvil.

Si lo combinamos con una tarjeta de AfterShip no oficial disponible para la nueva interfaz Lovelace, conseguiremos un aspecto como el que se muestra en la imagen.

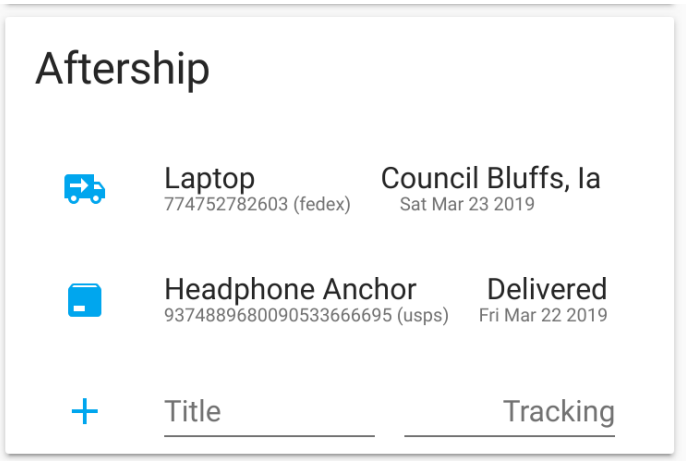


Ilustración 55: Tarjeta no oficial de AfterShip compatible con Lovelace

6.3.11. Wake On Lan

El protocolo Wake On Lan[10] permite, si lo hemos habilitado previamente en la *BIOS*/*UEFI* de la máquina, despertar el ordenador a través del llamado "Magic Packet". Es un paquete de red que funciona en la capa 2 del modelo OSI. El ordenador dejará alimentada la tarjeta de red seleccionada esperando el paquete, si lo recibe, se producirá el inicio del sistema.

Home Assistant dispone de un complemento que soporta el protocolo WoL. Podemos configurar todos los dispositivos compatibles simplemente con la dirección MAC de la tarjeta de red, y obtendremos una entidad tipo interruptor en el sistema.

[10] https://www.testdevelocidad.es/redes/wake-on-lan-utilizarlo/

6.3.12. Reproductores multimedia

En este proyecto existen tres componentes que integran dispositivos multimedia en el sistema.

Google Cast

Aunque se implementa de forma manual mediante un archivo *YAML*, a día de hoy Home Assistant puede descubrir los dispositivos de forma automática mediante la integración "discovery[11]". Este complemento es compatible con todos los dispositivos Chromecast y Google Home disponibles. Existe una tarjeta oficial en Lovelace para integrarlos en la interfaz, aunque también existen tarjetas no oficiales con características interesantes que podemos descargar desde la tienda HACS.

Sonos

Existe también soporte directo para dispositivos de la marca Sonos. En la casa había uno de estos, y la forma de incluirlo es similar a Google Cast, teniendo también disponible el descubrimiento automático.

Spotify

También puede integrarse el control directo de una cuenta de Spotify mediante la *API* oficial, aunque tiene una configuración más compleja y necesita que Home Assistant esté funcionando sobre *SSL*. De esta forma, podremos controlar la reproducción de Spotify de forma general como si de otro reproductor se tratase, independientemente del dispositivo en el que se esté escuchando en ese momento. Si tenemos este complemento activado, podemos enviar listas que existan en la cuenta de Spotify mediante algunos scripts.

Debemos acceder a la web de desarrolladores de Spotify, crear una clave de *API* para el sistema domótico y establecer las *URI*s de redirección de Spotify hacia nuestro sistema. Toda la información relativa al proceso está disponible en la documentación del componente.

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

[11] https://www.home-assistant.io/components/discovery/

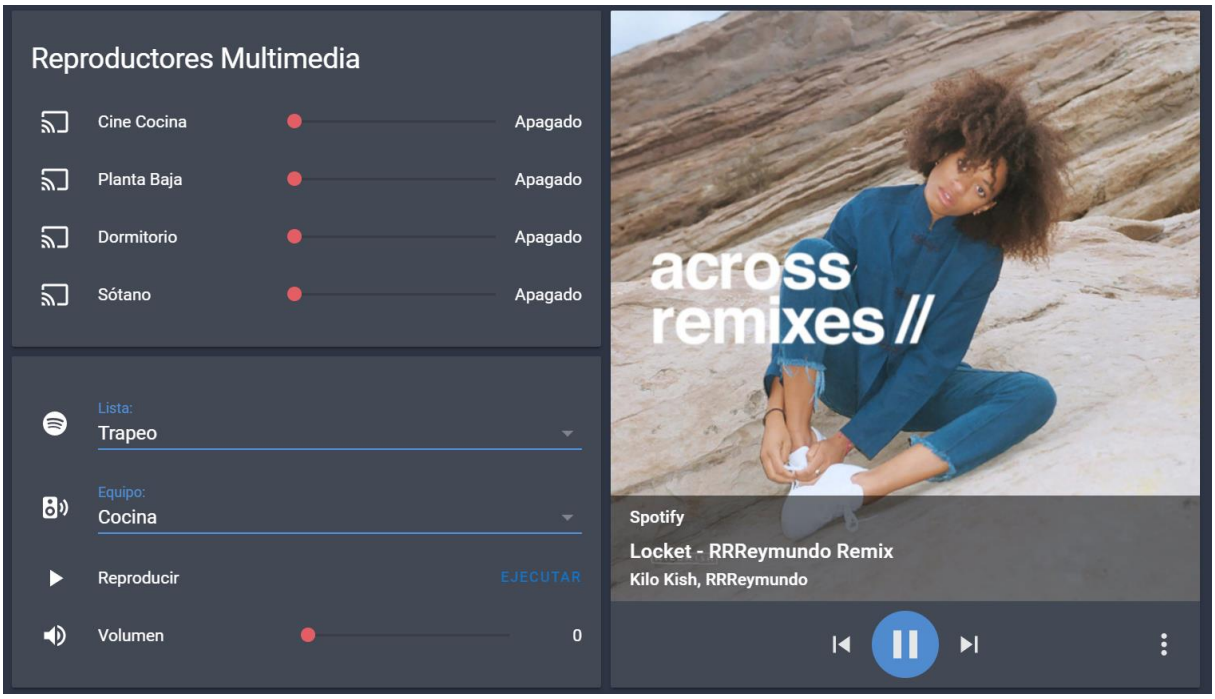


Ilustración 56: Vista de reproductores multimedia en Lovelace

Además de tener el control de los reproductores multimedia conectados al sistema, Home Assistant permite, si activamos la integración de *TTS* (Text to Speech), enviar frases desde scripts y automatizaciones para que se reproduzcan a través de los equipos. Esto es realmente potente para informar a los usuarios de la vivienda de los eventos que suceden.

6.3.13. Información de sistema y red

Es interesante tener varios sensores relacionados con el funcionamiento del sistema, la red y las versiones de HA disponibles.

Puede hacerse a través de diferentes complementos, uno de los más potentes es el “Command Line Sensor”. En la casa de este proyecto se ha diseñado una interfaz que utiliza varios de ellos.

Información de Sistema

Mediante el complemento “System Monitor”, se obtienen sensores para visualizar el porcentaje del uso del disco, la RAM libre disponible, el porcentaje de uso de RAM y el uso del procesador.

Información de Red

Para obtener información sobre la red utilizamos la integración de *AsusWRT* que nos aporta tres sensores. Uno para obtener la velocidad de bajada, otro para la de subida y un último que indica el ping.

En este caso, hay que ser conscientes de que, en una Raspberry Pi, la tarjeta de red ethernet transmite a través del bus de datos USB. Así que, si tenemos conexiones de fibra de alta velocidad, tanto la bajada como la subida están limitadas a la transferencia máxima soportada por el bus del puerto USB. Es literalmente un cuello de botella.

Versiones de Home Assistant

Existe un componente llamado “version” que crea entidades relativas a la versión de Home Assistant. Para este sistema se crean dos, uno para saber la versión actual instalada, y otro para obtener la última versión disponible en el servidor. Con ambos sensores disponibles, es posible crear una automatización que nos notifique si la versión del servidor es superior a la instalada, es decir, cuando existe una actualización de Home Assistant.

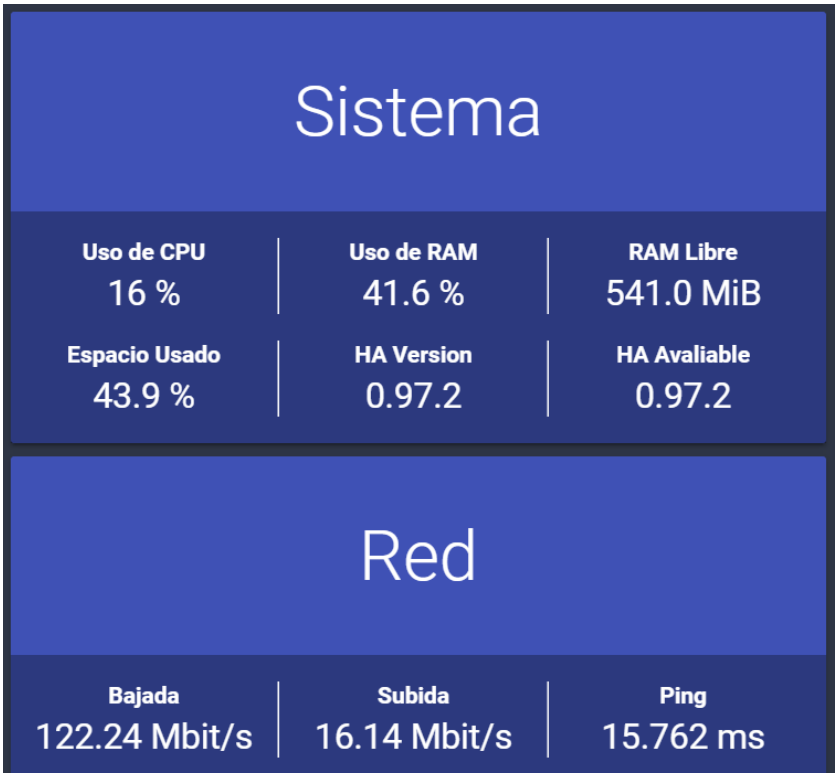


Ilustración 57: Sensores para la monitorización del sistema

6.3.14. Integración con Octoprint

Octoprint es un servidor de monitorización y control para impresoras 3D instalable en Raspberry Pi. Hay una impresora Anet A8 en la casa donde se encuentra nuestro sistema domótico conectada a Octoprint, y Home Assistant tiene un componente disponible para obtener información de este sistema. Esta RPi con el servidor de Octoprint instalado, tenía conectados dos relés a sus *GPIO*. Cuando se integraron los sensores mediante el componente compatible con este servidor, también se implementó en el mismo servidor un software para exponer los relés al protocolo MQTT y poder activar tanto la alimentación como la iluminación de la impresora desde nuestro sistema con entidades tipo interruptor.

Además, la RPi de la impresora tenía conectada una cámara mediante el puerto para cámaras. Esta cámara también se ha añadido al sistema. Con la cámara, los sensores, y los interruptores, se ha creado una interfaz de control completa para una impresora 3D.

En esta parte de la interfaz, introducimos dos nuevas tarjetas. Una no oficial para la barra de progreso que controla el porcentaje de impresión del tipo “bignumber_card” Y dos más tipo “gauge” para visualizar las temperaturas del extrusor y de la cama caliente.

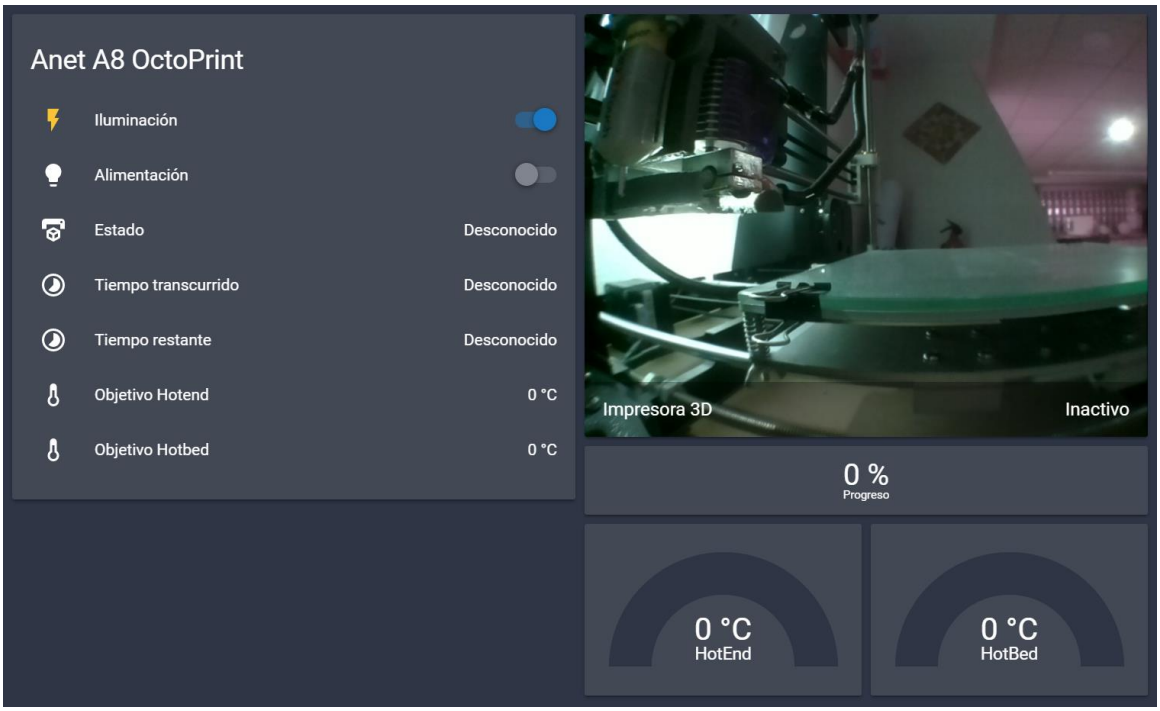


Ilustración 58: Interfaz para control de OctoPrint en Lovelace

6.4. Amazon Dash Buttons

Aunque actualmente no se encuentran a la venta, la empresa vendía unos botones que se configuraban mediante Bluetooth desde la aplicación oficial de Amazon. Pueden conectarse a la red Wi-Fi y están diseñados para comprar el producto que especifiquemos mediante la APP solo con pulsar el botón.

Esta información puede no resultarnos relevante hasta que estudiamos el funcionamiento interno del dispositivo, pues, a fin de cuentas, no es más que una placa que se conecta a nuestra red Wi-Fi y envía un paquete de red a los servidores de Amazon. El Dash Button permanece siempre apagado y solo se despierta cuando lo pulsamos.

Lo primero que debemos hacer una vez que hemos configurado los botones, es prohibir el acceso a internet de todos ellos para que no puedan hacer compras. Una vez que están bloqueados en el firewall del router, existe un servidor en Python llamado "amazon-dash[12]" que detecta mediante la MAC, cuando uno de estos dispositivos intenta enviar un paquete. Este servidor escucha las peticiones enviadas que, aunque no salgan fuera de la red local, es suficiente para saber que el botón se ha accionado. La configuración es bastante sencilla, y el servidor permite la ejecución de comandos o llamadas a URL's cuando detecta la petición de uno de estos botones. Además, es compatible con Home Assistant y puede enviar eventos al sistema. Estos "Dash Buttons" estaban a la venta por 1.99€, y pueden usarse como interruptores Wi-Fi totalmente funcionales, se usan para activar los circuitos de riego. Es necesario crear un token de largo acceso[13] para hacer peticiones al sistema.

```yaml
settings:
  delay: 10
devices:
  FC:65:DE:E9:63:BE:
    name: Haribo
    homeassistant: http://127.0.0.1:8123  # Address to the hass server
    event: DASH_IRRIGATION  # Event name to send
    data: '{"zona_riego": "1"}'
    access_token: long_live_access_token
```

Ilustración 59: Botón Amazon Dash y YAML para enviar el evento a HomeAssistant

[12] https://github.com/Nekmo/amazon-dash
[13] https://www.home-assistant.io/docs/authentication/

6.5. Ariela (Android APP)

Es actualmente el mejor cliente de Home Assistant para Android. Aunque la interfaz está diseñada mediante “responsive design” y se adapta perfectamente a todo tipo de dispositivos, utilizar esta aplicación aporta algunas características adicionales que son importantes para el sistema.

- Sensores del dispositivo móvil mediante la integración “mobile_app” (información bluetooth, wifi, contador de pasos, batería, cámaras y algunos más que no son utilizados actualmente en este proyecto)
- Soporte para notificaciones *PUSH* enriquecidas (imágenes, cámaras, botones e incluso audio)
- Soporte para la localización GPS del Smartphone

Tiene un proceso de configuración totalmente guiado en su primer inicio. Hay más información disponible en la web oficial de Ariela[14].

Si no se necesita disponer de estas características mencionadas, es posible acceder a Home Assistant desde Chrome e incluso recibir notificaciones básicas a través de FCM (Firebase Cloud Messaging).

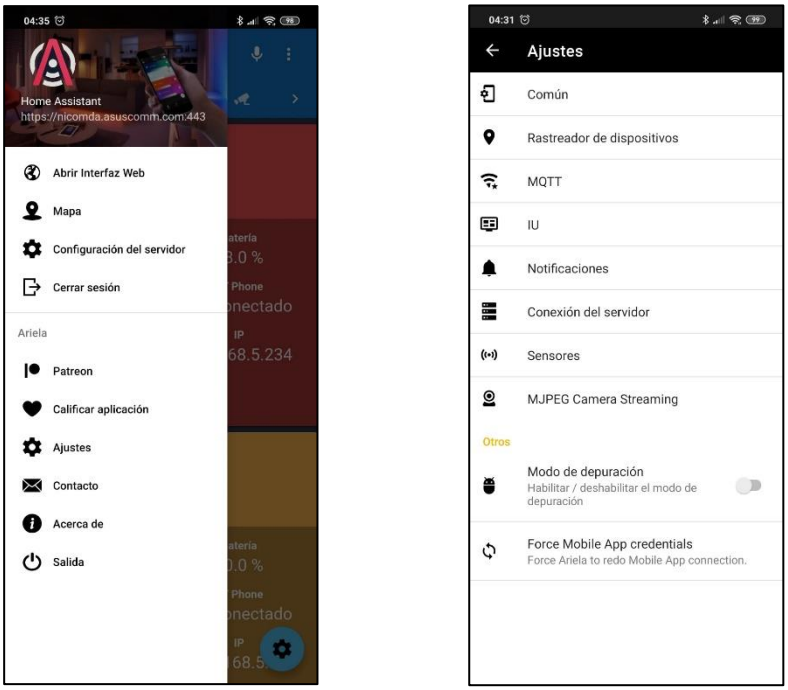


Ilustración 60: Menú lateral y pantalla principal de ajustes en Ariela

[14] http://ariela.surodev.com/

6.6. Notificaciones

Si accedemos a la sección de integraciones de la web de Home Assistant y filtramos todas las que son para notificar, hay más de 60 complementos diferentes para lanzar notificaciones desde Home Assistant.

Para hacerlo lo más transparente posible y evitar la instalación de más aplicaciones en los dispositivos de los usuarios, se notifica directamente mediante el complemento “mobile_app” que se configura de forma automática cuando iniciamos sesión mediante Ariela. Así la única configuración necesaria para notificar dispositivos es añadir las llamadas al servicio de notificación desde las automatizaciones y scripts que deseemos.

Se ha utilizado también el componente “notify group”, que permite unificar los dispositivos de los usuarios en un único notificador. De esta forma, cuando hagamos una llamada al servicio “notify.family”, la notificación llegará a los dispositivos de todos los miembros de la familia.

Veamos cómo se implementa la llamada a estos servicios desde una automatización en *YAML*. Para el ejemplo, usamos una de las automatizaciones reales del sistema que se encarga de notificar a los usuarios cuando la zona de riego pertinente ha sido activada. En este caso, llamada se hace al notificador de grupo, pero la llamada es idéntica cuando elegimos un destinatario de forma individual. Pueden agregarse más elementos[15] a la notificación, como imágenes o una entidad.

```yaml
- id: "Zone_1_ON_Notification"
  alias: "Zone 1 Active Notification"
  hide_entity: False
  trigger:
    - platform: state
      entity_id: switch.zona_1
      from: 'off'
      to: 'on'
  action:
    - service: notify.family
      data:
        title: "Riego Zone 1"
        message: "La zona de riego ha sido activada"
```

Ilustración 61: Automatización que notifica a la familia cuando se activa la zona de riego 1

[15] https://www.home-assistant.io/components/notify/

6.7. Integración con Google Assistant (Cloud)

Actualmente existen dos métodos disponibles para conectar las entidades de nuestro sistema con Google Assistant. Para ambos métodos, es necesario que el servidor funcione sobre *SSL*, de lo contrario Home Assistant no podrá comunicarse con los servidores de Google.

- De forma manual, generando un nuevo proyecto en la consola del servicio "Actions on Google". El proceso es bastante largo, pero gratuito. Antes de la existencia de Nabu Casa, la conexión de nuestro servidor con Google Assistant se hacía de esta forma. Todo el proceso de configuración manual está disponible desde la integración de Google Assistant[16] en la web oficial.
- Contratando el servicio de la nube de Home Assistant, Nabu Casa, que además de darnos acceso a Google Assistant, nos proporciona acceso externo si no tenemos un servicio de DNS dinámico en nuestra red. El precio de este servicio es de 5$ mensuales. Con este servicio, podemos evitarnos también la configuración *SSL*, pues el acceso externo de Nabu Casa ya dispone de ello. En nuestro caso si se configura *SSL* y un *DDNS*. Además, siempre tenemos el link de Nabu Casa como acceso de emergencia en caso de caídas del servicio *DDNS* de Asus (lo obtenemos de forma gratuita al adquirir un router con firmware *AsusWRT*).

Podemos exponer la mayoría de entidades domóticas que tenemos a nuestra disposición en el sistema (sensores de temperatura, cámaras, interruptores, luces, dispositivos de aire acondicionado, etc…).

Si utilizamos dispositivos Sonoff para abrir puertas, se recomienda configurarlos también como cerraduras con el componente "MQTT Lock". Así Google Assistant nos permite establecer un código de apertura para estas entidades, e invocarlas diciendo "Abre (nombre de la cerradura)" en vez de "Activa (nombre del interruptor)".

[16] https://www.home-assistant.io/components/google_assistant/#first-time-setup

Los atributos necesarios para exponer entidades en Google Assistant son los que se detallan a continuación.

name: Nombre asignado a la entidad

aliases: Denominaciones alternativas, la primera de ellas es la que aparecerá asignada al dispositivo en la aplicación Google Home.

room: Habitación que se asigna en Google Home. Es opcional, si no lo incluimos, se nos preguntará al detectar el nuevo dispositivo

```
switch.zona_1:
    name: Riego 1
    aliases:
      - Riego Zona 1
      - Aspersores 1
      - Zona 1
    room: Jardín
```

Ilustración 62: Entidad tipo interruptor expuesta en Google Assistant

Se recomienda incluir los tres atributos para cada entidad, de esta forma si invocamos a Google Assistant solicitando apagar “Luces del sótano”, se apagarán todos los dispositivos de tipo luz que estén asignados al sótano.

Cuando tengamos expuestas todas las entidades que deseemos, debemos activar Google Assistant en la nube de Home Assistant. Si se añaden nuevas entidades tras la configuración inicial, debemos volver a sincronizarlas con Google. Ambas cosas pueden hacerse desde el apartado de configuración de Google Assistant que podemos encontrar en “Configuración – Home Assistant Cloud”.

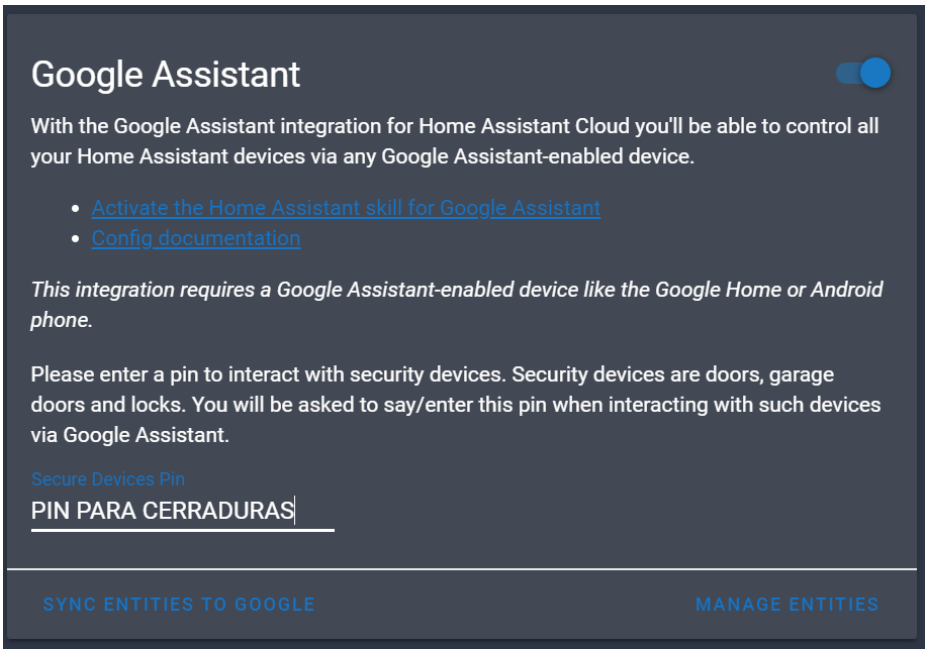


Ilustración 63: Configuración de Google Assistant

6.8. HACS (Home Assistant Community Store)

Es un componente personalizado que permite integrar un gestor de componentes y tarjetas no oficiales dentro de la propia interfaz de Home Assistant. Su instalación es opcional pero muy recomendable, ya que nos avisa si existen actualizaciones para todo lo que tengamos instalado de forma adicional. Podemos encontrar las instrucciones de instalación aquí:

https://hacs.netlify.com/installation/manual/

Aunque el componente es muy reciente, lleva semanas usándose para instalar tanto componentes, como tarjetas en este proyecto. Permite gestionar los paquetes instalados e incluso buscar e instalar nuevos.

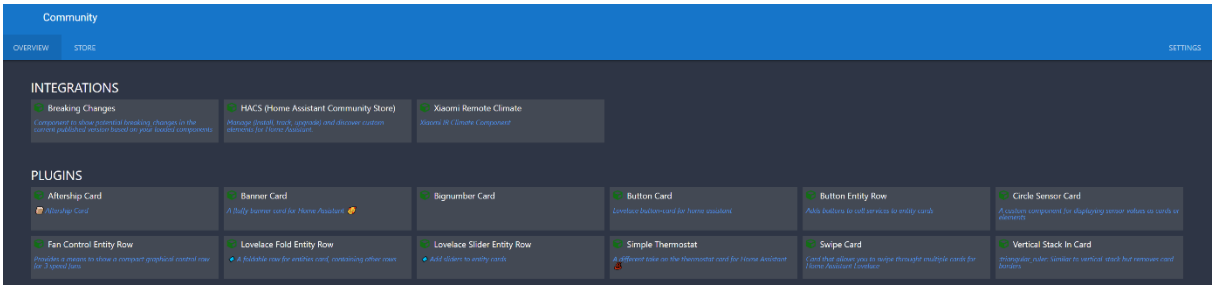


Ilustración 65: Vista en HACS para administrar componentes y tarjetas instaladas

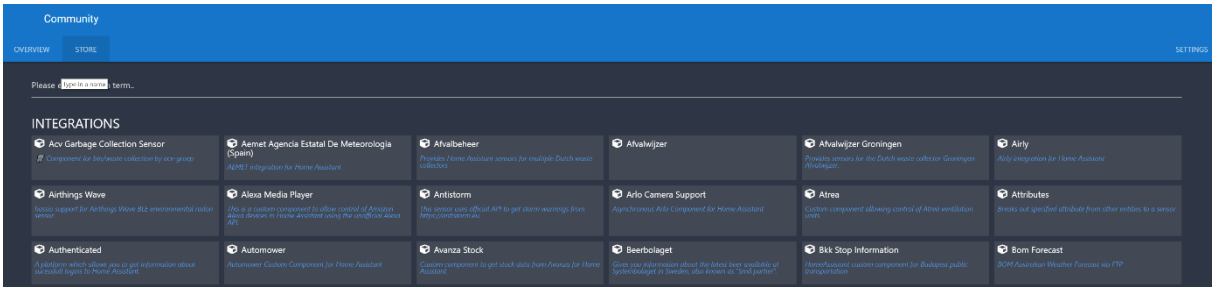


Ilustración 64: Vista en HACS de la tienda de integraciones y tarjetas disponibles para instalar

6.9. TasmoAdmin

Anteriormente conocido como SonWEB, es un servidor que funciona sobre PHP y sirve para administrar dispositivos flasheados con el firmware Tasmota. Está disponible en Windows, Linux o Docker, y aparece como add-on en Hass.io. Si decidimos instalarlo, encontraremos las instrucciones en su GitHub.

https://github.com/reloxx13/TasmoAdmin

Se recomienda su instalación en un equipo diferente de aquel donde se encuentre instalado el sistema. Es totalmente opcional y actualmente no todas las opciones de Tasmota son configurables mediante TasmoAdmin, sin embargo, tiene dos características realmente interesantes cuando tenemos muchos dispositivos Tasmota en nuestro sistema de domótica.

- Panel para visualizar el estado de todos los dispositivos Tasmota que tengamos conectados a la red. Es útil principalmente para saber la versión del firmware en la que se encuentra cada placa.

Ilustración 66: TasmoAdmin vista de lista detallada

- Panel para activar o desactivar interruptores en caso de que un fallo grave del sistema nos impida el acceso a la interfaz de Home Assistant. No necesita un servidor MQTT ya que las peticiones se hacen directamente a las placas con Tasmota vía HTTP.

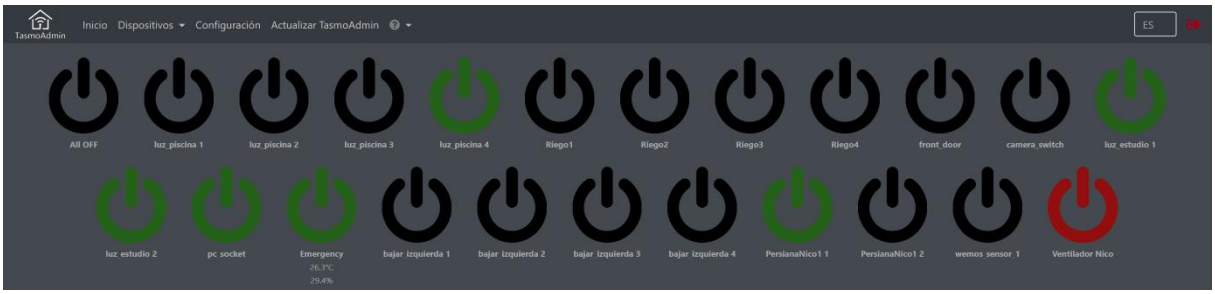


Ilustración 67: TasmoAdmin vista general

7. Seguridad y Red

7.1. Mapa de red local

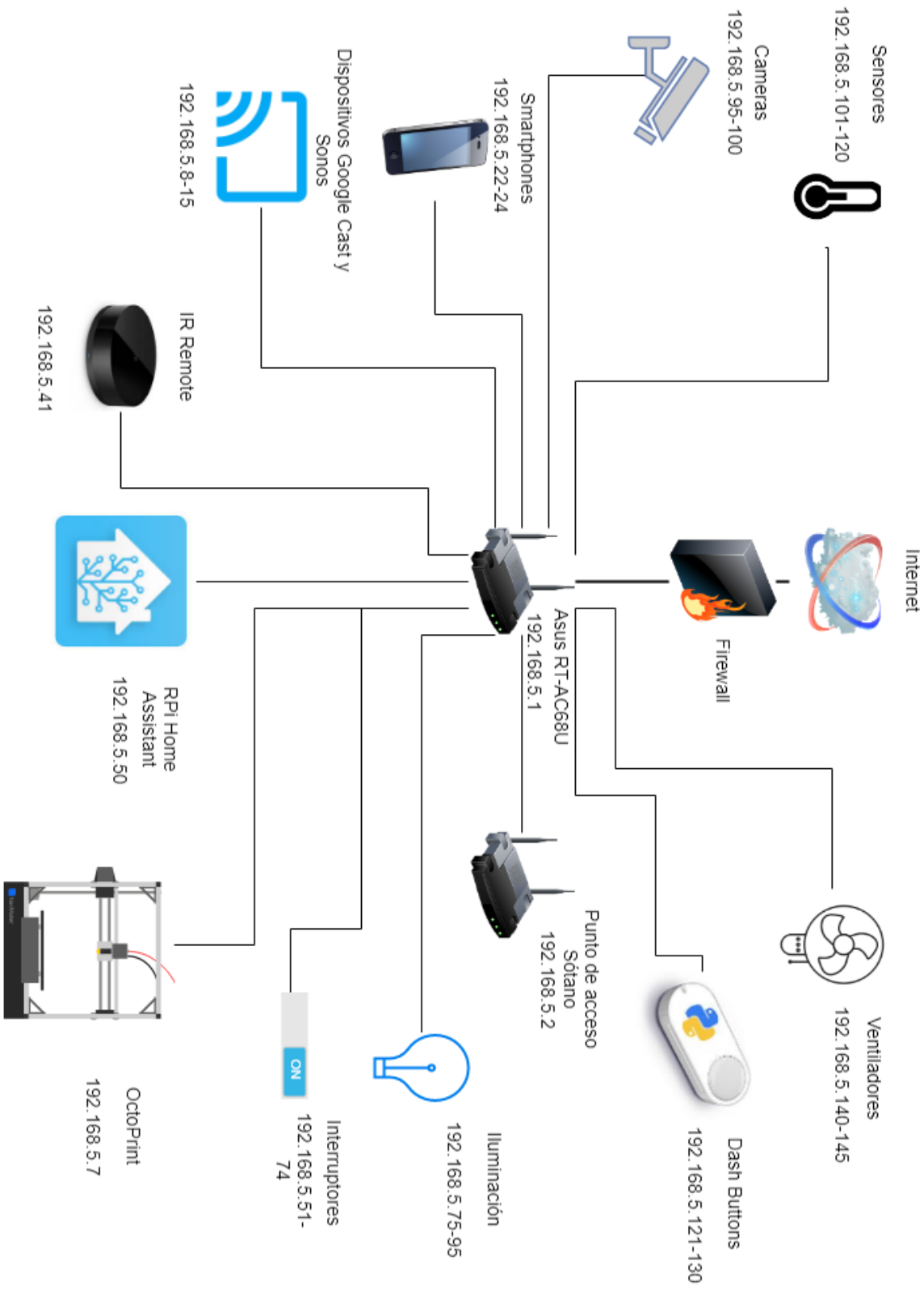


Ilustración 68: Ilustración del mapa de red del sistema

7.2. Red Mallada

Probablemente el mayor problema inherente a las conexiones inalámbricas son los cortes de conexión debido a las interferencias, y a los cambios de punto de acceso por distancia.

Por defecto si hay varios puntos de acceso disponibles, los dispositivos permanecen conectados al punto de acceso en el que se encuentran mientras tengan alcance, aunque haya otro más cercano a ellos. Esto produce conexiones de mala calidad y nos obliga a asignar diferentes *SSID* a los puntos de acceso de la red, para al menos, poder seleccionar el que más nos convenga de forma manual.

Para solucionar estos problemas, se idearon las redes malladas (Mesh Network). Es una topología de red en la que todos los nodos en alcance están interconectados entre sí. Si el espacio a cubrir es demasiado grande, la interconexión de nodos permite el siguiente escenario:

- Existe un nodo B que tiene alcance a un nodo A y uno C
- El nodo A tiene acceso a internet vía Ethernet
- C no está al alcance de A
- B hará de puente entre A y C para que el nodo C tenga conexión.

Pero no es solo ésta la ventaja de las redes malladas. Los clientes de la red pueden cambiar el nodo de conexión sin que existan cortes en la transferencia de datos, e incluso mantener una conexión con más de un nodo de forma simultánea para aumentar la velocidad de su enlace. También pueden compartir clientes entre nodos a modo de balanceador de carga. Hace unos años, las redes malladas quedaban relegadas al terreno empresarial, sin embargo, aunque son caras aún, la inversión merece la pena.

Ilustración 69: Escenarios habituales en redes malladas y dispositivo Lyra Mini compatible con AiMesh de Asus

7.3. Configuración SSL

Al adquirir un router de gama alta de Asus, la empresa nos ofrece de forma gratuita un servicio de DNS dinámico sobre el que generar los certificados. Además, la configuración de estos mediante "Let's Encrypt" se vuelve trivial gracias al firmware del router, que lo hace todo de forma automática. Al comienzo de este TFG, era necesario regenerar cada tres meses los certificados para mantener el *SSL* activado. Let's Encrypt, que es un servicio gratuito, limita también a solo uno los certificados válidos simultáneos que podemos crear para un dominio. Como hay más servidores disponibles instalados en el hogar que usan *SSL*, se diseñaron algunos scripts en bash que copiaban los certificados desde la RPi donde reside Home Assistant, al resto de servidores mediante SCP[17].

Actualmente, al ser el router quien genera los certificados, no es necesario todo ese proceso. Let's Encrypt está perfectamente integrado en *AsusWRT*. Lo único necesario para hacer funcionar Home Assistant sobre *SSL*, es incluir en el componente HTTP, el atributo "base_url" y poner nuestro nombre de dominio.

En principio, si se sigue este proceso, aún es necesario copiar los certificados que genera el router a Home Assistant y al resto de servidores que tengamos detrás del router. Pero en esta implementación esto no es necesario, y se explicará la razón en el siguiente apartado que habla sobre la funcionalidad de un proxy inverso.

7.4. Proxy inverso (NGINX)

El concepto de un servidor proxy inverso es fácil de entender. Es un tipo de proxy que genera peticiones contra servidores solicitadas por clientes. Los servidores devuelven la petición al servidor proxy, y este a su vez la devuelve al cliente como si fuera el servidor proxy el que generase las peticiones para él.

Tiene múltiples usos a nivel profesional. Permiten ocultar la visibilidad y el funcionamiento de servidores internos a la red, comprimir las peticiones para ahorrar datos, balancear la carga en servidores replicados, cachear parte del contenido para reducir la carga de estos servidores, etc… Es un concepto muy versátil, y nosotros vamos a utilizarlo principalmente para dos cosas.

[17] https://www.techopedia.com/definition/26142/secure-copy

- Como nuestro router genera los certificados digitales de forma automática, si instalamos un proxy inverso en el router, todos los servicios que expongamos a través de él tendrán *SSL* activado, ya que es el router quien posee los certificados.
- Evitamos la apertura de puertos. Ahora accederemos a todos los servidores detrás del proxy a través del puerto 443. Esta técnica aumenta mucho la seguridad de la red al completo porque toda la información que transmitimos irá cifrada hasta el router, independientemente de la seguridad implementada en los servidores de forma interna.

Es posible instalar un proxy inverso en un router Asus siguiendo los siguientes tutoriales en orden. El proceso dura aproximadamente 1 hora, pero es recomendable invertir este tiempo ya que nos ahorrará mucho posteriormente.

1. Cambiar el firmware oficial por “Merlin Firmware”

   https://github.com/RMerl/asuswrt-merlin/wiki/Installation

2. Activar *SSH* y la partición JFFS

   https://github.com/pedrom34/TutoAsus#2-activate-ssh-et-jffs-partition

3. Activa Entware en el router

   https://github.com/RMerl/asuswrt-merlin/wiki/Entware

4. Instala y configura NGINX (Pasos 5 y 6)

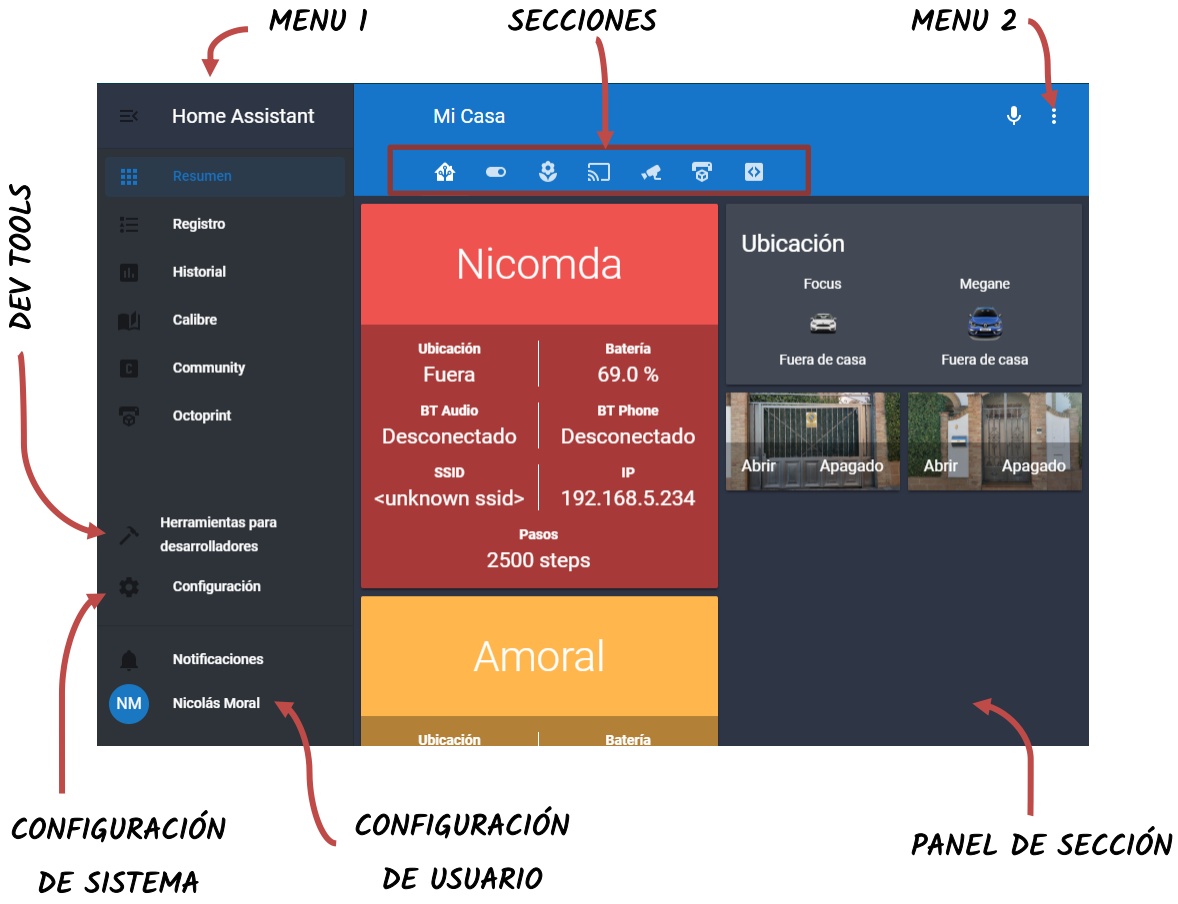
   https://github.com/pedrom34/TutoAsus#5-install-nginx

5. Modifica el archivo de configuración de NGINX para añadir soporte para Home Assistant al proxy inverso

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

8. Manual de usuario

Cuando accedemos al servidor a través de un navegador, se nos solicitan nuestras credenciales de entrada. Una vez dentro, vemos la página principal de Home Assistant.



8.1. Menú 1

En este apartado tenemos acceso a utilidades de Home Assistant. El resumen contiene todas las secciones de interfaz de control domótico y es donde pasaremos el mayor tiempo de uso. También existe un registro de eventos y un historial.

- Registro: muestra los eventos que se producen en el sistema en forma de línea temporal
- Historial: refleja de forma gráfica el cambio de estados de las entidades del sistema

- Mapa: Aunque está desactivado en el proyecto, permite ver sobre un mapa generado por la *API* de Google Maps, la posición de los device_tracker
- Community: Es el acceso directo a la interfaz de HACS

El resto de elementos que aparecen en el menú, son links hacia servidores dentro de la casa, para facilitar el acceso. No forman parte de Home Assistant.

8.2. Menú 2

Tiene dos utilidades muy importantes. La primera es la opción de poner Lovelace en modo edición. Desde él podemos crear nuevas tarjetas en la interfaz e incluso modificar las ya existentes con un editor desde el propio navegador. Una vez estemos en modo edición, si volvemos a acceder al menú, veremos un editor RAW de la configuración; es decir, directamente el archivo completo que contiene la estructura de la interfaz.

La segunda utilidad es una opción que permite mostrar todas las entidades que no se han añadido en ninguna de las secciones de la interfaz. Así podemos ver el aspecto de las mismas cuando creamos nuevas o cuando queremos acceder a alguna que no es de uso habitual.

8.3. Secciones

- General

En esta sección hay información general sobre los dispositivos de los usuarios del sistema, la ubicación de los vehículos e interruptores de acceso rápido para activar las puertas de la casa.

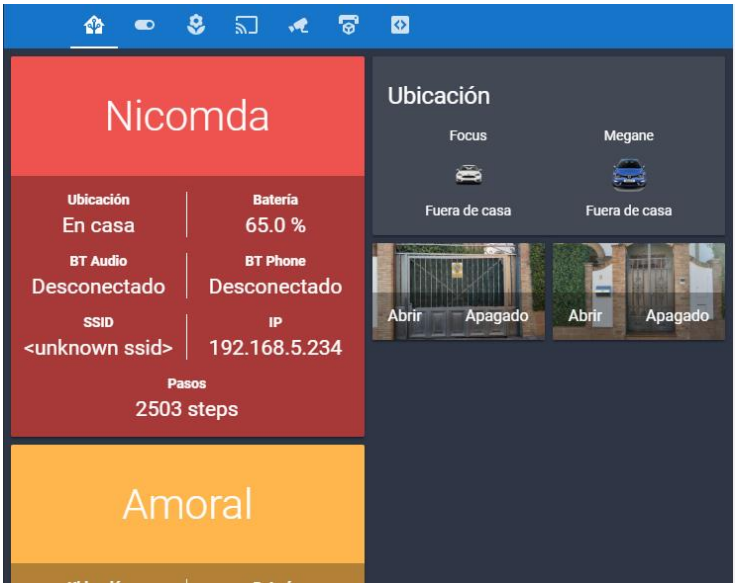


Ilustración 70: Sección General en Lovelace

- Activable

La parte de interfaz relativa a interruptores, iluminación, climatización y ventilación de las habitaciones del hogar se encuentra en esta sección. Aunque actualmente no están todas las habitaciones de la casa, sí que aparecen todas las que tienen algún tipo de domotización.

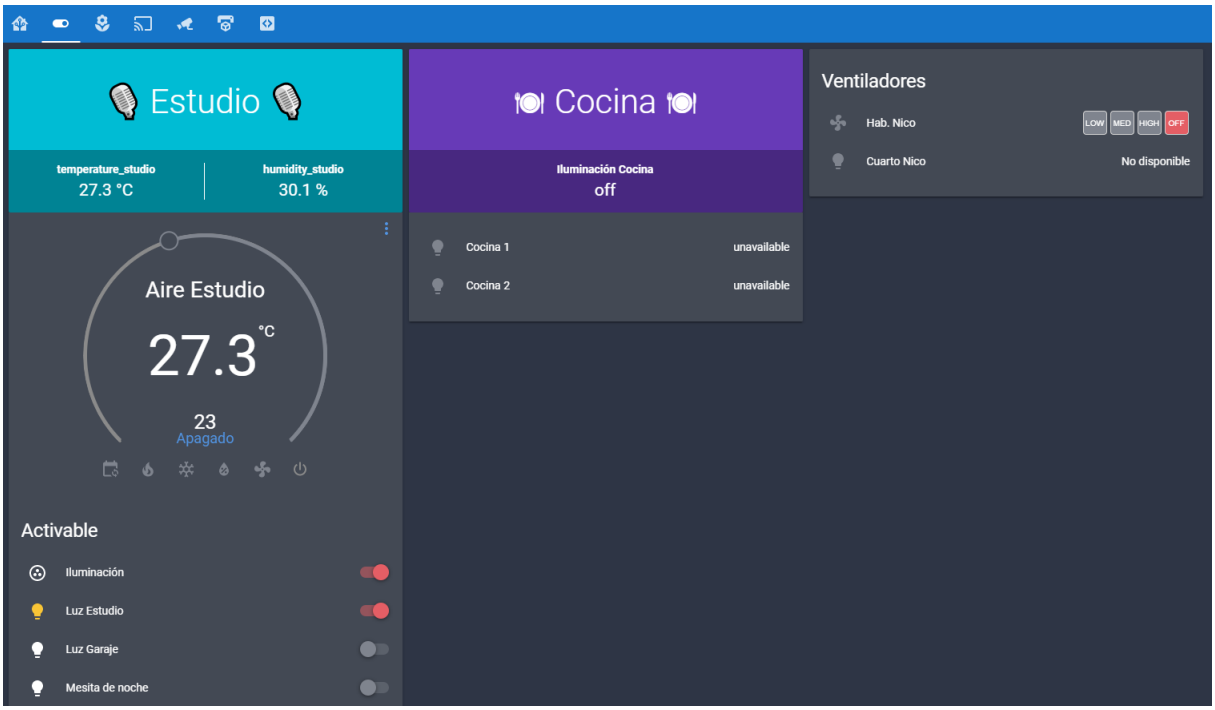


Ilustración 71: Sección Activable en Lovelace

- Jardín

Podemos ver tres partes bien diferenciadas en la sección de Jardín. La parte izquierda, posee la configuración de la piscina y la activación manual de las zonas de riego. La parte central contiene la programación de riego, y la última información relativa al estado de esta programación.

Para programar la depuradora solo es necesario pulsar en el botón del sol o el copo de nieve, dependiendo del modo de depuración que deseemos. El sol establece una programación adecuada para el uso habitual de la piscina, y el copo sólo para el mantenimiento en buen estado del agua.

Para programar el riego debemos seleccionar la hora de inicio y si queremos que se active diariamente o de forma alterna. Cuando esas opciones estén seleccionadas,

asignamos en los “sliders” inferiores el tiempo de riego que deseamos para cada zona. Finalmente, podemos activar el primer interruptor de la sección que habilita la programación, o pulsar el botón siguiente que inicia un ciclo completo en el instante que lo activemos.

Si necesita saberse qué zona se encuentra regando, tiempo restante, cuanto tiempo se ha regado u otro tipo de información adicional, podemos verla en la última parte de la sección.

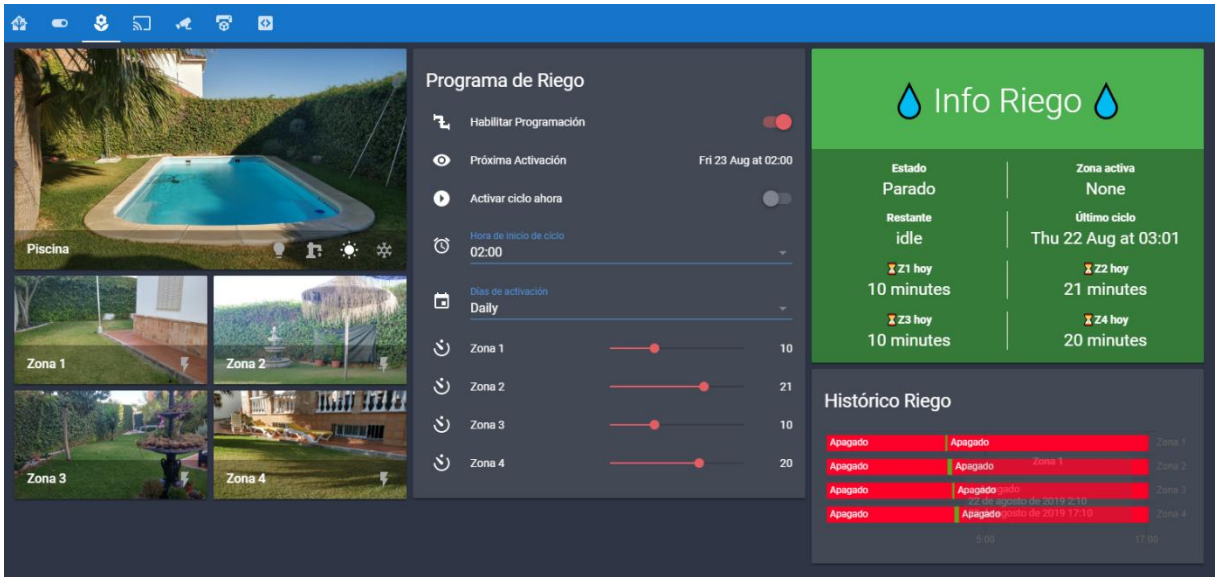


Ilustración 72: Sección riego en Lovelace

- Cast

Respecto a la sección de dispositivos multimedia, el funcionamiento es simple, y posee las mismas opciones que cualquier reproductor habitual. Solamente es necesario indicar, que si pulsamos en una entidad multimedia, aparece un campo de texto para producir audio *TTS* a través de ese dispositivo.

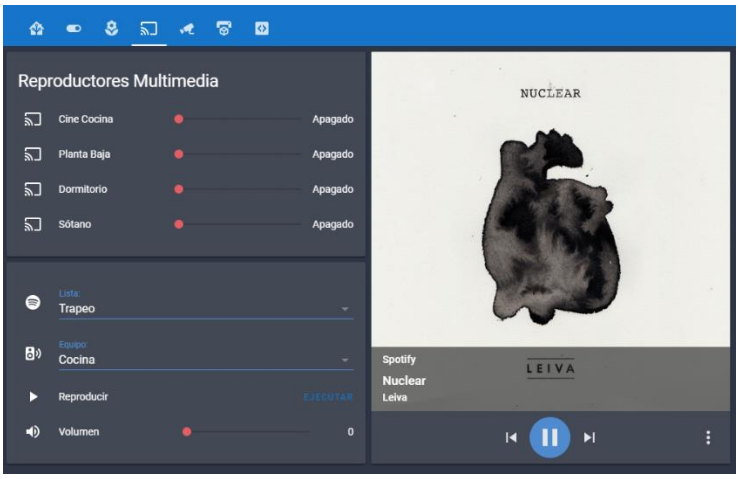


Ilustración 73: Sección Cast en Lovelace

- Monitorización

Como su propio nombre indica, esta sección está totalmente dedicada a la monitorización del sistema. La única entidad activable es la cámara interna.

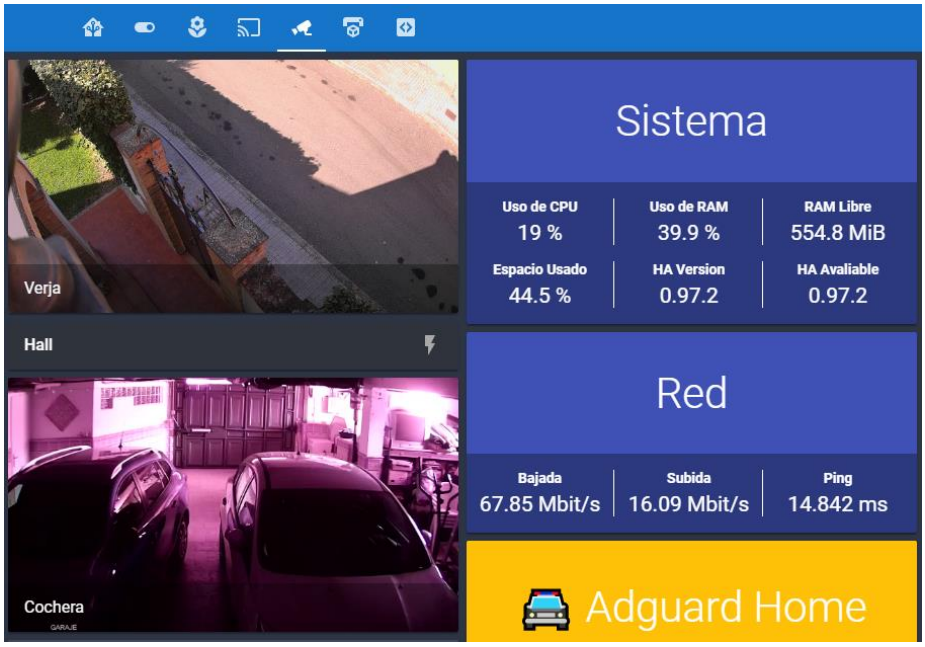


Ilustración 74: Sección de Monitorización en Lovelace

- Impresora 3D

Dispone una interfaz para controlar y mostrar información de la impresora 3D mediante la *API* de OctoPrint. Puede verse incluso la cámara integrada en la Anet A8.

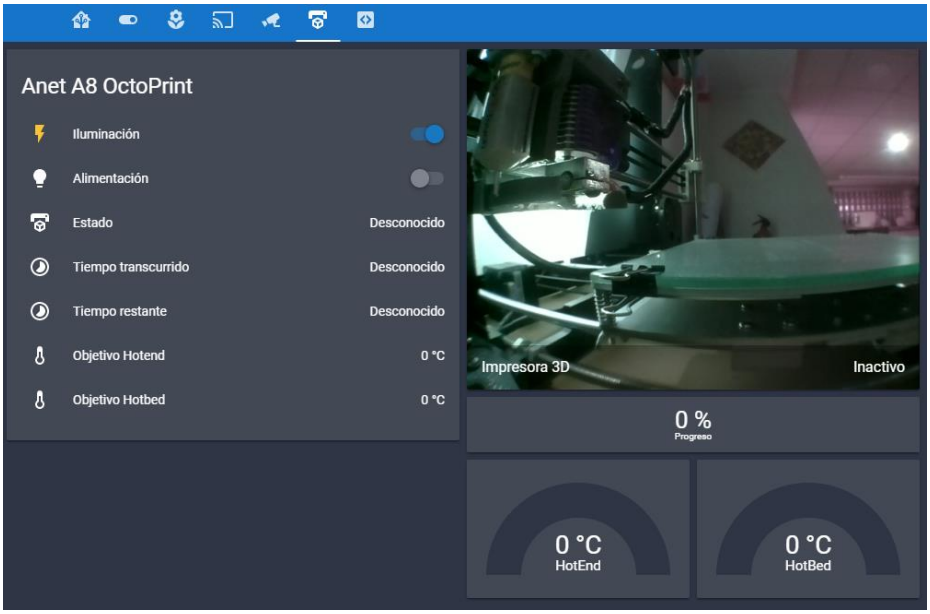


Ilustración 75: Sección de impresión 3D en Lovelace

- Desarrollo

Aunque no sea una sección para los usuarios habituales, se ha usado durante todo el proyecto para introducir de forma previa a la interfaz de uso, todas las nuevas implementaciones en el sistema. Además, contiene un listado con algunos de los scripts y automatizaciones desarrollados, cuyo uso es conveniente poder desactivar. Existe también una sección para limpiar los topic de MQTT de posibles mensajes retenidos.

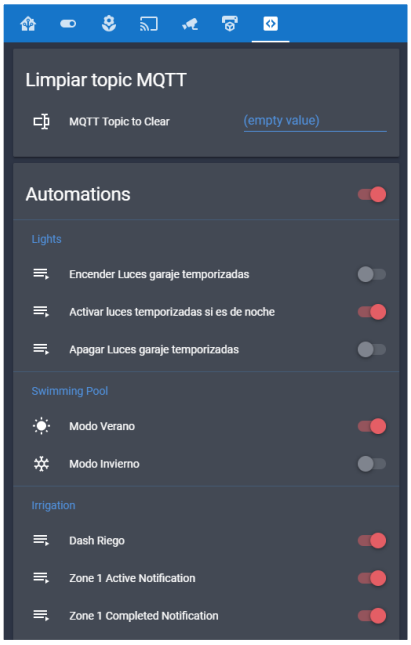


Ilustración 76: Sección de Desarrollo en Lovelace

8.4. Dev Tools

En el apartado para desarrolladores, si estamos implementando el sistema, hay muchas herramientas interesantes que nos permiten operar desde la interfaz gráfica.

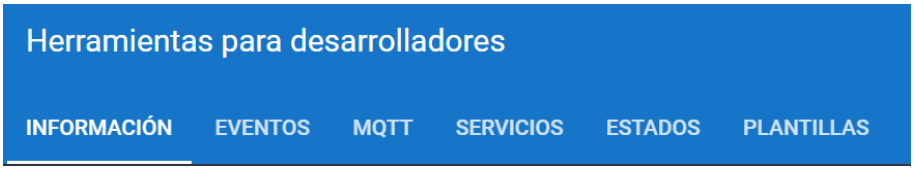


Ilustración 77: Herramientas para desarrolladores

La primera sección es sobre información del sistema. La segunda sirve para simular el envío de eventos al bus y ver cómo se comportan nuestras automatizaciones. La tercera, para enviar información mediante MQTT a cualquier topic. La cuarta permite hacer llamadas a los servicios registrados en Home Assistant. La quinta es un registro de estados detallado para ver las entidades disponibles en el sistema y todos sus atributos, y la última es un editor de templates en Jinja2 para poder testearlos antes de agregarlos a código *YAML*. Hay mucha más información relativa a estas herramientas de desarrolladores[18] en la sección dedicada a ellas en la web oficial.

[18] https://www.home-assistant.io/docs/tools/dev-tools/

9. Conclusiones

9.1. Conclusión final

Durante la realización de este trabajo he obtenido muchos conocimientos relacionados con el funcionamiento intrínseco de la domótica actual. La potencia que tenemos al disponer de microcontroladores con conexión a la red, prácticamente no tiene límites.

He adquirido también bastantes conocimientos de electrónica y electricidad integrando las placas en los sistemas ya existentes en la vivienda. Tener una casa domotizada era un sueño por cumplir desde que comencé a conocer el mundo de la informática, y he conseguido implantarlo de manera funcional.

El valor de los datos personales de los usuarios, hace que un sistema con procesamiento local, sea la opción más segura para protegernos.

Además, durante el proceso de despliegue, he aprendido conceptos de red que no conocía, como el de proxy inverso, que creo que me resultarán útiles en el futuro.

Personalmente pienso que el precio total del proyecto para implementarlo en un hogar es realmente bajo, pues no solo hay que considerar el gasto de la instalación, sino también la estimación del ahorro en términos de consumo de energía.

La domótica no es el futuro, sino el presente. La llegada de los asistentes personales, no solo a los Smartphones sino también a los altavoces de las casas, televisiones, coches y demás dispositivos electrónicos, ha hecho que todas las marcas que fabrican productos domóticos se encaminen en la misma dirección, acelerando el desarrollo general de todo lo relacionado con domótica. Además, gracias a chips como ESP-8266, ahora es posible comercializar productos domóticos a un precio que prácticamente cualquiera puede permitirse. Esto, y la facilidad para la configuración de estos dispositivos provocada por la carrera en el mercado de los fabricantes, han hecho explotar a nivel publicitario la domótica para llegar al gran público.

9.2. Trabajo futuro

Con el proyecto de investigación concluido, y obviando el hecho de que añadir nuevos dispositivos al sistema como los ya existentes es un proceso sencillo, no querría terminar sin detallar los enfoques en los que considero interesante la posibilidad de abrir nuevas líneas de investigación.

La primera de ellas, tiene que ver con automatizaciones y scripts. Integrar la base de datos de Home Assistant en un sistema de procesamiento, que es actualmente posible a través de integraciones de complementos, probablemente nos permita ver mediante técnicas de Big Data qué comportamientos de los usuarios del sistema pueden ser automatizados o asistidos.

La segunda, es con respecto a las notificaciones. Aunque los dispositivos multimedia sean capaces de convertir texto a audio mediante *TTS*, actualmente todas estas notificaciones se reciben únicamente a través de los dispositivos móviles de los usuarios con la aplicación Ariela. Sin embargo, añadiendo algunos sensores de presencia en interiores y más Google Home, sería posible implementar notificaciones en función de la ubicación y el estado del usuario.

Otra de las vías de investigación que considero interesantes es el intercambio de información con SmartBands. Si consiguiésemos esta comunicación entre alguna pulsera de actividad y nuestro sistema, no solo podríamos mostrar datos al usuario, sino utilizarlos en scripts y automatizaciones para hacerlo más inteligente.

También podría investigarse en el terreno relacionado con sensores para integrarlos en Tasmota. Actualmente casi todos los que hay, parecen útiles a nivel doméstico. Sin embargo, el sensor de flujo utilizado en el proyecto no estaba disponible desde el firmware. La implementación de este y otros sensores no tan habituales puede resultar interesante para el terreno empresarial.

Por último, sería interesante desarrollar algún complemento propio, o sobre las plataformas AppDaemon[19] o Node-RED[20] que están disponibles en Home Assistant.

[19] https://appdaemon.readthedocs.io/en/stable/
[20] https://nodered.org

10. Tabla de Ilustraciones

11. Referencias

Información 5G

https://www.5gfixedwireless.org/2019/03/what-is-5g-fixed-wireless-fwa/

Información MQTT

https://ricveal.com/blog/primeros-pasos-mqtt/

ESP-8266

https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/

Zigbee Info

https://www.alexaenespanol.com/que-es-zigbee-y-como-funciona/

Z-Wave info

https://www.xataka.com/seleccion/zigbee-z-wave-que-que-se-diferencian-que-marcas-domotica-compatibles

MQTT Mosquitto

https://clouding.io/kb/introduccion-a-eclipse-mosquitto/

Wemos D1 Mini Info y drivers

https://wiki.wemos.cc/products:d1:d1_mini

UART y USB

https://aprendiendoarduino.wordpress.com/2016/11/09/uart-y-usb-en-arduino/

Platform IO IDE

https://platformio.org/platformio-ide

Arquitectura de Home Assistant

https://developers.home-assistant.io/docs/en/architecture_index.html

Sensor de ultra sonidos HC-SR04

https://naylampmechatronics.com/blog/10_Tutorial-de-Arduino-y-sensor-ultras%C3%B3nico-HC-S.html

Sensor de caudal YF-S201

http://www.theorycircuit.com/water-flow-sensor-yf-s201-arduino-interface/

Sensor de temperatura D18B20

https://www.luisllamas.es/temperatura-liquidos-arduino-ds18b20/

Sensor de temperatura y humedad DHT22

https://naylampmechatronics.com/blog/40_Tutorial-sensor-de-temperatura-y-humedad-DHT1.html

BLE *Beacon*s Info

https://www.*Beacon*stac.com/what-is-a-bluetooth-*Beacon*

Servidor *Beacon*s BLE con filtro de Kalman

https://blog.truthlabs.com/*Beacon*-tracking-with-node-js-and-raspberry-pi-794afa880318

Servidor de video Shinobi

https://shinobi.video/docs/

Servidor de vídeo Zoneminder en Raspberry Pi3

https://francoconidi.it/zoneminder-su-raspberry-pi-3-b-raspbian-stretch/

Servidor para gestión de Dash Buttons

https://github.com/Nekmo/amazon-dash

La biblia de Home Assisant

https://www.awesome-ha.com/

Tareas habituales en Hassbian

https://www.home-assistant.io/docs/installation/hassbian/common-tasks/

Tutorial de *YAML*

https://www.tutorialspoint.com/yaml/index.htm

Tutorial de Jinja2

https://pythonista.io/cursos/py201/introduccion-a-jinja-2

Firmware Tasmota

https://github.com/arendst/Sonoff-Tasmota

Web de Owntracks

https://owntracks.org/booklet/

Web de Ariela para Android

http://ariela.surodev.com/

Redes malladas

https://www.xataka.com/especiales/redes-wifi-mesh-que-son-como-funcionan-y-por-que-pueden-mejorar-tu-red-wifi-en-casa

Proxy Inverso

https://picodotdev.github.io/blog-bitix/2016/07/como-crear-un-proxy-inverso-entre-el-servidor-web-nginx-y-un-servidor-de-aplicaciones-java/

12. Glosario

Addons: Es una mejora instalable en un sistema. Son programas que funcionan sobre otro para ampliar funcionalidades

API: Es un conjunto de funciones que permite acceso a un software concreto desde el exterior del mismo

AsusWRT: Firmware de router específico de la marca Asus

Beacon: Dispositivo de bajo consumo que emite una señal única, generalmente mediante bluetooth

BIOS: Software básico que reconoce los dispositivos necesarios para cargar el arranque del sistema en RAM

BLE: Especificación del protocolo bluetooth de baja energía

Conector Schuko: Nombre coloquial para denominar un sistema de toma de corriente, generalmente a 220V

Data Payload: Parte de la trama de un paquete de red que comprende el conjunto de datos enviado

DDNS: Servicio de DNS, que permite enlazar una IP dinámica a un dominio estático que se actualiza de forma automática

Dupont: Es un cable que sirve para interconectar componentes con un conector en cada extremo que puede ser macho o hembra

Echo: Mensaje que se envía a través de un protocolo con la intención de que regrese al dispositivo emisor

Endpoint: Punto de entrada a un servicio,proceso o cola en arquitecturas orientadas a servidores

FFmpeg: Librería de software libre que permite grabación, transcodificación y streaming de vídeo y audio

Flag: Bit que almacena un valor binario cuyo significado se ha asignado previamente

Framework: Es un esquema o estructura de software que sirve para desarrollar una aplicación de manera más sencilla

GPIO: Pines de entrada/salida de propósito general en un chip

GPU: Unidad de procesamiento gráfico

Hardening: Es una serie de procesos que se llevan acabo en un sistema para reforzar la seguridad del mismo

Header: Punto de conexión en una placa electrónica

HVAC: Calefacción, ventilación y aire acondicionado

ICMP: Protocolo de control de mensajes de Internet

JSON: Lenguaje de serialización de datos más utilizado en aplicaciones web

Log: Registro de eventos y acciones de un software o sistema informático

Material Design: Es una normativa de diseño creada para mejorar la interfaz de visualización en Android, o cualquier otra plataforma

Notificaciones Push: Notificaciones instantáneas en tiempo real en un dispositivo

PlatformIO: Ecosistema para programación IoT de código abierto

Raíl DIN: Barra de metal normalizada para montaje de elementos eléctricos

RF: Radio Frecuencia

Sandbox: Entorno de ejecución aislado para software

SSH: Protocolo que permite la conexión cifrada con la consola de otro equipo

SSID: Nombre de un punto de acceso Wi-Fi a la red

SSL: Es un protocolo de seguridad que permite establecer un canal totalmente cifrado entre dos máquinas a través de la red

TTS: Software que permite convertir texto a audio

UEFI: Interfaz extensible de firmware que releva a la antigua *BIOS*. Añade soporte para gráficos avanzado y acceso remoto

URI: Identificador de recursos en red. Un enlace web, o URL es un tipo de *URI*

YAML: Lenguaje diseñado para serializar datos de forma legible por personas